1998

AEGIS Data Analysis and Reduction (ADAR) in Support of the AEGIS Weapon System (AWS)

June Bullard Gaines

College of Humanities and Sciences
Virginia Commonwealth University

This is to certify that the thesis prepared by June B. Gaines entitled "AEGIS Data Analysis and Reduction (ADAR) in Support of the AEGIS Weapon System (AWS)" has been approved by his committee as satisfactory completion of the thesis requirement for the degree of Master of Science in Computer Science.

[redacted]

Dr. James E. Ames, IV
Professor, Department of Mathematical Sciences
Director of Thesis

[redacted]

Dr. Lorraine M. Parker
Associate Professor, Department of Mathematical Sciences
Committee Member

[redacted]

Dr. Ena Gross
Associate Professor, Teacher Education Division
Outside Committee Member

[redacted]

Dr. James A. Wood
Director of Graduate Studies
Department of Mathematical Sciences

[redacted]

Dr. J. Richard Morris
Chair, Department of Mathematical Sciences

[redacted]

Dr. Stephen D. Gottfredson
Dean, College of Humanities and Sciences

[redacted]

Dr. Jack L. Haar
Dean of Graduate Studies

10/9/98
Date

AEGIS Data Analysis and Reduction (ADAR) in Support of the AEGIS Weapon System (AWS)

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

By

June Bullard Gaines
B.S. Mars Hill College, Mars Hill, NC 1965-1969
Mathematical Sciences Certificate, Virginia Commonwealth University, Richmond, VA 1986-1989

Director: James E. Ames, IV, Associate Professor
Department of Mathematical Sciences

Virginia Commonwealth University
Richmond, Virginia
August, 1998

Acknowledgment

I would like to thank my husband Jack, my son David, and my daughter Angela for their support in this endeavor over the years.

Special thanks go to Tom Poley who has always encouraged me during this process even when I was discouraged. You have my grateful thanks for all of your help, guidance, comments, and advice.

Thanks go to Kay Pollack for providing advice regarding the origins of the ADAR system as well as providing needed editorial and content review.

Likewise, thanks go to Steven Canup for providing advice regarding the current state of ADAR as well as future plans. Steven has been a great help over the years, especially regarding changes in ADAR while I was working with it.

Additionally, I want to thank Dr. Ames for all of his support and help.

Table of Contents

List of Figures

List of Appendices

List of Abbreviations

AAW	AntiAir Warfare
ACC	AEGIS Computer Center
ACS	AEGIS Combat System
ACSC	AEGIS Combat System Center, Wallops Island, Virginia
ACSS	AEGIS Classified Support System
ACTS	AEGIS Combat Training System
ADAR	AEGIS Data Analysis and Reduction
ADS	AEGIS Display System
API	Application Program Interfaces
ASCII	American Standard Code for Information Interchange
ASP	ADAR Sequential Processor
ASUW	AntiSurface Warfare
ASW	AntiSubmarine Warfare
ATEP	AEGIS Tactical Executive Program (later ATES)
ATES	AEGIS Tactical Executive System
AWS	AEGIS Weapon System
B/L	BaseLine

BPI	Bytes per inch
C&D	Command and Decision system (also CND)
CC	Command Control (later became Command and Decision)
CG	Gas turbine-powered Guided Cruiser
CGN	Nuclear-powered Cruiser
CGTN	Control Group Track Number
CS-1	Compiling System-1
CND	Command and Decision system (also C&D)
CNO	Chief of Naval Operations
COTS	Commercial-off-the-Shelf
CSED	Combat System Engineering Development
CSEDS	CSED Site
CSG	Strike Cruiser
CSGN	Nuclear-powered Strike Cruiser
CSSQT	Combat Systems Ship Qualification Trials
DC	Dictionary Compare program
DCL	Digital Command Language
DD	Data Dictionary
DD	Destroyer
DD	Display Dictionary program
DDFM	Double Density Film Memory

DDG	Gas turbine-powered Guided Missile Destroyer
DEC	Digital Equipment Corporation
DG	Gas turbine-powered Destroyer
DLGN	Nuclear-powered Destroyer
DOD	Department of Defense
DR	Data Recording
DRIST	Data Recording Item Selection Table
DSARC	Defense System Acquisition Review Council
DXEP	Data Extraction Point Analyzer program
DXMOV	Data eXtraction MOVe program
DXR	Data eXtraction and Recording
DXSUM	Data eXtraction SUMmary program
EDM	Engineering Development Model
EISA	Extended Industry Standard Architecture
EP	Extraction Point
EPAC	Extraction Point Attribute Collection program
EPID	Extraction Point IDentifier
FCDSSA	Fleet Combat Direction Systems Support Activity, Dam Neck, Virginia
FCS	Fire Control System
FORTRAN	Formula Translation programming language
GenFLD	Generate FieLDs program

GenMEP	Generate Multiple Extraction Points program
GenPOC	Generate Print-On-Change program
GMLS	Guided Missile Launching System
GOTS	Government-off-the-Shelf
GRD	Get Recorded Data program
GUI	Graphical User Interface
HOLWG	High Order Language Working Group
HP	Hewlett-Packard
HPUX	Hewlett-Packard computers running the UNIX operating system
IDS	Interface Design Specification
LAMPS	Light Airborne Multi-Purpose System
LBTS	Land Based Test Site
LSE	Lifetime Support Engineering
LSEA	Lifetime Support Engineering Agent
Mk	Mark
MIT	Massachusetts Institute of Technology
MW	MegaWords
NFS	Network File System
NSWC	Naval Surface Weapons Center (1975-1987), Naval Surface Warfare Center (1987-1994), former names for NSWCDD

NSWCDD	Naval Surface Warfare Center Dahlgren Division, Dahlgren, Virginia (since 1994), previously NSWC, NWL, etc.
NSWCPHD	Naval Surface Warfare Center Port Hueneme Division, Port Hueneme, California
NWL	Naval Weapons Laboratory (1959-1975), a former name for NSWCDD
OPREDEX	Operational Readiness Exercise
ORTS	Operational Readiness Test System
PC	Personal Computer
PGC	Program Generation Center
POC	Print-On-Change
PTC	Production Test Center
RCA	Radio Corporation of America
RAM	Random Access Memory
RDT	Recorded Data Tapes
RISC	Reduced Instruction Set Computer
SECDEF	Secretary of Defense
SECNAV	Secretary of the Navy
SGI	Silicon Graphics Incorporated
SM	STANDARD Missile
SPY	AN/SPY Radar system
STN	System Track Number

TWS	TOMAHAWK Weapon System
U.S.	United States
VAX	Virtual Address eXtension
VLS	Vertical Launching System
WCS	Weapons Control System
WDS	Weapon Direction System (later WCS)

Abstract

AEGIS Data Analysis and Reduction (ADAR) and the AEGIS Weapon System (AWS)

By June Bullard Gaines, B.S.

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 1998.

Major Director: James E. Ames, IV, Associate Professor, Department of Mathematical Sciences

The AEGIS Weapons System (AWS), part of the AEGIS Combat System (ACS), is an integral part of the defense system on U.S. Navy AEGIS-class ships. AEGIS Data Analysis and Reduction (ADAR) has been developed to assist in the evaluation of the AWS data. ADAR, along with the AWS and ACS, has evolved through the years to accommodate advances in technology and computer programming languages. Additionally, ADAR has evolved so that users located at sites other than the Naval Surface Warfare System Dahlgren Division (NSWCDD), Dahlgren, Virginia, can reduce tactical system data and perform data analysis using the reduced data.

This paper is a study of the growing pains experienced by the ADAR system as it has evolved. Some of the changes affecting ADAR have included the addition of new elements to both the AWS and the ACS, the multiplication of AEGIS baselines, and the provision of portability both to a multitude of platforms and to several sites supporting the AEGIS program.

1 Introduction

AEGIS is the United States' most advanced shipboard antiair warfare weapon system. Further, AEGIS is the highly integrated total ship combat system built around the weapon system. It is capable of simultaneous warfare on many fronts: air, surface, subsurface, and strike. As one might gather, the word "AEGIS" in the U.S. Navy has several meanings. At the top there is the AEGIS program under which are the AEGIS facilities including the AEGIS ships as well as anything connected to them. The land-based AEGIS facilities encompass the sites that contribute a variety of services to the AEGIS program such as system engineering, development, manufacturing, and installation. The AEGIS class ships, beginning with the guided cruiser *USS Ticonderoga,* CG-47, and the guided missile destroyer *USS Arleigh Burke*, DDG 51, have been specially designed and built to accommodate the requirements of the AEGIS Combat System (ACS) with the AEGIS Weapon System (AWS).

Figure 1-1, the *USS Normandy*, CG-60, shows is an example of the guided cruiser Ticonderoga ship class. Figure 1-2, the *USS Ramage*, DDG-61, shows an example of the guided missile destroyer Arleigh Burke ship class.

The name "AEGIS" was chosen for this program due to its meaning in both Roman and Greek mythology. The goatskin breastplate worn by Zeus, the supreme

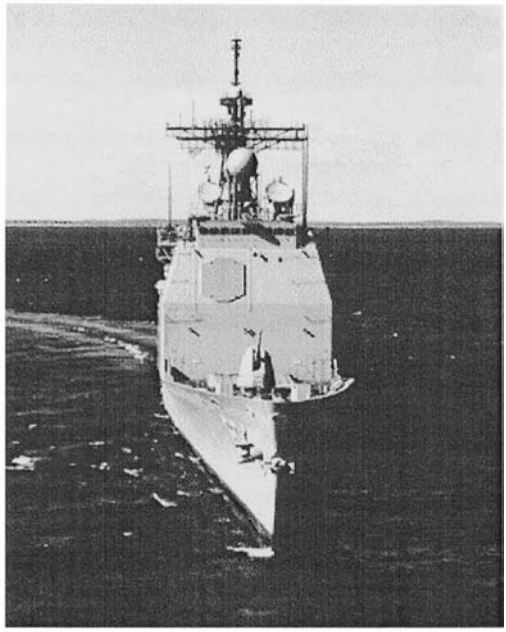

Figure 1-1: The *USS Normandy*, CG-60

Figure 1-2: The *USS Ramage*, DDG-61

Greek god, was called an "aegis" and represented power and majesty. The ancient Greeks saw Zeus as a protector. On the other hand, Athena, the Greek goddess of wisdom, who also had an aegis, was seen as the patron deity of war. Athena represented intelligence and strategy. A protector with intelligence and strategy is defensive in nature. However, the ancient aegis also had an offensive nature. The aegis of Zeus was said to have been made by the god of fire and has been described as being "awful" and "fearful to behold." Similarly, Athena's aegis was a shield covered with goatskin, bordered with snakes, and had the head of Medusa, the Gorgon, attached to it. The head of Medusa was said to turn anyone who looked at it into stone. From this one can determine that an aegis was considered to be an instrument of both defense and divine protection.

Several of the U.S. Navy's AEGIS facilities are located at the Naval Surface Warfare Center, Dahlgren Division (NSWCDD) in Dahlgren, Virginia. Of these facilities, the AEGIS Computer Center (ACC) is of primary concern in this paper. The original mission of the ACC was to provide a site which was adequately equipped and staffed to support the Lifetime Support Engineering Agent (LSEA) in the activities of generating, maintaining, updating, and certifying ACS computer programs. As the lifetime support engineering (LSE) activity of the ACS has matured, the evolving mission of the ACC has changed into one which provides the facilities, equipment, and support services needed for the ACS LSEA activities, the evaluation of the applicability of emerging technology to future combat systems, and advanced systems modeling

In support of these missions, the ACC contains computers as well as peripherals and display consoles needed to run the tactical system and simulate the ACS. Additionally, the ACC contains a general purpose computer suite that is used in the reduction and analysis of data. Both test data generated in the ACC and real data generated on ships are processed in the ACC. The system of application programs that has been built to reduce then begin the process of analysis of the AEGIS data is called the AEGIS Data Analysis and Reduction (ADAR) system.

ADAR is an evolving collection of programs that have been and continue to be developed at NSWCDD for use with data recorded by an AEGIS tactical system. ADAR programs are used to interpret, analyze, and reduce AEGIS data then generate reports with the results. As the AEGIS computer programs have continued to develop and change with advances in technology, early decisions regarding ADAR programs have created problems that have required changes in methodology. While some of these changes have been minor, many of them have been major. Currently the ADAR system is evolving to handle new types of data and media that were not even imagined when the AEGIS program began.

This paper is a study of the early decisions regarding ADAR programs, the reasons for these decisions, and the problems they later caused. Additionally, this paper examines the evolving technology both within the AEGIS program and the computer industry that has led to or dictated additional changes within the ADAR system.

The audience of this paper is anyone who is interested in the development of the ADAR system. This person may or may not be from the AEGIS community. While the examples given within this paper are intended to accurately illustrate the concepts discussed, they are not intended to be exact representations of AEGIS data. This paper is not intended to be a "how to" for working with ADAR.

The ADAR user or analyst may be anyone from the AEGIS community who works with AEGIS data. They could be engineers, data analysis, programmers, or testers.

2 The AEGIS Weapon System and AEGIS Combat System

AWS is a fast-reaction, high-performance, computer-controlled weapon system. The radar used by AWS was developed specifically for it. This radar does not have a dish or any moving parts. Instead, it appears as special flat sections on the ship's surface. The roughly octagonal-shaped sections can be seen in Figures 1-1 and 1-2. While there are four of these panels on each ship, only one can be seen in Figure 1-1 and two can barely be seen in Figure 1-2. This radar is able to detect contacts in all directions. It can detect and track hundreds of targets, engage threats, and, at the same time, continuously maintain surveillance of the skies from the horizon through the zenith. AWS integrates the functions of shipboard detection, control, and engagement. It is also capable of a fully automatic reaction to an intense air attack.

ACS is a federation of AWS, the U.S. Navy antiair, antisubmarine, and surface warfare systems, and the weapons located aboard an AEGIS ship. With the diversity of weapons available, the ACS is able to adapt its system so as to be able to fight in all warfare areas simultaneously.

ACS is composed of the nine subsystems or elements of AWS plus more than fifteen additional elements. These elements are integrated system components that are loaded primarily into AN/UYK-43B or AN/UYK-7 mainframe computers, depending on

the applicable baseline. These elements are able to communicate with other ACS elements. In addition, these elements are able to communicate with the other AWS elements, as well as with systems external to the AEGIS systems.

The AWS is composed of the following nine elements: the AN/SPY Radar System (SPY), the Weapons Control System (WCS), the Fire Control System (FCS), the Command and Decision System (CND), the Operational Readiness Test System (ORTS), the AEGIS Combat Training System (ACTS), the AEGIS Display System (ADS), the STANDARD Missile (SM), and the Vertical Launching System (VLS). Originally, CND was called the Command and Control System, and the Guided Missile Launching System (GMLS) was used to launch missiles rather than the current VLS. The elements of the ACS are grouped into functional groups for detection, control, engagement, and mission support.

The U.S. Navy uses Block upgrades when implementing improvements to ships. Block upgrades provide for the installation of specified changes or change packages to a group of ships. The intention is to standardize the configuration of a specific group of ships while at the same time permitting the modification of ship groups to take advantage of improvements in either technology or capabilities. Block upgrades along with their associated configurations are referred to as baselines.

While there are nine elements, a minimal configuration, for baselines 1 through 3 testing purposes, consists of CND, WCS, and SPY. On the other hand, for baselines 4 and 5, a minimal configuration consists of CND, WCS, SPY, and ADS. There are

simulation programs that interface with these systems to provide the needed inputs via messages to WCS, SPY, and CND and to receive the messages output by them. These simulation programs also have the ability to provide any feedback needed after receiving messages. The messages generated by the simulations include tracks for SPY to detect as well as the communications between SPY and the STANDARD missile. Tracks are items that may be detected by SPY, usually moving objects. Some examples of tracks are outgoing missiles, incoming missiles, chaff, and debris. Chaff is material put into the air in an attempt to confuse the radar in the hope that an incoming missile will go undetected. Debris is created when a target is hit.

Each of the AWS elements communicates directly only with specific elements either in the ACS or external to it. The AWS elements also may communicate with other elements indirectly via messages sent through the elements with which they directly communicate. This direct and indirect communication is defined in various interface design specifications (IDSs). For example, the WCS element communicates directly with the CND and SPY elements along with the engagement grouping of elements. However, any communication between the WCS element and the detection grouping must be through the CND element. Also, WCS serves as the interface element between CND and all the ship's weapon systems.

The main computers housing the tactical system programs are the Univac designed AN/UYK-7 for baselines 1 through 3 and the Sperry designed AN/UYK-43B for baselines 4 and above. While the AN/UYK-7 computer can be a stand-alone unit,

most of the time within the AWS, the AN/UYK-7 computers are found in four-bay computer suites. Figure 2-1 shows an example of a two-bay AN/UYK-7 computer suite.

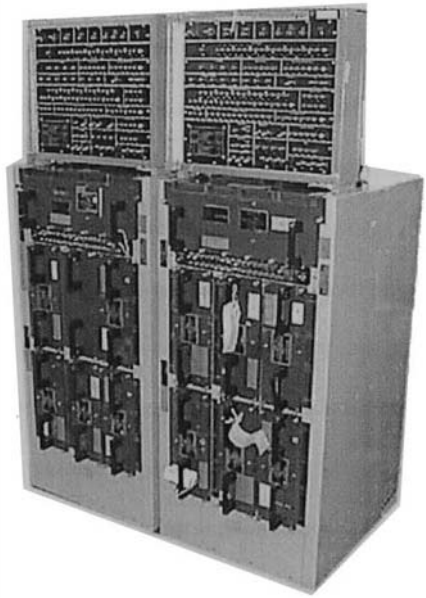

Figure 2-1: Two-bay AN/UYK-7 computer suite

A four-bay computer suite consists of four separate cabinets containing a total of four central processing units, four input-output interface adapters, and four input-output controllers. The input-output interface adapters and controllers as well as the memory modules are located in removable plug-in modules. These four computer bays are connected via cables in the back and are able to support memory sharing and memory overlap data processing. The cables serve as computer buses for instruction, operand, and I/O processes.

The ACS in the baselines 1 through 3 configurations utilizes 17 AN/UYK-7 computer bays. One of these bays is used for maintenance with the others arranged in four four-bay computer suites for the elements CND, SPY, WCS and FCS combined, and ADS. Baselines 1 through 3 are still deployed on active ships. Due to the expense of converting the physical configuration of the earlier baseline ships, it will be difficult to convert them to higher configurations. However, there are several proposals to convert and upgrade the earlier baseline ships.

Originally these AN/UYK-7 four-bay computers contained a total of 16 standard magnetic core memory modules. Four of these memory modules, one per four-bay computer suite, have since been replaced with double density film memory (DDFM) modules that, as the name implies, can store twice the amount of data. The DDFM module, due to the larger number of address available, counts as two memory modules. The first module is DDFM, containing the addresses for two modules. This essentially gives each four-bay AN/UYK-7 computer suite seventeen memory units. Each standard magnetic core memory module contains $40K_8$ addresses, that is, $16{,}384_{10}$ 32-bit words. Each DDFM holds $77{,}777_8$ addresses, that is, $32{,}767_{10}$ 32-bit words. With seventeen memory units the number of addresses available comes to just under the maximum of one megabyte that can be addressed.

While each AN/UYK-7 memory module contains eight access ports, a given memory module can address only seven other modules. As a result, not all modules are

reachable from a given module. On the other hand, each central processor and input-output controller can address up to 262K words or 16 modules.

The AN/UYK-7 computers are built so that the memory modules, with their handles protruding away from the computer, occupy the bottom two-thirds of the front of each computer. The control panel for each computer contains rows of button lights with identifying words for each grouping of lights printed on the cabinet. Additionally, there are several flip switches. Commands are issued via entering the correct binary numbers by pushing the correct buttons or, in some cases, by flipping the switches to the correct position. Feedback to the user is likewise displayed via these button lights. There is no front cover for the computer, one bay of which measures approximately 20 inches across the front, 22 inches front to back, and 41 inches high. The control panel, sitting on top of the computer, measures approximately 19 inches across the front, 6 inches front to back, and 19 inches high.

Beginning with baseline 4, the tactical code was ported to AN/UYK-43B mainframe computers with most elements housed on one AN/UYK-43B per element. Since then, the AN/UYK-43B computers have been upgraded with more memory and other capabilities. Figure 2-2 shows an example of an AN/UYK-43B computer.

The AN/UYK-43B computers provide more processing capability, memory, and I/O capacity than the AN/UYK-7 computers. The amount of memory in the AN/UYK-43B computers varies between the baselines with baseline 5 requiring more memory than baseline 4. Each AN/UYK-43B computer contains 10 memory modules along with

two central processing units, two input-output interface adapters, and two input-output controllers. Each memory module can contain one or two megabytes, depending on the

Figure 2-2: AN/UYK-43B computer

specifications provided by the U.S. Navy. The memory modules are still located in removable plug-in modules. However, the input-output controllers and interface adapters are located on plug-in memory cards. The memory cards are located behind an easily removed panel above the memory modules. All of these modules and the panel are hidden behind a front cover that contains buttons and a light panel. The buttons have either numbers or words displayed on them. Commands are entered via these buttons. Frequently the commands used on the AN/UYK-43B computers are the same as are used

on the AN/UYK-7 computers but converted to octal numbers. Feedback to the user is provided via a light panel that displays the appropriate words and numbers.

Generally, the AN/UYK-43B computers emulate AN/UYK-7 computers except that they can address higher memory addresses. The tactical program was written originally for the AN/UYK-7 computers and has never been completely rewritten. In a given AN/UYK-43B, the applicable executable tactical program is loaded into modules 0 through 4, then redundantly into modules 5 through 9. The redundant executable is kept updated by the main executable. This way, if there is a problem with the tactical program in the first grouping of modules, the second grouping is ready to take over the processing. This would be especially important if there were a real engagement occurring at the time that a problem occurred in the first group of modules.

The ACS in the baselines 4 and above configurations utilizes eight AN/UYK-43B computer bays. One of these bays is used for maintenance. The other AN/UYK-43B bays contain one element per bay.

While the basic AEGIS computer for baselines 1 through 3 is the AN/UYK-7, AN/UYK-20 computers provide additional processing capacity. Several AN/UYK-20 computers help distribute the processing load, keep the AN/UYK-7s informed of new statuses for the AWS, the associated hardware, and configuration as well as provide dedicated processors as required. The AN/UYK-20 computer was designed to handle small and medium-sized applications within military environments and physically is approximately half the size of an AN/UYK-7. They are stacked two high when in use.

When the AN/UYK-43B computers for later baselines replaced the AN/UYK-7 computers, the AN/UYK-44 computers replaced the AN/UYK-20 computers. Figure 2-3 shows an AN/UYK-20 computer while Figure 2-4 shows an AN/UYK-44 computer.

Figure 2-3: AN/UYK-20 computer

Figure 2-4: AN/UYK-44 computer

SPY detects targets automatically then tracks these targets using computer control. SPY also transmits uplink commands and receives downlink data from the STANDARD missiles via Link-11 and Link-16 datalinks. For baselines 1 through 3 the radar is under the direction of the SPY computer program housed in one four-bay AN/UYK-7 computer suite. All baselines since baseline 3 use one AN/UYK-43 computer for the SPY computer program.

CND is a manned display and computer complex that performs management and coordination functions for the Combat System. Additionally, CND performs threat evaluation, establishes reaction modes and priorities, and makes weapon assignments for ownship weapons, assigned aircraft, and other participating units while coordinating the inputs from the SPY system. For baselines 1 through 3 the control functions of Command and Decision are provided via the CND computer program housed in one four-bay AN/UYK-7 computer suite. All baselines since baseline 3 use one AN/UYK-43B computer for the CND computer program.

WCS provides the weapon engagement scheduling, control, and assessments for the combat system. WCS uses computers that interface with the operator display consoles via the Command and Decision System to coordinate and direct the employment of interceptor aircraft, STANDARD missiles, close-in defense weapons, surface-to-surface weapons, and antisubmarine warfare weapons. Additionally, WCS controls STANDARD missiles, both SM-1s and SM-2s, via both the FCS and either GMLS for Baseline 1 ships or VLS for all other ships. Update commands to and data

from STANDARD missiles are communicated via the data contained in uplinks and downlinks transmitted through SPY. For baselines 1 through 3 the weapon control functions are provided via the WCS computer program housed in a four-bay AN/UYK-7 computer suite. This computer suite also houses the FCS computer program. All baselines since baseline 3 use one AN/UYK-43B computer for WCS by itself.

FCS controls the guidance of STANDARD missiles during the last moments of flight before intercepting their targets. Additionally, FCS controls launcher pointing and missile initialization on the AEGIS ships that have GMLS. For baselines 1 through 3 the FCS computer program resides in one bay of the WCS four-bay AN/UYK-7 computer and four AN/UYK-20 computers. Since baseline 4, the FCS has been housed in one AN/UYK-43B computer of its own.

GMLS is capable of launching both STANDARD missiles and antisubmarine rockets. GMLS is used only in baseline 1 ships.

VLS is capable of launching both STANDARD missiles and TOMAHAWK cruise missiles. VLS is on all ships in baselines 2 through 6 beginning with CG-52.

STANDARD missiles, the primary weapon in the ACS, are launched from either GMLS or VLS. They have surface-to-air as well as surface-to-surface capabilities.

ORTS performs on-line monitoring, testing, and readiness assessments. All functions are handled automatically and are controlled by the ORTS computer program that is resident in several AN/UYK-20 computers, depending on the configuration being used, for baselines 1 through 3 and an AN/UYK-43B for baselines 4 and above.

ADS, a manned computer-driven display complex, provides management information and tactical situation displays for ownship command. This system receives tactical data from CND via a direct digital interface along with manually entered data. The various data, including maps, are displayed overhead on large screens and automatic status boards as well as on cathode ray tubes at the ADS associated consoles. The ADS computer program was added after baseline 3 to provide large screen displays in the operations room and is housed in one AN/UYK-43B computer. Later, the ADS was back-fitted on the baselines 1 through 3 ships where ADS is housed in one four-bay AN/UYK-7 computer.

ACTS is designed to train operators using on-line operator displays and controls. Previously generated scenarios, loaded into the ACTS computer, are used to provide a variety of antiair, antisubmarine, and antisurface warfare situations for training operators. Large-scale force training in battleground situations is handled via datalinks from Link-11 and Link-16. During this training sailors do not need to be in the same local area. The ACTS computer program is housed in one AN/UYK-43B computer for baselines 4 and above.

SPY and the other radar and surveillance systems compose the detection grouping of ACS. These elements provide the capability to detect, identify, and track air, surface, and subsurface contacts.

The control group includes both CND and WCS. These elements provide the needed management, coordination, and control of the combat system from the time

detection is made through the time that either engagement occurs or CND determines that there is no threat.

The engagement grouping of elements includes all the forms of weapons located on an AEGIS ship as well as on aircraft in the vicinity of the ship. Some of these elements are FCS, STANDARD missile, VLS, and the Light Airborne Multi-Purpose System (LAMPS) with its associated helicopter housed on the AEGIS ship. These elements work together to provide defense against long-range and short-range air, surface, and subsurface targets.

The mission support group contains the elements that provide support to the overall functions of the detection, control, and engagement groups. The ship's Navigation and the Radio Communications Systems, along with ORTS are included in this group.

The AEGIS Tactical Executive System (ATES) computer program manages the elements in the ACS. ATES, coded in the standard Navy CMS-2Y programming language, is resident in the memory of each of the AN/UYK computers. While ATES performs several different functions, the one of concern in this paper is the management of data recording for all the ACS elements.

Most of the AWS tactical code was written using one of the CMS-2 programming languages, either CMS-2Y or CMS-2M. While some tactical code is still being written in CMS-2, much of the tactical code developed for baseline 6 or code that

needed extensive modifications from previous baselines has been written in either C++ or Ada 83.

Baseline changes are not limited to the installation of physical improvements. Through the years some of the changes have included the addition of the TOMAHAWK Weapon System (TWS), the addition, then improvement, of ADS, the replacement of older operator consoles by more advanced equipment, and the replacement of the AN/UYK-7 tactical system computers by the AN/UYK-43B computers. The AN/UYK-43B computers take up less space while at the same time providing more memory, more processing capability, and more I/O capacity than the AN/UYK-7 computers. Also, the tactical system software has been enhanced with both additional capability and corrections of previously undetected errors both in logic and coding.

Baseline 6 of the AWS, using AN/UYK-43B computers, is currently being developed and tested. Baselines 1 through 5, in various partial baseline increments, are currently deployed on ships. For instance, baseline 5 may actually be deployed as baselines such as 5.0.3, 5.1.4, or 5.2.5.

The executable tactical computer programs for all deployed and developmental baselines are housed in the ACC. The tactical computers in the ACC may be configured and loaded with the executable code for any of the existing baselines. This is done to check out reported problems or to answer questions that might arise regarding the baselines.

The four computer suites in the ACC accommodate baselines 1 through 3, 4 and 5, 5 only, and 6 only. The computer suites in the ACC can be reconfigured easily and quickly at any time to specific desired baselines. Currently this is routinely done every four hours to accommodate testing schedules.

3 The CMS-2 Programming Language

Originally, all of the AWS tactical code was written one of the CMS-2 programming languages. CMS-2Y is used on both the AN/UYK-43B and AN/UYK-7 computers while CMS-2M is used on the AN/UYK-44 and AN/UYK-20 computers.

The CMS-2 programming languages are high-level, structured languages that are table and data driven. They are used to write programs for military and scientific applications. Military type applications include real-time command and control, radar control, and weapons control systems. The large-scale tactical data processing and real-time systems required by the U.S. military make stringent space and time demands. These demands require a language that is flexible and provides programmer control of machine features. Most high-level computer programming languages do not provide enough flexibility or programmer control to serve the needs of the U.S. military.

The CMS-2 programming languages, both the M and Y variations, originated with CS-1. CS-1 stood for Compiling System-1. The U.S. Navy developed CS-1 in the 1950s for use on the CP-642/USQ-20 family of second-generation computers. Once third generation computers were introduced, with the realization that more advanced computers were in development, it was decided that a programming language with more power and flexibility was needed.

CMS-2Q, the first version of CMS-2, was used for programs running on the CP-642 and the AN/UYK-7 computers. Since CMS-2Q recognized some of the CS-1 statements, it served as a transition during the development of the full CMS-2 programming language. The CMS-2M version of CMS-2 incorporated some features that were specifically designed for mini-computers using 16-bit words. On the other hand, the CMS-2Y version, using 32-bit words, implements the full scope of CMS-2.

While many of the constructs and declarations are different in CMS-2Y and CMS-2M, they are, nevertheless, very similar. The CMS-2 programming languages use block structures and have structured programming capabilities. They use English words and phrases along with algebraic phrases and arithmetic statements to solve problems. They also feature the ability to compile program segments independently. Two other features, providing a relatively machine-independent programming language as well as programs that are easily read and understood, are debatable by today's standards.

A CMS-2 program has two parts. Declarative statements, grouped into units called data blocks, both provide direction to the compiler and define the data to be manipulated. Grouped into procedures and functions, dynamic statements both manipulate data and describe logic flow. Data declarations include variables representing single pieces of data as well as tables consisting of groups of related items. A data table defines a database that can be both used within the context of the program for data storage and referenced as an extraction point. Extraction points will be explained in section 4.

The CMS-2 languages have six types of variables. Three of these variable types are numeric and three are non-numeric. Figure 3-1 illustrates the general pattern used for single data unit definitions in CMS-2.

VRBL	<identifier>	<type>	[P	<preset value>]$

Figure 3-1: Generic CMS-2 Single Data Definition

In Figure 3-1, "VRBL" is a reserved word that must precede all individual variable definitions. "Identifier" is the name of the variable. "Type" is a description of the variable that includes all information needed to describe the kind of value, its size, and its sign. While type is optional in the CMS-2 languages, it is generally spelled out in AEGIS code. Additionally, "P" indicates there is a preset value. Both the "P" and the following preset value are also optional. If "P" is present, a value for the variable must follow the "P." All lines are ended with a "$" sign.

The three numeric variable types in the CMS-2 languages are integer, fixed-point, and floating point. Integer variables, represented by "I," may be only whole number values. Fixed-point variables, represented by "A," are real numbers and may represent fractional values with a fixed number of bits. The number of fractional bits is declared in the definition of a fixed-point variable. Floating-point variables, represented by "F," have values expressed in the format of the executing computer's floating-point hardware. The computer determines the sign of floating-point numbers, along with the

size of the fractional parts. Floating-point operations tend to consume great amounts of computer time; therefore, they generally are not used in AEGIS tactical code.

```
VRBL    COUNT   I  8 U    P     6        $
VRBL    PAY     A 19 U 7                $
VRBL    RATE    A 19 S 5                $
VRBL    FPAY    F          P  3.75E-4   $
```

Figure 3-2: CMS-2 Numeric Data Definition Examples

Figure 3-2 illustrates single numeric data unit declarations that can be found in CMS-2 code. The first line, with type "I 8 U," is read "an integer with 8 bits that is unsigned with a preset value of 6." Unsigned values can only be positive. The second line, with type "A 19 U 7," is read "a fixed-point value of 19 bits that is unsigned and has 7 fractional bits." The third line, with type "A 19 S 5," is read "a fixed-point value of 19 bits that is signed and has 5 fractional bits." The last line, with type "F," is read "a floating-point number with a preset value of 3.75E-4." The variable assignment values must be written in specific columns.

The three non-numeric variable types in the CMS-2 languages are Boolean, character, and status. Boolean variables, represented by "B," have a size of one bit and may only have the values of "0" or "1." Character variables, represented by an "H," are strings whose size must be specified. Status variables, represented by an "S," are similar to enumerated types in other programming languages. When a status variable is defined,

the possible values that can be assigned to the variable also are specified. Status variables generally are not used in AEGIS tactical system code.

```
VRBL   HOLD     B                                                    $
VRBL   LINE     H      120                                           $
VRBL   HEADER   H       15    P       H(NUMBER OF ITEMS)            $
VRBL   CHECK    S      'LOW','MEDIUM','HIGH'                        $
```

Figure 3-3: CMS-2 Non-numeric Data Definition Examples

Figure 3-3 illustrates single non-numeric data unit declarations that can be found in CMS-2 code. The first line, with type "B," is read "a Boolean value." The second line, with type "H 120," is read "a character data unit with a maximum of 120 characters." Character data units may contain a maximum of 132 characters. The third line, with type "H 15," is read "a character data unit with a maximum of 15 characters with a preset character constant that has a value of 'NUMBER OF ITEMS'." The "H" before the parenthesis indicates a character constant. The last line, with type "S," is read "a status constant that may have the values of LOW, MEDIUM, and HIGH and no others."

```
TABLE <name of table> <type> <number of words > <number of items >$
FIELD <field specification>                                        $
[FIELD <field specification>                                       $]
END-TABLE <name of table >                                         $
```

Figure 3-4: A CMS-2 Table Block Definition Example

Figure 3-4 illustrates the pattern used when defining a table block in CMS-2Y. The first line in a table block contains the table declaration, the last line has the end-table declaration, and all lines in between contain the declarations of any data units to be considered within the table. These data units, also called fields, are declared in the same manner as the single data unit definitions.

A table declaration consists of the word "TABLE," the name and type of the table, and the number of words per item as well as the number of items in the table. In the above pattern, the word "TABLE" indicates a table declaration. Each table must have a unique name, indicated by "name of table." Type can be either "H," for horizontal, or "V," for vertical. The number of words refers to the number of 32-bit words in each item. The number of items refers to the number of times the set of fields is repeated each time that the table is accessed.

A CMS-2Y word has 32 bits with the high order bits on the left. Figure 3-9 illustrates the layout of a CMS-2Y word.

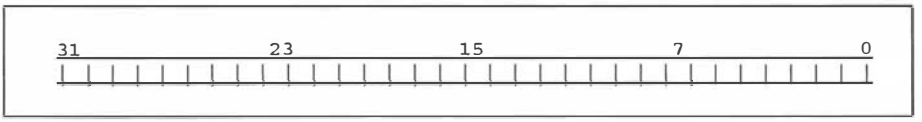

Figure 3-5: Representation of a CMS-2Y Word

Data storage can be either horizontal or vertical. Horizontal storage permits a rapid search of one word or field in all items while vertical storage permits a rapid search

of an entire item. The following illustrations show the differences between horizontal and vertical storage.

Figure 3-5 illustrates an example of a CMS-2 horizontal table declaration. The representation of the corresponding internal computer data storage of the various items and words in this horizontal table is shown in Figure 3-6.

```
TABLE      DUMMYH    H  2  3      $
```

Figure 3-6: First Line of an Example CMS-2 Horizontal Table Declaration

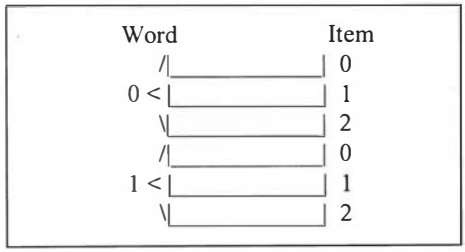

Figure 3-7: Internal Storage Representation of the Horizontal Table Declaration Example

Figure 3-7 illustrates an example of a CMS-2 vertical table declaration. The representation of the corresponding internal computer data storage of the various items and words in this vertical table is provided in Figure 4-8.

```
TABLE      DUMMYV    V  2  3      $
```

Figure 3-8: First Line of an Example CMS-2 Vertical Table Declaration

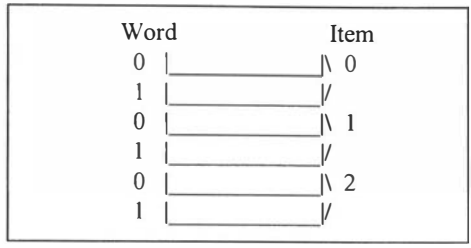

Figure 3-9: Internal Storage
Representation of the Vertical Table
Declaration Example

The word "FIELD" indicates a field declaration within a table declaration. There maybe one or more field specifications in a table. The field definitions look much like a single data unit declaration with the "VRBL" replaced by "FIELD." Complete table specifications must include all field names and their applicable descriptions. Each data word may contain from 1 field to 32 fields. The starting bit position is the left most bit of the field. The items needed for each field are written in columnar format. The pattern used for field specifications is shown in Figure 3-10.

FIELD <field name> <data type> <number of bits occupied for non-Boolean fields> <field sign for non-Boolean fields> <number of fractional bits> <word number> <starting bit> <optional P and preset value>$

Figure 3-10: CMS-2 Field Specification Pattern

The word "END-TABLE" indicates the end of the table declaration. The line with "END-TABLE" must also contain the name of the table along with the end of line marker, "$" as shown in Figure 3-4.

Tying all of the above items together, a more concrete illustration for a horizontal table definition is given in Figure 3-11, and that for a vertical table is given in Figure 3-12.

```
TABLE  DUMMYH  H  2  3
FIELD N1  I   8 U    0 31 $
FIELD N2  I   8 U    0 23 $
FIELD N3  I  13  S    0 15 $
FIELD N4  B          0  2 $
FIELD N5  B          0  1 $
FIELD N6  B          0  0 $
FIELD N7  A 32 S 10 1 31 $
END-TABLE DUMMYH  $
```

Figure 3-11: Horizontal Table Definition

```
TABLE DUMMYV  V  2  3
FIELD N1  I   8  U   0 31 $
FIELD N2  I   8  U   0 23 $
FIELD N3  I  13  S   0 15 $
FIELD N4  B          0  2 $
FIELD N5  B          0  1 $
FIELD N6  B          0  0 $
FIELD N7  A 32 S 10 1 31 $
END-TABLE DUMMYV  $
```

Figure 3-12: Vertical Table Definition

Tables in the AEGIS tactical system code are always stored as vertical tables during the recording process, regardless of their original definition. In other words, a table that is defined to be horizontal is converted to a vertical representation during the recording process. Since the horizontal representation is not used, only the internal computer representation for vertical table storage is given. The internal computer representation for table DUMMYV, defined in Figure 3-12, is given in Figure 3-13.

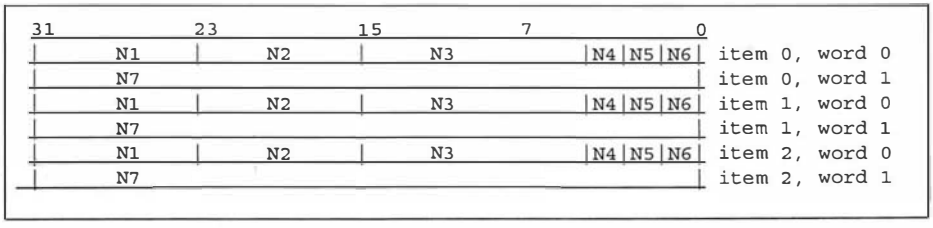

Figure 3-13: Table DUMMYV Computer Representation

4 AEGIS Data

Data is recorded during tactical system testing for later analysis. The analyst uses the recorded data to validate various operational aspects of the AEGIS tactical system. Some examples are system performance and effectiveness, testing of differences between baselines, determination of problems caused by software, and determination of the accuracy of data received from external sources. The recorded data falls into the categories of mission assignments, inserted doctrine, operating configurations, tactical engagement data, maintenance reports, fault isolation reports, and fault detection reports. The fault detection reports can be subdivided into the areas of system operability, element operability, element monitoring, and element error detection.

Data recording consists of extracting the desired data from the memory of the computer running the tactical computer program and writing the data onto magnetic media. Until recently, the primary magnetic media used has been tapes. The data to be extracted are stored in predefined areas of a tactical system's memory.

The AN/UYK computers have two buffers for storing recorded data. In baselines 1 through 3, the buffers in the AN/UYK-7s have a maximum size of 1024 32-bit words. In baselines 4 and up, the buffers in the AN/UYK-43Bs have a maximum size of 4096 32-bit words. When the active buffer is full, the buffer is marked as inactive and the data

recording function of the operating system initiates the transfer of the buffer's contents to the recording medium. At the same time, the other buffer becomes the active buffer. The buffer is full when there is not enough room left in the buffer to record another extraction point without exceeding the buffer's size limit. As a result, while one buffer has data written to it, the data in the other buffer is being written to another location. The magnetic tapes used are 9-track, i.e., 9 bits wide, 1600 BPI, phase encoded with a capacity of 10 MW. The amount of data generated per test can run from a small portion of one tape per element to spanning several tapes per element. Some of the factors affecting the amount of data generated are as follows:

- The particular extraction point(s) specified, which in turn depends on the purpose of the test,
- The number of items in the environment to be tracked, and
- The length of the test.

The ATES handles the management of data recording for the ACS elements running on the AN/UYK computers by writing the data in ATES format. Data are written to the tapes only when data recording is turned on via the data recording function of ATES during testing of the tactical system. Since the different elements run on separate computers, the data recording for each element is controlled separately. Data recording needs to be turned on only for the desired element(s).

When data recording is turned on, the operator must enter the needed identification information. This generally consists of the current date, the number pre-assigned to the tape, and the numbers for the desired extraction point sets for each element of interest.

All tapes for recording data are obtained from the tape library located in the ACC. Each tape has a unique identifying number assigned to it by the library. It is left to the user to keep track of his or her tapes along with their contents.

An extraction *point* is a data structure, i.e., table, in the tactical program code from which data will be recorded if data recording is turned on. An extraction point *number,* also called the extraction point identifier (EPID), is a unique number that identifies the data table that is to be recorded. An extraction point *set* consists of one or more extraction point numbers. While there are pre-defined extraction point sets, users can also either create an extraction point set or modify an existing extraction point set to contain only the desired extraction points. Additionally, each element has a default extraction point set that is recorded unless the operator indicates otherwise.

When an extraction point set is turned on during tactical system testing, each time an associated recording point for one of the extraction points is reached in the tactical program, the data stored in the computer's memory are written to an internal buffer. Each time the data is written, the specific instance reflects the state of the data structure at the time. The recording of extraction points is much like the printing of statements with the current value of variables while debugging a program. There can be a delay in transferring the stored data from the buffer to the tape. The various functions performed by the tactical system have priorities that determine which takes precedence when more than one function needs the computer resources. The data recording function has a lower priority than any of the tactical functions do.

The extraction point numbers for each baseline and element correspond to specific data tables in the CMS-2Y or CMS-2M generated tactical code. The extraction point number is used as a pointer to an area in the core memory for the applicable element. These tables are stored in the user common data area of the computer memory for the applicable element. The characteristics of the data tables, i.e., extraction points, that can be recorded are defined in a data recording item selection table (DRIST). The table definitions, i.e., DRISTs, are available to the ATES recording function when the tactical executable programs are loaded into the applicable AN/UYK computers. DRISTs later are used to produce data dictionaries. A DRIST specification file contains the DRIST definition section that defines a data dictionary. That is, the DRIST definition section contains the specific information needed to determine both the location of the data to be recorded and the amount of data to record for each extraction point. Each extraction point can be defined as an item of DRIST.

While each extraction point contains related data, several groupings of extraction points also contain related data. For example, an identifying track number, the x, y, and z coordinate positions of an item being tracked along with the x, y, and z velocities, and other related data may be contained in one extraction point. Due to the movement of a tracked item, the values for these items change frequently. As a result, this extraction point is recorded frequently. Another extraction point may contain the same identifying track number along with many flags that are set based on the type of the track as determined by either the tactical system or user entry. The values for these items tend to

change relatively seldom. As a result, this second extraction point does not need to be recorded very often. These two extraction points, in turn, may contain data that are related both by the identifying track numbers and by the time frame. These related data might need to be coordinated when evaluating the meaning of the data.

As the baselines have evolved the number of available extraction points have increased as well as the number of fields in some of the extraction points. Additionally, some of the extraction points have been modified in other ways. For instance, the fields in the CND extraction points 178 and 179 for baselines 1 through 3 were reorganized to form extraction points 104 and 105 for baselines 4 and above. Both extraction points 178 and 179 are recorded on a periodic basis. On the other hand, the recording of both extraction points 104 and 105 is handled on an event-driven basis. When extraction points 178 and 179 are requested, the data tapes fill up quickly. When the fields in these extraction points were reorganized, all fields that changed frequently, thus needing to be recorded frequently, were put in one extraction point while the fields that changed less often, thus needing to be recorded less frequently, were put in the other extraction point. Some examples of frequently changing fields are those involving distances, velocities, and accelerations. Many of the less frequently changing fields involved statuses. Both extraction points 104 and 105 are recorded when any of their values change, i.e., print-on-change (POC). This allows the values that change frequently to be recorded in one extraction point while the other extraction point, containing fields that do not change

often, is recorded much less frequently. As a result, more time can pass during a test of the tactical system before the CND-recorded data tape is full.

Each AWS baseline has its own set of extraction points divided by elements. The definitions for these extraction points are stored in data dictionaries on the classified Virtual Address Extension (VAX) mainframes used for ADAR programs. Data dictionaries are used to provide a relatively easy to read format of the definitions of the extraction points used by the tactical programs. Data dictionaries and data files are the primary input for ADAR programs.

The data dictionaries are initially generated from the DRIST via a data dictionary generator program, SYSBLD. The SYSBLD program links the CMS-2Y object to a DRIST specification file. The SYSBLD program uses the DRIST definition section from the DRIST specification file to generated data dictionaries. A new dictionary is produced every time a user links CMS-2Y object code. Once the data dictionary has been generated, the ADAR group may make modifications and additions, if desired. Several forms of data dictionaries are available, both in binary and American Standard Code for Information Interchange (ASCII) format. Programs reading data from either the recorded tape or the data file, created when the data were moved from the magnetic tape to a computer disk, use the binary data dictionary. One of the ASCII text file formats is a listing that is printed out and bound into documents. While ADAR programs can use several of the ASCII text file formats, it is faster to use the binary data dictionary.

With the exception of the listing form, all types of data dictionaries available on the classified cluster local-area network (LAN) contain all the information needed, by element and baseline, to unpack the raw recorded data. The data dictionaries contain specifications that match those used in the ACS tactical code. These specifications include all applicable extraction points and their fields. Figure 4-1 illustrates the general format used for the FIELD specification within an ASCII TABLE declaration.

```
FIELD <field name> <data type> <number of bits> <field sign>
<number of fractional bits> <word number> <starting bit>$
```

Figure 4-1: General FIELD Format Used in Data Dictionaries

In Figure 4-1, the number of bits refers to the size of the field. The field sign can be either "U" for unsigned or "S" for signed values. Word number refers to the number of the data word in the extraction point where the field is written. The starting bit is the first bit in the applicable word for the field.

In an ASCII data dictionary, the field values are written in columns. A specific number of character positions are allocated for each column with one space between the columns. If a field does not apply to a data type, it is left blank. For example, Boolean values generally are assumed to require one bit and do not have any fractional bits so those columns are not specified. All lines end with a "$" sign. Comments are indicated with the keyword "COM. Figure 4-2 contains an example of an ASCII data dictionary.

```
TABLE TAB1 V 3 1$
COM: TAB1 IS A 3 WORD TABLE WITH ONE ITEM.
FIELD T1IAA  I   8 U      0 31$
FIELD T1IBB  I   8 U      0 23$
FIELD T1ICC  I  16 S      0 15$
FIELD T1IDD  I  16 U      0 31$
FIELD T1BAA  B            1 15$
FIELD T1BBB  B            1 14$
FIELD T1IEE  I  14 U      1 13$
FIELD: T1DAE A 32 S 14    2 31$
TABLE TAB2 V 5 1$
COM: TAB2 IS A 4 WORD TABLE.
FIELD T2IAA  I  16 U      0 31$
FIELD T2IBB  I  16 S      0 15$
FIELD T2AAX  A 32 S 16    1 31$
FIELD T2AAY  A 32 S 16    2 31$
FIELD T2AAZ  A 32 S 16    3 31$
TABLE TABB V 5 1$
FIELD T4IBB  I  32 S      0 31$
FIELD T4ICC  I  16 S      1 31$
FIELD T4IDD  I   8 U      1 15$
FIELD T4IEE  I   4 U      1  7$
FIELD T4BAA  B            1  3$
FIELD T4BBB  B            1  2$
FIELD T4BCC  B            1  1$
FIELD T4BDD  B            1  0$
```

Figure 4-2: Data Dictionary Example

The first line in Figure 4-2, the header, contains the keyword "TABLE" followed by the fields <table name>, <V for vertical or H for horizontal>, <number of words in the table>, and <number of items in the table>. These fields are separated by spaces. The lines after the comment line contain the field definitions beginning with the keyword "FIELD", e.g., FIELD T1IAA.

If the ASCII listing is classified, the possible valid values may be listed along with the interpretation of these values. However, the meanings of combinations of fields are generally not contained in these listings.

Figure 4-3 illustrates the how the data recorded via the data dictionary in Figure 4-2 is represented within the computer memory and as stored on data tapes. In Figure 4-3, a blank line separates the various extraction point representations. The top line shows the bit positions within each word.

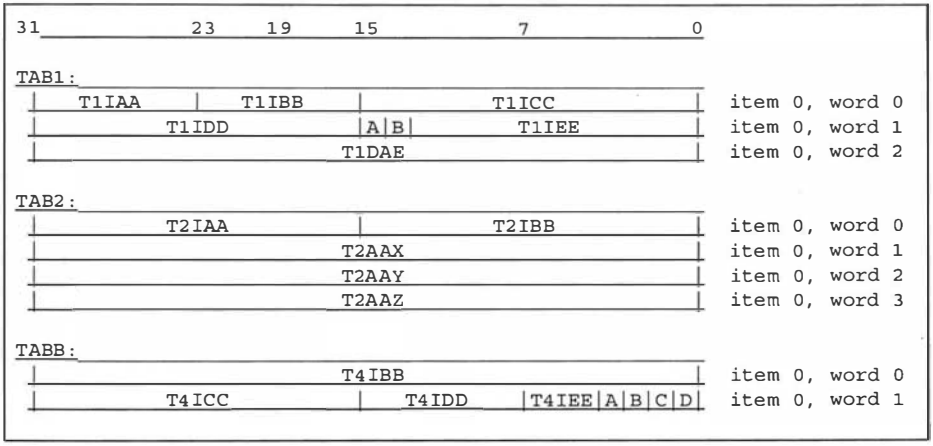

Figure 4-3: Storage of Data Dictionary Illustrated in Figure 4-2

Every time an extraction point is recorded, a header is attached to the data fields. This header always has the following fields:

- T (for Third word header) containing 1 if the header has three words and 0 if the header has two words,
- EPID containing the extraction point identifier, i.e., number of the extraction point,

- SID containing the system element identification number associated with the applicable AEGIS element,
- Number-of-Words containing the number of data words in the extraction point, including the header, and
- Time-Of-Day containing the time the recording of the extraction point occurred.

The third word of the header, if applicable, contains the following two additional fields: Module ID containing the number associated with the module from which the data is recorded, and Item Index containing the identification of the single item from the multi-item table recorded by the extraction point.

Application programs use the header information to determine how to read the recorded data. Figure 4-4 contains a representation of a three-word header along with extraction point fields that correspond to the vertical table definition given in Figure 3-12. Since tables in AEGIS tactical code are always recorded as vertical tables regardless of their definition, there is no need to illustrate the storage of horizontal tables separately.

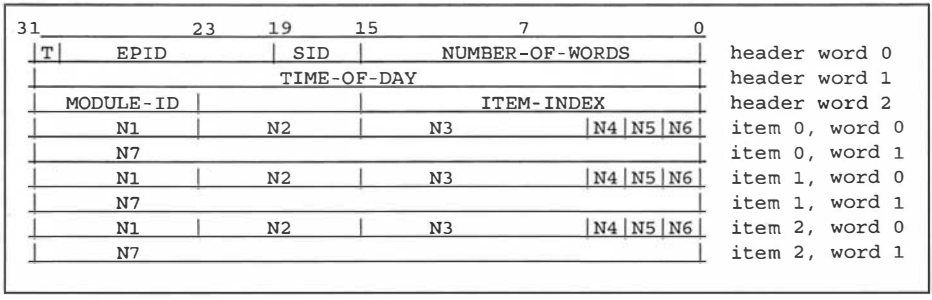

Figure 4-4: Three-word Header and Vertical Extraction Point Data Storage

Since CMS-2Y has been the primary programming language used for the tactical program code, the data tables used when extracting data from tapes are defined via CMS-2Y table constructs. Data from non-AEGIS elements, which have been recorded using other formats, also can be extracted by having appropriate data tables defined in a CMS-2Y format.

5 Functions of ADAR

5.1 Introduction

Before data can be examined effectively, it needs to be accessible to the programs chosen by the user. The application programs included in the ADAR system handle this accessibility in addition to providing special purpose analysis tools for the user. Additionally, new special purpose programs are developed and added to the ADAR system as the need arises.

5.2 Recorded Data

In the simplest terms, ADAR reduces the amount of data to be processed and aids the analyst in examining the data. Since a data tape easily can contain over half-a-million extraction point records, being able to reduce the amount of data to be examined is a must. During the early days of ADAR development, database and spreadsheet application program development was in its infancy. Therefore, the ADAR system was developed independently of any commercial database or spreadsheet applications.

Originally, even the mainframe computers could not hold the data from many data tapes. Therefore, the analysis programs frequently processed the data directly from the tapes without copying the data to disk files. Currently, in order for the data to be accessible to the programs, the data are usually copied from the recorded data tape onto a

disk drive of a mainframe computer, creating an ATES tape image disk file. This process is referred to as "moving the data." The only changes in the data at this point are those dictated by the manner in which the various computer platforms record their data words. That is, data recorded on an AN/UYK-43B must be converted to the correct format when it is transferred to a VAX mainframe and to yet another format if it is transferred to a PC.

The AN/UYK-7 computer records data in 32-bit words, from bit number 31 as the highest order bit to bit 0 as the lowest. This is referred to a little Endian order. On the other hand, the DEC 2020 used 36-bit words, with bit number 35 as the highest order bit. Any programs used to copy the recorded data from tapes to the DEC 2020 memory had to be written to change one AN/UYK-7 word into one DEC word. This change included right shifting all bits in each word four bit positions. When VAX computers replaced the DEC computers, these same programs had to be rewritten to accommodate the VAX's 32-bit words. The VAX's 32-bit words had the added wrinkle that the byte order was the reverse of that found in the AN/UYK-7 generated data. The word order on VAX computers is referred to as Big Endian order.

bits	31-24	23-16	15-8	7-0
AN/UYK-7 bytes	A	B	C	D
VAX bytes	D	C	B	A

Figure 5-1: Comparison of VAX and AN/UYK-7 Byte Layouts

As illustrated in Figure 5-1, bits 31 through 24 (Group A) on AEGIS data tapes became bits 7 though 0 when stored on the VAX. Bits 23 through 16 (Group B) on

AEGIS data tapes became bits 15 though 8 when moved to the VAX. Bits 15 through 8 (Group C) on AEGIS data tapes became bits 23 though 16 when stored on the VAX. Bits 7 through 0 (Group D) on AEGIS data tapes became bits 31 though 24 when moved to the VAX.

One of the utility programs in the ADAR system handles the moving of the data from the recorded data tape to a file location on the disk drive. Then a program that reduces the amount of data to be analyzed may be run on the data contained in the raw data file. The resulting data file may then be used "as is" by the analyst, as input into another special-purpose program, or as input into an application such as a spreadsheet. Through the years, many analysis programs have been developed for processing the data generated by the ACS. These analysis programs are also part of the ADAR system. Additionally, the ADAR system includes several programs that generate data reduction program skeletons.

5.3 Computers

The mainframes used by the analysts during the early days of ADAR, DECSYSTEM-20s, had limited memory compared to the amount of data that was generated. Due to the amount of disk space required to store all data from a given tape, the analysts were encouraged to not move the data to disk files. This meant a desired tape had to be mounted on a tape drive each time a program was to be run on the data contained on tape.

At the current time, the entire contents of many recorded data tapes are stored on the disk drives of the mainframes, primarily VAX 6460s and 8650s. Since the data that are recorded are generally classified, the mainframes used are part of a LAN that is generally known as the "classified cluster." The classified cluster LAN is only available via terminals, desktop computers, and workstations that are connected directly to the classified cluster LAN or are connected via a classified wide-area network (WAN). At one time all of these computers and mainframe terminals were located on or near the NSWCDD base. Now other sites can access this classified cluster. Some of these additional sites are NWAS; NSWC Port Hueneme (NSWCPHD), California; ACSC; CSEDS; and Lockheed Martin, Moorestown, NJ. This list of sites is always growing.

Since approximately 20 to 30 files can be stored per gigabyte of available space, the amount of disk space now available for desktop computers is large enough to allow the contents of several data tapes to be put on the hard drive of a classified personal computer. Lap top computers can hold the contents of approximately 50 data tapes. This provides for easier use of the data by the analysts. However, the data are not moved directly from the recorded data tapes to the various desktop computers and workstations. The data are first put onto VAX disk drives, then using File Transfer Protocol (FTP), are transferred to the desired personal computer. However, this ability is available only to those personnel whose offices are housed in the building adjacent to the ACC.

Special types of computers have been used to host the various application programs that operate on the recorded data. Originally, only DEC 2020s were used; then

DEC 2060s replaced the DEC 2020s. VAX computers later replaced these computers. The original VAX computers were first augmented then replaced by other VAX computers. Now the recorded data can be processed on many other platforms. Some of these platforms are the following: PCs, Silicon Graphics Incorporated (SGI) workstations, Alpha DECstations, Suns, etc. Currently, VAX 6460s, VAX 8650s, Alpha DECstation 3400s, a Silicon Graphics Onyx, and Silicon Graphics Indigo2s are being used along with PCs and Suns. Additionally, the ability to process the data tapes has been extended to other sites. This allows users who are familiar with one set of programs to continue using the same familiar programs while working at other sites.

5.4 Processing Data

Once the recorded data are transferred to the classified cluster, any of the various programs that have been written to process these data maybe used. Since the data are stored in binary format, processing is required in order to obtain results that are printable and humanly readable. In order to process the recorded data, the correct data dictionary first must be referenced. The binary data dictionaries can be referenced both by application programs when unpacking the recorded data and by programs that create either listings in ASCII format or plots for reference by users.

The two types of data manipulation programs either process data to create ASCII output or create plots. Since the amount of data recorded on any one tape can be overwhelming, an important aspect of processing data is to be able to select only those fields of data that are desired. This process helps to reduce the amount of data to a more

manageable size. The various special purpose programs that have been developed for use with ACS data either automatically handle this or provide a means for the user to specify, as a minimum, the desired extraction points and field or fields.

Additionally, the ADAR system includes several programs that generate data reduction program skeletons. Two such programs are Generate Multiple Extraction Points (GenMEP) and Generate Print-On-Change (GenPOC). They will be discussed in section 7.

Originally, the ADAR Sequential Processor (ASP) was the only method available for processing the recorded data. The ASP system was created in the 1970s at RCA (now Lockheed Martin), in Moorestown, NJ, during the early development of the ACS. ASP was designed for the general-purpose data reduction and analysis of the data generated by the AEGIS tactical computer programs during testing. Even though ASP proved to be cumbersome to use, its use continued through the early 1990s.

5.5 Job Processor

The Job Processor began as a tool to automate the development of ASP programs. Later, separate programs, written in the Formula Translation (FORTRAN) programming language, were incorporated into the Job Processor. The job processor gathered user input then created a command file to actually run the application program. The input provided by the user was written in the command file along with other inputs needed for the chosen application program. Once the command file had been completed, the job processor submitted the command file to the batch queue. As more programs and

AEGIS baselines were developed, the maintenance of the Job Processor and its various related programs became increasingly difficult.

Later, different programs, written in the C programming language, were developed to replace the most generic and frequently used FORTRAN programs. The most general of these programs is the Data Extraction Point Analyzer (DXEP). The original DXEP was introduced in 1995. It has continued to be enhanced extensively so that it now is a very versatile program. Consequently, it has become the most frequently used of the ADAR programs. Since the mid-1990s, many of the special purpose data analysis programs either have been rewritten in C or have been replace by new programs written in C.

6 Beginnings of ADAR: the ASP Interpreter

6.1 Origins

The tactical computer code and the original ADAR system were developed at RCA in Moorestown, NJ, during the early development cycle of the ACS. This original ADAR system centered on the ASP command language, which was designed for general-purpose data reduction and analysis of the data generated by the AEGIS tactical computer programs during testing. The design philosophy behind ASP was to create an interactive system whose basic commands could be learned quickly and easily. At the same time the ASP system would be flexible and powerful enough to do most data reduction tasks using short, easily written command sequences.

The first effort to bring the ADAR system to the Naval Surface Weapons Center (NSWC) occurred in 1979. A version of ASP on tape in binary format was hand-carried down to NSWC by base personnel and put on a DEC 2020 in the ACC. At that time, RCA was not willing to release the source code for ASP. Existing ASP programs, i.e., files containing a list of ASP commands that could be run via batch processing, for each of the AEGIS elements were also brought down to be used as examples. Since development of the elements was just starting, only the CND, SPY, WCS, and FCS

elements existed. Additionally, the user's guide for ASP, AEGIS Data Reduction System (ADAR) User's Guide (Preliminary), Version 2.0, published January 12, 1978, was hand-carried down to NSWC along with some existing data tapes for each of the existing elements. The person who was working on this project at NSWC spent many months trying to get the programs from Moorestown to work with the ACC computers. Finally, in 1989, some of the ASP source code was delivered to NSWC.

Once the people using ASP at NSWC had determined how to make it work, they found the ASP programs from Moorestown were very inefficient. When these ASP programs were rewritten, frequently they ran in half, or better, of their original time.

Originally, each ASP control file had to be generated manually. This very repetitive work was quite time consuming as well as being prone to many errors. In order to speed program development and reduce errors, a job processor was written to generate the control files needed to run ASP programs.

6.2 The ADAR Sequential Processor

The ASP Data Reduction System had two components: a language processor and a data processor. The language processor interpreted the commands the user entered via the defined ASP command language. Once the commands had been interpreted, the language processor generated another form of the command sequence called the process control table. This form was used by the data processor to direct the various data processing activities. The language processor and data processor communicated only via the process control table.

The ASP system was an interpreted language rather than being compiler driven. As a result, the issued commands, unless the user specified otherwise, were checked for correctness and acted upon immediately. If the user specified, the commands could be stored for later use.

In order to use ASP, the user must first have a copy of the data dictionary available on the computer system to be used. This data dictionary had to be compatible with the data file to be read. The data file could be either a disk file or the contents of a magnetic data tape. In order to be used, the magnetic data tape was "mounted" on a tape drive. The computer room operators handled "mounting" tapes.

Generally, the users were encouraged to copy both the data file and the associated data dictionary to the their area on the computer system, then to rename them. The data dictionary file was renamed "ASPCDT.DAT" while the data file was renamed to "RDT1.ATS." If a data tape was used, the user needed to set a variable in his environment to identify the tape drive on which was mounted the tape whose contents were to be used for input. The renaming of the files was done primarily because the ASP system used only these names when looking for files. Otherwise, the user needed to set variables to relate these default names to the actual names of the files. In addition, by copying the files, the user would be sure he did not change the original files.

The user then entered the System Identification (SID) and extraction point numbers that were to be used following the ASP keyword "SELECT." Figure 6-1 contains two examples of the SELECT command.

```
SELECT   7, 256;
SELECT   6, 45;
```

Figure 6-1: ASP SELECT Command

The first line in Figure 6-1 uses the SELECT command to tell the ASP system to use 7 for the SID value, i.e., the WCS element, and 256 for the extraction point number. The second line uses the SELECT command to tell the ASP system to use 6 for the SID value, i.e., the SPY element, and 46 for the extraction point number.

A line indicating where and how the output is to be produced follows the SELECT command. Figure 6-2 contains several examples of the OUTPUT command.

```
OUTPUT TO PRINTER;                                              (1)
OUTPUT TO TERMINAL(XSM,YSM,ZSM);                                (2)
OUTPUT TO FORT1.FORTRAN(MSLNUM,TIME);                           (3)
OUTPUT TO PRT2.PRINTER(LNCHNUM,MSLNUM);                         (4)
OUTPUT PRNT1 ($ALT, $AZ, $ELEV) (E15.7, E15.7, E15.7, /);       (5)
```

Figure 6-2: ASP OUTPUT Command

In Figure 6-2, the output from line (1) is to be printed straight to the line printer. Since there are no indications as to which fields should be printed out, the output, using the default value, will have the contents of all fields for every instance of the extraction point specified before this point. On the other hand, the output from line (2) will be displayed to the terminal screen. Since the XSM, YSM, and ZSM fields are specified, this output will have the contents of only those fields.

The output from line (3) in Figure 6-2 will be printed to an unformatted binary file named FORT1.UFD. A FORTRAN program with the proper read statements will be able to process the data from the unformatted binary file. This unformatted binary file will consist of the occurrences of the MSLNUM and TIME fields.

Line (4) in Figure 6-2 shows how to cause the output to be written to an ASCII file, named PRT2, in a form suitable for printing or reading at the terminal. The output from this line will consist of the contents of the LNCHNUM and MSLNUM fields.

While the field specifications in the data dictionary would be used for any default output formatting, the user could specify any desired formatting using FORTRAN-like commands. In the line (5) of Figure 6-2, the fields ALT, AZ, ELEV would each be written to a file with a maximum of fifteen spaces allowed for each value. Seven spaces would be allowed for the fractional parts with a carriage return placed at the end of each line.

As should be obvious, the above steps could be very tedious, especially when repeated every time data in any file were to be analyzed. Creating files, or even modifying existing files, containing the needed command lines took much time. In all cases, it was very easy to make typographical errors.

Using some of the examples above, Figure 6-3 displays simple instructions to the ASP program that will determine which extraction point is to be used then which fields in the extraction point to write to the output and the type of the output.

```
SELECT      7, 256;
OUTPUT TO TERMINAL(XSM,YSM,ZSM);
```

Figure 6-3: Simple ASP Command Sequence

The example in Figure 6-3 instructs ASP to look in the default data file "RDT1.ATS" for data and to use the default data dictionary "ASPCDT.DAT" for the data definitions. ASP will be expecting both of these files to be in the user's current directory. ASP is then told the data dictionary is for the WCS element, i.e., the SID value is 7, and to look for every instance of extraction point 256 in the data file. Every time extraction point 256 is found, the values found in the XSM, YSM, and ZSM fields are to be printed to the terminal screen using the formatting found in the data dictionary file.

Eventually, one of the users working with the WCS element group grew tired of generating the ASP control files individually. She developed a command line generator program, called a job processor, to automate the creation of these files. There were two major advantages to creating these files via an automated means. First, the files were created quickly with relatively little effort on the part of the analysts. This allowed them to make better use of their time. Second, fewer typographical mistakes were made. This also allowed the analysts to be more productive. While the job processor started out with one major objective, its complexity grew as more functionality was added to it.

When the user ran the job processor, several things happened. First, the job processor posed a series of questions to the user regarding the element, the baseline, the

extraction points, and the fields to be used. The user was expected to make any desired formatting changes in the local copy of the data dictionary before running the job processor. The job processor then used the answers supplied by the user to first create then submit, in batch mode, a command line file. This command line file, in turn, ran ASP by supplying the needed information to the ASP program to produce the results desired by the user.

7 Second stage of ADAR: the Job Processor

7.1 Introduction

The original Job Processor helped to automate the use of ASP. Later commonly used utility programs, written in FORTRAN, were developed then were incorporated into the Job Processor. To aid the users developing FORTRAN programs, program generators were developed. These program generators created skeleton programs, also in FORTRAN, that could be modified to produce the results desired by the users. While the missing ASP source code modules were never delivered to NSWC, the people at NSWC developed a work-around for this lack by developing their own routines and functions that did not require the use of ASP for processing AEGIS data. Consequently, gradually the use of ASP was discontinued as FORTRAN programs replaced generated ASP script files. Additional special purpose FORTRAN programs were also incorporated into the Job Processor. Eventually the Job Processor provided a way for users to locate and use programs that might have been unfamiliar to them.

7.2 Beginnings - Job Processor to Automate the Use of ASP

The generation of ASP control files by hand could be tedious and certainly was error prone. In addition, it could be very time consuming. Frequently, the same, or very similar, command files were needed for several data tapes. Also, similar command files

could be needed for the same tape but for a different slice of data. Sometimes modifying an existing command file provided a relatively easy way to create these similar command files. Additionally, new command sequences frequently were needed. A job processor, written in FORTRAN, was developed to help automate the development of the new command files. Each of these command line scripts was put into separate files. When used, these scripts acted as programs written in an interpreted language.

A separate subroutine was created within the job processor to handle each specialized use of ASP. Within each subroutine the job processor presented the user with a series of questions about ACS elements, extraction point(s), fields, and output format. All of the questions were centered on obtaining the information needed to create a command file to perform the desired operations. In addition to creating command files, the job processor would submit the command file for batch processing. When requested by the user, output files could also be sent to the printer queue. The data dictionaries located on the main frame computer provided the needed default formatting for the generated output.

By having a copy of the data dictionary in the user's disk space, the user was also able to modify the data dictionary so that the desired output format could be created by default. In addition, by removing unwanted fields from the local data dictionary file, only the desired fields would be put into the output. This eliminated the need to specify the fields in the command file, simplifying the creation of command files. This was

especially true when the command procedures required one or more of the following items:

a) Many items of input from the user,
b) Specific items of input from the user, or
c) Multiple programs to reach the final results.

7.3 Addition of Utility Programs to the Job Processor

The next step in automating some of the data processing steps involved enhancing the job processor to run commonly used FORTRAN programs. The first programs to be handled via the job processor addressed the most generic functions. Some examples of these FORTRAN programs are Data eXtraction MOVe (DXMOV), Data eXtraction SUMmary (DXSUM), and Extraction Point Attribute Collector (EPAC).

The DXMOV program copied the extraction point data from the recorded data tapes to a file on the disk of the current mainframe computer. DXMOV also handled the formatting changes to the data required by this move. The required changes, discussed previously in section 5.2, were created by the differences in computer storage requirements. The headers that were attached to each extraction point when put into the buffer before being written to the recording tape were read and used by DXMOV. These headers provided the information needed to break the data into logical records by allowing DXMOV to find the beginning and ending of the logical records on the tape.

The DXSUM program produced a summary of the extraction points that were recorded on a data tape or were in a disk file created from a recorded data tape. This

summary included the beginning and ending recording times as well as the number of instances of each extraction point on the tape or in the file.

The EPAC program generated ASCII and binary data dictionaries. ADAR Librarians or the ADAR group could change the ASCII data dictionaries to add more fields or to make corrections. Once a data dictionary was modified, EPAC could use it as input to check the changes and create a binary data dictionary for use on the system.

7.4 Data Dictionary Map

When there were just a few AEGIS baselines with which to work, it was not difficult to keep track of their associated data reduction programs. As the number of baselines multiplied, the amount of space required for storing separate data reduction programs for each baseline and element likewise multiplied. Eventually, the amount of disk storage space required to handle the baseline associated data reduction programs became prohibitive. Several factors contributed to the space difficulties: the number of elements increased, the number of extraction points for each element increased, and the number of fields in some of the extraction points also increased. As the number of extraction points increased towards the maximum of 1024 for each element, the memory and disk space requirements also increased.

However, not all of the extraction points changed between each baseline. Since this was the case, it was not necessary to store multiple copies of the Print-On-Change (POC) data reduction programs for the unchanged extraction points. Each POC program, developed for only one extraction point in one baseline, when run, would write

a new line whenever the value for any of the contained fields changed. The POC programs provided an easy way for the users to determine the times and new values for each change. Each time a POC data reduction program was developed, it was stored for future use. To help control the rapidly expanding memory and disk requirements needed to store all of the POC programs, a scheme was devised that created a Data Dictionary Map. This Data Dictionary Map is illustrated in Figure 7-1.

When a new baseline arrived at NSWC, a new computer directory was created and named for the baseline. Then POC data reduction programs for any new or changed extraction points were generated and stored in the directory named for the new baseline. The Data Dictionary Map then was updated to show the presence of the new computer directory and the new POC programs. When a user asked the job processor to run a POC program, he first indicated the baseline of interest then the job processor used the Data Dictionary Map to determine where to start looking for the program. If an applicable POC program were found, it would be run. Otherwise, an applicable POC program would be generated by the system, stored for future use, and then run.

With the Data Dictionary Map in place, a POC program for a given extraction point only needed to be stored if it were new or had been changed. Any POC job request would look at the Data Dictionary Map first. In the Data Dictionary Map, as illustrated in Figure 7-1, the "1" meant that the data reduction program existed for that extraction point in that particular baseline directory. On the other hand, the "0" meant the data reduction program for that extraction point would be found in the directory of an earlier

	extraction point 257	extraction point 315	extraction point 370	explanation
BL 5	1	0	1	Both extraction points 257 and 370 have changed. Data reduction programs may be found in the BL5 directory for extraction points 257 and 370 and in the directory of an earlier baseline for extraction point 315.
BL 4	0	0	1	Only extraction point 370 has changed. Data reduction programs may be found in the BL 4 directory for extraction point 370 in the directory of an earlier baseline for extraction points 257 and 315.
BL 3	0	1	0	Only extraction point 315 has changed. Data reduction programs may be found in the BL 3 directory for extraction point 315 and the directory of a previous baseline for extraction points 257 and 370.
BL 2	1	0	0	Only extraction point 257 has changed. Data reduction programs may be found in the BL 2 directory for extraction point 257 in the directory of a previous baseline for extraction points 315 and 370.
BL 1	1	1	1	The original directory, i.e., BL 1, contains POC programs for all extraction points.

Figure 7-1: Data Dictionary Map

baseline. So if a "0" was found, the Data Dictionary Map was consulted for the previous baseline. This process was repeated, one baseline at a time, until a "1" was found. Only when a "1" was found would the desired data reduction program be found. When this example is expanded many times, the savings in disk space requirements is obvious.

Reading from the bottom of Figure 7-1, BL 1 represents the starting baseline directory containing data reduction programs for all extraction points. The next line up, BL 2, has a "1" for extraction point 257, a "0" for extraction points 315 and 370. Only extraction point 257 has changed, so a data reduction program for extraction point 257 would be found in the computer directory named for BL 2. To find the data reduction programs for extraction points 315 and 370, one must fall through until a "1" is found, in this case, in the BL 1 directory.

Going up the table, BL 3 has a "0" for extraction point 257, a "1" for extraction point 315, and a "0" for extraction point 370. Only extraction point 315 has changed. The data reduction program for extraction point 315 would be found in the computer directory for BL 3. To find the data reduction programs for extraction points 257 and 370, one must fall through until a "1" is found. The data reduction program for extraction point 257 would be found in the BL 2 directory while that for extraction point 370 would be found in the BL 1 directory. A similar situation exists for both BL 4 and BL 5.

Referencing Figure 7-1, a POC program for baseline 4 would be obtained from the BL 2 directory for extraction point 257, from the BL 3 directory for extraction point 315, or from the BL 4 directory for extraction point 370.

7.5 Program Generators

In addition to the job processor, other programs were developed that created skeleton FORTRAN data analysis programs. These program generators were called

GenMEP and GenPOC. When run, the GenMEP and GenPOC programs produced FORTRAN programs designed for specific extraction points and fields. These FORTRAN programs were developed by users to provide specific information. These generated programs, in turn, would unpack the binary data contained in their selected extraction points and their associated fields then print this data out in the user-determined format. It was left up to the user to modify these programs to produce the desired output data. This modification frequently involved putting in program constructs to help with the filtering of the data, any repetition needed, and changing the output statements to produce data in an easier to read format.

The GenMEP and GenPOC programs required the output produced by the Generate Fields (GenFLD) program as input.

7.5.1 The GenFLD Program

Before either the GenMEP or the GenPOC program could be used, a data definition file had to be generated. This file, created by the GenFLD program, contained field definitions from the data dictionary, including default output format specifications, for every field chosen by the user.

When running GenFLD, the user was guided via a series of questions to choose the desired items needed for the output file. The user had to enter information for the following items:

- A data dictionary based on the desired baseline and element,
- The extraction point(s) to be used,
- The field(s) to be used in each of the selected extraction points,
- The sort field(s) to be used, and

- The column order in which the fields would be output.

The sort field could be none or one of several ways of sorting the data by specific fields. When no sort order was chosen, the desired fields would be output in the order in which they appeared in the data dictionary. While the user could choose any of the selected non-Boolean fields on which to sort, the usual choices were one or more of the identification numbers used within the ACS. These identification numbers, assigned by different AEGIS elements, are Control Group Track Number (CGTN), Weapons Control Index (WCI), Central Track Stores Locator (CTSL), and System Track Number (STN). The column order in which the desired data was to be output could be the order the fields appeared in the data dictionary, i.e., dictionary order; in the order the fields were chosen; alphabetically by fieldname; by field type only; or by field type then alphabetically by fieldname. However, only the GenPOC program used any of the sorting choices. Any time data were output in sorted order, the data were first written to a temporary file, sorted, then written to the final data file.

Implicit in this process was the assumption that the user knew exactly which field(s) to choose and in which extraction point they would be found. A novice at this process would need to do some preliminary research in order to determine the desired information. Frequently, the user would check with other users to determine the needed information. If this was not possible, the user could check the descriptions of the various fields in an ASCII version of the data dictionary. Once the user knew which extraction point was needed, the user could proceed. If the user was unsure of which fields to be

used, it was better to choose all possible fields in the applicable extraction point. Any fields that were later determined to be unneeded could be removed easily.

Once the data definition file was created, the user could easily edit it to remove unwanted fields, change the order in which the fields would be processed and/or output, or change the output format specifications. The contents of this file looked much like the contents of the ASCII formatted version of the data dictionary. If the file indicated a field was to be output as an integer, the user could easily change the format specifications for the field to be octal or binary. Once this file was created, the user could run either the GenMEP or the GenPOC program, specifying the name of the file when prompted.

7.5.2 The GenMEP Program

The GenMEP program produced data reduction programs written in FORTRAN for any number of extraction points from the same element and baseline. When the user ran the GenMEP program, only the name of the data definition file to be used needed to be specified; the type of output format and the classification to be assigned to any output data files. To make things easier, let us call the GenMEP generated program ExamMEP.

The output from ExamMEP could be in ASCII format, i.e., "formatted", or unformatted binary format. If the user wanted to be able to personally read the file generated by ExamMEP, the user would choose the ASCII format. On the other hand, if the user were planning to use the output from ExamMEP as input into yet another program, the user would generally choose the unformatted binary output. Since

unformatted binary data is faster for a computer to process, this was an important consideration when working with large amounts of data on the slower computers of years ago. Once ExamMEP was created, it had to be compiled and linked like any other FORTRAN program before it, in turn, could be run.

The programs produced by the GenMEP program started out with a main program body followed by one unpack subroutine and one print subroutine for each extraction point specified by the user. When ExamMEP was run, it would read a data file specified by the user. Every time ExamMEP found a new instance of an extraction point, it would determine whether or not the program handled the extraction point. If it was handled by ExamMEP, the main program of ExamMEP called the unpack subroutine for the given extraction point that in turn called the print subroutine for the same extraction point. This process was repeated, via a loop, until the end of the data file was reached. Since the original data was stored as unformatted binary, the term "unpacked" was used when converting the data into readable form.

The original output generated by ExamMEP was in namelist format. As shown in Figure 7-2, this output format consisted of each instance of an extraction point along with the number and title of the extraction point printed as a heading. Under the heading, the name of each unpacked field was printed out, followed by, first, an equal sign then by the value for the field as determined from the data. Consecutive field names and their values were printed on the same line until the 132-character line was full. When the next field name or value could not fit onto the current line, a new line was started.

EPID 256: EO THREAT VALUE UPDATE

DXTIME = 68500.235, CTSL = 0004, WCI = 035, XSM = 456.015, YSM = 23.987, ZSM = 23.001

EPID 360: CENTRAL TRACK STORES LOCATOR

DXTIME = 68500.238, CTSL = 0003, CGTN = 250, MTTAG = 68500.238, TVAL = 4, PREEM = 0

EPID 256: EO THREAT VALUE UPDATE

DXTIME = 68500.255, CTSL = 0004, WCI = 035, XSM = 456.915, YSM = 24.054, ZSM = 23.504

Figure 7-2: Namelist Format Example

This line wrapping was repeated until all desired fields in the particular extraction point had been unpacked. As can be seen, the original namelist output format was difficult to read when a large amount of data was processed.

The programs produced by GenMEP could be used as "starter" programs when developing analysis programs. The user could modify the initial program to produce an analysis program that would perform the needed operations on the data and output the resulting data in a desired format. The user could make many possible modifications to provide very complicated processing. For example, the program could be modified to do any of the following:

- Put all the fieldnames on one line and their associated values below them as illustrated in Figure 7-3 in the next section.
- Only print when a certain value of a given field was found, e.g., only when the value of the CTSL field was 4.
- Write the values found in different extraction points to different files.

- Relate fields found in one extraction point to fields found in another extraction point via the values found in specific fields then print the values for these fields and other associated fields.

7.5.3 The GenPOC Program

The GenPOC program also produces data reduction programs written in FORTRAN but for only one extraction point. When the user ran the GenPOC program, only the name of the data definition file to be used and the classification to be assigned to any output data files had to be specified. GenFLD previously had created the data definition file. To make things easier, let us call the GenPOC generated program ExamPOC. Whenever the value for any of the fields handled by ExamPOC changed, the values for all of the fields were printed, hence the POC name. The ExamPOC generated output was only in ASCII format. This output was written out with field names as column headings at the top of each page with the corresponding values for each field written under its heading in columnar format. The last column, labeled "OCCURS," provided a count of the number of times each printed line occurred without a change. While programs produced by the GenPOC program could be modified, this was more limited than with the GenMEP generated programs. Figure 7-3 contains an example of output produced by ExamPOC.

DXTIME	CTSL	WCI	MTTAG	TVAL	PREEM	OCCURS
68500.235	0004	004	456.015	3	0	10
68500.255	0004	004	456.915	3	0	5
68500.273	0004	004	458.205	4	0	15

Figure 7-3: Columnar Format Example

7.6 First Data Manipulation Programs

While many of the first FORTRAN-based data manipulation programs started out as GenMEP generated programs, others were developed without the use of the program generators. Two examples of the latter type of program were Get Recorded Data (GRD) and Do_Plot. The GRD program and the Do_Plot program were designed to be used for all baselines and all elements. When run, GRD would obtain the desired extraction point number and the names of fields from the user then produce output files with columns of data. The Do_Plot program could read the files produced by GRD and use the data to produce plots, i.e., graphs, using the field(s) selected by the user. While these programs were versatile, they were also very difficult to use. GRD and Do_Plot frequently required many user responses in order to produce desirable results. In order to produce comparable results, many of the same user entries were required each time. To make it easier to obtain consistent results using GRD and Do_Plot, subroutines were developed in the job processor. These subroutines worked like the subroutines that created batch files to run ASP.

The job processor subroutines, created for specific results, first queried the user for any needed information then created batch files to run the underlying program, for example GRD. The batch file would then supply the specific extraction point numbers and field names, in the proper order, to the program. Sometimes running one job processor subroutine would cause the same program to be run more than once with different inputs. In that case, the batch program would use the first output files as input

into yet another program that, in turn, would process and combine the data in some way. This new program expected input files that had the specific names and contents that were produced by the batch file before the new program was run. These batch files also took care of any needed file renaming in addition to the printing of the final results and the removal of intermediate files.

For example, the subroutine called WCS_SUM created batch files that ran GRD three times, once for each extraction point used. After each run of GRD, the generated files were rename to specific names. Once the three files were created, the batch file ran another program, also called WCS_SUM. The WCS_SUM program expected as input the three files, with their specific names, that had been created before it was run. WCS_SUM would process the data read from the three files, then generate a fourth file. Once the WCS_SUM program finished, the batch file deleted the files created by GRD and printed the file produced by WCS_SUM.

As the ADAR system matured, many specialized programs were developed. Some of these additional programs were used in place of the GRD and Do_Plot programs. While GRD and Do_Plot have not been removed from the classified cluster, they are no longer being maintained.

As each element group at NSWC developed programs, they were also incorporated into the job processor. Although the job processor was originally WCS-specific, it was eventually turned over to the ADAR group where the WCS-portion of the

processor became one of several modules. The other modules were for the other elements. Job processor modules also were developed for utility programs.

At first, each data analysis program handled all of its own processing. Only the system FORTRAN libraries were linked to the data analysis program. The programmers duplicated the coding of common functions from program to program. After a while functions and subroutines to handle commonly used program functions were developed. These functions were stored in libraries on the classified computer system. They could be linked with the programs after compilation. Once these functions and subroutines were developed, those sections were taken out of the older programs and replaced by calls to these functions and subroutines. This helped to standardized some of the output produced by the programs, particularly regarding page breaks, headers, banners, input and output of time and tape number.

7.7 Creation of the Training Manual

When the ASP system was brought to NSWC, a user's guide was provided to those working with ASP, but as the ADAR system developed, it soon became obvious that a training manual was needed to introduce basic AEGIS concepts and VAX commands. A training manual was especially important for people who did not know anything about these topics. Consequently, the Introducing ADAR manual was developed.

This training manual was intended to serve as a self-guided course to help new ADAR users become familiar with the most commonly used ADAR programs. The new

ADAR user was stepped through the use of batch command files, the common utility programs, and data dictionary determination. This manual also included instructions to help guide novice computer users in the most basic computer functions, such as viewing and editing ASCII files. Basic AEGIS concepts and other helpful background information needed for understanding the required steps to be performed during data reduction and analysis were also included in the manual.

While some new ADAR users may not have needed all the subjects covered in the Introducing ADAR manual, almost without exception they needed some basic instruction regarding AEGIS elements. Once the Introducing ADAR manual was developed, regular login accounts on the classified cluster computers were not assigned until each person could pass a verbal test covering the most important concepts in the manual. This helped to ensure computer users had become familiar with the information in Introducing ADAR before having access to data available via regular accounts on the classified cluster computers.

The ADAR group on the NSWC base produced Introducing ADAR in a PVC-bound format. As changes developed in the ADAR system, updates were made to the training manual. The ADAR group also became the ones responsible for testing the knowledge of the new users.

7.8 Problems

Early in the AEGIS program there was a great need for data analysis programs. The primary driving force behind program development in those early days was the need

to quickly produce something that worked. Little thought was given to making the programs easy to use or modify. In addition, little thought was given to creating a library of common functions beyond a few basic input and output functions.

As more programs and AEGIS baselines were developed, the maintenance of the Job Processor as well as of the various programs became increasingly difficult. Many things contributed to making the data analysis programs difficult to use and maintain. The following were some of the contributing factors:

- Many baseline-specific programs,
- Little or no coordination between element groups,
- Lack of uniformity in programming standards, and
- No standard user interface in regards to input, questions for the user, or output.

When an analysis program, written to handle the data dictionary for baseline 1.2, was rewritten to handle the changed data dictionary for another baseline, say 2.1, two similar programs existed on the system. The older program had to be kept along with the newer program. These programs probably had either the same name or names that were very similar. These two programs needed to be to kept track of and differentiated for both the programmers and the analysts. The addition of lines of code to the job processor inquiring about the baseline helped the user but not the programmer.

The next time the data dictionary changed for the applicable extraction point, say with baseline 3.2, the same thing happened. By the time baseline 5 was reached, the number of baseline-specific programs had grown quite large. These baseline-specific programs required a large amount of disk space to hold both the source code and the

executable programs. If functionality was added to one program, the corresponding programs for the other baselines might also need to be modified. This was another source of duplication of effort. Keeping the different versions straight also became a configuration problem. In order to use the program version for the correct baseline when processing data, the user had to know exactly which baseline was used during the recording of the data. While not generally a problem, this was just one more detail of which the users had to keep track.

As each element group at NSWC developed programs, there was little or no coordination between the groups. Consequently, there were some programs across the various groups that handled similar data. While the duplicated programs were never exactly the same, there was still a duplication of effort.

The different programmers in the element groups who had developed the data analysis programs had many different backgrounds. Additionally, no programming standards had been developed. Consequently, many programming styles and standards were represented among the data analysis programs. This lack of uniformity in programming standards made it difficult for other programmers to modify the programs.

The original ADAR programs did not have a standard way to query the user for input. This lack of uniformity with the user interface resulted in a variety of needed inputs and questions. Often the programs required something special that would not be obvious to the users. The following are some of the possible special required items:

- Special user input information items,
- A certain order for the input,

- A special input file,
- The need to run one or more programs to create intermediate results to be used as input into the final program,
- The need to work from a special directory, and
- Special print commands.

There was no way for the users to determine these special items unless messages regarding them were displayed to the users. Even when messages were displayed, they could be wrong. This might have occurred because either the message had never been right or some situation had changed without the message being updated. Frequently, the results were not what the user expected. Running the programs from the job processor reduced some of these problems. In order to do this, subroutines were written for the job processor to handle frequently desired input and output combinations. These subroutines were given names and added to the job processor. When these subroutines were run using the job processor, many program use surprises were avoided. Additionally, the user had the advantage with these subroutines of not needing to know which extraction points, the which fields and their required order, or which programs were used. On the other hand, the job processor was not always right.

Once the initial data analysis programs had been developed, some of the common user interface functions were pulled out of the programs and put onto the system in libraries of functions. While this helped to produce some uniformity for the programs regarding input and output items, it also helped to point out problem areas that could have been avoided if the process had been reversed. Some of these problems were discussed above. Additionally, there were common computations, constants, and unit

conversions used by many programs. These items had never been placed in system libraries and frequently were implemented differently from one programmer or element to another. The value of Pi and the conversion between nautical miles and feet are just two examples of these items.

Another set of problems centered on the portability of the ADAR programs. It gradually became necessary to be able to run ADAR programs on other computer platforms and at other AEGIS sites. Since the ADAR programs had been developed using the common functions, subroutines, and data dictionaries only available on VAX computers at NSWC, this was not possible with the existing setup and the lack of portability built into the programs.

Additionally, planning had already begun for baseline 6 tactical computer code. During this planning a decision was made to have the new tactical code developed using both the C and Ada programming languages. At this time, the ADAR group realized the programs that had been developed in FORTRAN would not necessarily be able to handle data that could be produced by these newer programming languages. The ADAR group and users had to adapt.

8 Migration of Tactical System Processing

Beginning with baseline 6 of the tactical code, all new code is being written in either the C or the Ada 83 programming languages. Additionally, some of the old code, as needed due to required changes involving large amounts of code, is being rewritten in these languages. However, some programming is still being done in the CMS-2 languages. The primary reason for requiring changes to the tactical code is due to changes in requirements for either the ACS or AWS. Another possible reason is errors in the code that require possibly extensive recoding to correct.

Also beginning with baseline 6, the user interface on the tactical consoles is being changed to use graphical user interfaces (GUIs), X-windows, and Motif on top of a UNIX operating system. These changes also require recoding of the tactical system code. In addition to both these changes and the usual types of changes between baselines, the data recording for the tactical system is moving from AN/UYK computers to Hewlett-Packard (HPs) workstations and other platforms.

C and Ada 83 can handle many data types and structures not available in either CMS-2Y or FORTRAN. As a result, data analysis programs that had been written using FORTRAN could not easily be adapted to handle the new data types and structures now possible. Instead, new programs were needed to be able to handle the unpacking of the

data created using these new data types and structures. Also, the changes being made to the tactical code and to the ADAR programs required changes to the data dictionary format.

When the AEGIS program began, any software development needed by a given government program was handled within that program. Since those early days, however, much commercial software has become available. Also, government programs have developed many hardware and software items. Consequently, the emphasis within the United States government has changed to encourage the use of both commercial software and government developed software rather than developing all new software for new programs. Commercially available software and hardware has come to be referred to as commercial-off-the-shelf (COTS) while government developed software and hardware is known as government-off-the-shelf (GOTS).

Additionally, there has been a push to incorporate hardware items created by outside groups when a government program has required additional functionality. The incorporation of both software and hardware from other developers into the AEGIS program has contributed to the increasing complexity of data analysis. The data dictionary structure used for the ADAR programs had to be modified to allow the inclusion of outside sources of data.

Since the ADAR group was made aware of these changes early in the development process, they were able to plan ahead. Being aware of the problems that

were created by the way things were handled previously, they decided to start over. They started by first developing an ADAR library of low-level functions using C.

Three main things contributed to the move from FORTRAN to C for ADAR programs:

1. A programming language was needed that supported complex data structures and memory allocation.

2. A programming language was needed that was portable to other platforms and locations.

3. A programming language was needed that was in the main stream of the programming industry. It is much easier to find programmers for current languages than for older languages. In addition, it is easier for other people to use material that is current.

For these same reasons, ADAR programming will be moving to the C++ programming language eventually.

9 Third stage of ADAR: Moving Away from the Job Processor

9.1 Introduction

The new ADAR C function library began with the most basic functions. The following represent some of these:

- Queue and stack processing,
- Common unit conversions,
- Variables to represent constants, and
- Reading data file headers.

As the first functions were developed and put on the classified system, additional, more complex functions that used these first functions were developed. Since then many more functions have been developed. Finally, programs were developed to use these functions. Some of the FORTRAN-based programs were updated to use these new functions. Other new programs were designed to replace or streamline old programs or methods. Additionally, since the old help files could be difficult to locate and use, improved on-line help resources were developed to replace the old help files.

9.2 Changing the Primary ADAR Programming Language

All new ADAR programs and functions are being developed using the C programming language. The language change for data analysis programs was dictated partially by the changes being made to the tactical program. Most new code in the tactical program is being written in C and Ada. These languages are capable of creating

and using complex data types and structures that the older FORTRAN-based programs cannot handle. At the time ADAR development was begun in C, it was a more capable language than was FORTRAN since it supported dynamic memory and complex data structures. Since C was one of the two new languages used for tactical code development, it seemed logical to also develop data analysis programs using C. By using the same language, any data type that can be recorded by the tactical code can be duplicated and handled by the data analysis programs. C has the added advantage that it is backward compatible with CMS-2Y-generated data. Consequently, any newly developed programs can be used with data generated by the older baseline tactical programs. Additionally, by changing to C it became easier to find programmers since C is closer to the main stream of the rest of the programming world.

Now, all of the planned C library functions have been developed. Additions and changes will be made as they are needed. Several of the new functions have been changed to incorporate new functionality as the need arose. Additionally, all the utility programs needed to replace the FORTRAN utility programs have been developed along with several new programs.

To make the library of ADAR functions more user-friendly, a user's guide to these functions was developed. The user's guide, the ADAR Library Functions manual, has been updated many times as new functions and functionalities have been added to the library. The ADAR Library Functions manual provides the following information for each function:

- The type of function, e.g., conversion, input/output, etc. it is,
- The number and type of parameters needed when the function is used,
- The meaning of each of these parameters,
- The value(s) returned when the function is used, and
- At least one example illustrating the use of the function.

9.3 Establishment of Programming Standards

At the same time the C function library was being developed, the ADAR group also set up programming standards to be used with all future program development. To guide the data analysis program developers, an ADAR Programming Standards guide also was written. The following are some of the areas covered by the new programming standards:

- All new data analysis programs are to be developed using the C programming language.

- The development of new data analysis programs is to be coordinated through the ADAR group.

- New data analysis programs will be non-baseline specific. Baseline differences will be checked and handled within each program.

- New data analysis programs will have a standard method for inputting the users' processing choices.

- A requirements document is to be developed for every new data analysis program.

- Once new data analysis programs and their accompanying requirement documents are complete, they are to be turned over to the ADAR group for maintenance.

With all of the element groups on base coordinating any new AEGIS data analysis programs with the ADAR group, the possibility of duplication of effort is reduced. The ADAR group serves as a clearinghouse for new program development.

When someone approaches the ADAR group from any of the element groups regarding a possible new data analysis program, the members of the ADAR group can do several things:

- Inform the person from the element group about another similar program that is being developed, if that is happening,
- Discuss with that person how an existing program can be modified to provide the needed ability, or
- Provide guidance to that person during the development of the new program and any associated documentation.

The same thing can happen if anyone wants to be able to examine data generated by COTS or GOTS software that has been incorporated into the AEGIS system. This generally requires the creation of a new data dictionary to handle the new data. The ADAR group will advise and help check the accuracy of the data dictionary being generated. Several of the most general of the ADAR programs can be used for data analysis once the new data dictionary has been developed. Even if only one extraction point has been put into the new data dictionary, it can be used by several of the general data extraction programs. This is because the user can specify which extraction point(s) is (are) to be examined when running many of the ADAR programs. This permits the analysts to use familiar programs while analyzing data. Additionally, the users are not dependent on whatever data analysis programs may have been provided by the originators of the non-AEGIS software.

Each new data analysis program contains code to provide for differences in the data dictionaries between the various baselines. Since the new function that opens the

data file also returns values for the items written in the file header, the program can use this header information. With this information available to the program, the code can test for the DRIST value that was written at the beginning of the data file. This information provides a means to determine the appropriate data dictionary to be referenced without requiring input from the user. The data dictionary value then can be used to determine any baseline appropriate processing. This portability across the baselines makes using the various ADAR programs easier for the users. With baseline differences handled internally to the programs, the user does not have to remember which version of a program to use. A major disadvantage to making the various programs handle multiple baselines is the extra programming, design and regression testing needed to obtain a single dependable working program. Additionally, a multiple-baseline program is much more complex than a single-baseline program.

While the new ADAR programs may be run via the job processor, they are very easy to run from the command line. All new programs must use standard expressions as input to specify the processing choices to be performed by the program. These standard expressions are called "qualifiers." While not all programs use all possible qualifiers, where a particular qualifier is applicable, its use is required for input into the program versus some other method. On the VAX computers, all qualifiers begin with a "/," while on HP computers running the UNIX operating system and on PCs, a "-" is used. The following are some examples of these qualifiers using the notation applicable for VAX computers:

- /EP - This qualifier permits the user to select one or more extraction points to be used by the program. While many programs require this qualifier, there is no default value provided.

- /field - This qualifier permits the user to specify which field(s) in the extraction point(s) are to be used. All specified fields must be in at least one of the selected extraction points. The default value for this qualifier is "all."

- /POC - This qualifier permits the user to specify fields that are to be printed out only when their value changes. The default value for this qualifier is "all."

- /time - The default time range is the beginning-of-the-tape (BOT) to the end-of-the-tape (EOT). This qualifier permits the user to select a shorter time range.

- /format - This qualifier permits the user to choose the format of the output. /format = vertical produces data in columns (see Figure 7-3). /format=namelist produces data in the "field-name = field-value" format (see Figure 7-1). /format=export produces comma delimited data that can be imported into a spreadsheet (see Figure 9-2 below.)

- /select - This qualifier permits the use of logical expressions to filter the data. There is no default value for this qualifier.

- /options - This qualifier allows the user to write all of the other qualifiers and their associated values in a file. This qualifier permits short entries on the command line along with reusable qualifier lists. There is no default for this qualifier.

An options file provides an easy means for the user to determine what command was used when unexpected results are produced. By having the option, via the /options qualifier, to store the qualifiers and their values in a file, the user can reuse the same set of qualifiers and values. The file also can be changed easily to correct mistakes or change values before being reused.

One of the new ADAR programs is Data Extraction Point (DXEP). While more will be said later in this paper about this program, Figure 9-1 illustrates the use of qualifiers with the DXEP program.

```
dxep/ep=360/field =measx,measy,measz/format=export z905
```

Figure 9-1: Example of Qualifier Use

The line in Figure 9-1 can be read as the following: using the DXEP program, examine the data found in the file created from tape z905. Print out, in comma delimited format (/format=export), the values for the measx, measy, and measz fields (/field=measx,measy,measz) in each instance of extraction point 360 (/ep=360). The output generated by the above command would look like that shown in Figure 9-2.

```
24:15:18.396(87318.396)  360,240.59462,189.00923,20.12348
24:15:18.697(87318.697)  360,241.49012,180.89293,19.78561
. . .
```

Figure 9-2: Example of Comma Delimited Output

If the qualifiers used in Figure 9-1 are put into a file, perhaps called z905_360.opt, the file contents would look like the contents of Figure 9-3. Each qualifier is put on a separate line. The default extension for options files is .opt.

```
/ep=360
/field=measx,measy,measz
/format=export
```

Figure 9-3: Qualifier File Example

The new DXEP call line referencing the file illustrated in Figure 9-3 is illustrated in Figure 9-4.

```
dxep/options=z905_360  z905
```

Figure 9-4: Example of /options File Use

The requirements document that is to be developed for each new data analysis program includes information regarding expected input, processing that occurs within the program, and sample output. Additionally, the requirements document includes any other information that might be needed by a user such as the applicable element, extraction point(s), and fields(s). Generally, data analysis program development is performed on the classified cluster computers. Even if not all of the development is handled there, the development must be finalized on that system to ensure that it will perform as expected. Consequently, when the data analysis program is turned over to the ADAR group, the program is already on the classified cluster computers. The act of turning over a program to the ADAR group consists of giving them the file name and the directory where it is located. At the same time, both hard and soft copies of the requirement document for the program also are turned over.

There are several advantages to the ADAR group being responsible for maintaining all of the ADAR programs. The various AEGIS element groups can concentrate on developing new data analysis programs without having to worry about redoing older programs to accommodate changes to either ADAR functions or AEGIS baselines. When something comes up that requires some change to the existing ADAR programs, the ADAR group can systematically change all affected programs without bothering the programmers or users in the element groups. This helps to make changes to both the ADAR system and programs transparent to the other programmers as well as to the users.

Once the ADAR group has control over a program and its documentation, they are able to provide information to the users regarding the program. This information includes the following: the existence of the program, what the program is called, what the program does, and how to use the program. The newest way for the ADAR group to let the users know this information is via a web Home Page on the classified cluster computers and workstations. This will be discussed later.

As the ADAR group has developed functions and programs using C, they have followed their own guidelines. As they did so, incomplete and confusing sections in the programming standards guide were found and corrected. After a basic core of new functions had been developed, the ADAR group started rewriting the older utility programs using C. Additionally, new programs were developed to replace some of the

older programs. Since then all of the older utility programs have been replaced by C versions. Additionally, most of the element programs have been rewritten in C.

9.4 Establishment of an ADAR Home Page

Originally, the only interface the classified cluster computer users had with the data analysis computers were dumb terminals that were only capable of displaying lines of text. While there were help files, they were difficult to use. Later, a few terminals became available for handling the graphics required by plot programs. However, these terminals were not capable of handling windowing environments. It was not until about 1995 that any X-windowing terminals became available. Now most of the available terminals are capable of handling X-windows.

With the availability of X-windows, the ADAR group was able to develop a web Home Page that could be accessed via a web browser such as Netscape. When a user goes to the ADAR Home Page, on the classified cluster terminals, he is able to find out all he could possibly want to know about data analysis programs. The following are some of the items found on this Home Page:

- Information about all ADAR functions and programs, such as requirements documents, source code, sample output, and reported problems;
- ASCII data dictionary access;
- The ability to e-mail comments, questions, or problems to the ADAR group;
- Information, including documentation, regarding the ADAR software development process; and
- Help files on other ADAR related items.

9.5 Old Programs and Changes

During the transition to C developed programs, the format used for the data dictionaries was modified both to contain more information and to make the data field specifications orthogonal. A new line was added to the header section of each dictionary definition. This line lists the type of computer that records the given extraction point using the format provided. Not only can the data dictionaries still have multiple tables, the new format can handle nested tables, long field names, long tables names, arrays of tables and fields, IEEE floating point numbers, enumerators, big and little Endian order, both 16 and 32 bit word sizes, and other items.

For extraction points that contained more than one table, another field is applied to the EP line to indicate the level of the corresponding table. By having this second field, while the extraction point number is the same, the multiple parts are distinguishable by the level number.

The new format for the FIELD lines within a TABLE declaration has both changed the order of the contained fields and required all contained fields to have a value. Figure 9-5 illustrates the new general format.

FIELD: <field name>,<data type>,<field sign>,<starting bit>,
<word number>,<number of fractional bits>,<number of bits>

Figure 9-5: New General FIELD Format Used in Data Dictionaries

Figure 9-6, modified from the example shown in Figure 4-2, shows the new format with the new lines indicated by bold Italics. The first line defines the DRIST number, i.e., M00122J5. The second line contains the keyword "TYPE" followed by the computer platform that records the following extraction points in the given formats. This field is needed since the same extraction points can be recorded from more than one computer platform, and not all platforms use the same format.

The third line, the first line of the extraction point definitions section, contains the extraction point number, i.e., 11. The fourth line contains the keyword "TABLE" followed by the name of the table, TAB1; the number of computer words used, 3; and the number of items in the extraction point, 1. The next several lines, beginning with the keyword "FIELD," contain field definitions.

Commas separate all fields description values. Contained fields that previously may have been blank now contain a value. For example, the data types that do not have fractional bits have a "0" in the fractional bits field. Since all tables are recorded in a vertical format, there is no need to indicate this information.

The rewritten programs that are used to manipulate the data dictionaries are EPAC, DD, and DC. The EPAC program is used to translate dictionaries from one format to another. Two possibilities are from binary to ASCII format or vice versa. The DD, for Display Dictionary, program is used to view dictionaries and perform various types of searches on the dictionaries. The DC, for Dictionary Compare, program is used

```
DICT: M00122J5
TYPE: UYK07
  EP: M,11,1
    TABLE: TAB1,3,1
COM: TAB1 IS A 3 WORD TABLE WITH ONE ITEM.
      FIELD: T1IAA, I,U,31,0,0,8
      FIELD: T1IBB,I,U,23,0,0,8
      FIELD: T1ICC,I,S,15,0,0,16
      FIELD: T1IDD,I,U,31,1,0,16
      FIELD: T1BAA,B,S,15,1,0,1
      FIELD: T1BBB,B,U,14,1,0,1
      FIELD: T1IEE,I,U,13,1,0,14
      FIELD: T1DAE,A,S,31,2,14,32
  EP: M,11,2
    TABLE: TAB2, 6,1
COM: TAB2 IS A 6 WORD TABLE.
      FIELD: T2IAA,I,U,31,0,0,16
      FIELD: T2IBB,I,S,31,1,0,32
      FIELD: T2BAA,B,U,7,2,0,1
      FIELD: T2BBB,B,U,6,2,0,1
      FIELD: T2AAX,A,S,31,3,16,32
      FIELD: T2AAY,A,S,31,4,16,31
      FIELD: T2AAZ,A,S,31,5,16,32
```

Figure 9-6: Data Dictionary Example in New Format

to compare data dictionaries from a whole dictionary down to a single field within a dictionary.

The rewritten programs used to manipulate recorded data tapes and files are DXMov, DXSum, and DXDump. The DXMov program translates data from one format to another. This format change could be from one computer format to another, e.g., AN/UYK-43 format to VAX format, or from the file format produced for one purpose into the file format required for another purpose. For instance, data files, produced at

Combat Systems Ship Qualification Trials (CSSQTs) in .tsf format must be converted to .ats format for processing by ADAR programs. The moving of the data from the recorded data tape to computer disk files on the classified cluster computer system includes any needed changing of computer formats.

The DXSum program creates a summary of the extraction points in the data file. This summary is useful when attempting to determine what extraction points were recorded, the number of times each was recorded, and the time span of the recording.

The DXDump program creates a dump file of the contents of a raw data file in octal, decimal, or hexadecimal format. This dump file is useful when debugging various analysis programs or developing data dictionaries for non-AEGIS data.

A new program, called DXEP, that requires only one step to obtain useful information was developed, in part, to replace the functionality provided by the GenFLD, GenMEP, and GenPOC programs. As a result the general purpose GenFLD, GenMEP, and GenPOC programs are being neither rewritten nor maintained. These programs, as discussed in chapter 7, required many steps to obtain useful information. Since most of their functionalities have been duplicated within the new DXEP program, the old programs no longer are needed. Additionally, the POC and other old ADAR job programs are not being maintained.

While these programs are not being maintained, they will remain available on the various platforms supported by the classified cluster computers. When any compiler problems arise with these programs, they will be removed from the applicable platform.

When the EPAC program was rewritten in C, the ability to create the data dictionaries needed to support the GenFLD program was not put into it. As discussed in section 7.3, the EPAC program generates the ASCII and binary data dictionaries that are available on the classified cluster computers.

9.6 New Program Development

When the ADAR group started planning for the coming changes, they had a major advantage over the original ADAR developers. First, they had a good idea of the functionalities that ADAR programs needed to support. Additionally, they had a good idea of the analysts' needs as well as what output formats had proved to be useful. Some of these items are reading the header messages found in the data fields, using this information to locate the applicable data dictionary, and outputting selected data from the data files in desired formats. From their knowledge, they were able to develop some general-purpose programs to replace the most commonly used older programs. While developing these programs, they were able to reduce the amount of work required by the users as well as providing for more versatility from the program results.

The work horse, and most general, of these programs is DXEP. The original DXEP was introduced in 1995. Since that time it has continued to be enhanced so that has become a very flexible and versatile program. DXEP uses an extensive set of qualifiers to provide the user with a wide variety of options. Some of the qualifiers, discussed previously in Section 9.3, need to be discussed further with regard to DXEP. When DXEP is used with the /POC qualifier, the output results are much the same as

would be produced by a program generated by the GenPOC program. When DXEP is used with the /format = vertical qualifier, the output is in column format similar to that produced by a program generated by the GenPOC program. When DXEP is used with the /format=namelist qualifier, the output is similar in appearance to that which would be produced by a program generated by the GenMEP program. When DXEP is used with the /format=export qualifier, comma delimited data is produced that can be imported into a spreadsheet for examination.

9.7 New Extraction Point Types

The CND and SPY elements have several extraction points that are frequently used together. Some of them are the same extraction points that were reorganized in the changes from baseline 3 to baseline 4. While some useful information may be determined from each of these extraction points individually, they must be correlated before most information can be determined. While this correlation was coded into several programs originally produced by the GenMEP program, there was no easy way to incorporate this ability into other programs without recoding for the specific situation.

The ADAR group was aware of the need for these correlations. Consequently, using the correlation that had been built into a GenMEP produced program as a base, an ADAR library function was developed in C with the ability to correlate the data in two extraction points. First this function was extended to handle all baselines. Later, correlation with another extraction point was added to the first correlation. After the correlation for the first set of extraction points had been completed and put on the

classified cluster system, another set of extraction point correlations, for another element, was developed. Now ADAR users can access these extraction points, and take advantage of the correlation computations, by specifying the qualifier /EP=CDTRK for the CND extraction points or /EP=SPYTRK for the SPY extraction points.

10 Expanding the ADAR Environment

With the explosion in the variety of computer platforms being used by the various ADAR users, a need has arisen to make the data analysis programs portable across many different platforms. Additionally, data recording for the tactical system is expanding from only AN/UYK computers to incorporate other platforms such as HP running their UNIX operating system (HPUX) and Personal Computers (PCs) with baselines 6 and 7. For instance, the ACTS element is being rehosted to Sun workstations using both the C and Ada-83 programming languages. Data reduction for the rehosted ACTS can be performed on the same Sun workstations. Also, other elements are being incorporated into the ACS that use non-ATES formats for recording their data.

Two of the new formats being used are Data Extraction and Recording (DXR) and C2P. DXR-formatted data is recorded from COTS systems using a set of Application Program Interface (API) functions. These API functions have been written in C. C2P-formatted data is recorded from a C2P system. With the analysis programs being made portable to other computer platforms, the users, if desired, will be able to analyze data on the same platform that was used to record the data.

While ADAR programs are still available on VAX computers using the VMS operating system, other computer platforms are beginning to be used for data analysis.

There are a variety of computers using their own version of the UNIX operating system being used for data analysis. Some of these are the following: SUN computers using the Solaris operating system, HPUX, SGI computers using the IRIX operating system, and Alpha computers using Digital UNIX. Additionally, Alpha computers using the VMS operating system and PC computers using either the Windows or NT operating systems are being used.

The explosion in the use of these various types of computer platforms is primarily due to the developments that have occurred in recent years. Some of these developments include the expansion of both the disk space and the memory available for non-mainframe computers. With this expansion, it is now possible to fit the data from at least twenty recorded data tapes on the disk drive of a desktop computer.

The AEGIS Classified Support System (ACSS) program has been developed to allow PCs in secure offices to be networked with the classified cluster VAX computers. In addition to the advantage to users of being able to work from their own offices, fewer common use terminals are needed in the ACC. Needing fewer terminals in the ACC has its own advantages such as taking up less space for data analysis and opening up more space for other uses.

Even though several different computer platforms can be networked to the classified VAX computer, at the present time this networking only allows the users to copy files from one platform to another. A copy of the ADAR function library, the needed data dictionaries, the data files, and program files must appear to be resident on

the same computer before they can be used together. This is generally accomplished via network file system (NFS) mounting. Therefore, although the computers are networked together, ADAR data analysis programs still work in a stand-alone mode.

Making the ADAR data analysis programs portable across different platforms makes the programs usable at other AEGIS sites. Testers, observers, and analysts travel to other AEGIS sites and need to be able to examine data while there. Since the users find familiar programs much easier to use than unfamiliar programs, there has been a great demand in the past for the ADAR data analysis programs to be available at the other sites. In addition to the output from unfamiliar programs being difficult to use due to different data layouts and contents, the output generated by these programs may not contain the information that a user needs.

In order to port ADAR programs to other sites, special programs have been developed that create executable versions of programs for the other sites. This was necessary since the computer configurations available at the other sites are generally not the same as in the ACC. At first, either someone from the ADAR group or users from the element groups would take executable versions of the ADAR programs to the site. Once there, either the person from the ADAR group or a systems person already at the site would install the programs. Now it is possible to use FTP to move the executables to the various sites. This makes it easier for the programs at the other sites to be kept up to date when changes are made. In addition to programs, the same situation applies to maintaining the ADAR function files and data dictionary files.

11 The Future

The ADAR group is continuing to convert the remaining element programs to C. While this process has been going on for about four years now, it will probably continue for several more years.

The ADAR group is working on having the different computer platforms networked together with a single ADAR in a client-server arrangement. That is, the ADAR job processor, programs, data dictionaries, and function library will be on only one computer, the server. The users will be able to access the ADAR system from the computer platforms of their choice, the clients. No matter which platform they may be working from, they will be accessing the ADAR system on the server where it resides. The platform differences will be invisible to the users. When a user wants to run a program, it will appear to run on his "home" platform. However, the ADAR system will determine which computer platform will be used based, in part, on the processing speed of the platforms as well as the availability of the platforms. As the need arises the ADAR programs may be moved to other, probably faster, platforms. When the ADAR system files are moved to another platform, the processing will occur on the new platform but the move will be transparent to the users.

Currently, the server platform is an HP computer. The server platform can be whichever computer platform the ADAR group might want. The computer platform in use at a particular time will probably be the fastest available.

There are two ways of achieving this client-server ability: write one's own code using sockets and API functions or use existing web servers and browsers. Since API functions are different from platform to platform, the disadvantage of this method is obvious. Additionally, it can be difficult to get programs to be portable to all platforms when writing one's own code. Not having to learn the API functions is the only real advantage of writing one's own code. Since web servers and browsers were not meant to run programs, the ADAR group is not sure that using the web servers and browsers in this way will work. If it does work, web servers and browsers may prove to be too slow to be useful. On the other hand, there are many advantages to using existing web servers and browsers. Three of the primary advantages are listed here. First, they are free, relatively, and can be used without needing to write new programs to perform their functions. Second, they are portable across computer platforms. Third, web servers and browsers are becoming increasing familiar environments to users. After weighing the advantages and disadvantages of each, the ADAR group has decided to use web servers and browsers, especially Netscape using HTML format, to handle this networking.

A major aspect of the networked ADAR system will be a new job processor. It will be a windowing version of the old job processor. This new job processor, called the ADAR Task Manager (TM), is currently under development by the ADAR group. Once

finished, when the ADAR Task Manager is run, the user interface will display file tabs from which the user will choose the desired program area to be used, e.g., utility programs, CND programs, etc. Once the user chooses the desired program area, the applicable programs for the indicated area will be displayed. When the user makes the program choice, the displayed window will have blanks to be filled in as applicable for the needed input to the chosen program. One set of choices will involve picking whether to run the program interactively or in the background, i.e., batch mode. Once all of the required blanks have been filled in, the user will select the "Submit" button to run the program.

The new job processor will permit access to the new graphical programs that are being developed. The users will able to create plots, i.e., charts, comparing the values in the user's choice of fields when these programs are used. While these graphical programs do not currently have a standardized development process, the developers are still able to draw on the standards that have been developed for the non-graphical programs. The graphical programs are also being developed to be portable to other computer platforms though not all platforms are able to handle the graphics involved. Some of the platforms to be used are X-windows on VAX, SGI, HP, and Alpha platforms as well as PCs using the Window operating system.

Most users are resistant to changing from the familiar VAX ADAR environment. This is the primary reason for making the server computer platform transparent to the

users. Rather than forcing the users to change, the ADAR group is just going to accommodate them.

In order to improve the installation process at other sites, the ADAR group has recently started developing an install program. This program will eventually be able to install programs at all sites on all major platforms. Even when the program can handle one platform type at one site, much work will remain to finish this task.

The ADAR group is planning several additional things to enhance the many ADAR programs. One possibility will effect how the output is displayed on the terminal. Currently, when the output is displayed to the terminal, all lines scroll up and off the screen. After the heading has disappeared, the user can have difficulty interpreting the displayed data when the scrolling is paused. One possible enhancement to help this situation will keep the headers at the top of the screen but let the data lines keep scrolling up and off the screen.

12 Conclusion

The ability to run the ADAR programs from the command line will not go away. Many users prefer this method and do not wish to change. The old, non-supported programs such as GenMEP and GenFLD will not disappear anytime soon. They just will not work with the new data formats. Additionally, there are compiler issues when they are ported to some computer platforms. For instance, they cannot be used on Alpha workstations. Since GenMEP and GenFLD are not being supported, the compiler issues will not be addressed. This situation was handled by removing these programs from the Alpha workstations. As the ADAR personnel change and new people start working with the ADAR system, they are only exposed to the new programs. Eventually the old programs will become outdated and will die.

The ADAR system will continue to be enhanced well into the future. There are several hundred people working with the AEGIS program at NSWCDD. There are many more people working with the AEGIS program both at other sites and onboard the AEGIS ships. Since new ships are still being commissioned, the AEGIS program is expected to continue well into the 21st century. With the life expectancy of the ships being around 50 years, data analysis will be needed for the AEGIS system for many more years.

When the ADAR system first began, the computer industry was very immature compared to what it is today. The user base frequently had limited exposure to any computers and data analysis programs. In addition to training the users to be able to use the needed computers, useful programs had to be developed for the user community. Despite the handicaps, the people working with the ADAR system still managed to produce a variety of useful programs.

While it would have been better for the original ADAR group to have done some things differently, they did the best that probably could have been done with the constraints put upon them. It would have been better to develop programs that were not baseline specific as well as those that did the whole job at once rather than piecemeal. At the time no one realized that so many baselines would be developed. As a result no one could have anticipated the configuration management problems that could arise. They were under pressure to produce useful programs quickly, which they were able to do. Additionally, they had to work within the knowledge, people, and technology base available to them.

As the AEGIS program has evolved, the ADAR group and system have also evolved and adapted. From the early, difficult beginnings to the present, the ADAR system has grown into a set of highly sophisticated data analysis tools that are flexible enough to be used at many sites, on many platforms, to produce a variety of outputs while at the same time being relatively easy to use. If the original ADAR group members had made different choices, either some of the growing pains could have been

avoided or there could have been more growing pains. The pioneers in any field need to be admired when they are able to overcome obstacles to achieve their goals. This certainly holds true for the pioneer ADAR developers and users.

The current ADAR developers also must be commended for their vision for the future. New technology and needs have provided challenges that the ADAR developers have tackled. They have been successful, thus far, in providing improved programs and user interfaces as well as enhancing the capabilities of the ADAR system. Thanks to their efforts, the users are able to use the ADAR programs at several sites with more ease to provide quicker answers than ever before.

Bibliography

Bibliography

Augarten, Stan. Bit by Bit. Ticknor and Fields, New York: 1984.

Bunch, Bryan, and Alexander Hellemans. The Timetables of Technology: A Chronology of the Most Important People and Events in the History of Technology. Simon and Schuster, New York: 1994.

Campbell, George P. AEGIS Data Reduction System (ADAR) User's Guide (Preliminary), January 12, 1978. RCA Missile and Surface Radar Division, Moorestown, NJ: 1978.

Canup, Steven. Conversations. ADAR group, NSWCDD, VA: 1995-1998.

CMS-2Y Programming: A Self-Instructional Manual. Fleet Combat Direction Systems Support Activity, San Diego: 1981.

Combat Systems Department. AEGIS Data Reduction System (ADAR) User's Guide, July 1985. Naval Surface Weapons Center, Dahlgren, VA: 1985.

Computer Program Performance Document for the AEGIS Data Analysis and Reduction (ADAR) Sequential Processor (ASP). Science Applications International Corporation, Dahlgren, VA: 1989.

Orientation Handbook of Personnel Qualification Program (PQP). AEGIS Shipbuilding Project: 1983.

N23/ADAR group. Introducing ADAR. NSWCDD, Dahlgren, VA: 1993.

N86/ADAR group. Introducing ADAR. NSWCDD, Dahlgren, VA: 1995.

N86/ADAR group. ADAR Library Functions. NSWCDD, Dahlgren, VA: 1996.

N86/ADAR group. ADAR Programming Standards. NSWCDD, Dahlgren, VA: 1996.

N86/ADAR group. ADAR Requirements Document for ADAR Task Manager, Version 1.0. NSWCDD, Dahlgren, VA: 1997

N86/ADAR group. ADAR User's Manual: AEGIS Display System (ADS) Programs. NSWCDD, Dahlgren, VA: 1998.

N86/ADAR group. ADAR User's Manual: Command and Decision (CND) Programs. NSWCDD, Dahlgren, VA: 1998.

N86/ADAR group. ADAR User's Manual: Introduction. NSWCDD, Dahlgren, VA: 1997.

N86/ADAR group. ADAR User's Manual: General Utilities. NSWCDD, Dahlgren, VA: 1997.

N86/ADAR group. ADAR User's Manual: SPY Programs. NSWCDD, Dahlgren, VA: 1998.

N86/ADAR group. ADAR User's Manual: Weapon Control System (WCS) Programs. NSWCDD, Dahlgren, VA: 1998.

Naval Sea Systems Command. Condition of AEGIS, March 1975. Naval Sea Systems Command: 1975.

Pollack, Kay. Conversations. ADAR Group (retired), NSWCDD, VA: 1997.

RCA Corp. RCA Proposal for Inclusion of the AN/UYK-43 Computers on the Lead DDG Ship. RCA Corp: 1982.

Smith, Pearl. Conversations. ADAR group (retired), NSWCDD, VA: 1997.

Sperry Univac Corp. AN/UYK-7 Technical Description. Sperry Univac Corp.: 1980.

US Navy. Pictures of the AN/UYK-7, AN/UYK-20, AN/UYK-43, and AN/UYK-44. Available via the internet at www.m42.crane.navy.mil/706/7063.

US Navy. Pictures of the *USS Normandy* and *USS Ramage*. Available via the internet at www.milnet.com/milnet/index.html.

Appendices

Appendix A Timelines Relating AEGIS Development and the Computer Industry

A.1 AEGIS Highlights

Pre-1960 - Developments preceding AEGIS program

- In the late 1940s radar, a critical component of AEGIS in its Anti-Air Warfare (AAW) role, was being developed.

- In 1945 the first flight of a supersonic ramjet missile was made.

- In 1947 successful control of a supersonic missile was gained.

- In 1948 a supersonic missile was made to ride a radar beam.

- In the 1950s the first higher-level computer languages for the United States (US) military were developed. The CS-1 programming language was developed for use on the CP-642/USQ-20 family of second-generation computers.

1960 through 1969 - Decisions made that resulted in the development of the AEGIS program

- In 1963 the Department of Defense directed the Navy to formulate an AAW missile system as a replacement for the 3-T AAW systems.

- In 1965 the Navy and contractors assessed the 28 system concepts with various system characteristics that had been proposed by seven different contractors for the AEGIS program. From these proposals they defined an optimum system then forwarded the report to the Chief of Naval Operations (CNO).

- In 1967 the Army/Navy Missile System Commonality Study concluded that unique systems were required for AEGIS. The Office of the Secretary of Defense (SECDEF) agreed.

- In 1968 the AEGIS Development Concept Paper was approved for Contract Definition. Three prime contractors were selected for the Phase B Contract Definition of the AEGIS program.

- In 1969 the AEGIS Engineering Development Contract was awarded to the Radio Corporation of America (RCA) on December 23rd.

1970 through 1979 - The beginnings of the AEGIS program

1970 - The AEGIS Preliminary Design Stage Completed

The RCA management of AEGIS was assigned to the AEGIS Program Office. The AEGIS Program Office was established at the Missile and Surface Radar Division of RCA in Moorestown, NJ. Other RCA locations were subcontracted as well as other contractors and vendors. The Preliminary Design Review (Milestone A) was completed towards the end of the year. The first AN/UYK-7 mainframe computers become operational and were delivered for the Command Control (CC) Mark (Mk) 130. The planned AEGIS Combat System elements at this time were the Radar System AN/SPY-1 (SPY), CC Mark 130, Operational Readiness Test System (ORTS), and a combined Fire Control System Mark 99 and Weapon Direction System Mark 12 (FCS/WDS). The AEGIS Weapon System (AWS) was planned to be the anti-air weapon system for the nuclear-powered destroyer (DLGN)-38 ship class. The STANDARD MISSILE (SM)-2 missile was to be launched by the Guided Missile Launching System (GMLS). The tactical system code development was begun using the CMS-2M programming language.

1971 - The AEGIS Critical Design Stage Began

The first modified AN/UYA-4 operator display consoles were delivered. Also, launchers, missiles, and other additional necessary equipment items were either in final tests or being delivered. The first launch of the SM-2 missile occurred at the end of the year. Many of these equipment items were modified from either the Navy's current store or from equipment originally intended for other developmental programs. The newly established program generation centers began receiving equipment. An intensive effort was being made at interface management between the elements. Work had started on the executive program to be run on an AN/UYK-7 computer. The first training of sailors was begun this year at contractor sites. Congress reduced the DLGN-38 ship class on which the AWS was to have been the anti-air weapon system from 23 ships to only five ships. The DLGN-38 ships were too few, and they would be ready too soon for

the AWS. The DLGN-38 ship class later became the Virginia nuclear-powered cruiser (CGN)-38 ship class.

1972 - AEGIS Fabrication Began

In early 1972 after a successful Critical Design Review (Milestone B), assembly of the system and segment level testing began. The system being assembled incorporated most of the systems modified or manufactured for the AEGIS program. The modified systems had been developed for other programs and required some modifications in order to be used by the AEGIS program. On the other hand, new systems were developed specifically for the AEGIS program. The AEGIS Tactical Executive Program (ATEP) was delivered and loaded in the AN/UYK-7 computers at program generation centers (PGCs). Four AN/UYK-7 computers were able to communicate at the PGCs. This demonstrated that SPY, the FCS/WDS, and the CC computer programs, each housed on separate computers, along with the ATEP, could be integrated to enable inter-computer communication. The ORTS was integrated with the other elements. These integrated elements formed the First Engineering Development Model (EDM-1). The *USS Norton Sound* was converted to house the AWS for test and development. The GMLS was installed on the *USS Norton Sound*. The CNO authorized a gas turbine-powered (DG) class to be designed specifically for the AWS.

1973 - AEGIS Ashore Tests Began

During the first part of 1972, the integration of missile and launching system simulators with the rest of the weapon system completed the federation forming a complete weapon system. This integration occurred at the Land Based Test Site (LBTS) on the East Coast. In the early spring the AN/SPY-1 radar began tracking live targets. Link-11 transmitted AEGIS data to Fleet Combat Direction Systems Support Activity (FCDSSA), in Dam Neck, Virginia, then to the *USS Farragut* operating in the Atlantic Ocean off the New Jersey coast. During the Operational Readiness Exercise (OPREDEX) 2-74, data on system availability was produced during simulated combat conditions. After the successful Land Based Test Completion (Milestone C), the AEGIS first EDM-1 equipment items were flown to Long Beach to be installed in the *USS Norton Sound*. Sea tests were conducted with the GMLS to validate launcher/fire-control system interfaces. The Navy and RCA worked to formulate requirements for the AEGIS Combat System (ACS). The Secretary of the Navy (SECNAV) directed that both a gas turbine (DG) ship and a nuclear-powered (DLGN) ship be considered for AEGIS ships.

1974 - AEGIS Sea Tests Begin

The AN/SPY-1 radar was installed in the *USS Norton Sound*. The AN/SPY-1 radar started tracking in the Pacific in March, only four months after being moved from the East Coast. The first live ship firings occurred in May when the AWS (EDM-1) on the *USS Norton Sound* automatically detected, transitioned to track, tracked, engaged, and intercepted drone aircraft targets, physically impacting and destroying the second target. (Note: Interception occurs when the interceptor comes within a predetermined distance of the target. Interception does not mean the interceptor actually impacts the target.) After the successful tests during June, the AEGIS Executive Council decided that Milestone D had been achieved. In December the AN/SPY-1 radar displayed its ability to provide the necessary data to control, track, and evaluate critical fleet exercises while surveying the entire exercise. Both the DG and the new DLGN ships under consideration for AEGIS ships were canceled. The SECDEF and the Defense System Acquisition Review Council (DSARC) directed that strike cruisers (CSG) be considered for future AEGIS ships.

1975 - AEGIS Sea Tests Continued

During 1975 the AWS Mk 7 (EDM-1) on the *USS Norton Sound* continued to intercept or destroy every type of active airborne target tested. During February the AWS Mk7, on the *USS Norton Sound*, destroyed a moving target using a single missile. From the detection of the target to the interception and killing of the target, this operation only required human intervention to close the firing key. Milestones E and F were achieved this year. Early in the year the DSARC considered the *USS Long Beach* (CGN-9) for future AEGIS ships. Later in the year the SECDEF directed a gas turbine-powered guided missile destroyer (DDG-47) and a nuclear-powered strike cruiser (CSGN) to be developed to house the AEGIS Combat System. Training materials for both instructors and operators continued to be developed. On-the-job training was begun on the *USS Norton Sound*.

1976 - Sea Tests and Combat System Development Continued

During 1976 the DDG-47 system studies were performed with a scale model being built. Training and testing continued both at sea and ashore. Congress rejected the CSGN and put the previously appropriated money towards putting the AWS in the CGN-9. This money was later turned back to Congress.

1977 - Combat System Development and Ship Design Continued

PMS-400 was established to be the AEGIS Program Office. The AEGIS Program Office was given the responsibility for AEGIS ships from design, development, engineering, and acquisition throughout a ship's lifetime. The preliminary design of the DDG-47 was completed. Congress authorized the first AEGIS destroyer to be built on a SPRUANCE class destroyer (DD-963) hull. The site of the former Missile and Surface Radar Division of RCA in Moorestown, NJ, which had been turned over to the Navy in 1970, was commissioned as the Combat System Engineering Development (CSED) EDM-3 site (CSEDS). In addition to engineering development work for baseline upgrades, the CSED site was intended to provide training for navy and civilian personnel.

1978 - DDG-47, the First AEGIS Destroyer

Contracts were awarded for the production of both the AWS and for the first AEGIS warship. The production contract was awarded to RCA by the US Navy for the first AWS to be put on an AEGIS warship. At this time the plan was for the first AEGIS warship, designated DDG-47, to be a Guided Missile Destroyer. Later in the year the US Navy signed a Detail Design and Construction contract for the DDG-47 with the Ingalls Shipbuilding Division. Plans for a nuclear cruiser, CGN-42, design was shelved. The AEGIS team at the CSEDS completed the AEGIS Intermediate Milestone I in the fall.

1979 - DDG 47 Fabrication Started

Construction was begun on the AEGIS Computer Center (ACC) at the Naval Surface Weapons Center (NSWC) in Dahlgren, Virginia. The EDM-1/Vertical Launch System Integration Program was begun. Construction was begun on the AEGIS Production Test Center (PTC) in Moorestown, NJ. The US Navy operated the AWS during Operational Test III B. The *USS Norton Sound* at Long Beach Naval Shipyard was modified to accept the mechanism being developed for the vertical launch system (VLS). The fabrication of the DDG-47 was begun with the cutting of steel at the Ingalls Shipbuilding Division. The designation of DDG-47 for Guided Missile Destroyer was changed to CG-47 for Guided Cruiser.

1980 through 1989 - The first AEGIS Cruisers and Destroyers Built

1980 - CG 47 *Ticonderoga* Integration Began

The first AEGIS equipment was delivered to the PTC for acceptance testing. The CG-47 was named *Ticonderoga*, and her keel was laid. Since the CG-47 was the

first AEGIS class guided cruiser, the CG-47 ship class was name the Ticonderoga class. The AWS and ACS equipment and computer programs started arriving at the PTC to prepare for their testing phase. The AN/SPY-1A radar arrays were accepted. The US Navy awarded the contract for construction of the second AEGIS warship, CG-48, to Ingalls Shipbuilding Division. Combat system integration continued at CSEDS. The PTC-installed AN/SPY-1A tracked real targets. The VLS computer was installed in the *USS Norton Sound*. The US Navy signed the contract with RCA to furnish the AWS for the CG-48. Planning continued to start the volume production of ACS specific items with CG-48 followed by CG-49 and CG-50. Integration of the hull modules began at the shipyard.

1981 - AEGIS installed in *Ticonderoga*

The first SM-2 was successfully flown at the White Sands Missile Range in the Vertical Launching Program. CG-47, *USS Ticonderoga*, was christened at Ingalls. After the successful completion of the Acceptance Test for the *USS Ticonderoga*'s AWS at the PTC, the weapon system was packed, shipped, and delivered to the *USS Ticonderoga.* Sea tests of VLS aboard the *USS Norton Sound* were begun. Testing the ACS aboard the *USS Ticonderoga* was begun. The keel for CG-48 was laid. The contracts for construction of the third and fourth AEGIS warships, the CG-49 and CG-50, were awarded.

1982 - *USS Ticonderoga* Passed Its Sea Trials

The ACC opened for business. Work was started to replace AN/UYK-7 computers with the AN/UYK-43B computers. Main engine lightoff for the *USS Ticonderoga* occurred followed a few months later by the turning for the first time of her propellers. Later in the same year all sea trials aboard the *USS Ticonderoga* were completed successfully in Trial Alpha, Trial Bravo, and Trial Charlie. Then the captain of the *USS Ticonderoga* took formal acceptance of the ship from Ingalls. The AN/SPY-1A radar system aboard the *USS Ticonderoga* actively tracked targets and received data from the CSEDS via Link-11. The first CG-48 radar array was tested then installed at the PTC. The contract for the CG-51 was awarded. Ingalls received the contract to build the CG-52 and the CG-53. VLS aboard the *USS Norton Sound* successfully fired SM-2s that then hit their targets. Cutting the steel for the CG-49 was started at Ingalls. A manager was named for the DDG-51 program. The US Navy decided to name the DDG-51 the *Arleigh Burke* and to include the AN/SPY-1D radar as well as VLS on it. Plans were made to implement the AN/UYK-43B computers into the DDG-51 lead ship. The CG-48 was named the *USS Yorktown.*

1983 - *USS Ticonderoga* Commissioned

The *USS Ticonderoga*, CG-47, was commissioned as a US Navy ship. The *USS Ticonderoga* was the first ACS baseline 1 ship. The *USS Ticonderoga* used the AN/SPY-1A radar. At this time the plans were to build 26 ships in the CG-47 ship class. The CG-48, *USS Yorktown,* was launched at Ingalls and later christened. VLS was modified to handle the TOMAHAWK cruise missile. The contracts for the CG-54, the CG-55, and the CG-56 were awarded. The CG-49 was named the *USS Vincennes*, and the CG-50 was named the *USS Valley Forge*.

1984 – *USS Yorktown* Commissioned

The *USS Yorktown*, CG-48, was commissioned as a US Navy ship. The first VLS for TOMAHAWK was slated to go into CG-52. The contracts for the CG-57, the CG-58, and the CG-59 were awarded. The CG-51 was named the *USS Thomas S. Gates*, the CG-52 was named the *USS Bunker Hill*, and the CG-53 was named the *USS Mobile Bay.*

1985 – Contract for First AEGIS Destroyer Awarded

The USS *Vincennes*, CG-49, was commissioned as a US Navy ship. The contracts for the CG-60, the CG-61, and the CG-62 were awarded. The CG-54 was named the *USS Antietam*, the CG-55 was named the *USS Leyte Gulf*, and the CG-56 was named the *USS San Jacinto.* The contract for the DDG-51, the AEGIS destroyer, was awarded.

1986 – AEGIS Cruisers Continue to be developed

The *USS Valley Forge*, CG-50, and the *USS Bunker Hill*, CG-52, were commissioned as US Navy ships. The *USS Bunker Hill* was the first ACS baseline 2 phase 2 ship to be outfitted with VLS. All AEGIS ships since then have been outfitted with VLS. The contracts for the CG-63, the CG-64, and the CG-65 were awarded. The CG-57 was named the *USS Lake Champlain*, the CG-58 was named the *USS Philippine Sea*, and the CG-59 was named the *USS Princeton*.

1987 – DDG-51 Ship Class Named

The *USS Thomas S. Gates*, CG-51; the *USS Mobile Bay*, CG-53; the *USS Antietam*, CG-54; and the *USS Leyte Gulf*, CG-55, were commissioned as US Navy ships. The contracts for the CG-66, the CG-67, and the CG-68 were awarded. The CG-60 was named the *USS Normandy*, the CG-61 was named the *USS Monterey*, and the CG-62 was named the *USS Chancellorsville*. The

contracts for the DDG-52 and the DDG-53 were awarded. The DDG-51 was named the *USS Arleigh Burke*. Since the DDG-51 was the first AEGIS class destroyer, the DDG-51 ship class was named the Arleigh Burke class.

1988 – Contract for Final AEGIS Cruiser Awarded

The *USS San Jacinto*, CG-56, and the *USS Lake Champlain*, CG-57, were commissioned as US Navy ships. The *USS San Jacinto* was the first ACS baseline 2 phase 3 ship. The contracts for the CG-69, the CG-70, the CG-71, the CG-72, and the CG-73 were awarded. The CG-73, the 27th AEGIS cruiser, was slated as the last guided cruiser in the CG-47 family. The CG-63 was named the *USS Cowpens*, the CG-64 was named the *USS Gettysburg*, and the CG-65 was named the *USS Chosin.*

1989 – Second and Third AEGIS Destroyers Named

The *USS Philippine Sea*, CG-58; the *USS Princeton*, CG-59; the *USS Normandy*, CG-60; and the *USS Chancellorsville*, CG-62, were commissioned as US Navy ships. The *USS Princeton* was the first ACS baseline 3 phase 2 ship as well as the first ship to use the AN/SPY-1B radar. The CG-66 was named the *USS Hue City*, the CG-67 was named the *USS Shiloh*, and the CG-68 was named the *USS Anzio.* The contracts for the DDG-54, the DDG-55, the DDG-56, the DDG-57, and the DDG-58 were awarded. The DDG-52 was named the *USS John Barry,* and the DDG-53 was named the *USS John Paul Jones.*

1990 through 1999 - Transitioning Into the Future

1990 – First AEGIS Destroyer Commissioned

The *USS Monterey*, CG-61,was commissioned as a US Navy ship. The *USS Monterey* was the first AEGIS Combat System baseline 3A ship. The CG-69 was named the *USS Vicksburg*, the CG-70 was named the *USS Lake Erie*, the CG-71 was named the *USS Cape St. George*, the CG-72 was named the *USS Vella Gulf*, and the CG-73 was named the *USS Port Royal.* The contracts for the DDG-59, the DDG-60, the DDG-61, the DDG-62, and the DDG-63 were awarded.

1991 – AN/SPY-1B(V) Radar Deployed

The *USS Cowpens*, CG-63; the *USS Gettysburg*, CG-64; the *USS Chosin*, CG-65; the *USS Hue City*, CG-66, and *USS Arleigh Burke*, DDG-51, were commissioned as US Navy ships. The *USS Cowpens* was the first AEGIS Combat System baseline 3 phase 3 ship while the *USS Chosin* was the first AEGIS Combat

System baseline 4 phase 1 ship using the AN/UYK-43 and AN/UYK-44 computers. The *USS Chosin* was the first ship to use the AN/SPY-1B(V) radar. The *USS Arleigh Burke* was the first AEGIS Combat System baseline 4 ship and also was the first ship to use the AN/SPY-1D radar. The contracts for the DDG-64, the DDG-65, the DDG-66, and the DDG-67 were awarded. The DDG-54 was named the *USS Curtis Wilbur,* the DDG-55 was named the *USS Stout*, the DDG-56 was named the *USS John S. McCain*, the DDG-57 was named the *USS Mitscher*, and the DDG-58 was named the *USS Laboon.*

1992 – Second and Third AEGIS Destroyers Commissioned

The *USS Shiloh*, CG-67; the *USS Anzio*, CG-68; the *USS Vicksburg*, CG-69; *the USS John Barry, DDG-52*; and the *USS John Paul Jones*, DDG-53, were commissioned as US Navy ships. The *USS Vicksburg* was the first AEGIS Combat System baseline 4 phase 2 ship. The contracts for the DDG-68, the DDG-69, the DDG-70, the DDG-71, and the DDG-72 were awarded. The DDG-59 was named the *USS Russell,* the DDG-60 was named the *USS Paul Hamilton*, the DDG-61 was named the *USS Ramage*, the DDG-62 was named the *USS Fitzgerald*, and the DDG-63 was named the *USS Stethem.*

1993 – Three Cruisers Commissioned

The *USS Lake Erie*, CG-70; the *USS Cape St. George*, CG-71; and the *USS Vella Gulf*, CG-72, were commissioned as US Navy ships. The contracts for the DDG-73, the DDG-74, the DDG-75, and the DDG-76 were awarded. The DDG-64 was named the *USS Carney,* the DDG-65 was named the *USS Benfold*, the DDG-66 was named the *USS Gonzalez*, and the DDG-67 was named the *USS Cole.*

1994 – Five Ships Commissioned

The *USS Port Royal*, CG-73; the *USS Curtis Wilbur*, DDG-54; *the USS Stout*, DDG-55; the *USS John S. McCain*, DDG-56; and the *USS Mitscher*, DDG-57, were commissioned as US Navy ships. The CG-73 was the last AEGIS class guided cruiser built. The contracts for the DDG-77 and the DDG-78 were awarded. The DDG-68 was named the *USS The Sullivans,* the DDG-69 was named the *USS Milius*, the DDG-70 was named the *USS Hopper*, the DDG-71 was named the *USS Ross*, and the DDG-72 was named the *USS Mahan.*

1995 – First Baseline 5 Phase 1 Ship Commissioned

The *USS Laboon*, DDG-58; the *USS Russell*, DDG-59; the *USS Paul Hamilton*, DDG-60; the *USS Ramage*, DDG-61; the *USS Fitzgerald*, DDG-62; and the *USS*

Stethem, DDG-63, were commissioned as US Navy ships. The *USS Russell* was the first AEGIS Combat System baseline 5 phase 1 ship. The contracts for the DDG-79, the DDG-80, the DDG-81, and the DDG-82 were awarded. The DDG-73 was named the *USS Decatur,* the DDG-74 was named the *USS McFaul*, the DDG-75 was named the *USS Donald Cook*, and the DDG-76 was named the *USS Higgins.*

1996 – Five Destroyers Commissioned

The *USS Carney*, DDG-64; the *USS Benfold*, DDG-65; the *USS Gonzalez*, DDG-66; the *USS Cole*, DDG-67; and the *USS Milius*, DDG-69, were commissioned as US Navy ships. The contracts for the DDG-83 and the DDG-84 were awarded. The DDG-77 was named the *USS O'Kane,* and the DDG-78 was named the *USS Porter.*

1997 – First Baseline 5 Phase 3 Ship Commissioned

The *USS The Sullivans*, DDG-68; the USS *Hopper*, DDG-70; and the *USS Ross*, DDG-71, were commissioned as US Navy ships. The *USS The Sullivans* was the first AEGIS Combat System baseline 5 phase 3 ship. The contracts for the DDG-85, the DDG-86, the DDG-87, and the DDG-88 were awarded. The DDG-79 was named the *USS Oscar Austin,* the DDG-80 was named the *USS Roosevelt*, and the DDG-81 was named the *USS Winston Churchill.*

1998 – Two Destroyers Commissioned

The *USS Mahan*, DDG-72, and the *USS Decatur*, DDG-73, have been commissioned as US Navy ships. The US Navy plans to commission the *USS McFaul*, DDG-74, and the *USS Donald Cook*, DDG-75, as Navy ships during this year.

1999 – Three Destroyers Scheduled to be Commissioned

In 1999 the US Navy plans to commission the *USS Higgins*, DDG-76; the *USS O'Kane*, DDG-77; and the *USS Porter*, DDG-78, as Navy ships.

The future, 2000 and on

- In 2000 the US Navy plans to commission the *USS Oscar Austin*, DDG-79; the *USS Roosevelt*, DDG-80; and the *USS Winston Churchill*, DDG-81, as Navy ships. The *USS Oscar Austin* is scheduled to be the first AEGIS Combat System baseline 6 phase 1 ship.

- In 2001 the US Navy plans to commission the DDG-82, the DDG-83, the DDG-84, and the DDG-85 as Navy ships. The *DDG-85* is scheduled to be the first AEGIS Combat System baseline 6 phase 3 ship.

- In 2002 the US Navy plans to commission the DDG-86 and the DDG-87 as Navy ships.

- In 2003 the US Navy plans to commission the DDG-88 as a Navy ship.

A.2 Computer Industry Highlights

Pre-1960

- In 1943 the IBM-Harvard Mark I was completed. Also, the first Colossus code-breaking machine was installed at Bletchley Park.

- In 1945 ENIAC, the first fully functional electronic calculator went into operation in November. In addition, IBM became the largest business machine manufacturer in the United States.

- In 1947 Bell Labs invented the point-contact transistor.

- In 1948 IBM assembled the SSEC electromechanical computer, which ran a stored program. Additionally, Manchester University's Mark I prototype ran the first fully electronic stored program.

- In 1949 the first full-scale electronic stored-program computer, the EDSAC, began operating at Cambridge University. Also, the first stored-program computer in the United States, the BINAC, was tested.

- In 1951 Ferranti Mark I, the first commercially manufactured computer, was installed at Manchester University. Additionally, the United States Census Bureau received its first UNIVAC. The first real-time, i.e., time-sharing, computer, Whirlwind, was completed. Also, William Shockley invented the junction transistor. In addition, Grace Hopper first conceived of an internal program to be known as a compiler.

- In 1952 Thomas Watson, Jr. became president of IBM. Additionally, UNIVAC successfully predicted the outcome of the presidential election.

- In 1953 IBM delivered its first electronic computer, the 701, to Los Alamos National Laboratory. Also, a successful full-scale test of Jay W. Forrester's magnetic-core memory was conducted at MIT.

- In 1954 IBM introduced a medium-size computer, the 650. Additionally, Texas Instruments started making silicon transistors.

- In 1955 Remington Rand merged with Sperry Corporation to form Sperry Rand. In addition, Shockley established the Shockley Semiconductor Laboratory company in Mountain View, California. Bell Telephone introduced the first computer that used transistors rather than electron tubes.

- In 1956 a team at IBM headed by John Backus completed the Formula Translation (FORTRAN) I computer programming language. FORTRAN was the first higher-level computer language.

- In 1957 Digital Equipment Corporation was established in Maynard, Massachusetts. Also Control Data Corporation was established in St. Paul, Minnesota.

- In 1958 Jack Kilby at Texas Instruments in Dallas, Texas, built the first integrated circuit. Additionally, the planar process for making transistors in silicon was developed by Jean Hoerni at Fairchild Semiconductor, Palo Alto, California.

- In 1959 Kurt Lehovec of Sprague Electric Company, North Adams, Massachusetts, designed an integrated circuit whose components were isolated with pn junctions. Also, Robert Noyce of Fairchild Semiconductor invented a planar integrated circuit. This allowed for the mass manufacture of reliable and efficient integrated circuits. The COBAL and LISP computer programming languages were developed.

1960 through 1969

- In 1960 Digital Equipment Corporation introduced its PDP-1. The ALGOL 60 programming language was developed.

- In 1961 the first computer time-sharing system was developed at the Massachusetts Institute of Technology (MIT). In addition, the Texas Instruments company built the first integrated circuit computer.

- In 1963 the Digital Equipment Corporation introduced the PDP-8, the first successful minicomputer. Also, the Bell Punch Company of Britain developed the first electronic calculators using discrete components.

- In 1964 IBM introduced its first family of computers, the System/360, to use ceramic modules with several discrete components on each unit. The BASIC computer programming language was developed.

- In 1966 Burroughs incorporated integrated circuit chips into parts of two medium-sized computers. The first object-oriented programming language, Simula I, was developed.

- In 1968 Robert Noyce and Gordon Moore established the Intel company in Santa Clara, California. Also, Intel introduced the first 1 K random-access memory (RAM). Additionally, both Control Data and NCR brought out computers composed entirely of integrated circuits. The computer mouse was demonstrated at a conference on computers.

1970 through 1979

- In 1970 IBM introduced the System/370, which was composed entirely of integrated circuits. A 256-bit RAM was introduced by Fairchild Semiconductor. Later in the year Intel introduced a 1K-bit RAM chip, the 1103.

- In 1971 the 4004 microprocessor was invented at Intel. Additionally, Texas Instruments introduced mass-produced pocket calculators in the U.S. Niklaus Wirth developed the Pascal computer programming language.

- In 1972 Intel introduced the first eight-bit microprocessor, the 8008. Dennis Ritchie and Kenneth Thompson developed the C programming language while at ITT Bell Labs. Alain Colmeraurer developed the PROLOG computer programming language. Alan Kay developed the SMALLTALK computer programming language, one of the first object-oriented languages. Word processing was introduced. The Philips Corporation introduced a disk-laser recording system.

- In 1973 the ENIAC patent was invalidated. Integrated computers were becoming commonplace.

- In 1974 an article describing the construction of a "personal minicomputer," the Mark-8, appeared in the July issue of *Radio-Electronics* magazine. Intel introduced the 8080 microprocessor to replace the 8008. The 8080 was the first

microprocessor powerful enough to run a minicomputer. Hewlett Packard introduced the programmable pocket calculator.

- In 1975 the Altair 8800 computer, based on the Intel 8080 microprocessor, was introduced in the January issue of *Popular Electronics* magazine. The Altair, with 256 bytes of memory, was the first personal computer on the market. William Gates and Paul Allen wrote a BASIC interpreter for the Altair. The Department of Defense's (DOD's) High Order Language Working Group (HOLWG) wrote a requirements document for a high-level language appropriate for embedded computer systems. This language later was named Ada.

- In 1977 the Apple II, with 16K RAM, was introduced. Paul Allen and William Gates founded the Microsoft Corporation.

- In 1978 Apple introduced the first disk drive for use with personal computers. DEC introduced a virtual address extension (VAX) computer using 32-bit words. Intel introduced the 8086, its first 16-bit processor.

- In 1979 Daniel Bricklin and Robert Frankston developed VisiCal, the first spreadsheet program for the microcomputer. Development on the Ada computer programming language was started. Motorola introduced the 68000 microprocessor chip. The new high-order programming language, whose requirements document had been started in 1975, was officially named Ada for Ada Byron.

1980 through 1989

- In 1980 Wayne Ratliff developed dBase II, a database software package that included a programming language.

- In 1981 IBM introduced its first PC using the Intel 8088 chip for its central processor unit. The Intel 8088 was a slightly simplified version of the Intel 8086 chip. Microsoft Corporation developed MS-DOS. IBM adopted MS-DOS for its PC. The Lotus 1-2-3 spreadsheet program was developed. Osborne built the first portable computer.

- In 1983 the IBM PC-XT was introduced, the first personal computer with a hard disk drive built into the computer. The Ada-83 programming language standard was published as ANSI/MIL-STD-1815A.

- In 1984 IBM developed a one-million bit RAM. The CD-ROM optical disk was introduced by Philips and SONY to store very large amounts of digital data.

- In 1985 Microsoft introduced Windows for the IBM PC. Intel introduced the 80386 32-bit microprocessor. C++ 1.0, developed by Bjarne Stroustrup of AT&T, was released.

- In 1986 Compaq introduced the DeskPro computer using the Intel 80386 microprocessor.

- In 1987 IBM and Nippon Telephone and Telegraph introduced experimental 4- and 16-megabit chips. One-megabit computer memory chips were being manufactured. IBM introduced its PS/2 group of computers that used a 3.5 inches disk drive, hard disks, enhanced graphics, and a new operating system that incorporated the Micro Channel Architecture bus that allowed for much faster data transfer.

- In 1988 Motorola introduced its 32-bit 88000 series of Reduced Instruction Set Computers (RISC) microprocessors. The Extended Industry Standard Architecture (EISA) was developed for 32-bit microprocessors, e.g., Intel's 80386 and 80486. A decision was made to revise Ada-83 with new features.

1990 through 1995

- In 1990 Intel introduced the I486 processor chip that was capable of operating at a rate of 33 MHz. Motorola introduced the 68040 chip, a version of the 68000 chip. The first Turbo C++ for DOS was released.

- In 1992 the first Microsoft C++ for DOS was released.

- In 1993 Intel started shipping its first Pentium microprocessors.

- In 1995 the expanded Ada programming language, named Ada-95, was made available to developers.

Vita