

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

1

AD-A154 433



<p>A MICRO-COMPUTER MODEL FOR ARMY AIR DEFENSE TRAINING</p>	
<p>THESIS</p>	
<p>David L. Bolté Captain, U.S. Army</p>	
<p>AFIT/GST/MATH/BSM-1</p>	
<p>Acces NTIS DTIC Unan Just</p>	<p>By— Dist</p>

DTIC FILE COPY

This document has been approved
for public release and sale; its
distribution is unlimited.

DTIC
ELECTE
JUN 4 1985
S D
E

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

85 5 07 100

AFIT/GST/MATH/85M-1

(1)

DTIC
ELECTE
S JUN 4 1985 **D**
E

A MICRO-COMPUTER MODEL
FOR
ARMY AIR DEFENSE TRAINING

THESIS

David L. Bolte
Captain, U.S. Army

AFIT/GST/MATH/85M-1

Accession For	
DTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	<input type="checkbox"/>
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A/ I	

Approved for public release; distribution unlimited

DTIC
COPY
RESERVED
3

AFIT/GST/MATH/85M-1

A MICRO-COMPUTER MODEL FOR ARMY AIR DEFENSE TRAINING

THESIS

**Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology**

Air University

**In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research**

David L. Bolté

Captain, U.S. Army

March 1985

Approved for public release; distribution unlimited

Table of Contents

	Page
Preface	ii
Abstract	v
Acronyms	vii
I. Introduction	1-1
Background	1-1
Problem Description	1-4
Objectives	1-4
Scope	1-5
Literature Review	1-6
Summary	1-13
Presentation	1-13
II. Methodology	2-1
Model Description	2-1
Parameter Considerations	2-3
Probability Calculations	2-5
Random Number Generation	2-11
Programming Considerations	2-12
III. Implementation	3-1
Overview	3-1
Detailed Description	3-1
Program Flow	3-6
Development System	3-7
Portability Tests	3-8
Input/Output	3-9
Error Handling	3-10
Verification	3-11
Validation	3-11
IV. Conclusions and Recommendations	4-1
Summary	4-1
Recommendations	4-3
Conclusion	4-5

Preface

The purpose of this study was to develop a computer program that could be used in place of pencil-and-paper calculations. My specific interest was the calculation of probabilities of engagement and kill for air defense/aircraft interactions in various war games. In my Army career, I have participated in several studies using war games, and while some of the games were computer supported, the air defenders were always left with table look-ups for what were sometimes extensive calculations.

I wanted to design a program that could be used on almost any micro-computer by people who were not programmers. I also wanted to design a program that could be changed by a programmer to reflect new weapons or a specific war game. The Air Defense War Game Support Program was the result of these desires.

I owe a debt of gratitude to my thesis advisor, Captain Patricia Lawlis, who showed a great deal of patience with me. Her advice on programming and thesis construction was invaluable. I would also like to thank Major James Coakley and Lieutenant Colonel Palmer Smith, both of whom reassured me at times that I wasn't off base.

My greatest thanks go to my wife, Robin, who supported me throughout the long haul, and my daughters, who managed to curb their desires to play on the computer while I worked.

David L. Bolté

	Page
Appendix A : User's Guide	A-1
Appendix B : Sample Session	B-1
Appendix C : Flow Diagrams	C-1
Appendix D : Source Code Listing	D-1
Appendix E : Spare Parts	E-1
Appendix F : Random Number Generator Tests	F-1
Appendix G : Default Base and Factor Values	G-1
Bibliography	Bib-1
Vita	V-1

ABSTRACT

thesis
This ~~report~~ ^{microComputer} details the development of the Air Defense War Game Support ^{which} Program. The Air Defense War Game Support Program provides one-on-one probabilities of engagement and kill for a variety of forward-area air defense systems against different types of aircraft. Each air defense system is described by type (Vulcan, SGT York, Redeye, Stinger, or Chaparral), location (Europe, desert or jungle) and effectiveness level (0.0 to 1.0). Each aircraft is described by type (helicopter, transport or fighter), tactic (pop-up, lay-down or fly-over), and profile or aspect presented to the air defense system (head-on, crossing, or tail). The results of an interaction between an air defense unit and an aircraft are determined by comparing a program generated random number with threshold values calculated from data tables. The displayed results of an engagement include whether the aircraft is engaged, whether the aircraft is hit and destroyed or mission suppressed, and whether the air defense system suffers any attrition from an aircraft that was engaged but not destroyed or suppressed. ←

Within the program the factors used in the calculation of the engagement and kill thresholds may be changed. Additionally, the probabilities of air defense system attrition and aircraft suppression, the attrition factor and the random number seed may be changed.

The program is designed specifically to be used by personnel without a computer background, yet has a structured programming style that permits easy modification by a programmer. The Air Defense War Game Support Program is written in standard Pascal and can run on a standard micro-computer with 64 thousand (64K) bytes of memory.

LIST OF ACRONYMS

ADCDP	Air Defense Capability Development Plan
AMSAA	Army Materiel Systems Analysis Activity
APC	Advanced Personal Computer
BATTLE	Battalion Analyzer and Tactical Trainer for Local Engagements
CDC	Control Data Corporation
DADENS	Division Air Defense Engagement Simulation
DEC	Digital Equipment Corporation
DIVAD	Division Air Defense Gun
EVADE	Evaluation of Air Defense Effectiveness
MAXINT	Maximum Integer
NEC	Nippon Electronics Corporation
SHAPE	Supreme Headquarters, Allied Powers Europe
STAGEM	Surface-to-Air Gun Effectiveness Model
TRADOC	Training and Doctrine Command
TRASANA	TRADOC Systems Analysis Activity
USI	User-System Interface

I. Introduction

In Army Air Defense Artillery war gaming one of the more tedious tasks can be the repetitive dice rolls and calculation of results from numerous paper tables. Random number generation and table look-ups are particularly suited for computers, but the programs that provide the proper tables and numbers are not always available for those that need them. One particular group of war game users in the Army are the trainers of the junior officers of forward-area air defense weapons units. They have had to resort to the dice and paper tables for the air defense system probabilities of engagement and kill in their training and in support of large-unit war games. In this thesis a computer program is developed to provide random numbers and air defense/aircraft interaction results, a program with characteristics of usability, portability and small size which make it particularly suited for use at the small-unit and training school levels.

Background

There are currently in use in the U.S. Army numerous methods for determining the probabilities of engagement and kill required for air defense support in war games and models. Some of the methods used by various commands involve probability tables that have been derived from studies done in other war games, by manufacturers of weapons

systems, and from commercial games simulating combat. Many of the probability calculation procedures are based on weapons systems that are currently being replaced with upgraded models or new systems.

Current Systems. At White Sands Missile Range, New Mexico, the TRADOC Systems Analysis Activity (TRASANA) maintains a continuous war game called the Battalion Analyzer and Tactical Trainer for Local Engagements (BATTLE). The scope of BATTLE is scenario restricted as it models only Fulda Gap, Germany, and it is designed to model one or two battalions of infantry or armor with provisions for some short-range air defense. In this model, a computer is used for random number generation, but paper tables are used for the air defense probabilities of engagement and kill.

The U.S. Army's Air Defense Artillery School at Fort Bliss, Texas, routinely uses war games in the training of officers in the basic and advanced courses. Various models are used to simulate the air defense activity, each lacking a standard procedure for determining the probabilities of engagement and kill. Some of the tables are derived from Fort Leavenworth's war game DUNN-KEMPF, others from a set of complex equations developed in an early model [Blum, 1963]. These were developed for the weapons systems of the 1960's: the Vulcan 20mm gun and Chaparral infrared-seeking missile. At Aberdeen Proving Grounds, Maryland, the Army Materiel

Systems Analysis Activity (AMSAA) runs a Monte Carlo model of platoon and battery forward-area Air Defense units. AMSAA's INCURSION model is implemented on CYBER and UNIVAC computers.

One of the best large air defense models in use is at the Technical Center of the Supreme Headquarters Allied Powers, Europe. Its complexity is permitted by the support of large computers and a dedicated programming staff [Cameron et al, 1980; Wilhelm, 1979]. Other models requiring similar support include: DADENS, the Divisional Air Defense Engagement Simulation [Semmens et al, 1977]; STAGEM, Surface-to-Air Gun Effectiveness Model [Young and Solomon, 1977]; and EVADE II, Evaluation of Air Defense Effectiveness [Paris et al, 1974].

The requirements for large computers (CDC 6600 or CDC Cyber usually) and programming expertise are the primary limitations on the use of larger air defense models. Computers that can run these models are not available at Army division, battalion or school levels. The Air Defense Artillery School's models are not designed for the new weapons systems, and, as a general rule, are cumbersome paper and pencil exercises of limited utility. The Teledyne Corporation has developed a set of air defense models for a mini-computer, but these are oriented towards the development of new air defense systems rather than the modeling of current systems and support of larger models [Hays and Linton, 1974].

Problem Description

Particularly for the forward-area air defense systems, there exists a need for a general model to produce reasonable engagement and kill statistics for small units and training commands. This model could also be used to complement war games and models of larger units. The specific problem environment identified is that of the U.S. Army Air Defense School at Fort Bliss, Texas, and numerous forward-area air defense battalions. Constrained by the limited availability of large computers, the immediate solutions include a model on paper that could be easily referenced and a model designed for a smaller computer that does not require extensive programming.

Objectives

The primary objective of this thesis is the development of a computer program to model a variety of forward-area air defense systems in different situations. The model is to provide the probabilities of engagement and kill for the interactions between an air defense system and one of a variety of aircraft necessary to support the training of student officers, to support small-unit war gaming, and to supplement large-unit war gaming. The model is to be designed as an interactive program, written for a micro-computer, designed to be flexible, portable and easy to use by inexperienced personnel.

A primary subobjective is to use a modular approach to the program, facilitating changes and additions. This will permit the addition of different weapons systems or scenarios and changing of the methods of probability calculations. Additional subobjectives are:

- to develop a model that is applicable to the current inventory of air defense weapons and flexible enough to reflect changes in either the weapons systems or the threat,

- to include characteristics in the program that will make the model usable by personnel without a computer background,

- to use a standard computer programming language to make the program portable to a variety of computer systems, and

- to provide sufficient documentation for the program to allow programmers to modify the program as necessary.

Scope

In this thesis, each air defense system is described by system type, location and effectiveness level; and each aircraft is described by type, tactic and aspect presented to the air defense unit. The program provides for one-on-one interactions between the air defender and the attacking aircraft and generates both the random numbers and results of an interaction. Radar detection and radar cross-section of the target are not considered essential for this model, as radar is not the primary detection means for the

dominant number of forward-area air defense systems. The modeling of electronic counter-measures is considered to be beyond the scope of the program. An engagement is modeled only as an interaction between the air defense system and the aircraft and does not consider the particular killing mechanism of the weapon.

Assumptions. The basic assumption in the program is that the probabilities used in the program are based solely on the factors in the air defense unit and aircraft descriptions. The reaction time of the defender and the range of the first shot, given engagement, are controlled by an assumption that engagement and the first shot is made inside the maximum range of the weapon.

The computers and support software necessary to run the program developed in this thesis are not currently available in the forward-area air defense artillery battalions or at the Air Defense Artillery School. A basic assumption has been made that micro-computers and a compiler will be available in the near future at the utilization levels for which the model is designed.

Literature Review

Research for this thesis is limited in scope. There are a plethora of models which deal, at least in part, with ground-based air defense, but there is a dearth of information available. The primary source for information on the models previously cited was the Defense Technical

Information Center, yet word-of-mouth and phone conversations brought other models to the attention of the writer. Many of the models and programs in use at various agencies are not documented. These models, having been developed for specific transient applications, are retained in the developing offices but not documented for use outside a particular agency. [Battilega and Grange, 1983:4] Other models involve classified information which the writer desires to avoid as restricting eventual usefulness of the final product. In the writer's practical experience there are no models or programs in use at the level for which this program is developed. As a result, the literature review serves to give a general idea of some of the parameters included in other programs and models.

Beyond the computer models currently in use, the author is concerned with modeling and programming in general. The type of model to be constructed had to be determined, as did both the computer and programming language to use.

Modeling. Research involving modeling requires some background on types and classification of models. The models reviewed are generally either Monte Carlo or expected value models. The difference between the two types is in the manner in which the results of chance events are determined. In the Monte Carlo model, chance events are handled by drawing random numbers from a distribution. By comparing these numbers with a previously established threshold, a determination of the consequences or results of a particular

action is made. In the expected value model, statistics are gathered and the results of a particular action then reflect an expected value, such as a number of aircraft that survive a particular attack. Another method for handling chance events uses mathematical analysis to deal with fluctuations in the statistics used for the expected value model. [Specht, 1968:211] The Monte Carlo model is the more widely used in the programs reviewed and is the more appropriate for this thesis. The choice of the Monte Carlo model reflects the objective of a descriptive model for training purposes. Extensive collection of classified data and use of these statistics for the results of the air defense/aircraft interactions would have produced a predictive model, one from which real world conclusions could be drawn. [Battilega and Grange, 1983] This is not an objective and is not permitted in the time allocated to construct a model.

Programming. The two major considerations for programming are the choice of computer system and language. Computer systems available to the author and expected to be available in the near future to potential users of the program were reviewed. The computers available to the author in the school environment range from 8- and 16-bit micro-computers to DEC VAX mini-computers and a CDC CYBER. An objective for the thesis is to develop a program for a micro-computer and the author's own 16-bit system is a

logical choice for a development system. Languages available on the chosen system are compiled and interpreted BASICs, FORTRAN, and Pascal. The objective of portability for the program caused an examination of standardization of languages so that the program would be compiler or interpreter independent. The subobjectives of modularity and flexibility in the program are considered to be filled by a Pascal program's structured design. Pascal's availability, standardization and structure make it the language of choice for this project. [Hill, ed, 1980]. Further research on Pascal included usage [Jensen and Wirth, 1974] and implementation [Barron, ed., 1981] of the language.

The user interface, and specific structures for naive users are of concern in construction of a program for those who are not 'computer users'. The use of menus is considered to be the best method for the interface for the untrained. [Smith and Aucella, 1983] The type and amount of information to display on a screen is set at a level close to that of the most ubiquitous of computer terminals used, the bank teller machine. [Rubinstein and Hersh, 1984].

Models currently in use are reviewed in the following sections. Chapter II fully explains the selection of particular parameters for the model under construction.

ADCDP. The Air Defense Capability Development Plan used by the Teledyne Corporation is designed to facilitate the weapons systems acquisition process. The model consists

of a family of simulation models for one-on-one, many-on-one, and many-on-many air defense-aircraft interactions. The emphasis of the models is on changing of design criteria and examination of the resultant changes in engagement and intercept probabilities. [Hays and Linton, 1974:4-1] The ability to change program parameters and examine the effects of the changes is considered a significant characteristic for the program under development.

DADENS. The Divisional Air Defense Engagement Simulation is primarily concerned with the command and control of air defense systems, yet is applicable as a reference for its scope and use of the Monte Carlo technique. The simulation includes bombers, fighters and helicopters, using a variety of attack techniques against both ground and air-to-air air defense systems. The interactions between air defense system and aircraft are influenced by the terrain and electronic counter-measures. [Semmens et al, 1977:1-2]

STAGEM. The Surface-To-Air Gun Effectiveness Model is a simulation relating statistics on gun performance to the probability of kill. Factors considered are slew rate, target flight path, and number and trajectories of the gun's projectiles. [Young and Solomon, 1977] The parameters for this expected value model are considered too specific for a training model.

EVADE. The Evaluation of Air Defense Effectiveness is a FORTRAN IV program currently running on a CDC 6600 computer. The primary emphasis is on the target flight path, and resultant vulnerability of an aircraft as it becomes unmasked to the air defense weapon's detection system. While multiple aircraft are flown over numerous air defense systems, the results returned are from one-on-one engagements. Attrition on both systems is calculated. [Paris et al, 1974:viii] The one-on-one engagements, flight path and attrition calculations from this model are pertinent for this thesis.

SHAPE models. The models in use at the Supreme Headquarters Allied Powers, Europe, evolved from several years' work. The programs combine Monte Carlo and expected value models. Both FORTRAN and Pascal subroutines are used in a very detailed description of air defense/aircraft interactions. Aircraft specifics include flight path, altitude, angle of approach and angle of attack. Air defense systems range from the forward-area systems to medium- and long-range systems and include primary detection means; line-of-sight for radars; Identification, Friend or Foe (IFF) systems; and electronic counter-measures. [Wilhelm, 1979,1980,1981] This model provides a good range of aircraft and air defense system characteristics. This range is applicable not only for the current program construction but also for expansion possibilities.

Mathematics. With the selection of the Monte Carlo model type, random number generation on micro-computers became an issue. The most widely used random number generator for models and simulation, the linear congruential pseudo-random number generator, is chosen. [Banks and Carson, 1984] The specific number for the modulus operation and initial random number seed are significant and specific to micro-computers. [Thesen and Wang, 1983] Suitability tests of the random number generator include the Chi-squared test for frequency, tests for runs up and down and runs above and below the mean, and tests for periodicity. [Banks and Carson, 1984].

DUNN-KEMPF. The DUNN-KEMPF war game is a pencil, paper and dice simulation developed at the Combined Arms Center, Training Developments Activity, Fort Leavenworth, Kansas. It is a training tool designed to provide instructors and students with practice in tactics and techniques of company level combat operations. One aspect of the simulation concerns calculations of probabilities of engagement, kill and suppression of aircraft with dice and tables of thresholds. This game is currently in use at the U.S. Army Air Defense Artillery School in the training of company grade officers. [Bolté, 1977, 1982] Adoption of this game as a program base permits the program to be integrated in the School's curriculum with ease.

Summary

Once the decision was made as to the objectives of this thesis, resources and current systems were examined. Compiler and computer availability and language suitability dictate the development system. A modular approach is taken to both model and program development. The initial step taken in development of the computer model is to define which characteristics are applicable to the model and needed by the anticipated user of the program. The next step is to determine the method of calculation for the engagement results. The program is designed by modules from a main program shell to encompass each characteristic selected. When the program is completed it is to be compiled and run on four different computer systems using three distinctly different Pascal compilers.

Presentation

Fully developed in succeeding chapters are the characteristics of the program and the parameters chosen for the model. The methodology used in the development of both the model and the program is covered in Chapter II. Specifically covered are the language and hardware selection and restrictions, program parameter considerations and probability calculations. Chapter III includes the specifics about program construction, data types, flow and size. Use of the program with different compilers, possible error areas, verification of the program and validation of

the model are also covered in Chapter III. The final chapter appraises the value of the program and model and makes recommendations for enhancements and applications. A user's guide for the Air Defense War Game Support Program is included as Appendix A, and a sample session is contained in Appendix B. Appendixes C and D contain the flow diagram and program source code listing respectively. Procedures for specific compilers to enable file input and output are listed in Appendix E. Appendix F covers the tests of the random number generator. Tables of the default values for the program data arrays are contained in Appendix G.

II. Methodology

The Air Defense Wargame Support Program is intended for use at the U.S. Army Air Defense Artillery School. A training vehicle currently in use at the School is used as the base for a program to replace tedious paper-and-pencil calculations.

Model Description

The model constructed in the Air Defense Wargame Support Program is a simplified representation of the interaction between an aircraft and a forward-area air defense artillery system. The purpose of the model is to provide the results of that interaction, results that may be used, as the program name suggests, to support larger war games, or the training of air defenders. The model is designed with the purpose of saving time for the users and providing a standard method of probability calculations for various Army commands. It is intended primarily to replace the current dice rolling and paper table look-up methods currently in use at the U.S. Army Air Defense Artillery School and in the forward-area weapons battalions of the Army. The model is a descriptive micro simulation of a one-on-one engagement and is not designed to be predictive of the results that would occur in actual combat. As a result, the model is extremely restrictive in the parameters and the number of variables used and the results that are generated.

With these restrictions in mind, the Air Defense Wargame Support Program is properly designated computer-assisted wargaming and not modeling. [Battilega and Grange, 1984:14,25]

The methodologies which are available to provide a basis for this model involve conversion of current computer or paper systems to a micro-computer model. The conversion and expansion of the current paper-and-pencil models would be of greater utility to the training of junior officers at the Air Defense Center. A model so constructed could be integrated in the present course of instruction for junior officers with ease. A more difficult and comprehensive project is the conversion of a mainframe computer model such as INCURSION for a micro-computer. This is not considered for this thesis project because of the time constraints for program and thesis completion. Conversion of a model of this size and type generally involves classified data which limits the use of the model and program in training environments.

Unclassified data for the Air Defense Center model conversions are available from the Tactics Department at the school in Fort Bliss, Texas; and it is from this data that the model is constructed.

Parameter Considerations

Different characteristics or parameters of the air defense problem may be specified in the models reviewed previously. The number and range of the parameters that can be varied is restricted by the size of the computer, as each parameter must be determined by either an equation or a data base. The following parameters were selected as essential either for basic description of the situation or the engagement:

1. Air Defense system: Vulcan, SGT York, Redeye, Stinger, Chaparral
2. Location: European, desert, jungle
3. Effectiveness level of the air defense system
4. Aircraft type: fighter, transport, helicopter
5. Aircraft tactic: pop-up, lay-down, fly-over
6. Aircraft aspect: head-on, crossing, tail.

These parameters are described in the following paragraphs.

Air Defense System. As this model is designed to support the near-future inventory of the U.S. Army, statistics in the model provide all the current active Army air defense weapons: the Sergeant York (DIVAD) 40mm twin-barrel gun, the improved Chaparral infrared-seeking missile, the shoulder-fired, infrared-seeking Stinger missile, the 20mm Vulcan cannon and the Redeye missile.

Location. Since the characteristic probability of detection and kill of an air defense system is affected by the area in which the system is deployed, the general area

of deployment may be specified. The three major scenarios modelled are the European, the desert, and the jungle. These are the three scenarios generally trained for and simulated in the military.

Effectiveness Level. The effectiveness level for the air defense system is designed to include in the model those characteristics of the air defense system that are difficult to quantify. These include the human factors such as the manning strength, alertness and disposition of the system squad. Additionally, the effectiveness level can reflect the maintenance status of the equipment and any attrition suffered by the air defense unit in an engagement.

Aircraft Type. There are two general attack characteristics that will affect the probabilities of engagement and kill for this model: type of aircraft and type of attack. The aircraft types and related speeds can be generalized to high-performance (fighters and close air support), planes (transports and reconnaissance) and helicopters. These categories reflect current Army Air Defense Artillery doctrine. [FM 44- series]

Aircraft Tactic. Attack types include the tactics of pop-up attack, lay-down bombing runs and flying over the defending weapons system. Army Air Defense Artillery doctrine considers these tactics as the most likely to be seen on a battlefield. [FM 44- series]

Aircraft Aspect. The aspect angle or profile that the aircraft presents to the air defense system is part of its attack characteristic. The aspect angle and profile determine the visible cross-sectional area which most influences an engagement for a forward-area weapon. When, as for the systems modelled, human sight is the primary detection and fire control means, the degree to which the aircraft is exposed is as important as the length of time it is visible. The importance of the aircraft aspect is covered extensively in Army Air Defense Artillery Field Manuals on weapon system characteristics and employment principles. Army Air Defense Artillery training routinely includes cards and slides of threat aircraft depicting the three main profiles of an aircraft: front, side and rear.

Probability Calculations

The probability calculations are a direct conversion of the DUNN-KEMPF tables used at the Army Air Defense Artillery School. The program values for engagement and effect probabilities are listed in Appendix G.

The results of an interaction between an Air Defense Unit and an Aircraft are determined by comparing a program generated random number with calculated threshold values. Postive results from an action are returned if the random number is less than the threshold; so if the threshold for the probability of engagement is 65 and a random number of 50 is returned, then the aircraft is considered engaged.

Random numbers are generated for each of the engagement, kill, suppression and attrition results. Threshold values are calculated separately for the probabilities of engagement and kill. The threshold values are calculated by multiplying a base value by the series of factors reflecting the descriptions of the Air Defense Unit and the Aircraft. The displayed results of an engagement include whether the aircraft is engaged, whether the aircraft is hit and destroyed or mission suppressed, and whether the air defense system suffers any attrition from an aircraft that was engaged but not destroyed or suppressed. Attrition on an air defense unit is reflected by a lowering of the unit's effectiveness level by the amount of the attrition factor.

Parameter Factors. The parameter factors in the program are the most subjective aspect of the model, and for that reason all of them may be changed by the user. In general, consideration is made of what impact each parameter has on an actual engagement between an air defense artillery system and the type of aircraft to be engaged. The relative values for each factor are based on the degree to which that air defense unit or aircraft characteristic would affect the actual engagement with the rank ordering based on air defense system capabilities documented in the appropriate Army Field Manuals. For the location factor, the desert is the optimal air defense environment and the jungle is the worst, while the European location provides a neutral environment (neither very good or very bad). These

assessments are based on the fact that the primary detection and fire control mechanism for the forward-area air defense artillery systems is human sight. For the tactic factor, the highest value is for the fly-over tactic based on the concept that that tactic results in the greatest aircraft exposure time to the air defense system; while a pop-up tactic results in the least exposure time, and a lay-down tactic provides a median value. The aspect factor reflects the engagement difficulty of a particular cross-sectional area exposure of the aircraft. For the initial values in the program, the head-on and tail shots are considered to be of approximately the same difficulty, and the crossing aspect is considered to enhance the engagement for the air defense system. Again, this rank ordering reflects the actual system capabilities.

Engagement Thresholds. The engagement thresholds are calculated from a probability of engagement base value multiplied by the values reflecting the description of the air defense unit and the aircraft. The base value is derived from the probability of engagement table from the war game DUNN-KEMPF. The DUNN-KEMPF tables cover Vulcan, Redeye and Chaparral engagements and are expanded to include the SGT York gun and the Stinger by increasing the Vulcan and Redeye figures respectively by an arbitrary 10 per cent to reflect the more modern systems' improved engagement capabilities. The specific base values for the engagement

threshold used are dependent on the air defense unit type and aircraft type.

The factors by which the base value is multiplied are dependent on the air defense and aircraft types in a similar manner. The location factors are dependent the air defense unit type and location; for the tactic factor array the aircraft type and tactic determine the value; the aspect factor is dependent on the aircraft type and aspect. The final step in calculation of the threshold is the multiplication of the base and factor product by the air defense unit effectiveness level and a scaling factor of 100.

This results in

$$\begin{aligned} \text{Engagement} \\ \text{Threshold} &= (\text{Probability of Engagement Base Value}) \\ &\quad \times (\text{Location Factor}) \times (\text{Tactic Factor}) \\ &\quad \times (\text{Aspect Factor}) \times (\text{Effectiveness Level}) \\ &\quad \times 100 \end{aligned} \quad [11]$$

The determination of aircraft engagement is made by generating a random number between 1 and 100 and comparing the random number with the calculated threshold value. If the random number is less than the threshold, the aircraft is considered to have been engaged by the air defense unit and calculations for the probability of hit are done.

Hit Thresholds. The probability of hit thresholds are calculated in the same manner as are the engagement thresholds. The probability of hit base value table is constructed from the DUNN-KEMPF tables as is the probability of engagement base value table. The dependencies for each

of the factor values are the same as for the engagement threshold. This results in

$$\begin{aligned} \text{Hit} \\ \text{Threshold} &= (\text{Probability of Hit Base Value}) \\ &\times (\text{Location Factor}) \times (\text{Tactic Factor}) \\ &\times (\text{Aspect Factor}) \times (\text{Effectiveness Level}) \\ &\times 100 \end{aligned} \quad [2]$$

Whether or not an aircraft is hit is determined by the generation of a random number and comparison with the calculated probability of hit threshold. If the random number is less than the threshold, then the aircraft is considered to have been hit by the air defense unit, and the determination of aircraft suppression is made by another random number calculation. If the aircraft is not hit, then a determination of air defense unit attrition is made.

Suppression. Suppression refers to the damage level incurred by the aircraft. The basic mission of Army air defense is to destroy or nullify the threat. [FM 44-series] For the purposes of the program, if an aircraft is hit, then it is either destroyed or its mission is aborted. The determination of aircraft suppression is made by comparison of the program's probability of suppression with a generated random number. The probability of suppression is set in the program at 50 per cent but, may be changed by the user. The 50 per cent level reflects a basic coin toss as to whether the aircraft is suppressed. If the random number generated

for the determination of aircraft suppression is greater than the probability of suppression, then the aircraft is considered to have been suppressed, otherwise the aircraft is considered destroyed.

Attrition. Attrition refers to any damage inflicted on the air defense system by an aircraft that is not hit. The determination of air defense unit attrition is made only if the aircraft is engaged but not hit. This reflects the real world situation that a forward-area air defense system is not normally exposed unless it fires and reveals its position. [FM 44- series] If an aircraft is not engaged then the air defense system does not reveal its position to draw fire from the aircraft. An aircraft that is fired on and not suppressed is free to attack the air defense system and cause damage and attrition. Determination of attrition is made by comparison of the probability of attrition with a generated random number. The probability of attrition is set in the program at 50 per cent, but may be changed by the user. The 50 per cent level reflects a basic coin toss as to whether the air defense system is attrited. If the random number generated for the determination of air defense unit attrition is greater than the probability of attrition, then attrition is effected by subtracting the program attrition factor from the air defense unit's effectiveness level.

Random Number Generation

Method. The random number generator is designed for a micro-computer, but provides the required numbers on larger systems as well. A simple function implements a linear congruential pseudo-random number generator. This random number generator is the one most widely used in simulation and modeling. [Banks and Carson, 1984:259] Each random number generated serves as the basis for the next. The linear congruential generator is of the form

$$I[i+1] = (a \times I[i] + 1) \text{ mod MAXINT} \quad [3]$$

where

$$a = k \times 4 + 1 \quad k = 0, 1, 2, \dots \quad [4]$$

and MAXINT is implementation dependent. Certain values of a provide better sequences of random numbers for micro-computers, some of these are specified in the program comments. [Thesen and Wang, 1983:8]

Range. The linear congruential generator provides for a uniform distribution of random numbers. This emulates the uniform distribution of numbers resulting from the roll of a die. The random numbers returned by the equation itself range from 0 to MAXINT, these are scaled in the program to the range 1 to 100 for ease of comparison.

Tests. Separate programs were run with the random number generator to determine the period and "random-ness" of the generator. The period is 32763 numbers generated before a repetitive sequence is encountered on the development system. Chi-Squared tests for frequency were made with satisfactory results. Tests for runs up and down

and runs above and below the mean value of 50 were also made with satisfactory results. [Banks and Carson, 1984:271-278] Specific test results are found in Appendix F.

Programming Considerations

The major programming considerations are the type of computer system used and the language in which the model is constructed.

Hardware. The restriction of the model to a micro-computer limits the memory for the main program to 64 thousand (64K) bytes which, is the common minimal micro-computer configuration at the time of writing. The memory limitation impacts on the method used for calculating results of engagements. Studies have determined that modeling using the Lanchester equations is possible on the micro-computer [Callahan and Crosby, 1981], but the use of these complex equations is beyond the scope of this program and model. A simpler relation of aircraft and air defense system characteristics to internal data arrays is used.

Software. There are numerous programming languages available for micro-computers, including high-level simulation languages. The Teledyne models, which bear the closest resemblance to the developed model, are written in BASIC. While BASIC has the advantage of straightforward and simple programming, it is restricted in its abilities to deal with complex models and does not lend itself to memory conservation. Additionally, micro-computer versions of

BASIC frequently are very specific to the hardware in the computer system for which they are designed and may rely on software residing in read-only memory, as does IBM BASIC. Those characteristics restrict the portability of programs written in BASIC. FORTRAN is available for micro-computers and has the advantage of being widely used for modeling, but is not considered one of the simplest languages in which to program. FORTRAN's lack of standardization [Hill, ed., 1980] restricts program portability. Pascal is another high-level language that is readily available for micro-computers and has been used for military modeling.

During selection of a programming language, compiler availability had to be considered. A survey of the computer systems available to the writer in the school environment showed five different Pascal compilers running on seven different operating systems and computers ranging from TURBO Pascal on 64K Kaypro and HeathKit computers to Pascal 6000 running on a CDC Cyber. This compiler variety became a major factor in the attempt to make a highly portable program. Pascal was selected for its availability and ease of programming even complex models. Pascal enables the development of a large model or program in modules. This eases program construction and enhances program maintenance. An additional advantage to the use of Pascal is its similarity to Ada, the chosen programming language for the Department of Defense. Standardization of the language is

also considered. Research shows that aside from Ada with its Department of Defense dictated standards, Pascal comes the closest to having a 'standard' instruction set. [Hill, ed., 1980] Ada itself was not chosen because of its limited availability on micro-computers at the time of writing.

III. Implementation

This chapter examines the structure, development and tests of the Air Defense War Game Support Program. The characteristics of the development, the portability tests of the program, and the computer systems used are reviewed and findings discussed. Verification of the program flow and validation of the program results are also covered.

Overview

The Air Defense War Game Support Program consists of approximately 2500 lines of standard Pascal code that provide the user with the results of a modelled engagement between a U.S. Army Air Defense Artillery system and an aircraft. It is highly transportable to a variety of Pascal compilers, operating systems and terminal types. A strictly modular design eases replacement of procedures and permits substantial program modification with a minimum of undesirable side effects. Program modifications are also facilitated by use of Pascal types and constants. The user interface to the program is menu driven and requires only a minimum of keyboard input.

Detailed Description

Structure. The program is designed with the top-down programming approach. Menu selection provides the method to control program flow for users who may not be "computer-

oriented". The initial decision was made that the main program would consist of a loop with a single control variable that determined the next procedure to be called. The resultant shell program gives the capability to call the major menu procedures and additional procedures that branch from the major menu procedures. The single main program loop permits interaction between menus that would not be possible if each set of procedures for a specific menu was isolated in a separate loop. This structure permits the exit to a higher level menu without stepping through each of the intervening menus, but does add to the complexity of the program. There are six major menus that access 20 main program procedures and 12 support procedures.

Main Program Procedures. The six major groupings of main program procedures deal with Main Menu operations, Air Defense Unit operations, Aircraft operations, and three sets of Parameter Menu operations. The main menu transfers control to the other menu groups; the air defense unit and aircraft operations are concerned with the describing, listing and file input/output operations with the respective data arrays. The parameter menus permit the listing, changing and file input/output operations for the model parameter factor and probability tables.

Menus provided by numerous procedures are in a standard format of a labeled vertical list of five choices. The number of choices is restricted to five to minimize the amount of information a novice user must read prior to being

able to make a choice. [Rubinstein and Hersh, 1984:119-121]

The limit of five displayed choices means that help screens, a sixth option from the command line, are not available in screens that need the sixth option in order to exit the menu. Each menu has the most frequently selected codes listed first and requires a single alphabetic key entry to proceed. Keys required for menu selection are consistently A thru E, H and X, in either upper or lower case. This standard menu form is adhered to throughout the program as an implementation of user-system interface (USI) standards. These standards cover the use, size and construction of menu driven programs. The particular menu and command structure used is designed for untrained users. [Smith and Aucella, 1983:118-130]

Support Procedures. The support procedures include procedures to clear the screen, to write the program name or a screen title, to provide random numbers and to solicit the user for specific responses needed by main program procedures. In these procedures, specific efforts made to adhere to standard Pascal result in some loss of efficiency and programming aesthetics, notably in the screen clearing procedure which simply scrolls previous information off the screen.

Constants and Types. Program changing and maintenance are facilitated by use of Pascal constants and types. This permits changing of values throughout the program without having to change each occurrence of a variable, constant or type, by placing the new value once in the program declarations section. Constants are used to dimension the data arrays. Other specific constants include the initial values for:

- the random number stream
- attrition factor
- probabilities of attrition and suppression

and the maximum values for:

- the number of air defense units and aircraft
- types and locations for air defense units
- types, tactics and aspects of aircraft.

Specific Pascal types include record structures for the descriptions of the air defense unit and aircraft. These records are maintained in single-dimension arrays of those records. Each record contains a boolean field, 'Isactive', to indicate whether a description has been entered. This field is used in the procedures that verify an user-entered unit or aircraft number, and in procedures that display lists of the currently entered descriptions. Each record contains integer fields that are used as indexes to the data arrays. The air defense unit record, 'ADRECORD', contains integer fields for the unit 'Kind' and 'Location' and an effectiveness level field, 'Effectlevel', of type real. The

aircraft record, 'ACRECORD' contains integer fields for 'Sort', 'Tactic' and 'Aspect'. A particular data array element is then specified by use of the integer fields as indexes. Type specifications are used for the two-dimension data arrays of factors and for a single-dimension data array of the character strings used to describe each of the factors. Examples of each are provided in the next section. The use of the record structure for the air defense unit and aircraft permits the expansion of the descriptions of each with additional characteristics. The use of arrays dimensioned by constants permits a similar expansion for more air defense units, aircraft or descriptive factors.

Data Arrays. The data arrays in the program are classified as variable data arrays, descriptive arrays or factor arrays. The key data manipulated by the user are the variable data arrays of air defense units and aircraft records. These arrays of records are modified in the 'Enter Description' procedures and referenced during engagements for indexes to the factor arrays. The indexes in the records are also used as indexes to the descriptive arrays. The descriptive arrays are arrays of character strings, providing textual descriptions of specific air defense unit or aircraft characteristics. Thus, for air defense system 'Kind' with the value 1, the array element Adunit[Kind] returns the string "Vulcan". The two-dimensional factor arrays contain the values for the calculation of the

probabilities of engagement and hit thresholds. Using the fields of the 'ADRECORD' and 'ACRECORD' as indexes, the probability of engagement base value of a 'Kind' of air defense system against a 'Sort' of aircraft is PEBase[Kind, Sort], the location factor for an air defense unit is Placefact[Kind, Location] and the tactic factor of an aircraft is Tactifact[Sort, Tactic]. Elements in the factor arrays may be set by the user, while changing the descriptive array elements must be done by a programmer. Default values for the descriptive and factor arrays are in Appendix G.

Program Flow

There is no prescribed program flow because the program is menu driven, and cross-travel is permitted between levels of menus. The main program consists of a repetitive case statement with single entry and exit points. The main menu provides access to three other menus, each of which calls at least one other menu. The general program flow as seen by the user would be from the Main Menu to the Air Defense Menu, from the Main Menu to the Aircraft Menu and then from the Main Menu to the Engagement procedure. At any point any of the other menus may be used to change or list data used. A sample session is included as Appendix B.

Development System

The development system for this thesis is the NEC Advanced Personal Computer (APC) using TURBO Pascal version 2.0 under the CP/M-86 operating system. The APC uses the 16-bit 8086 microprocessor. The program source code is approximately 60 thousand (60K) 8-bit bytes on a double density disk, while the executable machine code is approximately 34K bytes. Compiler restrictions on the amount of source code require that the program be broken into separate files which are compiled as "include" files from a single program file. The most efficient way to structure the program is to make a separate file for each of the lettered procedures called from the main program. This simplifies editing and compiling.

While the TURBO Pascal compiler supports many non-standard features, it does not support the 'standard' Pascal functions GET and PUT, used for input and output (I/O) of variables of user-defined types. External file operations require non-standard functions that are specific to the compiler and operating system used. As a consequence, the program as written supports no external file I/O. Discussion of the I/O problems and possible solutions are provided in a later section.

Portability Tests

Portability, for the purpose of this thesis, means that the source code and program are machine, Pascal compiler and operating system independent. The first test of the program on another compiler was with Berkeley Pascal version 2.0 under the UNIX (7th Edition) operating system running on a DEC VAX 11/780 minicomputer. Berkeley Pascal has a compiler option that generates warning flags for non-standard Pascal programming. This option was used extensively during the development of the program, though it should be noted that the non-standard use of I/O of typed variables was not flagged as expected. The full program was also run from Apple Pascal on an Apple IIc and TURBO Pascal under CP/M-80 on a Kaypro microcomputer with an 8-bit 8080 microprocessor. The source code for each of the test systems was the same. The major problem that was identified during these tests was a restriction on the maximum size of a single procedure. Particularly for the Apple Pascal compiler, long procedures (75 to 100 lines) had to be separated into smaller procedures. Another problem identified on the Apple system was the requirement for procedure and variable names to be unique in the first eight characters. These Apple Pascal requirements resulted in increased modularity for the program and procedure names of various lengths that are unique in the first eight characters. Files were transported from one computer to another by modem.

Input/Output

The aspect of Pascal that most restricts portability is input/output to files and to the standard I/O devices (terminals in the case of microcomputers). [Lecarme, 1982:24] For this reason, implementation of file I/O is left to the programmer. Included as Appendix E are 'Spare Part' procedures, labeled by system, for a variety of compilers that enable file I/O. The specific uses for files are to save and restore the descriptive data for the Air Defense Unit and Aircraft arrays. The factor data may also be saved to or restored from files if those procedures are replaced. These 'Spare Part' procedures perform identical functions on different systems and are designed for simple substitution in the source code. [Gilb, 1982:213].

Character I/O differences were noted in some of the systems tested. To insure portability to a variety of terminals, all output is restricted to 40 characters per line. Differences in the formatting of output has the minor result of mis-aligning data displayed in tabular form. This was not a significant problem. Character input differences are more important. Major differences in the I/O routines for each compiler mean that input routines with minimal error checking are used.

Error Handling

Ideal error handling for a novice user in this program would include procedures for trapping input in the wrong range or of the wrong type. Alphanumeric range checking is implemented where necessary and the user is re-prompted for input. Type checking is not implemented, as explained in the next paragraph.

As noted above, the differences in the I/O routines for various compilers force minimal error checking in some of the input procedures. The differences noted involved I/O buffering and interpretations of the Pascal end-of-line function. Those differences prevent the implementation of a procedure to interpret all input as characters to be converted to the appropriate type. Attention should be paid to those procedures requiring numeric entry. Range checking and conversion to integer form is performed if a number in a real format is entered when an integer is expected. However, if an alphabetic character is entered when the program is expecting numeric input, the program aborts. All alphabetic input is filtered so that either upper or lower case may be used. Looping in the routines for alphabetic input causes the prompt to be reprinted until a valid letter is entered. On the UNIX version of the program, buffering of the input causes invalid letters typed in response to a prompt to be ignored. The program halts until an appropriate letter is typed. On other systems, the prompt is reprinted as soon as an invalid letter is entered.

Verification

Program flow tests were conducted by extensive use of the program on various systems and by flow tracing through the main program calls. Each of the menu items was selected and control was transferred from menu to menu successfully. Descriptions of air defense units and aircraft were entered and listed. Engagements were conducted between different combinations of elements of the unit and aircraft arrays. Each of the factor arrays were listed, elements changed and engagements re-run. Attrition and suppression probabilities were manipulated to produce different results with the re-seeded random number generator. Several different users participated in the portability and usability tests. Their comments concerning the compilation of the program on other systems and on the type and amount of information provided are reflected in the smaller procedures and help screens. On both the development and portability test systems the program is found to function correctly.

Validation

As this program is a descriptive training model, little correspondence can be made with actual encounters between air defense artillery systems and aircraft. The DUNN-KEMPF war game on which the program is based was used to generate the results of 75 engagements. Dice were rolled for 15 each of the interactions supported by the war game: Vulcan and helicopter, Vulcan and fighter, Chaparral and fighter,

Comparison Results				
<u>75 Interactions:</u>	<u>Engaged</u>	<u>Hit</u>	<u>Suppressed</u>	<u>Destroyed</u>
DUNN-KEMPF	46	18	8	10
ADWGSP	51	21	10	11

Table 1

Redeye and helicopter, and Redeye and fighter. Results were compiled from the table look-ups required for DUNN-KEMPF. Engagements of the same type were generated by the program using the European scenario which DUNN-KEMPF uses. The following values for aircraft factors were used: tactic of fly-over (1.0) and aspect of crossing (1.1). These factors are not reflected in DUNN-KEMPF. Air defense units were given an effectiveness level of 1.0 and suffered no attrition. The results summarized in Table 1 show that approximately the same engagement and hit ratios were achieved with the Air Defense War Game Support Program as with DUNN-KEMPF.

IV. Conclusions and Recommendations

This chapter presents a summary of this thesis, some recommendations for use of the program, and the conclusions drawn.

Summary

A need for a microcomputer model of air defense artillery situations is identified. The specific need is in the training environment for a program that provides the probabilities of engagement and kill for forward-area air defense weapons against a variety of aircraft.

The Air Defense War Game Support Program is designed to provide one-on-one probabilities of engagement and kill for a variety of forward-area air defense systems against different types of aircraft. Each air defense system is described by type (Vulcan, SGT York, Redeye, Stinger, or Chaparral), location (Europe, desert or jungle) and effectiveness level (0.0 to 1.0). Each aircraft is described by type (helicopter, transport or fighter), tactic (pop-up, lay-down or fly-over), and profile or aspect presented to the air defense system (head-on, crossing, or tail). As currently programmed, 20 numbered air defense unit and 20 numbered aircraft descriptions consisting of the above attributes may be entered and the probabilities of engagement and kill calculated for any entered air defense system against any entered aircraft.

The results of an interaction between an air defense unit and an aircraft are determined by comparing a program generated random number with calculated threshold values. Random numbers are generated for each of the engagement, kill, suppression and attrition results. Threshold values are calculated separately for the probabilities of engagement and kill. The threshold values are calculated by multiplying a base value by the series of factors reflecting the descriptions of the air defense unit and the aircraft. The displayed results of an engagement include if the aircraft is engaged, if the aircraft is hit and destroyed or mission suppressed, and if the air defense system suffers any attrition from an aircraft that was engaged but not destroyed or suppressed. Attrition on an air defense unit is reflected by a lowering of the unit's effectiveness level by the amount of the attrition factor.

Within the program the factors used in the calculation of the engagement and kill thresholds may be changed. Additionally, the probabilities of air defense system attrition and aircraft suppression, the attrition factor (that value subtracted from the air defense system effectiveness level) and the random number seed may be changed. Setting the random number seed permits an engagement to be conducted again with unchanged parameters or with new factors.

If file input and output operations are implemented, the data arrays for the air defense units and aircraft, and the arrays for threshold values and factors may be saved to or restored from files.

The program is designed specifically to be used by personnel without a computer background, yet has a structured programming style that permits easy modification by a programmer. The program is written in standard Pascal and, as a result, is highly portable to a variety of compilers and computers. The program has been used on four different computers and compilers without modification by personnel without either computer or air defense backgrounds.

Recommendations

The Air Defense War Game Support Program was designed to be a tool for use at the U.S. Army's Air Defense Artillery School. It is based on the war game, DUNN-KEMPF, currently in use to train junior officers. Designed to replace the tedious process used in that war game to calculate the probabilities of engagement and kill for air defense encounters, this tool is recommended for use at the School. The program capability of generating probabilities of engagement and kill for different scenarios and different weapons systems make it particularly suited for use by an instructor desiring to demonstrate to students differences in deployment and situation specifics.

Forward-area air defense artillery battalions can make use of this program during war games played within the battalion. The program may also be used to supplement higher level, larger war games supported by the battalion. Use by other services is possible though some confusion may be generated by the use of the words suppression and attrition. In the program these refer to aircraft and air defense systems respectively, and were misinterpreted by some Air Force officers using the program.

Enhancements and Expansion. The model programmed reflects the air defense artillery weapons currently in use in the active Army. A logical expansion of the model is to include additional weapons specific to the National Guard, such as the Roland, or specific to the Reserve Army, such as the Duster. A replacement of the currently programmed weapons systems could make the model reflect the weapons of a foreign nation. The model may be enhanced with additional characteristics for the air defense systems or the aircraft. An air defense system could be given a maintenance or ammunition status that expands on the effectiveness level or a weather status that extends the location factor. Aircraft could be given additional characteristics reflecting a specific speed or altitude. Aircraft target selection - whether the aircraft is attacking the air defense unit or another target could be added to the aircraft description as a boolean function. This type of additional factor could impact on the probability of

engagement by assuming that an aircraft is more likely to be engaged by an air defense unit if the aircraft is already firing on the unit.

Enhancements like those suggested would be programmed by extending the records that describe the air defense unit and the aircraft. Additional factor arrays may be added in the main program variable declarations and enabled in the engagement procedure. To permit user modification of additional factor arrays, new levels of menus may be created with access from Parameter Menu 3 which currently has two exit routines.

Conclusion

The Air Defense War Game Support Program is a descriptive model of a one-on-one engagement between a U.S. Army forward-area weapons system and an aircraft. Designed and running on a variety of microcomputers, it is usable by personnel without a computer background. The model is applicable to the current inventory of air defense weapons but flexible enough to reflect changes in either the weapons systems or the threat.

The program is limited in its description of the air defense situation to three generalized scenarios. The descriptions of the air defense systems and aircraft are not detailed and do not directly include significant

characteristics such as aircraft altitude and speed or air defense system detection capabilities. Also not considered in the model are in-range calculations or electronic counter-measures.

The objectives of this thesis were met with the Air Defense War Game Support Program. The program can be expected to run on micro- or larger computers having a Pascal compiler that supports the Pascal instruction set as described in the User Manual and Report. [Jensen and Wirth, 1974] Copies of the program may be obtained on a magnetic medium from the Air Force Institute of Technology or from the author.

APPENDIX A: User's Guide

	Page
Overview	A-2
Starting Up	A-4
Entering Air Defense Systems	A-5
Entering Aircraft	A-8
Conducting Engagements	A-10
Changing Parameters	A-11
Ending the Program.....	A-15

Figures

A-1: Main Menu	A-4
A-2: Air Defense Menu	A-5
A-3: Aircraft Menu	A-8
A-4: Parameter Menu 1	A-11
A-5: Parameter Lists	A-12
A-6: Parameter Menu 2	A-13
A-7: Parameter Menu 3	A-14

U S E R ' S G U I D E

Overview

The Air Defense War Game Support Program is designed to provide probabilities of engagement and kill for a variety of forward air defense systems against different types of aircraft. Each air defense system is described by type (Vulcan, SGT York, Redeye, Stinger, or Chaparral), location (Europe, Desert or Jungle) and effectiveness level (0.0 to 1.0). Each aircraft is described by type (Helicopter, Transport or Fighter), tactic (Pop-up, Lay-Down or Fly-Over), and profile or aspect presented to the air defense system (Head-on, Crossing, or Tail). As currently programmed, 20 numbered Air Defense Unit and 20 numbered Aircraft descriptions consisting of the above attributes may be entered and the probabilities of engagement and kill calculated for any entered air defense system against any entered aircraft.

The results of an interaction between an Air Defense Unit and an Aircraft are determined by comparing a program generated random number with calculated threshold values. Random numbers are generated for each of the engagement, kill, suppression and attrition results. Threshold values are calculated separately for the probabilities of engagement and kill. The threshold values are calculated by multiplying a base value by the series of factors reflecting

the descriptions of the Air Defense Unit and the Aircraft. The displayed results of an engagement include if the aircraft is engaged, if the aircraft is hit and destroyed or mission suppressed, and if the air defense system suffers any attrition from an aircraft that was engaged but not destroyed or suppressed. Attrition on an Air Defense Unit is reflected by a lowering of the Unit's effectiveness level by the amount of the attrition factor.

Within the program the factors used in the calculation of the engagement and kill thresholds may be changed. Additionally, the probabilities of air defense system attrition and aircraft suppression, the attrition factor (that value subtracted from the air defense system effectiveness level) and the random number seed may be changed. Resetting the random number seed to a previous value permits an engagement to be conducted again with unchanged parameters or with new factors. Each of the factors mentioned above may be changed from one of the parameter menus.

If file input and output operations are implemented, the data arrays for the Air Defense Units and Aircraft, and the arrays for threshold values and factors may be saved to or restored from files. Use of external files for storage of the data arrays permits several sets of data to be entered and changed during a program run without having to enter the individual values each time a new data set is desired.

```
-----  
M A I N M E N U  
  
A : DISPLAY AIR DEFENSE MENU  
B : DISPLAY AIRCRAFT MENU  
C : CONDUCT ENGAGEMENTS  
D : DISPLAY PARAMETER MENU  
E : END PROGRAM  
  
Choose A, B, C, D, E, or H for Help >  
-----
```

Figure A-1

Starting Up

When the program is started no air defense systems or aircraft are available for engagements but a set of default values for the calculation of engagement results is loaded. The default values are listed in Appendix G. After the program sign-on, the user is presented with the Main Menu (Figure A-1). From this menu either option A or B is selected to proceed with the descriptions of air defense systems and aircraft. Selection of option D allows the default factors to be displayed or changed.

```
-----  
A I R D E F E N S E M E N U  
  
A : ENTER DESCRIPTION OF  
AIR DEFENSE UNITS  
  
B : LIST CURRENTLY ENTERED  
AIR DEFENSE UNITS  
  
C : SAVE DATA  
  
D : RESTORE DATA  
  
E : EXIT TO MAIN MENU  
  
Choose A, B, C, D, E, or H for Help >  
-----
```

Figure A-2

Entering Air Defense Systems

Option A from the Main Menu brings up the Air Defense Menu (Figure A-2). From this menu, options A or D are selected to enter Air Defense Unit descriptions. Option D may only be used if the file input/output procedures are implemented and the program has been run previously and the Air Defense Unit descriptions saved. When option A, Enter Description of Air Defense Units, is selected, the user is told what the next available (undescribed) Unit number is and asked to enter the number of the Unit to describe. There is no requirement to enter Air Defense Units in any order, but entering the number of a Unit that was previously described will result in the new information overwriting the previous description. There is also no requirement to enter any set amount of Air Defense Units - a single Unit may be

entered and an engagement conducted with an entered Aircraft. A list of the currently entered Air Defense Units is seen by selecting option B of the Air Defense Menu, List Currently Entered Air Defense Units. Entering an Air Defense Unit number of 0 returns the user to the Air Defense Menu.

Options on the Air Defense Menu include C, Save Data and D, Restore Data. If the program has been run previously and option C, Save Data, was used, descriptions of Air Defense Units may be entered from the external file ADSAVE. Two items of caution: if the file ADSAVE does not exist, the program aborts to the operating system, and if data is restored from the file ADSAVE, any descriptions of Air Defense Units currently in use are overwritten. In like manner, data saved overwrites the data currently in the file ADSAVE. The user may make a decision not to continue with saving or restoring the Air Defense Unit data at which time the program returns to the Air Defense Menu.

The process of describing an Air Defense Unit is a matter of selecting the appropriate items from the subsequent menus. The initial choice is for the type of system, the second choice is for the location of the system. The third choice, effectiveness level, may be thought of as a decimal percent of effectiveness, so if the Unit is considered 100% effective, 1 or 1.0 is entered for the effectiveness level. In like manner, a 75% effective Unit

is described with an effectiveness level of 0.75. Caution should be exercised to enter only numbers for the effectiveness level, as a non-numeric entry causes the program to abort to the operating system.

When a description is completed, a Unit number of 0 is entered to return to the Air Defense Menu. Option B, List Currently Entered Air Defense Units, may be made to verify that the correct information has been entered. Incorrect information is corrected by selecting option A at the Air Defense Menu, entering the Unit number for which the information was incorrect, and reselecting the menu items to complete the description of the Unit.

```
-----  
A I R C R A F T M E N U  
  
A : ENTER DESCRIPTION OF AIRCRAFT  
B : LIST CURRENTLY ENTERED AIRCRAFT  
C : SAVE DATA  
D : RESTORE DATA  
E : EXIT TO MAIN MENU  
  
Choose A, B, C, D, E, or H for Help >  
-----
```

Figure A-3

Entering Aircraft

Option B from the Main Menu brings up the Aircraft Menu (Figure A-3). From this menu, options A or D are selected to enter Aircraft descriptions. Option D may only be used if the file input/output procedures are implemented and the program has been run previously and the Aircraft descriptions saved. When option A, Enter Description of Aircraft, is selected, the user is told what the next available (undescribed) Aircraft number is and asked to enter the number of the Aircraft to describe. There is no requirement to enter Aircraft in any order, but entering the number of an Aircraft that was previously described results in the new information overwriting the previous description. There is also no requirement to enter any set amount of Aircraft - a single Aircraft may be entered and an engagement conducted with an entered Air Defense Unit. A

list of the currently entered Aircraft is seen by selecting option B of the Aircraft, List Currently Entered Aircraft. Entering an Aircraft number of 0 returns the user to the Aircraft Menu.

Options on the Aircraft Menu include C, Save Data and D, Restore Data. If the program has been run previously and option C, Save Data, was used, descriptions of Aircraft may be entered from the external file ACSAVE. Two items of caution: if the file ACSAVE does not exist, the program aborts to the operating system, and if data is restored from the file ACSAVE, any descriptions of Aircraft currently in use are overwritten. In like manner, data saved overwrites the data currently in the file ACSAVE. The user may make a decision not to continue with saving or restoring the Aircraft data at which time the program returns to the Aircraft Menu.

The process of describing an Aircraft is a matter of selecting the appropriate items from the subsequent menus. The initial choice is for the type of aircraft, the second choice is for the tactic used and the third choice is for the aspect the Aircraft is presenting to the Air Defense Unit. When a description is completed, an Aircraft number of 0 is entered to return to the Air Defense Menu. Option B, List Currently Entered Aircraft, may be made to verify that the correct information has been entered. Incorrect information is corrected by selecting option A on the

Aircraft Menu, entering the Aircraft number for which the information was incorrect, and reselecting the menu items to complete the description of the Aircraft.

Conducting Engagements

From the Main Menu, option C, Conduct Engagements, is selected after both Air Defense Units and Aircraft are entered. The user is prompted for the Air Defense Unit and the Aircraft numbers for the engagement. Only numbers for Units and Aircraft that have been described or 0 to exit are accepted. If an invalid number is entered, a list of valid numbers is displayed and the prompt redisplayed. When valid numbers for the Air Defense Unit and the Aircraft are entered, characteristics of both systems are listed and the results of the engagement are displayed. The first result is whether the Aircraft was engaged by the Air Defense Unit. If the Aircraft was not engaged, the engagement process is finished and the user is asked if another engagement is to be conducted. If the Aircraft was engaged, whether or not the Aircraft was hit is displayed, and, if the Aircraft was hit, whether it was destroyed or its mission aborted is determined. If the Aircraft was engaged but not hit, whether or not the Air Defense Unit was suppressed is displayed, and if the Unit was suppressed, the effectiveness is decreased by the amount of the attrition factor.

P A R A M E T E R M E N U 1

- A : LIST CURRENT PARAMETERS
- B : SET/SAVE PARAMETERS
- C : SET RANDOM NUMBER SEED
- D : SET ATTRITION/SUPPRESSION FACTORS
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help >

Figure A-4

When the engagement process is concluded and the results displayed, the user is given the option of conducting another engagement. If no other engagement is to be conducted, the user is returned to the Main Menu.

Changing Parameters

Option D from the Main Menu brings up the Parameter Menu (Figure A-4). Parameters refer to the base values for the probabilities of engagement and hit, and the factor values for each of the Air Defense Unit and Aircraft descriptors (with the exception of the Air Defense Unit effectiveness level). Other values that may be changed include the probabilities of Air Defense Unit attrition and Aircraft suppression, the attrition factor and the seed for the random number generator.

P A R A M E T E R L I S T S

A : ENGAGEMENT PROBABILITY BASE

B : HIT PROBABILITY BASE

C : LOCATION FACTOR

D : TACTIC FACTOR

E : ASPECT FACTOR

Enter letter for parameters to display
Choose A, B, C, D, E, or X to Exit >

Figure A-5

Option A from Parameter Menu 1 brings up the Parameter Lists (Figure A-5) from which a set of threshold values or description factors may be selected to be listed. Once a set of values is displayed, the user is asked if another set is to be displayed. If the Parameter Lists are showing and the user does not choose to continue, option X to Exit returns the user to the previous menu.

P A R A M E T E R M E N U 2

- A : CHANGE PARAMETERS
- B : SAVE BASE VALUES
- C : RESTORE BASE VALUES
- D : SAVE FACTORS
- E : RESTORE FACTORS

Choose A, B, C, D, E, or X to Exit >

Figure A-6

Selection of option B (Set/Save Parameters) from Parameter Menu 1 brings up Parameter Menu 2 (Figure A-6). Option A, Change Parameters, displays the Parameter Lists. Selection of a parameter list allows values used by the program to be changed by specifying the row and column of value to change, and the new value. The new value is displayed and the user given the option of changing another value. Options at Parameter Menu 2 also include B, Save Base Values; C, Restore Base Values; D, Save Factors and E, Restore Factors. If the program has been run previously and the save options were used, base values may be restored from the file BASAVE and factor values may be restored from the file FASAVE. Two items of caution: if the files do not exist, the program aborts to the operating system; and if data is restored from the files, any data currently in use is overwritten. In like manner, data saved overwrites the

P A R A M E T E R M E N U 3

- A : SET ATTRITION FACTOR
- B : SET ATTRITION PROBABILITY
- C : SET SUPPRESSION PROBABILITY
- D : EXIT TO PARAMETER MENU 1
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help >

Figure A-7

data currently in the files. The user may make a decision not to continue with saving or restoring the data at which time the program returns to Parameter Menu 2. If Parameter Menu 2 is showing and the user does not choose to continue, option X to Exit returns the user to the previous menu.

Selection of option C, Set Random Number Seed or D, Set Attrition/Suppression Factors from Parameter Menu 1 permits those values to be changed. Option D brings up Parameter Menu 3 (Figure A-7).

For each option on Parameter Menu 3 and option C, Set Random Number Seed, from Parameter Menu 1, the current and original factors and permissible range for a new value are displayed.

Ending The Program

The program is ended from the Main Menu by selecting option E, Exit Program.

APPENDIX B: Sample Session

AIR DEFENSE
WAR GAME
SUPPORT PROGRAM

Press Return or Enter to Continue

Screen 1

This sample session with the Air Defense War Game Support Program demonstrates the sequence of entering and listing the Air Defense Units and Aircraft, conducting an engagement, listing and changing the program parameters, and conducting an engagement with the new parameters. User input in the following screens has been underlined. When the program is started, the initial screen displays the program name and prompts the user to continue.

M A I N M E N U

- A : DISPLAY AIR DEFENSE MENU
- B : DISPLAY AIRCRAFT MENU
- C : CONDUCT ENGAGEMENTS
- D : DISPLAY PARAMETER MENU 1
- E : END PROGRAM

Choose A, B, C, D, E, or H for Help > a

Screen 2

The Main Menu allows the transfer of program control to the Air Defense, Aircraft or Parameter Menus, or to the procedure for conducting an engagement. In this case, Choice A, selection of the Air Defense Menu is made. This is normally the first step in the program.

A I R D E F E N S E M E N U

- A : ENTER DESCRIPTION OF
AIR DEFENSE UNITS
- B : LIST CURRENTLY ENTERED
AIR DEFENSE UNITS
- C : SAVE DATA
- D : RESTORE DATA
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > a

Screen 3

From the Air Defense Menu, the Air Defense Units may be entered, listed and saved to or restored from a file. Saving or restoring data from files is operating system dependent and must be implemented by the user. When the program is initialized, no Air Defense Units are available for engagements. In this screen, selection A is made to enter Air Defense Units.

Air Defense Unit Descriptions

Next available Unit number: 1

Enter Air Defense Unit number
between 1 and 20
or 0 to exit > 1

Screen 4

The Air Defense Units are identified by a Unit number ranging from 1 to the maximum number the program permits. In the above screen the maximum number of Air Defense Units permitted is 20 and the first available Unit number is 1. A Unit number is available if no Air Defense Unit has been entered for that number. If all the the numbers in the permissible range have an active Air Defense Unit (one that has been entered as in the succeeding screens) associated with it, the message 'Air Defense Unit array is full' is displayed. Attempts to enter a number outside the

permissible range results in the prompt being printed until a correct number is entered. Any number in the permissible range may be entered - there is no requirement to enter the number which is displayed as available, but entering a number for a Unit that has already been described results in the old Unit description being overwritten. Entering 0 indicates that no (more) Air Defense Units are to be entered and control is returned to the Main Menu.

Air Defense Unit Descriptions

Specify Air Defense Unit Type

- 1 : Vulcan
- 2 : SGT York
- 3 : Redeye
- 4 : Stinger
- 5 : Chaparral

Enter Air Defense Unit type > 1

Screen 5

Selection is made of the Air Defense Unit type by entering the number corresponding to the desired type. In this case, Air Defense Unit type Vulcan is selected.

Air Defense Unit Descriptions

Specify Air Defense Unit Location

1 : Europe

2 : Desert

3 : Jungle

Enter Air Defense Unit location > 1

Screen 6

Selection is made of the Air Defense Unit location by entering the number corresponding to the desired location. In this case, Air Defense Unit location Europe is selected.

Air Defense Unit Descriptions

Specify Air Defense Unit Effectiveness

Enter the decimal effectiveness level
of the Air Defense Unit (0.00 - 1.00) > 1.0

Screen 7

Determination is made of the Air Defense Unit effectiveness level by entering the number corresponding to the desired effectiveness level. In this case, Air Defense Unit effectiveness level 1.0 is specified indicating that the Unit is operating at its peak effectiveness.

Air Defense Unit Descriptions

Next available Unit number: 2

Enter Air Defense Unit number
between 1 and 20
or 0 to exit > 0

Screen 8

When an Air Defense Unit has been entered the user is given the choice of entering another Unit description or finishing the description process. The program displays the number of the next available Unit to describe. In this case, no more Air Defense Units are to be described and 0 is entered.

A I R D E F E N S E M E N U

- A : ENTER DESCRIPTION OF
AIR DEFENSE UNITS
- B : LIST CURRENTLY ENTERED
AIR DEFENSE UNITS
- C : SAVE DATA
- D : RESTORE DATA
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > b

Screen 9

From the Air Defense Unit description process, control is returned to the Air Defense Menu. Selection B is made to list the currently entered Air Defense Units.

Air Defense Unit Descriptions

Number	Type	Location	Effectiveness
1	Vulcan	Europe	1.00

Press Return or Enter to Continue

Screen 10

The list of the currently entered Air Defense Units is made in a table format. If more than 10 Units have been entered, the first 10 are displayed and the user is prompted to press Return or Enter to continue. When all of the entered Units have been displayed the user is prompted to continue. When Return or Enter is pressed, control is returned to the Air Defense Menu.

A I R D E F E N S E M E N U

- A : ENTER DESCRIPTION OF
AIR DEFENSE UNITS
- B : LIST CURRENTLY ENTERED
AIR DEFENSE UNITS
- C : SAVE DATA
- D : RESTORE DATA
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > e

Screen 11

After the Air Defense Units have been entered and listed (if desired) selection E is made to return to the Main Menu.

M A I N M E N U

- A : DISPLAY AIR DEFENSE MENU
- B : DISPLAY AIRCRAFT MENU
- C : CONDUCT ENGAGEMENTS
- D : DISPLAY PARAMETER MENU 1
- E : END PROGRAM

Choose A, B, C, D, E, or H for Help > b

Screen 12

Menu Choice B is selected to transfer to the Aircraft Menu in order to enter Aircraft.

A I R C R A F T M E N U

- A : ENTER DESCRIPTION OF AIRCRAFT
- B : LIST CURRENTLY ENTERED AIRCRAFT
- C : SAVE DATA
- D : RESTORE DATA
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > a

Screen 13

From the Aircraft Menu, the Aircraft may be entered, listed and saved to or restored from a file. Saving or restoring data from files is operating system dependent and must be implemented by the user. When the program is initialized, no Aircraft are available for engagements. In this screen, selection A is made to enter Aircraft.

Aircraft Descriptions

Next available aircraft number is 1

Enter Aircraft number
between 1 and 20
or 0 to exit > 1

Screen 14

The Aircraft are identified by a number ranging from 1 to the maximum number the program permits. In the above screen the maximum number of Aircraft permitted is 20 and the first available Unit number is 1. An Aircraft number is available if no Aircraft has been entered for that number. If all the numbers in the permissible range have an active Aircraft (one that has been entered as in the succeeding screens) associated with it, the message 'Aircraft array is full' is displayed. Attempts to enter a number outside the permissible range results in the prompt being printed until

a correct number is entered. Any number in the permissible range may be entered - there is no requirement to enter the number which is displayed as available, but entering a number for an Aircraft that has already been described results in the old Aircraft description being overwritten. Entering 0 indicates that no (more) Aircraft are to be entered and control is returned to the Main Menu.

Aircraft Descriptions

Specify Aircraft Type

1 : Helicopter

2 : Transport

3 : Fighter

Enter Aircraft type > 1

Screen 15

Selection is made of the Aircraft type by entering the number corresponding to the desired type. In this case, Aircraft type Helicopter is selected.

Aircraft Descriptions

Specify Aircraft Tactic

- 1 : Pop-Up
- 2 : Lay-Down
- 3 : Fly-Over

Enter Aircraft tactic > 1

Screen 16

Selection is made of the Aircraft tactic by entering the number corresponding to the desired tactic. In this case, Aircraft tactic Pop-Up is selected.

Aircraft Descriptions

Specify Aircraft Aspect

- 1 : Head-On
- 2 : Crossing
- 3 : Tail

Enter Aircraft aspect > 1

Screen 17

Selection is made of the Aircraft aspect by entering the number corresponding to the desired aspect. In this case, Aircraft aspect Head-On is selected.

Aircraft Descriptions

Next available aircraft number is 2

Enter Aircraft number
between 1 and 20
or 0 to exit > 0

Screen 18

When an Aircraft has been entered the user is given the choice of entering another Aircraft description or finishing the description process. The program displays the number of the next available Aircraft to describe. In this case, no more Aircraft are to be described and 0 is entered.

A I R C R A F T M E N U

- A : ENTER DESCRIPTION OF AIRCRAFT
- B : LIST CURRENTLY ENTERED AIRCRAFT
- C : SAVE DATA
- D : RESTORE DATA
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > b

Screen 19

From the Aircraft description process, control is returned to the Aircraft Menu. Selection B is made to list the currently entered Aircraft.

Aircraft Descriptions

Number	Type	Tactic	Aspect
1	Helicopter	Pop-Up	Head-On

Press Return or Enter to Continue

Screen 20

The list of the currently entered Aircraft is made in a table format. If more than 10 Aircraft have been entered, the first 10 are displayed and the user prompted to press Return or Enter to continue. When all of the entered Aircraft have been displayed the user is prompted to continue. When Return or Enter is pressed, control is returned to the Aircraft Menu.

A I R C R A F T M E N U

- A : ENTER DESCRIPTION OF AIRCRAFT
- B : LIST CURRENTLY ENTERED AIRCRAFT
- C : SAVE DATA
- D : RESTORE DATA
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > e

Screen 21

After the Aircraft have been entered and listed (if desired) selection E is made to return to the Main Menu.

M A I N M E N U

- A : DISPLAY AIR DEFENSE MENU
- B : DISPLAY AIRCRAFT MENU
- C : CONDUCT ENGAGEMENTS
- D : DISPLAY PARAMETER MENU 1
- E : END PROGRAM

Choose A, B, C, D, E, or H for Help > C

Screen 22

Once both Air Defense Units and Aircraft have been described, an engagement is conducted by selecting C, Conduct Engagements.

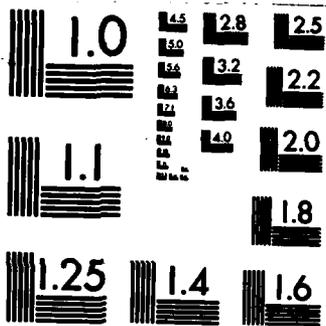
ENGAGEMENTS

Enter Air Defense Unit number
between 1 and 20
or 0 to exit > 1

Enter Aircraft number
between 1 and 20
or 0 to exit > 1

Screen 23

The user is asked for the number of the Air Defense Unit and the Aircraft for the engagement. In this case, Air Defense Unit number 1 and Aircraft 1 are selected for the engagement. Entering the number of a Unit or Aircraft that has not been described results in the message 'That Air Defense Unit/Aircraft has not been entered' and a list of the valid Unit or Aircraft numbers is displayed. The user is again asked for a Unit/Aircraft number or 0 to exit.



MICROCOPY RESOLUTION TEST CHART
 NATIONAL BUREAU OF STANDARDS-1963-A

ENGAGEMENT RESULTS

Air Defense Unit: 1
AD System Type: Vulcan
Location: Europe
Effectiveness: 1.00
Aircraft: 1
Aircraft Type: Helicopter
Tactic: Pop-Up
Aspect: Head-On

>> Aircraft was engaged.
Random number: 1 Threshold: 67.5

>> Aircraft was not hit.
Random number: 75 Threshold: 40.5

>> Air Defense Unit was suppressed.
>> Effectiveness reduced by 0.00.

Conduct another engagement?
Enter y or Y for yes, n or N for no > n

Screen 24

The engagement results are preceded by the descriptions of the Air Defense Unit and Aircraft. Each result is accompanied by the random number and threshold value that were used. Another engagement is conducted by responding Y.

M A I N M E N U

- A : DISPLAY AIR DEFENSE MENU
- B : DISPLAY AIRCRAFT MENU
- C : CONDUCT ENGAGEMENTS
- D : DISPLAY PARAMETER MENU 1
- E : END PROGRAM

Choose A, B, C, D, E, or H for Help > d

Screen 25

If the decision is made to change or view the program parameters, option D from the Main Menu is chosen.

P A R A M E T E R M E N U

- A : LIST CURRENT PARAMETERS
- B : SET/SAVE PARAMETERS
- C : SET RANDOM NUMBER SEED
- D : SET ATTRITION/SUPPRESSION FACTORS
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > a

Screen 26

Choice A is made to list the program parameters.

P A R A M E T E R L I S T S

- A : ENGAGEMENT PROBABILITY BASES
- B : HIT PROBABILITY BASES
- C : LOCATION FACTOR
- D : TACTIC FACTOR
- E : ASPECT FACTOR

Enter letter for parameters to display
Choose A, B, C, D, E, or X to Exit > a

Screen 27

Selection A is made to display the base values for the
engagement probabilities.

Probability of Engagement Base Values

	Helicopter	Transport	Fighter
Vulcan	0.83	0.67	0.50
SGT York	0.90	0.70	0.60
Redeye	0.50	0.67	0.50
Stinger	0.60	0.70	0.60
Chaparral	0.50	0.67	0.50

Continue to display parameters?
Enter y or Y for yes, n or N for no > y

Screen 28

The base values for the probability of engagement are displayed and the user is asked if another set of parameters is to be displayed.

P A R A M E T E R L I S T S

- A : ENGAGEMENT PROBABILITY BASES
- B : HIT PROBABILITY BASES
- C : LOCATION FACTOR
- D : TACTIC FACTOR
- E : ASPECT FACTOR

Enter letter for parameters to display
Choose A, B, C, D, E, or X to Exit > b

Screen 29

Selection B is made to display the base values for the
hit probabilities.

Probability of Hit Base Values

	Helicopter	Transport	Fighter
Vulcan	0.50	0.50	0.33
SGT York	0.60	0.60	0.40
Redeye	0.17	0.33	0.33
Stinger	0.20	0.40	0.40
Chaparral	0.33	0.33	0.33

Continue to display parameters?
Enter y or Y for yes, n or N for no > y

Screen 30

The base values for the probability of hit are displayed and the user is asked if another set of parameters is to be displayed.

P A R A M E T E R L I S T S

- A : ENGAGEMENT PROBABILITY BASES
- B : HIT PROBABILITY BASES
- C : LOCATION FACTOR
- D : TACTIC FACTOR
- E : ASPECT FACTOR

Enter letter for parameters to display
Choose A, B, C, D, E, or X to Exit > C

Screen 31

Selection C is made to display the values for the
location factors.

Location Factors

	Europe	Desert	Jungle
Vulcan	1.00	1.10	0.80
SGT York	1.00	1.10	0.80
Redeye	1.00	1.10	0.80
Stinger	1.00	1.10	0.80
Chaparral	1.00	1.10	0.80

Continue to display parameters?
Enter y or Y for yes, n or N for no > y

Screen 32

The values for the location factor are displayed and the user is asked if another set of parameters is to be displayed.

P A R A M E T E R L I S T S

- A : ENGAGEMENT PROBABILITY BASES
- B : HIT PROBABILITY BASES
- C : LOCATION FACTOR
- D : TACTIC FACTOR
- E : ASPECT FACTOR

Enter letter for parameters to display
Choose A, B, C, D, E, or X to Exit > d

Screen 33

Selection D is made to display the base values for the
tactic factors.

Tactics Factors

	Pop-Up	Lay-Down	Fly-Over
Vulcan	0.90	1.00	1.10
SGT York	0.90	1.00	1.10
Redeye	0.90	1.00	1.10
Stinger	0.90	1.00	1.10
Chaparral	0.90	1.00	1.10

Continue to display parameters?
Enter y or Y for yes, n or N for no > y

Screen 34

The values for the tactic factor are displayed and the user is asked if another set of parameters is to be displayed.

P A R A M E T E R L I S T S

- A : ENGAGEMENT PROBABILITY BASES
- B : HIT PROBABILITY BASES
- C : LOCATION FACTOR
- D : TACTIC FACTOR
- E : ASPECT FACTOR

Enter letter for parameters to display
Choose A, B, C, D, E, or X to Exit > e

Screen 35

Selection E is made to display the values for the
aspect factors.

Aspect Factors

	Head-On	Crossing	Tail
Vulcan	0.90	1.10	0.90
SGT York	0.90	1.10	0.90
Redeye	0.90	1.10	0.90
Stinger	0.90	1.10	0.90
Chaparral	0.90	1.10	0.90

Continue to display parameters?
Enter y or Y for yes, n or N for no > n

Screen 36

The values for the aspect factor are displayed and the user is asked if another set of parameters is to be displayed. Selection N is made to exit and continue with the program.

P A R A M E T E R M E N U

- A : LIST CURRENT PARAMETERS
- B : SET/SAVE PARAMETERS
- C : SET RANDOM NUMBER SEED
- D : SET ATTRITION/SUPPRESSION FACTORS
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > b

Screen 37

Selection B is made to set or save parameter values.

P A R A M E T E R M E N U 2

A : CHANGE PARAMETERS

B : SAVE BASES

C : RESTORE BASES

D : SAVE FACTORS

E : RESTORE FACTORS

Choose A, B, C, D, E, or H for Help > a

Screen 38

Selection A is made to change a parameter value.

P A R A M E T E R L I S T S

- A : ENGAGEMENT PROBABILITY BASES
- B : HIT PROBABILITY BASES
- C : LOCATION FACTOR
- D : TACTIC FACTOR
- E : ASPECT FACTOR

Enter letter for parameters to change
Choose A, B, C, D, E, or X to Exit > a

Screen 39

One of the values for the probability of engagement
base table is to be changed.

Probability of Engagement Base Values

	Helicopter	Transport	Fighter
Vulcan	0.83	0.67	0.50
SGT York	0.90	0.70	0.60
Redeye	0.50	0.67	0.50
Stinger	0.60	0.70	0.60
Chaparral	0.50	0.67	0.50

Enter row of factor to change > 1
Enter column of factor to change > 1
Enter new value > 1.0

Screen 40

A table value is changed by entering the numbers of the row, 1, and column, 1, of the value to change. The new value, 1.0, is then entered.

Probability of Engagement Base Values

	Helicopter	Transport	Fighter
Vulcan	1.00	0.67	0.50
SGT York	0.90	0.70	0.60
Redeye	0.50	0.67	0.50
Stinger	0.60	0.70	0.60
Chaparral	0.50	0.67	0.50

Change another value?

Enter y or Y for yes, n or N for no > n

Continue to change parameters?

Enter y or Y for yes, n or N for no > n

Screen 41

The table is re-displayed with the new value in place and the user asked if any more values are to be changed or if values in another table are to be changed.

P A R A M E T E R M E N U

- A : LIST CURRENT PARAMETERS
- B : SET/SAVE PARAMETERS
- C : SET RANDOM NUMBER SEED
- D : SET ATTRITION/SUPPRESSION FACTORS
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > d

Screen 42

Selection D is made to set the attrition or suppression factors. This transfers control to Parameter Menu 3.

P A R A M E T E R M E N U 3

- A : SET ATTRITION FACTOR
- B : SET ATTRITION PROBABILITY
- C : SET SUPPRESSION PROBABILITY
- D : EXIT TO PARAMETER MENU 1
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > a

Screen 43

Selection A is made to set the attrition factor.

SET ATTRITION FACTOR

Current attrition factor: 0.00
Initial attrition factor: 0.00

The attrition factor must be a decimal between 0.00 and 1.00. This value may be subtracted from the Air Defense Unit level of effectiveness as a result of an engagement if the Aircraft is not suppressed or destroyed.

Enter new attrition factor > 0.01
New attrition factor: 0.01

Press Return or Enter to Continue

Screen 44

The original and current values of the attrition factor are displayed as is the permissible range for a new value. The new value, 0.01, is entered. The program displays the new value and waits for user input before continuing.

P A R A M E T E R M E N U 3

- A : SET ATTRITION FACTOR
- B : SET ATTRITION PROBABILITY
- C : SET SUPPRESSION PROBABILITY
- D : EXIT TO PARAMETER MENU 1
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > b

Screen 45

Selection b is made to set the probability of attrition.

SET PROBABILITY OF ATTRITION

Current probability of attrition: 0.50
Initial probability of attrition: 0.50

Probability of attrition must be
a decimal between 0.00 and 1.00
This value is used to determine
if the Air Defense Unit level of
effectiveness will be degraded.

Enter new probability of attrition > 0.6
New probability of attrition: 0.60

Press Return or Enter to Continue

Screen 46

The original and current values of the probability of attrition are displayed as is the permissible range for a new value. The new value, 0.6, is entered. The program displays the new value and waits for user input before continuing.

P A R A M E T E R M E N U 3

- A : SET ATTRITION FACTOR
- B : SET ATTRITION PROBABILITY
- C : SET SUPPRESSION PROBABILITY
- D : EXIT TO PARAMETER MENU 1
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > c

Screen 47

Selection c is made to set the probability of suppression.

SET PROBABILITY OF SUPPRESSION

Current probability of suppression: 0.50
Initial probability of suppression: 0.50

Probability of suppression must be
a decimal between 0.00 and 1.00
This value is used to determine if
the Aircraft will be suppressed or
destroyed after being engaged.

Enter new probability of suppression > 0.6
New probability of suppression: 0.60

Press Return or Enter to Continue

Screen 48

The original and current values of the probability of suppression are displayed as is the permissible range for a new value. The new value, 0.6, is entered. The program displays the new value and waits for user input before continuing.

P A R A M E T E R M E N U 3

- A : SET ATTRITION FACTOR
- B : SET ATTRITION PROBABILITY
- C : SET SUPPRESSION PROBABILITY
- D : EXIT TO PARAMETER MENU 1
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > d

Screen 49

Selection D is made to return to Parameter Menu 1.

P A R A M E T E R M E N U

- A : LIST CURRENT PARAMETERS
- B : SET/SAVE PARAMETERS
- C : SET RANDOM NUMBER SEED
- D : SET ATTRITION/SUPPRESSION FACTORS
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > C

Screen 50

Selection C is made to set the random number seed.

SET RANDOM NUMBER SEED

Current random number seed: 573259288
Initial random number seed: 12347

The random number seed must be
an integer between 0 and 2147483647

Enter new random number seed > 12347
New random number seed: 12347

Press Return or Enter to Continue

Screen 51

The original and current values for the random number seed are displayed, as is the permissible range for a new value. In this case, the random number seed is reset to the original value in order to repeat the first engagement with the new attrition and suppression probabilities.

P A R A M E T E R M E N U

- A : LIST CURRENT PARAMETERS
- B : SET/SAVE PARAMETERS
- C : SET RANDOM NUMBER SEED
- D : SET ATTRITION/SUPPRESSION FACTORS
- E : EXIT TO MAIN MENU

Choose A, B, C, D, E, or H for Help > e

Screen 52

Selection E is made to return to the Main Menu.

M A I N M E N U

- A : DISPLAY AIR DEFENSE MENU
- B : DISPLAY AIRCRAFT MENU
- C : CONDUCT ENGAGEMENTS
- D : DISPLAY PARAMETER MENU 1
- E : END PROGRAM

Choose A, B, C, D, E, or H for Help > ε

Screen 53

Another engagement is to be run with the new probabilities of attrition and suppression.

ENGAGEMENTS

Enter Air Defense Unit number
between 1 and 20
or 0 to exit > 1

Enter Aircraft number
between 1 and 20
or 0 to exit > 1

Screen 54

The Air Defense Unit and Aircraft numbers are entered
as before.

ENGAGEMENT RESULTS

Air Defense Unit: 1
AD System Type: Vulcan
Location: Europe
Effectiveness: 1.00
Aircraft: 1
Aircraft Type: Helicopter
Tactic: Pop-Up
Aspect: Head-On

>> Aircraft was engaged.
Random number: 1 Threshold: 81.0

>> Aircraft was not hit.
Random number: 75 Threshold: 40.5

>> Air Defense Unit was suppressed.
>> Effectiveness reduced by 0.01.

Conduct another engagement?
Enter y or Y for yes, n or N for no > y

Screen 55

The random number generator, having been re-seeded with the original value, returns the same numbers as for the initial engagement. The engagement threshold reflects the higher value caused by increasing the base value to 1.0 in a previous screen, and the suppression of the air defense system reflects the higher value for the probability of suppression. The attrition factor can now be seen as the value of 0.01 that was also set in a previous screen.

ENGAGEMENTS

Enter Air Defense Unit number
between 1 and 20
or 0 to exit > 1

Enter Aircraft number
between 1 and 20
or 0 to exit > 1

Screen 56

Another engagement with the same Unit and Aircraft is run to examine the effect of the attrition suffered by the Air Defense Unit.

ENGAGEMENT RESULTS

Air Defense Unit: 1
AD System Type: Vulcan
Location: Europe
Effectiveness: 0.99
Aircraft: 1
Aircraft Type: Helicopter
Tactic: Pop-Up
Aspect: Head-On

>> Aircraft was not engaged.
Random number: 92 Threshold: 80.2

Conduct another engagement?
Enter y or Y for yes, n or N for no > n

Screen 57

The engagement results reflect the lowered effectiveness level of the Air Defense Unit. The engagement threshold, a product of effectiveness level and other parameters is also lower. The different result in this case is primarily due to the higher random number generated, though a higher attrition factor could have decreased the engagement threshold considerably. Selection is made to terminate the engagement sequences.

M A I N M E N U

A : DISPLAY AIR DEFENSE MENU

B : DISPLAY AIRCRAFT MENU

C : CONDUCT ENGAGEMENTS

D : DISPLAY PARAMETER MENU 1

E : END PROGRAM

Choose A, B, C, D, E, or H for Help > e

Screen 58

Selection E is made to end the program.

AIR DEFENSE
WAR GAME
SUPPORT PROGRAM

Screen 59

B-62

APPENDIX C: PROGRAM FLOW

Chart	Page
C-1: Main Menu Selections	C-2
C-2: Air Defense Menu Selections	C-3
C-3: Aircraft Menu Selections	C-4
C-4: Parameter Menu 1 Selections	C-5
C-5: Parameter Menu 2 Selections	C-6
C-5: Parameter Menu 3 Selections	C-7

Chart C-1: Main Menu Selections

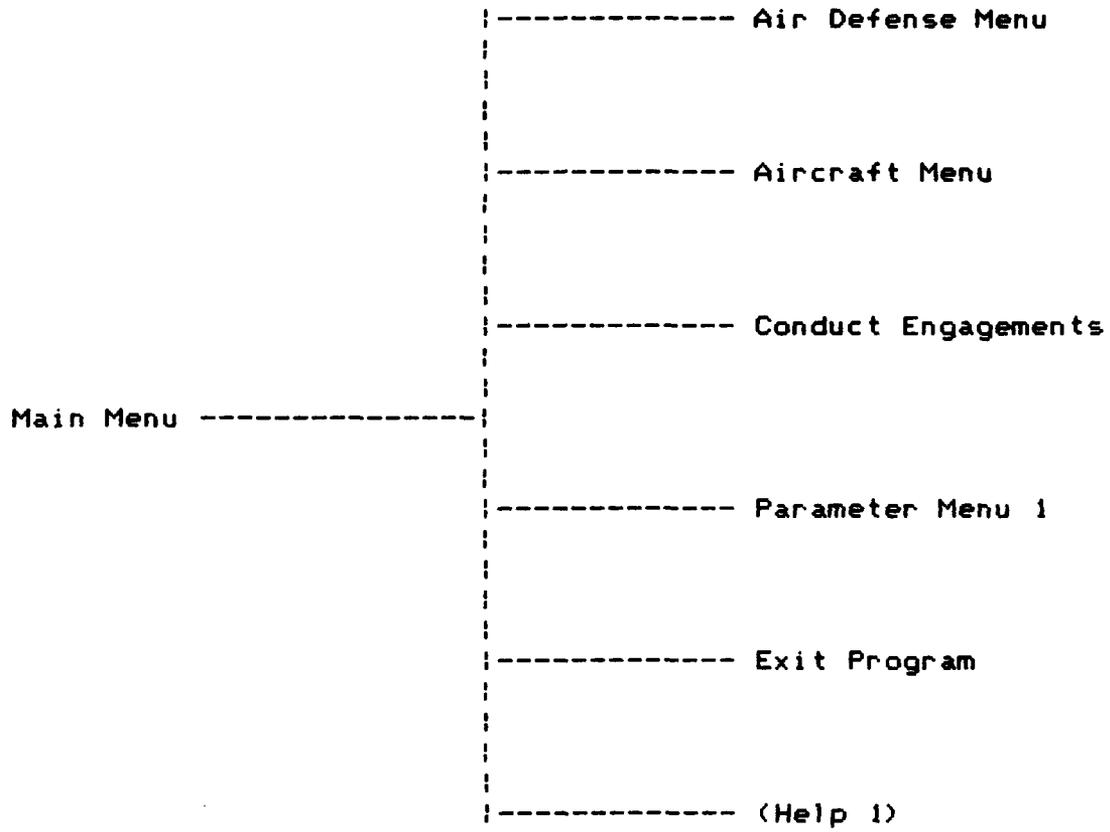


Chart C-2: Air Defense Menu Selections

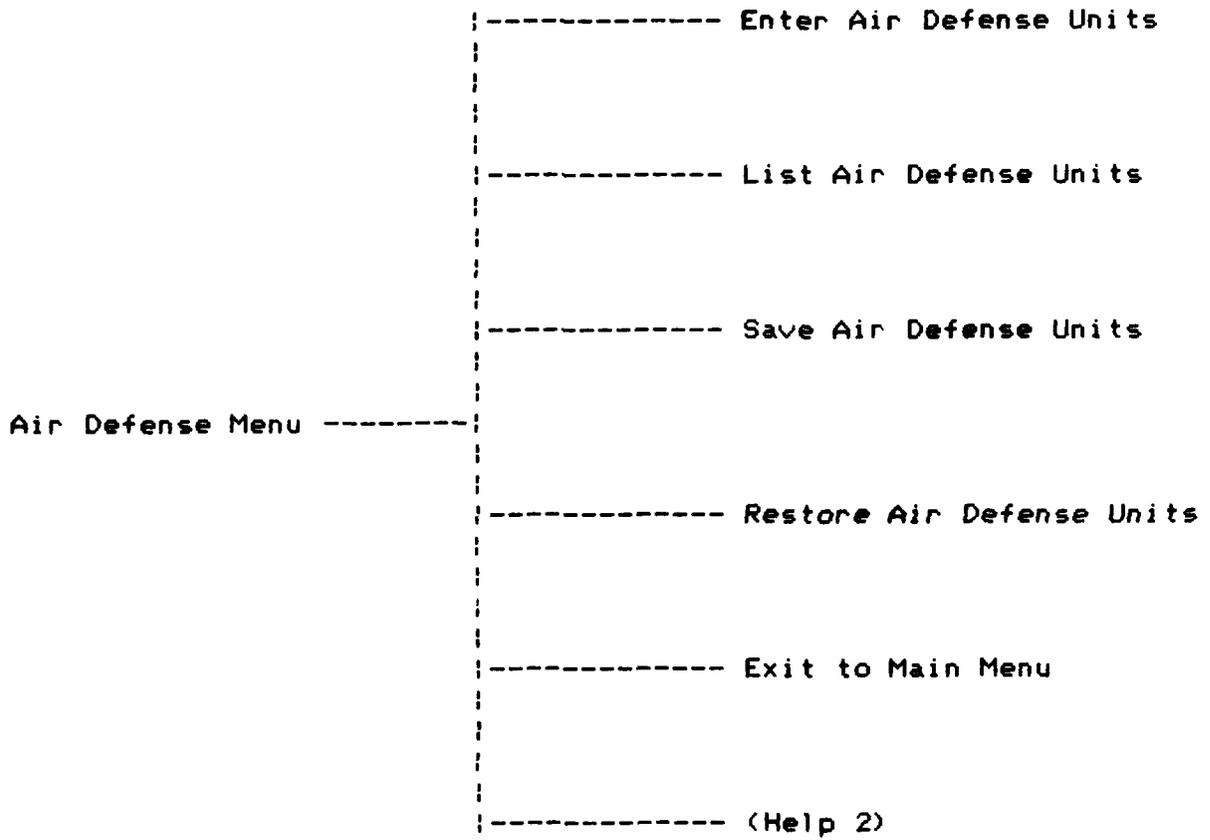


Chart C-3: Aircraft Menu Selections

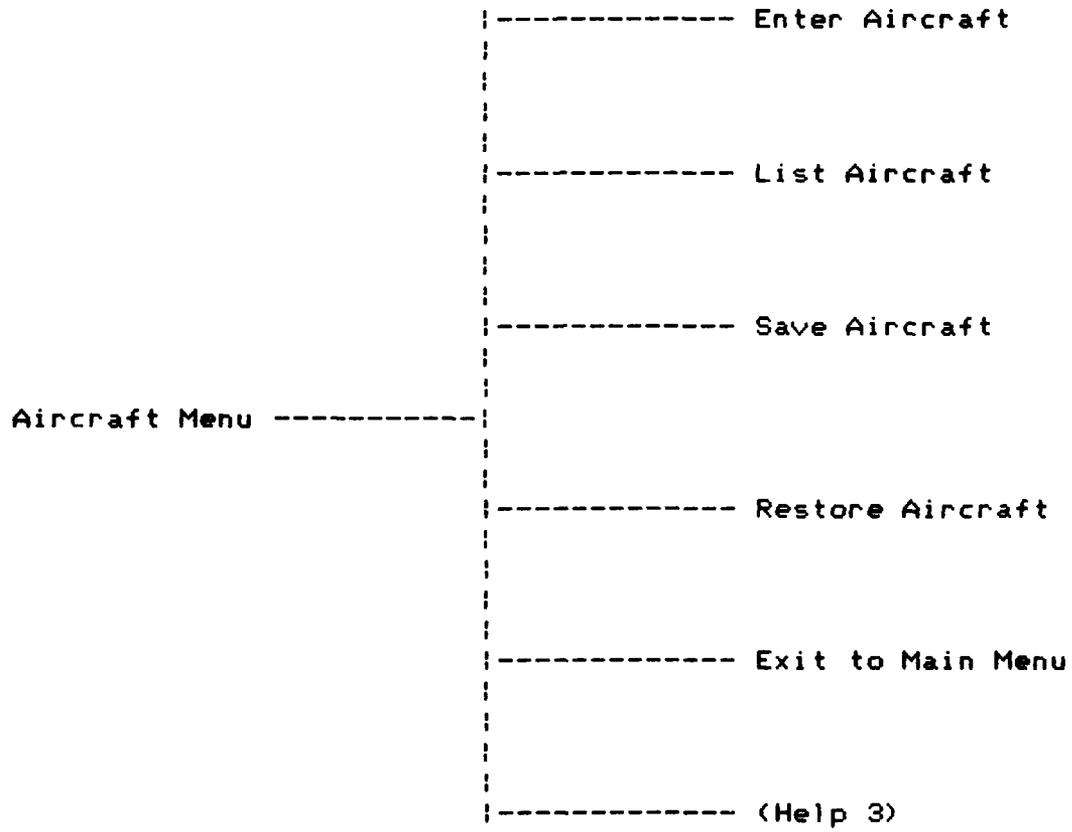


Chart C-4: Parameter Menu 1 Selections

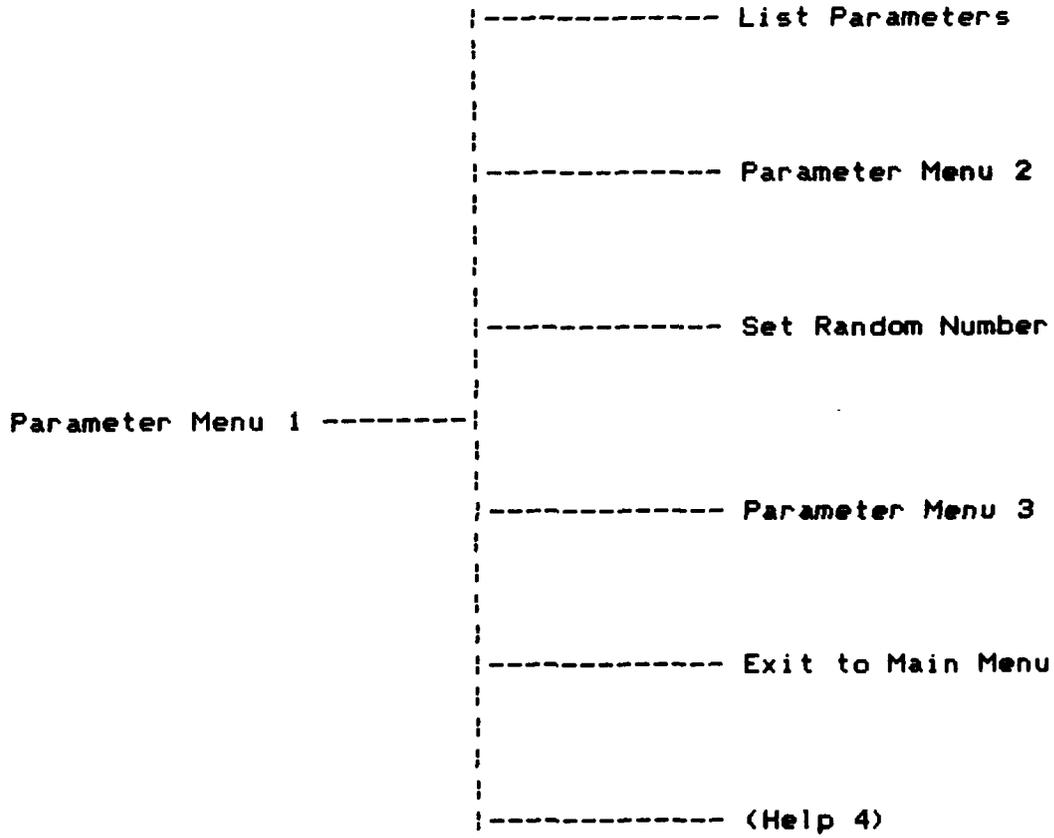


Chart C-5: Parameter Menu 2 Selections

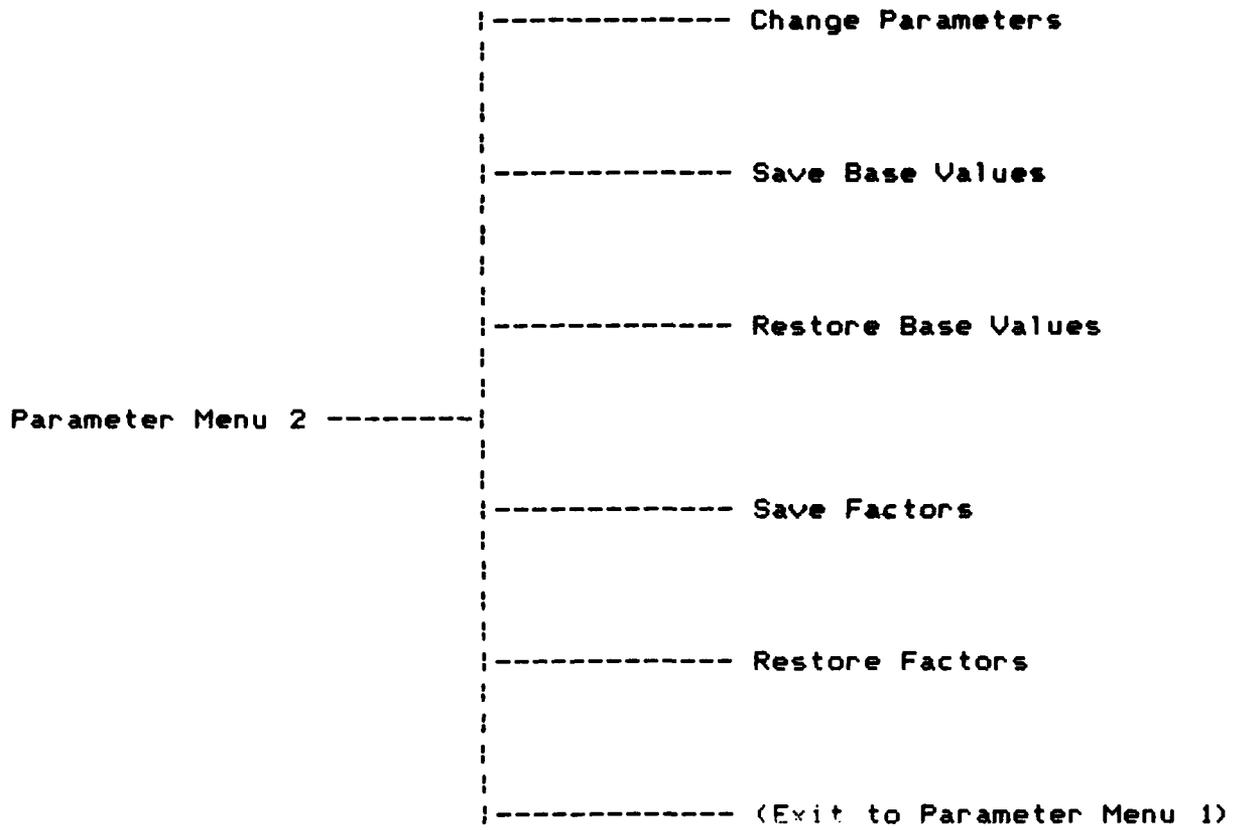
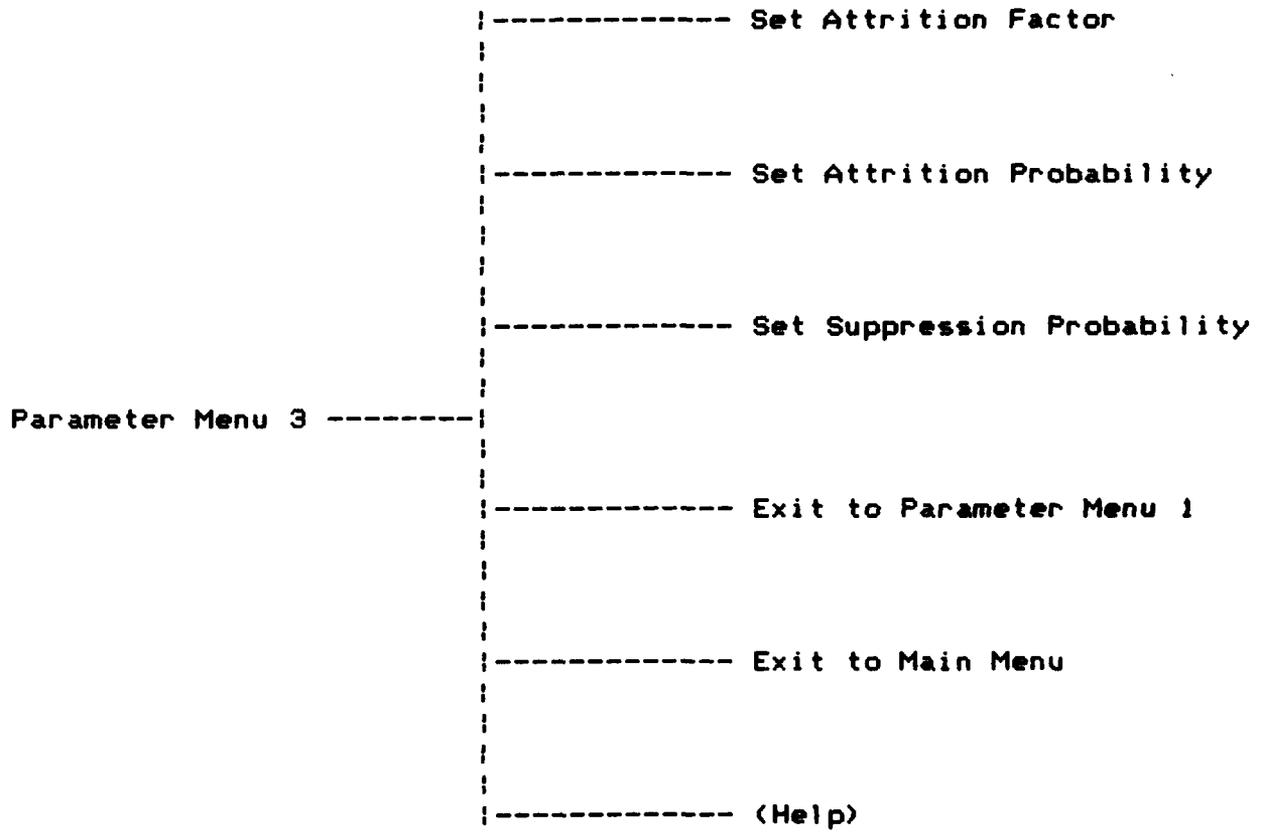


Chart C-6: Parameter Menu 3 Selections



APPENDIX D: Source Code

	Page
Constants	D-4
Types	D-4
Variables	D-5
Functions	
Uniform	D-9
ReadInt	D-9
Okay	D-10
Support Procedures	
ClearScreen	D-5
ShowProgramName	D-6
WaitforReturn	D-6
GetChoice	D-7
GetXChoice	D-8
WriteTitle	D-10
ListBaseFacts	D-11
ListOtherFacts	D-12
ShowLists	D-12
Main Program	D-64
Main Program Procedures	
Initialize	D-48
(Help Choices)	
'1' : HelpOne	D-52
'2' : HelpTwo	D-56
'3' : HelpThree	D-59
'4' : HelpFour	D-62

(cont)

	Page
Main Program Procedures	
'A' : WriteMainMenu	D-13
(Main Menu Choices)	
'B' : WriteDefenseMenu	D-14
'C' : WriteAircraftMenu	D-15
'D' : ConductEngagements	D-16
'E' : ChangeParameters	D-22
(Air Defense Menu Choices)	
'F' : EnterADunits	D-23
'G' : ListADunits	D-26
'H' : AdSaveData	D-27
'I' : AdGetData	D-28
(Aircraft Menu Choices)	
'J' : EnterAircraft	D-29
'K' : ListAircraft	D-32
'L' : AcSaveData	D-33
'M' : AcGetData	D-34
(Parameter Menu 1 Choices)	
'N' : ListFactors	D-35
'O' : WriteParaMenu2	D-36
'P' : SetSeed	D-37
'Q' : SetAttritSupMenu	D-38
(Parameter Menu 2 Choices)	
'R' : SetFactors	D-39
'S' : BaseSaveData	D-41
'T' : BaseGetData	D-42
'U' : FactSaveData	D-43
'V' : FactGetData	D-44
(Parameter Menu 3 Choices)	
'W' : GetAttritionFact	D-45
'X' : SetAttritionProb	D-46
'Y' : SetSuppressionProb	D-47

program ADMGSP(input,output(,Adsave,Accsave,Basave,Fasave)) ;

(X

AIR DEFENSE WAR GAME SUPPORT PROGRAM

AUTHOR : CPT David L. Bolté, U.S. Army

LANGUAGE : Standard Pascal

PURPOSE : To provide a training tool and war game support by generating probabilities of engagement and hit for a variety of Air Defense Units with different characteristics against a variety of Aircraft with different characteristics.

DESCRIPTION :

The program presents the user with a series of menus which allow description of arrays of Air Defense Units and Aircraft. These arrays, and a set of factors concerning various attributes of the Air Defense Unit (type, location and effectiveness) and Aircraft (type, tactic and aspect) are used to calculate probabilities of engagement and hit for an interaction between the systems. A random number generator is used to determine results of an engagement. All of the pertinent factors used in calculating the results of the engagement may be set within the program, and a programmer may add additional characteristics to the Air Defense Unit or the Aircraft and incorporate those characteristics in the calculation of engagement results.

X)

```

const
  STREAMSEED = 12347 ;           (Initial random number seed)

  TRITFACTOR = 0.0 ;           (Initial attrition factor)
  ATTRITPROB = 0.5 ;           (Initial probability of attrition)
  SUPRESPROB = 0.5 ;           (Initial probability of suppression)

  MAXADUNITS = 20 ;            (Max number of Air Defense Units)
  MAXAIRCRAF = 20 ;            (Max number of Aircraft)
                                (Factors for probability calculations)
  MAXADKINDS = 5 ;             (Max number of Air Defense Unit types)
  MAXADPLACE = 3 ;             (Max number of Air Defense Unit locations)
  MAXACSORTS = 3 ;             (Max number of Aircraft types)
  MAXACTACTS = 3 ;             (Max number of Aircraft tactics)
  MAXACASPEC = 3 ;             (Max number of Aircraft aspects)
  MAXFACTORS = 5 ;             (Equal to the largest of the above factors)

```

```

type
  ADRECORD = record             (Air Defense Unit Description)
    Isactive      : boolean ;
    Kind, Location : integer ;
    Effectlevel   : real ;
  end ;

  ACRECORD = record             (Aircraft Description)
    Isactive      : boolean ;
    Sort, Tactic, Aspect : integer ;
  end ;

  STR11 = packed array[1..11] of char ;
  STR35 = packed array[1..35] of char ;

  DESCRIPS = array[0..MAXFACTORS] of STR11 ;
  ADTYPE = array[1..MAXADUNITS] of ADRECORD ;
  ACTYPE = array[1..MAXAIRCRAF] of ACRECORD ;
  BASEFACTS = array[1..MAXADKINDS,1..MAXACSORTS] of real ;
  FACTORS = array[1..MAXADKINDS,1..MAXFACTORS] of real ;

```

```

var
    (Global Variables)
    Seed      : integer ;      (Random number seed)
    Choice    : char ;        (Program control variable)
    Atritfactor : real ;      (Attrition factor)
    Atritprob  : real ;      (Probability of attrition)
    Suprsprob  : real ;      (Probability of suppression)

    (Variable Data Arrays)
    Adunit    : ADTYPE ;
    Aircraft  : ACTYPE ;

    (Descriptive Arrays)
    Adkinds   : DESCRIPS ;
    Adplaces  : DESCRIPS ;
    Ackinds   : DESCRIPS ;
    Actactics : DESCRIPS ;
    Acaspects : DESCRIPS ;

    (Factor Arrays)
    PEbase    : BASEFACTS ;   (array of engagement probabilities)
    PHbase    : BASEFACTS ;   (array of hit probabilities)
    Placefact : FACTORS ;     (array of location factors)
    Tactifact : FACTORS ;     (array of tactics factors)
    Aspectfact : FACTORS ;    (array of aspect factors)

(*)
    (Data Files for file I/O implementation)
    Adsave    : text ;
    Acsave    : text ;
    Basave    : text ;
    Fasave    : text ;
*)

procedure ClearScreen ;
(
    Called from : Numerous procedures

    Returns to  : Calling procedure

    Purpose     : Clears CRT screen of previous information

    Description : Scrolls previous information off screen. Should
                  be replaced with CRT specific screen clearing
                  sequence for slow CRTs
)
    var Counter : integer ;
begin
    for Counter := 1 to 25 do writeln ;
end ; (procedure ClearScreen)

```

```

procedure ShowProgramName ;
(
  Called from : Main Program

  Returns to   : Main Program

  Purpose      : Clears CRT screen and displays program name

  Description  : Scrolls previous information off screen and
                writes program name.
)
  var Counter : integer ;
begin
  ClearScreen ;
  writeln('      A I R  D E F E N S E') ;
  writeln('      W A R  G A M E') ;
  writeln('  S U P P O R T  P R O G R A M ') ;
  for Counter := 1 to 10 do writeln ;
end ; (procedure ShowProgramName)

procedure WaitForReturn ;
(
  Called from : Numerous procedures

  Returns to   : Calling procedure

  Purpose      : Suspends program execution

  Description  : Displays message and waits for user to
                enter the CR/LF key.
)
begin
  writeln ;
  write('Press Return or Enter to Continue') ;
  readln ;
  writeln ;
end ; (procedure WaitForReturn)

```

```
procedure GetChoice(Alpha, Bravo, Charlie, Delta, Echo, Help : char ;  
                   var Choice : char) ;
```

```
(
```

```
  Called from : Numerous procedures
```

```
  Returns to   : Calling procedure
```

```
  Purpose      : Converts displayed menu choice to Main Program choice.
```

```
  Description  :
```

```
  A uniform set of choices is offered (A..E,H) in each menu and are  
  converted here to the set of choices used by the main program.  
  Uppercase is filtered out in this procedure. The procedure is passed  
  the set of choices that correspond to the available procedures and  
  returns the converted choice from user input.
```

```
)
```

```
begin
```

```
  repeat
```

```
    write('Choose A, B, C, D, E, or H for Help >') ;
```

```
    readln(Choice) ;
```

```
  until Choice in ['a','A','b','B','c','C','d','D','e','E','h','H'] ;
```

```
  case Choice of
```

```
    'a', 'A' : Choice := Alpha ;
```

```
    'b', 'B' : Choice := Bravo ;
```

```
    'c', 'C' : Choice := Charlie ;
```

```
    'd', 'D' : Choice := Delta ;
```

```
    'e', 'E' : Choice := Echo ;
```

```
    'h', 'H' : Choice := Help ;
```

```
  end ; (case)
```

```
end ; (procedure GetChoice)
```

```

procedure GetXChoice(Alpha, Bravo, Charlie, Delta, Echo, Exit : char ;
                    var Choice : char) ;
{
  Called from : ListFactors, WriteParaMenu, SetFactors

  Returns to   : Calling procedure

  Purpose      : Converts displayed menu choice to calling procedure choice.

  Description  :

  A uniform set of choices is offered (A..E,X) in each menu and are
  converted here to the set of choices used by the calling procedure.
  Uppercase is filtered out in this procedure. The procedure is passed
  the set of choices that correspond to the available procedures and
  returns the converted choice from user input.
}

begin
  repeat
    write('Choose A, B, C, D, E, or X to Exit > ') ;
    readln(Choice) ;
  until Choice in ['a','A','b','B','c','C','d','D','e','E','x','X'] ;
  case Choice of
    'a', 'A' : Choice := Alpha ;
    'b', 'B' : Choice := Bravo ;
    'c', 'C' : Choice := Charlie ;
    'd', 'D' : Choice := Delta ;
    'e', 'E' : Choice := Echo ;
    'x', 'X' : Choice := Exit ;
  end ; {case}
end ; {procedure GetXChoice}

```

```

function Uniform : integer ;
(
  Called from : ConductEngagement procedure

  Returns to   : Calling procedure

  Purpose      : Returns a random integer

  Description  :

  Generates sequence of 32,763 random integers in the range 1..100
  Requires the external global variable Seed
  MULT may be 1221,1287,3993,4189,4293,9237,14789,15125 or 17245
)

  const
    MULT = 3993 ;
begin
  Seed := MULT * Seed + 1 ;
  if Seed < 0 then Seed := Seed + maxint + 1 ;
  Uniform := trunc((Seed/maxint)*100) + 1 ;
end ; (function Uniform)

function ReadInt(var Int : integer ; Low, High : integer) : boolean ;
(
  Called from : Numerous procedures

  Returns to   : Calling procedure

  Purpose      : Attempts to read an integer

  Description  :

  Function ReadInt attempts to read an integer between Low and High,
  returning the integer in Int and true iff successful. Note that
  minimal error trapping is done here, and a non-numeric character
  will cause the program to abort.
)

  var Temp : real ;
begin
  readln(Temp) ;
  Int := abs(round(Temp)) ;
  ReadInt := (Int >= Low) and (Int <= High) ;
end ; (function ReadInt)

```

```

function Okay : boolean ;
(
  Called from : Numerous procedures

  Returns to   : Calling procedure

  Purpose      : Suspends execution until the prompting
                 question is answered with a y/Y or n/N.

  Description  : Solicits user for y/Y or n/N and returns true for y/Y.
)

var Response : char ;
begin
  repeat
    write('Enter y or Y for yes, n or N for no > ') ;
    readln(Response) ;
  until Response in ['y','Y','n','N'] ;
  Okay := Response in ['y','Y'] ;
end ; (function Okay)

procedure WriteTitle(Title : STR35) ;
(
  Called from : Numerous procedures

  Returns to   : Calling procedure

  Purpose      : Displays a single line title and 2 blank lines

  Description  : Scrolls previous information off screen and writes
                 title for screen.
)

begin
  ClearScreen ;
  writeln(Title) ;
  writeln ;
  writeln ;
end ; (procedure WriteTitle)

```

```

procedure ListBaseFacts(Title : STR35 ; Basetype : BASEFACTS) ;
(
  Called from : ListFactors and SetFactors procedures

  Returns to : Calling procedure

  Purpose      : Displays the table of factor information

  Description  : Scrolls previous information off screen and writes
                 title for screen. Two loops write the information
                 lines for the particular table requested.
)

var Row, Col : integer ;
begin
  WriteTitle(Title) ;
  write('      ') ;
  for Col := 1 to MAXACSORTS do write(Ackinds[Col]) ;
  for Row := 1 to MAXADKINDS do
    begin
      writeln ;
      write(Adkinds[Row], ' ') ;
      for Col := 1 to MAXACSORTS do
        write(Basetype[Row,Col]:4:2, ' ') ;
      end ;
      writeln ;
    end ;
end ; {procedure Lis+BaseFacts}

```

```

procedure ListOtherFacts(Title : STR35 ; Factor : FACTORS ;
                        Numatribs : integer ; Attribute : DESCRIPS) ;
(
  Called from : ListFactors and SetFactors procedures

  Returns to   : Calling procedure

  Purpose      : Displays the table of factor infomation

  Description  : Scrolls previous information off screen and writes
                title for screen. Two loops write the information
                lines for the particular table requested.
)

  var Row, Col : integer ;
begin
  WriteTitle(Title) ;
  write(' ') ;
  for Col := 1 to Numatribs do write(Attribute[Col]) ;
  for Row := 1 to MAXADKINDS do
    begin
      writeln ;
      write(Adkinds[Row], ' ') ;
      for Col := 1 to Numatribs do
        write(Factor[Row,Col]:4:2, ' ') ;
      end ;
      writeln ;
    end ;
end ; (procedure ListOtherFacts)

procedure ShowLists ;
(
  Called from : ListFactors and SetFactors procedures

  Returns to   : Calling procedure

  Purpose      : Displays the list of factors

  Description  : Scrolls previous information off screen and writes
                title for screen and list information.
)
begin
  WriteTitle(' P A R A M E T E R L I S T S ') ;
  writeln('A : ENGAGEMENT PROBABILITY BASE') ;
  writeln ;
  writeln('B : HIT PROBABILITY BASE') ;
  writeln ;
  writeln('C : LOCATION FACTOR') ;
  writeln ;
  writeln('D : TACTIC FACTOR') ;
  writeln ;
  writeln('E : ASPECT FACTOR') ;
  writeln ;
end ; (procedure ShowLists)

```

```

procedure WriteMainMenu(var Choice : char) ;
(
    Main Program Choice 'A'
)
(
    Called from : Main Program

    Returns to : Operating System

    Purpose : To present the choices for entering the Air Defense
              Unit data or Aircraft data, conducting an engagement,
              setting parameters or ending the program.

    Description : The user is given 5 choices in a menu presentation.
)
begin
    WriteTitle('          M A I N M E N U          ');
    writeln('A :  DISPLAY AIR DEFENSE MENU');
    writeln;
    writeln('B :  DISPLAY AIRCRAFT MENU');
    writeln;
    writeln('C :  CONDUCT ENGAGEMENTS');
    writeln;
    writeln('D :  DISPLAY PARAMETER MENU 1');
    writeln;
    writeln('E :  END PROGRAM');
    writeln;
    GetChoice('B','C','D','E','.', '1',Choice)
end ; {procedure WriteMainMenu}

```

```

procedure WriteDefenseMenu(var Choice : char);

(%)
      Main Program Choice 'B'
%)
(
  Called from : Main Menu

  Returns to  : Main Menu

  Purpose     : To present the choices for entering the Air Defense
                Unit data.

  Description : The user is given 5 choices in a menu presentation.
)

begin
  WriteTitle('  A I R  D E F E N S E  M E N U  ');
  writeln('A :  ENTER DESCRIPTION OF');
  writeln('  AIR DEFENSE UNITS');
  writeln;
  writeln('B :  LIST CURRENTLY ENTERED');
  writeln('  AIR DEFENSE UNITS');
  writeln;
  writeln('C :  SAVE DATA');
  writeln;
  writeln('D :  RESTORE DATA');
  writeln;
  writeln('E :  EXIT TO MAIN MENU');
  writeln;
  GetChoice('F','G','H','I','A','2',Choice);
end ; {procedure WriteDefenderMenu}

```

```

procedure WriteAircraftMenu(var Choice : char);
(*
                                Main Program Choice 'C'
*)
(
  Called from : Main Menu

  Returns to  : Main Menu

  Purpose     : To present the choices for entering the Aircraft data

  Description : The user is given 5 choices in a menu presentation.
)

begin
  WriteTitle('   A I R C R A F T   M E N U   ');
  writeln('A : ENTER DESCRIPTION OF AIRCRAFT');
  writeln;
  writeln('B : LIST CURRENTLY ENTERED AIRCRAFT');
  writeln;
  writeln('C : SAVE DATA');
  writeln;
  writeln('D : RESTORE DATA');
  writeln;
  writeln('E : EXIT TO MAIN MENU');
  writeln;
  GetChoice('J','K','L','M','A','3',Choice);
end; {procedure WriteAircraftMenu}

```

```
procedure ConductEngagements(var Choice : char) ;
```

```
(X
```

```
        Main Program Choice 'D'
```

```
X)
```

```
(
```

```
    Called from : Main Menu
```

```
    Returns to  : Main Menu
```

```
    Purpose     : To produce the probabilities of engagement and hit  
                  for the entered Air Defense Unit and Aircraft number.
```

```
    Description :
```

```
    The user is solicited for an Air Defense Unit number and an Aircraft  
    number, given the option of ending the engagement routine by entering  
    a 0 for either. If active Unit and Aircraft numbers are entered, the  
    statistics for each is displayed and the engagement conducted.
```

```
    The probability of engagement is calculated (Threshold) from the  
    engagement base (PEbase), location, tactic and aspect factors, and  
    effectiveness level. If the random number generator returns a number  
    less than this Threshold then the calculations for determining whether  
    the Aircraft was hit are begun. A new Threshold for probability of hit  
    is calculated with PHbase and the same factors and determination is made  
    with the random number generator if the Aircraft was hit.
```

```
    If the Aircraft was hit, tests are made with random numbers against the  
    current probabilities of suppression (Suprsprob) and attrition (Atritprob)  
    and, if necessary, the effectiveness level of the Air Defense Unit is  
    decremented by the current attrition factor (Atritfactor).
```

```
    The user is then asked if another engagement is to be conducted.
```

```
)
```

```
var
```

```
    Adindex,Acindex,Counter : integer ;
```

```
    Sawaircraft,Done,None   : boolean ;
```

```

procedure GetADIndex(var Adindex : integer ; var Done : boolean) ;
  var Activeindex : boolean ;
      Counter      : integer ;
begin
  Activeindex := false ;
  while not Activeindex do
    begin
      repeat
        writeln('Enter Air Defense Unit number') ;
        writeln('between 1 and ',MAXADUNITS:2) ;
        write('or 0 to exit ) ') ;
      until ReadInt(Adindex,0,MAXADUNITS) ;
      Done := Adindex = 0 ;
      if Done then Activeindex := Done
        else Activeindex := Adunit[Adindex].Isactive ;
      if not Done and not Activeindex then
        begin
          writeln('That Unit has not been entered.') ;
          writeln ;
          writeln('Available Units are:') ;
          for Counter := 1 to 10 do
            if Adunit[Counter].Isactive then write(Counter:3) ;
          writeln ;
          for Counter := 11 to MAXADUNITS do
            if Adunit[Counter].Isactive then write(Counter:3) ;
          writeln ;
          writeln ;
        end ; (if)
      end ; (while)
      writeln ;
    end ; (procedure GetADIndex)

```

```

procedure GetAcIndex(var Acindex : integer ; var Done : boolean) ;
  var Activeindex : boolean ;
      Counter      : integer ;

begin
  Activeindex := false ;
  while not Activeindex do
    begin
      repeat
        writeln('Enter Aircraft number') ;
        writeln('between 1 and ',MAXAIRCRAF:2) ;
        write('or 0 to exit ) ') ;
      until ReadInt(Acindex,0,MAXAIRCRAF) ;
      Done := Acindex = 0 ;
      if Done then Activeindex := Done
        else Activeindex := Aircraft[Acindex].Isactive ;
      if not Done and not Activeindex then
        begin
          writeln('That Aircraft has not been entered.') ;
          writeln ;
          writeln('Available Aircraft are:') ;
          for Counter := 1 to 10 do
            if Aircraft[Counter].Isactive then write(Counter:3) ;
          writeln ;
          for Counter := 11 to MAXADUNITS do
            if Aircraft[Counter].Isactive then write(Counter:3) ;
          writeln ;
          writeln ;
        end ; (if)
      end ; (while)
    end ; (procedure GetAcIndex)

procedure ShowSpecs(Adindex,Acindex : integer) ;
begin
  with Adunit[Adindex] do with Aircraft[Acindex] do begin
    WriteTitle('E N G A G E M E N T   R E S U L T S ') ;
    writeln('Air Defense Unit: ',Adindex:2) ;
    writeln('  AD System Type: ',Adkinds[Kind]) ;
    writeln('      Location: ',Adplaces[Location]) ;
    writeln('  Effectiveness: ',Effectlevel:4:2) ;
    writeln('      Aircraft: ',Acindex:2) ;
    writeln('  Aircraft Type: ',Ackinds[Sort]) ;
    writeln('      Tactic: ',Actactics[Tactic]) ;
    writeln('      Aspect: ',Acaspects[Aspect]) ;
    writeln ;
  end ; (with)
end ; (procedure ShowSpecs)

```

```

procedure GetEngaged(Adindex,Acindex : integer ; var Sawaircraft : boolean) ;
  var
    Threshold : real ;
    Randnum   : integer ;
begin
  with Adunit[Adindex] do with Aircraft[Acindex] do begin
    Threshold := PBase[Kind,Sort] * Placefact[Kind,Location]
               * Tactifact[Kind,Tactic] * Aspectfact[Kind,Aspect]
               * Effectlevel * 100 ;
    Randnum   := Uniform ;
    if Randnum > Threshold then
      Sawaircraft := false else Sawaircraft := true ;
    if Sawaircraft then
      writeln('>> Aircraft was engaged.')
    else writeln('>> Aircraft was not engaged.') ;
    writeln('Random number: ',Randnum:2,' Threshold: ',Threshold:4:1) ;
    writeln ;
  end ; (with)
end ; (procedure GetEngaged)

```

```

procedure CalcResults(Adindex, Acindex : integer) ;
var
  Threshold      : real ;
  Randnum        : integer ;
  Hitaircraft    : boolean ;
begin
  with Adunit[Adindex] do with Aircraft[Acindex] do begin
    Threshold := PHbase[Kind,Sort] * Placefact[Kind,Location]
                * Tactifact[Kind,Tactic] * Aspectfact[Kind,Aspect]
                * Effectlevel * 100 ;
    Randnum    := Uniform ;
    if Randnum > Threshold then
      Hitaircraft := false else Hitaircraft := true ;
    if Hitaircraft then begin
      writeln('>> Aircraft was hit.') ;
      write('Random number: ',Randnum:2) ;
      writeln(' Threshold: ',Threshold:4:1) ;
      writeln ;
      if Uniform > Suprsprob * 100 then
        begin
          writeln('>> Aircraft was suppressed.') ;
          writeln('>> Aircraft mission was aborted.') ;
          end (if Uniform)
        else writeln('>> Aircraft was destroyed.') ;
      end (if Hitaircraft)
    else begin
      writeln('>> Aircraft was not hit.') ;
      write('Random number: ',Randnum:2) ;
      writeln(' Threshold: ',Threshold:4:1) ;
      writeln ;
      if Uniform > Atritprob * 100 then
        begin
          writeln('>> Air Defense Unit was suppressed.') ;
          writeln('>> Effectiveness reduced by ',Atritfactor:3:2, '.') ;
          Effectlevel := Effectlevel - Atritfactor ;
          end (if Uniform)
        else writeln('>> Air Defense Unit was not suppressed.') ;
      end ; (else)
    end ; (with)
  end ; (procedure CalcResults)

```

```

begin {procedure ConductEngagements}
  Done := false ;
  while not Done do begin
    None := true ;
    WriteTitle('      E N G A G E M E N T S      ');
    for Counter := 1 to MAXADUNITS do {a check for entered Units}
      if Adunit[Counter].Isactive then None := false ;
    if None then
      begin
        Done := true ;
        writeln('No Air Defense Units have been entered.') ;
        writeln('Please enter Air Defense Units from the') ;
        writeln('Air Defense Menu.') ;
        WaitforReturn ;
      end
    else
      begin
        None := true ;
        for Counter := 1 to MAXAIRCRAF do {a check for entered Aircraft}
          if Aircraft[Counter].Isactive then None := false ;
        if None then
          begin
            Done := true ;
            writeln('No Aircraft have been entered.') ;
            writeln('Please enter Aircraft from the') ;
            writeln('Aircraft Menu.') ;
            WaitforReturn ;
          end ;
        end ; {else}
      if not Done then GetADIndex(Adindex,Done) ;
      if not Done then GetAcIndex(Acindex,Done) ;
      if not Done then
        begin
          ShowSpecs(Adindex,Acindex) ;
          GetEngaged(Adindex,Acindex,Sawaircraft) ;
          if Sawaircraft then CalcResults(Adindex,Acindex) ;
          writeln ;
          writeln('Conduct another engagement?') ;
          Done := not Okay
        end ; {if not Done}
      end ; {while not Done}
      Choice := 'A' ; {return to Main Menu}
    end ; {procedure ConductEngagements}

```

```

procedure ChangeParameters(var Choice : char);
(
    Main Program Choice 'E'
)
(
    Called from : Main Menu

    Returns to : Main Menu

    Purpose      : To present the choices for changing the program
                  parameters of random number seed, probabilities
                  of engagement and hit, and threshold, attrition
                  and suppression factors.

    Description  : The user is given 5 choices in a menu presentation.
)

begin
    WriteTitle(' P A R A M E T E R M E N U 1 ');
    writeln('A : LIST CURRENT PARAMETERS');
    writeln;
    writeln('B : SET/SAVE PARAMETERS');
    writeln;
    writeln('C : SET RANDOM NUMBER SEED');
    writeln;
    writeln('D : SET ATTRITION/SUPPRESSION FACTORS');
    writeln;
    writeln('E : EXIT TO MAIN MENU');
    writeln;
    GetChoice('N','O','P','Q','A','4',Choice);
end; {procedure ChangeParameters}

```

```
procedure EnterADunits(var Choice : char) ;
```

```
(*)
```

```
                Main Program Choice 'F'
```

```
*)
```

```
(
```

```
    Called from : Air Defense Menu
```

```
    Returns to  : Air Defense Menu
```

```
    Purpose     : To enter information about the Air Defense Units  
                 in the array Adunits.
```

```
    Description :
```

```
    The array of Air Defense Units is tested for the first available  
    record with the Isactive field set to false. The user is told  
    either that the array is full or what the next available record is  
    and is solicited for the Unit number to enter or 0 to terminate.
```

```
    The user then is presented with the choices for Air Defense Unit  
    type, location and effectiveness level. Once the Unit record is  
    completed the user is asked if another Unit is to be described.
```

```
)
```

```
var
```

```
    Index : integer ;
```

```
    Done  : boolean ;
```

```
procedure GetIndex ;
```

```
begin
```

```
    WriteTitle(' Air Defense Unit Descriptions  ');
```

```
    Index := 0 ;
```

```
    repeat
```

```
        Index := Index + 1 ;
```

```
    until ((not Adunit[Index].Isactive) or (Index = MAXADUNITS)) ;
```

```
    if ((Index = MAXADUNITS) and (Adunit[Index].Isactive)) then
```

```
        writeln('Air Defense Unit array is full')
```

```
    else writeln('Next available Unit number: ',Index:2) ;
```

```
    repeat
```

```
        writeln ;
```

```
        writeln('Enter Air Defense Unit number') ;
```

```
        writeln('between 1 and ',MAXADUNITS:2) ;
```

```
        write ('or 0 to exit > ');
```

```
    until ReadInt(Index,0,MAXADUNITS) ;
```

```
end ; (procedure GetIndex)
```

```

procedure GetAdKind ;
  var Counter : integer ;
begin
  WriteTitle( ' Air Defense Unit Descriptions ' ) ;
  writeln( ' Specify Air Defense Unit Type' ) ;
  writeln ;
  for Counter := 1 to MAXADKINDS do
    begin
      writeln( ' ', Counter:2, ' : ', Adkinds[Counter] ) ;
      writeln ;
    end ;
  repeat
    write( 'Enter Air Defense Unit type > ' )
  until ReadInt( Adunit[Index].Kind, 1, MAXADKINDS ) ;
end ; {procedure GetAdKind}

procedure GetAdLocation ;
  var Counter : integer ;
begin
  WriteTitle( ' Air Defense Unit Descriptions ' ) ;
  writeln( ' Specify Air Defense Unit Location' ) ;
  writeln ;
  for Counter := 1 to MAXADPLACE do
    begin
      writeln( ' ', Counter:2, ' : ', Adplaces[Counter] ) ;
      writeln ;
    end ;
  repeat
    write( 'Enter Air Defense Unit location > ' )
  until ReadInt( Adunit[Index].Location, 1, MAXADPLACE ) ;
end ; {procedure GetAdLocation}

```

```

procedure GetEffectLevel ;
begin
  WriteTitle('  Air Defense Unit Descriptions  ');
  writeln('Specify Air Defense Unit Effectiveness');
  with Adunit[Index] do
    repeat
      writeln ;
      writeln('Enter the decimal effectiveness level');
      write ('of the Air Defense Unit (0.00 - 1.00) ');
      readln(Effectlevel) ;
    until (Effectlevel >= 0.0) and (Effectlevel <= 1.0) ;
  end ; (procedure GetEffectLevel)

begin (procedure EnterADunits)
  Done := false ;
  repeat
    GetIndex ;
    if Index = 0 then Done := true
    else
      begin
        Adunit[Index].Isactive := true ;
        GetAdKind ;
        GetAdLocation ;
        GetEffectLevel ;
      end ;
  until Done ;
  Choice := 'B' ; (return to the Air Defense Menu)
end ; (procedure EnterADunits)

```

```

procedure ListADunits(var Choice : char) ;

(*
                                Main Program Choice 'G'
*)
(
  Called from : Air Defense Menu

  Returns to   : Air Defense Menu

  Purpose      : To display the currently entered Air Defense Units.

  Description  : The Air Defense Unit array Adunit is examined for any
                 Isactive entries which are displayed 10 at a time. If
                 no Air Defense Units have been entered an appropriate
                 message is displayed.
)

var
  Index,Counter : integer ;
  None          : boolean ;
begin
  WriteTitle('  Air Defense Unit Descriptions  ');
  Counter := 0 ;                {for number of lines displayed}
  None := true ;
  writeln('Number Type      Location      Effectiveness');
  writeln ;
  for Index := 1 to MAXADUNITS do
    with Adunit[Index] do
      if Isactive then
        begin
          Counter := Counter + 1 ;
          None := false ;
          if Counter = 11 then      {Display only 10 at a time}
            begin
              writeln ;
              writeln('          - More -');
              WaitForReturn ;
              Counter := 0 ;
            end ; {if}
          writeln(Index:3,'      ',Adkinds[Kind],
                 Adplaces[Location],Effectlevel:7:2) ;
        end ;
      if None then writeln('No Air Defense Units have been entered') ;
      WaitForReturn ;
      Choice := 'B' ; {return to Air Defense Menu}
    end ; {procedure ListADunits}
end ;

```

```

procedure AdSaveData(var Choice : char) ;

(×
                                Main Program Choice 'H'
×)
(
  Called from : Air Defense Menu

  Returns to  : Air Defense Menu

  Purpose     : To save the current Air Defense Unit array Adunit
                in the file Adsave.

  Description : This procedure is highly operating system
                and compiler dependant.
)

begin
  ClearScreen ;
  writeln('Saving the array of currently entered') ;
  writeln('Air Defense Units will overwrite the') ;
  writeln('information currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
(××)
      (NO implementation)
      writeln ;
      writeln('This procedure to be implemented by user.') ;
(××)
      WaitForReturn ;
    end ;
  Choice := 'B' ; (return to the Air Defense Unit menu)
end ; (procedure AdSaveData)

```

```

procedure AdGetData(var Choice : char) ;

(%)
      Main Program Choice 'I'
%)
(
  Called from : Air Defense Menu

  Returns to  : Air Defense Menu

  Purpose     : To restore the current Air Defense Unit array Adunit
               from the file Adsave.

  Description : This procedure is highly operating system
               and compiler dependant.
)

begin
  ClearScreen ;
  writeln('Restoring Air Defense Unit Data') ;
  writeln('will overwrite the information') ;
  writeln('currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (XX)
        (NO implementation)
        writeln ;
        writeln('This procedure to be implemented by user.') ;
      (XX)
        WaitForReturn ;
    end ;
  Choice := 'B' ; (return to Air Defense Unit menu)
end ; (procedure AdGetData)

```

```
procedure EnterAircraft(var Choice : char) ;
```

```
(*
```

```
          Main Program Choice 'J'
```

```
*)
```

```
(
```

```
  Called from : Aircraft Menu
```

```
  Returns to  : Aircraft Menu
```

```
  Purpose     : To enter information about the Aircraft in the  
                array Aircraft.
```

```
  Description :
```

```
  The array of Aircraft is tested for the first available record  
  with the Isactive field set to false. The user is told either  
  that the array is full or what the next available record is and  
  is solicited for the Aircraft number to enter or 0 to terminate.
```

```
  The user then is presented with the choices for Aircraft sort,  
  tactic, and aspect. Once the Aircraft record is completed the  
  user is asked if another Aircraft is to be described.
```

```
)
```

```
var
```

```
  Index : integer ;
```

```
  Done  : boolean ;
```

```
procedure GetIndex ;
```

```
begin
```

```
  WriteTitle('      Aircraft Descriptions      ');
```

```
  Index := 0 ;
```

```
  repeat
```

```
    Index := Index + 1 ;
```

```
  until ((not Aircraft[Index].Isactive) or (Index = MAXAIRCRAF)) ;
```

```
  if ((Index = MAXAIRCRAF) and (Aircraft[Index].Isactive)) then
```

```
    writeln('Aircraft array is full')
```

```
  else writeln('Next available aircraft number is ',Index:2) ;
```

```
  repeat
```

```
    writeln ;
```

```
    writeln('Enter Aircraft number') ;
```

```
    writeln('between 1 and ',MAXAIRCRAF:2) ;
```

```
    write ('or 0 to exit > ');
```

```
  until ReadInt(Index,0,MAXAIRCRAF) ;
```

```
end ; (procedure GetIndex)
```

```

procedure GetAcSort ;
  var Counter : integer ;
begin
  WriteTitle('      Aircraft Descriptions      ') ;
  writeln(' Specify Aircraft Type') ;
  writeln ;
  for Counter := 1 to MAXACSORTS do
    begin
      writeln('      ',Counter:2,' :      ',Ackinds[Counter]) ;
      writeln ;
    end ;
  repeat
    write('Enter Aircraft type > ')
  until ReadInt(Aircraft[Index].Sort,1,MAXACSORTS) ;
end ; {procedure GetAcSort}

```

```

procedure GetAcTactic ;
  var Counter : integer ;
begin
  WriteTitle('      Aircraft Descriptions      ') ;
  writeln(' Specify Aircraft Tactic') ;
  writeln ;
  for Counter := 1 to MAXACTACTS do
    begin
      writeln('      ',Counter:2,' :      ',Actactics[Counter]) ;
      writeln ;
    end ;
  repeat
    write('Enter Aircraft tactic > ')
  until ReadInt(Aircraft[Index].Tactic,1,MAXACTACTS) ;
end ; {procedure GetAcTactic}

```

```

procedure GetAcAspect ;
  var Counter : integer ;
begin
  WriteTitle('      Aircraft Descriptions      ');
  writeln(' Specify Aircraft Aspect');
  writeln ;
  for Counter := 1 to MAXACASPEC do
    begin
      writeln('      ',Counter:2,' :      ',Acaspects[Counter]) ;
      writeln ;
    end ;
  repeat
    write('Enter Aircraft aspect > ')
  until ReadInt(Aircraft[Index].Aspect,1,MAXACASPEC) ;
end ; {procedure GetAcAspect}

begin {procedure EnterAircraft}
  Done := false ;
  repeat
    GetIndex ;
    if Index = 0 then Done := true
    else
      begin
        Aircraft[Index].Isactive := true ;
        GetAcSort ;
        GetAcTactic ;
        GetAcAspect ;
      end ;
  until Done ;
  Choice := 'C' ; {return to Aircraft Menu}
end ; {procedure EnterAircraft}

```

```

procedure ListAircraft(var Choice : char) ;

(%)
                                Main Program Choice 'K'
%)
(
  Called from : Aircraft Menu

  Returns to  : Aircraft Menu

  Purpose     : To display the currently entered Aircraft.

  Description : The Aircraft array Aircraft is examined for any
                 Isactive entries which are displayed 10 at a time.
                 If no Aircraft have been entered an appropriate
                 message is displayed.
)

var
  Index,
  Counter : integer ;
  None    : boolean ;
begin
  Counter := 0 ;                                (for number of lines displayed)
  None    := true ;
  WriteTitle('    Aircraft Descriptions    ') ;
  writeln('Number  Type    Tactic    Aspect') ;
  writeln ;
  for Index := 1 to MAXAIRCRAF do
    with Aircraft[Index] do
      if Isactive then
        begin
          Counter := Counter + 1 ;
          None := false ;
          if Counter = 11 then                    (Display only 10 at a time)
            begin
              writeln ;
              writeln('          - More -') ;
              WaitForReturn ;
              Counter := 0 ;
            end ; (if)
          writeln(Index:3,'    ',Ackinds[Sort],' ',
                  Actactics[Tactic],Aaspects[Aspect]) ;
        end ; (if)
      if None then writeln('No Aircraft have been entered') ;
      WaitForReturn ;
      Choice := 'C' ; (return to Aircraft Menu)
    end ; (procedure ListAircraft)
end ;

```

```

procedure AcSaveData(var Choice : char) ;

(%)
      Main Program Choice 'L'
(%)
(
  Called from : Aircraft Menu

  Returns to  : Aircraft Menu

  Purpose     : To save the current Aircraft array Aircraft
               in the file Acsave.

  Description : This procedure is highly operating system
               and compiler dependant.
)

begin
  ClearScreen ;
  writeln('Saving the array of currently entered') ;
  writeln('Aircraft will overwrite the information') ;
  writeln('currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
(%%)
      (NO implementation)
      writeln ;
      writeln('This procedure to be implemented by user.') ;
(%%)
      WaitForReturn ;
    end ;
  Choice := 'C' ; (return to the Aircraft menu)
end ; (procedure AcSaveData)

```

```

procedure AcGetData(var Choice : char) ;
(
    Main Program Choice 'M'
)
(
    Called from : Aircraft Menu

    Returns to : Aircraft Menu

    Purpose      : To restore the current Aircraft array Aircraft
                  from the file Acsave.

    Description  : This procedure is highly operating system
                  and compiler dependant.
)
begin
    ClearScreen ;
    writeln('Restoring the Aircraft data') ;
    writeln('will overwrite the information') ;
    writeln('currently in use.') ;
    writeln ;
    writeln ;
    writeln('Do you wish to continue?') ;
    if OKay then
        begin
            (XX)
                (NO implementation)
                writeln ;
                writeln('This procedure to be implemented by user.') ;
            (XX)
                WaitForReturn ;
            end ;
        Choice := 'C' ; (return to Aircraft menu)
    end ; (procedure AcGetData)

```

```

procedure ListFactors(var Choice : char) ;

(%)
                                Main Program Choice 'N'
%)
(
  Called from : Parameter Menu

  Returns to  : Parameter Menu

  Purpose     : To present the choices for displaying the current
                probabilities of engagement and hit and factor
                data arrays.

  Description : The user is given 5 choices in a menu presentation.
)

var Done : boolean ;
begin
  Done := false ;
  repeat
    ShowLists ;
    writeln('Enter letter for parameters to display') ;
    GetXChoice('A','B','C','D','E','X',Choice) ;
    case Choice of
      'A' : ListBaseFacts (' Probability of Engagement Base ',PEbase) ;
      'B' : ListBaseFacts (' Probability of Hit Base ',PHbase) ;
      'C' : ListOtherFacts(' Location Factors ',Placefact,
                          MAXADPLACE, Adplaces) ;
      'D' : ListOtherFacts(' Tactics Factors ',Tactifact,
                          MAXACTACTS, Actactics) ;
      'E' : ListOtherFacts(' Aspect Factors ',Aspecfact,
                          MAXACASPEC, Acaspects) ;
      'X' : Done := true ;
    end ; (case)
    if not Done then WaitforReturn ;
  until Done ;
  Choice := 'E' (return to Parameter Menu)
end ; (procedure Listfactors)

```

```

procedure WriteParaMenu2(var Choice : char);
(
    Main Program Choice '0'
)
(
    Called from : Parameter Menu

    Returns to : Parameter Menu

    Purpose      : To present the choices for changing the parameters.

    Description  : The user is given 5 choices in a menu presentation.
                  May be exited without changing any parameters.
)

begin
    WriteTitle(' P A R A M E T E R M E N U 2 ');
    writeln('A : CHANGE PARAMETERS');
    writeln;
    writeln('B : SAVE BASE VALUES');
    writeln;
    writeln('C : RESTORE BASE VALUES');
    writeln;
    writeln('D : SAVE FACTORS');
    writeln;
    writeln('E : RESTORE FACTORS');
    writeln;
    GetXChoice('R','S','T','U','V','E',Choice);
end ; {procedure WriteParaMenu2}

```

```

procedure SetSeed(var Choice : char) ;

(*
                                Main Program Choice 'P'
*)
(
  Called from : Parameter Menu 2

  Returns to   : Parameter Menu 2

  Purpose      : To allow changing of the random number seed.

  Description  : The current random number seed is displayed,
                  the user is told the acceptable value range
                  and is solicited for a new value. The new
                  value is displayed.
)

var Counter : integer ;
begin
  WriteTitle('          SET RANDOM NUMBER SEED          ');
  writeln('Current random number seed: ',Seed) ;
  writeln('Initial random number seed: ',STREAMSEED) ;
  writeln ;
  writeln('The random number seed must be ');
  writeln('an integer between 0 and ',maxint) ;
  for Counter := 1 to 5 do writeln ;
  repeat
    write('Enter new random number seed > ');
  until ReadInt(Seed,0,maxint) ;
  writeln('New random number seed: ',Seed) ;
  writeln ;
  WaitForReturn ;
  Choice := 'E' ; (return to Parameter Menu)
end ; (procedure SetSeed)

```

```

procedure SetAtritSupMenu(var Choice : char) ;
(*
                                Main Program Choice 'Q'
*)
(
  Called from : Parameter Menu 1

  Returns to   : Parameter Menu 1 or Main Menu

  Purpose      : To present the choices for changing the attrition
                 probability and factor and suppression probability.

  Description  : The user is given 5 choices in a menu presentation.
)

begin
  WriteTitle('  P A R A M E T E R  M E N U  3  ');
  writeln('A : SET ATTRITION FACTOR') ;
  writeln ;
  writeln('B : SET ATTRITION PROBABILITY') ;
  writeln ;
  writeln('C : SET SUPPRESSION PROBABILITY') ;
  writeln ;
  writeln('D : EXIT TO PARAMETER MENU 1') ;
  writeln ;
  writeln('E : EXIT TO MAIN MENU') ;
  writeln ;
  GetChoice('W','X','Y','E','A','H',Choice) ;
  if Choice = 'H' then (dummy choice since no other help is available)
  begin
    writeln('This menu allows the attrition factor and') ;
    writeln('attrition and suppression probabilities') ;
    writeln('factors to be changed.') ;
    writeln('Continue with parameter changing?') ;
    if Okay then Choice := 'Q' (go on with this Parameter Menu)
    else Choice := 'E' ; (return to Parameter Menu 1)
  end ; (if)
end ; (procedure SetAtritSupMenu)

```

```

procedure SetFactors(var Choice : char) ;

(x
                                Main Program Choice 'R'
x)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To present the choices for and enabling the changing
                of the probabilities of engagement and hit and the
                factor data arrays.

  Description : The user is given 5 choices in a menu presentation.

  Once a parameter to change has been selected the factors are displayed
  and the user is solicited for the row and column of the factor to change,
  and the new value. The factors are redisplayed with the new value and
  the user is asked if another change is to be made, and then if another
  set of parameters is to be changed.
)
var (Global to SetFactors for Obtain, BaseChange and FactorChange)
    Row,Col : integer ;
    Value : real ;
    Done : boolean ;

procedure Obtain(var Row,Col : integer ; var Value : real) ;
var Maxrow, Maxcol : integer ;
begin
  Maxrow := Row ; Maxcol := Col ;
  writeln ;
  repeat
    write('Enter row of factor to change > ') ;
  until ReadInt(Row,1,Maxrow) ;
  repeat
    write('Enter column of factor to change > ') ;
  until ReadInt(Col,1,Maxcol) ;
  write('Enter new value > ') ;
  readln(Value) ;
  writeln ;
end ; (procedure Obtain)

```

```

procedure BaseChange(Title : STR35 ; var Basetype : BASEFACTS) ;
begin
  repeat
    ListBaseFacts(Title,Basetype) ;
    Row := MAXADKINDS ; Col := MAXACSORTS ;
    Obtain(Row,Col,Value) ;
    Basetype[Row,Col] := Value ;
    ListBaseFacts(Title,Basetype) ;
    writeln ;
    writeln('Change another value?') ;
  until not Okay ;
end ; (procedure BaseChange)
procedure FactorChange(Title : STR35 ; var Factor : FACTORS ;
                      Numatribs : integer ; Attribute : DESCRIPS) ;
begin
  repeat
    ListOtherFacts(Title,Factor,Numatribs,Attribute) ;
    Row := MAXADKINDS ; Col := Numatribs ;
    Obtain(Row,Col,Value) ;
    Factor[Row,Col] := Value ;
    ListOtherFacts(Title,Factor,Numatribs,Attribute) ;
    writeln ;
    writeln('Change another factor?') ;
  until not Okay ;
end ; (procedure FactorChange)

begin (procedure SetFactors)
  Done := false ;
  repeat
    ShowLists ;
    writeln('Enter letter for parameters to change') ;
    GetXChoice('A','B','C','D','E','X',Choice) ;
    case Choice of
      'A' : BaseChange (' Probability of Engagement Base ',PEbase) ;
      'B' : BaseChange (' Probability of Hit Base ',PHbase) ;
      'C' : FactorChange(' Location Factors ',
        Placefact, MAXADPLACE, Adplaces) ;
      'D' : FactorChange(' Tactics Factors ',
        Tactifact, MAXACTACTS, Actactics) ;
      'E' : FactorChange(' Aspect Factors ',
        Aspecfact, MAXACASPEC, Acaspects) ;
      'X' : Done := true ;
    end ; (case)
  until Done ;
  Choice := '0' (return to Parameter Menu 2)
end ; (procedure SetFactors)

```

```

procedure BaseSaveData(var Choice : char) ;
(
    Main Program Choice 'S'
)
(
    Called from : Parameter Menu 2

    Returns to : Parameter Menu 2

    Purpose      : To save the current base value arrays
                  PEbase and PHbase in the file Basave.

    Description  : This procedure is highly operating system
                  and compiler dependant.
)
begin
    ClearScreen ;
    writeln('Saving the arrays of currently entered' ) ;
    writeln('engagement and hit base values will over-' ) ;
    writeln('write the information currently saved.' ) ;
    writeln ;
    writeln ;
    writeln('Do you wish to continue?') ;
    if OKay then
        begin
            (XX)
                (NO implementation)
                writeln ;
                writeln('This procedure to be implemented by user.' ) ;
            (XX)
                WaitForReturn ;
            end ;
        Choice := '0' ; (return to Parameter Menu 2)
    end ; (procedure BaseSaveData)

```

```

procedure BaseGetData(var Choice : char) ;
(
    (X
        Main Program Choice 'T'
    X)
    (
        Called from : Parameter Menu 2

        Returns to : Parameter Menu 2

        Purpose      : To restore the current base value arrays
                       PEbase and PHbase from the file Basave.

        Description  : This procedure is highly operating system
                       and compiler dependant.
    )
begin
    ClearScreen ;
    writeln('Restoring the arrays of currently entered') ;
    writeln('engagement and hit base values will over-') ;
    writeln('write the information currently in use.') ;
    writeln ;
    writeln ;
    writeln('Do you wish to continue?') ;
    if Okay then
        begin
            (XX)
                (NO implementation)
                writeln ;
                writeln('This procedure to be implemented by user.') ;
            (XX)
                WaitForReturn ;
            end ;
        Choice := '0' ; (return to Parameter Menu 2)
    end ; (procedure BaseGetData)

```

```

procedure FactSaveData(var Choice : char) ;

(%)
                                Main Program Choice 'U'
%)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To save the current factor arrays Placefact,
                Tactifact and Aspectfact in the file Fasave.

  Description : This procedure is highly operating system
                and compiler dependant.
)

begin
  ClearScreen ;
  writeln('Saving the arrays of currently entered') ;
  writeln('factors will overwrite the') ;
  writeln('information currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (%%)
        (NO implementation)
        writeln ;
        writeln('This procedure to be implemented by user.') ;
      (%%)
        WaitforReturn ;
        end ;
      Choice := '0' ; (return to Parameter Menu 2)
    end ; (procedure FactSaveData)

```

```

procedure FactGetData(var Choice : char) ;
(*
                                Main Program Choice 'U'
*)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To restore the current factor arrays Placefact,
                Tactifact and Aspectfact from the file Fasave.

  Description : This procedure is highly operating system
                and compiler dependant.
)
begin
  ClearScreen ;
  writeln('Restoring the arrays of currently entered') ;
  writeln('factors will overwrite the') ;
  writeln('information currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (**)
        (NO implementation)
        writeln ;
        writeln('This procedure to be implemented by user.') ;
      (**)
        WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure FactGetData)

```

```

procedure GetAttritionFact(var Choice : char) ;

(*
                                Main Program Choice 'W'
*)
(
  Called from : Parameter Menu 3

  Returns to  : Parameter Menu 3

  Purpose     : To allow changing of the attrition factor seed.

  Description : The current attrition factor is displayed,
                 the user is told the acceptable value range
                 and is solicited for a new value. The new
                 value is displayed.
)

begin
  WriteTitle('          SET ATTRITION FACTOR          ');
  writeln('Current attrition factor: ',Atritfactor:3:2) ;
  writeln('Initial attrition factor: ',TRITFACTOR:3:2) ;
  writeln ;
  writeln('The attrition factor must be a') ;
  writeln('decimal between 0.00 and 1.00') ;
  writeln('This value may be subtracted') ;
  writeln('from the Air Defense Unit level') ;
  writeln('of effectiveness as a result of') ;
  writeln('an engagement if the Aircraft') ;
  writeln('is not suppressed or destroyed.') ;
  writeln ;
  repeat
    write('Enter new attrition factor > ') ;
    readln(Atritfactor) ;
  until (Atritfactor >= 0.0) and (Atritfactor <= 1.0) ;
  writeln('New attrition factor: ',Atritfactor:3:2) ;
  writeln ;
  WaitForReturn ;
  Choice := 'Q' ; (return to Parameter Menu 3)
end ; (procedure GetAttritionFact)

```

```

procedure SetAttritionProb(var Choice : char) ;
(
    Main Program Choice 'X'
)
(
    Called from : Parameter Menu 3

    Returns to : Parameter Menu 3

    Purpose      : To allow changing of the probability of attrition.

    Description  : The current probability of attrition is displayed,
                  the user is told the acceptable value range
                  and is solicited for a new value. The new
                  value is displayed.
)

begin
    WriteTitle('      SET PROBABILITY OF ATTRITION  ');
    writeln('Current probability of attrition: ',Atritprob:3:2) ;
    writeln('Initial probability of attrition: ',ATTRITPROB:3:2) ;
    writeln ;
    writeln('Probability of attrition must be') ;
    writeln('a decimal between 0.00 and 1.00') ;
    writeln('This value is used to determine') ;
    writeln('if the Air Defense Unit level of') ;
    writeln('effectiveness will be degraded.') ;
    writeln ;
    repeat
        write('Enter new probability of attrition > ');
        readln(Atritprob) ;
    until (Atritprob >= 0.0) and (Atritprob <= 1.0) ;
    writeln('New probability of attrition: ',Atritprob:3:2) ;
    writeln ;
    WaitForReturn ;
    Choice := 'Q' ; (return to Parameter Menu 3)
end ; (procedure SetAttritionProb)

```

```

procedure SetSuppressionProb(var Choice : char) ;
(
    Main Program Choice 'Y'
)
(
    Called from : Parameter Menu 3

    Returns to : Parameter Menu 3

    Purpose      : To allow changing of the probability of suppression.

    Description  : The current probability of suppression is displayed,
                  the user is told the acceptable value range
                  and is solicited for a new value. The new
                  value is displayed.
)

begin
    WriteTitle(' SET PROBABILITY OF SUPPRESSION ') ;
    writeln('Current probability of suppression: ',Suprsprob:3:2) ;
    writeln('Initial probability of suppression: ',SUPRESPROB:3:2) ;
    writeln ;
    writeln('Probability of suppression must be') ;
    writeln('a decimal between 0.00 and 1.00') ;
    writeln('This value is used to determine if') ;
    writeln('the Aircraft will be suppressed or') ;
    writeln('destroyed after being engaged. ') ;
    writeln ;
    repeat
        write('Enter new probability of suppression > ') ;
        readln(Suprsprob) ;
    until (Suprsprob >= 0.0) and (Suprsprob <= 1.0) ;
    writeln('New probability of suppression: ',Suprsprob:3:2) ;
    writeln ;
    WaitForReturn ;
    Choice := 'Q' ; (return to Parameter Menu 3)
end ; (procedure SetSuppressionProb)

```

```

procedure Initialize(var Choice : char) ;

(%)
                                Main Program Initialize
(%)
(
  Called from : Main Program

  Returns to   : Main Menu

  Purpose      : To initialize the random number stream, probabilities
                 of attrition and suppression, and all data arrays.

  Description  : The pertinent factors for engagements are set to the
                 appropriate constants, the record elements for the
                 Air Defense Unit array Adunit and Aircraft array Aircraft
                 are set to 0, and descriptive and factor data arrays are
                 initialized. The array initializations are in separate
                 procedures to facilitate compilation and changing of the
                 various factors.
)

var Counter : integer ;

procedure DAInitialize ;
begin
  (Initialize Descriptive Arrays)
  Adkinds [0] := 'Unspecified' ;
  Adkinds [1] := 'Vulcan      ' ;
  Adkinds [2] := 'SGT York   ' ;
  Adkinds [3] := 'Redeye     ' ;
  Adkinds [4] := 'Stinger    ' ;
  Adkinds [5] := 'Chaparral  ' ;

  Adplaces [0] := 'Unspecified' ;
  Adplaces [1] := 'Europe     ' ;
  Adplaces [2] := 'Desert     ' ;
  Adplaces [3] := 'Jungle     ' ;

  Ackinds [0] := 'Unspecified' ;
  Ackinds [1] := 'Helicopter  ' ;
  Ackinds [2] := 'Transport   ' ;
  Ackinds [3] := 'Fighter    ' ;

  Actactics[0] := 'Unspecified' ;
  Actactics[1] := 'Pop-Up     ' ;
  Actactics[2] := 'Lay-Down   ' ;
  Actactics[3] := 'Fly-Over   ' ;

  Acaspects[0] := 'Unspecified' ;
  Acaspects[1] := 'Head-On    ' ;
  Acaspects[2] := 'Crossing   ' ;
  Acaspects[3] := 'Tail      ' ;
end ; (procedure DAInitialize)

```

```

procedure PEInitialize ;
begin (Initialize Probability of Engagement Array)
  PEbase[1,1] := 5/6 ;
  PEbase[2,1] := 0.9 ;
  PEbase[3,1] := 0.5 ;
  PEbase[4,1] := 0.6 ;
  PEbase[5,1] := 0.5 ;
  PEbase[1,2] := 4/6 ;
  PEbase[2,2] := 0.7 ;
  PEbase[3,2] := 4/6 ;
  PEbase[4,2] := 0.7 ;
  PEbase[5,2] := 4/6 ;
  PEbase[1,3] := 0.5 ;
  PEbase[2,3] := 0.6 ;
  PEbase[3,3] := 0.5 ;
  PEbase[4,3] := 0.6 ;
  PEbase[5,3] := 0.5 ;
end ; (procedure PEInitialize)

```

```

procedure PHInitialize ;
begin (Initialize Probability of Hit Array)
  PHbase[1,1] := 0.5 ;
  PHbase[2,1] := 0.6 ;
  PHbase[3,1] := 1/6 ;
  PHbase[4,1] := 0.2 ;
  PHbase[5,1] := 2/6 ;
  PHbase[1,2] := 0.5 ;
  PHbase[2,2] := 0.6 ;
  PHbase[3,2] := 2/6 ;
  PHbase[4,2] := 0.4 ;
  PHbase[5,2] := 2/6 ;
  PHbase[1,3] := 2/6 ;
  PHbase[2,3] := 0.4 ;
  PHbase[3,3] := 2/6 ;
  PHbase[4,3] := 0.4 ;
  PHbase[5,3] := 2/6 ;
end ; (procedure PHInitialize)

```

```

procedure LFInitialize ;
begin
    (Initialize Location Factor Array)
    Placefact[1,1] := 1.0 ;
    Placefact[2,1] := 1.0 ;
    Placefact[3,1] := 1.0 ;
    Placefact[4,1] := 1.0 ;
    Placefact[5,1] := 1.0 ;
    Placefact[1,2] := 1.1 ;
    Placefact[2,2] := 1.1 ;
    Placefact[3,2] := 1.1 ;
    Placefact[4,2] := 1.1 ;
    Placefact[5,2] := 1.1 ;
    Placefact[1,3] := 0.8 ;
    Placefact[2,3] := 0.8 ;
    Placefact[3,3] := 0.8 ;
    Placefact[4,3] := 0.8 ;
    Placefact[5,3] := 0.8 ;
end ; (procedure LFInitialize)

```

```

procedure TFInitialize ;
begin
    (Initialize Tactic Factor Array)
    Tactifact[1,1] := 0.9 ;
    Tactifact[2,1] := 0.9 ;
    Tactifact[3,1] := 0.9 ;
    Tactifact[4,1] := 0.9 ;
    Tactifact[5,1] := 0.9 ;
    Tactifact[1,2] := 1.0 ;
    Tactifact[2,2] := 1.0 ;
    Tactifact[3,2] := 1.0 ;
    Tactifact[4,2] := 1.0 ;
    Tactifact[5,2] := 1.0 ;
    Tactifact[1,3] := 1.1 ;
    Tactifact[2,3] := 1.1 ;
    Tactifact[3,3] := 1.1 ;
    Tactifact[4,3] := 1.1 ;
    Tactifact[5,3] := 1.1 ;
end ; (procedure TFInitialize)

```

```

procedure AFInitialize ;
begin
    Aspectfact[1,1] := 0.9 ;
    Aspectfact[2,1] := 0.9 ;
    Aspectfact[3,1] := 0.9 ;
    Aspectfact[4,1] := 0.9 ;
    Aspectfact[5,1] := 0.9 ;
    Aspectfact[1,2] := 1.1 ;
    Aspectfact[2,2] := 1.1 ;
    Aspectfact[3,2] := 1.1 ;
    Aspectfact[4,2] := 1.1 ;
    Aspectfact[5,2] := 1.1 ;
    Aspectfact[1,3] := 0.9 ;
    Aspectfact[2,3] := 0.9 ;
    Aspectfact[3,3] := 0.9 ;
    Aspectfact[4,3] := 0.9 ;
    Aspectfact[5,3] := 0.9 ;
end ; (procedure AFInitialize)

begin (procedure Initialize)
    Seed      := STREAMSEED ;      (Seed random number stream)
    Atritfactor := TRITFACTOR ;    (Set attrition factors)
    Atritprob  := ATTRITPROB ;    (Set attrition probability)
    Suprsprob  := SUPRESPROB ;    (Set suppression probability)

    for Counter := 1 to MAXADUNITS do (Initialize Air Defense Unit Data Array)
        with Adunit[Counter] do
            begin
                Isactive := false ;
                Kind      := 0 ;
                Location  := 0 ;
                Effectlevel := 0.0
            end ; (for/with)

    for Counter := 1 to MAXAIRCRAF do (Initialize Aircraft Data Array)
        with Aircraft[Counter] do
            begin
                Isactive := false ;
                Sort     := 0 ;
                Tactic   := 0 ;
                Aspect   := 0 ;
            end ; (for/with)

    (Inititalize descriptive, probability of engagement/hit and factor arrays)

    DAInitialize ;
    PEInitialize ;
    PHInitialize ;
    LFInitialize ;
    TFInitialize ;
    AFInitialize ;
    Choice := 'A' ; (Start program with Main Menu)
end ; (procedure Initialize)

```

```

procedure HelpOne(var Choice : char);

(
    Main Program Choice '1'
)

(
    Called from : Main Menu

    Returns to : Main Menu

    Purpose      : To display information about the current screen choices.

    Description  : Text help information is displayed.
)

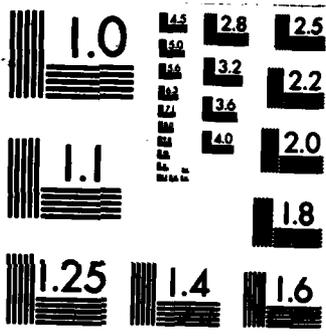
begin
    ClearScreen ;
    writeln('      Main Menu Help') ;
    writeln ;
    writeln('      The Air Defense War Game') ;
    writeln('Support Program is designed to') ;
    writeln('provide probabilities of') ;
    writeln('engagement and kill for a') ;
    writeln('variety of forward air defense') ;
    writeln('systems against different') ;
    writeln('types of aircraft. Each air') ;
    writeln('defense system is described by') ;
    writeln('type (Vulcan, SGT York,') ;
    writeln('Redeye, Stinger, or') ;
    writeln('Chaparral), location (Europe,') ;
    writeln('Desert or Jungle) and') ;
    writeln('effectiveness level (0.0 to') ;
    writeln('1.0).') ;
    WaitForReturn ;
    writeln('      Each aircraft is') ;
    writeln('described by type (Helicopter,') ;
    writeln('Transport or Fighter), tactic') ;
    writeln('(Pop-up, Lay-Down or Fly-') ;
    writeln('Over), and profile or aspect') ;
    writeln('presented to the air defense') ;
    writeln('system (Head-on, Crossing, or') ;
    writeln('Tail). As currently') ;
    writeln('programmed, 20 numbered Air') ;
    writeln('Defense Unit and 20 numbered') ;
    writeln('Aircraft descriptions') ;
    writeln('consisting of the above') ;
    writeln('attributes may be entered and') ;
    writeln('the probabilities of') ;
    writeln('engagement and kill calculated') ;
    writeln('for any entered air defense') ;
    writeln('system against any entered') ;
    writeln('aircraft.') ;
    WaitForReturn ;

```

```

writeln('    The results of an') ;
writeln('interaction between an Air') ;
writeln('Defense Unit and an Aircraft') ;
writeln('are determined by comparing a') ;
writeln('program generated random') ;
writeln('number with calculated') ;
writeln('threshold values. Random') ;
writeln('numbers are generated for each') ;
writeln('of the engagement, kill,') ;
writeln('suppression and attrition') ;
writeln('results. Threshold values are') ;
writeln('calculated separately for the') ;
writeln('probabilities of engagement') ;
writeln('and kill. The threshold') ;
writeln('values are calculated by') ;
writeln('multiplying a base value by') ;
writeln('the series of factors') ;
writeln('reflecting the descriptions of') ;
writeln('the Air Defense Unit and the') ;
writeln('Aircraft.') ;
WaitforReturn ;
writeln('    The displayed') ;
writeln('results of an engagement') ;
writeln('include if the aircraft is') ;
writeln('engaged, if the aircraft is') ;
writeln('hit and destroyed or mission') ;
writeln('suppressed, and if the air') ;
writeln('defense system suffers any') ;
writeln('attrition from an aircraft') ;
writeln('that was engaged but not') ;
writeln('destroyed or suppressed.') ;
writeln('Attrition on an Air Defense') ;
writeln('Unit is reflected by a') ;
writeln('lowering of the Unit's') ;
writeln('effectiveness level by the') ;
writeln('amount of the attrition') ;
writeln('factor.') ;
WaitforReturn ;
writeln('    Within the program the') ;
writeln('factors used in the') ;
writeln('calculation of the engagement') ;
writeln('and kill thresholds may be') ;
writeln('changed. Additionally, the') ;
writeln('probabilities of air defense') ;
writeln('system attrition and aircraft') ;
writeln('suppression, the attrition') ;
writeln('factor (that value subtracted') ;
writeln('from the air defense system') ;
writeln('effectiveness level) and the') ;
writeln('random number seed may be') ;
writeln('changed. Resetting the random') ;
writeln('number seed to a previous') ;
writeln('value permits an engagement to') ;

```

MICROCOPY RESOLUTION TEST CHART
 NATIONAL BUREAU OF STANDARDS-1963-A

```

writeln('be conducted again with') ;
writeln('unchanged parameters or with') ;
writeln('new factors. Each of the') ;
writeln('factors mentioned above may be') ;
writeln('changed from one of the') ;
writeln('parameter menus.') ;
WaitforReturn ;
writeln(' If file input and output') ;
writeln('operations are implemented,') ;
writeln('the data arrays for the Air') ;
writeln('Defense Units and Aircraft,') ;
writeln('and the arrays for threshold') ;
writeln('values and factors may be') ;
writeln('saved to or restored from') ;
writeln('files. Use of external files') ;
writeln('for storage of the data arrays') ;
writeln('permits several sets of data') ;
writeln('to be entered and changed') ;
writeln('during a program run without') ;
writeln('having to enter the individual') ;
writeln('values each time a new data') ;
writeln('set is desired.') ;
WaitforReturn ;
writeln('Starting Up') ;
writeln ;
writeln(' When the program is') ;
writeln('started no air defense systems') ;
writeln('or aircraft are available for') ;
writeln('engagements but a set of') ;
writeln('default values for the') ;
writeln('calculation of engagement') ;
writeln('results is loaded. From the') ;
writeln('Main Menu either option A or B') ;
writeln('should be selected to proceed') ;
writeln('with the descriptions of air') ;
writeln('defense systems and aircraft.') ;
writeln('Selection of option D allows') ;
writeln('the default factors to be') ;
writeln('displayed or changed.') ;
WaitforReturn ;
writeln('Conducting Engagements') ;
writeln ;
writeln(' From the Main Menu,') ;
writeln('option C, Conduct Engagements,') ;
writeln('is selected after both Air') ;
writeln('Defense Units and Aircraft are') ;
writeln('entered. The user is prompted') ;
writeln('for the Air Defense Unit and') ;
writeln('the Aircraft numbers for the') ;
writeln('engagement. Only numbers for') ;
writeln('Units and Aircraft that have') ;
writeln('been described or 0 to exit') ;
writeln('are accepted. If an invalid') ;

```

```

writeln('number is entered, a list of') ;
writeln('valid numbers is displayed and') ;
writeln('the prompt redisplayed.') ;
WaitforReturn ;
writeln('                When') ;
writeln('valid numbers for the Air') ;
writeln('Defense Unit and the Aircraft') ;
writeln('are entered, characteristics') ;
writeln('of both systems are listed and') ;
writeln('the results of the engagement') ;
writeln('are displayed. The first') ;
writeln('result is whether the Aircraft') ;
writeln('was engaged by the Air Defense') ;
writeln('Unit. If the Aircraft was not') ;
writeln('engaged, the engagement') ;
writeln('process is finished and the') ;
writeln('user is asked if another') ;
writeln('engagement is to be conducted.') ;
WaitforReturn ;
writeln('If the Aircraft was engaged,') ;
writeln('whether or not the Aircraft') ;
writeln('was hit is displayed, and, if') ;
writeln('the Aircraft was hit, whether') ;
writeln('it was destroyed or its') ;
writeln('mission aborted is determined.') ;
writeln('If the Aircraft was engaged') ;
writeln('but not hit, whether or not') ;
writeln('the Air Defense Unit was') ;
writeln('suppressed is displayed, and') ;
writeln('if the Unit was suppressed,') ;
writeln('the effectiveness is decreased') ;
writeln('by the amount of the attrition') ;
writeln('factor.') ;
WaitforReturn ;
writeln('                When the engagement') ;
writeln('process is concluded and the') ;
writeln('results displayed, the user is') ;
writeln('given the option of conducting') ;
writeln('another engagement. If no') ;
writeln('other engagement is to be') ;
writeln('conducted, the user is') ;
writeln('returned to the Main Menu.') ;
writeln ;
writeln('Ending The Program') ;
writeln ;
writeln('                The program is ended from') ;
writeln('the Main Menu by selecting') ;
writeln('option E, Exit Program.') ;
WaitforReturn ;
Choice := 'A'
end ; (procedure HelpOne)

```

```

procedure HelpTwo(var Choice : char);
(
    Main Program Choice '2'
)
(
    Called from : Air Defense Menu

    Returns to : Air Defense Menu

    Purpose      : To display information about the current screen choices.

    Description  : Text help information is displayed.
)
begin
    ClearScreen ;
    writeln(' Air Defense Menu Help') ;
    writeln ;
    writeln('Entering Air Defense Systems') ;
    writeln ;
    writeln('    Option A from the Main') ;
    writeln('Menu brings up the Air Defense') ;
    writeln('Menu. From this menu, options') ;
    writeln('A or D are selected to enter') ;
    writeln('Air Defense Unit descriptions.') ;
    writeln('Option D may only be used if') ;
    writeln('the file input/output') ;
    writeln('procedures are implemented and') ;
    writeln('the program has been run') ;
    writeln('previously and the Air Defense') ;
    writeln('Unit descriptions saved. When') ;
    writeln('option A, Enter Description of') ;
    writeln('Air Defense Units, is') ;
    writeln('selected, the user is told') ;
    writeln('what the next available') ;
    writeln('(undescribed) Unit number is') ;
    writeln('and asked to enter the number') ;
    writeln('of the Unit to describe.') ;
    WaitForReturn ;
    writeln('There is no requirement to') ;
    writeln('enter Air Defense Units in any') ;
    writeln('order, but entering the number') ;
    writeln('of a Unit that was previously') ;
    writeln('described will result in the') ;
    writeln('new information overwriting') ;
    writeln('the previous description.') ;
    writeln('There is also no requirement') ;
    writeln('to enter any set amount of Air') ;
    writeln('Defense Units - a single Unit') ;
    writeln('may be entered and an') ;
    writeln('engagement conducted with an') ;
    writeln('entered Aircraft. A list of') ;

```

```

writeln('the currently entered Air') ;
writeln('Defense Units is seen by') ;
writeln('selecting option B of the Air') ;
writeln('Defense Menu, List Currently') ;
writeln('Entered Air Defense Units.') ;
writeln('Entering an Air Defense Unit') ;
writeln('number of 0 returns the user') ;
writeln('to the Air Defense Menu.') ;
WaitforReturn ;
writeln('Options on the Air') ;
writeln('Defense Menu include C, Save') ;
writeln('Data and D, Restore Data. If') ;
writeln('the program has been run') ;
writeln('previously and option C, Save') ;
writeln('Data, was used, descriptions') ;
writeln('of Air Defense Units may be') ;
writeln('entered from the external file') ;
writeln('ADSAVE. Two items of caution:') ;
writeln('if the file ADSAVE does not') ;
writeln('exist, the program aborts to') ;
writeln('the operating system, and if') ;
writeln('data is restored from the') ;
writeln('file ADSAVE, any descriptions') ;
writeln('of Air Defense Units currently') ;
writeln('in use are overwritten. In') ;
writeln('like manner, data saved') ;
writeln('overwrites the data currently') ;
writeln('in the file ADSAVE. The user') ;
writeln('may make a decision not to') ;
writeln('continue with saving or') ;
writeln('restoring the Air Defense Unit') ;
writeln('data at which time the program') ;
writeln('returns to the Air Defense') ;
writeln('Menu.') ;
WaitforReturn ;
writeln('The process of describing') ;
writeln('an Air Defense Unit is a') ;
writeln('matter of selecting the') ;
writeln('appropriate items from the') ;
writeln('subsequent menus. The initial') ;
writeln('choice is for the type of') ;
writeln('system, the second choice is') ;
writeln('for the location of the') ;
writeln('system. The third choice,') ;
writeln('effectiveness level, may be') ;
writeln('thought of as a decimal') ;
writeln('percent of effectiveness, so') ;
writeln('if the Unit is considered 100%') ;
writeln('effective, 1 or 1.0 is entered') ;
writeln('for the effectiveness level.') ;
writeln('In like manner, a 75%') ;
writeln('effective Unit is described') ;
writeln('with an effectiveness level of') ;

```

```

writeln('0.75. Caution should be') ;
writeln('exercised to enter only') ;
writeln('numbers for the effectiveness') ;
writeln('level, as a non-numeric entry') ;
writeln('causes the program to abort to') ;
writeln('the operating system.') ;
WaitforReturn ;
writeln('    When a description is') ;
writeln('completed, a Unit number of 0') ;
writeln('is entered to return to the') ;
writeln('Air Defense Menu. Option B,') ;
writeln('List Currently Entered Air') ;
writeln('Defense Units, may be made to') ;
writeln('verify that the correct') ;
writeln('information has been entered.') ;
writeln('Incorrect information is') ;
writeln('corrected by selecting option') ;
writeln('A at the Air Defense Menu,') ;
writeln('entering the Unit number for') ;
writeln('which the information was') ;
writeln('incorrect, and reselecting the') ;
writeln('menu items to complete the') ;
writeln('description of the Unit.') ;
WaitforReturn ;
Choice := 'B'
end ; (procedure HelpTwo)

```

```

procedure HelpThree(var Choice : char);
(
    Main Program Choice '3'
)
(
    Called from : Aircraft Menu

    Returns to : Aircraft Menu

    Purpose      : To display information about the current screen choices.

    Description  : Text help information is displayed.
)
begin
    ClearScreen ;
    writeln(' Aircraft Menu Help') ;
    writeln ;
    writeln('Entering Aircraft') ;
    writeln ;
    writeln(' Option B from the Main') ;
    writeln('Menu brings up the Aircraft') ;
    writeln('Menu. From this menu, options') ;
    writeln('A or D are selected to enter') ;
    writeln('Aircraft descriptions. Option') ;
    writeln('D may only be used if the file') ;
    writeln('input/output procedures are') ;
    writeln('implemented and the program') ;
    writeln('has been run previously and') ;
    writeln('the Aircraft descriptions') ;
    writeln('saved. When option A, Enter') ;
    writeln('Description of Aircraft, is') ;
    writeln('selected, the user is told') ;
    writeln('what the next available') ;
    writeln('(undescribed) Aircraft number') ;
    writeln('is and asked to enter the') ;
    writeln('number of the Aircraft to') ;
    writeln('describe.') ;
    WaitForReturn ;
    writeln(' There is no') ;
    writeln('requirement to enter Aircraft') ;
    writeln('in any order, but entering the') ;
    writeln('number of an Aircraft that was') ;
    writeln('previously described results') ;
    writeln('in the new information') ;
    writeln('overwriting the previous') ;
    writeln('description. There is also no') ;
    writeln('requirement to enter any set') ;
    writeln('amount of Aircraft - a single') ;
    writeln('Aircraft may be entered and an') ;
    writeln('engagement conducted with an') ;
    writeln('entered Air Defense Unit. A') ;

```

```
writeln('list of the currently entered') ;
writeln('Aircraft is seen by selecting') ;
writeln('option B of the Aircraft, List') ;
writeln('Currently Entered Aircraft.') ;
writeln('Entering an Aircraft number of') ;
writeln('0 returns the user to the') ;
writeln('Aircraft Menu.') ;
WaitforReturn ;
writeln('Options on the Aircraft') ;
writeln('Menu include C, Save Data and') ;
writeln('D, Restore Data. If the') ;
writeln('program has been run') ;
writeln('previously and option C, Save') ;
writeln('Data, was used, descriptions') ;
writeln('of Aircraft may be entered') ;
writeln('from the external file ACSAVE.') ;
writeln('Two items of caution: if the') ;
writeln('file ACSAVE does not exist,') ;
writeln('the program aborts to the') ;
writeln('operating system, and if data') ;
writeln('is restored from the file') ;
writeln('ACSAVE, any descriptions of') ;
writeln('Aircraft currently in use are') ;
writeln('overwritten. In like manner,') ;
writeln('data saved overwrites the data') ;
writeln('currently in the file ACSAVE.') ;
writeln('The user may make a decision') ;
writeln('not to continue with saving or') ;
writeln('restoring the Aircraft data at') ;
writeln('which time the program returns') ;
writeln('to the Aircraft Menu.') ;
```

```

WaitforReturn ;
writeln('    The process of describing') ;
writeln('an Aircraft is a matter of') ;
writeln('selecting the appropriate') ;
writeln('items from the subsequent') ;
writeln('menus. The initial choice is') ;
writeln('for the type of aircraft, the') ;
writeln('second choice is for the') ;
writeln('tactic used and the third') ;
writeln('choice is for the aspect the') ;
writeln('Aircraft is presenting to the') ;
writeln('Air Defense Unit. When a') ;
writeln('description is completed, an') ;
writeln('Aircraft number of 0 is') ;
writeln('entered to return to the Air') ;
writeln('Defense Menu. Option B, List') ;
writeln('Currently Entered Aircraft,') ;
writeln('may be made to verify that the') ;
writeln('correct information has been') ;
writeln('entered.') ;
WaitforReturn ;
writeln('                Incorrect') ;
writeln('information is corrected by') ;
writeln('selecting option A on the') ;
writeln('Aircraft Menu, entering the') ;
writeln('Aircraft number for which the') ;
writeln('information was incorrect, and') ;
writeln('reselecting the menu items to') ;
writeln('complete the description of') ;
writeln('the Aircraft.') ;
WaitforReturn ;
Choice := 'C'
end ; {procedure HelpThree}

```

```

procedure HelpFour(var Choice : char);

(X
                                Main Program Choice '4'
X)
(
  Called from : Parameter Menu

  Returns to   : Parameter Menu

  Purpose      : To display information about the current screen choices.

  Description  : Text help information is displayed.
)

begin
  ClearScreen ;
  writeln(' Parameter Menu Help') ;
  writeln ;
  writeln('Changing Parameters') ;
  writeln ;
  writeln(' Option D from the Main') ;
  writeln('Menu brings up the Parameter') ;
  writeln('Menu. Parameters refer to') ;
  writeln('the base values for the') ;
  writeln('probabilities of engagement') ;
  writeln('and hit, and the factor values') ;
  writeln('for each of the Air Defense') ;
  writeln('Unit and Aircraft descriptors') ;
  writeln('(with the exception of the Air') ;
  writeln('Defense Unit effectiveness') ;
  writeln('level). Other values that may') ;
  writeln('be changed include the') ;
  writeln('probabilities of Air Defense') ;
  writeln('Unit attrition and Aircraft') ;
  writeln('suppression, the attrition') ;
  writeln('factor and the seed for the') ;
  writeln('random number generator.') ;
  WaitForReturn ;
  writeln(' Option A from Parameter') ;
  writeln('Menu 1 brings up the Parameter') ;
  writeln('Lists from which a set of') ;
  writeln('threshold values or') ;
  writeln('description factors may be') ;
  writeln('selected to be listed. Once a') ;
  writeln('set of values is displayed,') ;
  writeln('the user is asked if another') ;
  writeln('set is to be displayed. If the') ;
  writeln('Parameter Lists are showing') ;
  writeln('and the user does not choose') ;
  writeln('to continue, option X to Exit') ;
  writeln('returns the user to the') ;
  writeln('previous menu.') ;

```

```

WaitforReturn ;
writeln('      Selection of option B') ;
writeln('(Set/Save Parameters) from') ;
writeln('Parameter Menu 1 brings up') ;
writeln('Parameter Menu 2. Option A,') ;
writeln('Change Parameters, displays') ;
writeln('the      Parameter      Lists.')
```

```

writeln('Selection of a parameter list') ;
writeln('allows values used by the') ;
writeln('program to be changed by') ;
writeln('specifying the row and column') ;
writeln('of value to change, and the') ;
writeln('new value. The new value is') ;
writeln('displayed and the user given') ;
writeln('the option of changing another') ;
writeln('value. Options at Parameter')
```

```

writeln('Menu 2 also include B, Save') ;
writeln('Base Values; C, Restore Base') ;
writeln('Values; D, Save Factors and E,')
```

```

writeln('Restore Factors.')
```

```

WaitforReturn ;
writeln('      If the') ;
writeln('program has been run') ;
writeln('previously and the save') ;
writeln('options were used, base values') ;
writeln('may be restored from the file')
```

```

writeln('BASAVE and factor values may')
```

```

writeln('be restored from the file')
```

```

writeln('FASAVE. Two items of caution:')
```

```

writeln('if the files do not exist, the')
```

```

writeln('program aborts to the')
```

```

writeln('operating system; and if data')
```

```

writeln('is restored from the files,')
```

```

writeln('any data currently in use is')
```

```

writeln('overwritten. In like manner,')
```

```

writeln('data saved overwrites the data')
```

```

writeln('currently in the files. The')
```

```

writeln('user may make a decision not')
```

```

writeln('to continue with saving or')
```

```

writeln('restoring the data at which')
```

```

writeln('time the program returns to')
```

```

writeln('Parameter Menu 2.')
```

```

WaitforReturn ;
writeln('      If')
```

```

writeln('Parameter Menu 2 is showing')
```

```

writeln('and the user does not choose')
```

```

writeln('to continue, option X to Exit')
```

```

writeln('returns the user to the')
```

```

writeln('previous menu.')
```

```

writeln ;
writeln('      Selection of option C,')
```

```

writeln('Set Random Number Seed or D,')
```

```

writeln('Set      Attrition/Suppression')
```

```
writeln('Factors from Parameter Menu 1') ;
writeln('permits those values to be') ;
writeln('changed. Option D brings up') ;
writeln('Parameter Menu 3.') ;
WaitforReturn ;
writeln(' For each option on') ;
writeln('Parameter Menu 3 and option C,') ;
writeln('Set Random Number Seed, from') ;
writeln('Parameter Menu 1, the current') ;
writeln('and original factors and') ;
writeln('permissible range for a new') ;
writeln('value are displayed.') ;
WaitforReturn ;
Choice := 'E'
end ; (procedure HelpFour)
```

```
begin ( MAIN PROGRAM )
  ShowProgramName ;
  WaitforReturn ;
  Initialize(Choice) ;
  repeat
```

```

case Choice of
  'A' : WriteMainMenu      (Choice) ;

      ( Main Menu Choices      )

  'B' : WriteDefenseMenu (Choice) ;
  'C' : WriteAircraftMenu (Choice) ;
  'D' : ConductEngagements(Choice) ;
  'E' : ChangeParameters (Choice) ;

      ( Air Defense Menu Choices )

  'F' : EnterADunits      (Choice) ;
  'G' : ListADunits       (Choice) ;
  'H' : AdSaveData        (Choice) ;
  'I' : AdGetData         (Choice) ;

      ( Aircraft Menu Choices   )

  'J' : EnterAircraft     (Choice) ;
  'K' : ListAircraft      (Choice) ;
  'L' : AcSaveData        (Choice) ;
  'M' : AcGetData         (Choice) ;

      ( Parameter Menu 1 Choices )

  'N' : ListFactors       (Choice) ;
  'O' : WriteParaMenu2    (Choice) ;
  'P' : SetSeed           (Choice) ;
  'Q' : SetAttritSupMenu  (Choice) ;

      ( Parameter Menu 2 Choices )

  'R' : SetFactors        (Choice) ;
  'S' : BaseSaveData      (Choice) ;
  'T' : BaseGetData       (Choice) ;
  'U' : FactSaveData      (Choice) ;
  'V' : FactGetData       (Choice) ;

      ( Parameter Menu 3 Choices )

  'W' : GetAttritionFact  (Choice) ;
  'X' : SetAttritionProb  (Choice) ;
  'Y' : SetSuppressionProb(Choice) ;

      ( Help Choices           )

  '1' : HelpOne           (Choice) ;
  '2' : HelpTwo           (Choice) ;
  '3' : HelpThree        (Choice) ;
  '4' : HelpFour          (Choice) ;
end (case)
until Choice = '.' ;

```

ShowProgramName
end.

(MAIN PROGRAM)

APPENDIX E: Spare Parts

	Page
Apple Pascal Procedures	E-3
'H' : AdSaveData	E-4
'I' : AdGetData	E-5
'L' : AcSaveData	E-6
'M' : AcGetData	E-7
'S' : BaseSaveData	E-8
'T' : BaseGetData	E-9
'U' : FactSaveData	E-10
'V' : FactGetData	E-11
TURBO Pascal Procedures	E-12
'H' : AdSaveData	E-13
'I' : AdGetData	E-14
'L' : AcSaveData	E-15
'M' : AcGetData	E-16
'S' : BaseSaveData	E-17
'T' : BaseGetData	E-18
'U' : FactSaveData	E-19
'V' : FactGetData	E-20
UNIX Pascal Procedures	E-21
'H' : AdSaveData	E-22
'I' : AdGetData	E-23
'L' : AcSaveData	E-24
'M' : AcGetData	E-25
'S' : BaseSaveData	E-26
'T' : BaseGetData	E-27
'U' : FactSaveData	E-28
'V' : FactGetData	E-29

Spare Parts

Procedures are provided in this appendix to enable file input and output for the Air Defense War Game Support Program under different compilers. The compilers supported are the TURBO, Apple and UNIX Pascal compilers. The specific procedures to be changed are listed below as they are found in the main program. The file handling procedures enable the saving and restoring of the arrays of ADRECORD, ACRECORD and the arrays of the base and factor values. Replacement of the procedures must be accompanied by addition of the respective file names to the initial program statement and to the variable section. Files are of type text.

(Air Defense Menu Choices)

'H' : AdSaveData (Choice) ;
'I' : AdGetData (Choice) ;

(Aircraft Menu Choices)

'L' : AcSaveData (Choice) ;
'M' : AcGetData (Choice) ;

(Parameter Menu 2 Choices)

'S' : BaseSaveData (Choice) ;
'T' : BaseGetData (Choice) ;
'U' : FactSaveData (Choice) ;
'V' : FactGetData (Choice) ;

Apple Pascal Procedures

The procedures on the following pages are designed for the Apple Pascal compiler.

```

procedure AdSaveData(var Choice : char) ;

(*
                                Main Program Choice 'H'
*)
(
  Called from : Air Defense Menu

  Returns to  : Air Defense Menu

  Purpose     : To save the current Air Defense Unit array Adunit
                in the file Adsave.

  Description : This procedure is for the Apple Pascal compiler.
)
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Saving the array of currently entered') ;
  writeln('Air Defense Units will overwrite the') ;
  writeln('information currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (Apple implementation)
      rewrite(Adsave,'ADSAVE') ;
      for Counter := 1 to MAXADUNITS do
        with Adunit[Counter] do
          begin
            writeln(Adsave,Isactive,Kind:2,Location:2,Effectlevel:4:2) ;
          end ;
        close(Adsave,lock) ;
        writeln ;
        writeln('Air Defense Unit data has been saved.') ;
        WaitForReturn ;
      end ;
      Choice := 'B' ; (return to the Air Defense Unit menu)
    end ; (procedure AdSaveData)

```

```

procedure AdGetData(var Choice : char) ;

  (X
    Main Program Choice 'I'
  X)
  {
    Called from : Air Defense Menu

    Returns to : Air Defense Menu

    Purpose      : To restore the current Air Defense Unit array Adunit
                  from the file Adsave.

    Description  : This procedure is for the Apple Pascal compiler.
  }
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Restoring Air Defense Unit Data') ;
  writeln('will overwrite the information') ;
  writeln('currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (Apple implementation)
      reset(Adsave, 'ADSAVE') ;
      for Counter := 1 to MAXADUNITS do
        with Adunit[Counter] do
          begin
            readln(Adsave, Isactive, Kind, Location, Effectlevel) ;
          end ;
        close(Adsave, lock) ;
        writeln ;
        writeln('Air Defense Unit data has been restored.') ;
        WaitForReturn ;
      end ;
      Choice := 'B' ; (return to Air Defense Unit menu)
    end ; (procedure AdGetData)

```

```

procedure AcSaveData(var Choice : char) ;

(%)
      Main Program Choice 'L'
(%)
{
  Called from : Aircraft Menu

  Returns to  : Aircraft Menu

  Purpose     : To save the current Aircraft array Aircraft
               in the file Acsave.

  Description : This procedure is for the Apple Pascal compiler.
}
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Saving the array of currently entered') ;
  writeln('Aircraft will overwrite the information') ;
  writeln('currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (Apple implementation)
      rewrite(Acsave,'ACSAVE') ;
      for Counter := 1 to MAXAIRCRAF do
        with Aircraft[Counter] do
          begin
            writeln(Acsave,Isactive,Sort:2,Tactic:2,Aspect:2) ;
          end ;
        close (Acsave,lock) ;
        writeln ;
        writeln('Aircraft data has been saved.') ;
        WaitForReturn ;
      end ;
    Choice := 'C' ; (return to the Aircraft menu)
  end ; (procedure AcSaveData)

```

```

procedure AcGetData(var Choice : char) ;

(%)
      Main Program Choice 'M'
(%)
{
  Called from : Aircraft Menu

  Returns to  : Aircraft Menu

  Purpose    : To restore the current Aircraft array Aircraft
              from the file Acsave.

  Description : This procedure is for the Apple Pascal compiler.
}
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Restoring the Aircraft data') ;
  writeln('will overwrite the information') ;
  writeln('currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (Apple implementation)
      reset(Acsave,'ACSAVE') ;
      for Counter := 1 to MAXAIRCRAF do
        with Aircraft[Counter] do
          begin
            readln(Acsave,Isactive,Sort,Tactic,Aspect) ;
          end ;
        close(Acsave,lock) ;
        writeln ;
        writeln('Aircraft data has been restored.') ;
        WaitForReturn ;
      end ;
      Choice := 'C' ; (return to Aircraft menu)
    end ; (procedure AcGetData)

```

```

procedure BaseSaveData(var Choice : char) ;

(%)
                                Main Program Choice 'S'
%)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To save the current base value arrays
                PEbase and PHbase in the file Basave.

  Description : This procedure is for the Apple Pascal compiler.
)
var
  AdCount, AcCount : integer ;

begin
  ClearScreen ;
  writeln('Saving the arrays of currently entered' ) ;
  writeln('engagement and hit base values will over-' ) ;
  writeln('write the information currently saved.' ) ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (Apple implementation)
      rewrite(Basave,'BASAVE') ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          write(Basave,PEbase[AdCount,AcCount]:4:2) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          write(Basave,PHbase[AdCount,AcCount]:4:2) ;
      close (Basave,lock) ;
      writeln ;
      writeln('Engagement and Hit threshold') ;
      writeln('data has been saved.' ) ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure BaseSaveData)

```

```

procedure BaseGetData(var Choice : char) ;

(%)
      Main Program Choice 'T'
%)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To restore the current base value arrays
                PEbase and PHbase from the file Basave.

  Description : This procedure is for the Apple Pascal compiler.
)
var
  AdCount, AcCount : integer ;

begin
  ClearScreen ;
  writeln('Restoring the arrays of currently entered') ;
  writeln('engagement and hit base values will over-') ;
  writeln('write the information currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (Apple implementation)
      reset(Basave,'BASAVE') ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          read(Basave,PEbase[AdCount,AcCount]) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          read(Basave,PHbase[AdCount,AcCount]) ;
      close (Basave,lock) ;
      writeln ;
      writeln('Engagement and Hit threshold') ;
      writeln('data has been restored.') ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure BaseGetData)

```

```

procedure FactSaveData(var Choice : char) ;

(*
                                Main Program Choice 'U'
*)
(
  Called from : Parameter Menu 2

  Returns to   : Parameter Menu 2

  Purpose      : To save the current factor arrays Placefact,
                Tactifact and Aspectfact in the file Fasave.

  Description  : This procedure is for the Apple Pascal compiler.
)
var
  AdCount, Counter : integer ;
begin
  ClearScreen ;
  writeln('Saving the arrays of currently entered') ;
  writeln('factors will overwrite the') ;
  writeln('information currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (Apple implementation)
      rewrite(Fasave,'FASAVE') ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          write(Fasave,Placefact[AdCount,Counter]:4:2) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          write(Fasave,Tactifact[AdCount,Counter]:4:2) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          write(Fasave,Aspectfact[AdCount,Counter]:4:2) ;
      close (Fasave,lock) ;
      writeln ;
      writeln('Factor data has been saved.') ;
      WaitforReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure FactSaveData)

```

```

procedure FactGetData(var Choice : char) ;
(
  (*
    Main Program Choice 'U'
  *)
  (
    Called from : Parameter Menu 2

    Returns to : Parameter Menu 2

    Purpose      : To restore the current factor arrays Placefact,
                  Tactifact and Aspectfact from the file Fasave.

    Description  : This procedure is for the Apple Pascal compiler.
  )
)
var
  AdCount, Counter : integer ;

begin
  ClearScreen ;
  writeln('Restoring the arrays of currently entered') ;
  writeln('factors will overwrite the') ;
  writeln('information currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (*Apple implementation*)
      reset(Fasave,'FASAVE') ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          read(Fasave,Placefact[AdCount,Counter]) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          read(Fasave,Tactifact[AdCount,Counter]) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          read(Fasave,Aspectfact[AdCount,Counter]) ;
      close (Fasave,lock) ;
      writeln ;
      writeln('Factor data has been restored.') ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure FactGetData)

```

TURBO Pascal Procedures

The procedures on the following pages are designed for the TURBO Pascal compiler.

```

procedure AdSaveData(var Choice : char) ;

(*
                                Main Program Choice 'H'
*)
{
  Called from : Air Defense Menu

  Returns to   : Air Defense Menu

  Purpose      : To save the current Air Defense Unit array Adunit
                 in the file Adsave.

  Description  : This procedure is for the TURBO Pascal compiler.
}
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Saving the array of currently entered') ;
  writeln('Air Defense Units will overwrite the') ;
  writeln('information currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (TURBO implementation)
      assign (Adsave,'ADSAVE') ;
      rewrite(Adsave) ;
      for Counter := 1 to MAXADUNITS do
        with Adunit[Counter] do
          begin
            writeln(Adsave,Isactive,Kind:2,Location:2,Effectlevel:4:2) ;
          end ;
        close(Adsave) ;
        writeln ;
        writeln('Air Defense Unit data has been saved.') ;
        WaitForReturn ;
      end ;
      Choice := 'B' ; (return to the Air Defense Unit menu)
    end ; (procedure AdSaveData)

```

```

procedure AdGetData(var Choice : char) ;
(*
           Main Program Choice 'I'
*)
{
  Called from : Air Defense Menu

  Returns to  : Air Defense Menu

  Purpose     : To restore the current Air Defense Unit array Adunit
               from the file Adsave.

  Description : This procedure is for the TURBO Pascal compiler.
}
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Restoring Air Defense Unit Data') ;
  writeln('will overwrite the information') ;
  writeln('currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (TURBO implementation)
      assign(Adsave,'ADSAVE') ;
      reset (Adsave) ;
      for Counter := 1 to MAXADUNITS do
        with Adunit[Counter] do
          begin
            readln(Adsave,Isactive,Kind,Location,Effectlevel) ;
          end ;
        close(Adsave) ;
        writeln ;
        writeln('Air Defense Unit data has been restored.') ;
        WaitForReturn ;
      end ;
      Choice := 'B' ; (return to Air Defense Unit menu)
    end ; (procedure AdGetData)

```

```

procedure AcSaveData(var Choice : char) ;

(%)
                                Main Program Choice 'L'
%)
{
  Called from : Aircraft Menu

  Returns to  : Aircraft Menu

  Purpose     : To save the current Aircraft array Aircraft
                in the file Acsave.

  Description : This procedure is for the TURBO Pascal compiler.
}
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Saving the array of currently entered') ;
  writeln('Aircraft will overwrite the information') ;
  writeln('currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (TURBO implementation)
      assign (Acsave,'ACSAVE') ;
      rewrite(Acsave) ;
      for Counter := 1 to MAXAIRCRAF do
        with Aircraft[Counter] do
          begin
            writeln(Acsave,Isactive,Sort:2,Tactic:2,Aspect:2) ;
          end ;
      close (Acsave) ;
      writeln ;
      writeln('Aircraft data has been saved.') ;
      WaitForReturn ;
    end ;
  Choice := 'C' ; (return to the Aircraft menu)
end ; (procedure AcSaveData)

```

```

procedure AcGetData(var Choice : char) ;

(*
                               Main Program Choice 'M'
*)
(
  Called from : Aircraft Menu

  Returns to  : Aircraft Menu

  Purpose     : To restore the current Aircraft array Aircraft
                from the file Acsave.

  Description : This procedure is for the TURBO Pascal compiler.
)
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Restoring the Aircraft data') ;
  writeln('will overwrite the information') ;
  writeln('currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (TURBO implementation)
      assign(Acsave,'ACSAVE') ;
      reset (Acsave) ;
      for Counter := 1 to MAXAIRCRAF do
        with Aircraft[Counter] do
          begin
            readln(Acsave,Isactive,Sort,Tactic,Aspect) ;
          end ;
        close(Acsave) ;
        writeln ;
        writeln('Aircraft data has been restored.') ;
        WaitForReturn ;
      end ;
      Choice := 'C' ; (return to Aircraft menu)
    end ; (procedure AcGetData)

```

```

procedure BaseSaveData(var Choice : char) ;

(*
                                Main Program Choice 'S'
*)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To save the current base value arrays
                PEbase and PHbase in the file Basave.

  Description : This procedure is for the TURBO Pascal compiler.
)
var
  AdCount, AcCount : integer ;

begin
  ClearScreen ;
  writeln('Saving the arrays of currently entered') ;
  writeln('engagement and hit base values will over-') ;
  writeln('write the information currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (TURBO implementation)
      assign (Basave, 'BASAVE') ;
      rewrite(Basave) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          write(Basave, PEbase[AdCount, AcCount]:4:2) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          write(Basave, PHbase[AdCount, AcCount]:4:2) ;
      close (Basave) ;
      writeln ;
      writeln('Engagement and Hit threshold') ;
      writeln('data has been saved.') ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure BaseSaveData)

```

```

procedure BaseGetData(var Choice : char) ;

(*
                                Main Program Choice 'T'
*)
{
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To restore the current base value arrays
                PEbase and PHbase from the file Basave.

  Description : This procedure is for the TURBO Pascal compiler.
}
var
  AdCount, AcCount : integer ;

begin
  ClearScreen ;
  writeln('Restoring the arrays of currently entered') ;
  writeln('engagement and hit base values will over-') ;
  writeln('write the information currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (TURBO implementation)
      assign(Basave,'BASAVE') ;
      reset (Basave) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          read(Basave,PEbase[AdCount,AcCount]) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          read(Basave,PHbase[AdCount,AcCount]) ;
      close (Basave) ;
      writeln ;
      writeln('Engagement and Hit threshold') ;
      writeln('data has been restored.') ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure BaseGetData)

```

```

procedure FactSaveData(var Choice : char) ;

(*
                                Main Program Choice 'U'
*)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To save the current factor arrays Placefact,
                Tactifact and Aspectfact in the file Fasave.

  Description : This procedure is for the TURBO Pascal compiler.
)
var
  AdCount, Counter : integer ;
begin
  ClearScreen ;
  writeln('Saving the arrays of currently entered') ;
  writeln('factors will overwrite the') ;
  writeln('information currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (TURBO implementation)
      assign (Fasave,'FASAVE') ;
      rewrite(Fasave) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          write(Fasave,Placefact[AdCount,Counter]:4:2) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          write(Fasave,Tactifact[AdCount,Counter]:4:2) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          write(Fasave,Aspectfact[AdCount,Counter]:4:2) ;
      close (Fasave) ;
      writeln ;
      writeln('Factor data has been saved.') ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure FactSaveData)

```

```

procedure FactGetData(var Choice : char) ;
(
    (*
        Main Program Choice 'U'
    *)
    (
        Called from : Parameter Menu 2

        Returns to : Parameter Menu 2

        Purpose      : To restore the current factor arrays Placefact,
                       Tactifact and Aspectfact from the file Fasave.

        Description  : This procedure is for the TURBO Pascal compiler.
    )
var
    AdCount, Counter : integer ;

begin
    ClearScreen ;
    writeln('Restoring the arrays of currently entered') ;
    writeln('factors will overwrite the') ;
    writeln('information currently in use.') ;
    writeln ;
    writeln ;
    writeln('Do you wish to continue?') ;
    if Okay then
        begin
            (*TURBO implementation*)
            assign(Fasave,'FASAVE') ;
            reset (Fasave) ;
            for AdCount := 1 to MAXADKIND do
                for Counter := 1 to MAXFACTORS do
                    read(Fasave,Placefact[AdCount,Counter]) ;
            for AdCount := 1 to MAXADKIND do
                for Counter := 1 to MAXFACTORS do
                    read(Fasave,Tactifact[AdCount,Counter]) ;
            for AdCount := 1 to MAXADKIND do
                for Counter := 1 to MAXFACTORS do
                    read(Fasave,Aspectfact[AdCount,Counter]) ;
            close (Fasave) ;
            writeln ;
            writeln('Factor data has been restored.') ;
            WaitForReturn ;
        end ;
    Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure FactGetData)

```

UNIX Pascal Procedures

The procedures on the following pages are designed for the UNIX Pascal compiler.

```

procedure AdSaveData(var Choice : char) ;

(*
                                Main Program Choice 'H'
*)
(
  Called from : Air Defense Menu

  Returns to   : Air Defense Menu

  Purpose      : To save the current Air Defense Unit array Adunit
                 in the file Adsave.

  Description  : This procedure is for the UNIX Pascal compiler.
)
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Saving the array of currently entered') ;
  writeln('Air Defense Units will overwrite the') ;
  writeln('information currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (UNIX implementation)
      rewrite(Adsave) ;
      for Counter := 1 to MAXADUNITS do
        with Adunit[Counter] do
          begin
            writeln(Adsave, Isactive, Kind:2, Location:2, Effectlevel:4:2) ;
          end ;
        writeln ;
      writeln('Air Defense Unit data has been saved.') ;
      WaitForReturn ;
    end ;
  Choice := 'B' ; (return to the Air Defense Unit menu)
end ; (procedure AdSaveData)

```

```

procedure AdGetData(var Choice : char) ;

(*
                               Main Program Choice 'I'
*)
{
  Called from : Air Defense Menu

  Returns to   : Air Defense Menu

  Purpose      : To restore the current Air Defense Unit array Adunit
                 from the file Adsave.

  Description  : This procedure is for the UNIX Pascal compiler.
}
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Restoring Air Defense Unit Data') ;
  writeln('will overwrite the information') ;
  writeln('currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (UNIX implementation)
      reset (Adsave) ;
      for Counter := 1 to MAXADUNITS do
        with Adunit[Counter] do
          begin
            readln(Adsave,Isactive,Kind,Location,Effectlevel) ;
            end ;
          writeln ;
          writeln('Air Defense Unit data has been restored.') ;
          WaitForReturn ;
        end ;
      Choice := 'B' ; (return to Air Defense Unit menu)
    end ; (procedure AdGetData)

```

```

procedure AcSaveData(var Choice : char) ;

(*
                                Main Program Choice 'L'
*)
(
  Called from : Aircraft Menu

  Returns to  : Aircraft Menu

  Purpose     : To save the current Aircraft array Aircraft
                in the file Acsave.

  Description : This procedure is for the UNIX Pascal compiler.
)
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Saving the array of currently entered') ;
  writeln('Aircraft will overwrite the information') ;
  writeln('currently saved.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (UNIX implementation)
      rewrite(Acsave) ;
      for Counter := 1 to MAXAIRCRAF do
        with Aircraft[Counter] do
          begin
            writeln(Acsave,Isactive,Sort:2,Tactic:2,Aspect:2) ;
          end ;
        writeln ;
        writeln('Aircraft data has been saved.') ;
        WaitforReturn ;
      end ;
      Choice := 'C' ; (return to the Aircraft menu)
    end ; (procedure AcSaveData)
end ;

```

```

procedure AcGetData(var Choice : char) ;

(*
                                Main Program Choice 'M'
*)
(
  Called from : Aircraft Menu

  Returns to  : Aircraft Menu

  Purpose     : To restore the current Aircraft array Aircraft
                from the file Acsave.

  Description : This procedure is for the UNIX Pascal compiler.
)
var
  Counter : integer ;

begin
  ClearScreen ;
  writeln('Restoring the Aircraft data') ;
  writeln('will overwrite the information') ;
  writeln('currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (UNIX implementation)
      reset (Acsave) ;
      for Counter := 1 to MAXAIRCRAF do
        with Aircraft[Counter] do
          begin
            readln(Acsave,Isactive,Sort,Tactic,Aspect) ;
            end ;
            writeln ;
            writeln('Aircraft data has been restored.') ;
            WaitForReturn ;
          end ;
          Choice := 'C' ; (return to Aircraft menu)
        end ; (procedure AcGetData)

```

```

procedure BaseSaveData(var Choice : char) ;

(%)
                                Main Program Choice 'S'
%)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To save the current base value arrays
                PEbase and PHbase in the file Basave.

  Description : This procedure is for the UNIX Pascal compiler.

)
var
  AdCount, AcCount : integer ;

begin
  ClearScreen ;
  writeln('Saving the arrays of currently entered' ) ;
  writeln('engagement and hit base values will over-' ) ;
  writeln('write the information currently saved.' ) ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (UNIX implementation)
      rewrite(Basave) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          write(Basave,PEbase[AdCount,AcCount]:4:2) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          write(Basave,PHbase[AdCount,AcCount]:4:2) ;
      writeln ;
      writeln('Engagement and Hit threshold' ) ;
      writeln('data has been saved.' ) ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure BaseSaveData)

```

```

procedure BaseGetData(var Choice : char) ;

(*
                                Main Program Choice 'T'
*)
(
  Called from : Parameter Menu 2

  Returns to  : Parameter Menu 2

  Purpose     : To restore the current base value arrays
                PEbase and PHbase from the file Basave.

  Description : This procedure is for the UNIX Pascal compiler.
)
var
  AdCount, AcCount : integer ;

begin
  ClearScreen ;
  writeln('Restoring the arrays of currently entered') ;
  writeln('engagement and hit base values will over-') ;
  writeln('write the information currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (UNIX implementation)
      reset (Basave) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          read(Basave,PEbase[AdCount,AcCount]) ;
      for AdCount := 1 to MAXADKIND do
        for AcCount := 1 to MAXACSORT do
          read(Basave,PHbase[AdCount,AcCount]) ;
      writeln ;
      writeln('Engagement and Hit threshold') ;
      writeln('data has been restored.') ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure BaseGetData)

```

```

procedure FactSaveData(var Choice : char) ;
(X
    Main Program Choice 'U'
X)
(
    Called from : Parameter Menu 2

    Returns to  : Parameter Menu 2

    Purpose     : To save the current factor arrays Placefact,
                  Tactifact and Aspectfact in the file Fasave.

    Description : This procedure is for the UNIX Pascal compiler.
)
var
    AdCount, Counter : integer ;
begin
    ClearScreen ;
    writeln('Saving the arrays of currently entered') ;
    writeln('factors will overwrite the') ;
    writeln('information currently saved.') ;
    writeln ;
    writeln ;
    writeln('Do you wish to continue?') ;
    if Okay then
        begin
            (UNIX implementation)
            rewrite(Fasave) ;
            for AdCount := 1 to MAXADKIND do
                for Counter := 1 to MAXFACTORS do
                    write(Fasave,Placefact[AdCount,Counter]:4:2) ;
            for AdCount := 1 to MAXADKIND do
                for Counter := 1 to MAXFACTORS do
                    write(Fasave,Tactifact[AdCount,Counter]:4:2) ;
            for AdCount := 1 to MAXADKIND do
                for Counter := 1 to MAXFACTORS do
                    write(Fasave,Aspectfact[AdCount,Counter]:4:2) ;
            writeln ;
            writeln('Factor data has been saved.') ;
            WaitForReturn ;
        end ;
    Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure FactSaveData)

```

```

procedure FactGetData(var Choice : char) ;
(X
                                Main Program Choice 'V'
X)
(
  Called from : Parameter Menu 2

  Returns to   : Parameter Menu 2

  Purpose      : To restore the current factor arrays Placefact,
                Tactifact and Aspectfact from the file Fasave.

  Description  : This procedure is for the UNIX Pascal compiler.
)
var
  AdCount, Counter : integer ;

begin
  ClearScreen ;
  writeln('Restoring the arrays of currently entered') ;
  writeln('factors will overwrite the') ;
  writeln('information currently in use.') ;
  writeln ;
  writeln ;
  writeln('Do you wish to continue?') ;
  if Okay then
    begin
      (UNIX implementation)
      reset (Fasave) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          read(Fasave,Placefact[AdCount,Counter]) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          read(Fasave,Tactifact[AdCount,Counter]) ;
      for AdCount := 1 to MAXADKIND do
        for Counter := 1 to MAXFACTORS do
          read(Fasave,Aspectfact[AdCount,Counter]) ;
      writeln ;
      writeln('Factor data has been restored.') ;
      WaitForReturn ;
    end ;
  Choice := '0' ; (return to Parameter Menu 2)
end ; (procedure FactGetData)

```

APPENDIX F: Random Number Generator Tests

Random Number Generator Tests

A simple function implements a linear congruential pseudo-random number generator. Each random number generated serves as the basis for the next. The linear congruential generator is of the form

$$I[i+1] = (a \times I[i] + 1) \text{ mod MAXINT} \quad [3]$$

where

$$a = k \times 4 + 1 \quad k = 0,1,2\dots \quad [4]$$

and MAXINT is implementation dependent. Certain values of a provide better sequences of random numbers for micro-computers. The following Pascal code used in the program returns random integers in the range 1 to 100:

```
function Uniform : integer ;
(
  Requires the external global variable Seed
  MULT may be 1221,1287,3993,4189,4293,
  9237,14789,15125 or 17245
)

const
  MULT = 3993 ;

begin
  Seed := MULT * Seed + 1 ;
  if Seed < 0 then Seed := Seed + maxint + 1 ;
  Uniform := trunc((Seed/maxint)*100) + 1 ;
end ; (function Uniform)
```

This emulates the uniform distribution of numbers resulting from the roll of a die.

Tests

Periodicity. The period is 32763 numbers generated before a repetitive sequence is encountered on the development system. This was determined in a separate program that placed the first five random numbers in an array and then consecutively compared each string of five numbers generated with the first five. Strings of five numbers were made by entering a generated number at one end of the comparison array and removing a number from the other end of the array, so that in the course of five comparisons with the first five numbers, a single number occupied each of the five positions in the test array. This was done with several different random number seeds. In each case 32763 random numbers were generated before a repetitive sequence of five numbers in a row was found.

Chi-Squared Test. A simplified Chi-Squared test was run in a separate program. One thousand random numbers were generated in the interval 1 to 100. The occurrence of each number in the interval was noted for five different seeds. The results for each seed and the Chi-Squared values are in Tables F-1 thru F-5. These Chi-Squared values have been divided by the number of 'observations', 100, and are acceptable values for the frequency expected for a uniform distribution. [Whitney, 1984:446] At a 0.05 level of significance, the hypothesis that there is no difference between the generated distribution and a uniform distribution is not rejected.

Runs Tests. Tests were made in separate programs for runs up and down and runs above and below the mean value of 50. [Banks and Carson, 1984:271-278] These tests examine the hypothesis that the numbers returned are independent of one another. A run is defined as a sequence of similar events bracketed by dissimilar events. Runs tests are concerned with both the number and length of runs. For runs up and down, the two types of runs are sequences of numbers in which each of the numbers is followed by a larger number, an up run, or a smaller number, a down run. A "+" or "-" is assigned each number generated depending on whether the following number is larger or smaller. For N numbers, the maximum number of runs is N-1, and the minimum is a single run. The mean, M, of A runs in a sequence of N numbers is

$$\frac{2 \times N - 1}{3} \quad [F1]$$

while the variance is

$$\frac{16 \times N - 29}{90} \quad [F2]$$

The Z statistic is calculated as (A - M)/S, where S is the standard deviation and Z is normally distributed on the interval (0,1). The results of the runs in 100 numbers are in Table F-6. The Z value calculated is 0.786, so failure to reject the hypothesis of independence occurs at a 0.05 level of significance.

For runs above and below the mean of 50, a "+" is assigned to each number greater than 50, and a "-" is assigned to each number less than or equal to the mean. For this test, n1 is the number of observations above the mean, n2 is the number of observations below the mean, B is the total number of runs, and N is the total number of observations. The mean, M, for the Z statistic is

$$\frac{1/2 + (2 \times n1 \times n2)}{N} \quad [F3]$$

and the variance is

$$\frac{2 \times n1 \times n2 \times (2 \times n1 \times n2 - N)}{N \times N \times (N - 1)} \quad [F4]$$

The Z statistic is calculated as (B - M)/S, where S is the standard deviation. The results of the runs in 100 numbers are in Table F-7. The Z value calculated is -0.037, so failure to reject the hypothesis of independence occurs at a 0.05 level of significance.

Conclusion

The period for the random number generator used is sufficient for the program since over 6000 engagements could be made without repeating a sequence of random numbers. The Chi-Squared test fails to reject the hypothesis that the numbers returned are uniformly distributed. The two different runs tests fail to reject the hypothesis that the numbers returned are independent. The conclusion may be drawn that the random number generator is suitable for the purposes of the program on the development system.

Seed: 1

13	13	7	9	8	13	9	7	12	10
7	8	6	12	12	13	15	14	10	13
11	11	10	10	8	11	10	11	12	15
16	10	8	8	14	14	8	13	15	6
10	13	9	13	13	12	14	6	9	5
9	9	9	7	9	10	10	9	10	9
8	9	9	13	11	4	10	8	4	7
10	12	12	13	7	11	6	9	10	9
9	12	8	10	7	10	6	13	7	10
9	10	9	7	6	13	11	7	13	14

Chi-Squared(1) = 0.692

Table F-1

Seed: 12

14	14	14	4	6	13	10	10	8	7
14	7	10	5	11	8	11	15	10	15
13	15	14	12	8	10	7	17	7	5
9	9	12	10	7	12	10	9	8	4
9	9	9	7	12	20	5	10	6	13
11	8	13	9	13	15	9	12	12	8
6	7	6	14	10	10	9	16	12	15
4	7	14	6	7	7	10	14	13	9
6	8	8	6	13	13	10	12	11	14
9	8	6	12	6	11	7	12	11	7

Chi-Squared(2) = 1.054

Table F-2

Seed: 123

7	9	18	10	7	8	9	14	8	12
6	11	8	8	13	10	7	14	14	15
9	6	8	12	9	7	11	8	13	4
7	6	10	10	9	10	8	8	16	11
9	12	11	11	16	11	12	6	9	17
7	10	6	10	14	12	12	13	7	10
13	8	13	6	11	14	8	7	8	12
7	4	12	9	15	7	8	9	4	15
8	10	6	10	10	10	8	10	11	13
10	16	11	8	10	7	11	12	14	10

Chi-Squared(3) = 0.880

Table F-3

Seed: 1234

8	11	7	13	11	2	10	14	7	10
16	13	13	7	6	20	10	7	5	7
10	13	14	10	6	11	10	6	11	17
16	10	14	5	9	12	12	11	12	9
10	11	7	9	7	12	11	7	11	11
5	12	8	9	6	13	8	9	11	7
7	25	8	13	9	9	7	12	10	13
7	14	8	7	9	11	8	11	14	9
3	12	12	8	13	6	9	9	6	15
8	10	12	10	9	10	11	7	8	12

Chi-Squared(4) = 1.134

Table F-4

Seed: 12345

7	12	7	9	8	9	8	11	11	6
10	13	11	14	9	13	6	8	12	7
7	7	4	13	22	13	10	7	8	12
12	9	14	11	10	5	8	7	10	7
11	8	8	9	11	13	8	10	7	6
7	20	5	12	11	10	14	9	15	9
10	8	14	9	10	5	15	5	11	10
13	9	19	8	13	14	9	8	14	5
5	6	7	13	14	10	12	7	8	7
9	18	13	12	10	8	7	10	12	13

Chi-Squared(5) = 1.120

Table F-5

RUNS UP/DOWN									
-	+	-	+	+	-	+	-	-	+
+	-	+	+	-	+	+	-	+	-
-	+	-	+	-	-	+	-	+	+
-	+	+	-	+	+	-	+	+	-
-	+	+	+	-	+	+	-	+	-
+	-	-	+	+	-	+	-	+	-
+	-	+	-	-	+	-	+	-	+
-	+	-	-	-	+	-	+	-	+
+	-	-	-	+	-	+	+	-	+
+	-	+	-	-	-	+	+	-	-

Table F-6

RUNS OVER/UNDER									
-	-	-	-	+	+	-	+	-	-
+	+	-	-	+	-	+	+	-	+
-	-	+	-	+	+	-	-	-	-
+	-	+	+	+	+	+	+	+	+
-	-	-	-	+	+	+	+	-	+
+	+	+	-	+	+	-	+	-	+
-	+	-	+	+	-	-	-	+	-
+	+	+	+	+	-	-	-	+	-
+	+	+	+	-	-	-	-	-	-
-	+	-	+	+	-	-	+	+	-

Table F-7

APPENDIX G: Default Base and Factor Values

Table	Page
G-1: Engagement Base Values	G-2
G-2: Hit Base Values	G-2
G-3: Location Factors	G-2
G-4: Tactic Factors	G-3
G-5: Aspect Factors	G-3
G-6: DUNN-KEMPF	G-4

Probability of Engagement Base Values

	Helicopter	Transport	Fighter
Vulcan	0.83	0.67	0.50
SGT York	0.90	0.70	0.60
Redeye	0.50	0.67	0.50
Stinger	0.60	0.70	0.60
Chaparral	0.50	0.67	0.50

Table G-1

Probability of Hit Base Values

	Helicopter	Transport	Fighter
Vulcan	0.50	0.50	0.33
SGT York	0.60	0.60	0.40
Redeye	0.17	0.33	0.33
Stinger	0.20	0.40	0.40
Chaparral	0.33	0.33	0.33

Table G-2

Location Factors

	Europe	Desert	Jungle
Vulcan	1.00	1.10	0.80
SGT York	1.00	1.10	0.80
Redeye	1.00	1.10	0.80
Stinger	1.00	1.10	0.80
Chaparral	1.00	1.10	0.80

Table G-3

Tactics Factors

	Pop-Up	Lay-Down	Fly-Over
Vulcan	0.90	1.00	1.10
SGT York	0.90	1.00	1.10
Redeye	0.90	1.00	1.10
Stinger	0.90	1.00	1.10
Chaparral	0.90	1.00	1.10

Table G-4

Aspect Factors

	Head-On	Crossing	Tail
Vulcan	0.90	1.10	0.90
SGT York	0.90	1.10	0.90
Redeye	0.90	1.10	0.90
Stinger	0.90	1.10	0.90
Chaparral	0.90	1.10	0.90

Table G-5

DUNKEMPF TABLES

ENGAGEMENT

<u>Die Roll</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>
<u>Vulcan vs</u>						
Helicopter	0	E	E	E	E	E
Fighter	0	0	0	E	E	E
<u>Chaparral vs</u>						
Fighter	0	0	0	E	E	E
<u>Redeye vs</u>						
Helicopter	0	0	0	E	E	E
Fighter	0	0	0	E	E	E

0 - no effect
E - engaged, consult effect

EFFECTS

<u>Die Roll</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>
<u>Vulcan vs</u>						
Helicopter	0	0	0	H	H	H
Fighter	0	0	0	0	H	H
<u>Chaparral vs</u>						
Fighter	0	0	0	0	H	H
<u>Redeye vs</u>						
Helicopter	0	0	0	0	0	E
Fighter	0	0	0	0	0	E

0 - no effect
H - hit, consult results

RESULTS

<u>Die Roll</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>
<u>Vulcan vs</u>						
Helicopter	S	S	D	D	D	D
Fighter	S	S	S	S	D	D
<u>Chaparral vs</u>						
Fighter	S	S	S	D	D	D
<u>Redeye vs</u>						
Helicopter	S	S	S	D	D	D
Fighter	S	S	S	S	D	D

S - suppressed
D - destroyed

Table G-6

BIBLIOGRAPHY

Addyman, Anthony M. "Pascal," Programming Language Standardization, edited by I.D. Hill. New York: Halsted Press, 1980.

Banks, Jerry and John S. Carson, II. Discrete-Event System Simulation. Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

Baron, D.W. "A Perspective on Pascal," Pascal - The Language and its Implementation, edited by D.W. Baron. New York: Wiley & Sons, 1981.

Battilega, John A. and Judith K. Grange, editors. The Military Applications of Modeling. Wright-Patterson AFB, Ohio: AFIT Press, 1983.

Blum, Ronald. "A Measure of Effectiveness for a Barrier Type of Surface to Air Defense," Operations Research, 9 (3): 48-56 (June 1963).

Bolté, David L. Classnotes for Air Defense Artillery Basic and Advanced Officer Courses, 1977, 1982.

Borland International. TURBO Pascal Reference Manual. 1983.

Brodheim, Eric et al. "A General Dynamics Model for Air Defense," Operations Research, 15 (5): 779-796 (October 1967).

Cameron, F.W.P. et al. STC Support for the AC/225 (Panel V) Study on Future SHORAD Systems. The Hague, Netherlands: SHAPE Technical Centre, 1980.

Dale, Nell and David Orshalick. Introduction to Pascal and Structured Design. Lexington, Mass: D.C. Heath and Company, 1983.

Department of the Army. U.S. Army Air Defense Artillery Employment. FM 44-1. Washington: Government Printing Office, 1981.

Department of the Army. U.S. Army Air Defense Artillery Operations. FM 44-1-1. Washington: Government Printing Office, 1969.

Department of the Army. U.S. Army Air Defense Artillery Employment, Chaparral/Vulcan. FM 44-3. Washington: Government Printing Office, 1979.

Department of the Army. Operations and Training, Chaparral.
FM 44-4. Washington: Government Printing Office, 1979.

Department of the Army. Operations and Training, Vulcan.
FM 44-5. Washington: Government Printing Office, 1978.

Department of the Army. Stinger Team Operations.
FM 44-18-1. Washington: Government Printing Office, 1980.

Department of the Army. U.S. Army Air Defense Artillery
Employment, Redeye. FM 44-3. Washington: Government Printing
Office, 1977.

Department of the Army. Operations and Training, Redeye.
FM 44-23-1. Washington: Government Printing Office, 1977.

Dresher, Melvin. "Mathematical Models of Conflict,"
Systems Analysis and Policy Planning. New York: Elsevier, 1968.

Dunn, Steve and Hilton Kempf. DUNN-KEMPF, Battle Guide to
Simulation. Fort Leavenworth, Kansas: Combined Arms
Training Activity, 1976.

Friedman, Yoran. "Optimal Strategy for One Against Many,"
Operations Research, 25 (6): 884-888 (July 1979).

Gilb, Tom. "Spare Parts Maintenance Strategy for Programs,"
Techniques of Program and System Maintenance, edited by
Girish Parikh. Cambridge, Massachusetts: Winthrop
Publishers, 1982.

Gilb, Tom. "Structured Program Coding," Techniques of Program
and System Maintenance, edited by Girish Parikh.
Cambridge, Massachusetts: Winthrop Publishers, 1982.

Hays, R.D. and D.G. Linton. Air Defense Engagement Models.
Huntsville, Alabama: Military Systems Division, 1974.

Jensen, Kathleen and Niklaus Wirth. Pascal User Manual
and Report. Berlin: Springer-Verlag, 1974.

Joy, William N. et al. Berkeley Pascal User's Manual.
Berkeley, California: University of California, 1980.

Lecarme, O. "Pascal and Portability," Pascal - The Language
and its Implementation, edited by D.W. Baron. New York:
Wiley & Sons, 1981.

Leibowitz, Martin L. and Gerald J. Lieberman. "Optimal
Composition and Deployment of Heterogeneous Local Air Defense
Systems," Operations Research, 8 (3): 324-337 (June 1960).

Mangulis, Visvaldis. "Lanchester Equations for Multiple Killing Hits," Operations Research, 28 (3): 568-569 (March 1980).

Ordway, Girard L. and Henry M. Rosenstock. "Analytic Expressions for the Probability of Penetrating a Point Defense," Operations Research, 11 (1): 48-56 (January 1963).

Paris, W. et al. Evaluation of Air Defense Effectiveness Model. Aberdeen Proving Grounds, Maryland: Army Materiel Systems Analysis Agency, 1974.

Rubenstein, Richard and Harry Hersh. The Human Factor. Digital Press, 1984.

Semmens, Paul et al. Division Air Defense Engagement Simulation Fort Bliss, Texas: Director of Combat Developments, 1977.

Specht, R.D. "The Nature of Models," Systems Analysis and Policy Planning. New York: Elsevier, 1968.

Thesen, Arne and Tzyh-Jong Wang. Some Efficient Random Number Generators for Micro-Computers. Madison, Wisconsin: University of Wisconsin, 1983.

Weiner, M.G. "Gaming," Systems Analysis and Policy Planning. New York: Elsevier, 1968.

Wilhelm, H.R. Simulation of Air Defense Operations and Multiple Air Combat. The Hague, Netherlands: SHAPE Technical Centre, 1979.

Young, Albert F. and Gerald Solomon. Surface-to-Air Gun Effectiveness Model. Eglin Air Force Base, Florida: Air Force Armament Laboratory, 1977.

VITA

Captain David L. Bolté was born on 13 October 1954 in Heidleberg, Germany. He graduated from high school in Carlisle, Pennsylvania, in 1972, and attended the United States Military Academy at West Point, New York, from which he received the degree of Bachelor of Science. He was commissioned in 1977 in the U.S. Army with a basic branch of Air Defense Artillery. He was assigned to the 82d Airborne Division's Vulcan/Redeye battalion from 1978 to 1980, serving as Vulcan platoon leader, Redeye Section leader, radar platoon leader, and Vulcan battery executive officer. He was assigned to the 2d Infantry Division's Vulcan/Chaparral battalion in Korea as Headquarters Battery executive officer until assuming command of Battery A, 2d Battalion (HAWK), 71st Air Defense Artillery, 38th ADA Brigade, Korea in 1981. He was assigned to the Air Defense Artillery School and Center from 1982 to 1983 when he entered the School of Engineering, Air Force Institute of Technology in August of 1983.

Permanent Address: 10715 Oakenshaw Court
Burke, VA 22015

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS			
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release Distribution unlimited			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GST/MATH/85M-1		5. MONITORING ORGANIZATION REPORT NUMBER(S)			
6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL (If applicable) AFIT/ENC	7a. NAME OF MONITORING ORGANIZATION			
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433		7b. ADDRESS (City, State and ZIP Code)			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER			
8c. ADDRESS (City, State and ZIP Code)		10. SOURCE OF FUNDING NOS.			
11. TITLE (Include Security Classification) see Box 19		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
		12. PERSONAL AUTHOR(S) David L. Bolte, B.S. CPT, U.S. Army			
13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM TO	14. DATE OF REPORT (Yr., Mo., Day) 1985 March	15. PAGE COUNT 250		
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB. GR.	Air Defense Artillery Micro-computer Models		
15	03				
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
Title: A Micro-computer Model for Army Air Defense Training Thesis Chairperson: Patricia K. Lawlis, Captain, USAF					
Approved for public release: IAW AFR 190-17 <i>John W. Wolaver</i> LYNN E. WOLAVER Dean for Research and Professional Development Air Force Institute of Technology (ATC) Wright-Patterson AFB OH 45433					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Patricia K. Lawlis, Captain, USAF		22b. TELEPHONE NUMBER (Include Area Code) 513-255-2915		22c. OFFICE SYMBOL AFIT/ENC	

This report details the development of the Air Defense War Game Support Program. The Air Defense War Game Support Program provides one-on-one probabilities of engagement and kill for a variety of forward-area air defense systems against different types of aircraft. Each air defense system is described by type (Vulcan, SGT York, Redeye, Stinger, or Chaparral), location (Europe, desert or jungle) and effectiveness level (0.0 to 1.0). Each aircraft is described by type (helicopter, transport or fighter), tactic (pop-up, lay-down or fly-over), and profile or aspect presented to the air defense system (head-on, crossing, or tail). The results of an interaction between an air defense unit and an aircraft are determined by comparing a program generated random number with threshold values calculated from data tables. The displayed results of an engagement include whether the aircraft is engaged, whether the aircraft is hit and destroyed or mission suppressed, and whether the air defense system suffers any attrition from an aircraft that was engaged but not destroyed or suppressed.

Within the program the factors used in the calculation of the engagement and kill thresholds may be changed. Additionally, the probabilities of air defense system attrition and aircraft suppression, the attrition factor and the random number seed may be changed.

The program is designed specifically to be used by personnel without a computer background, yet has a structured programming style that permits easy modification by a programmer. The Air Defense War Game Support Program is written in standard Pascal and can run on a standard micro-computer with 64 thousand (64K) bytes of memory.

END

FILMED

7-85

DTIC