DTIC FILE COPY

(2)

CAORA/TR-3/8

JULY 1987

ACN 85358

AD-A199 942

DIVISION MAP EXERCISE (DIME 4.0) VOLUME II DIME DOCUMENTATION/MODEL

Technical Report CAORA/TR-3/84

UNITED STATES ARMY COMBINED ARMS CENTER

COMBINED ARMS
OPERATIONS RESEARCH ACTIVITY
FORT LEAVENWORTH, KS 66027

DISTRIBUTION UNLIMITED: APPROVED FOR PUBLIC RELEASE

87-3324

Technical Report CAORA/TR-3/84
May, 1987

ACN 85358

DIVISION
MAP EXERCISE
(DIME 4.0)

VOLUME II

DIME DOCUMENTATION/MODEL

TECHNICAL REPORT CAORA/TR-3/84

Studies and Analysis Directorate
Combined Arms Operations Research Activity
US Army Combined Arms Center
Fort Leavenworth, KS 66027-5230

Approved by:

RONALD G. MAGEE
Director, Scientific and
Technical Support Directorate

REED E. DAVIS, JR.
Deputy, CAORA

DAVID M. MADDOX
Brigadier General, USA
Commander, CAORA

NOTICES

DESTRUCTION NOTICE

When this report is no longer needed, Department of the Army organizations will destroy it in accordance with the procedures given in AR 380-5. Department of Defense contractors will destroy the report according to the requirements of Section 14 of the Industrial Security Manual for Safeguarding Classified Information. All others will return the report to Headquarters, United States Army Training and Doctrine Command.

DISTRIBUTION STATEMENT

In addition to security regulations applicable to this document, each transmittal outside the Department of Defense must have prior approval of the United States Army Training and Doctrine Command. The DIME combat model documentation consists of three volumes. This volume is Volume II, DIME Documentation/Model. The other volumes are Volume I, Game Protocol, and Volume III, Classified Data Base and Data Description. A copy of the DIME model may be obtained by forwarding requests to:

Commander
US Army Combined Arms Operations Research Activity
ATTN: ATOR-CAT-AP (Strong)
Fort Leavenworth, Kansas 66027-5230

DISCLAIMERS

The findings and recommendations of this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

ACKNOWLEDGEMENTS

THE DIvision Map Exercise (DIME) is the fourth enhancement of a battle simulation originally designed to meet the requirements for a quick-running analytical war game. DIME's continued success has resulted from the efforts of many people. In particular, the authors wish to thank MAJ Timothy J. Reischl and Stephan Arrington for their tireless efforts. While at CAORA, Tim conceived the original version of DIME, carrying it through two upgrades and finally laying out most of the model structure for the present version. Stephan Arrington developed the air defense code. The authors would also like to thank Mr. William Ralph of CAORA who edited all three volumes.

Volume I	Volume II	Volume III
MAJ (P) Al Resnick	Mr. Kent Pickett	Mrs. Wilma Strong
Mr. Jim Schneider	MAJ Jim Lunn	Mr. Terry Brashley
Mrs. Melinda Saunders	Mrs. Cindy Dean	
	Ms. Lori Carr	

CONTENTS

CONTENTS (CONTINUED)

CONTENTS (CONTINUED)

CONTENTS (CONCLUDED)

LIST OF FIGURES

LIST OF FIGURES
(CONCLUDED)

LIST OF TABLES

LIST OF TABLES

ABSTRACT

The DIvision Map Exercise (DIME) is a comprehensive, computer-assisted war game designed to portray the significant aspects of Air-Land Battle operational doctrine for the Army's primary division strike force structures. It was developed in response to a need for a quick-running low resolution battle simulation with which to address critical analytical combat development questions. Using a map board, a set of computer algorithms and manual rules, DIME portrays the important aspects of the modern battlefield which must be considered in the context of the Air-Land Battle. These include ground combat, air operations and air defense, maneuver, command and control, chemical effects and logistics.

The DIME combat model documentation consists of three volumes. This volume is Volume II, DIME Documentation/Model. The other volumes are Volume I, Game Protocol, and Volume III, Classified Data Base and Data Description.

CHAPTER 1

MODEL OVERVIEW

1. BACKGROUND AND PURPOSE.

A. One of the roles of the Combined Arms Operations Research Activity (CAORA) in Army analysis is to determine the effectiveness of tactical and doctrinal innovations for corps and division-level forces. In performing this role, CAORA is often required to develop division-level combat simulations or war games to be used as study tools. In December 1982, the Deputy Undersecretary of the Army for Operations Research (DUSA/OR) office requested that CAORA build a division-level war game to be used in the effectiveness evaluation of the High Technology Light Division (HTLD). The game was to be used in a study comparing HTLD and a conventional light infantry division.

The design criteria for the war game were:

(1) The model must fairly represent the ability of HTLD to execute the innovative tactics and maneuverability associated with the division's advanced equipment and organizational structure.

(2) The game should be "transportable" so it can be moved to Ft. Lewis for play by HTLD personnel.

(3) The game should play six hours of division combat to include resupply within one 8-hour working day.

The Division Map Exercise (DIME) was the result of an eight-month software development effort by CAORA. It was completed in October 1983 and used as CAORA's principal wargaming tool to evaluate the High Technology Light Division.

B. This document is a programmer's manual for DIME. It contains a general description of the play of the war game (a complete description can be found in the DIME Volume I, Game Protocol) and a list of the hardware necessary to run the game. This manual also contains the methodology used in the combat simulation and software documentation to include logic/data flows, variable descriptions and program listings.

C. This volume is organized into 11 chapters. This chapter contains a description of the overall model structure, the unit status file ("UNITFILE"), and the computer hardware necessary to operate the model. Chapters 2 through 11 contain documentation for the programs forming the DIME software system.

2. DIME OVERVIEW.

A. The Division Map Exercise (DIME) is a computer-assisted map exercise representing forces of up to a Blue Division engaging a Red Army. The location, movement, and deployment of the forces is represented by unit symbols placed on a 1:50,000 map. The model is ideally structured for units of Blue Battalions and Red Regiments. However, units to the resolution of brigade command posts and brigade fuel/ammunition (POL/AMMO) dumps can also be accommodated in the model. Red and Blue gamers are required to plan the distribution of ammunition and POL to all units for a six-hour period. They also maneuver their forces and structure the battles initiated during the six-hour period. The DIME model uses a set of computerized attrition and detection algorithms to determine elements surviving unit engagements. Likewise, resupply algorithms are used to maintain the current levels of ammunition and POL available to the units.

B. The structure of the model is shown in Figure 1-1. The DIME model consists of a set of BASIC software programs representing each functional aspect of the division battle. These programs operate independently, interacting only through a common unit status file containing one record for each unit. The programs are also supported by random access files containing weapon effects data (e.g., probability of kill, movement rates, etc.). The DIME programs are accessed from a menu-driven executive controller program shown in Figure 1-4 of this chapter. The normal play of six hours of division combat requires use of the programs in the following order:

(1) The game initialization program (P1). At the beginning of each six-hour gaming period, the user may update any of the records on the unit status file ("UNITFILE"). It is often necessary to change a unit's mission, resupply its ammunition and POL, and cross-attach or add combat elements to the unit. This is done with the menu-driven game initialization program (P1). If it is necessary to resupply a unit, this program must be run before the logistic support program (P2).

(2) The logistic support program (P2). This program also interacts with the "UNITFILE" to disperse the ammunition and POL available for use by each unit during the six-hour gaming period. The quantities resupplied to each unit (specified during the running of P1) are moved into an "available for consumption" status (see entries 131 through 133 on the "UNITFILE"). The logistics support program (P2) must be run after P1 and prior to any other DIME programs. If P2 is not run, available ammunition and POL will not be placed in the proper "UNITFILE" entries and the artillery, air defense, and direct fire systems will not fire in the DIME combat programs (P3 and P4).

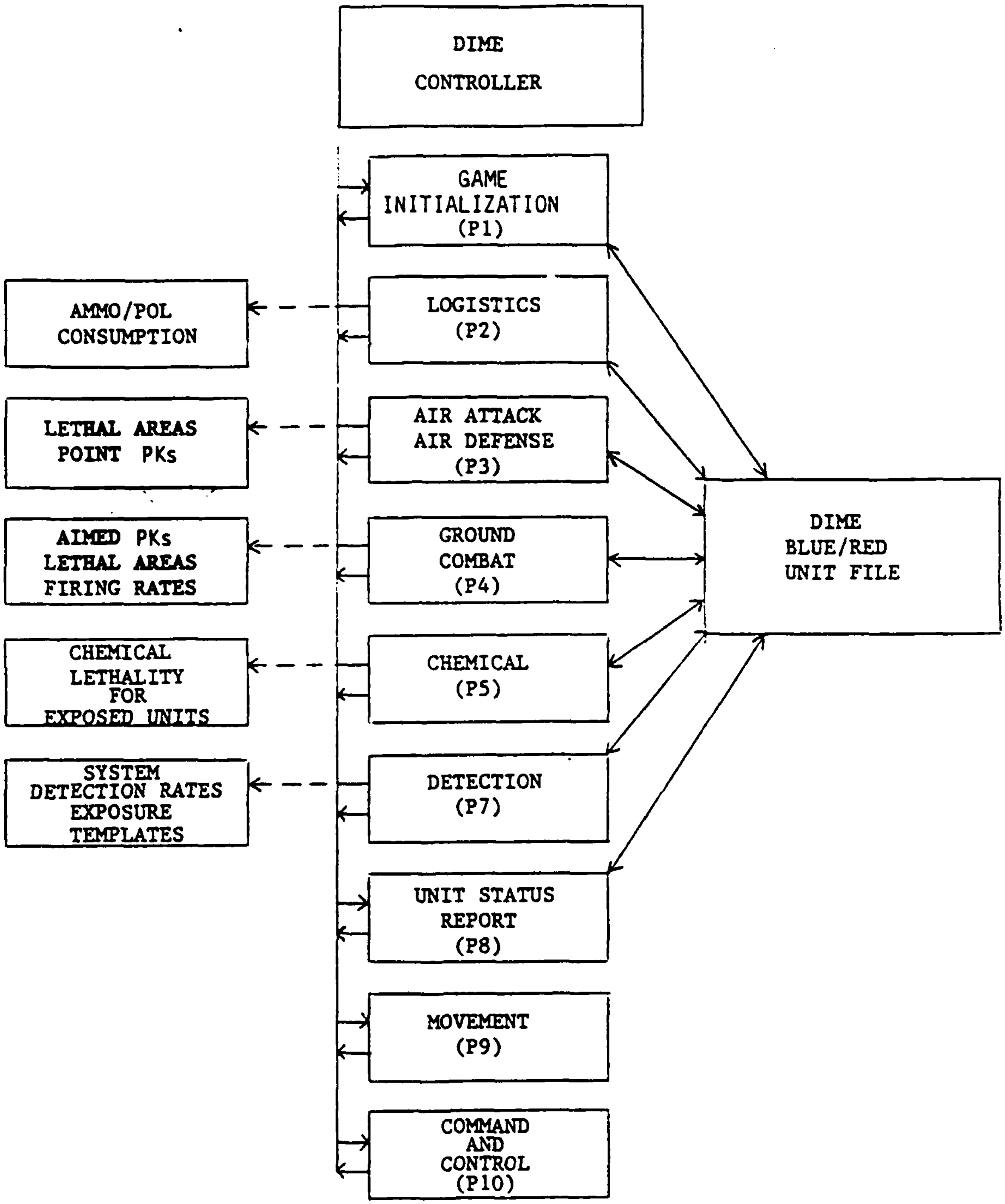


Figure 1-1. DIME program structure.

(3) The air attack/air defense program (P3). This program calculates the losses to both air and ground elements for fixed-wing air attacks on the DIME units. The program also provides losses to both helicopters and fixed-wing aircraft when air defense (AD) units are inadvertently overflown during ingress/egress to a target. The program must be run once for each air/ground attack that occurs during the six- hour period. These runs should occur before ground combat sectors are played. The model outputs element losses in both hard copy for game use and updates the DIME "UNITFILE". The model also updates air defense ammunition consumption on the unit status file.

(4) The ground combat program (P4). This program calculates losses to helicopters and ground elements resulting from ground combat engagements during the six-hour gaming period. The program requires as input those units engaged in combat plus a set of parameters describing Red and Blue command decisions influencing the battle; e.g., opening range for artillery and direct fire systems and break ranges for potential overrun situations. The ground combat module must be executed for each sector battle that occurs during the six-hour gaming period. The model outputs a hard copy of each battle history containing half-hour updates on force movements, helicopter attack status, and indirect fire tonnage consumptions. A killer/victim scoreboard is output at the end of the battle, summarizing losses to each force. The module also updates the "UNITFILE" with element losses and ammunition consumption for the engaging units.

(5) The chemical combat program (P5). The chemical program calculates losses to exposed forces attacked with artillery-delivered chemical munitions. The losses represent those incurred during the first 15 seconds of chemical attack while the unit is moving into a mission-oriented protective posture (MOPP) status. The program posts losses to the "UNITFILE" and updates the MOPP status of the unit. The chemical program is included in this manual for the sake of completeness. At the writing of this documentation, it has not been debugged nor used in a CAORA study. Anyone desiring to use this program is encouraged to contact one of the authors prior to its use.

(6) The detection program (P7). The detection program provides a list of detected units to both Red and Blue commanders. The list actually represents the commander's intelligence map and shows units on the opposing force as either:

(a) not detected,

(b) detected but not identified as to mission and composition,

(c) identified mission and composition, or

(d) lost, previously detected, but friendly sensors unable to track.

Although DIME is played as an open game, it is structured for closed play with the detection lists representing the current status of the friendly commander's knowledge of enemy positions. These lists are used by the gamers to plan attacks on enemy units in the next six hours and to justify placing units in defensive missions when detected enemy units are approaching. The detection lists are also used by DIME controllers in selecting "not detected" enemy units which are overflown by friendly forces during an air strike. The units are input to the air attack module (P3) for overflight and possible air interdiction by overflown air defense elements.

(7) The unit status report (P8). Following the running of all ground combat and air strike battles for current six hours of combat, preparations begin for running the next six hours of combat. This process involves DIME programs P8, P7, and P9. The unit status report (P8) provides a status listing of all units currently in play. The listing includes the number of elements currently in the unit, ammunition and POL currently available to the unit, and the current mission of the unit. The report is used by the gamers to determine which units should be resupplied with both equipment and ammunition/POL. P8 also generates game forms for the resupply of each unit. These forms are used as input documents to P1. The unit status report also generates a "Game Run Summary" showing total losses of equipment across the division for the preceding six-hour period.

(8) The movement program (P9). The movement program calculates the time required to move units across several types of terrain or by helicopter to deployed positions specified by the gamers. Note that the movement represented by this program is of a strategic nature rather than the maneuver that occurs during combat (P4 calculates movement times during combat). Inputs to the model consist of the unit being moved and the distance it must travel to reach final deployment. Output from the model consists of the arrival time of the first element in the unit and the arrival time of the last element. In cases where the helicopters have been used to aid in movement of equipment and personnel, output also contains the number of helicopter sorties flown.

(9) The command and control program (P10). This program produces the time required for the command element to send a message changing the mission of a subordinate unit and for the subordinate unit to execute the mission change. Inputs to the program consist of the level of the command unit, the level of the subordinate unit, and the desired and current mission of the subordinate unit. Output is the time to affect the mission change. The command and communication nets are not explicitly modeled, but rather, the program uses a simple "table look-up" structure.

3. **THE DIME "UNITFILE".**

A. The DIME "UNITFILE" is central to the operation of all programs. The model currently will support 191 Blue units (the first 191 records on the file) and 209 Red units. Each unit record contains 150 entries describing the number of weapon systems in the unit, its ammunition/POL status, and its current mission. Table 1-1 provides a brief description of each entry in the "UNITFILE".

B. Elements 1 - 70 in the "UNITFILE" contain the weapons list for the unit. The structure of this list requires that systems performing certain battlefield functions be placed in specific positions on this list. Table 1-2 *shows these system functions for each position for both the* Blue and Red units. Typical systems are shown in Table 1-3.

(1) Direct fire platforms. Entries 1 - 20 (Table 1-2) for both Red and Blue contain the number of elements which are direct fire platforms. These locations are interrogated by the combat program (P4) for play in the direct fire portion of the battle. Note that locations 16 - 20 also serve as infantry carriers. P4 "mounts" and "dismounts" infantry personnel into these locations depending on unit mission and proximity to the enemy force.

(2) Artillery. The number of artillery elements are held in locations 21 - 27. DIME plays only one caliber of artillery for Blue and one for Red.

(3) Mortars. Entries 28 - 31 contain the number of mortars available to both Blue and Red units.

(4) MLRS/MRL. The number of multiple launch rocket systems (MLRS) for a Blue unit and the number of multiple rocket launchers (MRL) for a Red unit are contained in entries 32 - 35.

(5) Infantry personnel. Positions 36 - 40 contains the infantry personnel which serve as the pool for mounting/dismounting vehicles in locations 16 - 20. Other small arms are located in positions 41 - 47. Infantry personnel are allowed to participate in the battle only during the last 500 meters to closure.

(6) Air defense systems. Air defense systems must be placed in entries 48 - 54. Only hand-held type systems can be placed in entries 53 - 54, while entries 48 - 52 contain only vehicular type systems.

(7) Tank trucks. The number of fuel trucks located in entry 55 and tank trucks in entry 56 are used by the logistics program (P2) to calculate fuel hauling capacity. Water trucks are in position 57.

(8) Cargo trucks. The number of cargo trucks located in entry 58 and 59 are used by the logistics program to calculate the cargo hauling capacity of the unit.

Table 1-1. The DIME "UNITFILE" structure.

The "UNITFILE" for the Division Map Exercise (DIME) consists of 400 records, each containing 150 elements. The assignment of records consist of:
records 1 - 191 Blue units
records 192 - 400 Red units.

Elements are assigned to each unit record as follows:

Element number	Description	Default value
1 - 70	Weapons list	The 70 weapon quantities contained in the units (see Table 1-2).
71 - 74	Vacant	
75	Major mission	1 = Attack 2 = Defend 3 = Reserve/idle 4 = Move
76	Unit size/echelon	1 = Blue battalion/Red regiment 2 = Blue company/Red battalion
77	Unit MOPP level	1 = Unit not in MOPP 2 = Unit in MOPP
78	Unit type (X.Y)	X = Player 1 = Blue 2 = Red Y = Unit type 0 = Combat unit 1 = Artillery unit 2 = Air defense (ADA) unit 3 = Attack helicopter ground/forward rearming and refueling point (FARP) 4 = Commandpost/headquarters (CP/HQ) 5 = Engineer unit 6 = POL/AMMO supply point 7 = Maintenance point 8 = Surface-to-air missile (SAM) site 9 = Communications/radar/electronic warfare (EW) site

Table 1-1. The DIME "UNITFILE" structure (continued).

Element number	Description	Default value
79	Unit effectiveness	Percent unit effectiveness as a function of surviving weapons and personnel
80	Percent ADA suppressed (XX.YY)	XX = Vehicle ADA systems suppressed YY = Handheld systems suppressed
81	Supporting corps ADA unit	
82	Activity code	Status of unit 0 = Not active in game 1 = Active
83	Mission status	Represents Blue/Red mission during current 6-hour period MISSIONS: 0 = Meeting Engagement 1 = Indirect Fire 2 = Movement 3 = Frontal Attack 4 = Envelopmental Attack 5 = Delay 6 = Hasty Defense 7 = Prepared Defense 8 = Reserve/Rear Area 9 = Ambush
84	Cargo trucks alive at start of turn	
85	Fuel trucks alive at start of turn	
86	JP4 trucks alive at start of turn	
87	Vacant	

Table 1-1. The DIME "UNITFILE" structure (continued).

Element number	Description	Default value
88	Vacant	
89	Sensor status	X = POTA zone values (1-5) for sensor (X,Y) group Y. Y = Sensor group (0-4) detecting this particular unit. 0 = Not covered 1-3 = Applicable Blue/Red sensor group 4 = Linear FEBA-oriented sensor array NOTE: Default for sensor status = 1.4 POTA = probability of operational target acquisition. FEBA = forward edge of battle area.
90	Unit fraction covered by sensor group	Value from 0-1.0
91	Detection status (X.Y)	X = Hours left until redetected Y = Unit status with respect to detection by the opposite commander 0 = Not detected 1 = Detected but not verified 2 = Acquired/verified 3 = Lost
92	Intelligence status	Total hours this target has been tracked this detection period
93 - 94	Vacant	
95 - 100	Enemy sensors detecting this unit	
101	Fuel status of unit vehicles	Value from 0-1.0
102	Fuel status of helicopters	Value from 0-1.0

Table 1-1. The DIME "UNITFILE" structure (continued).

Element number	Description	Default value
103	Fuel on tankers (gallons)	
104	JP4 on tankers (gallons)	
105	Fuel on ground (gallons)	
106	JP4 on ground (gallons)	
107	Fuel use profile	
108	Fuel consumed (gallons)	
109	JP4 consumed (gallons)	
110	Fuel resupplied (gallons)	
111	JP4 resupplied (gallons)	
112	Fuel dispensed to other units	
113	JP4 dispensed to other units	
114	Vacant	
115	Helo ammo at beginning of CI	
116	DF at beginning of CI	
117	IF at beginning of CI	
118	AD at beginning of CI	
119	Direct fire ammo status (vehicles)	Value 0-1.0
120	Indirect fire ammo status (vehicles)	Value 0-1.0
121	Air defense ammo status (vehicles)	Value 0-1.0

Table 1-1. The DIME "UNITFILE" structure (continued).

Element number	Description	Default value
122	Helicopter ammo status	Value 0-1.0
123	Ammo on cargo vehicle, short tons (STONS)	
124	Distribution of cargo by type (XXX.YYY)	XXX = DF ammo percent (XXX = XX.X%) YYY = IF ammo fraction (YYY = YY.Y%)
125	Ammo on ground (STONS)	
126	Distribution of ground ammo by type (XXX.YYY)	XXX = DF ammo percent (XXX = XX.X%) YYY = IF ammo fraction (YYY.Y%)
127	DF ammo use profile	
128	IF ammo use profile	
129	AD ammo use profile	
130	Helicopter ammo use profile	
131	DF ammo available to be consumed	
132	IF ammo available to be consumed	
133	AD ammo available to be consumed	
134	Helo ammo available to be consumed	
135	Ammo resupplied (STONS)	
136	Ammo resupply profile (XXX.YYY)	XXX = DF ammo percent (XXX = XX.X%) YYY = IF ammo fraction (YYY = YY.Y%)
137	Ammo dispensed to other units	

Table 1-1. The DIME "UNITFILE" structure (concluded).

Element number	Description	Default value
138	Dispensed ammo profile (XXX.YYY)	XXX = DF ammo percent (XXX = XX.X%) YYY = IF ammo fraction (YYY = YY.Y%)
139	Cumulative DF ammo consumed to date	
140	Cumulative IF ammo consumed to date	
141	Cumulative AD ammo consumed to date	
142	Cumulative helo ammo consumed to date	
143	Fuel consumed to date	
144	JP4 consumed to date	
145	Vacant	
146	KM traveled this turn	
147	Fuel left	
148	JP4 fuel left	
149 - 150	Vacant	

Table 1-2. The DIME element list structure.

Number	Element Type
1 - 15	Direct Fire Platform
16 - 20	Direct Fire Platform (Infantry Carrier)
21 - 27	Artillery
28 - 31	Mortar
32 - 35	MLR/MRL
36 - 40	Small Arms (Infantry for DF Carrier)
41 - 47	Small Arms
48 - 52	ADA
53 - 54	ADA (Hand-held)
55	Fuel Truck
56	JP4 Fuel Truck
57	Water Truck
58	Ammo Truck
59	Non-ammo Truck
60 - 61	E/W Truck
62	Mine Clearing Equipment
63	Obstacle Clearing Equipment
64	AVLB
65	Pontoon Bridge
66 - 67	Engineer Equipment
68 - 70	Material Handling Equipment

Table 1-3. Example of weapon lists in the DIME "UNITFILE".

Element Number	HTLD	C-SERIES	THREAT
1	LAV/25-TOW	M1	T72
2	FAV/TOW	M2	Vacant
3	HMMWV/TOW	M3	BMP 81
4	FAV/40	ITV	Vacant
5	HMMWV/40	HMMWV/TOW	BRDM-2
6	Vacant	HMMWV/40	BRDM-AT
7	DRAGON	DRAGON	AT4
8	Vacant	Vacant	ASU-85
9	Vacant	Vacant	BMD
10	Command Vehicle	Command Vehicle	Command Vehicle
11 - 15	Vacant	Vacant	Vacant
16	Vacant	Vacant	BMP
17	Vacant	Vacant	BTR
18 - 20	Vacant	Vacant	Vacant
21	155MM	155MM	152MM
22 - 27	Vacant	Vacant	Vacant
28	107MM	181MM	120MM
29 - 31	Vacant	Vacant	Vacant
32	MLRS(T)	MLRS(Sp)	MRL
33 - 35	Vacant	Vacant	Vacant
36	Viper	Viper	RPG-16
37 - 47	Vacant	Vacant	Vacant
48	VULCAN	DIVAD	XSU-X
49	ICHAP(T)	ICHAP(Sp)	SA-13
50	IHAWK	IHAWK	SA-8
51 - 52	Vacant	Vacant	Vacant
53	Stinger Post	Stinger Post	SA-14
54	Vacant	Vacant	Vacant
55	Fuel Truck	Fuel Truck	Fuel Truck
56 - 57	Vacant	Vacant	Vacant
58	Cargo Truck	Cargo Truck	Cargo Truck
59 - 61	Vacant	Vacant	Vacant
62	Sp. Vehicle	Sp. Vehicle	Sp. Vehicle
63 - 70	Vacant	Vacant	Vacant

(9) Special vehicles. The number of special vehicles must be placed in positions 60 - 70 of the "UNITFILE". These locations are reserved for vehicles that support combat (i.e. bridging equipment, communication vans, mine clearing equipment) but are not usually involved in direct combat. These vehicles are subject to attrition from artillery, helicopters, and direct fire systems. The importance of maintaining the functional positions of each system on the "UNITFILE" cannot be overemphasized. The DIME programs have been constructed to access "expected" systems in these functional positions to perform these roles in battle.

C. All DIME programs except command and control (P10) and movement (P9) interface with the "UNITFILE" records at the time of this writing. Movement (P9) as shown in Table 1-4 is capable of accessing the "UNITFILE" but is not currently operational. Table 1-4 lists the programs showing the elements on the unit record serving as program input and the elements which are updated by the program and returned as output to the "UNITFILE".

4. THE DIME CONTROLLER.

A. The DIME controller (P0) serves as the menu-driven executive routine calling each of the DIME programs. The general logic flow of the controller is quite simple and is shown in Figure 1-2. The user is required to input the desired program from the DIME menu shown in Figure 1- 3. The chosen program is loaded and execution begins. Following execution of the program, control is returned to the menu for other program selections. If the QUIT option is invoked from the menu, the executive controller closes all files and terminates operations.

B. A listing of the BASIC code for the DIME executive controller is found in Figure 1-4.

5. THE DIME HARDWARE.

The DIME system was built for operation on a Hewlett Packard HP 9816. The original configuration consisted of the 9816, a printer, one floppy disk and a Winchester disk. The HP extended BASIC is also required. Table 1-5 contains a detailed listing of the hardware necessary to execute the program. Subsequent versions of the DIME combat program (P4), currently available at CAORA, have required expanded memory hardware. This has also been listed in the table under expanded hardware requirements.

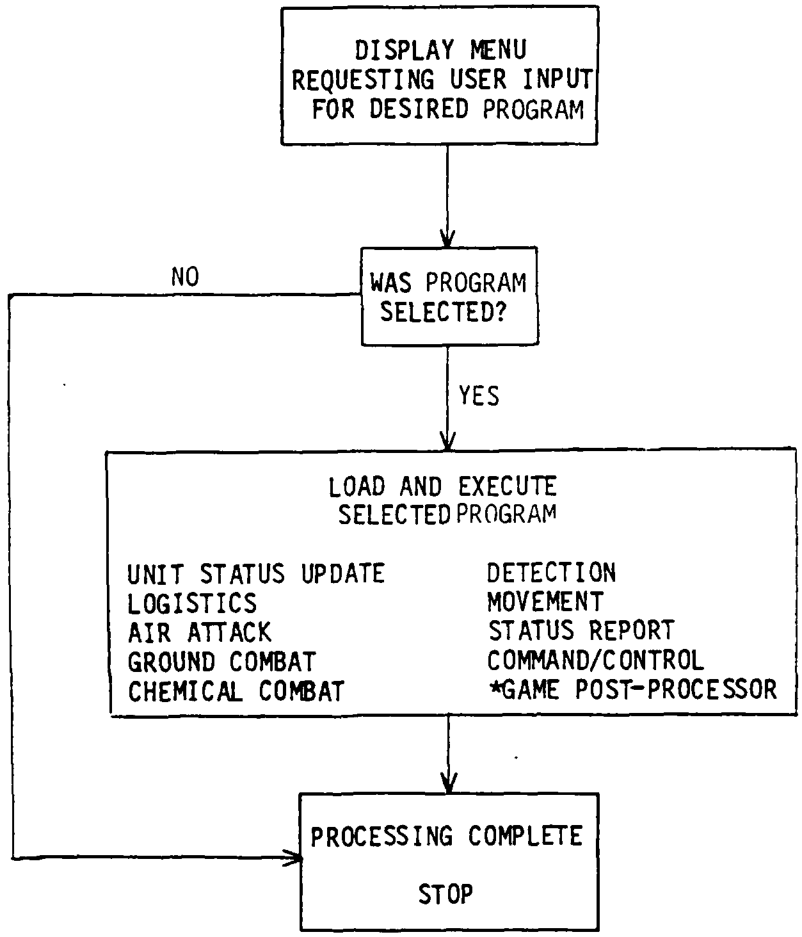
Table 1-4. Impact of DIME programs on "UNITFILE" entries.

DIME program	"UNITFILE" record elements: Input	"UNITFILE" record elements: Output	Description of program principal activity on "UNITFILE" record
Game initialization (P1)	1-150	1-70, 75, 78-83, 89, 90, 101, 103, 105,107, 110, 119-121, 123-129, 135-138	Program constructs and updates basic unit attributes of number of elements, mission and amount of AMMO/POL on hand.
Logistic support program (P2)	1-70, 75, 78, 101, 103,105, 110, 112, 119-121, 123-126, 135-138	62-63, 84-85, 101, 103, 105, 108, 116-121, 123-126, 131-133, 139-141, 143, 147	Program calculates AMMO/POL consumed for non-combat functions, updates current levels of AMMO/POL and places available ammunition in locations 131 to 133 for use by P3 and P4.
Air attack/air defense program (P3)	1-70, 75, 78, 123, 126, 133	1-70, 91-92, 125-126, 133	Program calculates losses to weapons on "UNITFILE" (1-70), and air defense AMMO consumed (133). These values are then updated along with previously undetected overflown units (91, 92).
Ground combat program (P4)	1-70, 75-78, 83, 131-133	1-70, 131-133	Program calculates losses to weapons as "UNITFILE" (1-70) and the amount of direct fire, air defense and indirect fire ammunition consumed (131-133) and updates these values on the "UNITFILE".
Chemical combat program (P5)	1-70, 75, 77, 78	1-70, 77	Program calculates losses to weapons on "UNITFILE" (1-70) from chemical attack based on mission (75), unit type (78) and MOPP level (77). Updated weapons lists (1-70) and MOPP level (77) are output to "UNITFILE".

Table 1-4. Impact of DIME programs on "UNITFILE" entries (concluded).

DIME program	"UNITFILE" record elements: Input	Output	Description of program's principal activity on "UNITFILE" record
Detection program (P7)	1-70, 75, 78, 83, 91, 92	91, 92	Program determines number of elements exposed to sensors based on unit mission and location from sensor. Program then calculates and updates unit detection status (91) and hours unit has been tracked by friendly sensors (92).
Movement program (P9)*	1-70, 75, 77	146	Program calculates the time to move elements described in weapon lists (1-70) the distance specified by gamer. Total distance moved by unit (146) is updated to "UNITFILE".
Unit status report (P8)	1-150	101, 107-113, 119 - 141	Program uses the entire "UNITFILE" to provide a hard copy summary of the unit. The program calculates and updates parameters describing current ammunition levels following 6 hours of divisional activities.

*Not currently operational with the "UNITFILE".



*Game post-processor not currently used.

Figure 1-2. The DIME Executive Controller Logic Flow is menu driven allowing the user to select any module for use.

```
DDDDDDD        III       M         M      EEEEEEEEE
D      D        I        MM       MM      E
D       D       I        M M     M M      E
D       D       I        M   MMMM  M      EEEEEE
D       D       I        M         M      E
D       D       I        M         M      E
D      D        I        M         M      E
DDDDDDD        III       M         M      EEEEEEEEE

                PROGRAM MENU ( GAME VERSION )

1.  DATA OPERATIONS                   7.  DETECTION/TARGET LIST
2.  LOGISTICS                         8.  CONSOLIDATE/TURN SUMMARY
3.  AIR ATTACK/AIR DEFENSE            9.  MOVEMENT CALCULATOR
4.  GROUND COMBAT                     10. COMMAND/CONTROL CALCULATOR
5.  CHEMICAL COMBAT                   11. GAME POST PROCESSOR *
6.  UNIT LOSS ASSESSMENT              12. QUIT!!!!!!!!!!!!!!!!
```

```
*  GAME POST PROCESSOR not currently used.
```

Figure 1-3. DIME input menu.

```
10 !!!   "DIME" IS THE MENU CONTROL PROGRAM FOR THE DIVISION MAP EXERCISE
20 !     (DIME).  DIME WAS DEVELOPED BY THE OPNS ANALYSIS  BRANCH OF THE
30 !     COMBINED ARMS OPERATIONS RESEARCH ACTIVITY.  THE CHIEF OF THE
40 !     PROJECT IS MR. H. KENT PICKETT. A/V 552-4613.  THE PROGRAM LAST
50 !     CHANGED ON 15 SEPT 83
60 !
70 Disk$=":9134,704,0"
80 PRINTER IS 1
90 PRINT USING "@,# "
100 PRINT TABXY(1,1),TAB(17),"DDDDDDD       III       M         M      EEEEEEEEE"
110 PRINT TAB(17),"D      D       I       MM       MM       E"
120 PRINT TAB(17),"D       D      I       M M     M M       E"
130 PRINT TAB(17),"D       D      I       M  MMMM  M        EEEEEE"
140 PRINT TAB(17),"D       D      I       M        M        E"
150 PRINT TAB(17),"D       D      I       M        M        E"
160 PRINT TAB(17),"D      D       I       M        M        E"
170 PRINT TAB(17),"DDDDDDD       III      M        M        EEEEEEEEE"
180 PRINT
190 PRINT TAB(24),"PROGRAM MENU ( CONTRACT BENCH )"
200 PRINT
210 PRINT "1.  DATA OPERATIONS ",TAB(40),"7.  DETECTION/TARGET LIST"
220 PRINT "2.  LOGISTICS ",TAB(40),"8.  CONSOLIDATE/TURN SUMMARY"
230 PRINT "3.  AIR ATTACK/AIR DEFENSE",TAB(40),"9.  MOVEMENT CALCULATOR"
240 PRINT "4.  GROUND COMBAT",TAB(40),"10. COMMAND/CONTROL CALCULATOR"
250 PRINT "5.  CHEMICAL COMBAT",TAB(40),"11. GAME POST PROCESSOR"
260 PRINT "6.  UNIT LOSS ASSESSMENT",TAB(40),"12. QUIT!!!!!!!!!!!!!!!!!!"
270 INPUT "TYPE DESIRED PROGRAM NUMBER, PRESS ENTER:  ",S$
280 IF S$="12" THEN GOTO Halt
300 LOAD "NEW_P"&S$&Disk$
310 Halt:  !
320 PRINT USING "@,#"
330 END
```

Figure 1-4. Executive controller code.

Table 1-5. Hardware configuration for the DIME model.

Hardware	Description
HP9816 Computer	The HP9816 computer is a member of the HP series 200 family of personal technical computers. It supports a number of programming languages and operating systems, and has the capacity to link up to diverse peripheral devices.
HP9121 D/S Disc Memory	The HP9121 D/S Disc memories are random access data storage devices. The HP 9121S contains a single 3 1/2-inch disc drive providing 286.72 Kbytes of storage capacity. The HP9121D contains two 3 1/2-inch disc drives providing a total storage capacity of 573.44 Kbytes.
HP9134A Disc Memory	The HP9134A disc memory is a random access data storage device containing a 5 1/4-inch Winchester disc drive providing 4.6 Mbytes of storage capacity.
HP82905B Printer	The HP82905B Printer is a general purpose printer featuring 80 character per second bi-directional printing. The printer utilizes a 9x9 dot matrix character format. It prints in 40, 66, 80, or 132 columns. You can choose among normal, expanded, compressed, or compressed expanded characters. Normal size character may also be emphasized. The printer has a graphics mode which has the ability to print illustrations, charts, graphs, block letters, etc. using patterns of dots under software control. Functions such as line spacing, form length, and skip over perforation are also under software control.
HP9888A Bus Expander	The HP 9888A Bus expander allows for connecting up to eight interface cards and eight memory cards or up to 16 memory cards to HP Series 200 Personal Technical Computers, using an I/O slot in the computer.

CHAPTER 2

GAME INITIALIZATION

1. PURPOSE.

The purpose of the DIME game initialization program (P1) is to create and edit the unit status file ("UNITFILE"). Listings of the "UNITFILE" may also be obtained through this program, along with many other options.

2. GENERAL.

This program deals with a 400-record "UNITFILE". Records 1-191 are reserved for Blue units and records 192-400 are reserved for Red units. The program consists of 10 subroutines that run from the menu which appears at the beginning of the program (see Figure 2-1).

3. DATA FLOW.

A. Input data. Data for this program comes through two means, data files and input data (see Figure 2-2).

(1) Data files. All data files are external to the program and stored on the hard disk. These consist of the "UNITFILE", the table of organization/equipment ("TOEFILE"), the "NAMEFILE", the ammunition capacities file, the fuel capacities file, system effectiveness files and weapon type files.

(2) Input data. Data is input to this program by means of the unit input sheets and the gamer/staff worksheets.

(a) The unit input sheets are used to create an original "UNITFILE." Figure 2-3 displays the unit input sheet. Option 12 ("Enter initial unit status") should be chosen in order to input these sheets.

1. Unit #. Represents the record number for the "NAMEFILE" being created.

2. Side. A number representing the side (Blue = 1, Red = 2). This number is stored in the "TOEFILE" (position 72) as a whole number.

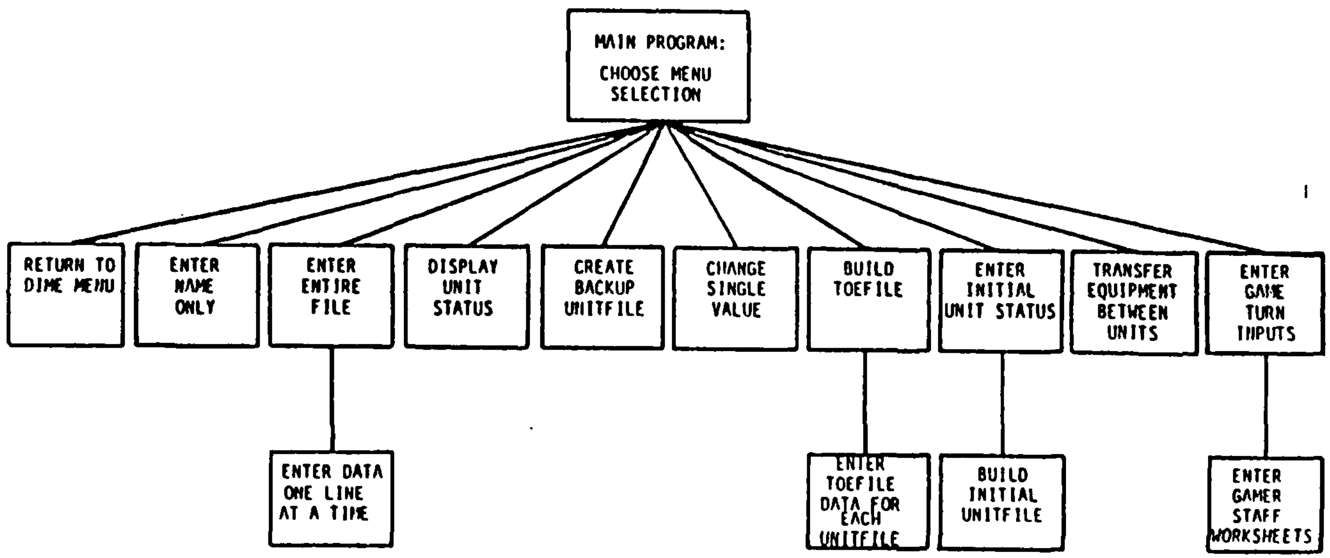


Figure 2-1. Game initialization program menu.

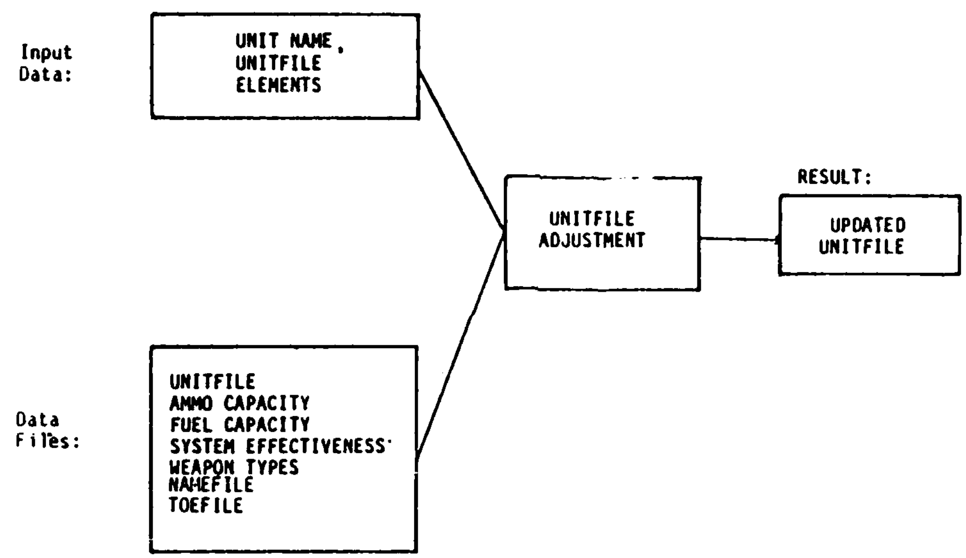


Figure 2-2. Game initialization information flow.

UNIT INPUT SHEET

UNIT NO. SIDE UNIT TYPE

** UNIT TYPE **

0: COMBAT	5: ENGINEER
1: ARTILLERY	6: SUPPLY
2: AIR DEFENSE	7: MAINTENANCE PT
3: FARRP	8: BRIDGE
4: COMMAND POST	9: COM/EW SITE

UNIT NAME

STARTEX SYSTEM QTY

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70

MUST EQUAL 100%

A	B	C	D	E	F	G	H

** ACTIVITY CODE **

0: INACTIVE
1: ACTIVE

A: % FUEL ON TANKERS
B: % AMMO ON CARGO VEHICLE
C: % DIRECT FIRE AMMO
D: % INDIRECT FIRE AMMO
E: % AIR DEFENSE AMMO
F: ACTIVITY CODE
G: TACT MISSION
H: ECHELON

** TACT MISSION **

0: MVMTN TO CONT	5: DELAY
1: INDIRECT FIRE	6: HASTY DEFENSE
2: MOVEMENT	7: PREP. DEFENSE
3: FRONTAL ATTACK	8: RESERVE/REAR
4: ENV. ATTACK	9: AMBUSH

** ECHELON **

0: BLUE BATTALION/RED REGIMENT
1: BLUE COMPANY/RED BATTALION

Figure 2-3 Unit input sheet.

3. Type. A number representing the unit type, where:

0 = Combat unit
1 = Artillery unit
2 = Air defense unit
3 = Attack helicopter site/FARP
4 = Command post/headquarters (CP/HQ)
5 = Engineer unit
6 = POL/AMMO supply unit
7 = Maintenance point
8 = Surface-to-air missile (SAM) site
9 = Communications/radar/electronic warfare (EW) site.

The unit number stored in the "TOEFILE" (position 72) is input as a "decimal" number. The number in front of the decimal is the unit's side, the number following the decimal represents the unit's type. Some examples of position in the "TOEFILE" follow:

1.3 represents a Blue air defense unit
1.8 represents a Blue SAM site
2.5 represents a Red engineer unit

4. Name. A 16 character alphanumeric string which represents the unit and is meaningful to the gamers. This string is held in the "NAMEFILE".

5. Input Lines 1 - 7 are used to input the actual quantity of 70 weapon elements assigned to a unit. These are the first 70 elements of each record of the "UNITFILE". These represent on-hand quantities as opposed to the authorized quantities listed in the "TOEFILE", above.

6. Line 8.

a. Fuel on tankers. The fraction of the unit's fuel, in gallons, loaded on tankers that are organic to the unit. Stored in the 103rd element of the "UNITFILE."

b. Ammo on cargo vehicle. Fraction of ammunition (loads) carried by the unit's ammunition trucks. Stored in the 123rd element of the "UNITFILE".

c. DF ammo status. Fraction of the unit's direct fire (DF) load capacity that is full (Note: DF status + IF status + AD status = 100). Stored in the 119th element of the "UNITFILE".

d. IF ammo status. Fraction of the unit's indirect fire (IF) basic load capacity that is full (Note: DF status + IF status + AD status = 100). Stored in the 120th element of the "UNITFILE".

e. AD ammo status. Fraction of the unit's air defense (AD) basic load capacity that is full (Note: DF status + IF status + AD status = 100). Stored in the 121st element of the "UNITFILE".

f. Activity code. If a unit is active in the critical incident (CI), set this to 1, otherwise it is 0. This is element 82 of the "UNITFILE".

g. Mission. Represents element 83 of the "UNITFILE", where:

0 = Meeting engagement
1 = Indirect fire
2 = Movement
3 = Frontal attack
4 = Envelopmental attack
5 = Delay
6 = Hasty defense
7 = Prepared defense
8 = Reserve/rear area
9 = Ambush.

h. Echelon. The unit's echelon. Enter 0 for Blue battalions/Red regiments (or larger) and 1 for Blue company/Red battalions. This is element 76 of the "UNITFILE".

(b) The gamer/staff worksheets (see Chapter 10, Figure 10-3) are completed by the gamers and input in P1 using Option 10 (enter game turn inputs). These worksheets are used to change "UNITFILE" information at the end of each critical incident (CI).

B. Output data. The only data output from this program is that data contained in the external files. The "UNITFILE" contains one array for each unit. Units not used should contain zeroes. The "TOEFILE" and "NAMEFILE" are normally output only during initial creation of the units.

4. FILE STRUCTURE.

Data files used in P1 are all external to the code. There are no internal files constructed as data statements within the code.

A. A(*). Contains the packaged weight of ammunition, in tons, for Red and Blue's 70 weapon elements.

B. F(*). Contains the capacity, in gallons, for Red and Blue's 70 weapon elements.

C. S(*). Contains the system effectiveness values for Red and Blue's 70 elements.

D. W(*). Contains a number (1-3) representing the ammunition type for Red and Blue's 70 elements, where:

1 = Direct fire
2 = Indirect fire (artillery)
3 = Air defense.

E. The "UNITFILE" is the major file within P1. A complete description of the elements within the "UNITFILE" may be found in Chapter 1.

F. The "TOEFILE" or table of organization and equipment (TOE) file consists of 400 records (one for each unit) and holds the number of starting force elements in positions 1 to 70. Position 71 of each record holds the total beginning effectiveness value for the unit. Position 72 contains a value representing the side (Red/Blue) and the type of unit.

G. The "NAMEFILE", developed into 400 records, holds the name for each of the units.

H. For further information on any data, please refer to volume III of the DIME documentation.

5. **ALGORITHMS.**

A. Figure 2-4 depicts the logic flow of the P1 program.

B. The major algorithm in this program calculates the combat effectiveness percentage for units. It begins by calculating the current total effectiveness:

$$C = \sum_{i=1}^{70} N_i * S_i \qquad \text{(Eq. 2-1)}$$

where:

C = current total effectiveness.
N_i = the current number of weapon systems of type i.
S_i= the system effectiveness value for weapon type i.

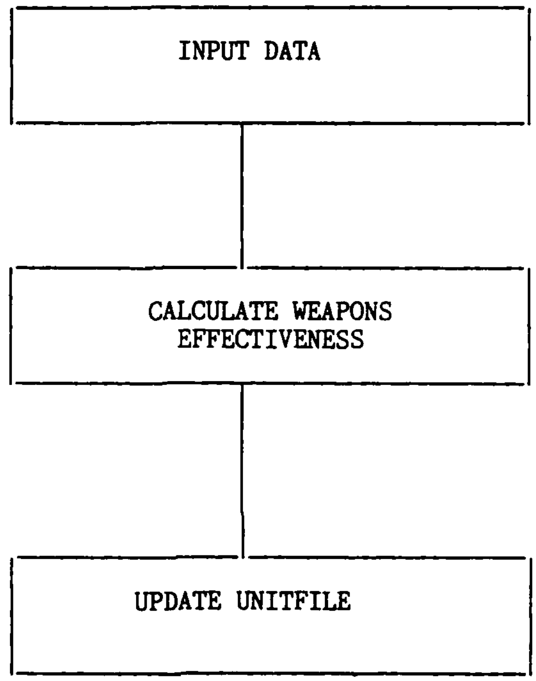


Figure 2-4. Game initialization logic flow.

The percentage is then calculated for each unit by:

$$Eff = C / U \qquad \text{(Eq. 2-2)}$$

where:

U = the total authorized effectiveness value for unit j. This value (U) is stored in the "TOEFILE" in record j.
Eff = the effectiveness percentage for unit j.

6. "UNITFILE" IMPACT.

All inputs to this program will affect the "UNITFILE" directly as they are entered. In addition, when item number 83 (mission status) of the "UNITFILE" is changed, it in turn causes item 75 (major mission) and items 107 and 127 - 129 (ammunition and fuel use profiles) to be changed. The game initialization program does not interact with any other program other than to return to the main DIME menu.

7. CODE.

A. The game initialization code is divided into 12 major subroutines. These subroutines correspond to the 12 items on the P1 menu (Figure 2-1). A short description of each menu option follows.

(1) Option 1 — Return to DIME menu. This merely returns control to the DIME menu (PO) discussed in Chapter 1.

(2) Option 2 — Enter name only. This option allows you to change the "NAMEFILE" for any given unit. It will not affect the "UNITFILE" or "TOEFILE".

(3) Option 3 — Enter entire file. With this option, one may enter all 150 elements of the "UNITFILE".

(4) Option 4 — Display unit status. This option prints the unit number, name, and the 150 elements of its "UNITFILE". It is possible to list specific units or all units with this option.

(5) Option 5 — Backup or Restore "UNITFILE". This copies the current "UNITFILE" from hard disk onto a floppy disc (or vice versa). The entire "UNITFILE" will not fit a single floppy disc, so one is used for Blue units and another for Red units.

(6) Option 6 — Change single value. This option permits correction of individual elements within the "UNITFILE". It does not allow, however, the changing of elements 75, 79, 107 127, 128, or 129. Element 75 (major mission) is dependent on element 83 (mission 0-9). Elements 107 and 127 -

129 are set automatically to the same number indicated by element 83. Element 79 is the current effectiveness which changes from CI to CI, due to single value change or transfer of unit elements. Please note: the setting of elements 75, 79, 107 and 127 - 129 are always reset before each output of the "UNITFILE" in **P1**.

(7) Option 7 — Build "TOEFILE". Within this option, a specific record of the "TOEFILE" may be built or displayed. Initial building is usually done in the next option (8), however, if changes need to be made to the "TOEFILE", it will have to be input in this option. Please note: If changes are made using this option, the total combat effectiveness is changed. Use the program EFF found in Chapter 11 to ensure that the effectiveness value in the "UNITFILE" is corrected.

(8) Option 8 — Enter initial unit status (old version). The old version of this procedure is no longer in use. See Option 12, below.

(9) Option 9 — Transfer equipment between units. This option allows transfer between units of the same side (Red/Blue). A specific weapon element (1-70) may be transferred. This routine is not restricted to transferring entire units. As described in Table 2-1, the gaining unit is the one which is receiving the equipment being transferred, and the losing unit is the one which is shipping out the equipment to the gaining unit. Not only are the weapon elements transferred, but their accompanying ammunition and fuel are transferred. This routine affects the "UNITFILE" and "TOEFILE" including the combat effectiveness which is recalculated after the transfer.

(10) Option 10 — Enter game turn inputs. This option works off the gamer/staff worksheet which was output at the end of the previous CI through **P8** (unit status report). After gamers have filled out the worksheet, it is input line by line. An echo of the inputs are also printed for record-keeping purposes.

(11) Option 11 — Copy one unit to another. This selection permits the user to copy the data entered for one unit to another unit.

(12) Option 12 — Enter initial unit status (new version). Data from the input sheet shown in Figure 2-3 is input using this option. Please refer to paragraph 3A(2) of this chapter for a complete discussion of the inputs.

B. Primary variables for each subroutine are listed in Table 2-1. A listing of **P1** code appears in Table 2-2.

Table 2-1. Game Initialization Subroutine Table.

Functional area(s): A. File Operations

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Main program	Allows menu selection and branches to all routines from here.	A. A(S,I)	Ammo data array for Red (S=1) and Blue (S=2) weapon elements (I=1-70).
		B. F(S,I)	Fuel data array (in gallons).
		C. S(S,I)	System effectiveness data.
		D. W(S,J)	Weapon category: 1 = Direct fire 2 = Indirect fire 3 = Air defense
Ret_master_menu	Return to DIME menu		
Input_unit_name	Allows input of individual unit names	A. M$	Unitname; 8 character maximum
Enter_unitfile	Enter all 150 elements of a selected unit.	A. N(I)	"UNITFILE" elements of one record (I=1-150).
Disp_unit_stat	Prints contents of "UNITFILE" and "NAMEFILE".		
Cre_backup_file	Creates a backup of "UNITFILE" on a floppy disk.	A. File_name$	File name under which "UNITFILE" is copied.

Table 2-1. Game Initialization Subroutine Table. (continued)

Functional area(s): A. File Operations. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Cha_single_value	Allows changes to any of the 150 elements of the "UNITFILE".		
Build_toefile	Build "TOEFILE"	A. U(I)	"TOEFILE" elements - number of starting force elements (I=1-70), and I=71 Total beginning effectiveness value for the unit. I=72 Side and unit type.
Transfer	Transfer equipment between units	A. Ugain(I)	"TOEFILE" for gaining unit when transferring equipment between units (I=1-72).
		B. U(I)	"TOEFILE" for losing unit.
		C. Ngain (I)	"UNITFILE" for gaining unit when transferring equipment (I=1-150).
		D. N(I)	"UNITFILE" for losing unit.
		E. Eff_lose	Effectiveness lost by losing unit.
		F. Eff_u	Effectiveness of losing unit.
		G. Eff_ug	Effectiveness of gaining unit.
		H. Avg_fuel	Average fuel consumed.
		I. Tot_avg_fuel	Total amount of average fuel for systems.
		J. Gain_df_frac	Fraction of direct fire weapon systems gained.

Table 2-1. Game Initialization Subroutine Table. (continued)

Functional area(s): A. File Operations. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Transfer (concluded)		K. Lose_df_frac	Fraction of DF systems lost.
		L. Gain_if_frac	Fraction of indirect fire systems gained.
		M. Lose_df_frac	Fraction of IF systems lost.
		N. Gain_ad_frac	Fraction of AD systems gained.
		O. Lose_ad_frac	Fraction of AD systems lost.
		P. Ammo_transfer	Amount of ammo transferred.
		Q. Df_ammo_trans	Amount of DF ammo transferred.
		R. If_ammo_trans	Amount of IF ammo transferred.
		S. Ad_ammo_trans	Amount of AF ammo transferred.
		T. Fuel_gain	Current amount of fuel for gaining unit.
		U. Fuel_lose	Current amount of fuel for losing unit.
		V. Ammo_gain	Current amount of ammo for gaining unit.
		W. Ammo_lose	Current amount of ammo for losing unit.
		X. Tag	Total ammo gained.
		Y. Tal	Total ammo lost.
		Z. Tot_fuel_gain	Total fuel gained.
		AA. Tot fuel lose	Total fuel lost.

Table 2-1. Logistics Subroutine Table. (concluded)

Functional area(s): A. File Operations (concluded).

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Game_turn	Accepts inputs from P8 worksheets.		
Calc_combat_eff	Calculates combat effectiveness percentage for each unit.		
Unitfile_disk	Before any updates are made to the "UNITFILE" this assures that positions 75, 110 and 127-129 of the "UNITFILE" correspond correctly with the unit's mission.		

Functional area(s): B. Check Inputs

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Ck_var	Checks inputs for incorrect ranges.		

```
10    ! P1--FILE OPERATIONS THIS VERSION PLAYS 400 UNITS AND DETAILED SENSORS
20    ! DATA CHANGED JANUARY 29, 1986 BY ROB BELFLOWER. BDM
30    ! EXPANDED VERSION -- JUNE 9, 1986 -- BY OAO CORP.
31    !      DECLASSIFIED -- AUG 7, 1986 -- BY OAO CORP.   ** DC **
40    OPTION BASE 1
50    Disk$=":9134,704,0"
56    Dcdisk$=":9134,704,0"  ! ** DC **
60    DIM Ammo_gain(3),Ammo_lose(3),Ammo_sys(3),Tal(3),Tag(3)
70    DIM A(2,70),F(2,70),S(2,70),W(2,70),U(72),Ugain(72)
80    DIM T(70),Toe(70),Sys(70),N(150),Ngain(150),M$[16],Type$[13]
90    ASSIGN @Pname TO "NAMEFILE"&Disk$
100   ASSIGN @Punit TO "UNITFILE"&Disk$
110   ASSIGN @Ptoe TO "TOEFILE"&Disk$
120   REM - INITIALIZE
121   !
122   ! ** DC **  7 AUG 1986
123   !
130   ASSIGN @Pammocap TO "AMMO_CAP"&Dcdisk$
140   ENTER @Pammocap,1;A(*)
150   ASSIGN @Pammocap TO *
160   !
290   ASSIGN @Pfuelcap TO "FUEL_CAP"&Dcdisk$
300   ENTER @Pfuelcap,1;F(*)
310   ASSIGN @Pfuelcap TO *
450   !
460   ASSIGN @Psyseff TO "SYS_EFF"&Dcdisk$
470   ENTER @Psyseff,1;S(*)
480   ASSIGN @Psyseff TO *
490   !
500   ASSIGN @Pwpntyp TO "WPN_TYP"&Dcdisk$
510   ENTER @Pwpntyp,1;W(*)
520   ASSIGN @Pwpntyp TO *
521   !
530   ! ** END DC **
540   !
690   PRINT USING "@,#"
700   PRINT "            DATA OPERATIONS MENU"
710   PRINT "     "
720   PRINT "      1 -- RETURN TO MASTER MENU"
730   PRINT "      2 -- ENTER OR CHANGE NAME ONLY"
740   PRINT "      3 -- ENTER ENTIRE FILE (1-150) FOR A UNIT"
750   PRINT "      4 -- DISPLAY UNIT STATUS"
760   PRINT "      5 -- BACKUP OR RESTORE UNITFILE"
770   PRINT "      6 -- CHANGE SINGLE VALUE"
780   PRINT "      7 -- TOEFILE UTILITIES"
790   PRINT "      8 -- ENTER INITIAL UNIT STATUS - OLD VERSION"
800   PRINT "      9 -- TRANSFER EQUIPMENT BETWEEN UNITS"
810   PRINT "     10 -- ENTER PLAYER P8 POST-GAMETURN RESUPPLY AND STATUS INFO"
820   PRINT "     11 -- COPY ONE UNIT TO ANOTHER UNIT"
830   PRINT "     12 -- ENTER INITIAL UNIT STATUS - NEW VERSION"
840   PRINT "     "
850   INPUT "ENTER NUMBER OF DESIRED ACTION--",I
```

Table 2-2. Game initialization code.

```
860    IF I=12 THEN GOSUB Init_stat
870    IF I=11 THEN GOSUB Record_number
880    IF I>10 OR I<1 THEN 690
890    ON I GOSUB Ret_master_menu,Input_unit_name,Entire_unitfile,Disp_unit_stat
Cre_backup_file,Cha_single_valu,Build_toefile,Init_unit_stat,Transfer,Game_tur
900    GOTO 690
910    !---------------------------------------------------- ROB
920 Record_number:    !
930    PRINT USING "@"
940    ASSIGN @Nname TO "NAMEFILE:9134,704,0"
950    ASSIGN @Nunit TO "UNITFILE:9134,704,0"
960    ASSIGN @Ntoe TO "TOEFILE:9134,704,0"
970    PRINT "THIS ROUTINE WILL COPY ONE UNIT RECORD INTO ANOTHER UNIT RECORD."
980    PRINT "IT WILL ERASE ANY DATA ALREADY STORED IN THE NEW RECORD"
990    PRINT "NUMBER AND REPLACE IT WITH THE CHANGED DATA.  IT WILL"
1000   PRINT "NOT ERASE THE DATA STORED IN THE OLD RECORD."
1010   PRINT
1020   PRINT "USE RECORD NUMBER 999 TO END THIS ROUTINE"
1030   X=0
1040   Y=0
1050   PRINT
1060   INPUT "WHAT IS THE OLD RECORD NUMBER",X
1070   IF X=999 THEN RETURN
1080   INPUT "WHAT IS THE NEW RECORD NUMBER",Y
1090   IF Y=999 THEN RETURN
1100   ENTER @Punit,X;N(*)
1110   ENTER @Ptoe,X;U(*)
1120   ENTER @Pname,X;M$
1130   OUTPUT @Nunit,Y;N(*)
1140   OUTPUT @Ntoe,Y;U(*)
1150   OUTPUT @Nname,Y;M$
1160   GOTO 920
1170   RETURN
1180   !------------------------------------------------------------------
1190 Ret_master_menu:!
1200   ASSIGN @Pname TO *
1210   ASSIGN @Punit TO *
1220   ASSIGN @Ptoe TO *
1230   GOTO Subprogram_end
1240   RETURN
1250   !------------------------------------------------------------------
1260 Input_unit_name:!
1270   PRINT USING "@"
1280   INPUT "UNIT NAME ENTRY (16 CHARACTER MAXIMUM, 999=STOP). ENTER UNIT NUMBE
 ",I
1290   IF I=999 THEN 1360
1300   IF I<1 OR I>400 THEN 1270
1310   INPUT "ENTER UNIT NAME",M$
1320   OUTPUT @Pname,I;M$
1330   ENTER @Pname,I;M$
1340   PRINT USING "@,16A";M$
1350   GOTO 1280
```

Table 2-2. Game initialization code.

```
1360  RETURN
1370  !-----------------------------------------------------------------------
1380 Entire_unitfile:!
1390  PRINT USING "@"
1400  PRINT "UNIT FILE ENTRY (NAME,PARAMETERS)"
1410  PRINT USING "@"
1420  INPUT "ENTER UNIT NUMBER (1-400, 999=STOP)",I
1430  IF I=999 THEN 1590
1440  IF I>400 OR I<1 THEN 1410
1450  PRINT USING "@"
1460  INPUT "ENTER UNIT NAME (16 CHARACTERS)",M$
1470  PRINT M$
1480  FOR K=1 TO 15
1490    PRINT "LINE";K;"  ";
1500    FOR J=1 TO 10
1510      INPUT "INPUT FIELD VALUE",N((K-1)*10+J)
1520      PRINT N((K-1)*10+J);
1530    NEXT J
1540    PRINT
1550  NEXT K
1560  OUTPUT @Pname,I;M$
1570  GOSUB Calc_combat_eff
1580  GOTO 1410
1590  RETURN
1600  !-----------------------------------------------------------------------
1610 Disp_unit_stat:!
1620  PRINT USING "@"
1630  PRINT "UNIT STATUS DISPLAY"
1640  PRINT
1650  INPUT "CHECK LOADSHEET VALUES <L>, OR SCRUTINIZE ALL 150 VALUES <S>?",Ans
1660  INPUT "ENTER OUTPUT DESIRED:   1-ALL UNITS   2-SPECIFIC UNITS",Z5
1670  INPUT "DISPLAY TO 1=SCREEN ONLY OR 2=SCREEN AND PRINTER",A$
1680  Page=0
1690  IF Z5=1 THEN
1700    Loadsheet=0
1710    PRINT USING "@,#"
1720    FOR I=1 TO 400
1730      GOSUB Print_unitfile
1740    NEXT I
1750  ELSE
1760    INPUT "UNIT NUMBER (900=START OVER, 999=MAIN MENU)",I
1770    IF I=900 THEN 1620
1780    IF I=999 THEN 1830
1790    IF I<1 OR I>400 THEN 1760
1800    GOSUB Print_unitfile
1810    GOTO 1760
1820  END IF
1830  RETURN
1840  !
1850 Print_unitfile:!
1860  ENTER @Punit,I;N(*)
1870  ENTER @Pname,I;M$
```

Table 2-2. Game initialization code.

```
1880  IF N(79)=0 THEN 2820
1890  Side=INT(N(78))
1900  Kind=INT((N(78)-Side+.05)*10)
1910  IF Kind=0 THEN Type$="COMBAT"
1920  IF Kind=1 THEN Type$="ARTILLERY"
1930  IF Kind=2 THEN Type$="AIR DEFENSE"
1940  IF Kind=3 THEN Type$="FARRP"
1950  IF Kind=4 THEN Type$="COMMAND POST"
1960  IF Kind=5 THEN Type$="ENGINEER"
1970  IF Kind=6 THEN Type$="SUPPLY"
1980  IF Kind=7 THEN Type$="MAINTENANCE"
1990  IF Kind=8 THEN Type$="BRIDGE"
2000  IF Kind=9 THEN Type$="COMMO/EW SITE"
2010  PRINTER IS 1
2020  X=INT(N(124))/10
2030  Y=(N(124)-INT(N(124)))*100
2040  IF Side=1 THEN
2050    Fuel=1800
2060    Tons=6.5
2070  ELSE
2080    Fuel=1288
2090    Tons=4.5
2100  END IF
2110  IF N(103)>N(101) THEN
2120    Gal=(N(103)/Fuel/N(55))*100
2130  ELSE
2140    Gal=N(101)*100
2150  END IF
2160  Ammo=N(123)
2170  IF Gal=0 THEN Gal=1
2180  IF Ammo=0 THEN Ammo=1
2190  IF A$="1" THEN
2200    PRINT " ";"UNIT NO."," SIDE","TYPE",Type$
2210    PRINT I,Side,Kind
2220    PRINT " ";M$
2230    IF Ans$="L" THEN
2240      FOR J=1 TO 70
2250        PRINT USING "6D.3D,1X,#";N(J)
2260        IF J MOD 7=0 THEN
2270          PRINT
2280        END IF
2290      NEXT J
2300      !PRINT ROUTINE BELOW WAS ADDED AFTER THE CORRECT FORMULA WAS ADDED TO T
HE INIT_STAT ROUTINE TO LOAD DF AND IDF AMMO.
2310      PRINT " ";"% FUEL ","% AMMO","   %DF "," %IF"," %ADA ","ACTIVITY","MIS
SION","ECHELON"
2320      PRINT USING "7D,8D,11D,8D,11D,10D,8D.1D,10D";Gal,100*(Tons*N(58))/Ammo
,X,Y,100-X-Y,N(82),N(83),N(76)
2330      PRINT
2340      PRINT
2350    ELSE
2360      PRINT USING "22(7(6D.3D,1X),/)";N(*)
```

Table 2-2. Game initialization code.

```
2370      PRINT
2380    END IF
2390  ELSE
2400    PRINTER IS 702
2410    PRINT " ";"UNIT NO."," SIDE","TYPE",Type$
2420    PRINT I,Side,Kind
2430    PRINT " ";M$
2440    IF Ans$="L" THEN
2450      FOR J=1 TO 70
2460        PRINT USING "6D.3D,1X,#";INT(N(J))
2470        IF J MOD 7=0 THEN
2480          PRINT
2490        END IF
2500      NEXT J
2510      PRINT " ";"% FUEL ","% AMMO","   %DF "," %IF"," %ADA ","ACTIVITY","MI
SION","ECHELON"
2520      PRINT USING "7D,8D,11D,8D,11D,10D,8D.1D,10D";Gal,100*(Tons*N(58))/Amm
,X,Y,100-X-Y,N(82),N(83),N(76)
2530      Loadsheet=Loadsheet+1
2540      IF Loadsheet=3 THEN
2550        PRINT USING "@,#"
2560        Loadsheet=0
2570      ELSE
2580        PRINT
2590        PRINT
2600        PRINT
2610      END IF
2620      PRINTER IS 1
2630      GOTO 2820
2640    ELSE
2650      PRINT USING "22(7(6D.3D,1X),/)";N(*)
2660      Page=Page+1
2670      IF Page=2 THEN
2680        PRINT USING "@,#"
2690        Page=0
2700      ELSE
2710        PRINT
2720        PRINT
2730      END IF
2740      PRINTER IS 1
2750    END IF
2760  END IF
2770 !Coun=Coun+1
2780 !IF Coun=2 THEN 1310
2790 !PRINTER IS 702
2800 !GOTO 1250
2810 !PRINTER IS 1
2820  RETURN
2830  !------------------------------------------------------------------
2840 Cre_backup_file:!
2850  PRINT USING "@,#"
2860  PRINT "            BACKUP/RESTORE MENU"
```

Table 2-2. Game initialization code.

```
2870  PRINT "     "
2880  PRINT "       1 -- BACKUP UNITFILE TO FLOPPY"
2890  PRINT "       2 -- RESTORE UNITFILE FROM FLOPPY"
2900  PRINT "       3 -- RETURN TO MAIN MENU"
2910  PRINT "     "
2920  INPUT "ENTER NUMBER OF DESIRED ACTION--",I
2930  IF I=1 THEN GOTO Backunit
2940  IF I=2 THEN GOTO Restunit
2950  IF I=3 THEN RETURN
2960  GOTO Cre_backup_file
2970 Backunit:  !
2980  Sunit=1
2990  Eunit=191
3000  Trecs=191
3010  Oname$="BLUE"
3020  GOSUB Make_unit
3030  Sunit=192
3040  Eunit=400
3050  Trecs=209
3060  Oname$="RED"
3070  GOSUB Make_unit
3080  RETURN
3090 Make_unit:     !
3100  DISP "INSERT ";Oname$;" UNITFILE DISK IN FLOPPY DRIVE 0 - PRESS CONT WHEN
READY"
3110  PAUSE
3120  Bunit$="UNITFILE"&Oname$[1,1]
3130  DISP "CREATING ";Bunit$;"...";
3140  CREATE BDAT Bunit$&":HP9121,700,0",Trecs,1200
3150  ASSIGN @Bunit TO Bunit$&":HP9121,700,0"
3160  DISP "COPYING ";"UNITFILE TO ";Bunit$;"...";
3170  FOR Recno=Sunit TO Eunit
3180    ENTER @Punit,Recno;N(*)
3190    OUTPUT @Bunit;N(*)
3200  NEXT Recno
3210  DISP "COMPLETE"
3220  WAIT 3
3230  RETURN
3240 Restunit:  !
3250  DISP "PURGE UNITFILE FROM HARDISK (Y/N) ";
3260  INPUT Purg$
3270  IF Purg$="Y" THEN PURGE "UNITFILE"&Disk$
3280  DISP "CREATING UNITFILE..."
3290  CREATE BDAT "UNITFILE"&Disk$,400,1200
3300  ASSIGN @Punit TO "UNITFILE"&Disk$
3310  Eunit=191
3320  Iname$="BLUE"
3330  GOSUB Runit
3340  Eunit=209
3350  Iname$="RED"
3360  GOSUB Runit
3370  RETURN
```

Table 2-2. Game initialization code.

```
3380 Runit:      !
3390  DISP "INSERT ";Iname$;" UNITFILE DISK IN FLOPPY DRIVE 0 - PRESS CONT WHEN
READY"
3400  PAUSE
3410  Runit$="UNITFILE"&Iname$[1,1]
3420  DISP "COPYING ";Runit$;" TO UNITFILE...";
3430  ASSIGN @Runit TO Runit$&":HP9121,700,0"
3440  FOR Recno=1 TO Eunit
3450    ENTER @Runit,Recno;N(*)
3460    OUTPUT @Punit;N(*)
3470  NEXT Recno
3480  DISP "COMPLETE"
3490  WAIT 3
3500  RETURN
3510  !-----------------------------------------------------------------------
3520 Cha_single_valu:!
3530  PRINT USING "@"
3540  PRINT "CHANGE SINGLE VALUES IN THE UNITFILE"
3550  INPUT "ENTER UNIT NUMBER 1-400 (999=STOP)",I
3560  IF I=999 THEN 3670
3570  IF I<1 OR I>400 THEN 3550
3580  ENTER @Punit,I;N(*)
3590  INPUT "ENTER ITEM NUMBER, NEW VALUE    (999,999=STOP)",J,NO
3600  IF J=999 THEN 3640
3610  IF J<1 OR J>150 THEN 3590
3620  N(J)=NO
3630  GOTO 3590
3640  GOSUB Calc_combat_eff
3650  OUTPUT @Punit,I;N(*)
3660  GOTO 3530
3670  RETURN
3680  !-----------------------------------------------------------------------
3690 Build_toefile: !
3700  PRINT USING "@"
3710  PRINT "TOEFILE BUILDER"
3720  INPUT "DESIRED ACTION: 1-BUILD UNIT 2-DISPLAY UNIT 3-RETURN TO MENU",Z1
3730  ON Z1 GOTO 3740,3900,690
3740  INPUT "ENTER UNIT NUMBER (999=STOP)",I
3750  IF I=999 THEN 3700
3760  IF I<1 OR I>400 THEN 3740
3770  INPUT "SIDE (1-BLUE 2-RED)",K
3780  INPUT "UNIT TYPE",L
3790  IF L<0 OR L>9 THEN 3780
3800  INPUT "ENTER NUMBER OF SYSTEMS 1-70",T(*)
3810  E=0
3820  FOR J=1 TO 70
3830    U(J)=T(J)
3840    E=E+U(J)*S(K,J)
3850  NEXT J
3860  U(71)=E
3870  U(72)=K+L/10
3880  OUTPUT @Ptoe,I;U(*)
```

Table 2-2. Game initialization code.

```
3890  GOTO 3740
3900  INPUT "1-SINGLE UNIT     2-ALL UNITS       999-STOP",Ch
3910  IF Ch=999 THEN GOTO 3720
3920  ON Ch GOSUB Sing,All
3930  GOTO 3900
3940 Sing:  !
3950  INPUT "ENTER UNIT NUMBER (999=STOP)",I
3960  IF I=999 THEN 4010
3970  IF I<1 OR I>400 THEN 3900
3980  ENTER @Ptoe,I;U(*)
3990  GOSUB Print_toe
4000  GOTO 3940
4010  RETURN
4020 All: !
4030  FOR I=1 TO 400
4040    ENTER @Ptoe,I;U(*)
4050    GOSUB Print_toe
4060  NEXT I
4070  RETURN
4080 Print_toe:  !
4090  PRINTER IS 702
4100  PRINT
4110  PRINT
4120  PRINT "UNIT ";I
4130  PRINT USING "7(10(5D.1D,1X),/),7D.1D,7X,7D.1D";U(*)
4140  PRINTER IS 1
4150  RETURN
4160  !---------------------------------------------------------------------------
4170 Init_unit_stat: !
4180  PRINT USING "@"
4190  PRINT "UNIT STATUS INITIALIZATION"
4200  INPUT "ENTER UNIT NUMBER 1-400 (999=STOP)",I
4210  GOSUB Clear_out
4220  IF I=999 THEN 4710
4230  IF I<1 OR I>400 THEN 4200
4240  INPUT "ENTER UNIT NAME (16 CHARACTER MAXIMUM)",M$
4250  OUTPUT @Pname,I;M$
4260  INPUT "SIDE (1-BLUE 2-RED)",K
4270  INPUT "UNIT TYPE",L
4280  K=INT(K)
4290  L=INT(L)
4300  IF L<0 OR L>9 THEN 4270
4310  INPUT "ENTER NUMBER OF SYSTEMS 1-70",T(*)
4320  E=0
4330  FOR J=1 TO 70
4340    U(J)=T(J)
4350    E=E+U(J)*S(K,J)
4360  NEXT J
4370  U(71)=E
4380  U(72)=K+L/10
4390  OUTPUT @Ptoe,I;U(*)
4400  INPUT "ENTER LINE 1 (1-10)",N(1),N(2),N(3),N(4),N(5),N(6),N(7),N(8),N(9),N
(10)
```

Table 2-2. Game initialization code.

```
4410  INPUT "ENTER LINE 2 (11-20)",N(11),N(12),N(13),N(14),N(15),N(16),N(17),N
8),N(19),N(20)
4420  INPUT "ENTER LINE 3 (21-30)",N(21),N(22),N(23),N(24),N(25),N(26),N(27),N
8),N(29),N(30)
4430  INPUT "ENTER LINE 4 (31-40)",N(31),N(32),N(33),N(34),N(35),N(36),N(37),N
8),N(39),N(40)
4440  INPUT "ENTER LINE 5 (41-50)",N(41),N(42),N(43),N(44),N(45),N(46),N(47),N
8),N(49),N(50)
4450  INPUT "ENTER LINE 6 (51-60)",N(51),N(52),N(53),N(54),N(55),N(56),N(57),N
8),N(59),N(60)
4460  INPUT "ENTER LINE 7 (61-70)",N(61),N(62),N(63),N(64),N(65),N(66),N(67),N
8),N(69),N(70)
4470  INPUT "ENTER LINE 8 (75,89,77,78,90,101,103,105; 90&78 IN 2 PARTS)",N(75)
A1,A2,N(77),B1,B2,N(90),N(101),N(103),N(105)
4480  IF N(75)<1 OR N(75)>4 THEN 4470
4490  IF A1<1 OR A1>5 THEN 4470
4500  IF A2<0 OR A2>9 THEN 4470
4510  N(89)=INT(A1)+A2/10
4520  IF N(77)<>1 AND N(77)<>2 THEN 4470
4530  IF B1<>1 AND B1<>2 THEN 4470
4540  IF B2<0 OR B2>9 THEN 4470
4550  N(78)=INT(B1)+B2/10
4560  IF N(90)<0 OR N(90)>1 THEN 4470
4570  IF N(101)<0 OR N(101)>1 THEN 4470
4580  INPUT "ENTER LINE 9 (119-125,81-83,76)",N(119),N(120),N(121),N(122),N(12
,N(124),N(125),N(81),N(82),N(83),N(76)
4590  IF N(119)<0 OR N(119)>1 THEN 4580
4600  IF N(120)<0 OR N(120)>1 THEN 4580
4610  IF N(121)<0 OR N(121)>1 THEN 4580
4620  IF N(81)<1 OR N(81)>400 THEN 4580
4630  IF N(82)<>0 AND N(82)<>1 THEN 4580
4640  IF N(83)<0 OR N(83)>9 THEN 4580
4650  IF N(76)<>0 AND N(76)<>1 THEN 4580
4660  N(139)=0
4670  N(140)=0
4680  N(141)=0
4690  GOSUB Unitfile_disk
4700  GOTO 4200
4710  RETURN
4720 Clear_out:!
4730  FOR Ikp=1 TO 150
4740    N(Ikp)=0
4750  NEXT Ikp
4760  RETURN
4770 !-------------------------------------------------------------------------
4780 Init_stat:!
4790  PRINT USING "@"
4800  PRINT "                    UNIT STATUS INITIALIZATION"
4810  INPUT "ENTER UNIT NUMBER 1-400 (999=STOP)",I
4820  IF I=999 THEN RETURN
4830  IF I<1 OR I>400 THEN 4810
4840  INPUT "SIDE (1=BLUE, 2=RED)",K
```

Table 2-2. Game initialization code.

```
4850  INPUT "UNIT TYPE",L
4860  K=INT(K)
4870  L=INT(L)
4880  IF L<0 OR L>9 THEN 4850
4890  INPUT "ENTER UNIT NAME (16 CHARACTER MAXIMUM)",M$
4900  OUTPUT @Pname,I;M$
4910  DISP "ENTER THE NUMBER OF SYSTEMS 1-70"
4920  FOR G=1 TO 7
4930    PRINT "LINE";G;"   ";
4940    FOR J=1 TO 10
4950      INPUT "INPUT FIELD VALUE",T((G-1)*10+J)
4960      PRINT T((G-1)*10+J);
4970    NEXT J
4980    PRINT
4990  NEXT G
5000  E=0
5010  FOR J=1 TO 70
5020    U(J)=T(J)
5030    E=E+U(J)*S(K,J)
5040    N(J)=T(J)
5050  NEXT J
5060  U(71)=E
5070  U(72)=K+L/10
5080  OUTPUT @Ptoe,I;U(*)
5090  N(89)=5.9
5100  N(77)=1
5110  N(78)=K+L/10
5120  N(90)=1
5130  PRINT USING "@"
5140  !PRINT "DOES THIS UNIT HAVE ITS NORMAL FUEL SUPPLY?"
5150  !INPUT "ANSWER <Y>ES IF IT DOES, <N>O IF IT HAS EITHER MORE OR LESS",A$
5160  !IF A$="Y" THEN
5170  !    N(101)=1
5180  !    IF K=1 THEN
5190  !       N(103)=N(55)*1800
5200  !       ELSE
5210  !       N(103)=N(55)*1288
5220  !    END IF
5230  !    ELSE
5240  INPUT "WHAT PERCENT OF NORMAL FUEL DOES THIS UNIT HAVE",X
5250  Percent=X/100
5260  IF Percent>1 THEN
5270    N(101)=1
5280  ELSE
5290    N(101)=Percent
5300  END IF
5310  IF K=1 THEN
5320    IF Percent>1 THEN
5330      N(103)=N(55)*1800
5340      N(105)=(N(55)*1800*Percent)-N(103)
5350    ELSE
5360      N(103)=N(55)*1800*Percent
```

Table 2-2. Game initialization code.

```
5370        N(105)=0
5380      END IF
5390  ELSE
5400      IF Percent>1 THEN
5410        N(103)=N(55)*1288
5420        N(105)=(N(55)*1288*Percent)-N(103)
5430      ELSE
5440        N(103)=N(55)*1288*Percent
5450        N(105)=0
5460      END IF
5470  END IF
5480 ! END IF
5490  PRINT USING "@"
5500 ! PRINT "DOES THIS UNIT HAVE ITS NORMAL AMMUNITION SUPPLY?"
5510 ! INPUT "ANSWER <Y>ES IF IT DOES, <N>O IF IT HAS EITHER MORE OR LESS AMMUNI
TION",A$
5520 ! IF A$="Y" THEN
5530 !     N(119)=1
5540 !     N(120)=1
5550 !     N(121)=1
5560 !     IF K=1 THEN
5570 !         N(123)=N(58)*6.5
5580 !         ELSE
5590 !         N(123)=N(58)*4.5
5600 !     END IF
5610 !     N(125)=0
5620 !     ELSE
5630  INPUT "WHAT PERCENT OF ITS NORMAL AMMUNITION DOES THIS UNIT HAVE",X
5640  Percent=X/100
5650  IF Percent>1 THEN
5660    N(119)=1
5670    N(120)=1
5680    N(121)=1
5690    IF K=1 THEN
5700      N(123)=N(58)*6.5
5710      N(125)=(N(58)*6.5*Percent)-N(123)
5720    ELSE
5730      N(123)=N(58)*4.5
5740      N(125)=(N(58)*4.5*Percent)-N(123)
5750    END IF
5760  ELSE
5770    N(119)=Percent
5780    N(120)=Percent
5790    N(121)=Percent
5800    IF K=1 THEN
5810      N(123)=N(58)*6.5*Percent
5820      N(125)=0
5830    ELSE
5840      N(123)=N(58)*4.5*Percent
5850      N(125)=0
5860    END IF
5870  END IF
```

Table 2-2. Game initialization code.

```
5880 ! END IF
5890  PRINT USING "@"
5900  INPUT "WHAT IS THE PERCENTAGE OF DIRECT FIRE AMMUNITION".X
5910  INPUT "WHAT IS THE PERCENTAGE OF INDIRECT FIRE AMMUNITION",Y
5920  INPUT "WHAT IS THE PERCENTAGE OF AIR DEFENSE AMMUNITION",Z
5930  IF X+Y+Z<>100 THEN
5940    PRINT USING "@,#"
5950    PRINT "PERCENTAGES OF AMMUNITION DO NOT ADD UP TO 100%.  TRY IT AGAIN."
5960    WAIT 3
5970    GOTO 5890
5980  END IF
5990  N(124)=X+Y/100   !UNITS WERE CREATED WITH FORMULA.  IT SHOULD BE
        N(124)=(X*10)+(Y/100).  NEXT LINE ALSO
6000  N(126)=X+Y/100
6010  !INPUT "WHAT IS THE SUPPORTING ADA UNIT",N(81)
6020  INPUT "IS THIS UNIT ACTIVE <1> OR INACTIVE <0>",N(82)
6030  INPUT "WHAT IS THIS UNIT'S TACTICAL MISSION",X
6040 ! INPUT "WHAT IS THIS UNIT'S TACTICAL MISSION FOR THE 2ND 3 HOURS",Y
6050  X=INT(X)
6060  Y=INT(X)
6070  N(83)=X+Y/10
6080  IF K=1 THEN
6090    INPUT "IS THIS UNIT A BATTALION (0) OR COMPANY (1) ",Esch
6100  ELSE
6110    INPUT "IS THIS UNIT A REGIMENT (0) OR BATTALION (1) ",Esch
6120  END IF
6130  IF Esch<>0 AND Esch<>1 THEN
6140    DISP "UNIT ECHELON MUST BE ""0"" OR ""1"" "
6150    WAIT 3
6160    GOTO 6080
6170  ELSE
6180    N(76)=Esch
6190  END IF
6200  GOSUB Unitfile_disk
6210  GOTO 4780
6220  RETURN
6230 Transfer:! THIS SUBROUTINE TRANSFERS EQUIPMENT BETWEEN UNITS
6240 Start_input:PRINT USING "@,#,23A";"UNIT EQUIPMENT TRANSFER"
6250  INPUT "ENTER UNIT # LOSSING ELEMENT AND UNIT # GAINING ELEMENT -- STOP(999
,999)",L,G
6260  IF L=999 AND G=999 THEN Transend
6270  IF L<1 OR L>400 OR G<1 OR G>400 THEN 6250
6280  IF L<192 THEN
6290    Side_l=1
6300  ELSE
6310    Side_l=2
6320  END IF
6330  IF G<192 THEN
6340    Side_g=1
6350  ELSE
6360    Side_g=2
6370  END IF
```

Table 2-2. Game initialization code.

```
6380  IF Side_l<>Side_g THEN
6390    DISP "UNITS NOT ON SAME SIDE, CHECK YOUR DATA, PRESS CONTINUE TO PROCEE!
"
6400    PAUSE
6410    GOTO 6250
6420  ELSE
6430    Side=Side_l
6440  END IF
6450 Element_input:  !
6460  FOR I=1 TO 70
6470    Sys(I)=0
6480    Toe(I)=0
6490  NEXT I
6500   !
6510  REM - ADJUST TOEFILE FOR LOSING AND GAINING UNITS
6520  ENTER @Ptoe,G;Ugain(*)
6530  ENTER @Ptoe,L;U(*)
6540  ENTER @Punit,G;Ngain(*)
6550  ENTER @Punit,L;N(*)
6560  INPUT "ENTER ELEMENT NUMBER TO BE TRANSFERED",Elmt_no_trans
6570  INPUT "NUMBER OF ELEMENTS TO BE TRANSFERED",Sys(Elmt_no_trans)
6580  Eff_gain=0
6590  Eff_lose=0
6600  Int_sys=N(Elmt_no_trans)-Sys(Elmt_no_trans)
6610  IF Int_sys<0 THEN
6620    PRINT USING "@"
6630    PRINT "ELEMENT ";Elmt_no_trans;" IS NEGATIVE FOR LOSING UNIT, CHECK YOUF
 DATA, PRESS CONTINUE TO PROCEED"
6640    PAUSE
6650    GOTO Start_input
6660  END IF
6670  Int_sys=U(Elmt_no_trans)-Sys(Elmt_no_trans)
6680  IF Int_sys<0 THEN
6690    PRINT USING "@"
6700    PRINT "TOE ITEM ";Elmt_no_trans;" IS NEGATIVE FOR LOSING UNIT. CHECK YOU
R DATA, PRESS CONTINUE TO PROCEED"
6710    PAUSE
6720    GOTO Start_input
6730  ELSE
6740    Ugain(Elmt_no_trans)=Ugain(Elmt_no_trans)+Sys(Elmt_no_trans)
6750    U(Elmt_no_trans)=Int_sys
6760    Eff_lose=Sys(Elmt_no_trans)*S(Side,Elmt_no_trans)
6770  END IF
6780  Ugain(71)=Eff_lose+Ugain(71)
6790  U(71)=U(71)-Eff_lose
6800  !
6810  IF U(71)=0 OR Ugain(71)=0 THEN Ni
6820  Eff_u=0
6830  Eff_ug=0
6840  FOR Le=1 TO 70
6850    Eff_u=Eff_u+S(Side,Le)*N(Le)
6860    Eff_ug=Eff_ug+S(Side,Le)*Ngain(Le)
```

Table 2-2. Game initialization code.

```
6870  NEXT Le
6880  N(79)=Eff_u/U(71)
6890  Ngain(79)=Eff_ug/Ugain(71)
6900 Ni:  '
6910      !
6920  ! UPDATE UNITFILE STATUS
6930  REM - ADJUST CARGO/FUEL TRUCK LEVELS
6940  IF Sys(55)=0 THEN Adjust_cargo
6950  Avg_fuel=N(103)/N(55)
6960  Tot_avg_fuel=Avg_fuel*Sys(55)
6970  N(103)=N(103)-Tot_avg_fuel
6980  Ngain(103)=Ngain(103)+Tot_avg_fuel
6990  IF N(103)<0 THEN N(103)=0
7000 Adjust_cargo:! ADJUST CARGO LEVELS
7010  IF Sys(58)=0 THEN Tally_systems
7020  Gain_df_frac=INT(Ngain(124))/1000
7030  Lose_df_frac=INT(N(124))/1000
7040  Gain_if_frac=Ngain(124)-INT(Ngain(124))
7050  Lose_if_frac=N(124)-INT(N(124))
7060  Gain_ad_frac=1-(Gain_df_frac+Gain_if_frac)
7070  Lose_ad_frac=1-(Lose_df_frac+Lose_if_frac)
7080  Avg_ammo_lose=N(123)/N(58)
7090  Ammo_transfer=Avg_ammo_lose*Sys(58)
7100  N(123)=N(123)-Ammo_transfer
7110  Df_ammo_trans=Ammo_transfer*Lose_df_frac
7120  If_ammo_trans=Ammo_transfer*Lose_if_frac
7130  Ad_ammo_trans=Ammo_transfer*Lose_ad_frac
7140  Df_ammo=0
7150  If_ammo=0
7160  Ad_ammo=0
7170  Df_ammo=Df_ammo_trans+Gain_df_frac*Ngain(123)
7180  If_ammo=If_ammo_trans+Gain_if_frac*Ngain(123)
7190  Ad_ammo=Ad_ammo_trans+Gain_ad_frac*Ngain(123)
7200  Tot_ammo=Df_ammo+If_ammo+Ad_ammo
7210  Ngain(123)=Tot_ammo
7220  N1=(Df_ammo/Tot_ammo)*1000
7230  N2=If_ammo/Tot_ammo
7240  Ngain(124)=INT(N1)+N2
7250 Tally_systems:! ADJUST VEHICLE AMMO LEVELS AND TRANSFER SYSTEMS
7260  Tot_fuel_lose=0
7270  Tot_fuel_gain=0
7280  Fuel_gain=0
7290  Fuel_lose=0
7300  Fuel_sys=0
7310  FOR I=1 TO 3
7320    Ammo_gain(I)=0
7330    Ammo_lose(I)=0
7340    Ammo_sys(I)=0
7350    Tal(I)=0
7360    Tag(I)=0
7370  NEXT I
7380  REM - TALLY FUEL/AMMO
```

Table 2-2. Game initialization code.

```
7390  FOR I=1 TO 70
7400    Fuel_gain=Fuel_gain+Ngain(I)*F(Side,I)*Ngain(101)
7410    Fuel_lose=Fuel_lose+N(I)*F(Side,I)*N(101)
7420    Fuel_sys=Fuel_sys+Sys(I)*F(Side,I)*N(101)
7430    Ammo_gain(W(Side,I))=Ammo_gain(W(Side,I))+A(Side,I)*Ngain(118+W(Side,I))
*Ngain(I)
7440    Ammo_lose(W(Side,I))=Ammo_lose(W(Side,I))+A(Side,I)*N(118+W(Side,I))*N(I
)
7450    Ammo_sys(W(Side,I))=Ammo_sys(W(Side,I))+A(Side,I)*N(118+W(Side,I))*Sys(I
)
7460  NEXT I
7470  Fuel_gain=Fuel_gain+Fuel_sys
7480  Fuel_lose=Fuel_lose-Fuel_sys
7490  FOR I=1 TO 3
7500    Ammo_gain(I)=Ammo_gain(I)+Ammo_sys(I)
7510    Ammo_lose(I)=Ammo_lose(I)-Ammo_sys(I)
7520  NEXT I
7530  FOR I=1 TO 70
7540    Ngain(I)=Ngain(I)+Sys(I)
7550    N(I)=N(I)-Sys(I)
7560    Tag(W(Side,I))=Tag(W(Side,I))+Ngain(I)*A(Side,I)
7570    Tal(W(Side,I))=Tal(W(Side,I))+N(I)*A(Side,I)
7580    Tot_fuel_gain=Tot_fuel_gain+Ngain(I)*F(Side,I)
7590    Tot_fuel_lose=Tot_fuel_lose+N(I)*F(Side,I)
7600  NEXT I
7610  REM - UPDATE UNIT AMMO/FUEL STATUS
7620  FOR I=1 TO 3
7630    IF Tag(I)=0 THEN
7640      Ngain(118+I)=0
7650    ELSE
7660      Ngain(118+I)=Ammo_gain(I)/Tag(I)
7670    END IF
7680    IF Ngain(118+I)>1 THEN Ngain(118+I)=1
7690    IF Tal(I)=0 THEN
7700      N(118+I)=0
7710    ELSE
7720      N(118+I)=Ammo_lose(I)/Tal(I)
7730      IF N(118+I)>1 THEN N(118+I)=1
7740    END IF
7750  NEXT I
7760  IF Tot_fuel_gain=0 THEN
7770    Ngain(101)=0
7780  ELSE
7790    Ngain(101)=Fuel_gain/Tot_fuel_gain
7800  END IF
7810  IF Ngain(101)>1 THEN Ngain(101)=1
7820  IF Tot_fuel_lose=0 THEN
7830    N(101)=0
7840  ELSE
7850    N(101)=Fuel_lose/Tot_fuel_lose
7860  END IF
7870  REM - CALCULATE UNIT EFFECTIVENESS
```

Table 2-2. Game initialization code.

```
7880  Eff_gain=0
7890  Eff_lose=0
7900  FOR I=1 TO 70
7910    Eff_gain=Ngain(I)*S(Side,I)+Eff_gain
7920    Eff_lose=N(I)*S(Side,I)+Eff_lose
7930  NEXT I
7940  IF Ugain(71)=0 THEN
7950    Ngain(79)=0
7960  ELSE
7970    Ngain(79)=Eff_gain/Ugain(71)
7980  END IF
7990  IF U(71)=0 THEN
8000    N(79)=0
8010  ELSE
8020    N(79)=Eff_lose/U(71)
8030  END IF
8040  REM - WRITE TO FILE
8050  OUTPUT @Punit,G;Ngain(*)
8060  OUTPUT @Punit,L;N(*)
8070  OUTPUT @Ptoe,G;Ugain(*)
8080  OUTPUT @Ptoe,L;U(*)
8090  INPUT "ANY MORE ELEMENTS TO TRANSFER? (Y OR N)",Q$
8100  IF Q$="N" THEN 8120
8110  IF Q$="Y" THEN Element_input
8120  INPUT "ANY MORE UNITS TO TRANSFER? (Y OR N)",Q$
8130  IF Q$<>"Y" AND Q$<>"N" THEN 8120
8140  IF Q$="Y" THEN Start_input
8150 Transend:  !
8160  RETURN
8170  !------------------------------------------------------------------------
8180 Game_turn:!
8190 !PRINT USING "@"
8200 !PRINTER IS 702
8210 !PRINT USING "@"
8220  PRINT "********** GAME TURN RESUPPLY ************"
8230  PRINTER IS 1
8240  PRINT "ENTER UNIT FILE CHANGES FOR GAME TURN"
8250  REPEAT
8260    PRINT USING "//"
8270    REPEAT
8280      INPUT "ENTER UNIT NUMBER (1-400, 999=STOP)",I
8290    UNTIL I<=400 AND I>=1 OR I=999
8300    IF I=999 THEN GOTO The_end
8310    PRINT "UNIT ";I
8320    PRINT
8330    PRINT
8340    ENTER @Punit,I;N(*)
8350     !
8360    REPEAT
8370      REPEAT
8380        INPUT "ENTER LINE # (1-6, 999=NEXT UNIT) ",L
8390      UNTIL L=1 OR L=2 OR L=3 OR L=4 OR L=5 OR L=6 OR L=999
```

Table 2-2. Game initialization code.

```
8400      IF L=999 THEN GOTO End_lines
8410      ON L GOSUB Line1,Line2,Line3,Line4,Line5,Line6
8420 End_lines: !
8430    UNTIL L=999
8440    OUTPUT @Punit,I;N(*)
8450  UNTIL I=999
8460 The_end:!
8470  RETURN
8480 Line1:!
8490  PRINT "    LINE 1:"
8500  INPUT "ENTER: ACTIVITY, MOPP LEVEL, MISSION, KV MOVED",N(82),N(77),N(83),N
(146)
8510  CALL Ck_var("ACTIVITY","OR",N(82),0,2)
8520  CALL Ck_var("MOPP LEVEL","OR",N(77),1,2)
8530  GOSUB Unitfile_disk
8540  PRINT "               ",N(82),N(77),N(83),N(146)
8550  PRINTER IS 1
8560  RETURN
8570 Line2:!
8580  PRINT "    LINE 2:"
8590  INPUT "ENTER: SENSOR GP, ZONE, PCT COVERED",N89y,N(89),N(90)
8600  CALL Ck_var("SENSOR GROUP","TO",N89y,0,9)
8610  CALL Ck_var("ZONE","TO",N(89),1,5)
8620  CALL Ck_var("PCT COVERED","THRU",N(90),0,100)
8630  PRINT "               ",N89y,N(89),N(90)
8640  PRINTER IS 1
8650  N(89)=(INT(N(89)))+(N89y/10)
8660  N(90)=N(90)/100
8670  RETURN
8680 Line3:!
8690  PRINT "    LINE 3:"
8700  INPUT "ENTER PCT ADA SUPPRESSION DATA: Vehicle, Hand held, Corps ada",N(8
),N80y,Xx
8710  IF Xx=0 THEN
8720    N(81)=N(81)
8730  ELSE
8740    N(81)=Xx
8750  END IF
8760  CALL Ck_var("Vehicle","TO",N(80),0,99)
8770  CALL Ck_var("Hand held","TO",N80y,0,99)
8780  PRINT "               ",N(80),N80y,N(81)
8790  PRINTER IS 1
8800  N(80)=(INT(N(80)))+(N80y/100)
8810  RETURN
8820 Line4:!
8830  PRINT "    LINE 4:"
8840  INPUT "ENTER RESUPPLY DATA: DF(Tons),IF(Tons),AD(Tons), FUEL(Gal)",N1,N2,N
3,N(110)
8850  N(135)=N1+N2+N3
8860  IF (N1+N2+N3)=0 THEN
8870    Ndf=0
8880    Nif=0
```

Table 2-2. Game initialization code.

```
8890     N(136)=0
8900   END IF
8910   PRINT "             "," DF"," IF"," ADA","TOTAL","FUEL"
8920   PRINT "             ",N1,N2,N3,N(135),N(110)
8930   IF N(135)<=0 THEN 9050
8940   !FOR J=1 TO 3
8950     !IF N(J)=0 THEN
8960       !GOTO 4467
8970     !ELSE
8980       !N(J)=100*N(J)/N(135)
8990     !END IF
9000   !NEXT J
9010   Ndf=N1/(N1+N2+N3)
9020   Nif=N2/(N1+N2+N3)
9030   IF Nif=1 THEN Nif=.999
9040   N(136)=(INT(Ndf*1000))+Nif
9050   PRINTER IS 1
9060   RETURN
9070 Line5:!
9080   PRINT "    LINE 5:"
9090   INPUT "ENTER DISPENSED DATA: DF(Tons),IF(Tons),AD(Tons), FUEL(Gal)",N1,N2,
N3,N(112)
9100   N(137)=N1+N2+N3
9110   IF N(137)=0 THEN
9120     Ndf=0
9130     Nif=0
9140     N(138)=0
9150   END IF
9160   IF N(137)<=0 THEN 9300
9170   PRINT "             "," DF"," IF"," ADA","TOTAL","FUEL"
9180   PRINT "             ",N1,N2,N3,N(137),N(112)
9190   !FOR J=1 TO 3
9200     !IF N(J)=0 THEN
9210       !GOTO 4492
9220     !ELSE
9230       !N(J)=100*N(J)/N(137)
9240     !END IF
9250   !NEXT J
9260   Ndf=N1/(N1+N2+N3)
9270   Nif=N2/(N1+N2+N3)
9280   IF Nif=1 THEN Nif=.999
9290   N(138)=(INT(Ndf*1000))+Nif
9300   PRINTER IS 1
9310   RETURN
9320 Line6: !
9330   IF I>191 THEN PRINT "    LINE 6: GR RADAR, CB RADAR, LRRP, SLAR, RPV, FO"
9340   IF I<192 THEN PRINT "   LINE 6: GR RADAR, CB RADAR, LRRP, RPV, SLAR, FO"
9350   INPUT N(95),N(96),N(97),N(98),N(99),N(100)
9360   PRINT "   LINE 6: ";N(95);N(96);N(97);N(98);N(99);N(100)
9370   PRINTER IS 1
9380   RETURN
9390   !-----------------------------------------------------------------------
```

Table 2-2. Game initialization code.

```
9400 Calc_combat_eff:!
9410  ENTER @Ptoe,I;U(*)
9420  U0=0
9430  K=INT(N(78))
9440  IF K=0 THEN 9490
9450  FOR J=1 TO 70
9460    U0=U0+N(J)*S(K,J)
9470  NEXT J
9480  N(79)=U0/U(71)
9490  GOSUB Unitfile_disk
9500  RETURN
9510  !----------------------------------------------------------------------
9520 Unitfile_disk:!
9530  N1=INT(N(83))
9540  IF N(78)=0 THEN 9760
9550  !SET MAJOR MISSION
9560  SELECT N1
9570  CASE 0
9580    N(75)=4
9590  CASE 1
9600    N(75)=1
9610  CASE 2
9620    N(75)=4
9630  CASE 3 TO 4
9640    N(75)=1
9650  CASE 5 TO 7
9660    N(75)=2
9670  CASE 8
9680    N(75)=3
9690  CASE 9
9700    N(75)=4
9710  END SELECT
9720  N(127)=N(83)
9730  N(128)=N(83)
9740  N(129)=N(83)
9750  N(107)=N(83)
9760  OUTPUT @Punit,I;N(*)
9770  RETURN
9780  !**********************************************************************
9790 Zero_unit:!
9800  M$="                  "
9810  IF Choice$="B" THEN
9820    INPUT "ZERO OUT WHICH UNIT?",Unit
9830    IF Unit<1 OR Unit>400 THEN 9820
9840    FOR K=1 TO 150
9850      N(K)=0
9860    NEXT K
9870    OUTPUT @Punit,Unit;N(*)
9880    OUTPUT @Ptoe;N(*)
9890    OUTPUT @Fname,Unit;M$
9900    RETURN
9910  END IF
```

Table 2-2. Game initialization code.

```
9920  IF Choice$="C" THEN
9930    FOR Unit=1 TO 191
9940      FOR K=1 TO 150
9950        N(K)=0
9960      NEXT K
9970      OUTPUT @Punit,Unit;N(*)
9980      OUTPUT @Ptoe;N(*)
9990      OUTPUT @Pname,Unit;M$
10000   NEXT Unit
10010   RETURN
10020 END IF
10030 IF Choice$="D" THEN
10040   FOR Unit=192 TO 400
10050     FOR K=1 TO 150
10060       N(K)=0
10070     NEXT K
10080     OUTPUT @Punit,Unit;N(*)
10090     OUTPUT @Ptoe;N(*)
10100     OUTPUT @Pname,Unit;M$
10110   NEXT Unit
10120   RETURN
10130 END IF
10140 IF Choice$="E" THEN
10150   FOR Unit=1 TO 400
10160     FOR K=1 TO 150
10170       N(K)=0
10180     NEXT K
10190     OUTPUT @Punit,Unit;N(*)
10200     OUTPUT @Ptoe;N(*)
10210     OUTPUT @Pname,Unit;M$
10220   NEXT Unit
10230   RETURN
10240 END IF
10250 IF Choice$="F" THEN
10260   PRINT "ZERO OUT FROM WHICH UNIT TO WHICH UNIT?"
10270   INPUT "(INPUT STARTING UNIT NUMBER,ENDING UNIT NUMBER)",First,Last
10280   IF First<1 OR First>Last THEN
10290     PRINT "ERROR IN STARTING UNIT NUMBER.  TRY AGAIN."
10300     GOTO 10260
10310   END IF
10320   IF Last<First OR Last>400 THEN
10330     PRINT "ERROR IN ENDING UNIT NUMBER.  TRY AGAIN."
10340     GOTO 10260
10350   END IF
10360   FOR Unit=First TO Last
10370     FOR K=1 TO 150
10380       N(K)=0
10390     NEXT K
10400     OUTPUT @Punit,Unit;N(*)
10410     OUTPUT @Ptoe;N(*)
10420     OUTPUT @Pname,Unit;M$
10430   NEXT Unit
```

Table 2-2. Game initialization code.

```
10440   RETURN
10450 END IF
10460 RETURN
10470 !-----------------------------------------------------------------------
10480 Subprogram_end: !
10490 LOAD "DIME"&Disk$
10500 END
10510 !*************************************************************************
10520 SUB Ck_var(Var_name$,T$,Variable,Min_value,Max_value)
10530   SELECT T$
10540   CASE "THRU"
10550     WHILE Variable<Min_value OR Variable>Max_value
10560       GOSUB Print_error
10570     END WHILE
10580   CASE "OR"
10590     GOTO Case_to
10600   CASE "TO"
10610 Case_to:FOR M=Min_value TO Max_value
10620       IF Variable=M THEN GOTO End_select
10630     NEXT M
10640     GOSUB Print_error
10650     GOTO Case_to
10660 End_select:!
10670   END SELECT
10680   GOTO Rtrn
10690 Print_error:    !
10700   PRINT
10710   PRINT "** ERROR: ";Variable;" IS INVALID FOR ";Var_name$
10720   PRINT "INPUT: ";Min_value;" ";T$;" ";Max_value;" ONLY"
10730   INPUT Variable
10740   RETURN
10750 Rtrn:!
10760 SUBEND
```

Table 2-2. Game initialization code.

CHAPTER 3

DETECTION

1. PURPOSE.

The DIME detection program attempts to portray the detection of units and the intelligence fusion process necessary to recognize unit type and location.

2. GENERAL.

A. The DIME detection program (P7) is a straightforward adaptation of the DAME detection module discussed in CAORA/TR-5/83, Deep Attack Map Exercise (DAME) Game Rules and Operation Procedures.

B. The detection program consists of a main program and 15 associated data bases.

C. The program requires gamer inputs describing the location of each enemy sensor group covering the targeted unit. Output from the program consists of a list of units which have been detected (simply found but not fully identified), verified (found, identified, and being tracked), or lost (previously verified/detected but now lost). The list represents the intelligence map held by the friendly commander. In essence, it is his view of the battlefield with respect to enemy units and is intended to be used as a basis for his decision to tactically respond to these units. The program provides one list for the Blue commander and one list for the Red commander.

3. DATA FLOW.

A. The detection program uses three types of data inputs: the unit status file ("UNITFILE"), auxiliary, and online data files.

(1) The detection program accesses the "UNITFILE" for the following inputs:

(a) Sensor status (element 89)

(b) Unit type (element 78)

(c) Unit fraction covered by sensor groups (element 90)

(d) Detection status (element 91)

(e) Intelligence status (element 92)

(f) Activity code (element 82)

(2) The auxiliary stored data is generated by running 14 support programs. Table 3-1 indicates these 14 support programs and the resulting files created by executing these programs. A complete discussion of the file structure for these 15 files is covered under paragraph 4 of this chapter. For further information, refer to Volume III of the DIME documentation.

(3) The detection program uses four online data files:

(a) Exposure profiles for individual elements in a unit (Table 3-2)

(b) Red intelligence thresholds (Table 3-3)

(c) Blue intelligence thresholds (Table 3-4)

(d) Unit movement profile (Table 3-5).

B. The detection program combines the inputs listed in paragraph 3A with the three-step methodology discussed under paragraph 5 to provide the following outputs:

(1) A list of units which have been detected, verified, or lost.

(2) An updated "UNITFILE" for the next critical incident (CI).

C. Figure 3-1 indicates the generalized data flow for the detection program.

4. FILE STRUCTURE.

The detection program requires 15 auxiliary files, four online files, and the "UNITFILE". This section will address the file structure for these data files.

A. Auxiliary. The auxiliary files consist of 15 files: REDPOTA, BLUPOT1, RFLEE, BFLEE, BLBDG, RDBDE, RDCOR, RS1TO6, BS1TO6, RTHOLD, BTHOLD, REI, BEI, RSENPRO, and BSENPRO.

Table 3-1. Detection data support programs.

Program Name	Purpose	Resulting File Name
LDBPOT1	Loads the POTA* data for Red sensors detecting Blue units.	BLUPOT1
LDRPOTA	Loads the POTA* data for Blue sensors detecting Red units.	REDPOTA
LBFLEE	Loads the target movement profile for the Blue units.	BFLEE
LRFLEE	Loads the target movement profile for the Red units.	RFLEE
LBLBDG	Loads the sensor data for Blue battalions in contact between 0-10 km.	BLBDG
LRDBDE	Loads the sensor data for Red regiments in contact between 0-10 km.	RDBDE
LRDCOR	Loads the sensor data for Red commanders for units between 0-3 km.	RDCOR
LBS1T06	Loads the detection probabilities for six Blue sensors against 5 signature types.	BS1T06
LRS1T06	Loads the detection probabilities for six Red sensors against 5 signature types.	RS1T06
LBTHOLD	Loads the Blue intelligence threshold percentages developed by DAME.	BTHOLD
LRTHOLD	Loads the Red intelligence threshold percentages developed by DAME.	RTHOLD
LBEI	Loads the Blue sensor profile data developed by DAME.	BEI
LREI	Loads the Red sensor profile data developed by DAME.	REI
GROUPBLD	Loads the Blue and Red sensor groups.	BSENPRO RSENPRO

*Probability of operational target acquisition.

Table 3-2. Fractional target exposure criteria. (Target fractions which are exposed in various mission postures.)

	Target element categories				
Mission	Personnel	Vehicles	Tanks/APC	Artillery	Rockets
Attack	.40	.75	.75	.75	.75
Defend	.10	.30	.30	.60	.50
Reserve	.30	.40	.40	.60	.50
Move	.10	.65	.65	.60	.50

Table 3-3. Red intelligence thresholds. (fraction of exposed Blue elements which must be detected to correctly associate elements with Blue unit types.)

	Target element categories				
Unit Type	Personnel	Vehicles	Tanks/APC	Artillery	Rockets
Combat	.20	.40	.50	.50	.40
Artillery	.20	.40	.50	.50	.40
ADA	.20	.40	.60	.50	.40
AH Gnd/FARP	.20	.40	.20	.20	.40
CP/HQ	.20	.30	.40	.20	.40
Engineer	.20	.40	.60	.20	.40
POL/Ammo	.20	.45	.80	.20	.40
Maintenance	.20	.50	.40	.20	.40
SAM	.20	.30	.20	.30	.40
Radar/EW	.20	.40	.10	0	.40

Table 3-4. Blue intelligence thresholds. (Fraction of exposed Red elements which must be detected to correctly associate with Red unit types.)

	Target element categories				
Unit Type	Personnel	Vehicles	Tanks/APC	Artillery	Rockets
Combat	.20	.25	.35	.30	.40
Artillery	.20	.25	.35	.35	.40
ADA	.20	.30	.35	.30	.40
AH Gnd/FARP	.20	.35	.20	.20	.40
CP/HQ	.20	.30	.40	.20	.40
Engineer	.20	.25	.40	.20	.40
POL/Ammo	.20	.40	.40	.20	.40
Maintenance	.20	.40	.40	.20	.40
SAM	.20	.30	.20	.30	.40
Radar/EW	.20	.40	.10	0	.40

Table 3-5. Time intervals between unit movements to avoid detection

Unit type	Red stationary time (hours)	Blue stationary time (hours)
Combat	2	2
Artillery	3	4
ADA	3	3
AH Gnd/FARP	3	3
CP/HQ	2	6
Engineer	4	6
POL/Ammo	12	12
Maintenance	24	24
SAM	24	24
Radar/EW	3	6

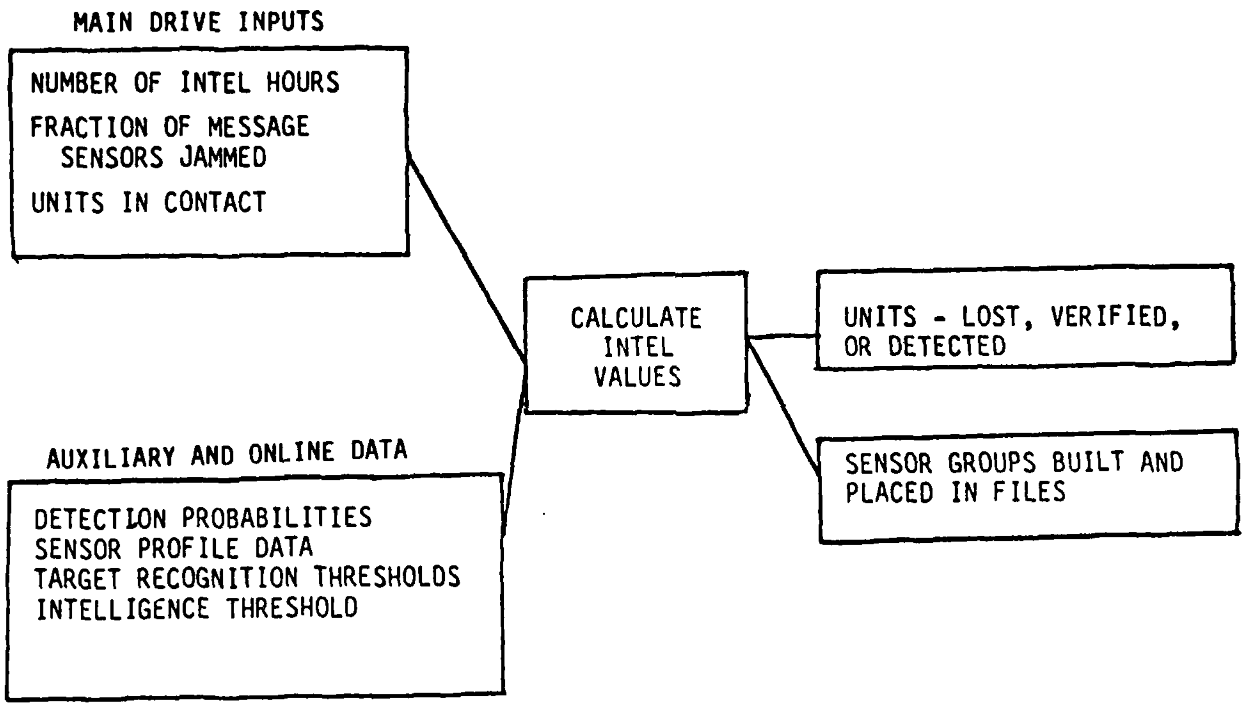


Figure 3-1. Detection data flow.

(1) REDPOTA. A file containing 10 records representing the probability of operational target acquisition (POTA) for the Blue sensors detecting the Red targets. Each record contains the probability of detection as a function of 10 targets/units and five zones.

(a) Targets/units.

1. Combat.

2. Artillery.

3. Air defense (ADA).

4. Attack helicopter ground forward arming and refueling point (FARP).

5. Command post/headquarters (CP/HQ).

6. Engineer.

7. Fuel/ammunition (POL/AMMO) supply point.

8. Maintenance point.

9. Surface-to-air missile (SAM) site.

10. Communication/radar/electronic warfare (EW) site.

(b) Target surveillance zones.

1. Zone I. 0–3 km beyond Forward Edge of the Battle Area.

2. Zone II. 3–12 km beyond FEBA.

3. Zone III. 12–25 km beyond FEBA.

4. Zone IV. 25–100 km beyond FEBA.

5. Zone V. 100–200 km beyond FEBA.

(2) BLUPOT1. Same as REDPOTA except this file contains the probability of operational target acquisition (POTA) of the Red sensors detecting the Blue targets.

(3) RFLEE. A file containing the stationary profile time for the 10 targets listed under REDPOTA above. This file contains one record with 10 entries. Each entry represents the time in hours a Red target remains stationary until it moves.

(4) BFLEE. Same as RFLEE except this file contains the stationary profile time for the Blue targets.

(5) BLBDG. A file containing the probability of a Blue reconnaissance party detecting any of the 10 Red targets within 10 kilometers of the Blue battalion.

(6) RDBDE. Same as BLBDG except this file contains the probability a Red reconnaissance party can detect any of the 10 Blue targets when the Red brigade is in contact.

(7) RDCOR. A file containing the probability a Red commander can detect and/or verify 10 Blue units within 0-3 km from Red commander.

(8) RS1TO6. A file containing the probability of detection for six Red sensors detecting five target types. The file consists of 30 records. The records are structured in the following manner where:

Sensor 1 is Small Fred.
Sensor 2 is Big Fred.
Sensor 3 is Forward Observer (FO), day.
Sensor 4 is Long-range reconnaissance patrol (LRRP).
Sensor 5 is Remotely piloted vehicle (RPV)/Drones.
Sensor 6 is Side-looking airborne radar (SLAR).

(a) Record 1 contains the probability that sensor 1 can detect personnel located in zones 1-5 (see REDPOTA above for zones). Each record has five entries.

(b) Record 2 contains the probability that sensor 1 can detect vehicles located in zones 1-5.

(c) Record 3 contains the probability that sensor 1 can detect tanks/armored personnel carriers located in zones 1-5.

(d) Record 4 contains the probability that sensor 1 can detect artillery targets located in zones 1-5.

(e) Record 5 contains the probability that sensor 1 can detect rocket targets located in zones 1-5.

(f) Records 6-10 contains the probability that sensor 2 can detect five targets (personnel, vehicles, tank/armored personnel carriers (APCs), artillery, rockets) located in one of the five zones.

(g) Records 11-15 contains the probability for sensor 3.

(h) Records 16-20 contains the probability for sensor 4.

(i) Records 21-25 contains the probability for sensor 5.

(j) Records 26-30 contains the probability for sensor 6.

(9) BS1T06. Same as RS1T06 except this file contains the probability of detection for six Blue sensors detecting the five target types where:

Sensor 1 is AN/PPS-15.
Sensor 2 is AN/TPQ-37.
Sensor 3 is FO (day).
Sensor 4 is LRRP.
Sensor 5 is target acquisition designation aerial reconnaissance system (TADARS).
Sensor 6 is SLAR (ASARS).

(10) RTHOLD. A file containing the threshold percentage for five Red target types in 10 unit types. The file has 10 records. Each record represents one of 10 unit types (see REDPOTA) against five targets (personnel, vehicle, tank/APC, artillery, rockets). Each record contains five entries representing the percent of target which must be detected before the unit is detected.

(11) BTHOLD. Same as RTHOLD except this file contains the threshold percentage for five Blue target types in 10 unit types.

(12) REI. A file containing four records representing the percent of Red target types available for detection in one of four missions.

(a) Target types.

1. Personnel.

2. Vehicle.

3. Tanks/APC.

4. Artillery.

5. Rockets.

(b) Missions.

1. Attack.

2. Defend.

3. Reserve.

4. Move.

(13) BEI. Same as REI except this file contains the percent of Blue target types available for detection in one of four missions.

(14) RSENPRO. A file containing three records describing three Red group profiles. The record structure is as follows.

(a) Record 1. For sensor group 1, each entry represents:

1. Number of ground radars in group.

2. Number of artillery radars in group.

3. Number of LRRP in group.

4. Number of RPV in group.

5. Number of SLAR in group.

6. Number of FO in group.

(b) Record 2. Same as record 1 for sensor group 2.

(c) Record 3. Same as record 1 for sensor group 3.

(15) BSENPRO. Same as RSENPRO except this file contains the three Blue group profiles as a function of the sensors in the group.

B. Online. The online files consist of four files: exposure profiles, Red intelligence threshold, Blue intelligence threshold, and unit movement profile.

(1) Exposure profiles. The exposure profiles shown in Table 3-2 represents an effort to relate an element exposure profile to each of the five signature types in one of four missions.

(a) The exposure profile data is contained in four records of five elements each, where each record represents one of four missions (attack, defend, reserve, move) and each element represents one of five signature types (personnel, vehicle, tank/APC, artillery, rockets).

(b) The exposure profiles were developed by the DAME project team using military judgment.

(2) Red/Blue intelligence thresholds. The intelligence threshold tables shown in Tables 3-3 and 3-4 represent the fraction of exposed Blue/Red elements which must be detected to correctly associate elements with Blue/Red unit types. The intelligence threshold data is contained in 10 records of five elements each, where each record represents one of 10 unit types (combat, artillery, ADA, FARP site, command post, engineer, POL/AMMO, maintenance, SAM, radar) and each element represents one of five signature types.

(3) Unit movement profile. The unit movement profiles shown in Table 3-5 represent the time intervals between unit movements to avoid detection. The movement profile data consists of 10 records of two elements each, where

each record represents one of 10 unit types and each element represents one of two forces (Blue, Red).

C. "UNITFILE". The "UNITFILE" is a 400-record file containing 150 elements. Records 1-191 represent the Blue units and records 192-400 represent the Red units. The detection program requires six elements on the "UNITFILE."

(1) Sensor status. Element 89 contains the sensor status (X.Y) where:

X = POTA zone values (1-5) for sensor group Y.

Y = Sensor groups (0-4) detecting this particular unit.
- 0 = Not covered
- 1-3 = Applicable Blue/Red sensor group
- 4 = Linear FEBA oriented - sensor array

Default = (1.4).

(2) Unit type. Element 78 contains the unit type (X.Y), where:

X = Player ID
- 1 = Blue
- 2 = Red

Y = Unit type
- 0 = Combat
- 1 = Artillery
- 2 = ADA
- 3 = FARP
- 4 = CP/HQ
- 5 = Engineer
- 6 = POL/AMMO supply
- 7 = Maintenance
- 8 = SAM site
- 9 = Com/radar/EW site.

(3) Unit fraction covered by sensor group. Element 90 contains a real value of 0 - 1.0 indicating the fraction of the unit covered by a sensor group.

(4) Detection status. Element 91 contains the detection status for each unit (X.Y), where:

X = Represents hours left until redetected

Y = Represents the unit status with respect to detection by the opposite commander
- 0 = Not detected
- 1 = Detected but not verified

2 = Acquired/verified
3 = Lost

(5) Intelligence status. Element 92 contains the total hours this target has been tracked this detection.

(6) Activity code. Element 82 indicates the status of the unit as not active (0) or active (1).

5. ALGORITHMS.

A. In simulating the production of the commander's intelligence map, the program considers three processes.

(1) The ability of the friendly sensors to detect individual elements of the enemy unit.

(a) Calculation of element detection by sensor groups considers elements of target units to be divided into five categories: personnel, armored combat vehicle, support vehicles, artillery, and large rockets. Target units and their associated elements can be located in one of five range bands from the sensors. In order to simplify game play, location of sensor groups is represented on the map board. Sensor groups consist of one or more individual sensors of varying types. Sensors in a group have individual detection probabilities limiting their range effectiveness. However, the program assumes that an individually exposed target element is covered by all persons in the group having an effective range to that element. The probability of the sensor group detecting an individual element is given by:

$$P_{Ge} = 1 - \prod_{\text{all } i} (1-P'_{ie})^{n_i} \qquad \text{(Eq. 3-1)}$$

where:

P_{Ge} = the probability that an exposed element of type e is detected by sensor group G.
P'_{ie} = the probability that sensor i can detect element e.
n_i = the number of sensors of type i in group G.

(b) As mentioned in the previous paragraph, the probabilities of detection P'_{ie} are range dependent and hence the probability of group detection P_{Ge} is also range dependent. The sensor detection probabilities P_{ie} were developed from sensor performance parameters found in the Target Acquisition Study (TAS II) conducted by Concepts Analysis Activity (CAA) in 1980. The data is classified and will not be presented in this document. However, it should be noted that the probabilities include the following parameters:

$$P'_{ie} = P_{ie} * Plos * Pa * Pc * Pj \qquad \text{(Eq. 3-2)}$$

where:

P_{ie} = the probability that sensor i can detect element e, given that the element is exposed and the sensor is functioning.

Plos = probability the sensor has line of sight to the target; in cases where sensors do not require line of sight, this probability is set to 1.

Pa = probability that the sensor is available at the moment the target is exposed.

Pc = probability that the crew manning the sensor will recognize the sensor representation of the target element.

Pj = probability that sensor is not jammed. The jamming probabilities are currently input estimates supplied by the gamer.

(2) The intelligence fusion process whereby numbers of detected enemy elements are mapped into the detection of enemy units by friendly personnel.

(a) The ability of the friendly force to recognize a unit from sensor detections of individual elements is represented in the program by two processes. The first process describes the exposure profile of the target unit. The second process calculates the probability that the unit will be recognized by intelligence personnel and correctly categorized as one of 10 unit types.

(b) The exposure profiles for individual elements in a unit are shown in Table 3-2. All target unit elements in the DIME unit files are categorized by the detection program as one of five signature types. The exposure percentages shown in Table 3-2 are then applied to the sum of the unit elements giving the number of elements exposed to the sensors. Table 3-2 was developed by the DAME project team using military judgment. It represents an effort to relate an element exposure profile to the unit mission.

(c) Tables 3-3 and 3-4 are percentages which are applied to the numbers of exposed target elements. The resulting numbers of elements are intelligence thresholds used by the program to simulate the fusion process of identifying a unit from its parts. The example in figure 3-2 provides an overview of the use of Tables 3-3 and 3-4 by the program. A Blue battalion executing a "move" mission has the number of elements as shown in the DIME "UNITFILE". These elements are then categorized into the five types shown

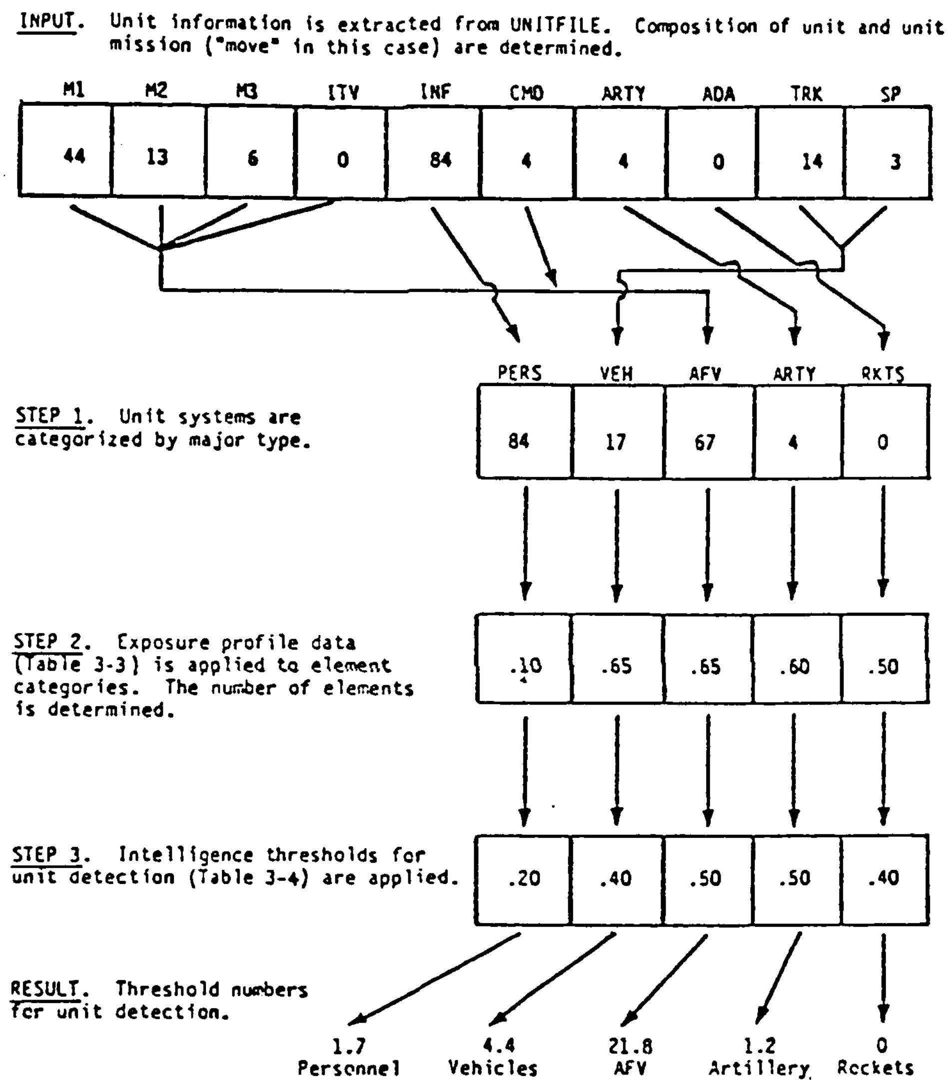


Figure 3-2. Example use of intelligence threshold data.

in step 1. In the second step, the program extracts the exposure profile percentages for a "move" from Table 3-2 and calculates these numbers of elements exposed to that sensor group. In step three, the intelligence profile is extracted from Table 3-3 and applied to the exposed elements to generate the numbers of unit elements representing intelligence thresholds. These represent the number of elements which must be detected before the intelligence map is posted with a unit detection. In the example, two personnel, five vehicles, 22 tanks, and two artillery pieces must be detected before the unit will be detected as a battalion.

(d) The detection program uses a normal approximation of the binomial distribution to calculate the probability of detecting the least threshold number of elements in each category in the following manner:

$$P_{ce} = 1 - \Phi[(T_e - \overline{X}_e)/\delta_e] \qquad \text{(Eq. 3-3)}$$

where:

P_{ce} = probability that the intel personnel receiving reports from sensor group c will detect enough category e elements to identify the unit.

$\Phi(X)$ = the cumulative normal density function evaluated to X.

T_e = the number of threshold elements of category e needed to detect the unit.

$\overline{X}_e, \delta e$ = the mean and standard deviation of the normal approximation to the binomial distribution.

T_e is represented by:

$$T_e = S_e * E_e * I_e * Z \qquad \text{(Eq. 3-4)}$$

where:

S_e = the sum of elements of type e in the unit.

E_e = the fraction of elements of type e exposed under the current mission (see Table 3-2).

I_e = the minimal fraction of elements of type e which must be detected before the unit can be detected (see Tables 3-3 and 3-4).

Z = a factor adjusting the intelligence threshold for units which were previously detected but are now lost. Z = 1.0 for units not previously detected and 0.75 for units lost during tracking.

$\overline{X}_e$ is represented by:

$$\overline{X}_e = S_e * E_e * P_{Ge} \qquad \text{(Eq. 3-5)}$$

where:

S_e and Ee are described above and P_{Ge} = probability that sensor group G will detect an element of type e.

δe is represented by:

$$\delta e = S_e * E_e * P_{Ge} * (1 - P_{Ge}) \qquad \text{(Eq. 3-6)}$$

where S_e, E_e and P_{Ge} are as described above.

The detection program calculates P_{Ge} for each element category within the unit. These probabilities are used in a Monte Carlo evaluation to determine which categories are detected within the unit. Units are "detected" and posted on the intelligence list if one-half or more of their element categories are detected.

(e) The use of target exposure profiles (Table 3-2), intelligence detection profiles (Tables 3-3 and 3-4), and the normal binomial approximation structure has some inherent assumptions and limitations. The reader is reminded that due to a paucity of field data, Table 3-2 was developed using military judgment. The underlying assumption in the use of this table is that the unit mission (not the unit type) describes the level of concealment the unit is able to achieve. Tables 3-3 and 3-4 are totally subjective in nature. They were also developed by the DAME programming team using military judgment. They were built under the assumption that a good intelligence officer will identify units with some prior knowledge of their type and mission. He will have maps to estimate reasonable deployment areas and some knowledge of enemy force structures to guide him in use of sensor reports. Finally, the use of the normal approximation of the binomial distribution works well when units have several elements. In a battalion/regimental game where most categories have more than 10 elements, it is adequate. Care should be used in applying it to smaller units.

(3) Unit losses and unit verification.

(a) Following the detection of a target unit, the program also represents the tracking of that unit by the sensor groups. Units are allowed to change positions to avoid detection at a tactically realistic rate. Table 3-5 contains the unit movement profile showing the time intervals at the end of which a unit will purposefully move to confuse the sensors. This table was also based on data used in the TAS II study. This movement should not be confused with a "move" mission where a unit is required to travel to another point on the battlefield. The avoidance movement is merely a changing of positions while maintaining the same mission and staying in the same tactical area. The program uses the following set of automated rules to represent tracking of detected units.

1. Units that are detected remain detected until they move.

2. Units that move to avoid detection must be redetected following their movement.

3. Units that are executing a move mission must be redetected every three hours.

4. Previously detected units that are not redetected during a move are placed in a "lost" status for three hours. If not redetected during the next three hours, they are moved to an "undetected" status and are removed from the intelligence list.

5. Detection thresholds are lowered to 75 percent of their normal values for redetecting moving units or units in a "lost" status.

(b) The program represents three states of unit detection.

1. Detected - a unit has been found and its type has been identified. Its mission, exact location, and strength are unknown.

2. Verified - The type and exact location of a unit are known. A reasonable estimate has been made of unit mission and strength.

3. Lost - A previously detected or verified unit has been lost during a move. Within three hours, the position and unit type are either known or moved to undetected status.

The program moves units from a "detected" to "verified" status if detection is maintained for four hours.

B. The previous paragraphs have described the principal methodology used in building the DIME target intelligence maps. This methodology is applied to both Blue and Red unit files during each three-hour period of the game. The program has three other features representing special intelligence situations which also impact the target hits.

(1) Begin game for Red/Blue. This feature initializes the target detection list and represents accumulated intelligence knowledge at the beginning of the game. The methodology is a simplified version of the one described above. Sensor groups and intelligence thresholds are ignored and replaced with tabular probabilities representing unit detections as a function of unit range from the forward line of own troops (FLOT). The unit detection probabilities were taken from the results of TAS II. This feature is required preceding the start of a game.

(2) Update intelligence for Blue/Red units in contact. The previous paragraphs have described a methodology for the detection of units which are

at ranges beyond the visual capabilities of localized battalion personnel. However, when battalions are within 10km of enemy units they are considered to be in contact by the DIME intelligence program. This is represented in the program by a special data base representing detection probabilities of long- range reconnaissance patrols against elements of enemy units. The methodology for calculating detection of enemy units is the same as that listed in paragraph 5a above.

(3) Red commander's verification of Blue units. One of the critical aspects of the deep strike game is Red's ability to identify Blue units operating in his rear area. It was believed that Red players would require a special representation of the sensor elements directly under the control of the Red commander. Specifically, air reconnaissance assets would probably be used by the Red commander to "verify" the presence of Blue units having a "detected" status operating in his rear area. In order to simulate this, the program uses probabilities of element detection representing airborne photo assets flying over a detected Blue unit. Any Blue unit redetected by the airborne units is immediately posted as "verified" on the Red commander's intelligence map. The methodology for unit identification is identical to that described in paragraph 5a above.

6. "UNITFILE" IMPACT.

The "UNITFILE" inputs and outputs are essential to the detection program.

A. Inputs. The detection program uses the unit information stored in elements 75, 89, 78, 90, 91, 92 and 82 to produce an intelligence map for both the Blue and Red commanders. (See the discussion under paragraphs 3, 4, and 5 of this chapter.)

B. Outputs. In addition to providing the intelligence map, the detection program updates the "UNITFILE" for the following words:

(1) Detection status (91). The detection status is represented by two numbers (X,Y), where the first number represents the hours left until the unit can be redetected and the second number represents the unit status as not detected (0), detected (1), verified (2), or lost (3).

(2) Total hours target has been tracked this detection (92). This value is an integer value identifying the total detection time, where maximum detection time is three hours.

7. CODE.

The detection code consists of a driver routine and seven subroutines: Load_pota, Decompos, Print_det_unit, Bld_sen_grp, Blue_in_contact, Prob_of_detect, and Normal_approx.

A. The driver routine controls the detection program through a menu-subroutine format. Figure 3-3 shows the menu with the accompanying subroutines.

B. Upon selection of the appropriate option, the driver routine calls one of three major subroutines: Load_pota, Blue_in_contact and BLd_sen_grp.

(1) Load_pota subroutine reads in the appropriate data files from both the auxiliary stored files and the "UNITFILE". This data, in turn, is passed to the Prob_of_detect routine.

(a) The Prob_of_detect routine applies a normal approximation methodology to determine a list of units which have been detected, verified, or lost.

(b) Once a unit has been classified as detected, verified, or lost, this information is passed to the Decompose routine and the Print det unit routine. The Decompose routine updates the "UNITFILE" for elements 91 and 92. The Print_det_unit routine prints out a list of the units and their intelligence status as either detected, verified, or lost.

(2) Blue_in_contact. The Blue_in_contact routine reads in the appropriate data files for battalions that are within 10 km of enemy units. This data, along with the unit information stored on the "UNITFILE" is passed to the Decompose routine and the Print_det_unit routine. Again the commander is provided a list of the unit's intelligence status and the "UNITFILE" is updated accordingly. Contrary to its name, this routine is used for both Red and Blue.

(3) Bld_sen_group. The Bld_sen_group subroutine allows the gamer to create, change, or list Red and Blue sensor groups. The gamer specifies the number of sensors, per type, he wishes to assign to each sensor group. The sensor groups consist of six types for both Blue and Red.

(a) The Blue sensor group is composed of the following six types of sensors:

1. Ground radar

2. Artillery radar

3. LRRP

4. SLAR

5. Air Force/Infrared (AF/IR)

6. FO.

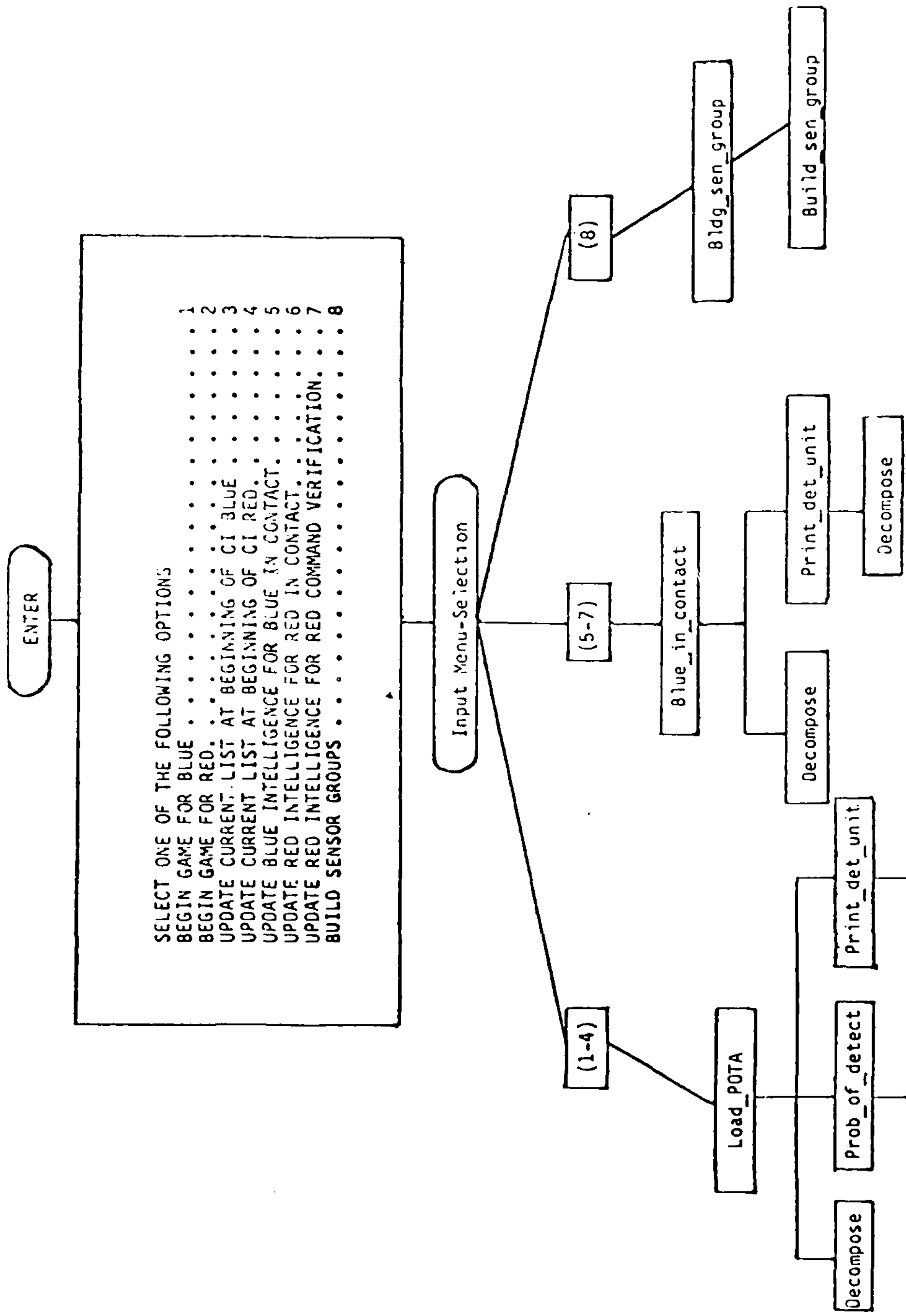


Figure 3-3. Detection menu selection diagram.

(b) The Red sensor group is composed of the following six types of sensors:

1. Ground radar

2. Artillery radar

3. LRRP

4. RPV

5. SLAR

6. FO.

These sensor groups are placed into the Bsenpro and Rsenpro files which are used within the Load_pota routine to determine the commander's intelligence list.

C. The subroutines and their primary variables are contained in Table 3-6. A listing of the original P7 code appears in Table 3-7.

Table 3-6. Detection Subroutine Table.

Functional area(s): A. Main Routine

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Driver	Sets the random number seed. Selects the appropriate option and transfers control to that section of code.		
Load_pota	Updates current list at beginning of CI.	A. H1(I,J)	Detection probabilities for sensor I where I represents the target type and J represents the sensor zone. I: 1 = Personnel 2 = Vehicles 3 = Tanks/APCs 4 = Artillery 5 = Rockets J: (1-5) sensor zones
		B. H2(*) - H6(*)	Detection probabilities for sensors 2 - 6.
		C. E1(I,J)	Array containing fraction of elements of unit I available for detection when unit is on mission J.
		D. F8(I,J)	Intelligence threshold array holding the fraction of a unit that must be detected before a unit detection can be assessed where: I = Element categories 1-5 J = Sensor groups 106
		E. G3(I,J)	Array containing the number of sensors of sensor type J (1-6) of group I (1-3).
		F. P1(I,J)	Array containing POTA detection probability for unit I (1-10) and zone J (1-5).

Table 3-6. Detection Subroutine Table.

Functional area(s): A. Main Routine (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Load_pota (concluded)		G. S1	Blue or Red detection flag. S1 = 1 if Blue detecting Red S1 = 2 if Red detecting Blue.
		H. O1	Variable containing value 1-8; user selected options for *acquisition module flow.*
		I. N1	The number of hours of intelligence needed (must be a multiple of 3).
		J. J1	Fraction of sensor messages jammed.
		K. N(*)	Copy of "UNITFILE" entries
		L. S5	Saved variable for sensor group covering current unit.
		M. Q1	*Integer part of packed data returned from decomposition* subroutine.
		N. Q2	Fractional part of packed data returned from decomposition subroutine.
		O. D2	*Unit detection flag:* D2 = 0 - unit not found D2 = 1 - unit found.

Table 3-6. Detection Subroutine Table.

Functional area(s): A. Main Routine (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Normal_approx	Uses the standardized normal distribution N(0,1) to approximate a binomial distribution. Subroutine returns the value D2 as the estimated parameter for the binomial.	A. X	Normalized random variable N(0,1) for approximating the binomial.
Bldg_sen_gp	Calls the Build_sen_group subroutine.		
Print_det_unit	Prints the current unit detection status for proper unit.		
Blue_in_contact	Updates Blue/Red unit commanders intelligence map for Blue/Red in contact	A. P7(I)	Contains the probability of detection for unit I (1-10) under one of the following searches: Blue battalion, Red regiment, Red commander.
		B. F(I)	Holds unit profile for Red or Blue unit I (1-10).
		C. N1	Number of Red or Blue units contacted.
Decompose	Decomposes the proper "UNITFILE" entry N(I) into integer and fractional parts.	A. D8	The absolute value of the "UNITFILE" entry being decomposed.
Prob_of_detect	Sums the elements in a unit (adjusted by the percent covered).	A. T2(I)	Array containing the total number of elements in unit I adjusted by the percent covered. (N(92))

Table 3-6. Detection Subroutine Table.

Functional area(s): A. Main Routine (concluded)

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Prob_of_detect (concluded)	Calculates detection of selected target type. Attempts to detect elements with calculated probabilities.	B. D	Proper zone unit
		C. M	Unit mission
		D. P9(I)	Probability of detection for target type I (1-5).
		E. M9	Probability of element detection for current element.
		F. T5	Number of element categories in this unit containing elements.
		G. T6	Number of element categories detected.
		H. Z8	Intelligence factor: Z8 = 1 - target has been detected. Z8 - .75 = target is being tracked.

Functional area(s): B. Build sen group

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Main Driver	Builds 3 sensor groups for Blue and 3 sensor groups for Red.	A. S(I)	Array containing number of each sensor type; I = 1-6.

Table 3-7. Detection code.

```
10     !!! - "P7" IS THE ACQUISITION PROGRAM FOR DIME. CODED BY MR. TOM BUTHERU
              SAD, CAORA, AV 552-5481.            DATED 22 OCT 83
20     ! EXPANDED VERSION -- JUNE 9, 1986 -- BY OAO CORP.
30     OPTION BASE 1
40     DIM P1(10,5),N(150),F(10),P7(10),H1(5,9),H2(5,9),H3(5,9),H4(5,9),H5(5,9)
6(5,9),F8(5,10),G3(400,6),P9(5),T2(5),E1(5,4),M$[16],Tm$[7],Temp(150)
50     Disk$=":9134,704,0"
60     Disk2$=":9134,704,0"
70     ! - SET RANDOM NUMBER SEED
80     PRINT USING "@"
90     PRINT "THIS IS THE DIME ACQUISITION PROGRAM"
100    INPUT "INPUT THE RANDOM NUMBER SEED FOR DETECTIONS ; RANGE 1-10000",R5
110    INPUT "WHAT IS THE GAME TIME?",Tm$
120    RANDOMIZE R5
130    FOR I=1 TO 10
140      R5=RND
150    NEXT I
160    PRINT USING "@"
170    PRINT "SELECT ONE OF THE FOLLOWING OPTIONS"
180    PRINT "BEGIN GAME FOR BLUE-1"
190    PRINT "BEGIN GAME FOR RED-2"
200    PRINT "UPDATE CURRENT LIST AT BEGINNING OF CI BLUE-3"
210    PRINT "UPDATE CURRENT LIST AT BEGINNING OF CI RED-4"
220    PRINT "UPDATE BLUE INTEL FOR BLUE IN CONTACT-5"
230    PRINT "UPDATE RED INTEL FOR RED IN CONTACT-6"
240    PRINT "UPDATE RED INTEL FOR RED COMMANDER VERIFICATION-7"
250    PRINT "BUILD SENSOR GROUPS-8"
260    PRINT "STOP RUN-9"
270    INPUT "INPUT OPTION",O1
280    IF O1=9 THEN 5270
290    ON O1 GOSUB Load_pota,Load_pota,Load_pota,Load_pota,Blue_in_contact,Blue
n_contact,Blue_in_contact,Bld_sen_grp
300    GOTO 160
310 Load_pota:!
320    ON O1 GOTO 1280,1420,340,810
330    ! - BLUE CI UPDATE
340    ASSIGN @Path1 TO "BS1T06"&Disk2$
350    FOR I=1 TO 5
360      ENTER @Path1,I;H1(I,1);H1(I,2);H1(I,3);H1(I,4);H1(I,5)
370    NEXT I
380    J=6
390    FOR I=1 TO 5
400      ENTER @Path1,J;H2(I,1);H2(I,2);H2(I,3);H2(I,4);H2(I,5)
410      J=J+1
420    NEXT I
430    FOR I=1 TO 5
440      ENTER @Path1,J;H3(I,1);H3(I,2);H3(I,3);H3(I,4);H3(I,5)
450      J=J+1
460    NEXT I
470    FOR I=1 TO 5
480      ENTER @Path1,J;H4(I,1);H4(I,2);H4(I,3);H4(I,4);H4(I,5)
490      J=J+1
```

Table 3-7. Detection code.

```
500   NEXT I
510   FOR I=1 TO 5
520     ENTER @Path1,J;H5(I,1);H5(I,2);H5(I,3);H5(I,4);H5(I,5)
530     J=J+1
540   NEXT I
550   FOR I=1 TO 5
560     ENTER @Path1,J;H6(I,1);H6(I,2);H6(I,3);H6(I,4);H6(I,5)
570     J=J+1
580   NEXT I
590   ASSIGN @Path1 TO *
600   ASSIGN @Path2 TO "RE1"&Disk2$
610   FOR I=1 TO 4
620     ENTER @Path2;E1(1,I);E1(2,I);E1(3,1);E1(4,I);E1(5,I)
630   NEXT I
640   ASSIGN @Path2 TO *
650   ASSIGN @Path3 TO "BTHOLD"&Disk2$
660   FOR J=1 TO 10
670     ENTER @Path3;F8(1,J);F8(2,J);F8(3,J);F8(4,J);F8(5,J)
680   NEXT J
690   ASSIGN @Path3 TO *
700   ASSIGN @Path4 TO "UNITFILE"&Disk2$
710   FOR I=1 TO 191
720     ENTER @Path4,I;Temp(*)
730     FOR J=95 TO 100
740       K=J-94
750       G3(I,K)=Temp(J)
760     NEXT J
770   NEXT I
780   ASSIGN @Path4 TO *
790   GOTO 1280
800   ! - RED CI UPDATE
810   ASSIGN @Path5 TO "RS1TO6"&Disk2$
820   FOR I=1 TO 5
830     ENTER @Path5,I;H1(I,1);H1(I,2);H1(I,3);H1(I,4);H1(I,5)
840   NEXT I
850   J=6
860   FOR I=1 TO 5
870     ENTER @Path5,J;H2(I,1);H2(I,2);H2(I,3);H2(I,4);H2(I,5)
880     J=J+1
890   NEXT I
900   FOR I=1 TO 5
910     ENTER @Path5,J;H3(I,1);H3(I,2);H3(I,3);H3(I,4);H3(I,5)
920     J=J+1
930   NEXT I
940   FOR I=1 TO 5
950     ENTER @Path5,J;H4(I,1);H4(I,2);H4(I,3);H4(I,4);H4(I,5)
960     J=J+1
970   NEXT I
980   FOR I=1 TO 5
990     ENTER @Path5,J;H5(I,1);H5(I,2);H5(I,3);H5(I,4);H5(I,5)
1000    J=J+1
1010  NEXT I
```

Table 3-7. Detection code.

```
1020 FOR I=1 TO 5
1030   ENTER @Path5,J;H6(I,1);H6(I,2);H6(I,3);H6(I,4);H6(I,5)
1040   J=J+1
1050 NEXT I
1060 ASSIGN @Path5 TO *
1070 ASSIGN @Path6 TO "BE1"&Disk2$
1080 FOR I=1 TO 4
1090   ENTER @Path6;E1(1,I);E1(2,I);E1(3,I);E1(4,I);E1(5,I)
1100 NEXT I
1110 ASSIGN @Path6 TO *
1120 ASSIGN @Path7 TO "RTHOLD"&Disk2$
1130 FOR J=1 TO 10
1140   ENTER @Path7;F8(1,J);F8(2,J);F8(3,J);F8(4,J);F8(5,J)
1150 NEXT J
1160 ASSIGN @Path7 TO *
1170 ASSIGN @Path8 TO "UNITFILE"&Disk2$
1180 FOR I=192 TO 400
1190   ENTER @Path8,I;Temp(*)
1200   FOR J=95 TO 100
1210     K=J-94
1220     G3(I,K)=Temp(J)
1230   NEXT J
1240 NEXT I
1250 ASSIGN @Path8 TO *
1260 GOTO 1420
1270 ! - BEGIN GAME FOR BLUE, LOAD RED POTA
1280 ASSIGN @Path9 TO "REDPOTA"&Disk2$
1290 FOR I=1 TO 10
1300   ENTER @Path9;P1(I,1);P1(I,2);P1(I,3);P1(I,4);P1(I,5)
1310 NEXT I
1320 ASSIGN @Path9 TO *
1330 ASSIGN @Path10 TO "RFLEE"&Disk2$
1340 ENTER @Path10;F(*)
1350 ASSIGN @Path10 TO *
1360 ! - SET FLAG FOR RED
1370 S1=1
1380 L6=192
1390 L7=400
1400 GOTO 1550
1410 ! - BEGIN GAME FOR RED, LOAD BLUE POTA
1420 ASSIGN @Path11 TO "BLUPOT1"&Disk2$
1430 FOR I=1 TO 10
1440   ENTER @Path11;P1(I,1);P1(I,2);P1(I,3);P1(I,4);P1(I,5)
1450 NEXT I
1460 ASSIGN @Path11 TO *
1470 ASSIGN @Path12 TO "BFLEE"&Disk2$
1480 ENTER @Path12;F(*)
1490 ASSIGN @Path12 TO *
1500 ! - SET FLAG FOR BLUE
1510 S1=2
1520 L6=1
1530 L7=191
```

Table 3-7. Detection code.

```
1540  ! - OPEN DIME UNITFILE
1550  ASSIGN @Path13 TO "UNITFILE"&Disk$
1560  INPUT "INPUT NUMBER OF HOURS OF INTEL NEEDED (MUST BE MULTIPLE OF 3)",N1
1570  N1=N1/3
1580  N1=INT(N1)
1590  J=0
1600  REPEAT
1610    INPUT "INPUT FRACTION OF SENSOR MESSAGES JAMMED",J1
1620  UNTIL J1>=0 AND J1<=1
1630  ! - SET TIME LOOP FOR DETECTIONS
1640  FOR T=1 TO N1
1650  ! - SET UNIT RECORD LOOP
1660    FOR J=L6 TO L7
1670      D1=1
1680  ! - RETRIEVE RECORD FOR UNIT TO BE DETECTED
1690      ENTER @Path13,J;N(*)
1700      IF S1=INT(N(78)) OR N(82)=0 THEN 2830
1710  ! - CHECK FOR LOST TARGET
1720      IF N(91)<.29 OR N(91)>.31 THEN 1750
1730      N(91)=0
1740  ! - CHECK FOR TARGET COVERAGE BY SOME SENSOR GROUP
1750      D=N(89)
1760      GOSUB Decompose
1770      S5=Q2
1780      Z5=Q1
1790      IF S5<>0 THEN 1890
1800  ! - TARGET NOT COVERED BY SENSOR
1810      D=N(91)
1820      GOSUB Decompose
1830      IF Q2=1 OR Q2=2 THEN 2420
1840  ! - TARGET NOT PREVIOUSLY DETECTED OR IN LOST STATUS
1850      N(91)=0
1860      N(92)=0
1870      GOTO 2820
1880  ! - TARGET UNDER SENSOR COVERAGE, DECOMPOSE TIME
1890      D=N(91)
1900      GOSUB Decompose
1910  ! - CHECK FOR DETECTION STATUS
1920      IF Q2>0 THEN 2460
1930  ! - SYSTEM NOT PREVIOUSLY DETECTED, DECOMPOSE LOCATION, PERFORM
1940  ! - DETECTION FUNCTION, AND SET UP POINTERS FOR PROPER POTA TABLE
1950      D=N(89)
1960      GOSUB Decompose
1970  ! - SAVE ZONE IN Z5 AND SAVE SENSOR GROUP IN S5
1980      Z5=Q1
1990      S5=Q2
2000  ! - SELECT UNIT TYPE
2010      D=N(78)
2020      GOSUB Decompose
2030      Q2=Q2+1
2040      Q1=Z5
2050  ! - SELECT PROPER PROBABILITY, RETRIEVE SENSOR
```

Table 3-7. Detection code.

```
2060      IF S5<>0 THEN 2090
2070      PRINT "ERROR AT 1110";S5;Q1;N(89);N(78)
2080      STOP
2090      IF S5>8.5 AND S5<=9.5 THEN 2160
2100      GOSUB Prob_of_detect
2110      D=N(78)
2120      GOSUB Decompose
2130      IF D2=0 THEN 2390
2140    ! - UNIT DETECTED
2150      GOTO 2250
2160    ! - COVERED BY GENERAL GROUP
2170      P5=P1(Q2,Q1)
2180      R5=RND
2190      IF R5>P5 THEN 2390
2200    ! - CHECK FOR EW INTERFERENCE
2210      R5=RND
2220      IF R5>1-J1 THEN 2390
2230    ! - SYSTEM HAS BEEN DETECTED, UPDATE FLEE TIME DETECTION STATUS
2240    ! - AND MOVE TO NEW RECORD
2250      D=N(78)
2260      GOSUB Decompose
2270      Q1=F(Q2+1)
2280      Q2=1
2290      IF D1=1 THEN 2350
2300    ! - TARGET PREVIOUSLY DETECTED AND BEING TRACKED, TAKE OLD STATUS
2310      S=Q1
2320      D=N(91)
2330      GOSUB Decompose
2340      Q1=S
2350      N(91)=Q1+Q2/10
2360    ! - UPDATE TIME WATCHED
2370      N(92)=N(92)+3
2380      GOTO 2720
2390      N(91)=0
2400    ! - CHECK FOR LOST UNIT
2410      IF D1<2 THEN 2430
2420      N(91)=.3
2430      N(92)=0
2440      GOTO 2820
2450    ! - SYSTEM PREVIOUSLY DETECTED
2460      IF N(75)<>3 THEN 2510
2470    ! - SYSTEM IS STATIONARY, UPDATE TIME WATCHED
2480      N(92)=N(92)+3
2490      GOTO 2720
2500    ! - CHECK FOR SYSTEM MOVING
2510      IF N(75)<>4 THEN 2580
2520    ! - SYSTEM IS MOVING, SET STATIONARY TIME TO ZERO, TAKE OFF
2530    ! - DETECTED LIST, AND TRY TO DETECT
2540      D1=2
2550      GOTO 1950
2560    ! - SYSTEM IS EITHER IN ATTACK OR DEFEND AND DETECTED, UPDATE FLEE
2570    ! - TIME AND DETECT TIME
```

Table 3-7. Detection code.

```
2580      N(92)=N(92)+3
2590      D=N(91)
2600      GOSUB Decompose
2610      Q1=Q1-3
2620      IF Q1<=0 THEN 2670
2630      N(91)=Q1+Q2/10
2640      GOTO 2720
2650   ! - SYSTEM HAS MOVED, TIME TO REDETECT SYSTEM. FLEE TIME HAS
2660   ! - EXPIRED, SET DETECTION TO ZERO AND GO TO NEW RECORD
2670      N(92)=N(92)-3
2680   ! - SUBTRACT FLEE TIME
2690      D1=2
2700      GOTO 1950
2710   ! - CHECK ON VERIFICATION OF UNIT
2720      D=N(92)
2730      GOSUB Decompose
2740      S3=N(92)
2750   ! - COMPARE TIME OF DETECT WITH FLEE TIME, RETRIEVE FLEE TIME
2760      D=N(78)
2770      GOSUB Decompose
2780      Q2=Q2+1
2790      IF S3<4 THEN 2820
2800   ! - SYSTEM IS VERIFIED, UPDATE UNITFILE
2810      N(91)=INT(N(91))+.2
2820      OUTPUT @Path13,J;N(*)
2830    NEXT J
2840   NEXT T
2850   ASSIGN @Path13 TO *
2860   GOSUB Print_det_unit
2870   RETURN
2880   !
2890 Bld_sen_grp:  !
2900   CALL Build_sen_group
2910   RETURN
2920   !
2930   !
2940 Print_det_unit:!
2950   ASSIGN @Path13 TO "UNITFILE"&Disk$
2960   ASSIGN @Path14 TO "NAMEFILE"&Disk$
2970   INPUT "TARGET LIST TO SCREEN OR PRINTER? (S/P)",S_p_$
2980   IF S_p_$="P" THEN PRINTER IS 702
2990   PRINT USING "@,#"
3000   IF S1=2 THEN 3030
3010   PRINT "BLUE COMMANDER TARGET LIST AS OF ";Tm$
3020   GOTO 3040
3030   PRINT "RED COMMANDER TARGET LIST AS OF ";Tm$
3040   PRINT "UNIT DETECTION STATUS"
3050   FOR I=L6 TO L7
3060     ENTER @Path13,I;N(*)
3070     ENTER @Path14,I;M$
3080     IF S1=INT(N(78)) OR N(82)=0 THEN 3160
3090     D=N(91)
```

Table 3-7. Detection code.

```
3100     GOSUB Decompose
3110     IF Q2=0 THEN 3160
3120     IF Q2=2 THEN 3150
3130     PRINT USING "10X,3D,5X,16A,5X,10A";I,"DETECTED","__________"
3140     GOTO 3160
3150     PRINT USING "10X,3D,5X,16A,5X,10A";I,M$,"__________"
3160   NEXT I
3170   PRINTER IS .
3180   PRINT USING "///////"
3190   PRINT "PRESS CONT TO PROCEED"
3200   PAUSE
3210   ASSIGN @Path13 TO *
3220   ASSIGN @Path14 TO *
3230   RETURN
3240   !
3250 Blue_in_contact:!UPDATE BLUE UNIT COMMANDER'S INTELLIGENCE MAP FOR BLUE IN
CONTACT
3260   S1=1
3270   L6=192
3280   L7=400
3290   IF O1<6 THEN 3340
3300   S1=2
3310   L6=1
3320   L7=191
3330   IF O1=6 THEN 3340
3340   ON O1-4 GOTO 3360,3440,3520
3350   ! - BLUE INTEL UNIT, LOAD BLUE DETECTION PROBABILITIES
3360   ASSIGN @Path14 TO "BLBDG"&Disk2$
3370   ENTER @Path14;P7(*)
3380   ASSIGN @Path14 TO *
3390   ASSIGN @Path10 TO "RFLEE"&Disk2$
3400   ENTER @Path10;F(*)
3410   ASSIGN @Path10 TO *
3420   GOTO 3580
3430   ! - RED INTEL UNIT, LOAD RED BDE DETECTION PROBABILITIES
3440   ASSIGN @Path15 TO "RDBDE"&Disk2$
3450   ENTER @Path15;P7(*)
3460   ASSIGN @Path15 TO *
3470   ASSIGN @Path12 TO "BFLEE"&Disk2$
3480   ENTER @Path12;F(*)
3490   ASSIGN @Path12 TO *
3500   GOTO 3580
3510   ! - RED COMMANDER'S INTELLIGENCE
3520   ASSIGN @Path16 TO "RDCOR"&Disk2$
3530   ENTER @Path16;P7(*)
3540   ASSIGN @Path16 TO *
3550   ASSIGN @Path12 TO "BFLEE"&Disk2$
3560   ENTER @Path12;F(*)
3570   ASSIGN @Path12 TO *
3580   ASSIGN @Path13 TO "UNITFILE"&Disk$
3590   PRINT USING "@"
3600   IF S1=2 THEN 3630
```

Table 3-7. Detection code.

```
3610  INPUT "INPUT NUMBER OF RED UNITS CONTACTED",N1
3620  GOTO 3640
3630  INPUT "INPUT NUMBER OF BLUE UNITS CONTACTED",N1
3640  FOR I=1 TO N1
3650    INPUT "INPUT UNIT NUMBER",J
3660    ENTER @Path13,J;N(*)
3670    IF S1<>INT(N(78)) AND N(82)<>0 THEN 3730
3680     ! - UNIT IS SAME SIDE AS SEARCHER
3690    IF N(82)=0 THEN PRINT "UNIT ";J;" IS INACTIVE, INPUT CORRECT UNIT"
3700    IF S1=INT(N(78)) THEN PRINT "UNIT ";J;" IS ON SAME FORCE AS INTEL MAP,
NPUT CORRECT UNIT"
3710    GOTO 3650
3720     ! - CHECK ON UNIT DETECTION
3730    D=N(78)
3740    GOSUB Decompose
3750    R5=RND
3760    V2=Q2+1
3770    IF R5>P7(V2) THEN 3860
3780     ! - UNIT IS ACQUIRED, SET ACQUISITION AND FLEE TIME
3790    D=N(91)
3800    GOSUB Decompose
3810     ! - CHECK FLEE TIME
3820    IF Q1>F(V2) THEN 3840
3830    Q1=F(V2)
3840    N(91)=Q1+.2
3850    OUTPUT @Path13,J;N(*)
3860  NEXT I
3870  GOSUB Print_det_unit
3880  ASSIGN @Path13 TO *
3890  RETURN
3900  !
3910 Decompose:!
3920  D8=ABS(D)
3930  Q1=INT(D8)
3940  Q2=INT((D8-Q1)*10+.1)
3950  RETURN
3960  !
3970 Prob_of_detect:!SUM ELEMENTS IN UNIT ADJUSTED BY PERCENT COVERED
3980  IF J<192 THEN
3990    Tot_sml_arms=0
4000    FOR I=36 TO 47
4010      Tot_sml_arms=Tot_sml_arms+N(I)
4020    NEXT I
4030    T2(1)=(Tot_sml_arms+N(7)+N(8)+N(53)+N(54))*N(90)
4040    Tot_veh=0
4050    FOR I=55 TO 70
4060      Tot_veh=Tot_veh+N(I)
4070    NEXT I
4080    FOR I=16 TO 20
4090      Tot_veh=Tot_veh+N(I)
4100    NEXT I
4110    T2(2)=(N(10)+N(2)+N(4)+Tot_veh)*N(90)
```

Table 3-7. Detection code.

```
4120     T2(3)=(N(1)+N(3))*N(90)
4130     Tot_if=0
4140     FOR I=21 TO 35
4150       Tot_if=Tot_if+N(I)
4160     NEXT I
4170     T2(4)=(Tot_if+N(5))*N(90)
4180     Tot_ada=0
4190     FOR I=48 TO 52
4200       Tot_ada=Tot_ada+N(I)
4210     NEXT I
4220     T2(5)=Tot_ada*N(90)
4230   ELSE
4240     Tot_sml_arms=0
4250     FOR I=36 TO 47
4260       Tot_sml_arms=Tot_sml_arms+N(I)
4270     NEXT I
4280     T2(1)=(N(7)+N(8)+Tot_sml_arms+N(53)+N(54))*N(90)
4290     Tot_trks=0
4300     FOR I=55 TO 61
4310       Tot_trks=Tot_trks+N(I)
4320     NEXT I
4330     T2(2)=(N(10)+Tot_trks)*N(90)
4340     Tot_df_car=0
4350     FOR I=16 TO 20
4360       Tot_df_car=Tot_df_car+N(I)
4370     NEXT I
4380     Tot_sp_veh=0
4390     FOR I=62 TO 70
4400       Tot_sp_veh=Tot_sp_veh+N(I)
4410     NEXT I
4420     T2(3)=(N(1)+N(3)+N(5)+N(6)+N(48)+Tot_df_car+Tot_sp_veh)*N(90)
4430     Tot_if=0
4440     FOR I=21 TO 35
4450       Tot_if=Tot_if+N(I)
4460     NEXT I
4470     T2(4)=(N(9)+Tot_if)*N(90)
4480     T2(5)=(N(49)+N(50)+N(51)+N(52))*N(90)
4490   END IF
4500   ! - OBTAIN PROPER ZONE FOR UNIT
4510   D=N(89)
4520   GOSUB Decompose
4530   ! - SELECT UNIT MISSION
4540   M=N(75)
4550   ! - CALCULATE PROBABILITY OF DETECTION OF THIS TARGET TYPE
4560   FOR K8=1 TO 5
4570     P9(K8)=1
4580     IF INT(T2(K8))<=0 THEN 4770
4590   ! - CHECK ALL ELEMENTS BY THIS GROUP
4600     T4=INT(T2(K8))
4610     FOR K9=1 TO 6
4620       IF G3(J,K9)=0 THEN 4760
4630       ON K9 GOTO 4640,4660,4680,4700,4720,4740
```

Table 3-7. Detection code.

```
4640        M9=H1(K8,Q1)
4650        GOTO 4750
4660        M9=H2(K8,Q1)
4670        GOTO 4750
4680        M9=H3(K8,Q1)
4690        GOTO 4750
4700        M9=H4(K8,Q1)
4710        GOTO 4750
4720        M9=H5(K8,Q1)
4730        GOTO 4750
4740        M9=H6(K8,Q1)
4750        P9(K8)=P9(K8)*(1-(1-J1)*M9)^G3(J,K9)
4760      NEXT K9
4770      P9(K8)=1-P9(K8)
4780    NEXT K8
4790    ! - INDIVIDUAL PROBABILITIES ARE NOW IN P9, ATTEMPT TO DETECT
4800    T5=0
4810    T6=0
4820    ! - SET UP UNIT TYPE
4830    D=N(78)
4840    GOSUB Decompose
4850    U4=Q2+1
4860    FOR I8=1 TO 5
4870      IF T2(I8)<=0 THEN 5050
4880      T5=T5+1
4890    ! - CALCULATE NUMBER OF ELEMENTS WHICH MUST BE DETECTED
4900      IF P9(I8)<.01 THEN 5050
4910      IF P9(I8)>.99 THEN 5040
4920    ! - Z8 IS INTELLIGENCE FACTOR, IF TARGET HAS BEEN DELETED THEN Z8 = FULL
HRESHOLD, IF TRACKING THEN Z8 = .75
4930      Z8=1
4940      IF D1=2 THEN Z8=.75
4950      D8=T2(I8)*F8(I8,U4)*E1(I8.M)*Z8
4960    ! - SET MEAN AND STANDARD DEVIATION FOR NORMAL
4970      X1=T2(I8)*P9(I8)*E1(I8,M)
4980      S8=T2(I8)*P9(I8)*(1-P9(I8))*E1(I8,M)
4990      X=(D8-X1)/SQR(S8)
5000      GOSUB Normal_approx
5010      R5=RND
5020      IF R5<D2 THEN 5050
5030    ! - UNIT ELEMENTS DETECTED
5040      T6=T6+1
5050    NEXT I8
5060    ! - IF ONE-HALF IS DETECTED THEN UNIT IS DETECTED
5070    D2=0
5080    IF T6=0 AND T5=0 THEN 5110
5090    IF T6<T5/2 THEN 5110
5100    D2=1
5110    RETURN
5120    !
5130 Normal_approx:'
5140    IF ABS(X)>20 THEN 5220
```

Table 3-7. Detection code.

```
5150  T8=1/(1+.2316419*ABS(X))
5160  D2=T8*(.31938153+T8*(-.356563782+1.781477937*T8))
5170  D2=D2+T8^4*(-1.821255978+1.330274429*T8)
5180  D2=SQR(1/(2*PI))*EXP(-X*X/2)*D2
5190  T8=10^(5+INT(-LGT(D2)))
5200  D2=INT(T8*D2+.5)/T8
5210  GOTO 5230
5220  D2=0
5230  IF X<0 THEN 5250
5240  D2=1-D2
5250  RETURN
5260  !
5270  PRINT USING "@"
5280  PRINT "THE RANDOM SEED FOR NEXT RUN IS"
5290  PRINT USING 5300;R5*10000
5300  IMAGE 7D.3D
5310  PRINT USING "//////"
5320  PRINT "PRESS CONT TO PROCEED"
5330  PAUSE
5340  LOAD "DIME"&Disk$
5350  END
5360  SUB Build_sen_group
5370  ! (GROUPBLD) BUILDS 9 SENSOR GROUPS FOR BLUE AND 9 FOR RED
5380    DIM S(1:6)
5390    Disk2$=":9134,704,0"
5400    PRINT "THIS IS THE DIME SENSOR GROUP BUILDER"
5410    PRINT "SELECT ONE OF THE FOLLOWING OPTIONS"
5420    PRINT "BUILD BLUE SENSOR GROUPS - 1"
5430    PRINT "BUILD RED SENSOR GROUPS - 2"
5440    PRINT "DUMPFILES/STOP - 3"
5450    INPUT O1
5460    ON O1 GOTO 5470,5560,5650
5470    PRINT "BUILDING BLUE GROUPS"
5480    ASSIGN @Path13 TO "BSENPRO"&Disk2$
5490    FOR I=1 TO 9
5500      PRINT "INPUT NUMBER OF SENSORS OF EACH TYPE FOR GROUP ";I;"GDRADAR,AR
RADAR,LRRP,SLAR,AF/IR,FO"
5510      INPUT S(1),S(2),S(3),S(4),S(5),S(6)
5520      OUTPUT @Path13;S(*)
5530    NEXT I
5540    ASSIGN @Path13 TO *
5550    GOTO 5410
5560    PRINT "BUILDING RED SENSOR GROUPS"
5570    ASSIGN @Path14 TO "RSENPRO"&Disk2$
5580    FOR J=1 TO 9
5590      PRINT "INPUT NUMBER OF SENSORS OF EACH TYPE FOR GROUP ";J;"GDRADAR,AR
RADAR,LRRP,RPV,SLAR,FO"
5600      INPUT S(1),S(2),S(3),S(4),S(5),S(6)
5610      OUTPUT @Path14;S(*)
5620    NEXT J
5630    ASSIGN @Path14 TO *
5640    GOTO 5410
```

Table 3-7. Detection code.

```
5650    PRINT "DUMPING SENSOR FILES"
5660    ASSIGN @Path13 TO "BSENPRO"&Disk2$
5670    PRINT "BLUE"
5680    PRINT "GDRADAR,ARTRADAR,LRRP,SLAR.AF/IR.FO"
5690    FOR K=1 TO 9
5700      ENTER @Path13;S(*)
5710      PRINT "GROUP ";K,S(1);S(2);S(3);S(4);S(5);S(6)
5720    NEXT K
5730    ASSIGN @Path13 TO *
5740    PRINT "RED"
5750    ASSIGN @Path14 TO "RSENPRO"&Disk2$
5760    PRINT "GDRADAR,ARTRADAR,LRRP,RPV,SLAR,FO"
5770    FOR L=1 TO 9
5780      ENTER @Path14;S(*)
5790      PRINT "GROUP ";L,S(1);S(2);S(3);S(4);S(5);S(6)
5800    NEXT L
5810    ASSIGN @Path14 TO *
5820  !
5830  SUBEND
```

CHAPTER 4

LOGISTICS

1. PURPOSE.

The purpose of the DIME logistics program (P2) is to establish initial levels of ammunition and fuel available for combat operations during the ground combat (P4) and air defense (P3) programs.

2. GENERAL.

The logistics program calculates the total ammunition and fuel (POL) available for the DIME model by:

A. Calculating the total combat and noncombat ammunition/fuel available as a function of:

(1) Basic load and/or fuel capacity for each of the 70 elements on a unit's weapons list.

(2) Resupplied ammunition/fuel.

(3) Excess ammunition/fuel returned from ground combat and air defense programs.

B. Subtracting the ammunition allocated for noncombat consumption from the total allocation and placing the total remaining combat allocation into elements 131-133 of the unit status file ("UNITFILE"). This combat allocation is depleted during the execution of the ground combat and air defense programs.

C. And, finally, distributing the remaining ammunition/fuel quantities available onto the trucks alive at the start of the game turn. Any excess ammunition or fuel is stored on the ground.

3. DATA FLOW.

A. The DIME logistics program contains two types of data inputs: auxiliary storage and online data statements. See Figure 4-1 for the data flow of this program.

(1) Auxiliary storage.

(a) "UNITFILE". Table 4-1 contains a list of "UNITFILE" elements required as inputs by the logistics program. Each element's location within

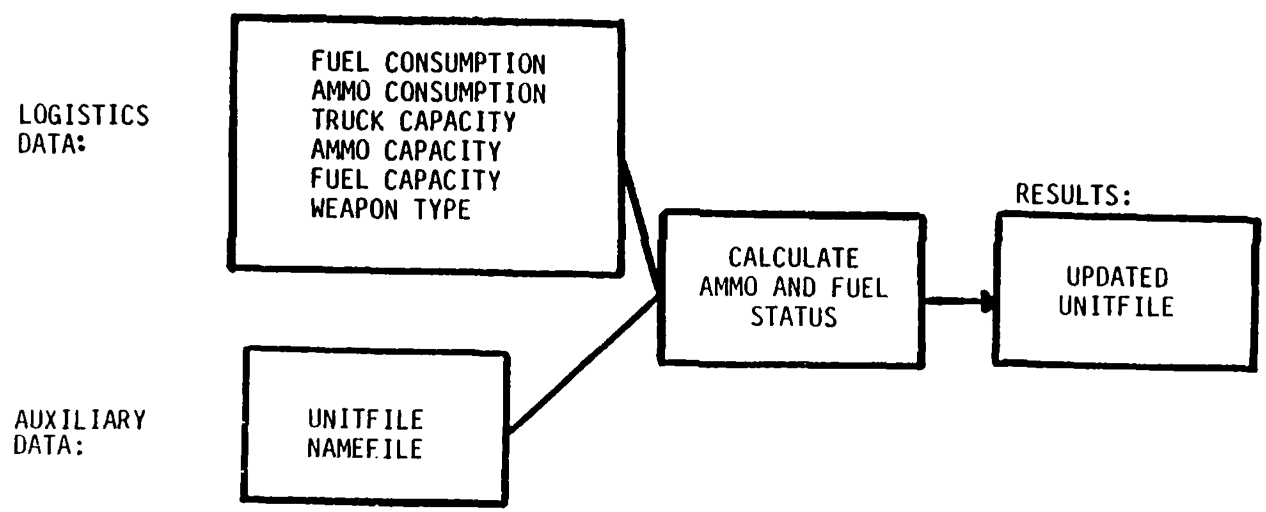


Figure 4-1. Logistics data flow.

Table 4-1. Logistics "UNITFILE" inputs.

"UNITFILE" Locations	Element Description
101	Fuel status of unit vehicles; value ranges from 0 to 1.0.
102	Fuel status of unit helicopters; value ranges from 0 to 1.0.
103	Fuel on tankers (gallons).
104	JP4 on tankers (gallons).
105	Fuel on ground (gallons).
106	JP4 on ground (gallons).
110	Fuel resupplied (gallons).
111	JP4 resupplied (gallons).
112	Fuel dispensed to other units (gallons).
113	JP4 dispensed to other units (gallons).
119 - 121	DF, IF, and AD vehicle ammunition status; values range from 0 to 1.0.
122	Helicopter ammunition status.
123	Ammunition on cargo vehicles (short tons).
124	Distribution of cargo by type (XXX.YYY).*
125	Ammunition on ground (short tons).
126	Distribution of ground ammunition by type (XXX.YYY).*

* (XXX.YYY) represents: XXX = DF ammo percent (XXX = XX.X%)
YYY = IF ammo percent (YYY = YY.Y%)
AD ammo percent is the amount remaining (100 - XX.X - YY.Y).

Table 4-1. Logistics "UNITFILE" inputs (continued).

"UNITFILE" Locations	Element Description
131 - 133	DF, IF, and AD ammunition available for consumption (short tons).
134	Helicopter ammunition availible for consumption.
135	Ammunition resupplied (short tons).
136	Ammunition resupply profile (XXX.YYY).*
137	Ammunition dispensed to other units (short tons).
138	Dispensed ammunition profile (XXX.YYY).*
139 - 141	DF, IF, and AD ammunition consumed to date (short tons).
142	Helicopter ammunition consumed to date (short tons).
143	Fuel consumed to date (gallons).
144	JP4 consumed to date (gallons).

Refer to Chapter 1 for a complete description of the entire "UNITFILE".

* (XXX.YYY) represents: XXX = DF ammo percent (XXX = XX.X%)
YYY = IF ammo percent (YYY = YY.Y%)
AD ammo percent is the amount remaining (100 - XX.X - YY.Y).

the "UNITFILE" and a short description are furnished. The elements are listed in ascending order according to their locations within the "UNITFILE".

(b) "NAMEFILE". The "NAMEFILE" contains the names for the units stored in the "UNITFILE".

(2) Online data. The logistics program uses six online data arrays.

(a) The fuel array contains fuel consumption values.

(b) The ammo array contains ammunition consumption values.

(c) The truck capacity array contains the carrying capacity of each truck.

(d) The ammo capacity array contains the package weight for each weapon in tons.

(e) The fuel capacity array contains the fuel required by each weapon element in gallons. If an element does not require fuel, a zero is assigned to the array.

(f) The weapon type array indicates the type of ammunition being fired (AD, IF, DF).

B. The logistics program combines the inputs listed in Table 4-1 with the methodology discussed in paragraph 5 to provide an updated "UNITFILE" for the current critical incident (CI).

4. FILE STRUCTURE.

The execution of the logistics program requires the "UNITFILE", two auxiliary files, and six online data arrays.

A. <u>Auxiliary storage.</u>

(1) The "UNITFILE" is created by the interactive running of the game initialization program (P1). It consists of 400 records, each containing 150 elements. The assignment of records is as follows: records 1-191 Blue units, and records 192-400 Red units. The logistics program requires the use of 32 elements on the "UNITFILE" as inputs and/or outputs. Refer to Tables 4-1 and 4-2 for these inputs and outputs.

(2) The "NAMEFILE" is the only other auxiliary file required by the logistics program except for the "UNITFILE". This file consists of a record containing the names of all units. It is created new for each game by the interactive running of the game initialization program (P1) and contains character strings that identify the units assigned to the game.

B. Online data. Online data consists of six data arrays: Fuel(*), Ammo(*), Truck_cap(*), Ammo_cap(*), Fuel_cap(*), Wpn_type(*).

(1) Fuel(I,J). Fuel(I,J) is a 10 x 70 array containing the consumption of fuel, in gallons, per six hours for 70 elements under 10 missions:

Missions:
- 0 = Meeting engagement
- 1 = Indirect fire
- 2 = Movement
- 3 = Frontal attack
- 4 = Envelopmental attack
- 5 = Delay
- 6 = Hasty defense
- 7 = Prepared defense
- 8 = Reserve/rear area
- 9 = Ambush

(2) Ammo(I,J). Ammo(I,J) is a 10 x 70 array containing the noncombat consumption of ammunition in tons per six hours for 70 elements operating under 10 missions (see above). These values represent leakage within the system.

(3) Truck_cap(J). Truck_cap(J) is a two-dimensioned array containing the fuel-carrying capacity in gallons for a single truck and the cargo-carrying capacity in tons for a single truck.

(4) Ammo_cap(J). Ammo_cap(J) is a 70-dimensioned array containing the packaged weight, in tons, for each of the 70 DIME elements. The packaged weight is constrained by the basic load for a weapon or the rate of fire of a weapon. If an element does not fire, then a zero value is assigned.

(5) Fuel_cap(J). Fuel_cap(J) is a 70-dimensioned array containing the capacity, in gallons, for those DIME elements carrying fuel. If an element does not carry fuel, then a zero value is assigned.

(6) Wpn_type(J). Wpn_type(J) is a 70-dimensioned array containing a pointer describing the type of ammunition being fired by the 70 elements on the DIME unit structure file. The entries are either: 1=direct fire (DF), 2=indirect fire (IF), or 3=air defense (AD). If a DIME element does not fire ammunition, then a value of 3 is assigned.

5. ALGORITHMS.

A. Figure 4-2 presents a generalized logic flow of the logistics program. The program first determines the total vehicle (weapon) capacity for both fuel and ammunition of an active unit. It then calculates combat

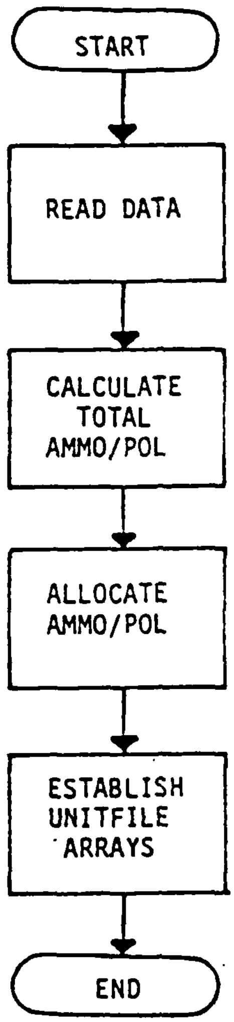


Figure 4-2. Logistics logic flow.

and noncombat fuel consumption as well as noncombat ammunition consumption. Totals of the supplies on hand are computed and then checked to see that the quantities of ammunition/fuel used do not exceed the amounts on hand. Consumption values within the "UNITFILE" are updated. Other figures calculated by the logistics program include ammunition/fuel status, quantities remaining, and truck/ground storage sums. The "UNITFILE" is updated as required. These calculations are performed for each active unit.

B. The following paragraphs provide a detailed description of the algorithms used in the calculation of fuel and ammunition values necessary for the running of the air defense and ground combat programs.

(1) Fuel calculations.

(a) Calculate the total fuel capacity of all weapons in gallons.

$$Ft = \sum_{i=1}^{70} Ne_i * Fc_i \qquad \text{(Eq. 4-1)}$$

where:

Ft = total fuel capacity.
Ne_i = number of elements of each weapon type i where i represents the 70 DIME weapons.
Fc_i = fuel capacity of each weapon type i.

(b) Calculate both combat and noncombat fuel consumption in gallons.

$$Fcon = \sum_{i=1}^{70} Ne_i * Ef_i \qquad \text{(Eq. 4-2)}$$

where:

Econ = total fuel consumed.
Ne_i = number of elements of each weapon type i.
Ef_i = fuel expended for combat and noncombat missions for each weapon type i.

(c) Calculate amount of fuel, in gallons, available for use at the beginning of the game turn.

$$Wf = \sum_{i=1}^{70} Ne_i * Fc_i * Fs \qquad \text{(Eq. 4-3a)}$$

$$Fava = Wf + Tf + Gf + Rf \qquad \text{(Eq. 4-3b)}$$

where:

- Wf = amount of fuel on the 70 DIME weapons.
- Ne_i = number of elements of each weapon type i.
- Fc_i = fuel capacity of each weapon type i.
- Fs = fuel status of unit weapons; value ranges from 0 to 1.0.
- Fava = fuel available for use.
- Tf = amount of fuel on trucks.
- Gf = amount of fuel on ground.
- Rf = amount of fuel resupplied.

(d) Calculate the quantity of fuel, in gallons, remaining after consumption and dispensation values are considered.

$$Frem = Fava - (Fcon + Df) \qquad \text{(Eq. 4-4)}$$

where:

- Frem = fuel remaining.
- Fava = fuel available for use; see (Eq. 4-3b) above.
- Fcon = fuel consumed; see (Eq. 4-2) above.
- Df = fuel dispensed to other units.

(2) Ammunition calculations.

(a) Calculate the ammunition capacity in tons of all weapons combined.

$$At_j = \sum_{i=1}^{70} Ne_{ij} * Ac_{ij} \qquad \text{(Eq. 4-5)}$$

where:

At_j = total ammunition capacity for weapon type i, ammunition category j where:
j = 1 direct fire
= 2 indirect fire
= 3 air defense.

Ne_{ij} = number of elements of each weapon type i, ammunition category j.

Ac_{ij} = ammunition capacity of each weapon type i, ammunition category j.

(b) Calculate noncombat ammunition consumption, in tons, for each category of direct fire (DF), indirect fire (IF) and air defense (AD).

$$Nac_j = \sum_{i=1}^{70} Ne_{ij} * Ea_{ij} \qquad \text{(Eq. 4-6)}$$

where:

Nac_j = noncombat DF, IF, or AD ammunition consumed.

Ne_{ij} = number of elements of each weapon type i, ammunition category j.

Ea_{ij} = ammunition expended for noncombat missions for each weapon type i, ammunition category j.

(c) Calculate the quantity of DF, IF, and AD ammunition available for use at the beginning of each game turn.

1. Calculate the amount of DF, IF, and AD ammunition loaded onto the 70 DIME elements.

$$Wa_j = \sum_{i=1}^{70} Ne_{ij} * Ac_{ij} * As_j \qquad \text{(Eq. 4-7a)}$$

where:

Wa_j = the total amount of ammunition loaded onto all weapon elements in ammo category j.

Ne_{ij} = number of elements of each weapon type i, ammunition category j.

Ac_{ij} = ammunition capacity of each weapon type i, ammunition category j.

As_j = ammunition status of each DF, IF or AD weapon. Value ranges from 0 to 1.0.

2. Calculate the amount of DF, IF, and AD ammunition loaded onto the cargo trucks.

$$Ta_j = Fl_j * At \qquad \text{(Eq. 4-7b)}$$

where:

Ta_j = the total amount of ammunition loaded onto all cargo trucks in ammo category j.
$F1_j$ = % of DF, IF, or AD ammunition on cargo vehicles.
At = quantity of ammunition on cargo vehicles.

3. Calculate the amount of DF, IF, and AD ammunition on the ground.

$$Ga_j = F2_j * Ag \quad \text{(Eq. 4-7c)}$$

where:

Ga_j = the total amount of ammunition on the ground in ammo category j.
$F2_j$ = % of DF, IF, or AD ammunition on the ground.
Ag = quantity of ammunition on the ground.

4. Calculate the amount of DF, IF, and AD ammunition resupplied (Ra_j).

$$Ra_j = F3_j * Ar \quad \text{(Eq. 4-7d)}$$

where:

$F3_j$ = % of DF, IF or AD ammunition resupplied
Ar = quantity of ammunition resupplied.

5. Calculate DF, IF, and AD ammunition available at game turn initialization.

$$Aava_j = Wa_j + Ta_j + Ga_j + Ra_j + Pa_j \qquad \text{(Eq. 4-7e)}$$

where:

Pa_j = amount of DF, IF, or AD ammunition left from previous turn.

(d) Calculate DF, IF, and AD ammunition remaining after consumption and dispensation values are considered.

$$Da_j = F4_j * Ad \qquad \text{(Eq. 4-8a)}$$

$$Arem_j = Aava_j - (Nac_j + Da_j) \qquad \text{(Eq. 4-8b)}$$

where:

Da_j = quantity of DF, IF, or AD ammunition dispensed to other units.
$F4_j$ = % of DF, IF, or AD ammunition dispensed to other units.
Ad = total ammunition dispensed to other units.
$Arem_j$ = ammunition remaining.
Aav_j = DF, IF, or AD ammunition available for use. See (Eq. 4-7e) above.
Nac = noncombat DF, IF, or AD ammunition consumed. See(Eq. 4-6) above.

6. "UNITFILE" IMPACT.

In order to fully understand the function of the logistics program within the DIME framework, it is necessary to have an understanding of the relationships existing between it, the air defense program (P3), the ground combat program (P4), and the unit status report program (P8). The following paragraphs describe these relationships.

A. The logistics program (P2) is executed once at the beginning of each turn. It calculates the available quantities of all ammunition and fuel and the net amounts resulting from leakage, dispensation, and resupply. It should be noted that fuel consumption values include combat as well as noncombat expenditures. This means that P2 is the only program in DIME to calculate fuel consumption values. From the net ammunition amount, P2 calculates the percent of direct fire, indirect fire, and air defense ammunition 'loaded' onto the weapons and trucks. The remainder is placed on the ground. The net DF, IF, and AD amounts are placed in positions 131- 133 of the "UNITFILE" to be consumed and/or destroyed during the execution of the air defense (P3) and ground combat (P4) programs.

B. The air defense (P3) and ground combat (P4) programs may be executed zero to N times, N being constrained only by ammunition availability. When each battle is over, both P3 and P4 place the unused fuel and ammunition back in positions 131-133 of the "UNITFILE."

C. The unit status report (P8) is executed once at the end of each game turn. It compares the remaining quantities of DF, IF, and AD ammunition placed in the "UNITFILE" by P3 and P4 with the combat and noncombat ammunition quantities available at the beginning of the turn, positions 116 - 118 of the "UNITFILE", in order to determine consumption values. It then redistributes the ammunition to the weapons, trucks, and onto the ground in a similar manner to that of P2. P8 then makes this information available to the gamer in the form of a unit history. The unit history also includes the number of systems remaining, unit mission, and detection status.

D. Because P2 calculates the noncombat consumption, resupply, and dispensation values of DF, IF, and AD ammunition, it must reevaluate the quantities P8 distributes among the weapons, trucks, and ground storage in the preceding game turn. It then allocates appropriate levels of ammunition for the execution of P3 and P4 in the current game turn.

E. Table 4-2 contains a list of all "UNITFILE" elements updated by the logistics program. Each element's location within the "UNITFILE" and a short description are furnished. The elements are listed in ascending order according to their locations within the "UNITFILE".

7. **CODE.**

A. The logistics code is written as one program without major subroutines. The code initiates the algorithms discussed in paragraph 5 of this chapter. After the appropriate Blue/Red data is read, the following will occur with each unit processed:

(1) Position 82 of the "UNITFILE" is tested for zero. If it is zero, the unit is inactive and logistics processing is not done for the unit.

(2) Calculations occur for active units which determine fuel supplies on hand, fuel supplies consumed (non-combat), and the current vehicle fuel capacity. Similar calculations occur for ammunition.

(3) Ammunition quantities remaining after non-combat consumption and dispensation to other units are tested. If the amounts consumed and dispensed exceed that which was on hand, a message is printed to make the controllers/gamers aware of this occurrence. The amount remaining is then set to zero.

Table 4-2. Logistics "UNITFILE" updates.

"UNITFILE" locations	Element description
84	Cargo trucks alive at start of turn.
85	Fuel trucks alive at start of turn.
86	JP4 trucks alive at start of turn.
101	Fuel status of unit vehicles; value ranges from 0 to 1.0.
102	JP4 status of unit vehicles; value ranges from 0 to 1.0.
103	Fuel on tankers (gallons).
104	JP4 on tankers (gallons).
105	Fuel on ground (gallons).
106	JP4 on ground (gallons).
108	Fuel consumed during the current game turn (gallons).
109	JP4 consumed during the current game turn
115	Helicopter ammunition at start of turn (short tons).
116 - 118	DF, IF and AD ammunition at start of turn (short tons).
119 - 121	DF, IF and AD vehicle ammunition status; values range from 0 to 1.0.
122	Helicopter ammunition status; values range from 0 to 1.0.
123	Ammunition on cargo vehicles (short tons).

Table 4-2. Logistics "UNITFILE" updates (concluded).

"UNITFILE" locations	Element description
124	Distribution of cargo by type (XXX.YYY).*
125	Ammunition on ground (short tons).
126	Distribution of ground ammunition by type (XXX.YYY).*
131 - 133	DF, IF and AD ammunition available for consumption (short tons).
134	Helicopter ammunition available for consumption (short tons).
139 - 141	Cumulative DF, IF, and AD ammunition consumed to date (short tons).
142	Cumulative helicopter ammunition consumed to date (short tons).
143	Cumulative fuel consumed to date (gallons).
144	Cumulative JP4 consumed to date (gallons).
147	Fuel left after consumption and dispensation values are figured.
148	JP4 left after consumption and dispensation values are figured.

* Refer to Table 4-1 of this chapter for a detailed description.

(4) The status for ammunition and fuel are calculated. The status amounts are calculated by dividing the remaining amounts (after non-combat consumption and dispensation) by the total vehicle capacity. Excess fuel and ammunition are calculated and stored on the ground for use during combat. This excess is determined by any fuel and ammunition which was not consumed, dispensed, or placed on vehicles.

B. Table 4-3 provides a summary of the logistics program's primary variables and their description. A listing of P2 code appears in Table 4-4.

Table 4-3. Logistics Subroutine Table.

Functional area(s): A. Initialize supply variables and read data

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Main program	Calls subroutines	A. N(*)	150 element array which holds data provided by the UNITFILE.
		B. Ammol	Holding variable for the noncombat consumption of ammo in tons.
		C. Fuell	Holding variable for the combat and noncombat consumption of fuel in gallons.
Blue_data	Reads Blue logistics data	A. Fuel (I,J)	Consumption of fuel in gallons per 6 hours for 70 elements under 10 missions: Missions: 0 = combat unit 1 = artillery unit 2 = air defense unit 3 = AH Ground Sites/FAARP 4 = CP/HQ 5 = Engineer Unit 6 = POL/Ammo Supply pt 7 = Maintenance pt 8 = SAM site 9 = Commo/Radar/EW site
		B. Ammo (I,J)	Noncombat consumption of ammo in tons per 6 hours for J=1-70 elements operating under I=1-10 missions (see above). These values represent leakage within the system.

Table 4-3. Logistics Subroutine Table.

Functional area(s): A. Initialize supply variables and read data (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Blue_data (continued)		C. Truck_cap(*)	Contains the carrying capacity in gallons/tons for a single fuel/single cargo truck.
		D. Ammo_cap(*)	Contains the packaged weight, in tons, for each of the 70 DIME elements. The packaged weight is constrained by: (1) basic load for weapon or (2) rate of fire of weapon. If an element does not fire, then a zero (0) value is assigned. NOTE: The values contained in Ammo_cap(*) must correspond with the weight files contained within the Operations Module (P1), the Ground Attrition Module (P4), and the Air Defense Module (P3).
		E. Fuel_cap(*)	Contains the capacity in gallons for those DIME elements carrying fuel. If an element does not carry fuel, then a zero (0) value is assigned. NOTE: The values contained in Fuel_cap(*) must correspond with the arrays F(1,J) and F(2,J) contained within the Operations Module, P1.

Table 4-3. Logistics Subroutine Table.

Functional area(s): A. Initialize supply variables and read data (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Blue_data (concluded)		F. Wpn_type(*)	Contains a pointer file describing the type of ammunition being fired by the 70 elements on the DIME unit structure file. The entries are either: (1) DF, (2) indirect fire, or (3) air defense. If a DIME element does not fire ammunition then a value of one (1) is assigned. NOTE: The values contained in Wpn_type(*) must have the same values as the arrays W(1,J) and W(2,J) contained within the Operations Module (P1).
Red_data	Reads Red logistics data	See Blue data above	
Init_unit_var	Initializes unit supply variables	A. Tot_veh_fuel	Contains the total fuel carrying capacity in gallons for the unit vehicles (weapons).
		B. Fuel_used	Contains the total quantity of POL in gallons consumed for combat and noncombat purposes.
		C. Fuel_on_hand	Contains the total quantity of POL available for combat and noncombat consumption at start of turn.
		D. Fuel_left	Excess fuel remaining after combat and noncombat consumption is determined.

Table 4-3. Logistics Subroutine Table.

Functional area(s): A. Initialize supply variables and read data (concluded)

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Init_unit_var (concluded)		E. Ammo_total	Total quantity of DF, IF, and AD ammo in tons remaining after noncombat consumption is determined.
		F. Load_cap	Variable representing the quantity of fuel in gallons loaded into the fuel trucks.
		G. Ammo_on_hand(*)	Three element array containing the quantities of DF, IF, and AD ammo in tons available at start of turn.
		H. Free(*)	Three element array which holds percents of DF, IF, and AD ammo used in the calculation of ammo on hand.
		I. Tot_veh_ammo(*)	A 3 element array which contains the total DF, IF and AD ammo capacities for the 70 DIME unit vehicles (weapons).
		J. Ammo_used(*)	Contains the quantities of DF, IF and AD ammo consumed by non-combat processes.
		K. Ammo_left(*)	Contains the DF, IF and AD ammo quantities remaining after non-combat consumption and dispensation values are subtracted.

Table 4-3. Logistics Subroutine Table.

Functional area(s): B. Calculation of ammunition and POL levels.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Main program	Determines ammo/POL available for consumption	See Functional area A variable lists. non-combat consumption values, and combat fuel expenditures.	

Functional area(s): C. Allocation of ammunition and POL.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Main program	Distributes ammo/POL into the trucks and	See Functional Area A variable lists.	

Functional area(s): D. Establish UNITFILE Arrays.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Main program	Updates UNITFILE elements for both Blue and Red forces.	See Functional Area A variable lists.	

Table 4-4. Logistics code.

```
10     REM    "P2" IS THE LOGISTICS PROGRAM FOR DIME
20     ! DATA CHANGED 21 FEB 85, ROB BELFLOWER   PLAYS 400 UNITS
30     ! EXPANDED VERSION -- JUNE 9, 1986 -- BY OAO CORP.
31     !      DECLASSIFIED -- AUG 7, 1986 -- BY OAO CORP.  ** DC **
40     OPTION BASE 1
50     !
60     Disk$=":9134,704,0"
64     Dcdisk$=":9134,704,0"   ! ** DC **
70     PRINTER IS 1
80     DIM N(150),Fuel(10,70),Ammo(10,70),Fuel_cap(70),Ammo_cap(70),Frac(3)
90     DIM Wpn_type(70),Ammo_used(3),Ammo_on_hand(3),Tot_veh_ammo(3),Truck_cap(7
100    DIM Ammo_left(3)
101    DIM Dataline(70),Dummyrec(70)    ! ** DC **
102    INTEGER Element,Recnum  ! ** DC **
110    INTEGER I,J,K
120    PRINTER IS 1
130    PRINT USING "@,#"
140    PRINT TABXY(30,19);"LOGISTICS MODULE"
150    INPUT "DO YOU WANT TO DO POST-GAMETURN RESUPPLY <R> OR DO YOU WANT TO RUN
P2 <P>?",Ans$
160    IF Ans$="R" THEN
170      PRINT "LOADING DATA OPERATIONS MENU...PLEASE BE PATIENT."
180      LOAD "P1:9134,704,0"
190    END IF
200    INPUT "HOW MANY HOURS OF SUPPLY DO YOU WANT TO RUN?",Hour
210   !IF Hour<6 THEN Hour=6
220    ASSIGN @Path TO "UNITFILE"&Disk$
230    ASSIGN @Pname TO "NAMEFILE"&Disk$
240    !
250    GOSUB Blue_data                                   !  READ BLUE LOG DATA
260    !
270    !  BEGIN UNIT PROCESSING LOOP
280    !
290    PRINT USING "@,#"
300    FOR I=1 TO 400
310      ENTER @Pname,I;M$
320      IF I=192 THEN GOSUB Red_data                     !  READ RED LOG DATA
330      ENTER @Path,I;N(*)
340      IF M$="          " THEN GOTO Next_unit
350      IF N(82)=0 OR N(82)=2 THEN                          !  UNIT IS IN
CTIVE
360        PRINT USING "3D,2X,8A";I,"INACTIVE"
370        GOTO Next_unit
380      END IF
390      GOSUB Init_unit_var                              !  ZERO UNIT VARIABLES
400      PRINT USING "3D,2X,10A";I,"PROCESSING"
410    !
420      Ipt1=N(83)
430    !
440    !   COMPUTE SUPPLIES ON HAND, CONSUMED, AND TOTAL VEHICLE CAPACITY
450      FOR J=1 TO 70
460        Tot_veh_fuel=Tot_veh_fuel+N(J)*Fuel_cap(J)
```

Table 4-4. Logistics code.

```
470       Tot_veh_ammo(Wpn_type(J))=Tot_veh_ammo(Wpn_type(J))+N(J)*Ammo_cap(J)
480      ! Fuel1=(Fuel(Ipt1,J)) !FUEL USE BY 3-HOURS ROB
490       FOR X=1 TO 2
500         X1st=INT(Ipt1)
510         X2nd=(Ipt1-X1st)*10
520         IF X=1 THEN Fuel1=(Fuel(X1st+1.J))
530         IF X=2 THEN Fuel1=Fuel1+(Fuel(X2nd+1.J))
540       NEXT X
550       Fuel1=(Fuel1/2)*INT(Hour/6) !ROB
560       Fuel_used=Fuel_used+N(J)*Fuel1
570       Fuel_on_hand=Fuel_on_hand+N(J)*Fuel_cap(J)*N(101)
580      !Ammo1=(Ammo(Ipt1,J))   ROB
590       Ammo1=0!ROB
600       Ammo_used(Wpn_type(J))=Ammo_used(Wpn_type(J))+N(J)*Ammo1
610       Ammo_on_hand(Wpn_type(J))=Ammo_on_hand(Wpn_type(J))+N(J)*Ammo_cap(J)*
(118+Wpn_type(J))
620     NEXT J
630    !
640    !      COMPUTE TOTAL OF SUPPLIES ON HAND
650     Fuel_on_hand=Fuel_on_hand+N(103)+N(105)+N(110)          ! TOTAL FUEL IN
NIT
660    !      ADD TRUCK CARRIED AMMO
670     Frac(1)=INT(N(124))/1000
680     Frac(2)=N(124)-INT(N(124))
690     Frac(3)=1-(Frac(1)+Frac(2))
700     IF Frac(3)<=0 THEN Frac(3)=0
710     FOR J=1 TO 3
720       Ammo_on_hand(J)=Ammo_on_hand(J)+Frac(J)*N(123)
730     NEXT J
740    !
750    !      ADD GROUND STORED SUPPLIES
760     Frac(1)=INT(N(126))/1000
770     Frac(2)=N(126)-INT(N(126))
780     Frac(3)=1-(Frac(1)+Frac(2))
790     IF Frac(3)<=0 THEN Frac(3)=0
800     FOR J=1 TO 3
810       Ammo_on_hand(J)=Ammo_on_hand(J)+Frac(J)*N(125)
820     NEXT J
830    !
840    !      ADD RESUPPLY
850     Frac(1)=INT(N(136))/1000
860     Frac(2)=N(136)-INT(N(136))
870     Frac(3)=1-(Frac(1)+Frac(2))
880     IF Frac(3)<=0 THEN Frac(3)=0
890     FOR J=1 TO 3
900       Ammo_on_hand(J)=Ammo_on_hand(J)+Frac(J)*N(135)
910     NEXT J
920    !
930    !ADD LEFT FROM PREVIOUS BATTLE
940     FOR J=1 TO 3
950       Ammo_on_hand(J)=Ammo_on_hand(J)+N(130+J)
960     NEXT J
```

Table 4-4. Logistics code.

```
970    '
980    '    CHECK THAT AMOUNT FIRED/USED DOES NOT EXCEED ON-HAND AMOUNT
990    Frac(1)=INT(N(138))/1000
1000   Frac(2)=N(138)-INT(N(138))
1010   Frac(3)=1-(Frac(1)+Frac(2))
1020   IF Frac(3)<=0 THEN Frac(3)=0
1030   FOR J=1 TO 3
1040     Ammo_left(J)=Ammo_on_hand(J)-(Ammo_used(J)+Frac(J)*N(137))
1050     IF Ammo_left(J)<0 THEN
1060       SELECT J
1070       CASE 1
1080         Ammotype$="DF"
1090       CASE 2
1100         Ammotype$="IF"
1110       CASE 3
1120         Ammotype$="AD"
1130       END SELECT
1140       PRINTER IS 702
1150       PRINT
1160       PRINT "*** UNIT ";M$;"(";I;")";" HAS DISPENSED/USED ";-Ammo_left(J
1170       PRINT "         MORE ";Ammotype$;" TONS THAN AVAILABLE!"
1180       PRINTER IS 1
1190       Ammo_used(J)=Ammo_on_hand(J)
1200       Ammo_left(J)=0
1210     END IF
1220   NEXT J
1230   Fuel_left=Fuel_on_hand-(Fuel_used+N(112))
1240   IF Fuel_left<0 THEN
1250     PRINTER IS 702
1260     PRINT
1270     PRINT "*** UNIT ";M$;"(";I;")";" HAS DISPENSED/USED ";-Fuel_left
1280     PRINT "         MORE GALLONS OF FUEL THAN AVAILABLE!"
1290     PRINTER IS 1
1300     Fuel_used=Fuel_on_hand
1310     Fuel_left=0
1320   END IF
1330   !
1340   N(147)=Fuel_left
1350   !
1360   !    UPDATE CONSUMPTION RECORD
1370   FOR J=1 TO 3
1380     N(130+J)=Ammo_left(J)
1390     N(138+J)=Ammo_used(J)+N(138+J)
1400     N(115+J)=Ammo_on_hand(J)
1410   NEXT J
1420   N(108)=Fuel_used
1430   N(143)=N(143)+Fuel_used
1440   '
1450   !    CALCULATE FUEL STATUS AND REMAINING FUEL
1460   IF Tot_veh_fuel<1 THEN 1550
1470   IF Tot_veh_fuel>=Fuel_left THEN
1480     N(101)=Fuel_left/Tot_veh_fuel
```

Table 4-4. Logistics code.

```
1490      Fuel_left=0
1500    ELSE
1510      N(101)=1
1520      Fuel_left=Fuel_left-Tot_veh_fuel
1530    END IF
1540  !
1550  !    CALCULATE AMMO STATUS AND REMAINING AMMO
1560    FOR J=1 TO 3
1570      IF Tot_veh_ammo(J)<=0 THEN
1580        N(118+J)=0
1590        GOTO 1680
1600      END IF
1610      IF Tot_veh_ammo(J)>=Ammo_left(J) THEN
1620        N(118+J)=Ammo_left(J)/Tot_veh_ammo(J)
1630        Ammo_left(J)=0
1640      ELSE
1650        N(118+J)=1
1660        Ammo_left(J)=Ammo_left(J)-Tot_veh_ammo(J)
1670      END IF
1680    NEXT J
1690  !
1700  !    CALCULATE TRUCK/GROUND FUEL STORAGE
1710    IF Fuel_left>0 THEN
1720      Load_cap=N(55)*Truck_cap(1)
1730      IF Load_cap>Fuel_left THEN
1740        N(103)=Fuel_left                                   !  FUEL ON TANKERS
1750        N(105)=0                                           !  FUEL ON GROUND
1760      ELSE
1770        N(103)=Load_cap
1780        N(105)=Fuel_left-Load_cap
1790      END IF
1800    ELSE
1810      N(103)=0
1820      N(105)=0
1830    END IF
1840  !
1850    N(85)=N(55)
1860    IF N(146)>0 THEN N(105)=0  ! DUMPS FUEL ON GROUND FOR MOVING UNIT
1870  !    CALCULATE TRUCK/GROUND AMMO STORAGE
1880    Ammo_total=Ammo_left(1)+Ammo_left(2)+Ammo_left(3)
1890    IF Ammo_total>0 THEN
1900      S1=Ammo_left(1)/Ammo_total*1000
1910      S2=Ammo_left(2)/Ammo_total
1920      N(124)=INT(S1)+S2
1930      N(126)=N(124)
1940      Load_cap=N(58)*Truck_cap(4)
1950      IF Ammo_total<=Load_cap THEN
1960        N(123)=Ammo_total
1970        N(125)=0
1980        N(126)=0
1990      ELSE
2000        N(123)=Load_cap
```

Table 4-4. Logistics code.

```
2010        N(125)=Ammo_total-Load_cap
2020      END IF
2030    ELSE
2040      N(123)=0                                        ! AMMO ON CGO VEHICLES
2050      N(124)=0                                        ! DIST OF AMMO ON CGO VEH
2060      N(125)=0                                        ! AMMO STORED ON GROUND
2070      N(126)=0                                        ! DIST OF AMMO ON GROUND
2080    END IF
2090    N(84)=N(58)
2100    IF N(146)>0 THEN N(125)=0 ! DUMPS AMMO ON GROUND FOR MOVING UNIT
2110   !
2120 Write_out:  !   WRITE OUT TO UNIT FILE
2130    OUTPUT @Path,I;N(*)
2140    PRINTER IS 1
2150 Next_unit: !  END OF UNIT PROCESSING
2160   NEXT I
2170   !
2180   !
2190   !
2200   ASSIGN @Path TO *
2210   PRINT
2220   PRINT
2230   PRINT "LOGISTICS PROCESSING COMPLETED "
2240   LOAD "DIME"&Disk$
2250   GOTO Halt
2260   !
2270   !
2280   ! ******************** END OF MAIN PROGRAM ******************************
2290   !
2300   !
2310 Blue_data:  !   THIS SBR HOLDS BLUE LOGISTICS DATA
2320   !
2321   ! ** DC **  7 AUG 1986
2322   !
2330   ASSIGN @Pammouse TO "BLAMMO_USE"&Dcdisk$
2331   FOR Recnum=1 TO 10
2332       ENTER @Pammouse,Recnum;Dataline(*)
2333       FOR Element=1 TO 70
2334           Ammo(Recnum,Element)=Dataline(Element)
2335       NEXT Element
2336   NEXT Recnum
2350   ASSIGN @Pammouse TO *
2360   !
2370   ASSIGN @Pfueluse TO "BLFUEL_USE"&Dcdisk$
2380   FOR Recnum=1 TO 10
2381       ENTER @Pfueluse,Recnum;Dataline(*)
2382       FOR Element=1 TO 70
2383           Fuel(Recnum,Element)=Dataline(Element)
2384       NEXT Element
2385   NEXT Recnum
2390   ASSIGN @Pfueluse TO *
2400   !
```

Table 4-4. Logistics code.

```
3760  ASSIGN @Ptrkcap TO "BL_TRK_CAP"&Dcdisk$
3770  ENTER @Ptrkcap,1;Truck_cap(*)
3780  ASSIGN @Ptrkcap TO *
3786  !
3790  ASSIGN @Pammocap TO "AMMO_CAP"&Dcdisk$
3800  ENTER @Pammocap,1;Ammo_cap(*)
3810  ASSIGN @Pammocap TO *
3820  !
3830  ASSIGN @Pfuelcap TO "FUEL_CAP"&Dcdisk$
3840  ENTER @Pfuelcap,1;Fuel_cap(*)
3850  ASSIGN @Pfuelcap TO *
3860  !
3870  ASSIGN @Pwpntyp TO "WPN_TYP"&Dcdisk$
3880  ENTER @Pwpntyp,1;Wpn_type(*)
3890  ASSIGN @Pwpntyp TO *
3900  ! ** END DC **
4070  !
4080  RETURN
4090  !
4100  ! **********************************************************************
4110  !
4120 Red_data:  !  THIS SBR HOLDS RED LOGISTICS DATA
4130  !
4140  ASSIGN @Pammouse TO "RDAMMO_USE"&Dcdisk$
4150  FOR Recnum=1 TO 10
4151      ENTER @Pammouse,Recnum;Dataline(*)
4152      FOR Element=1 TO 70
4153          Ammo(Recnum,Element)=Dataline(Element)
4154      NEXT Element
4155  NEXT Recnum
4160  ASSIGN @Pammouse TO *
4170  !
4860  ASSIGN @Pfueluse TO "RDFUEL_USE"&Dcdisk$
4870  FOR Recnum=1 TO 10
4871      ENTER @Pfueluse,Recnum;Dataline(*)
4872      FOR Element=1 TO 70
4873          Fuel(Recnum,Element)=Dataline(Element)
4874      NEXT Element
4875  NEXT Recnum
4880  ASSIGN @Pfueluse TO *
4890  !
4900  ASSIGN @Ptrkcap TO "RD_TRK_CAP"&Dcdisk$
5590  ENTER @Ptrkcap,1;Truck_cap(*)
5600  ASSIGN @Ptrkcap TO *
5610  !
5611  ASSIGN @Pammocap TO "AMMO_CAP"&Dcdisk$
5620  ENTER @Pammocap,1;Dummyrec(*),Ammo_cap(*)
5630  ASSIGN @Pammocap TO *
5640  !
5700  ASSIGN @Pfuelcap TO "FUEL_CAP"&Dcdisk$
5710  ENTER @Pfuelcap,1;Dummyrec(*),Fuel_cap(*)
5720  ASSIGN @Pfuelcap TO *
```

Table 4-4. Logistics code.

```
5730  !
5790  ASSIGN @Pwpntyp TO "WPN_TYP"&Dcdisk$
5800  ENTER @Pwpntyp,1;Dummyrec(*),Wpn_type(*)
5810  ASSIGN @Pwpntyp TO *
5811  !
5880  ! ** END DC **
5890  !
5900  RETURN
5910  !
5920  ! ***********************************************************************
5930  !
5940 Init_unit_var:  !  THIS SBR INITIALIZES UNIT SUPPLY VARIABLES
5950  !
5960  Tot_veh_fuel=0
5970  Fuel_used=0
5980  Fuel_on_hand=0
5990  Fuel_left=0
6000  Ammo_total=0
6010  Load_cap=0
6020  FOR J=1 TO 3
6030    Ammo_on_hand(J)=0
6040    Frac(J)=0
6050    Tot_veh_ammo(J)=0
6060    Ammo_used(J)=0
6070    Ammo_left(J)=0
6080  NEXT J
6090  RETURN
6100  !
6110  ! ***********************************************************************
6120 Halt:END
```

CHAPTER 5

AIR ATTACK/AIR DEFENSE

1. PURPOSE.

The purpose of the DIME air defense program is to realistically game air-to-ground and ground-to-air interactions between ground forces and opposing aircraft. Four types of aircraft and 70 types of ground elements, which include four types of air defense weapon systems, are played.

2. GENERAL.

A. Both Red and Blue forces are given appropriate capabilities and limitations in order to realistically mirror current and future equipment, vehicles, weapons, aircraft, and munitions.

B. Air operations are separated into two phases: ingress/egress and air strike.

(1) The ingress/egress phase calculates the attrition suffered by aircraft which fly over previously undetected air defense units.

(2) The air strike phase calculates the attrition in both aircraft and ground elements resulting from interdiction at the target area.

C. Program characteristics include:

(1) Four aircraft types, two fixed wing and two helicopters, are played for each force (Red and Blue).

(2) The following four generic air defense types are played:

(a) Man-portable, shoulder-fired missile system.

(b) Self-propelled, low-altitude guided missile system.

(c) Self-propelled gun system.

(d) Low to medium altitude over watch missile system.

(3) Aircraft losses during ingress and egress are calculated using the methodology from the Air Defense Air-to-Ground Engagement (ADAGE) model.

(4) Aircraft and ground losses during the strike phase are calculated using methodology from the Joint Munitions Effectiveness Manual (JMEM) and ADAGE.

(5) Aircraft have multiple munition loads including cluster munitions, guided missiles, area rockets, and guns.

(6) Aircraft loads are tailored to the expected target types.

(7) Attrition to both ground targets and aircraft is driven by delivery profiles for aircraft munitions.

(8) Air strikes are terminated by one of the following:

(a) Delivery of all aircraft munitions.

(b) Aircraft losses exceed threshold.

(c) Perceived ground losses exceed threshold.

(9) Mounted infantry casualties are assessed in proportion to carrier vehicle losses.

3. DATA FLOW.

The air defense program is dependent upon inputs from the user. Blue or Red aircraft are first chosen. It is then decided whether those aircraft will fly ingress, strike, or egress. Figure 5-1 shows the data flow.

A. Ingress/egress inputs.

(1) The number of aircraft making the flight.

(2) The type of aircraft:

Input	Blue	Red
1	A10	M28 (Frogfoot)
2	F16	M27 (Flogger)
3	UH60	Halo
4	AH64	Hind

(3) Altitude, in feet, which the aircraft is flying.

(4) Number of ground units overflown by aircraft.

(5) CAS/BAI. The two types of aircraft flights possible are close air support and battlefield air interdiction.

(6) High/low density. This input represents the density of the air defense systems as either high or low.

(7) Unit identification. The unit number(s), up to a maximum of five, of the enemy unit(s) overflown.

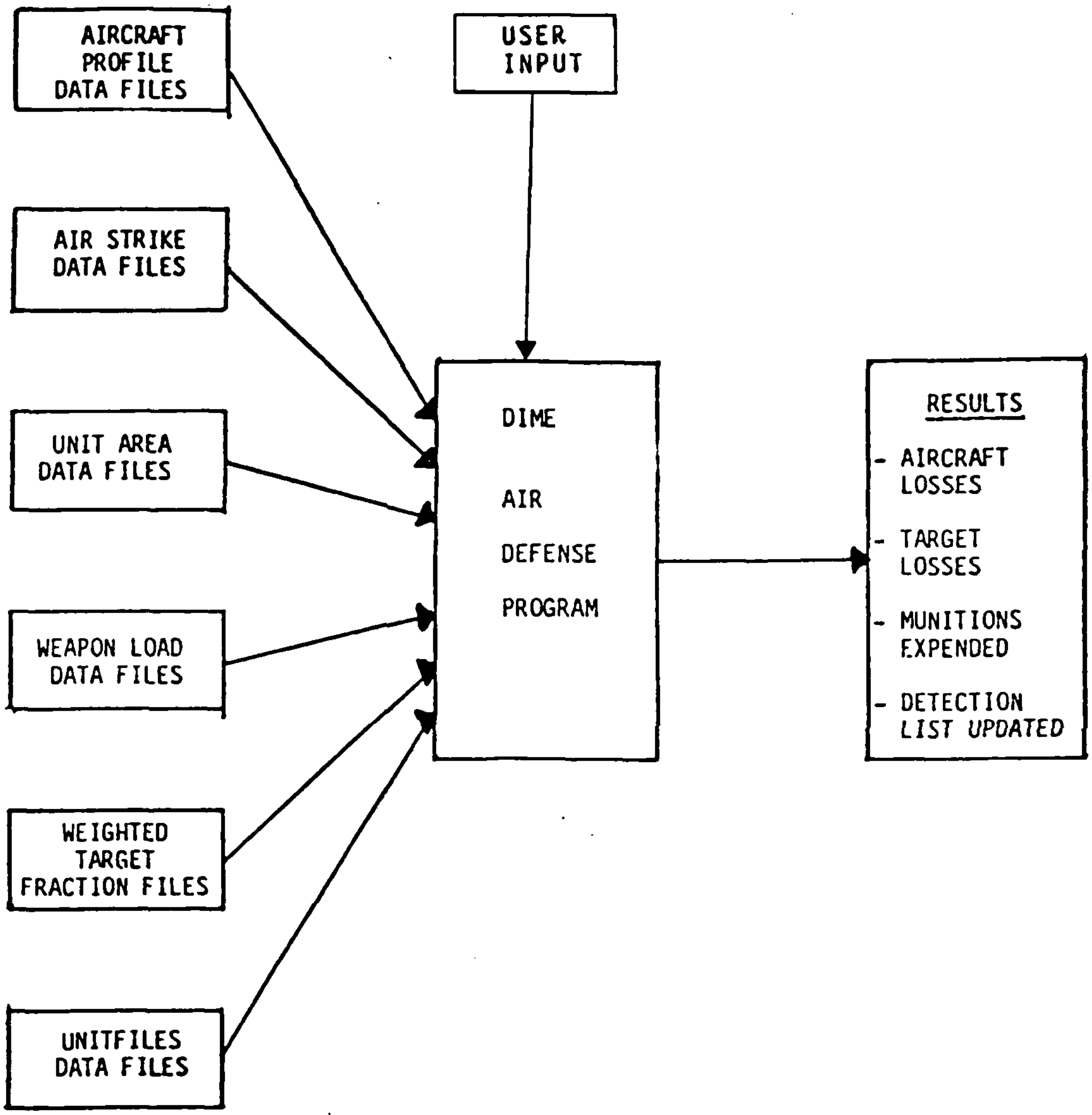


Figure 5-1. Air defense information flow.

(8) Terrain. Corresponding terrain occupied by each unit overflown.

1 = Open
2 = Rolling
3 = Hilly
4 = Mountainous

B. <u>Strike inputs.</u>

(1) The number of aircraft making the strike.

(2) Aircraft type.

(3) Aircraft mission.

1 = Bridge
2 = Antiarmor
3 = Antipersonnel
4 = Strike POL point
5 = Strike AMMO point

(4) Aircraft break point. The maximum percentage of attrition the aircraft will incur before breaking the attack.

(5) The percent of the target unit in the open (having limited overhead protective cover).

(6) CAS/BAI.

(7) High/low density.

(8) Target length. The length dimension of the area occupied by the target unit(s) in meters.

(9) Target width. The width of the target in meters.

(10) Target posture. The strategic (major) mission of the target.

1 = Attack
2 = Defend
3 = Reserve
4 = Move

(11) Target type. The unit type of the target.

0 = Combat unit
1 = Artillery unit
2 = Air defense unit
3 = Attack helicopter ground/forward rearming and refueling point (FARP)

4 = Command post/headquarters (CP/HQ)
5 = Engineer unit
6 = POL/AMMO supply point
7 = Maintenance point
8 = Bridge
9 = Communications/radar/electronic warfare (EW) sites.

(12) Number of targeted units.

(13) Unit identification. The unit number(s), up to a maximum of five, of the enemy unit(s) targeted.

(14) Terrain.

(15) Percent targeted. The percentage of each unit targeted depending on air intelligence, target location, camouflage of target, operational mission, and terrain.

C. Additional options beside ingress, egress, and strike, include:

(1) Status report. This report may be used to print information or help debug any changes to the program. Options available are as follows:

(a) List current results.

(b) List Blue air cumulative results.

(c) List Red air cumulative results.

(d) List debug unit information.

(e) List debug flight information.

(f) List debug strike information.

(2) Update results. This option allows the choice of storing the information from the current ingress, egress, or strike flight or purging its information to enable rerunning.

(3) Exit module. If return to the DIME driver is desired, the exit option is chosen. A prompt then appears to ensure updates to files. "Reminder to save final results (Y/N)" is answered N if an update had been accomplished and will continue exiting from the program.

4. FILE STRUCTURE.

DIME air defense data files are used by the program to describe aircraft performance, weapons and munitions capabilities, and target characteristics. The data are stored on random access files as shown in Figure 5-2.

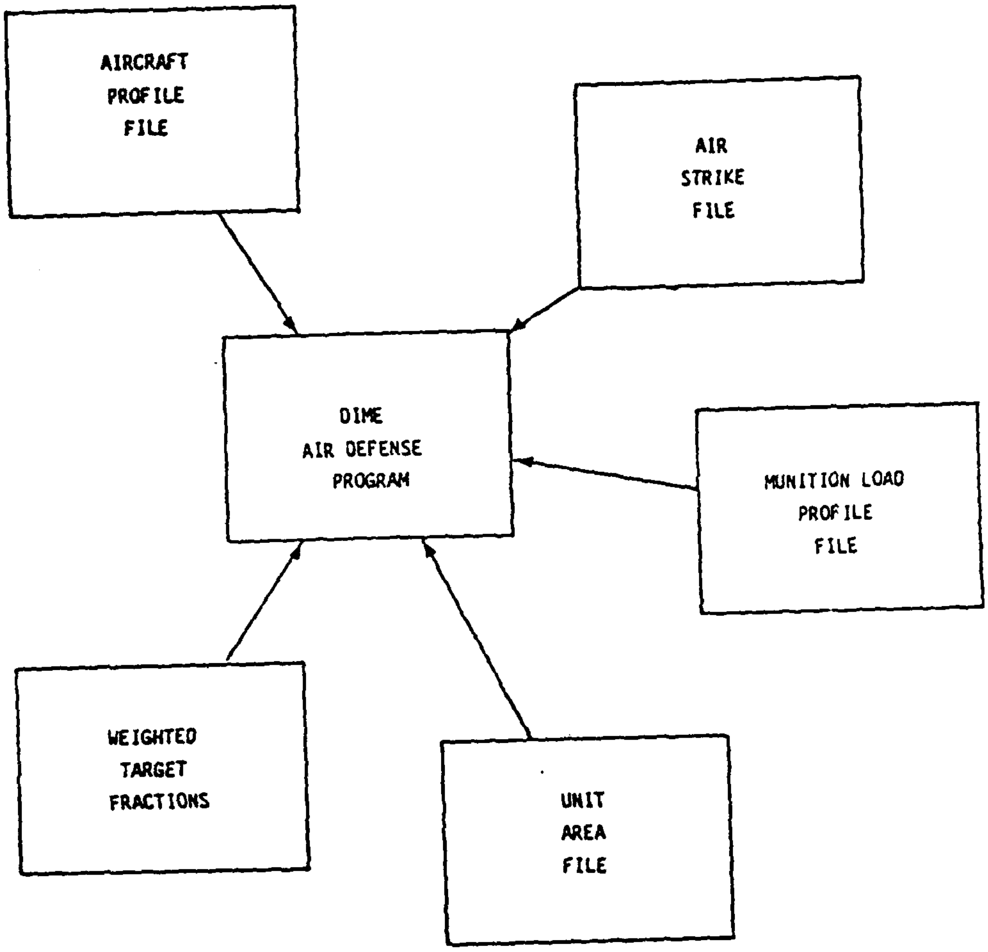


Figure 5-2. Air defense data bases.

A. Aircraft profile files.

(1) The aircraft profile files contain data describing air defense weapon characteristics against a specific type aircraft. Two files exist for each aircraft played, one for ingress and one for egress.

(2) The structure of the random access file is as follows. Air defense (ADA) weapons represent positions 48 - 54 of the "UNITFILE".

Array Name: Flt_profile(*). File Name: The file name is based on the aircraft type and whether the aircraft is ingressing or egressing (example: A10_ING).

File Positions	Description
1 through 7	Minimum altitude (meters) that the 7 ADA weapons can engage the aircraft.
8 through 14	Maximum altitude (meters) that the 7 ADA weapons can engage the aircraft.
15 through 21	Radius of effects for range of the 7 ADA weapons to engage the aircraft in meters.
22 through 28	Probability of participation of 7 ADA weapons against the aircraft.
29 through 35	Number of rounds/basic load for each of the 7 ADA weapons.
36 through 42	Weight (pounds) of one round for the 7 ADA weapons.
43 through 49	Number of rounds fired/engagement for the 7 ADA weapons against the aircraft.
50 through 56	Probability of five-minute survivability for aircraft against ADA weapons.

B. Air strike files.

(1) The air strike files contain data describing aircraft performance against each of the 70 target element types. Also included is data describing air defense weapon characteristics against the aircraft with its particular flight profile driven by the type of munition it is carrying. Multiple files exist for each aircraft type, one for each type of aircraft-delivered munition used.

(2) The structure of the random access files is as follows. ADA weapons represent positions 48 - 54 on the "UNITFILE".

Array Name: Stk_profile (*)
File Name: Aircraft type-munitions load: (example: A10 MK20)

File Position	Description
1	Code 1 = Bomb 2 = Missile 3 = Gun
2	Effective lethal area of principal target for above munition type (meters squared; m^2).
3	Minimum number of passes to deliver the above munition; for a gun, the maximum number of passes.
4	Rounds delivered per engagement. This consists of one for a guided missile ordnance load and guns dropped from bombs, given a target is engaged.
5	Number of rounds/basic load for this ammunition type carried by aircraft.
6 - 9	Probability of detecting the principal target per pass given the target is exposed for attack, defend, move, and reserve.
10 - 13	Probability that aircraft has line of site (LOS) with principal target in the woods for attack, defend, move, and reservc.
14 - 83	Probability that the target element in the lethal area will be destroyed by munition type. For point fire, single-shot PK are used.
84	Probability that bridge is destroyed.
85	Probability of ammunition (AMMO) point destruction.

File Position	Description
86	Probability of POL point destruction.
87 - 156	Effective lethal area for each element in DIME which corresponds to above target element PKs for munition (m^2).
157	Effective lethal area for a bridge (m^2).
158	Effective lethal area for an AMMO point (m^2). Note: Use specific target size of structured target if applicable.
159	Effective lethal area for POL point (m^2). Note: Use specific target size of structured target if applicable.
160	Scaling factor used for targets in the woods. Note: When AMMO PK is given for a specific target density, enter AMMO density here.
161	Scaling factor used for personnel in foxholes. Note: When POL PK is given for a specific target density, enter POL density here.
162 - 168	Minimum altitude that 7 ADA weapons can engage the aircraft for this delivery profile. Over target area assume 0 for all 7 ADA weapons (meters).
169 - 175	Maximum altitude that the 7 ADA weapons can engage the aircraft for this delivery profile. Over target area, assume 100,000 for all 7 ADA weapons (meters).
176 - 182	Radius of effects for range of the 7 ADA weapons against the aircraft for this delivery profile (meters).
183 - 189	Probability of participation for ADA weapons against the aircraft for this delivery profile.
190 - 196	Number of rounds/basic lcad for each of the 7 ADA weapons (pounds).
197 - 203	Weight of one round for the 7 ADA weapons (pounds).
204 - 210	Number of rounds fired/engagement for the ADA weapons against the aircraft for this delivery.

File Position	Description
211 - 217	Probability of five-minute survivability for aircraft under this delivery profile against ADA weapon.
218 - 224	Probability of 30-second survivability for aircraft under this delivery profile against ADA weapon.

C. Munitions load profile files.

(1) The munitions load profile files contain data describing the ammunition carried by a particular aircraft type. Multiple files exist for each aircraft type, one for each type mission flown.

(2) These files are used to access the strike profile files.

File Name: Aircraft Type-Missions (example: A10ARM, for anti-armor mission)

File Position	Description
1	Number of munition types carried by the aircraft. This number reflects the following number of records to be read. The order of the records reflects the sequential order of munitions dropped over the target area.
2	File name of the first munition type of ADA profile for the aircraft.
.	.
.	.
.	.
N	File name of the Nth munition type and ADA profile for the aircraft.

D. Unit area files.

(1) The unit area files contain data describing the size in square meters of 10 unit types. Two files exist, one for Red units and one Blue units. Each file lists 10 unit types and contains four unit postures for each type.

(2) The array is Unit type area (I,J) where I represents the unit types and J represents unit posture.

I:
1 = Combat unit
2 = Artillery unit
3 = Air defense (ADA) unit
4 = Attack helicopter ground/forward arming and refueling point (FARP)
5 = Command post/headquarters (CP/HQ)
6 = Engineer unit
7 = POL/AMMO supply point
8 = Maintenance point
9 = Surface-to-air missile (SAM) site
10 = Communications/radar/electronic warfare (EW) site.

J:
1 = Attack
2 = Defend
3 = Reserve
4 = Move.

E. Weighted target fractions.

(1) The weighted target fractions files contain data describing the aircrafts' preference for the 70 elements plus POL dumps and ammunition dumps. Two files exist, one for Red elements and one for Blue elements. Each file lists an integer value between zero and 10 as a function of the five air combat missions. The higher the value, the greater the preference for firing at that target.

(2) The array is Ac_ms_tgt_wts (I,J) where:

I: Target elements
1-70 plus POL and AMMO

J: Aircraft mission
1 = Bridge
2 = Armor
3 = Personnel
4 = POL
5 = AMMO

5. ALGORITHMS.

A. Figure 5-3 presents a generalized logic flow of the processes occurring in the DIME air defense program. The purpose of this diagram is to provide a framework for consideration of the various algorithms used in the program. The DIME air defense program is a deterministic model using expected value techniques for calculating losses on both aircraft and ground

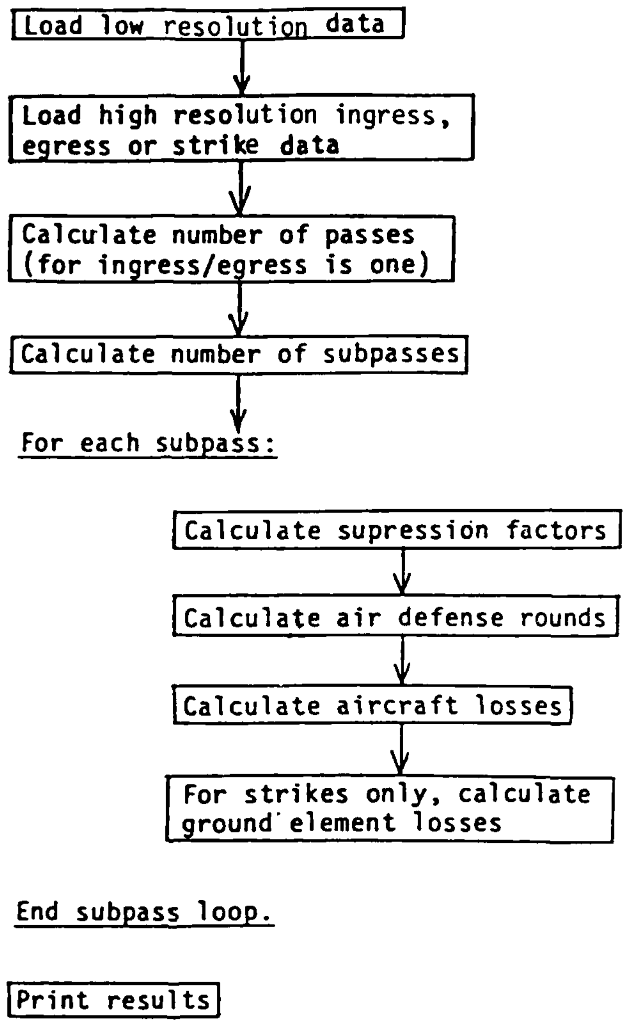


Figure 5-3. Air defense logic flow.

elements. The model first determines the appropriate parameters for aircraft and target elements. Next, specific characteristics are determined for the targeted units. Then aircraft attrition is figured by determining the number of air defense elements available in the battle area, calculating the number of subpasses in the flight, calculating the air defense artillery suppression factor, calculating the amount of air defense munitions expended, and, finally, calculating the number of aircraft killed. Attrition to ground elements is figured by calculating equipment losses due to area munitions, equipment losses due to point munitions, and mounted infantry losses as vehicles are destroyed. The "UNITFILE" is updated to reflect elements lost and amount of ammunition expended.

B. The following paragraphs provide a detailed description of the algorithms used for the calculation of subpasses, passes, air defense suppression, air defense munition expenditures, aircraft kills, area losses, point losses, and mounted infantry losses. When MIN() appears in the following algorithms, it represents a minimum function where the minimum of the items within the parentheses are used in the calculations.

C. The following pass and subpass calculations are performed for each group of aircraft sent on an ingress, strike or egress mission.

(1) Calculate subpasses. The number of tactical groupings the flight is divided into, based on the mission (close air support or basic air interdiction) and on tactical considerations. The number of subpasses that a flight is divided into is calculated as follows:

$$Nsp = Nac / (2 * Nmax) \qquad \text{(Eq. 5-1)}$$

where:

Nsp = the number of subpasses.
Nac = the number of aircraft in the flight.
Nmax = the maximum number of aircraft that would be flown on that specific type of mission due to tactical considerations.

Note: Nmax is multiplied by two in order to simulate air defense saturation achieved by tactical spacing between subpasses.

(2) Calculate passes. The number of times an aircraft must fly over the target area to deliver its ordnance is calculated as follows:

$$Np = Mp/Pd \qquad \text{(Eq. 5-2)}$$

where:

Np = the number of passes.
Mp = the minimum number of passes required to deliver the load of munitions.
Pd = the probability of detecting the principal target per pass.

D. All of the following equations are calculated for each subpass.

(1) Calculate air defense suppression. Air defense suppression is based on the unit's mission/activities at the time of the strike.

(a) Vehicular mounted air defense weapons are suppressed as follows:

$$Nwa_i = \prod^{i} [Nw_i * (1 - Sf_i)] \qquad \text{(Eq. 5-3)}$$

where:

Nwa_i = the number of air defense weapons available of type i after suppression.
Nw_i = the number of air defense weapons available of type i.
Sf_i = the suppression factor for type i air defense weapons.

(b) Hand-held air defense weapons are suppressed as follows:

$$Nwa_i = \prod^{i} [Nw_i * (1 - Sf_i)] * Df_i \qquad \text{(Eq. 5-4)}$$

where:

Df_i = the fraction of hand-held air defense weapons dismounted at the time of attack.

(2) Calculations of air defense ammunition expenditures are as follows:

(a) Target area:

$$Ta = Tl * Tw \qquad \text{(Eq. 5-5)}$$

where:

Ta = target area.
Tl = target length.
Tw = target width.

(b) Unit area:

$$Ua = MIN(U,Ta) \qquad \text{(Eq. 5-6)}$$

where:

Ua = the unit area.
U = the area required by the units given the current posture.

(c) Air defense area factor:

$$Af_i = PI * [(R_i) \uparrow 2]/Ua \qquad \text{(Eq. 5-7)}$$

where:

Af_i = the target area factor for air defense system type i.
PI = the symbol designating the ratio of the circumference of a circle to its diameter; approximately 3.1415927.
R_i = the range of air defense system type i.

(d) Number of weapons engaging aircraft:

$$Nwe_i = Af_i * Nwa_i * Pp_i \qquad \text{(Eq. 5-8)}$$

where:

Nwe_i = the number of air defense weapons of type i engaging the aircraft.
Pp_i = the percentage of air defense system type i that will participate.

(e) Basic load for air defense weapons:

$$Bl_i = Nr_i * Rw_i \qquad \text{(Eq. 5-9)}$$

$$Tl = \sum_i Bl_i \qquad \text{(Eq. 5-10)}$$

where:

Bl_i = the basic load weight for air defense weapon type i.
Nr_i = the number of rounds of ammunition in the basic load for air defense weapon system type i.
Rw_i = the weight of one round of ammunition for air defense weapon system type i.
Tl = the total basic load weight for all air defense weapon types in the battle area.

(f) Number of rounds available:

$$Ra_i = Bl_i * Ta * 2000/Tl \quad \text{(Eq. 5-11)}$$

where:

Ra_i = the number of rounds of air defense ammunition available of type i.
Bl_i = the basic load weight for air defense weapon type i.
Ta = the total tons of air defense ammunition available.
Tl = the total basic load weight for all air defense weapon types in the battle.

(g) Number of loads fired:

$$Rf_i = MIN(Ra_i\ ,\ Rf_i\ ) * Nwe_i \quad \text{(Eq. 5-12)}$$

$$Lf_i = Rf_i\ /Rfe_i \quad \text{(Eq. 5-13)}$$

where:

Rf_i = the number of air defense rounds fired per type i.
Rfe_i = the number of rounds fired per engagement by air defense weapon type i.
Lf_i = the number of air defense ammunition loads fired per type i.

(3) Number of aircraft kills. The number of aircraft kills is calculated as follows:

(a) Probability of kill:

$$Psk_i = (1 - Pk_i)/Npa \quad \text{(Eq. 5-14)}$$

$$Psa_i = (1 - Pa_i)/Npa \quad \text{(Eq. 5-15)}$$

$$Pskp = \prod^{i} [(1 - Psk_i) \uparrow Lf_i] \quad \text{(Eq. 5-16)}$$

$$Psap = \prod^{i} [(1 - Psa_i) \uparrow Lf_i] \quad \text{(Eq. 5-17)}$$

where:

Psk_i = the probability of one aircraft in the pass being killed in 30 seconds by air defense weapons type i.

Pk_i = the probability of one aircraft surviving for 30 seconds against air defense weapons type i.

Npa = the number of aircraft in a subpass

Psa_i = the probability of one aircraft in the subpass being killed in five minutes against air defense weapon type i.

Pa_i = the probability of one aircraft surviving for five minutes against air defense weapon type i.

Pskp = the probability of killing an aircraft out of the subpass, in 30 seconds based on the number of loads fired.

Psap = the probability of killing an aircraft out of the subpass, in five minutes based on the number of loads fired.

(b) Number of aircraft lost:

$$Npsk = (1 - Pskp) * Npa * Ee \quad \text{(Eq. 5-18)}$$

$$Nspa = (1 - Psap) * Npa * Ee \quad \text{(Eq. 5-19)}$$

$$Nsac = Npa - Npsk \quad \text{(Eq. 5-20)}$$

$$Nacl = \sum Npsa \quad \text{(Eq. 5-21)}$$

where:

Npsk = the number of aircraft lost within 30 seconds of engagement.

Ee = the effective engagement factor, based on the perceived density of enemy air defense.

Npsa = the number of aircraft lost within five minutes of engagement.

Nsac = the number of aircraft available for the strike.

Nacl = the number of aircraft lost on the flight.

(4) Number of ground elements lost to area munitions during a strike. The number of elements lost is calculated in the following manner.

(a) Kill equations are dependent on whether one plane can cover the target area or if more are necessary. In order to determine this, the following calculation are performed.

$$Ua = MIN(Uatp, L * W) \quad \text{(Eq. 5-22)}$$

$$Ta = MIN(L * W, 3 * Ua) \quad \text{(Eq. 5-22a)}$$

$$Pc = Ata/Ta \quad \text{(Eq. 5-22b)}$$

where:

Uatp = area (in meters squared) required by elements in the units, given the current posture of the unit.
L = target length (in meters)
W = target width (in meters)
Ua = unit area (in square meters)
Ta = size of target area (in square meters)
Ata = effective lethal area for the aircraft (in square meters)
Pc = the number of planes to cover the target area

(b) The probability of kill for an element targeted by aircraft must be adjusted according to the terrain as follows:

$$PK_j = Pac_j * [Tfo + (Tfw * Sfw)] \quad \text{(Eq. 5-23)}$$

where:

PK_j = the probability that target element type j is killed subject to terrain considerations.
Pac_j = the probability of kill of target element type j by the aircraft-delivered area munition.
Tfo = the fraction of the target in the open.
Tfw = the fraction of the target in the woods.
Sfw = the scaling factor used for targets in the woods.

(c) If the target area is small enough to be covered by one plane ($Pc \leq 1$), the number of ground elements lost is:

$$Nei_j = [1 - (1 - PK_j) \uparrow (Nsac * Ef)] * Ne_j] \quad \text{(Eq. 5-24)}$$

where:
Nei_j = the number of type j elements lost to the flight of aircraft.
Nsac = the number of aircraft performing the strike.
Ef = the effectiveness factor for aircraft munitions based on the perceived density of air defense weapons.
Ne_j = the number of elements of type j in the target area.

(b) If the target area is larger and $Pc > 1$, the coverage attained by a group of aircraft is calculated as:

$$C_j = Cac_j * Ef * Nsac/Ta \quad \text{(Eq. 5-25)}$$

where:

C_j = the coverage ratio of the target area to the munitions coverage area.

Cac_j = the munitions area coverage of one aircraft with respect to target elements type j.

Ta = the size of the target area

(c) If the target area is greater than the area which can be covered by all aircraft ($C_j < 1$), then the ground element losses are:

$$Nel_j = C_j * PK_j * Ne_j \qquad \text{(Eq. 5-26)}$$

where:

Nel_j = the number of type j elements lost to the flight of aircraft.

(d) If the aircraft can cover the entire area ($C_j \geq 1$), then ground element losses (Nel_j) are:

$$Nel_j = [1 - (1 - PK_j) \uparrow C_j] * Ne_j \qquad \text{(Eq. 5-27)}$$

(5) The amount of POL and/or ammunition lost to munitions during a strike is calculated in the following manner:

(a) Loss equations are dependent on whether one plane can cover the target area or if more are necessary. In order to determine this, the following calculations are performed.

$$Ua = MIN(Uatp, L * W) \qquad \text{(Eq. 5-28)}$$

$$Ta = MIN(L * W, 3 * Ua) \qquad \text{(Eq. 5-28a)}$$

$$Pc = Ata/Ta \qquad \text{(Eq. 5-28b)}$$

(b) If the target area is small enough to be covered by one plane ($Pc \leq 1$), the POL and ammunition lost is:

$$Nm\_elmt\_lost = [1 - (1 - PK) \uparrow (Nm\_stk\_ac * Eff\_shots)] * Num\_elmt\_left \qquad \text{(Eq. 5-29)}$$

where:

Nm_elmt_lost = the number of elements lost to a flight of aircraft.

PK = the probability that the element is killed

Nm_stk_ac = the number of strike aircraft

Eff_shots = the effectiveness factor for aircraft munitions, based on the perceived density of air defense weapons

Nm_elmt_left = the number of elements in the target area

(c) If it is necessary for more than one plane to cover the target area (Pc > 1), the ammunition and POL dumps must be translated into equivalent targets:

Equiv_area = Nm_elmt_tgt/Dens (Eq. 5-30)

Equiv_tgt = Equiv_area/PK_area (Eq. 5-31)

where:

Equiv_area = the perceived target area due to the target elements and target density

Nm_elmt_tgt = the amount of POL or ammunition in the target area.

Dens = the density of the target area for either POL or ammunition.

Equiv_tgt = the number of equivalent Pk targets in the target area.

PK_area = the effective lethal area for either an ammo point or a POL point; this is found in the strike files, position 158 or 159.

(d) If the targeted area is larger and Pc > 1, the coverage capability of the entire strike group is calculated:

C = (Eff_shots * Nm_stk_ac) / Equiv_tgt (Eq. 5-32)

where:

C = the coverage ratio of the target area to the munitions coverage area.

If the aircraft can cover the entire area or more ($C \geq 1$), POL and ammunition losses are calculated:

Nm_elmt_lost = [1 - (1 - PK) ↑ C] * Nm_elmt_left (Eq. 5-32a)

where:

Nm_elmt_lost = the amount of POL or ammunition lost to the strike aircraft.

If the target area is greater than that which can be covered by all aircraft ($C \leq 1$), then the POL and ammunition losses are calculated as:

$$Nm\_elmt\_lost = PK * C * Nm\_elmt\_left \qquad \text{(Eq. 5-32b)}$$

(6) The number of ground elements lost to point munitions during a strike is calculated in the following manner:

(a) The total ammunition used by the aircraft strike force is:

$$Rac = Np * Nrfe \qquad \text{(Eq. 5-33)}$$

$$Nb = Bl * Nsac/Rac \qquad \text{(Eq. 5-34)}$$

where:

Rac = the number of rounds an aircraft could deliver per strike.
Np = the number of passes required by an aircraft to deliver its ordnance.
Nrfe = the number of rounds fired per engagement by an aircraft.
Nb = the effective number of aircraft in the strike based on ammunition availability.
Bl = the number of rounds of ammunition in the basic load for an aircraft.
Nsac = the number of aircraft performing the strike.

(b) Calculate target element preference factor. The preference factor associated with target elements is calculated as follows:

$$Pf_j = Mtw_j * Ne_j * Pk_j \qquad \text{(Eq. 5-35)}$$

$$Spf_j = \sum_j Pf_j \qquad \text{(Eq. 5-36)}$$

where:

Pf_j = the preference factor for target element j.
Mtw_j = the mission related target weight for element j.
Ne_j = the number of elements of type j in the target area.
Pk_j = the aircraft's probability of killing target element type j.
Spf = the sum of all preference factors for all elements of the target, for one aircraft.

(c) Calculate target element preference ratio factor. The preference ratio factor is calculated as follows:

$$Pfr_j = Pf_j / Spf \qquad \text{(Eq. 5-37)}$$

$$Tpf_j = Pfr_j * Nb \qquad \text{(Eq. 5-38)}$$

where:

Pfr_j = the preference ratio factor for target element j.
Tpf_j = the total preference factor for all aircraft in the flight against target element type j.
Nb = the effective number of aircraft in the strike.

(d) Calculate number of target elements lost. The losses represent the target selection process as being dependent; all aircraft fire at different targets. The number of elements lost is calculated as follows:

$$Nel_j = Pk_j * Tpf_j * Ef \qquad \text{(Eq. 5-39)}$$

where:

Nel_j = the number of type j elements lost to the flight of aircraft.
Pk_j = the aircraft's probability of killing target element type j.
Ef = the effectiveness factor for aircraft munitions based on the perceived density of air defense weapons.

(7) The number of mounted infantry losses during a strike. The number of mounted infantry lost when vehicles are destroyed is calculated as follows:

(a) Mount ratio; the number of infantry mounted on vehicles is calculated as follows:

$$Mr = MIN\ (NI/Nv,\ 8) \qquad \text{(Eq. 5-40)}$$

where:

Mr = the mount ratio; the number of infantry mounted on a troop-carrying vehicle.
NI = the number of infantry troops in the battle area.
Nv = the number troop-carrying vehicles in the battle area.

(b) Vehicles lost. The number of infantry-carrying vehicles lost is calculated as follows:

$$Fe = Nv * Pe/Nvu \quad \text{(Eq. 5-41)}$$

$$VL = Fe * Nvl \quad \text{(Eq. 5-42)}$$

where:

Fe = the fraction of troop-carrying vehicles exposed.
Pe = the percent of the targeted unit in the battle area.
Nvu = the number of vehicles in the targeted unit.
VL = the number of troop-carrying vehicles lost in the battle.
Nvl = the number of vehicles lost in the battle.

(c) Mount losses. The number of infantry lost in vehicles is calculated as follows:

$$Per = VL * Mr \quad \text{(Eq. 5-43)}$$

where:
Per = the number of personnel lost in vehicles.
VL = the number of troop carrying vehicles lost in the battle.
Mr = the number of troops mounted in vehicles.

6. "UNITFILE" IMPACT.

Table 5-1 provides a summary of the principal variables in "UNITFILE" by the air defense program.

7. CODE.

A. Introduction. This section contains information on the DIME air defense The second paragraph describes the functional areas of the program and a system flowchart. The third paragraph contains figures and tables that briefly explain the subroutines called from each functional area and the primary variables influenced by each subroutine.

B. Functional areas. This section contains a brief overview of the functional areas in the DIME air defense program. As shown in Figure 5-4 (functional flow diagram), the air defense program actually contains two distinct sections, one modeling ingress/egress operations and one modeling strike operations. The two operations share many of the same subroutines in the program.

Table 5-1. "UNITFILE" variables affected by air defense.

Element number	Name	Description
1-70	Weapons list	The 70 weapon type quantities present within the unit.
75	Major mission	1 = Attack 2 = Defend 3 = Reserve/Idle 4 = Move
78	Unit Type (X.Y)	X = Player identification (X.Y) 1 = Blue 2 = Red Y = Unit type 0 = Combat unit 1 = Artillery unit 2 = Air defense (ADA) unit 3 = Attack helicopter ground/ forward arming and refueling point 4 = Command post/headquarters (CP/HQ) 5 = Engineer unit 6 = Fuel/ammunition (POL/AMMO) supply unit 7 = Maintenance point 8 = Surface-to-air missile (SAM) site 9 = Communication/radar/electronic warfare (EW) site
80	Percent ADA suppressed (XX.YY)	XX = Vehicle ADA systems suppressed YY = Handheld systems suppressed
91	Detection status (X.Y)	X = Hours left until redetected Y = Unit status with respect to detection by the opposite commander 0 = Not detected 1 = Detected but not verified 2 = Acquired/verified 3 = Lost
92	Intelligence Status	Total hours this target has been status tracked this detection period
133	AD ammo	The amount of AD ammunition available to be consumed

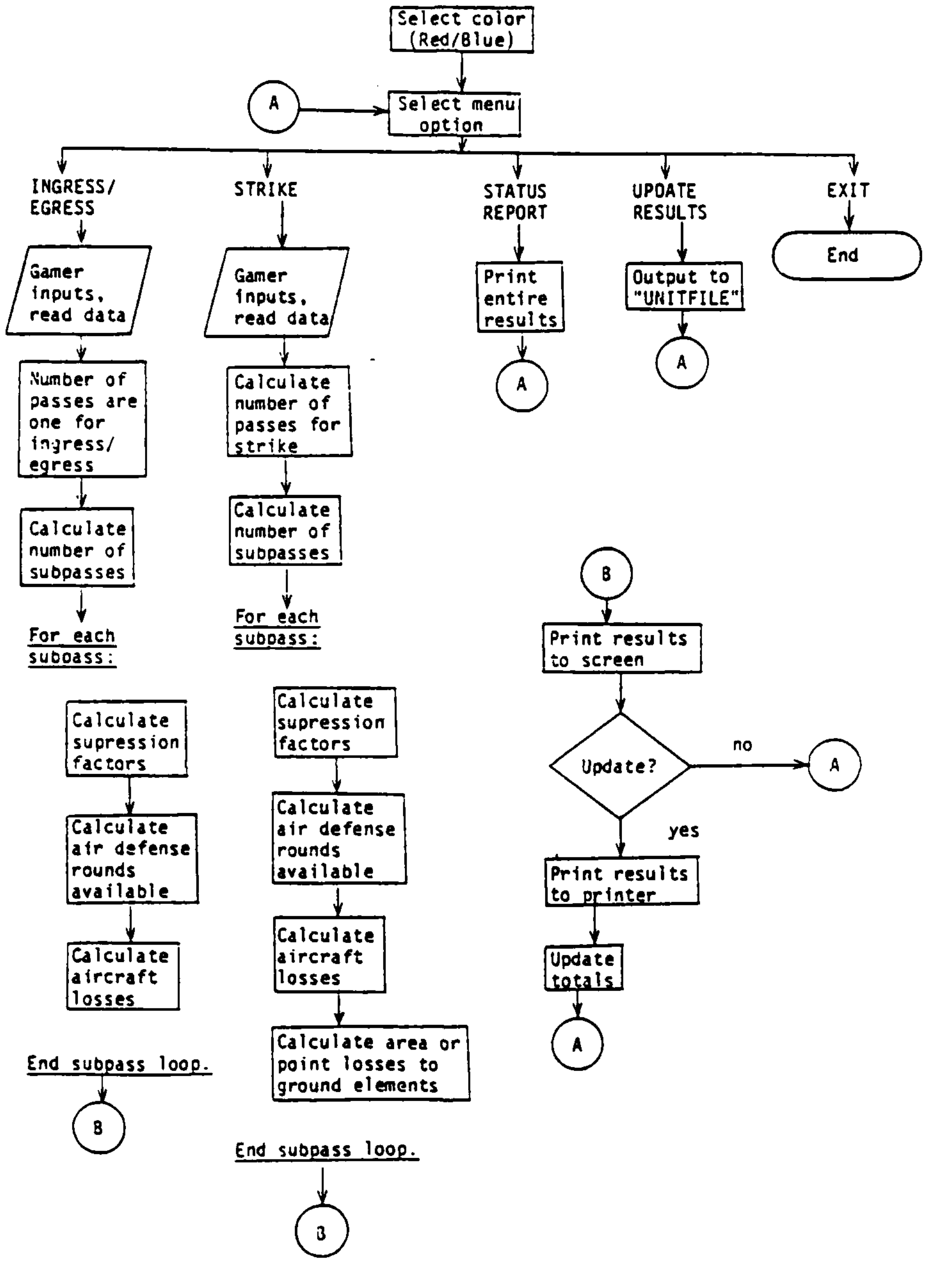


Figure 5-4. Air defense functional flow.

(1) As shown in Figure 5-4, both operations are triggered by the input of low-resolution data. Low-resolution data include the force (Red or Blue), the operation (ingress/egress or strike), aircraft information (number, type, and altitude of aircraft), and target information (width, length, and identification of units). An air defense input sheet is used to facilitate loading of this input data.

(2) After a gamer selects the force, aircraft, and target information, the DIME air defense program selects the appropriate high- resolution data corresponding to the operation (ingress/egress or strike) selected.

(3) For ingress/egress operations, the following procedures are used:

(a) High-resolution data include air defense weapon characteristics such as maximum and minimum altitude, range, basic load, rates of fire, and air defense probabilities of kill for each defense weapon played. Target information is also included, such as the number and type of elements in the target area and the amount of ammunition available to the ground elements. This information is contained in data files which are accessed by the type of aircraft and mission. Currently, four types of aircraft for each force are played, two fixed wing and two rotary wing. Ground units are made up of elements selected from the data base developed for DIME. DIME allows for 70 different elements per side plus two special elements.

(b) The next major functional area calculates aircraft losses. This major functional area is composed of the following four subareas:

1. First, calculating the number of subpasses is required in order to partition the flight of aircraft into increments of a size corresponding to the actual number of aircraft that would pass over a unit at any one time. Aircraft losses are calculated for each subpass increment, then are added together to determine total aircraft losses.

2. Next, air defense suppression calculates the readiness/reaction time of the air defense systems. These computations are based on the unit's mission at the time of the overflight.

3. The third subarea calculates the amount of air defense ammunition available and the amount fired during the engagement.

4. The last subarea calculates the number of aircraft lost and updates the number of aircraft available for follow-on missions.

(4) For strike operations, the following procedures are used:

(a) High-resolution strike data are similar to ingress/egress data with the addition of aircraft capabilities against ground elements. This information includes munition capabilities against each target element, flight profiles for each type munition carried, and tactical considerations based on mission requirements.

(b) The next major functional area calculates results of the strike operation. Aircraft kills and kills of ground elements are calculated on an interactive basis. This major functional area is composed of the following areas:

1. First, the number of passes required by each aircraft in order to deliver the load of munitions carried must be calculated.

2. Then, four subareas are used to calculate the number of aircraft lost. These subareas are the same four used in ingress/egress operations to calculate aircraft losses.

3. Next, losses to ground elements as a result of area-type munitions are calculated.

4. Losses to ground elements as a result of point-type munitions are then calculated.

5. The last of these functional areas calculates the number of mounted infantry lost when their vehicles are destroyed.

(5) The next major functional area of the DIME air defense program deals with the results of both ingress/egress and strike operations. Actions in this area include updating files with the number of remaining operational elements, listing results of surface-to-air and air-to-surface action, and to ready files for the next iteration of the battle. This functional area is composed of three subareas.

(a) The first of these subareas is used to output the results of the operation. Results include the number and type of aircraft killed and the number of each ground element destroyed.

(b) The next subarea is used to either update or purge the files with the results of the last battle increment.

(c) The last of these subareas is used to ready the program for the next increment.

(6) Finally, the last procedure allows for entry of new low-resolution data for a new battle or for exit from the DIME air defense program.

C. Subroutine summary. Table 5-2 provides a summary of the DIME air defense program subroutines and their primary variables. Subroutines called by each functional area are shown and the function of each subroutine is described. The primary variables for each subroutine are listed and described.

D. Code listing. A listing of the P3 program code appears in Table 5-3.

Table 5-2. DIME Air Defense Subroutines

A. Functional Area: Load low resolution data.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main program	Calls subroutines	A. Fct_hnd_ada_dmt	Fraction of hand-to-hand and weapons dismounted
Choose_color	Initializes Blue and Red forces	A. Frc_clr$	Force color (Red or Blue)
Flight_input	Inputs initial ingress/egress information	A. Nm_ac_input	Number of aircraft input
		B. Ac_mission	Aircraft mission
		C. Tgt_width	Target width
		D. Tgt_length	Target length
		E. Nm_units	Number of units
		F. Terr	Type Terrain
Strike_input	Inputs initial strike information	A. Ac_type	Aircraft type
		B. Ac_brkp_pct	Aircraft breakpoint percentage
		C. Tgt_pct_open	Percent of target in open
		D. Tgt_epsr_pct	Percent of target in area
Change_answer	Allows changes to information entered	A. Nm_units_chosen	Number of units chosen

Table 5-2. DIME Air Defense Subroutines (continued)

B. Functional Area: Load high resolution ingress/egress data.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Select_prmtrs	Equates low resolution entries to specific high resolution data	A. Aircraft$	Type aircraft
		B. Flt$	Aircraft mission
		C. Activity	Target mission
		D. Tgt_type	Target type
Flight_data	Sets air defense parameters for the type of aircraft.	A. Unit_type_area	Size of area required by type of unit.
		B. Min_ada_alt	Minimum altitude for air defense weapon
		C. Max_ada_alt	Maximum altitude for air defense weapon
		D. Ada_range	Range of air defense weapon
		E. Ada_prtcptn	Probability of participation for air defense weapon
		F. Ada_rnd_bs_ld	Number of rounds of basic load for air defense weapon
		G. Flt_profile	Data file contents
		H. Ada_rnd_wt	Weight of one round of ammunition
		I. Ada_rnd_frd_eng	Number of rounds fired per engagement

Table 5-2. DIME Air Defense Subroutines (continued)

B. Functional Area: Load high resolution ingress/egress data. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
FLight_data		J. Ada_psk_ac	Probability of surviving for 30 seconds
		K. Altitude	Altitude
Select_units	Stores target unit information in temporary files	A. Unit_id_ptr	Unit Identification Pointer
Flight_attrition	Calls subroutines to calculate aircraft losses	A. Break_point	Break Point
		B. Tgt_posture	Target Posture
		C. T_sub_pass	Total number of sub passes
		D. Nm_ac_left_1	Number of aircraft left after first pass
		E. Tl_ada_ton_frd	Total tons of air defense ammunition fired
		F. Tac_lost_pass	Total aircraft lost per pass
		G. Nm_psa_ac_lost	Number of aircraft lost in five minutes
		H. Nm_ac_left	Number of aircraft left

Table 5-2. DIME Air Defense Subroutines (continued)

B. Functional Area: Load high resolution ingress/egress data. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Select_ada_info	Sets air defense counts to totals contained in target units	A. Nm_ada_wpn	Number of air defense weapons
		B. Ada_suprsn_pet	Percentage of air defense
		C. Unit_posture	Unit posture
		D. Unit_type	Unit type
Calc_sub_pass	Calculates the number of aircraft exposed to ground units at any one time	A. Eff_shots	Scaling factor for aircraft
		B. Eff_engage	Sealing factor for air defense
		C. Nm_sub_pass	Number of sub passes
		D. Pass_acft	Number of aircraft in the pass
		E. Cas_bai	Close air support or Battlefield air interdiction
Calc_ada_suprsn	Calculates the suppression factor for air defense weapons	A. Suprsn_fre	Suppressed fraction
		B. Nm_ada_wpn_avl	Number of air defense weapons available

Table 5-2. DIME Air Defense Subroutines (continued)

B. Functional Area: Load high resolution ingress/egress data. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Calc_ada_rnds	Calculates the number of air defense rounds fired	A. Tgt_a	Target Area
		B. Unit_area	Unit Area
		C. Ada_area_fctr	Air defense area factor
		D. Nm_ada_wpn_eng	Number of air defense weapons engaging
		E. Tl_ada_bs_ld_wt	Total air defense in basic load weight
		F. Ada_bs_ld_wt	Air defense basic load weight
		G. Ada_rnd_avl	Number of air defense rounds available
		H. Ada_rnd_frd	Number of air defense rounds fired
		I. Ada_lds_fd	Number of air defense loads fired
Calc_ac_losses	Calculates the number of aircraft lost.	A. Ada_tons_frd	Tons of air defense ammunition fired
		B. Nm_psk_ac_lost	Number of aircraft lost in 30 seconds
		C. Nm_stk_ac	Number of strike aircraft
		D. Nm_ac_lost_flt	Number of aircraft lost this flight
		E. Ada_tons_avl	Tons of air defense ammunition available

Table 5-2. DIME Air Defense Subroutines (continued)

C. Functional area: Load high resolution strike data

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Strike_data; Strike_init	Calls subroutines to set strike data. Initialize strike variables.	A. Ac_brkpt_fre	Aircraft break point fraction
		B. Tgt_fre_open	Fraction of target in the open
		C. Tgt_fre_woods	Fraction of target in the woods
		D. Altitude_meters	Altitude in meters
		E. Tgt_expsr_fre	Portion of target exposed
		F. Tl_ada_tons	Total tons of air defense ammunition
		G. Nm_elmt_lost_fl	Number of elements lost to the flight
		H. Nm_elmt_tgt	Number of elements in the target
Select_stk_info	Sets parameters for strike operation	A. Nm_vehicles	Number of vehicles
		B. Suprsn_frc_1	Suppression factor number 1
		C. Suprsn_frc_2	Suppression factor number 2
		D. Nm_elmt left	Number of elements left
		E. Mount_ratio	Ratio of mounted infantry
		F. Nm_infantry	Number of infantry

Table 5-2. DIME Air Defense Subroutines (continued)

C. Functional area: Load high resolution strike data (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Strike_apportion	Sets parameters for mounted infantry losses	A. Tl_ac_passes	Total aircraft passes
		B. Frct	Fraction of troop carrying vehicles lost in the unit
		C. V_lost_unit	Vehicles lost in the unit
		D. Mount_losses	Mount infantry losses
		E. Vehicles_lost	Vehicles lost
Strike_atrition	Calculates mounted infantry losses	A. T_plos	Total probability of Loss
Strike_profile	Sets air defense parameters for the type aircraft flown	A. Wpn_load_code	Aircraft munition type
		B. Ac_area_prn_tgt	Effective lethal area for munition
		C. Min_ac_passes	Minimum number of passes required to deliver munitions
		D. Nm_ac_rnds_eng	Rounds delivered for engagement
		E. Nm_ac_rnd_bs_ld	Number of rounds in basic load
		F. Prn_tgt_wds_frc	Scaling factor for targets in the woods
		G. Prsnl_fxhl_frc	Scaling factor for personnel in foxholes

Table 5-2. DIME Air Defense Subroutines (continued)

C. Functional area: Load high resolution strike data (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Strike_profile (concluded)		H. Ac_plos_prn_tgt	Probability of aircraft loss
		I. Ac_pdtc_prn_tgt	Probability of detecting target
		J. Ac_tgt_pk	Probability of destroying target
		K. Ac_tgt_area	Lethal area for type munitions carried by aircraft
Strike_info	Calculates aircraft passes when more than one munition type is carried	A. Nm_ac_passes	Number of aircraft passes
Calc_area_ losses	Calculates elements lost to area munitions	A. Tgt_area	Target area
		B. Nm_areas	Number of arms
		C. Nm_elmt_lost	Number of elements lost
Calc_point_ losses	Calculates elements lost to area	A. Tgt_area	Target area
		B. Nm_areas	Number areas
		C. Nm_elmt_lost	Number of elements lost

Table 5-2. DIME Air Defense Subroutines (continued)

C. Functional area: Load high resolution strike data (concluded)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Calc_point_ loss	Calculates elements lost to point munitions	A. Nm_elmt	Number of elements

D. Functional Area: Results

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Flight_results	Lists results to include number of aircraft and ground elements killed	A. R_target_names$	Red target names
		B. B_target_names$	Blue target names
Store_results	Routes units in temporary files	A. Nm_units_played	Number of units played
Add_results	Calculates number of aircraft for next phase of air operation	A. Nm_ac_flt_ing	Number of aircraft ingressing
		B. Nm_ac_flt_egr	Number of aircraft egressing
		C. Nm_ac_flt_stk	Number of aircraft striking
		D. Nm_ac_lost_ing	Number of aircraft lost ingressing
		E. Nm_ac_lost_egr	Number of aircraft lost egressing

Table 5-2. DIME Air Defense Subroutines (continued)

D. Functional area: Results

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Add_results (concluded)		F. Nm_ac_lost_stk	Number of aircraft lost striking
		G. Nm_elmt_lost_ac	Number of elements lost to aircraft
		H. Nm_ac_lost_ada	Number of aircraft lost to ADA
		I. Nm_ac_avl_egr	Number of aircraft available to ingress
		J. Nm_ac_avl_stk	Number of aircraft available to strike

E. Functional Area: Update Results

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Update_results	Calls subroutines to update and purge results.	A. Tl_ac_flt_ing	Total aircraft ingressing
		B. Tl_ac_flt_egr	Total aircraft egressing
		C. Tl_ac_flt_stk	Total aircraft striking
Cumulate_totals	Totals aircraft information for future use.	A. Tl_ac_lost_ing	Total aircraft lost ingressing
		B. Tl_ac_lost_egr	Total aircraft lost egressing
		C. Tl_ac_lost_stk	Total aircraft lost striking
		D. Tl_ac_lost_ada	Total aircraft lost
		E. Tl_eimt_lost_ac	Total target elements lost

Table 5-3. Air attack/Air defense code.

```
10 ! THIS PROGRAM IS THE AIR ATTACK / AIR DEFENSE MODULE FOR DIME (P3).
20 ! THIS PROGRAM WAS CODED BY STEVE ARRINGTON AND CINDY JAHNKE
30 ! 22 OCT  1983.
40 ! DATA LAST CHANGED BY ROB BELFLOWER, 21 MARCH 1985, BDM
50 ! EXPANDED VERSION -- JUNE 9, 1986 -- BY OAO CORP.
51 !      DECLASSIFIED -- AUG 7, 1986 -- BY OAO CORP.   ** DC **
60      OPTION BASE 1
70      DIM Ac_burst_wt(72),Ac_burst_wt_frc(72)
80      DIM Ac_pdtc_prn_tgt(4),Ac_plos_prn_tgt(4)
90      DIM Ac_tgt_area(72),Ac_tgt_pk(72)
100     DIM Ac_ms_tgt_wts(72,5),Ada_area_fctr(7)
110     DIM Ada_bs_ld_wt(7),Ada_lds_frd(7)
120     DIM Ada_mobility_tp(7),Ada_prtcptn(7)
130     DIM Ada_psa_ac(7),Ada_psk_ac(7)
140     DIM Ada_range(7),Ada_rnd_bs_ld(7)
150     DIM Ada_rnd_frd(7),Ada_rnd_frd_eng(7)
160     DIM Ada_rnd_wt(7),B_target_name$[350]
170     DIM Fct_hnd_ada_dmt(4,2),Flt_profile(56)
180     DIM Max_ada_alt(7),Min_ada_alt(7)
190     DIM Nm_ac_avl_egr(4),Nm_ac_avl_stk(4)
200     DIM Nm_ac_flt_ing(4),Nm_ac_flt_egr(4)
210     DIM Nm_ac_flt_stk(4),Nm_ac_lost_ada(4)
220     DIM Nm_ac_lost_ing(4),Nm_ac_lost_egr(4)
230     DIM Nm_ac_lost_stk(4),Nm_ada_wpn(7)
240     DIM Nm_ada_wpn_avl(7),Nm_ada_wpn_eng(7)
250     DIM Nm_elmt_left(72),Nm_elmt_lost_ac(72)
260     DIM Nm_elmt_lost_fl(72),Nm_elmt_tgt(72)
270     DIM O(72),R_target_name$[350]
280     DIM Stk_profile(224),Suprsn_frc(2)
290     DIM T$[350],T_plos(4),Terr(5)
300     DIM Tgt_expsr_pct(5),Tgt_expsr_frc(5)
310     DIM Tl_ac_flt_ing(4),Tl_ac_flt_egr(4)
320     DIM Tl_ac_flt_stk(4),Tl_ac_lost_ada(4)
330     DIM Tl_ac_lost_egr(4),Tl_ac_lost_ing(4)
340     DIM Tl_ac_lost_stk(4),Tl_elmt_ac(72)
350     DIM Unit(150),Unit_id(5),Unit_id_20_150(20)
360     DIM Unit_info(150),Unit_store_5_15(5,150)
370     DIM Unitstore_20_15(20,150),Unit_type_area(10,4)
380     DIM Wpn_ld_template$(3)[14],Tm$[7],Frac_inf(5)
390!
400!
410     Nm_units_played=0
420 !
430     Disk$=":HP9134,701"
431     Dcdisk$=":HP9134,701,0"   !   ** DC **
440!
450 Main_program:!
460!
470     !LOAD TERRAIN PLOS DATA
471     !
472     !  ** DC **
473     !
```

Table 5-3. Air attack/Air defense code.

```
480   ASSIGN @Ptploss TO "TOT_PLOSS"&Dcdisk$
490   ENTER @Ptploss,1;T_plos(*)
500   ASSIGN @Ptploss TO *
510  !LOAD FRACTION DISMOUNTED
520   ASSIGN @Pfrcdsm TO "FRAC_DISM"&Dcdisk$
530   ENTER @Pfrcdsm,1;Fct_hnd_ada_dmt(*)
540   ASSIGN @Pfrcdsm TO *
580  ! ** END DC **
590 New_color: !
600   PRINTER IS 1
610   GOSUB Choose_color
620   Menu$="REPEAT"
630   WHILE Menu$="REPEAT"
640     GOSUB Menu_selection
650     SELECT Option$
660     CASE "INGRESS","EGRESS"
670       GOSUB Flight_input
680       GOSUB Flight_output
690       GOSUB Change_answer
700       IF Answer$="N" THEN
710         GOSUB Flight_data
720         GOSUB Flight_atrition
730         GOSUB Flight_results
740       END IF
750     CASE "STRIKE"
760       GOSUB Strike_input
770       GOSUB Strike_output
780       GOSUB Change_answer
790       IF Answer$="N" THEN
800         GOSUB Strike_data
810         GOSUB Strike_atrition
820         GOSUB Flight_results
830       END IF
840     CASE "STATUS"
850       GOSUB Status_report
860     CASE "UPDATE"
870       GOSUB Update_results
880     CASE "EXIT"
890       GOSUB Exit_module
900     CASE ELSE
910       BEEP
920     END SELECT
930   END WHILE
940   !
950   STOP
960 !
970 !*************************************************************************
*
980 !
990 Add_results: !
1000!
1010  SELECT Flight$
```

Table 5-3. Air attack/Air defense code.

```
1020  CASE "INGRESS"
1030    Nm_ac_flt_ing(Ac_type)=Nm_ac_flt_ing(Ac_type)+Nm_ac_input
1040    Nm_ac_lost_ing(Ac_type)=Nm_ac_lost_ing(Ac_type)+Nm_ac_lost_flt
1050  CASE "EGRESS"
1060    Nm_ac_flt_egr(Ac_type)=Nm_ac_flt_egr(Ac_type)+Nm_ac_input
1070    Nm_ac_lost_egr(Ac_type)=Nm_ac_lost_egr(Ac_type)+Nm_ac_lost_flt
1080  CASE "STRIKE"
1090    Nm_ac_flt_stk(Ac_type)=Nm_ac_flt_stk(Ac_type)+Nm_ac_input
1100    Nm_ac_lost_stk(Ac_type)=Nm_ac_lost_stk(Ac_type)+Nm_ac_lost_flt
1110    FOR I=1 TO 72
1120      Nm_elmt_lost_ac(I)=Nm_elmt_lost_ac(I)+Nm_elmt_lost_fl(I)
1130    NEXT I
1140  END SELECT
1150  !
1160  Nm_ac_lost_ada(Ac_type)=Nm_ac_lost_ada(Ac_type)+Nm_ac_lost_flt
1170  Nm_ac_avl_stk(Ac_type)=Nm_ac_flt_ing(Ac_type)-Nm_ac_lost_ing(Ac_type)-Nm_a
c_flt_stk(Ac_type)
1180  Nm_ac_avl_egr(Ac_type)=Nm_ac_flt_ing(Ac_type)-Nm_ac_lost_ing(Ac_type)-Nm_a
c_lost_stk(Ac_type)-Nm_ac_flt_egr(Ac_type)
1190  !
1200  RETURN
1210!
1220!***********************************************************************
*
1230!
1240 Calc_ac_losses:!
1250  Tl_ada_tons_frd=0
1260  Psk_product=1
1270  Psa_product=1
1280  FOR I=1 TO 7
1290    IF Altitude_meters<=Max_ada_alt(I) AND Altitude_meters>=Min_ada_alt(I) A
ND Ada_lds_frd(I)>0 THEN
1300      Psk_term=(1-Ada_psk_ac(I))/FNMax(Pass_acft,1)
1310      Psa_term=(1-Ada_psa_ac(I))/FNMax(Pass_acft,1)
1320        !
1330      Psk_product=Psk_product*(1-Psk_term)^Ada_lds_frd(I)
1340      Psa_product=Psa_product*(1-Psa_term)^Ada_lds_frd(I)
1350        !
1360      Ada_tons_frd=Ada_rnd_frd(I)*Ada_rnd_wt(I)/2000
1370      Tl_ada_tons_frd=Tl_ada_tons_frd+Ada_tons_frd
1380    END IF
1390      !
1400  NEXT I
1410  Nm_psk_ac_lost=(1-Psk_product)*Pass_acft*Eff_engage
1420  Nm_psa_ac_lost=(1-Psa_product)*Pass_acft*Eff_engage
1430   !
1440  Nm_stk_ac=Pass_acft-Nm_psk_ac_lost
1450  Nm_ac_lost_flt=Nm_ac_lost_flt+Nm_psa_ac_lost
1460   !
1470   !
1480  Ada_tons_avl=Ada_tons_avl-Tl_ada_tons_frd
1490  Unit(133)=Ada_tons_avl
```

Table 5-3. Air attack/Air defense code.

```
1500  Unit(141)=Unit(141)+Tl_ada_tons_frd
1510   !
1520  IF Option$="INGRESS" OR Option$="EGRESS" THEN
1530    PRINT USING Fmt_call;Nm_psa_ac_lost,Acprint$," LOST TO AD ELEMENTS OF U
IT ",Unit_id(This_unit)," DURING ",Option$
1540 Fmt_call:IMAGE 3D.D,1X,4A,29A,3D,8A,7A
1550  END IF
1560   !
1570  RETURN
1580!
1590!*********************************************************************
*
1600!
1610 Calc_ada_rnds:!
1620!
1630  Tgt_a=Tgt_length*Tgt_width
1640  Unit_area=FNMin(Unit_type_area(Unit_type,Unit_posture),Tgt_a)
1650  FOR I=1 TO 7
1660    Ada_area_fctr(I)=PI*(Ada_range(I)^2)/FNMax(Unit_area,.0000001)
1670    Ada_area_fctr(I)=FNMin(Ada_area_fctr(I),1)
1680  NEXT I
1690  FOR I=1 TO 7
1700    Nm_ada_wpn_eng(I)=Ada_area_fctr(I)*Nm_ada_wpn_avl(I)*Ada_prtcptn(I)
1710  NEXT I
1720  Nm_ada_wpn_eng(6)=Nm_ada_wpn_eng(6)*Plos
1730  Nm_ada_wpn_eng(7)=Nm_ada_wpn_eng(7)*Plos
1740  Tl_ada_bs_ld_wt=0
1750  FOR I=1 TO 7
1760    Ada_bs_ld_wt(I)=Ada_rnd_bs_ld(I)*Ada_rnd_wt(I)*Nm_ada_wpn(I)
1770    Tl_ada_bs_ld_wt=Tl_ada_bs_ld_wt+Ada_bs_ld_wt(I)
1780  NEXT I
1790  FOR I=1 TO 7
1800    Ada_rnd_avl=Ada_rnd_bs_ld(I)*Ada_tons_avl*2000/FNMax(Tl_ada_bs_ld_wt,.0
00001)
1810    Ada_rnd_frd(I)=FNMin(Ada_rnd_avl,Ada_rnd_frd_eng(I))*Nm_ada_wpn_eng(I)
1820    Ada_lds_frd(I)=Ada_rnd_frd(I)/FNMax(Ada_rnd_frd_eng(I),.0000001)
1830  NEXT I
1840  RETURN
1850!
1860!*********************************************************************
*
1870!
1880 Calc_ada_suprsn:!
1890!
1900   !SET TYPE OF MOBILITY FOR ADA WEAPONS
1910  Ada_mobility_tp(1)=1        !VEHICLE
1920  Ada_mobility_tp(2)=1        !VEHICLE
1930  Ada_mobility_tp(3)=1        !VEHICLE
1940  Ada_mobility_tp(4)=1        !VEHICLE
1950  Ada_mobility_tp(5)=1        !VEHICLE
1960  Ada_mobility_tp(6)=2        !HAND-HELD
1970  Ada_mobility_tp(7)=2        !HAND-HELD
```

Table 5-3. Air attack/Air defense code.

```
1980   !
1990   !UNPACK ADA SUPRESSION PERCENTAGE FOR HAND-HELD/VEHICLE ADA WEAPONS
2000  Suprsn_frc(1)=INT(Ada_suprsn_pct)
2010  Suprsn_frc(2)=Ada_suprsn_pct-Suprsn_frc(1)        !HAND-HELD
2020  Suprsn_frc(1)=Suprsn_frc(1)/100                   !VEHICLE
2030   !
2040  FOR I=1 TO 7
2050    Nm_ada_wpn_avl(I)=Nm_ada_wpn(I)*(1-Suprsn_frc(Ada_mobility_tp(I)))
2060  NEXT I
2070   !CONSIDER HAND HELD WEAPONS OUT OF CARRIER
2080  Nm_ada_wpn_avl(6)=Nm_ada_wpn_avl(6)*Fct_hnd_ada_dmt(Tgt_posture,Ada_side)
2090  Nm_ada_wpn_avl(7)=Nm_ada_wpn_avl(7)*Fct_hnd_ada_dmt(Tgt_posture,Ada_side)
2100   !
2110  RETURN
2120!
2130!**************************************************************************
2140!
2150 Calc_area_loss: !
2160!
2170  Tgt_area=Tgt_length*Tgt_width
2180  IF Tgt_area>3*Unit_area THEN
2190    Tgt_area=3*Unit_area
2200  END IF
2210   !
2220  Tgt_area=FNMax(Tgt_area,1)
2230   !
2240  !IF Tgt_type=6 THEN GOSUB Calc_answer
2250  FOR I=1 TO 72
2260    Pk=Ac_tgt_pk(I)*(Tgt_frc_open+Tgt_frc_woods*Prn_tgt_wds_frc)
2270    Nm_areas=FNMin(Ac_tgt_area(I)/Tgt_area,1)
2280    IF Nm_areas=1 THEN
2290          ! ONE PLANE CAN COVER TARGET
2300      Nm_elmt_lost=(1-(1-Pk)^(Nm_stk_ac*Eff_shots))*Nm_elmt_left(I)
2310    ELSE
2320          ! CALCULATE COVERAGE BY STRIKE GROUP.
2330          ! THIS ASSUMES DEPENDENCE FROM PASS TO PASS
2340      C=((Ac_tgt_area(I)*Eff_shots*Nm_stk_ac))/Tgt_area
2350      IF C>=1 THEN
2360            !COVERAGE BY ALL AIRCRAFT IN PASS MORE THAN ONCE OVER THE
2370            !TARGET AREA.
2380        Nm_elmt_lost=(1-(1-Pk)^C)*Nm_elmt_left(I)
2390      ELSE
2400            !COVERAGE BY ALL AIRCRAFT IN PASS LESS THAN THE TARGET AREA.
2410        Nm_elmt_lost=C*Pk*Nm_elmt_left(I)
2420      END IF
2430    END IF
2440    Nm_elmt_lost_fl(I)=Nm_elmt_lost_fl(I)+Nm_elmt_lost
2450    Nm_elmt_left(I)=Nm_elmt_left(I)-Nm_elmt_lost
2460  NEXT I
2470   !
2480  FOR I=1 TO 7
2490    Nm_ada_wpn(I)=Nm_elmt_left(I+47)
```

Table 5-3. Air attack/Air defense code.

```
2500  NEXT I
2510   !
2520  RETURN
2530!
2540!**************************************************************************
2550!
2560 Calc_answer:  !
2570!
2580  IF Ac_mission=4 THEN
2590    I=72
2600  ELSE
2610    I=71
2620  END IF
2630  Pk_area=Stk_profile(I+38)
2640  Dens=Stk_profile(I+40)
2650  Pk=Ac_tgt_pk(I)
2660  Equiv_area=Nm_elmt_tgt(I)/Dens
2670  Equiv_tgt=Equiv_area/Pk_area
2680  Equiv_tgt=FNMin(Equiv_tgt.1)
2690  IF Equiv_tgt<1 THEN GOTO 2730
2700     ! CASE 1     ONE AC COVERS TARGET
2710  Nm_elmt_lost=(1-(1-Pk)^(Nm_stk_ac*Eff_shots))*Nm_elmt_left(I)
2720  GOTO 2800
2730  C=(Eff_shots*Nm_stk_ac)/Equiv_tgt
2740  IF C<1 THEN GOTO 2780
2750     ! CASE 2     ALL AC TOGETHER COVER TARGET
2760  Nm_elmt_lost=(1-(1-Pk)^C)*Nm_elmt_left(I)
2770  GOTO 2800
2780     ! CASE 3     ALL AC COVER LESS THAN TARGET
2790  Nm_elmt_lost=Pk*C*Nm_elmt_left(I)
2800  Nm_elmt_lost_fl(I)=Nm_elmt_lost_fl(I)+Nm_elmt_lost
2810  Nm_elmt_left(I)=Nm_elmt_left(I)-Nm_elmt_lost
2820!
2830  RETURN
2840!
2850!**************************************************************************
2860!
2870 Calc_point_loss: !
2880!
2890  Brgg=0
2900  IF Tgt_type=8 THEN GOTO 2930
2910  GOTO 3120
2920  IF Destroy=2 THEN 3120
2930  Pk=Stk_profile(86)
2940  R3=Unit_info(94)                  !***  GET SEED  ************
2950  IF R4=0 THEN R3=R5
2960  R4=1
2970  RANDOMIZE R3                      !***  RANDOMIZE NUMBER  ****
2980  FOR I=1 TO 10
2990    R3=RND
3000  NEXT I
3010  Psurvb=(1-Pk)^Nm_stk_ac
```

Table 5-3. Air attack/Air defense code.

```
3020  Seed=R3*100
3030  Unit_store_5_15(1,94)=Seed
3040  Unit_info(94)=Seed
3050  PRINTER IS 1
3060  IF Psurvb<R3 THEN GOTO 3090
3070  Brgg=1
3080  GOTO 3230
3090  Brgg=2
3100  Destroy=1
3110  GOTO 3230
3120  Divisor=FNMax(Nm_ac_passes*Nm_ac_rnds_eng,1)
3130  Nm_ac_bursts=Nm_ac_rnd_bs_ld*Nm_stk_ac/Divisor
3140  Constant=Tgt_frc_open+Tgt_frc_woods*Ac_plos_prn_tgt(Tgt_posture)
3150  Constant=Ac_pdtc_prn_tgt(Tgt_posture)*Constant
3160  Tl_ac_burst_wt=0
3170   !
3180  FOR I=1 TO 72
3190    Ac_burst_wt(I)=Ac_ms_tgt_wts(I,Ac_mission)*Nm_elmt_left(I)*Ac_tgt_pk(I)
3200    Tl_ac_burst_wt=Tl_ac_burst_wt+Ac_burst_wt(I)
3210  NEXT I
3220   !
3230  FOR I=1 TO 72
3240    Ac_burst_wt_frc(I)=Ac_burst_wt(I)/FNMax(Tl_ac_burst_wt,.0000001)
3250    Expnt=Ac_burst_wt_frc(I)*Nm_ac_bursts
3260    Nm_elmt=Nm_elmt_left(I)*Constant
3270      !
3280      ! IT IS ASSUMED THAT ALL AIRCRAFT FIRE AT DIFFERENT TARGETS
3290      ! (i.e. THE TARGET SELECTION IS DEPENDENT)
3300      ! CONSEQUENTLY, THE EXPECTED KILLS  FROM A BINOMIAL SEEMS APPROPRIATE
3310      !
3320    Nm_elmt_lost=Ac_tgt_pk(I)*Expnt*Eff_shots
3330    Nm_elmt_lost=FNMin(Nm_elmt_lost,Nm_elmt)
3340      !
3350    Nm_elmt_lost_fl(I)=Nm_elmt_lost_fl(I)+Nm_elmt_lost
3360    Nm_elmt_left(I)=Nm_elmt_left(I)-Nm_elmt_lost
3370  NEXT I
3380   !
3390  FOR I=1 TO 7
3400    Nm_ada_wpn(I)=Nm_elmt_left(I+47)
3410  NEXT I
3420   !
3430  RETURN
3440!
3450!**********************************************************************
3460!
3470 Calc_sub_pass:!
3480 !CALCULATES # OF SUBPASSES MADE BY A WING OF A/C AS THEY STRIKE THE TARGET
3490 !           OR INGRESS/EGRESS OVER THE TARGET.
3500 !           THE AD ATTRITION DATA IS ON A PASS BASIS HENCE, WE MUST
3510 !           SUBDIVIDE THE WING TO SEPERATE THE NUMBER OF A/C AS THEY
3520 !           ARE EXPOSED TO THE AIR DEFENSE ELEMENTS OF THE TARGET.
3530 !
```

Table 5-3. Air attack/Air defense code.

```
3540 !CALC THE EFECTIVENESS OF THE PLANE DELIVERY BASED ON PRECEIVED DENSITY OF
3550 !      AIR DEFENSE.
3560 !
3570  SELECT High_low
3580  CASE 1 !AIR COMMANDER BELIEVES THAT AD IS HIGH DENSITY. PLANES ARE CLOSEL
3590         !SPACED ; THEIR EFFECITVENESS IS DECREASED.
3600    Eff_shots=.8
3610    Eff_engage=.9
3620  CASE 2 !AIR COMMANDER BELIEVES THAT AD IS LOW DENSITY. PLANES ARE FURTHER
3630         !SPACED ; THEIR EFFECITVENESS IS INCREASED.
3640    Eff_shots=.95
3650    Eff_engage=1
3660  END SELECT
3670 !
3680 !CALC THE NUMBER OF EXPOSED PLANES PER PASS AND THE NUMBER OF PASSES
3690  SELECT Cas_bai
3700  CASE 1  !AIR COMMANDER PERFORMING CLOSE AIR SUPPORT.
3710    Nm_sub_pass=INT(Nm_ac_left1/4+.5)
3720    IF Nm_sub_pass=0 THEN
3730      Nm_sub_pass=1
3740      Pass_acft=1
3750      GOTO End_sub_pass
3760    END IF
3770    Pass_acft=Nm_ac_left1/Nm_sub_pass
3780  CASE 2  !AIR COMMANDER PERFORMING BASIC AIR INTERDICTION.
3790    Nm_sub_pass=INT(Nm_ac_left1/8+.5)
3800    IF Nm_sub_pass=0 THEN
3810      Nm_sub_pass=1
3820      Pass_acft=1
3830      GOTO End_sub_pass
3840    END IF
3850    Pass_acft=Nm_ac_left1/Nm_sub_pass
3860  END SELECT
3870 End_sub_pass:    !
3880  RETURN
3890!
3900!
3910!***************************************************************************
3920!
3930 Change_answer:!
3940!
3950  REPEAT
3960    PRINT
3970    PRINT "DO YOU WISH TO CHANGE ANSWERS? (Y/N)"
3980    INPUT Answer$
3990  UNTIL Answer$="Y" OR Answer$="N"
4000  !
4010  Nm_units_chosen=Nm_units+Nm_units_played
4020  IF Nm_units_chosen>20 THEN
4030    BEEP
4040    PRINT
4050    PRINT " ** ERROR:  Number of units played will exceed storage"
```

Table 5-3. Air attack/Air defense code.

```
4060    PRINT "                capability.  You must update or purge results."
4070    Answer$="Y"
4080  END IF
4090!
4100  RETURN
4110!
4120!************************************************************************
*
4130!
4140 Choose_color: !
4150!
4160  PRINT
4170  PRINT
4180  PRINT "AIR ATTACK / AIR DEFENSE MODULE"
4190  REPEAT
4200    PRINT
4210    PRINT "SELECT BLUE OR RED AIR (B/R)"
4220    INPUT Answer$
4230  UNTIL Answer$="B" OR Answer$="R"
4240  IF Answer$="B" THEN
4250    Frc_clr$="BLUE"
4260    Clr$="BL"
4270    Ada_side=2
4280  ELSE
4290    Frc_clr$="RED"
4300    Clr$="RD"
4310    Ada_side=1
4320  END IF
4330   !
4340  RETURN
4350!
4360!************************************************************************
4370!
4380 Cumulate_totals:!
4390!
4400  ASSIGN @Path TO Clr$&"_AIR_INF"&":HP9134,701"
4410  ENTER @Path,1;Tl_ac_flt_ing(*),Tl_ac_flt_egr(*),Tl_ac_flt_stk(*),Tl_ac_lost_ing(*),Tl_ac_lost_egr(*),Tl_ac_lost_stk(*),Tl_ac_lost_ada(*),Tl_elmt_ac(*)
4420!
4430  FOR I=1 TO 4
4440    Tl_ac_flt_ing(I)=Tl_ac_flt_ing(I)+Nm_ac_flt_ing(I)
4450    Tl_ac_flt_egr(I)=Tl_ac_flt_egr(I)+Nm_ac_flt_egr(I)
4460    Tl_ac_flt_stk(I)=Tl_ac_flt_stk(I)+Nm_ac_flt_stk(I)
4470    Tl_ac_lost_ing(I)=Tl_ac_lost_ing(I)+Nm_ac_lost_ing(I)
4480    Tl_ac_lost_egr(I)=Tl_ac_lost_egr(I)+Nm_ac_lost_egr(I)
4490    Tl_ac_lost_stk(I)=Tl_ac_lost_stk(I)+Nm_ac_lost_stk(I)
4500    Tl_ac_lost_ada(I)=Tl_ac_lost_ada(I)+Nm_ac_lost_ada(I)
4510  NEXT I
4520   !
4530  FOR I=1 TO 72
4540    Tl_elmt_ac(I)=Tl_elmt_ac(I)+Nm_elmt_lost_ac(I)
4550  NEXT I
```

Table 5-3. Air attack/Air defense code.

```
4560   !
4570  OUTPUT @Path,1;Tl_ac_flt_ing(*),Tl_ac_flt_egr(*),Tl_ac_flt_stk(*),Tl_ac_lo
st_ing(*),Tl_ac_lost_egr(*),Tl_ac_lost_stk(*),Tl_ac_lost_ada(*),Tl_elmt_ac(*)
4580  ASSIGN @Path TO *
4590!
4600  RETURN
4610!
4620!************************************************************************
4630!
4640 Exit_module:!
4650!
4660  REPEAT
4670    PRINT
4680    INPUT "REMINDER TO SAVE FINAL RESULTS (Y/N)",Answer$
4690  UNTIL Answer$="Y" OR Answer$="N"
4700  IF Answer$="Y" THEN
4710    PRINT
4720    PRINT "SELECT UPDATE RESULTS AND STATUS REPORT FROM MENU"
4730    WAIT 7
4740  ELSE
4750    REPEAT
4760      INPUT "CHOOSE BLUE/RED OR EXIT TO DIME MENU  (COLOR/EXIT)",New_color$
4770    UNTIL New_color$="COLOR" OR New_color$="EXIT"
4780    IF New_color$="COLOR" THEN LOAD "NEW_F3:HP9134,701"
4790    PRINT
4800    PRINT "EXIT FROM AIR DEFENSE/AIR ATTACK MODULE"
4810    Menu$="FINISHED"
4820    LOAD "DIME:HP9134,701"
4830  END IF
4840   !
4850   !
4860  RETURN
4870!
4880!************************************************************************
4890!
4900 File_error_1: !
4910  BEEP
4920  PRINT
4930  PRINT "**ERROR:  ";File$&Disk$;" DOES NOT EXIST ON THE AA/AD DATABASE"
4940  RETURN
4950!
4960 File_error_2: !
4970  BEEP
4980  PRINT
4990  PRINT "ERROR:  ";File$&Disk$;" DOES NOT EXIST ON THE AA/AD DATABASE"
5000  PRINT "            RESUBMIT A/C AND MISSION BY CHOOSING THE FOLLOWING:"
5010  PRINT
5020  PRINT "ENTER A/C TYPE (1-4)"
5030  INPUT Ac_type
5040  CALL Check_var("A/C TYPE",Ac_type,1,4)
5050  IF Flight$="STRIKE" THEN
5060    PRINT
```

Table 5-3. Air attack/Air defense code.

```
5070     PRINT "ENTER AIR COMBAT MISSION (1-5)"
5080     INPUT Ac_mission
5090      !IF Ac_mission=1 THEN INPUT "WHAT IS THE RANDOM SEED",R
5100     CALL Check_var("AC MISSION",Ac_mission,1,5)
5110   END IF
5120   GOSUB Select_prmtrs
5130   File$=Aircraft$&"_"&Flt$
5140   RETURN
5150!
5160!**************************************************************************
5170!
5180 Flight_atrition:!
5190!
5200   Break_point=.5*Nm_ac_left
5210   !
5220   FOR This_unit=1 TO Nm_units
5230     FOR J=1 TO 150
5240       Unit(J)=Unit_store_5_15(This_unit,J)
5250     NEXT J
5260     GOSUB Select_ada_info
5270     Tgt_posture=Unit(75)
5280     T_sub_pass=0
5290     Nm_ac_left1=Nm_ac_left
5300     IF Nm_ac_left<.1 THEN Store_loop
5310     Tac_lost_pass=0
5320     GOSUB Calc_sub_pass
5330     REPEAT
5340       T_sub_pass=T_sub_pass+1
5350       GOSUB Calc_ada_suprsn
5360       ! SET PLOS FOR ADA  HAND HELD ROUNDS
5370       Plos=T_plos(Terr(This_unit))
5380       GOSUB Calc_ada_rnds
5390       GOSUB Calc_ac_losses
5400       Tac_lost_pass=Tac_lost_pass+Nm_psa_ac_lost
5410     UNTIL T_sub_pass>=Nm_sub_pass OR (Tac_lost_pass+Nm_ac_lost_flt)>=Break_
oint OR (Nm_ac_left1-Tac_lost_pass)<.1
5420     Nm_ac_left=FNMax(0,Nm_ac_left-Tac_lost_pass)
5430     Nm_psa_ac_lost=Tac_lost_pass
5440 Store_loop: !
5450     FOR J=1 TO 150
5460       Unit_store_5_15(This_unit,J)=Unit(J)
5470     NEXT J
5480   NEXT This_unit
5490    !
5500   RETURN
5510!
5520!**************************************************************************
5530!
5540 Flight_data:!
5550!
5560   Ac_load=1
5570   Nm_ac_left=Nm_ac_input
```

Table 5-3. Air attack/Air defense code.

```
5580  Nm_ac_lost_flt=0
5590  Altitude_meters=Altitude*.3048
5600  GOSUB Select_units
5610  GOSUB Select_prmtrs
5620   !
5630  File$=Aircraft$&"_"&Flt$
5640  ON ERROR GOSUB File_error_2
5650  ASSIGN @Path TO File$&Disk$
5660  OFF ERROR
5670  PRINTER IS 702
5680  GOSUB Flight_output
5690  PRINTER IS 1
5700  ENTER @Path,1;Flt_profile(*)
5710  ASSIGN @Path TO *
5720  FOR I=1 TO 7
5730    Min_ada_alt(I)=Flt_profile(I)
5740    Max_ada_alt(I)=Flt_profile(I+7)
5750    Ada_range(I)=Flt_profile(I+14)
5760    Ada_prtcptn(I)=Flt_profile(I+21)
5770    Ada_rnd_bs_ld(I)=Flt_profile(I+28)
5780    Ada_rnd_wt(I)=Flt_profile(I+35)
5790    Ada_rnd_frd_eng(I)=Flt_profile(I+42)
5800    Ada_psa_ac(I)=Flt_profile(I+49)
5810    Ada_psk_ac(I)=1
5820  NEXT I
5830   !
5840  File$=Clr$&"_UN_AREA"
5850  ASSIGN @Path TO File$&Disk$
5860  ENTER @Path,1;Unit_type_area(*)
5870  ASSIGN @Path TO *
5880   !
5890  RETURN
5900!
5910!************************************************************************
5920!
5930 Flight_input:!
5940!
5950  PRINT
5960  PRINT "      INPUT THE FOLLOWING FOR THE ";Frc_clr$;" AIR ";Flight$
5970  IF Flight$="EGRESS" THEN
5980    PRINT
5990    PRINT USING Fmtfi;"TOTAL A/C AVAILABLE FOR EGRESS:  ",Nm_ac_avl_egr(*)
6000  END IF
6010  IF Flight$="INGRESS" THEN
6020    PRINT
6030    INPUT "WHAT IS THE GAME TIME?",Tm$
6040  END IF
6050  PRINT
6060  PRINT "ENTER:  # OF A/C, A/C TYPE, ALTITUDE, # UNITS OVERFLOWN"
6070  INPUT Nm_ac_input,Ac_type,Altitude,Nm_units
6080   !
6090  CALL Check_var("A/C TYPE",Ac_type,1,4)
```

Table 5-3. Air attack/Air defense code.

```
6100    !
6110    !
6120  PRINT
6130  PRINT "ENTER: TYPE OF STRIKE -- CAS OR BAI  (INPUT 1 OR 2)"
6140  INPUT Cas_bai
6150  CALL Check_var("TYPE OF STRIKE (CAS OR BAI)",Cas_bai,1,2)
6160  PRINT
6170  PRINT "ENTER: EXPECTED DENSITY OF AD -- HIGH OR LOW  (INPUT 1 OR 2)"
6180  INPUT High_low
6190  CALL Check_var("EXPECTED DENSITY OF AD",High_low,1,2)
6200    !
6210    !
6220  CALL Unit_read(Frc_clr$,Nm_units,Unit_id(*),Terr(*))
6230  Ac_mission=0
6240    !
6250    !SET THE TARGET AREA PARAMETERS FOR USE IN AREA FIRE
6260  Tgt_width=1.0E+8
6270  Tgt_length=1.0E+8
6280    !
6290 Fmt1:IMAGE 31A,2X,4(3D.D,2X)
6300!
6310  RETURN
6320!
6330!**************************************************************************
6340!
6350 Flight_output:!
6360!
6370  GOSUB Select_prmtrs
6380    !
6390  PRINT USING "///"
6400  PRINT "THE FOLLOWING INPUTS WERE CHOSEN FOR THE ";Frc_clr$;" AIR ";Flight$
;":"
6410  PRINT
6420  IF Flight$="INGRESS" THEN
6430    PRINT
6440    PRINT "FLIGHT TIME IS ",Tm$
6450    PRINT
6460  END IF
6470  PRINT " # OF A/C, A/C TYPE, ALTITUDE, # UNITS OVERFLOWN"
6480  PRINT USING Fmt1;Nm_ac_input,Acprint$,Altitude,"ft",Nm_units
6490    !
6500    !
6510  PRINT
6520  SELECT Cas_bai
6530  CASE 1
6540    PRINT "Aircraft performing CAS."
6550  CASE 2
6560    PRINT "Aircraft performing BAI."
6570  END SELECT
6580  PRINT
6590  SELECT High_low
6600  CASE 1
```

Table 5-3. Air attack/Air defense code.

```
6610    PRINT "In a perceived high-density AD environment."
6620   CASE 2
6630    PRINT "In a perceived low-density AD environment."
6640   END SELECT
6650    !
6660    !
6670   PRINT
6680   PRINT USING Fmt2;"UNIT-ID'S OVERFLOWN:   ";Unit_id(*)
6690   PRINT USING Fmt2;"              TERRAIN:   ";Terr(*)
6700   !
6710 Fmt1:IMAGE 9D,5X, 5A, 8D,2A,19D
6720 Fmt2:IMAGE 20A,3X,5(3D,3X)
6730!
6740   RETURN
6750!
6760!****************************************************************************
6770!
6780 Flight_results: !
6790!
6800   IF Nm_ac_lost_flt>Nm_ac_input THEN Nm_ac_lost_flt=Nm_ac_input
6810   IF Flight$="STRIKE" THEN
6820     PRINT
6830     PRINT USING Fmtfr2;"RESULTS OF THE",Acprint$,Flt$,"STRIKE:  ",Nm_ac_inpu
t,"FLOWN",Nm_ac_lost_flt,"KILLED",Tl_ac_passes,"PASSES"
6840     PRINT
6850     R_target_name$[1,125]="T55  DF   BMP73DF   BRDM3BRDM5AT-75AGS17T12  CMD-
VDF   DF   DF   DF   DF   BMPATBTR  DF-ICDF-ICDF-ICARTY ARTY ARTY ARTY ARTY "
6860     R_target_name$[126,250]="ARTY ARTY MORTRMORTRMORTRMORTRMRL  MRL  MRL  MR
L  INF  INF  INF  INF  INF  SARMSSARMSSARMSSARMSSARMSSARMSSARMSZSU-XSA-13SA-6 "
6870     R_target_name$[251,350]="ADA  ADA  SA-14ADAHHF-TRKJ4TRKWATERCGO-TNATRKEW
TRKEWTRKENGR OBSCEAVLB PONBRENGEQENGEQMATHEMATHEAATHE"
6880     B_target_name$[1,125]="DF   FAV-TM551 FAV40HMV-GDF   DRAGNLAW  DF   CMD-
VDF   DF   DF   DF   DF   HMV40DF-ICDF-ICDF-ICDF-ICARTY ARTY ARTY ARTY ARTY "
6890     B_target_name$[126,250]="ARTY ARTY MORTRMORTRMORTRMORTRMLRSTMLRSTMLRSTML
RSTINF  INF  INF  INF  INF  SARMSSARMSSARMSSARMSSARMSSARMSSARMSVULCNAVNGRIHAWK"
6900     B_target_name$[251,350]="ADA  ADA  STINGADAHHF-TRKJ4TRKWATERCGO-TNATRKEW
TRKEWTRKENGR OBSCEAVLB PONBRENGEQENGEQMATHEMATHEAATHE"
6910     SELECT Frc_clr$
6920     CASE "BLUE"
6930       Target_victim$="RED"
6940       T$=R_target_name$
6950     CASE "RED"
6960       Target_victim$="BLUE"
6970       T$=B_target_name$
6980     END SELECT
6990     FOR Out=1 TO 72
7000       O(Out)=Nm_elmt_lost_fl(Out)
7010     NEXT Out
7020     PRINT Target_victim$:" ELEMENT LOSSES  INFLICTED:"
7030     PRINT
7040     FOR X=10 TO 70 STEP 10
7050       PRINT USING Fmtfr1a:T$[(X-10)*5+1;5],T$[(X-9)*5+1;5],T$[(X-8)*5+1;5],T
$[(X-7)*5+1;5],T$[(X-6)*5+1;5],T$[(X-5)*5+1;5],T$[(X-4)*5+1;5],T$[(X-3)*5+1;5]
```

Table 5-3. Air attack/Air defense code.

```
7060        PRINT USING Fmtfr1b:T$[(X-2)*5+1;5],T$[(X-1)*5+1;5]
7070        PRINT USING Fmtfr1:O(X-9),O(X-8),O(X-7),O(X-6),O(X-5),O(X-4),O(X-3),O(
X-2),O(X-1),O(X)
7080      NEXT X
7090      PRINT USING "12A,8D.D,5X,12A,8D.D":"  POL(gal) ";O(71);" AMMO(tons) ";O(
72)
7100      PRINT
7110      IF Brgg=0 THEN GOTO 7180
7120     !IF Brgg=1 THEN GOTO 7150
7130     !IF Brgg=2 THEN GOTO 7130
7140      IF Destroy=0 THEN 7170
7150      PRINT "BRIDGE DESTROYED"  '   THE SEED IS",R3
7160      GOTO 7180
7170      PRINT "BRIDGE NOT DESTROYED"  ! THE SEED IS ",R3
7180    ELSE
7190      PRINT
7200      PRINT USING Fmtfr3;"RESULTS OF THE",Acprint$,Flight$,":  ",Nm_ac_input,"
FLOWN",Nm_ac_lost_flt,"KILLED"
7210    END IF
7220     !
7230    IF Answer$<>"XYZ" THEN
7240      REPEAT
7250        PRINT
7260        PRINT "DO YOU WISH TO SUBTRACT THE ABOVE LOSSES? (Y/N)"
7270        INPUT Answer$
7280      UNTIL Answer$="Y" OR Answer$="N"
7290    END IF
7300     !
7310    IF Answer$="Y" THEN
7320      Answer$="XYZ"
7330      PRINTER IS 702
7340      GOSUB Flight_results
7350      GOSUB Store_results
7360      GOSUB Add_results
7370      PRINTER IS 1
7380    END IF
7390     !
7400 Fmtfr1:IMAGE   10(2X,3D.D)
7410 Fmtfr1a:IMAGE #,7(2X,5A)
7420 Fmtfr1b:IMAGE   2(2X,5A)
7430 Fmtfr2:IMAGE 15A,4A,7A,10A,3D.D,1X,6A,3D.D,1X,7A,3D,1X,7A
7440 Fmtfr3:IMAGE 15A,5A,7A,3A,3D.D,1X,6A,3D.D,1X,7A
7450!
7460    RETURN
7470!
7480!*************************************************************************
7490!
7500 Init_totals: !
7510!
7520    FOR I=1 TO 4
7530      Nm_ac_flt_ing(I)=0
7540      Nm_ac_flt_egr(I)=0
```

Table 5-3. Air attack/Air defense code.

```
7550     Nm_ac_flt_stk(I)=0
7560     Nm_ac_lost_ing(I)=0
7570     Nm_ac_lost_egr(I)=0
7580     Nm_ac_lost_stk(I)=0
7590     Nm_ac_lost_ada(I)=0
7600     Nm_ac_avl_stk(I)=0
7610     Nm_ac_avl_egr(I)=0
7620   NEXT I
7630   FOR I=1 TO 20
7640     FOR J=1 TO 150
7650       Unitstore_20_15(I,J)=0
7660     NEXT J
7670     Unit_id_20_150(I)=0
7680   NEXT I
7690   FOR I=1 TO 72
7700     Nm_elmt_lost_ac(I)=0
7710   NEXT I
7720   Nm_units_played=0
7730     !
7740   RETURN
7750!
7760'***************************************************************************
7770!
7780 List_debug_flt: '
7790!
7800   PRINT
7810   PRINT Aircraft$&Flt$&Disk$,Flt_profile(*)
7820   PRINT
7830   PRINT "MIN_ADA_ALT",Min_ada_alt(*)
7840   PRINT
7850   PRINT "MAX_ADA_ALT",Max_ada_alt(*)
7860   PRINT
7870   PRINT "ADA_RANGE  ",Ada_range(*)
7880   PRINT
7890   PRINT "ADA_PRTCPTN",Ada_prtcptn(*)
7900   PRINT
7910   PRINT "ADA_RND_BS_LD",Ada_rnd_bs_ld(*)
7920   PRINT
7930   PRINT "ADA_RND_WT ",Ada_rnd_wt(*)
7940   PRINT
7950   PRINT "ADA_RND_FRD_ENG",Ada_rnd_frd_eng(*)
7960   PRINT
7970   PRINT "ADA_PSA_AC ",Ada_psa_ac(*)
7980   PRINT
7990   PRINT "ADA_PSK_AC ",Ada_psk_ac(*)
8000   PRINT
8010   PRINT "NM_ADA_WPN ",Nm_ada_wpn(*)
8020   PRINT
8030   PRINT "NM_ADA_WPN_AVL",Nm_ada_wpn_avl(*)
8040   PRINT
8050   PRINT "UNIT_TYPE ";Unit_type,"UNIT_POSTURE ";Unit_posture,"UNIT_AREA ";Uni
t_area
```

Table 5-3. Air attack/Air defense code.

```
8060   PRINT
8070   PRINT "ADA_AREA_FCTR",Ada_area_fctr(*)
8080   PRINT
8090   PRINT "NM_ADA_WPN_ENG",Nm_ada_wpn_eng(*)
8100   PRINT
8110   PRINT "ADA_BS_LD_WT",Ada_bs_ld_wt(*)
8120   PRINT
8130   PRINT "TL_ADA_BS_LD_WT",Tl_ada_bs_ld_wt
8140   PRINT
8150   PRINT "ADA_RND_FRD",Ada_rnd_frd(*)
8160   PRINT
8170   PRINT "ADA_LDS_FRD",Ada_lds_frd(*)
8180   PRINT
8190   PRINT "TL_ADA_TONS_FRD",Tl_ada_tons_frd,"ADA_TONS_AVL",Ada_tons_avl
8200    !
8210   RETURN
8220!
8230!***************************************************************************
8240!
8250 List_debug_stk: !
8260!
8270   PRINT
8280   PRINT Wpn_ld_template$(Ac_load)&Disk$,Stk_profile(*)
8290   PRINT
8300   PRINT "WPN_LOAD_CODE",Wpn_load_code
8310   PRINT
8320   PRINT "AC_AREA_PRN_TGT",Ac_area_prn_tgt
8330   PRINT
8340   PRINT "MIN_AC_PASSES ",Min_ac_passes
8350   PRINT
8360   PRINT "NM_AC_RNDS_ENG  ",Nm_ac_rnds_eng
8370   PRINT
8380   PRINT "NM_AC_RND_BS_LD",Nm_ac_rnd_bs_ld
8390   PRINT
8400   PRINT "AC_PDTC_PRN_TGT",Ac_pdtc_prn_tgt(*)
8410   PRINT
8420   PRINT "AC_PLOS_PRN_TGT",Ac_plos_prn_tgt(*)
8430   PRINT
8440   PRINT "AC_TGT_PK      ",Ac_tgt_pk(*)
8450   PRINT
8460   PRINT "AC_TGT_AREA  ",Ac_tgt_area(*)
8470   PRINT
8480   PRINT "PRN_TGT_WDS_FRC",Prn_tgt_wds_frc
8490   PRINT
8500   PRINT "PRSNL_FXHL_FRC",Prsnl_fxhl_frc
8510   PRINT
8520   PRINT "TGT_AREA   ",Tgt_area
8530   PRINT
8540   PRINT "NM_AC_PASSES",Nm_ac_passes
8550   PRINT
8560   PRINT "NM_AC_BURSTS",Nm_ac_bursts
8570   PRINT
```

Table 5-3. Air attack/Air defense code.

```
8580  PRINT "CONSTANT",Constant
8590  PRINT
8600  PRINT "AC_BURST_WT",Ac_burst_wt(*)
8610  PRINT
8620  PRINT "TL_AC_BURST_WT",Tl_ac_burst_wt
8630  PRINT
8640  PRINT "AC_BURST_WT_FRC",Ac_burst_wt_frc(*)
8650  PRINT
8660  PRINT "NM_VEHICLES ";Nm_vehicles,"VEHICLES_LOST ",Vehicles_lost
8670  PRINT
8680  PRINT "NM_INFANTRY ";Nm_infantry,"MOUNT_RATIO ";Mount_ratio
8690  PRINT
8700  PRINT "MOUNT_LOSSES ";Mount_losses,"NM_ELMT_LOST_FL(10) ";Nm_elmt_lost_fl(
10)
8710   !
8720  RETURN
8730!
8740!*************************************************************************
8750!
8760 List_debug_unit: !
8770!
8780  ASSIGN @Path TO "UNITFILE:HP9134,701"
8790  FOR I=1 TO Nm_units
8800    ENTER @Path,Unit_id(I);Unit_info(*)
8810    PRINT
8820    PRINT "UNITFILE RECORD";Unit_id(I),Unit_info(*)
8830  NEXT I
8840  ASSIGN @Path TO *
8850   !
8860  FOR I=1 TO Nm_units
8870    FOR J=1 TO 150
8880      Unit_info(J)=Unit_store_5_15(I,J)
8890    NEXT J
8900    PRINT
8910    PRINT "UNIT_5_150 RECORD";Unit_id(I),Unit_info(*)
8920  NEXT I
8930   !
8940  FOR I=1 TO Nm_units_played
8950    FOR J=1 TO 150
8960      Unit_info(J)=Unitstore_20_15(I,J)
8970    NEXT J
8980    PRINT
8990    PRINT "UNIT_20_150 RECORD";Unit_id_20_150(I),Unit_info(*)
9000  NEXT I
9010   !
9020  RETURN
9030!
9040!*************************************************************************
9050!
9060 List_results:!
9070!
9080  PRINT
```

Table 5-3. Air attack/Air defense code.

```
9090  PRINT
9100  PRINT USING Fmtlr1:Frc_clr$;" AIR CURRENT RESULTS"
9110  PRINT USING Fmtlr2:"TYPE (1-4)"
9120  PRINT USING Fmtlr3:"TOTAL AIRCRAFT INGRESS FLIGHTS",Nm_ac_flt_ing(*)
9130  PRINT USING Fmtlr3;"TOTAL AIRCRAFT EGRESS  FLIGHTS",Nm_ac_flt_egr(*)
9140  PRINT USING Fmtlr3;"TOTAL AIRCRAFT STRIKES         ",Nm_ac_flt_stk(*)
9150  PRINT USING Fmtlr3;"TOTAL AIRCRAFT INGRESS LOSSES ",Nm_ac_lost_ing(*)
9160  PRINT USING Fmtlr3;"TOTAL AIRCRAFT EGRESS  LOSSES ",Nm_ac_lost_egr(*)
9170  PRINT USING Fmtlr3;"TOTAL AIRCRAFT STRIKE  LOSSES ",Nm_ac_lost_stk(*)
9180  PRINT USING Fmtlr3;"TOTAL AIRCRAFT LOST TO ADA    ",Nm_ac_lost_ada(*)
9190  PRINT USING Fmtlr3;"AIRCRAFT AVAILABLE FOR COMBAT ",Nm_ac_avl_stk(*)
9200  PRINT USING Fmtlr3;"AIRCRAFT AVAILABLE FOR EGRESS ",Nm_ac_avl_egr(*)
9210  PRINT USING Fmtlr4;"TYPE ( 1-70, POL, AMMO)"
9220  PRINT USING Fmtlr5;"TOTAL TARGET ELEMENT LOSSES    ",Nm_elmt_lost_ac(*)
9230  PRINT USING Fmtlr2;"UNIT_ID"
9240  PRINT USING Fmtlr6;"UNITS REMAINING FOR UPDATE     ",Unit_id_20_150(*)
9250  !
9260 Fmtlr1:IMAGE     20X,4A,25A
9270 Fmtlr2:IMAGE /,42X,10A,/
9280 Fmtlr3:IMAGE 30A,4(2X,4D.1D)
9290 Fmtlr4:IMAGE /,34X,24A,/
9300 Fmtlr5:IMAGE 30A,1(4(2X,6D.1D),/),17(30X,4(2X,6D.1D),/)
9310 Fmtlr6:IMAGE 30A,1(4(3X,3D),/),5(30X,4(3X,3D),/)
9320  !
9330  RETURN
9340!
9350!**************************************************************************
9360!
9370 List_totals:!
9380!
9390  File$=Status_clr$&"_AIR_INF"
9400  ASSIGN @Path TO File$&":HP9134,701"
9410  ENTER @Path,1;Tl_ac_flt_ing(*),Tl_ac_flt_egr(*),Tl_ac_flt_stk(*),Tl_ac_los
t_ing(*),Tl_ac_lost_egr(*),Tl_ac_lost_stk(*),Tl_ac_lost_ada(*),Tl_elmt_ac(*)
9420  ASSIGN @Path TO *
9430!
9440  PRINT
9450  PRINT
9460  PRINT USING Fmtlt1;Status_frc_clr$;" AIR ACCUMULATIVE RESULTS"
9470  PRINT USING Fmtlt2;"TYPE (1-4)"
9480  PRINT USING Fmtlt3;"TOTAL AIRCRAFT INGRESS FLIGHTS",Tl_ac_flt_ing(*)
9490  PRINT USING Fmtlt3;"TOTAL AIRCRAFT EGRESS  FLIGHTS",Tl_ac_flt_egr(*)
9500  PRINT USING Fmtlt3:"TOTAL AIRCRAFT STRIKES         ",Tl_ac_flt_stk(*)
9510  PRINT USING Fmtlt3;"TOTAL AIRCRAFT INGRESS LOSSES ",Tl_ac_lost_ing(*)
9520  PRINT USING Fmtlt3;"TOTAL AIRCRAFT EGRESS  LOSSES ",Tl_ac_lost_egr(*)
9530  PRINT USING Fmtlt3;"TOTAL AIRCRAFT STRIKE  LOSSES ",Tl_ac_lost_stk(*)
9540  PRINT USING Fmtlt3;"TOTAL AIRCRAFT LOST TO ADA    ",Tl_ac_lost_ada(*)
9550  PRINT USING Fmtlt4;"TYPE ( 1-70, AMMO, POL)"
9560  PRINT USING Fmtlt5;"TOTAL TARGET ELEMENT LOSSES   ",Tl_elmt_ac(*)
9570  !
9580 Fmtlt1:IMAGE     20X,4A,25A
9590 Fmtlt2:IMAGE /,42X,10A,/
```

Table 5-3. Air attack/Air defense code.

```
9600 Fmtlt3:IMAGE 30A,4(2X,3D.1D)
9610 Fmtlt4:IMAGE /,34X,24A,/
9620 Fmtlt5:IMAGE 30A,1(4(2X,3D.1D),/),17(30X,4(2X,3D.1D),/)
9630  !
9640  RETURN
9650!
9660!**************************************************************************
9670!
9680 Menu_selection:!
9690!
9700  PRINT USING "///"
9710  PRINT "DIME AIR DEFENSE/AIR ATTACK MENU:  SELECT OPTION"
9720  PRINT "     (1) ";Frc_clr$;" AIR INGRESS"
9730  PRINT "     (2) ";Frc_clr$;" AIR EGRESS"
9740  PRINT "     (3) ";Frc_clr$;" AIR STRIKE"
9750  PRINT "     (4) STATUS REPORT"
9760  PRINT "     (5) UPDATE RESULTS"
9770  PRINT "     (6) EXIT AIR DEFENSE MODULE"
9780  INPUT Menu_optn
9790  !
9800  SELECT Menu_optn
9810  CASE 1
9820    PRINTER IS 702
9830    PRINT USING "@"
9840    PRINTER IS 1
9850    Option$="INGRESS"
9860    Flight$="INGRESS"
9870  CASE 2
9880    PRINTER IS 702
9890    PRINT USING "@"
9900    PRINTER IS 1
9910    Option$="EGRESS"
9920    Flight$="EGRESS"
9930  CASE 3
9940    PRINTER IS 702
9950    PRINT USING "@"
9960    PRINTER IS 1
9970    Option$="STRIKE"
9980    Flight$="STRIKE"
9990  CASE 4
10000   Option$="STATUS"
10010 CASE 5
10020   Option$="UPDATE"
10030 CASE 6
10040   Option$="EXIT"
10050 CASE ELSE
10060   Option$="ERROR"
10070 END SELECT
10080  !
10090 RETURN
10100!
10110!**************************************************************************
*
```

Table 5-3. Air attack/Air defense code.

```
10120!
10130 Select_ada_info:!
10140!
10150 FOR X=1 TO 7
10160   Nm_ada_wpn(X)=Unit(X+47)
10170 NEXT X
10180   !
10190 Tl_ada_tons_frd=0
10200 Ada_tons_avl=Unit(133)
10210 CALL Check_var("UNIT(80)",Unit(80),0.,100.100)
10220 Ada_suprsn_pct=Unit(80)
10230 CALL Check_var("UNIT(75)",Unit(75),1,4)
10240 Unit_posture=Unit(75)
10250 CALL Check_var("UNIT(78)",Unit(78),1.0,2.9)
10260 Unit_clr_type=Unit(78)
10270   !
10280   !UNPACK UNIT TYPE AND COLOR
10290 Clr=INT(Unit_clr_type)
10300 Unit_type=Unit_clr_type-Clr
10310 Unit_type=Unit_type*10+1
10320   !
10330   !RESET DETECTION STATUS
10340 Unit(91)=3.2
10350 IF Unit(92)<=3 THEN
10360   Unit(92)=3
10370 END IF
10380   !
10390 RETURN
10400!
10410!***************************************************************************
*
10420!
10430 Select_prmtrs:!
10440!
10450 IF Frc_clr$="BLUE" THEN
10460   SELECT Ac_type
10470   CASE 1
10480     Acprint$="A10"
10490     Aircraft$="A10"
10500   CASE 2
10510     Acprint$="F16"
10520     Aircraft$="F16"
10530   CASE 3
10540     Acprint$="AH-1"
10550     Aircraft$="AH64"
10560   CASE 4
10570     Acprint$="OH58"
10580     Aircraft$="OH58"
10590   END SELECT
10600 ELSE
10610   SELECT Ac_type
10620   CASE 1
```

Table 5-3. Air attack/Air defense code.

```
10630     Acprint$="SU25"
10640     Aircraft$="M28"
10650   CASE 2
10660     Acprint$="MI27"
10670     Aircraft$="M27"
10680   CASE 3
10690     Acprint$="HIND"
10700     Aircraft$="HIND"
10710   CASE 4
10720     Acprint$="HIP"
10730     Aircraft$="HIP"
10740   END SELECT
10750 END IF
10760  !
10770 SELECT Flight$
10780 CASE "INGRESS"
10790   Flt$="ING"
10800 CASE "EGRESS"
10810   Flt$="EGR"
10820 CASE "STRIKE"
10830   SELECT Ac_mission
10840   CASE 1
10850     Flt$="BRG"
10860   CASE 2
10870     Flt$="ARMOR"
10880   CASE 3
10890     Flt$="PRSNNL"
10900   CASE 4
10910     Flt$="POL"
10920   CASE 5
10930     Flt$="AMMO"
10940   END SELECT
10950 END SELECT
10960!
10970 SELECT Tgt_posture
10980 CASE 1
10990   Activity$=" ATTACK"
11000 CASE 2
11010   Activity$=" DEFEND"
11020 CASE 3
11030   Activity$="RESERVE"
11040 CASE 4
11050   Activity$="   MOVE"
11060 END SELECT
11070!
11080 SELECT Tgt_type
11090 CASE 0
11100   Tgt_type$="  COMBAT"
11110 CASE 1
11120   Tgt_type$="    ARTY"
11130 CASE 2
11140   Tgt_type$="     ADA"
```

Table 5-3. Air attack/Air defense code.

```
11150 CASE 3
11160   Tgt_type$="    FARP"
11170 CASE 4
11180   Tgt_type$="   CP/HQ"
11190 CASE 5
11200   Tgt_type$="    ENGR"
11210 CASE 6
11220   Tgt_type$="POL/AMMO"
11230 CASE 7
11240   Tgt_type$="   MAINT"
11250 CASE 8
11260   Tgt_type$="  BRIDGE"
11270 CASE 9
11280   Tgt_type$="COMMO/EW"
11290 END SELECT
11300!
11310 RETURN
11320!
11330!**************************************************************************
*
11340!
11350 Select_stk_info:!
11360!
11370 Nm_vehicles=0
11380 Nm_infantry=0
11390 FOR I=1 TO Nm_units
11400   FOR J=1 TO 70
11410     Nm_elmt_tgt(J)=Nm_elmt_tgt(J)+Unit_store_5_15(I,J)*Tgt_expsr_frc(I)
11420   NEXT J
11430   Nm_elmt_tgt(71)=Nm_elmt_tgt(71)+Unit_store_5_15(I,105)*Tgt_expsr_frc(I)
11440   Nm_elmt_tgt(72)=Nm_elmt_tgt(72)+Unit_store_5_15(I,125)*Tgt_expsr_frc(I)
11450   Tl_ada_tons=Tl_ada_tons+Unit_store_5_15(I,133)*Tgt_expsr_frc(I)
11460     !
11470   Suprsn_frc_1=INT(Unit_store_5_15(I,80))
11480   Suprsn_frc_2=Unit_store_5_15(I,80)-Suprsn_frc_1
11490   Suprsn_frc_1=Suprsn_frc_1/100
11500   Tl_suprsn_1=Tl_suprsn_1+Suprsn_frc_1
11510   Tl_suprsn_2=Tl_suprsn_2+Suprsn_frc_2
11520 NEXT I
11530  !
11540 Tl_suprsn_1=Tl_suprsn_1/FNMax(Nm_units,.0000001)
11550 Tl_suprsn_2=Tl_suprsn_2/FNMax(Nm_units,.0000001)
11560 Ada_suprsn_pct=100*Tl_suprsn_1+Tl_suprsn_2
11570  !
11580 SELECT Tgt_posture
11590 CASE 1,4
11600   FOR J=36 TO 40
11610     Nm_infantry=Nm_elmt_tgt(J)+Nm_infantry
11620   NEXT J
11630   FOR J=16 TO 20
11640     Nm_vehicles=Nm_elmt_tgt(J)+Nm_vehicles
11650   NEXT J
```

Table 5-3. Air attack/Air defense code.

```
11660   FOR J=1 TO 5
11670     Inf=J+35
11680     Frac_inf(J)=Nm_elmt_tgt(Inf)/FNMax(Nm_infantry,1)
11690   NEXT J
11700   Mount_ratio=Nm_infantry/FNMax(Nm_vehicles,1)
11710   Mount_ratio=FNMin(Mount_ratio,8)
11720   FOR J=1 TO 5
11730     Inf=J+35
11740     Nm_elmt_tgt(Inf)=FNMax(Nm_infantry-Nm_vehicles*Mount_ratio,0)*Frac_in
(J)
11750   NEXT J
11760 END SELECT
11770   !
11780 FOR X=1 TO 7
11790   Nm_ada_wpn(X)=Nm_elmt_tgt(X+47)
11800 NEXT X
11810 Ada_tons_avl=Tl_ada_tons
11820   !
11830 FOR I=1 TO 72
11840   Nm_elmt_left(I)=Nm_elmt_tgt(I)
11850 NEXT I
11860!
11870!*************************************************************************
*
11880!
11890 Select_units:!
11900!
11910 FOR I=1 TO Nm_units
11920   Unit_id_ptr=0
11930   Cnt=0
11940   REPEAT
11950     Cnt=Cnt+1
11960     IF Unit_id_20_150(Cnt)=Unit_id(I) THEN
11970       Unit_id_ptr=Cnt
11980       FOR J=1 TO 150
11990         Unit_store_5_15(I,J)=Unitstore_20_15(Cnt,J)
12000       NEXT J
12010     END IF
12020   UNTIL Cnt>=Nm_units_played OR Unit_id_20_150(I)=Unit_id(I)
12030     !
12040   IF Unit_id_ptr=0 THEN
12050     ASSIGN @Path TO "UNITFILE:HP9134,701"
12060     ENTER @Path,Unit_id(I);Unit_info(*)
12070     ASSIGN @Path TO *
12080     FOR J=1 TO 150
12090       Unit_store_5_15(I,J)=Unit_info(J)
12100     NEXT J
12110   END IF
12120 NEXT I
12130   !
12140 FOR I=Nm_units+1 TO 5
12150   FOR J=1 TO 150
```

Table 5-3.   Air attack/Air defense code.

```
12160      Unit_store_5_15(I,J)=0
12170    NEXT J
12180 NEXT I
12190  !
12200 RETURN
12210!
12220!*************************************************************************
*
12230!
12240 Status_report:!
12250!
12260 Nm_option=0
12270 WHILE Nm_option<>7
12280   PRINT
12290   PRINT "STATUS REPORT MENU:  SELECT OPTION"
12300   PRINT "     (1) LIST CURRENT RESULTS"
12310   PRINT "     (2) LIST BLUE AIR ACCUMULATIVE RESULTS"
12320   PRINT "     (3) LIST RED AIR ACCUMULATIVE RESULTS"
12330   PRINT "     (4) LIST DEBUG UNIT INFORMATION"
12340   PRINT "     (5) LIST DEBUG FLIGHT INFORMATION"
12350   PRINT "     (6) LIST DEBUG STRIKE INFORMATION"
12360   PRINT "     (7) EXIT"
12370   INPUT Nm_option
12380     !
12390   SELECT Nm_option
12400   CASE 1,2,3,4,5,6
12410     REPEAT
12420       PRINT
12430       PRINT "DISPLAY RESULTS ON PRINTER OR SCREEN? (P/S)"
12440       INPUT Answer$
12450     UNTIL Answer$="P" OR Answer$="S"
12460     IF Answer$="P" THEN
12470       PRINTER IS 702
12480       PRINT USING "@"
12490     END IF
12500   END SELECT
12510     !
12520   SELECT Nm_option
12530   CASE 1
12540     GOSUB List_results
12550   CASE 2
12560     Status_clr$="BL"
12570     Status_frc_clr$="BLUE"
12580     GOSUB List_totals
12590   CASE 3
12600     Status_clr$="RD"
12610     Status_frc_clr$="RED"
12620     GOSUB List_totals
12630   CASE 4
12640     GOSUB List_debug_unit
12650   CASE 5
12660     GOSUB List_debug_flt
```

Table 5-3. Air attack/Air defense code.

```
12670   CASE 6
12680     GOSUB List_debug_stk
12690   CASE 7
12700   CASE ELSE
12710     BEEP
12720   END SELECT
12730   PRINTER IS 1
12740 END WHILE
12750   !
12760 RETURN
12770!
12780!**********************************************************************
*
12790!
12800 Store_results:!
12810!
12820 FOR I=1 TO Nm_units
12830   Unit_id_ptr=0
12840   Cnt=0
12850   REPEAT
12860     Cnt=Cnt+1
12870     IF Unit_id_20_150(Cnt)=Unit_id(I) THEN
12880       Unit_id_ptr=Cnt
12890       FOR J=1 TO 150
12900         Unitstore_20_15(Cnt,J)=Unit_store_5_15(I,J)
12910       NEXT J
12920     END IF
12930   UNTIL Cnt>=Nm_units_played OR Unit_id_20_150(Cnt)=Unit_id(I)
12940     !
12950   IF Unit_id_ptr=0 AND Nm_units_played<20 THEN
12960     Nm_units_played=Nm_units_played+1
12970     Cnt=Nm_units_played
12980     FOR J=1 TO 150
12990       Unitstore_20_15(Cnt,J)=Unit_store_5_15(I,J)
13000     NEXT J
13010     Unit_id_20_150(Cnt)=Unit_id(I)
13020   END IF
13030 NEXT I
13040   !
13050 RETURN
13060!
13070!**********************************************************************
*
13080!
13090 Strike_aportion:!
13100!
13110 Mount_loss_tot=0
13120   !LOOP TO APPORTION INFANTRY LOSSES
13130 FOR I=1 TO Nm_units
13140   Mount_losses=0
13150     !TAKE INFANTRY ON GROUND OUT FIRST
13160   FOR J=1 TO 5
```

Table 5-3. Air attack/Air defense code.

```
13170      Inf=J+35
13180      Frct=Unit_store_5_15(I,Inf)*Tgt_expsr_frc(I)/FNMax(Nm_vehicles*Mount_
atio*Frac_inf(J)+Nm_elmt_tgt(Inf),.0000001) !RESTORE DENOMINATOR TO REPRESENT
13190                               ! INFANTRY BEFORE MOUNTING
13200      Unit_store_5_15(I,Inf)=Unit_store_5_15(I,Inf)-Frct*Nm_elmt_lost_fl(I
)
13210    NEXT J
13220     !NOW REMOVE PERSONNEL LOSSES FROM VEHICLES
13230    SELECT Tgt_posture
13240    CASE 1,4
13250      Perrsnl=0
13260      FOR J=16 TO 20
13270        Frct=Unit_store_5_15(I,J)*Tgt_expsr_frc(I)/FNMax(Nm_elmt_tgt(J),.00
0001)
13280        V_lost_unit=Frct*Nm_elmt_lost_fl(J)
13290        Perrsnl=V_lost_unit*Mount_ratio+Perrsnl
13300      NEXT J
13310      Tot_uni_inf=0
13320      FOR J=36 TO 40
13330        Tot_uni_inf=Tot_uni_inf+Unit_store_5_15(I,J)
13340      NEXT J
13350      Perrsnl=FNMin(Tot_uni_inf,Perrsnl)
13360      Mount_losses=Mount_losses+Perrsnl
13370      FOR J=1 TO 5
13380        Inf=J+35
13390        Unit_store_5_15(I,Inf)=Unit_store_5_15(I,Inf)-Perrsnl*Frac_inf(J)
13400      NEXT J
13410    END SELECT
13420    Mount_loss_tot=Mount_loss_tot+Mount_losses
13430 NEXT I
13440 FOR J=1 TO 5
13450    Inf=J+35
13460    Nm_elmt_lost_fl(Inf)=Nm_elmt_lost_fl(Inf)+Mount_loss_tot*Frac_inf(J)
13470 NEXT J
13480 FOR I=1 TO Nm_units
13490    FOR J=1 TO 70
13500      Frct=Unit_store_5_15(I,J)*Tgt_expsr_frc(I)/FNMax(Nm_elmt_tgt(J),.0000
01)
13510      IF J>35 AND J<41 THEN 13530
13520      Unit_store_5_15(I,J)=Unit_store_5_15(I,J)-Frct*Nm_elmt_lost_fl(J)
13530    NEXT J
13540    Frct=Unit_store_5_15(I,105)*Tgt_expsr_frc(I)/FNMax(Nm_elmt_tgt(71),.0000
001)
13550    Unit_store_5_15(I,105)=Unit_store_5_15(I,105)-Frct*Nm_elmt_lost_fl(71)
13560      !
13570    Frct=Unit_store_5_15(I,125)*Tgt_expsr_frc(I)/FNMax(Nm_elmt_tgt(72),.0000
001)
13580    Unit_store_5_15(I,125)=Unit_store_5_15(I,125)-Frct*Nm_elmt_lost_fl(72)
13590      !
13600    Frct=Unit_store_5_15(I,133)*Tgt_expsr_frc(I)/FNMax(Tl_ada_tons,.0000001)
13610    Unit_store_5_15(I,133)=Unit_store_5_15(I,133)-Frct*Tl_ada_tons_frd
13620      !
```

Table 5-3. Air attack/Air defense code.

```
13630    Unit_store_5_15(I.141)=Unit_store_5_15(I,141)+Frct*Tl_ada_tons_frd
13640    Unit_store_5_15(I,91)=3.2
13650    IF Unit_store_5_15(I,92)<=3 THEN
13660      Unit_store_5_15(I,92)=3
13670    END IF
13680 NEXT I
13690  !
13700 RETURN
13710!
13720!***************************************************************************
*
13730!
13740 Strike_atrition:!
13750!
13760 Break_point=Nm_ac_input*Ac_brkpt_frc
13770 Ac_load=0
13780 Tl_ac_passes=0
13790 R4=0
13800 REPEAT
13810    Ac_load=Ac_load+1
13820    Ac_pass=0
13830    GOSUB Strike_profile
13840    GOSUB Strike_info
13850    PRINT
13860    REPEAT
13870      Ac_pass=Ac_pass+1
13880      Tl_ac_passes=Tl_ac_passes+1
13890      T_sub_pass=0
13900      Nm_ac_left1=Nm_ac_left
13910      IF Nm_ac_left<.1 THEN Strk_apport
13920      Tac_lost_pass=0
13930      GOSUB Calc_sub_pass
13940      REPEAT
13950        T_sub_pass=T_sub_pass+1
13960        GOSUB Calc_ada_suprsn
13970         ! SET PLOS FOR ADA HAND HELD ROUNDS
13980        Plos=T_plos(Terr(1))
13990        GOSUB Calc_ada_rnds
14000        GOSUB Calc_ac_losses
14010         !
14020         !
14030        SELECT Wpn_load_code
14040        CASE 1
14050          GOSUB Calc_area_loss
14060        CASE 2,3
14070          GOSUB Calc_point_loss
14080        CASE ELSE
14090          BEEP
14100          PRINT "**ERROR: IN WPN_LOAD_CODE FOR ";File$&Disk$
14110        END SELECT
14120        Tac_lost_pass=Tac_lost_pass+Nm_psa_ac_lost
14130      UNTIL T_sub_pass>=Nm_sub_pass OR Nm_ac_lost_flt>=Break_point OR (Nm_ac
_left1-Tac_lost_pass)<.1
```

Table 5-3. Air attack/Air defense code.

```
14140      Nm_ac_left=FNMax(0,Nm_ac_left1-Tac_lost_pass)
14150      Nm_psa_ac_lost=FNMin(Nm_ac_left1,Tac_lost_pass)
14160       !
14170      PRINT USING Fmt_sa1;"DELIVERY PROFILE:   ";File$;"  PASS ";Ac_pass;"
;Acprint$;"  LOST ";Nm_psa_ac_lost
14180      PRINTER IS 702
14190        !PRINT
14200      PRINT USING Fmt_sa1;"DELIVERY PROFILE:   ";File$;"  PASS ";Ac_pass;"
;Acprint$;"  LOST ";Nm_psa_ac_lost
14210      IF Brgg=0 THEN GOTO 14300
14220    !IF Brgg=1 THEN GOTO 14270
14230    !IF Brgg=2 THEN GOTO 14250
14240      IF Destroy=1 THEN
14250       !PRINT "BRIDGE DESTROYED"!  THE SEED IS",R3
14260        Destroy=2
14270      ELSE
14280       !PRINT "BRIDGE NOT DESTROYED"!  THE SEED IS",R3
14290      END IF
14300      PRINTER IS 1
14310    UNTIL Ac_pass>=Nm_ac_passes OR Nm_ac_lost_flt>=Break_point OR Nm_ac_le
<.1
14320 UNTIL Ac_load>=Nm_wpn_loads OR Nm_ac_lost_flt>=Break_point OR Nm_ac_left
1
14330 Strk_apport:  !
14340 GOSUB Strike_aportion
14350   !
14360 IF Nm_ac_lost_flt>Break_point AND Ac_load<Nm_wpn_loads THEN
14370    PRINTER IS 702
14380    PRINT
14390    PRINT Frc_clr$;" ATTACK DISCONTINUED DUE TO EXCESSIVE ";Acprint$;" LOSSE
S "
14400    PRINTER IS 1
14410 END IF
14420   !
14430 Fmt_sa1:IMAGE 20A,9A,7A,3D,10X,1A,5A,7A,5D.D
14440 RETURN
14450!
14460!***************************************************************************
*
14470!
14480 Strike_data:!
14490!
14500 GOSUB Strike_init
14510 GOSUB Select_prmtrs
14520 GOSUB Select_units
14530 GOSUB Select_stk_info
14540   !
14550 File$=Aircraft$&"_"&Flt$
14560 ON ERROR GOSUB File_error_2
14570 ASSIGN @Path TO File$&Disk$
14580 OFF ERROR
14590 PRINTER IS 702
```

Table 5-3. Air attack/Air defense code.

```
14600 GOSUB Strike_output
14610 PRINTER IS 1
14620 ENTER @Path,1;Nm_wpn_loads,Wpn_ld_template$(*)
14630 ASSIGN @Path TO *
14640   !
14650 File$=Clr$&"_UN_AREA"
14660 ASSIGN @Path TO File$&Disk$
14670 ENTER @Path,1;Unit_type_area(*)
14680 ASSIGN @Path TO *
14690   !
14700 File$=Clr$&"_AC_TGWT"
14710 ASSIGN @Path TO File$&Disk$
14720 ENTER @Path,1;Ac_ms_tgt_wts(*)
14730 ASSIGN @Path TO *
14740   !
14750 RETURN
14760!
14770!************************************************************************
*
14780!
14790 Strike_info: !
14800!
14810 IF Wpn_load_code>=2 THEN
14820   Divisor=FNMax(Ac_pdtc_prn_tgt(Tgt_posture),.0000001)
14830   Nm_ac_passes=FNMin(Min_ac_passes/Divisor,1.3*Min_ac_passes)
14840   Nm_ac_passes=INT(Nm_ac_passes+.5)
14850     !
14860   IF Tgt_posture<=2 THEN
14870     Ac_tgt_pk(7)=Ac_tgt_pk(7)*Prsnl_fxhl_frc
14880     FOR X=36 TO 47
14890       Ac_tgt_pk(X)=Ac_tgt_pk(X)*Prsnl_fxhl_frc
14900     NEXT X
14910     Ac_tgt_pk(53)=Ac_tgt_pk(53)*Prsnl_fxhl_frc
14920     Ac_tgt_pk(54)=Ac_tgt_pk(54)*Prsnl_fxhl_frc
14930     IF Frc_clr$="BLUE" THEN
14940       Ac_tgt_pk(8)=Ac_tgt_pk(8)*Prsnl_fxhl_frc
14950       FOR X=28 TO 31
14960         Ac_tgt_pk(X)=Ac_tgt_pk(X)*Prsnl_fxhl_frc
14970       NEXT X
14980     END IF
14990   END IF
15000 ELSE
15010   Nm_ac_passes=Min_ac_passes
15020 END IF
15030   !
15040 RETURN
15050!
15060!************************************************************************
15070!
15080 Strike_init:!
15090!
15100 Ac_brkpt_frc=Ac_brkpt_pct/100
```

Table 5-3. Air attack/Air defense code.

```
15110 Tgt_frc_open=Tgt_pct_open/100
15120 Tgt_frc_woods=1-Tgt_frc_open
15130 Altitude_meters=500
15140 Nm_ac_left=Nm_ac_input
15150 Unit_type=Tgt_type+1
15160 Unit_posture=Tgt_posture
15170 FOR I=1 TO Nm_units
15180   Tgt_expsr_frc(I)=Tgt_expsr_pct(I)/100
15190 NEXT I
15200  !
15210 Tl_ada_tons=0
15220 Tl_ada_tons_frd=0
15230 Nm_ac_lost_flt=0
15240 Tl_suprsn_1=0
15250 Tl_suprsn_2=0
15260 FOR I=1 TO 72
15270   Nm_elmt_lost_fl(I)=0
15280   Nm_elmt_tgt(I)=0
15290 NEXT I
15300  !
15310 RETURN
15320!
15330!*************************************************************************
15340!
15350 Strike_input:!
15360!
15370 Destroy=0
15380 PRINT
15390 PRINT
15400 PRINT "    INPUT THE FOLLOWING INFORMATION FOR A ";Frc_clr$;" AIR STRIKE"
15410 PRINT
15420 PRINT USING Fmtsi;"TOTAL A/C AVAILABLE FOR COMBAT:",Nm_ac_avl_stk(*)
15430 PRINT
15440 PRINT "ENTER: # OF A/C, A/C TYPE, A/C MISSION, A/C BREAKPOINT, TGT % IN OP
EN"
15450 INPUT Nm_ac_input,Ac_type,Ac_mission,Ac_brkpt_pct,Tgt_pct_open
15460  !IF Ac_mission=1 THEN INPUT "WHAT IS THE SEED",R3
15470 CALL Check_var("A/C TYPE",Ac_type,1,4)
15480 CALL Check_var("AC MISSION",Ac_mission,1,5)
15490 CALL Check_var("A/C BREAKPOINT %",Ac_brkpt_pct,0,100)
15500 CALL Check_var("TGT % IN OPEN",Tgt_pct_open,0,100)
15510   !
15520 PRINT
15530 PRINT "ENTER: TYPE OF STRIKE -- CAS OR BAI  (INPUT 1 OR 2)"
15540 INPUT Cas_bai
15550 CALL Check_var("TYPE OF STRIKE (CAS OR BAI)",Cas_bai,1,2)
15560 PRINT
15570 PRINT "ENTER: EXPECTED DENSITY OF AD -- HIGH OR LOW  (INPUT 1 OR 2)"
15580 INPUT High_low
15590 CALL Check_var("EXPECTED DENSITY OF AD",High_low,1,2)
15600   !
15610 PRINT
```

Table 5-3. Air attack/Air defense code.

```
15620 PRINT "ENTER: TGT LENGTH, TGT WIDTH, TGT POSTURE, TGT TYPE, # OF TGT UNI
"
15630 INPUT Tgt_length,Tgt_width,Tgt_posture,Tgt_type,Nm_units
15640 CALL Check_var("TGT POSTURE",Tgt_posture,1,4)
15650 CALL Check_var("TGT TYPE",Tgt_type,0,9)
15660 CALL Unit_read(Frc_clr$,Nm_units,Unit_id(*),Terr(*))
15670   !
15680 IF Nm_units>0 THEN
15690   PRINT
15700   PRINT "ENTER: PERCENT OF UNIT IN TARGET AREA FOR EACH UNIT_ID"
15710   INPUT Tgt_expsr_pct(*)
15720   FOR I=1 TO Nm_units
15730     CALL Check_var("% TARGETED",Tgt_expsr_pct(I),0,100)
15740   NEXT I
15750 END IF
15760   !
15770 IF Ac_mission=1 AND Tgt_type=8 THEN INPUT "ENTER RANDOM NUMBER",R5
15780 FOR I=Nm_units+1 TO 5
15790   Tgt_expsr_pct(I)=0
15800 NEXT I
15810   !
15820 Fmtsi:IMAGE 31A,2X,4(3D.D,2X)
15830 RETURN
15840!
15850!**************************************************************************
*
15860!
15870 Strike_output:!
15880!
15890 GOSUB Select_prmtrs
15900   !
15910 PRINT USING "///"
15920 PRINT "THE FOLLOWING ARE THE INPUTS SELECTED FOR THE ":Frc_clr$:" AIR STR
KE:"
15930 PRINT
15940 PRINT " # OF A/C, A/C TYPE, A/C MISSION, A/C BREAKPOINT, TGT % IN OPEN"
15950 PRINT USING Fmtto1;Nm_ac_input,Acprint$,Flt$,Ac_brkpt_pct,Tgt_pct_open
15960   !
15970 PRINT
15980 SELECT Cas_bai
15990 CASE 1
16000   PRINT "Aircraft performing CAS."
16010 CASE 2
16020   PRINT "Aircraft performing BAI."
16030 END SELECT
16040 PRINT
16050 SELECT High_low
16060 CASE 1
16070   PRINT "In a perceived high-density AD environment."
16080 CASE 2
16090   PRINT "In a perceived low-density AD environment."
16100 END SELECT
```

Table 5-3.   Air attack/Air defense code.

```
16110   !
16120 PRINT
16130 PRINT "TGT LENGTH, TGT WIDTH, TGT ACTIVITY, TGT TYPE, # OF TGT UNITS"
16140 PRINT USING Fmtto2;Tgt_length,"m",Tgt_width,"m",Activity$,Tgt_type$,Nm_un
ts
16150 PRINT
16160 PRINT USING Fmtto3;"TARGET UNIT-ID'S CHOSEN:",Unit_id(*)
16170 PRINT USING Fmtto3;"                 TERRAIN:",Terr(*)
16180 PRINT
16190 PRINT USING Fmtto3;"PERCENT OF UNIT TARGETED:",Tgt_expsr_pct(*)
16200  !
16210 PRINT
16220 Fmtto1:IMAGE 7D.D,5X,5A, 7X,6A,13X,3D,12X,3D
16230 Fmtto2:IMAGE  9D,1A,10D,1A,7X,7A,2X,8A,14X,2D
16240 Fmtto3:IMAGE 25A,5(2X,3D)
16250  !
16260 RETURN
16270!
16280!******************************************************************
*
16290!
16300 Strike_profile: !
16310!
16320 File$=Wpn_ld_template$(Ac_load)
16330 ON ERROR GOSUB File_error_1
16340 ASSIGN @Path TO File$&Disk$
16350 OFF ERROR
16360 ENTER @Path,1;Stk_profile(*)
16370 ASSIGN @Path TO *
16380   !
16390 Wpn_load_code=Stk_profile(1)
16400 Ac_area_prn_tgt=Stk_profile(2)
16410 Min_ac_passes=Stk_profile(3)
16420 Nm_ac_rnds_eng=Stk_profile(4)
16430 Nm_ac_rnd_bs_ld=Stk_profile(5)
16440 Prn_tgt_wds_frc=Stk_profile(160)
16450 Prsnl_fxhl_frc=Stk_profile(161)
16460 FOR I=1 TO 4
16470   Ac_pdtc_prn_tgt(I)=Stk_profile(5+I)
16480   Ac_plos_prn_tgt(I)=Stk_profile(9+I)
16490 NEXT I
16500 FOR I=1 TO 7
16510   Min_ada_alt(I)=Stk_profile(161+I)
16520   Max_ada_alt(I)=Stk_profile(168+I)
16530   Ada_range(I)=Stk_profile(175+I)
16540   Ada_prtcptn(I)=Stk_profile(182+I)
16550   Ada_rnd_bs_ld(I)=Stk_profile(189+I)
16560   Ada_rnd_wt(I)=Stk_profile(196+I)
16570   Ada_rnd_frd_eng(I)=Stk_profile(203+I)
16580   Ada_psa_ac(I)=Stk_profile(210+I)
16590   Ada_psk_ac(I)=Stk_profile(217+I)
16600 NEXT I
```

Table 5-3. Air attack/Air defense code.

```
16610 FOR I=1 TO 72
16620   Ac_tgt_pk(I)=Stk_profile(13+I)
16630   Ac_tgt_area(I)=Stk_profile(86+I)
16640 NEXT I
16650   !
16660 RETURN
16670!
16680!*********************************************************************************
16690!
16700 Update_results:!
16710!
16720 REPEAT
16730   PRINT
16740   PRINT "UPDATE MENU:  SELECT OPTION"
16750   PRINT "     (1) UPDATE CURRENT RESULTS"
16760   PRINT "     (2) PURGE CURRENT RESULTS"
16770   PRINT "     (3) EXIT"
16780   INPUT Nm_option
16790 UNTIL Nm_option=1 OR Nm_option=2 OR Nm_option=3
16800   !
16810 IF Nm_option=1 THEN
16820   GOSUB Cumulate_totals
16830   GOSUB Update_storage
16840 END IF
16850 IF Nm_option<>3 THEN
16860   GOSUB Init_totals
16870 END IF
16880   !
16890 RETURN
16900!
16910!*********************************************************************************
*
16920!
16930 Update_storage:!
16940!
16950 ASSIGN @Path TO " UNITFILE:HP9134,701"
16960 Cnt=0
16970 REPEAT
16980   Cnt=Cnt+1
16990   FOR I=1 TO 150
17000     Unit_info(I)=Unitstore_20_15(Cnt,I)
17010   NEXT I
17020   OUTPUT @Path,Unit_id_20_150(Cnt);Unit_info(*)
17030 UNTIL Cnt>=Nm_units_played
17040 ASSIGN @Path TO *
17050 PRINT
17060 PRINT "UPDATE COMPLETED"
17070   !
17080 RETURN
17090!
17100!*********************************************************************************
*
```

Table 5-3. Air attack/Air defense code.

```
17110!
17120 END
17130!
17140!***********************************************************************
*
17150!
17160 SUB Check_var(Var_name$,Variable,Min_value,Max_value)
17170!
17180   WHILE Variable<Min_value OR Variable>Max_value
17190     BEEP
17200     PRINT
17210     PRINT "** ERROR:  ";Variable;" IS INVALID FOR ";Var_name$
17220     PRINT "   INPUT:  ";Min_value;" THROUGH ";Max_value;" ONLY"
17230     INPUT Variable
17240   END WHILE
17250  !
17260 SUBEND
17270!
17280!***********************************************************************
*
17290!
17300 SUB Unit_read(Frc_clr$,Nm_units,Unit_id(*),Terr(*))
17310!
17320   CALL Check_var("# UNITS",Nm_units,0,5)
17330   IF Frc_clr$="RED" THEN
17340     Min_value=1
17350     Max_value=191
17360   ELSE
17370     Min_value=192
17380     Max_value=400
17390   END IF
17400   PRINT
17410   IF Nm_units>0 THEN
17420     FOR I=1 TO Nm_units
17430       INPUT "ENTER: UNIT-ID  ,  TERRAIN",Unit_id(I),Terr(I)
17440       CALL Check_var("UNIT-ID",Unit_id(I),Min_value,Max_value)
17450       CALL Check_var("TERRAIN",Terr(I),1,4)
17460     NEXT I
17470   END IF
17480   !
17490   FOR I=Nm_units+1 TO 5
17500     Unit_id(I)=0
17510     Terr(I)=0
17520   NEXT I
17530   !
17540 SUBEND
17550!
17560!***********************************************************************
*
17570!
17580 DEF FNMin(A,B)
17590!
```

Table 5-3. Air attack/Air defense code.

```
17600    IF A<=B THEN
17610      C=A
17620    ELSE
17630      C=B
17640    END IF
17650    RETURN C
17660    '
17670 FNEND
17680'
17690'*******************************************************************************
17700'
17710 DEF FNMax(A,B)
17720'
17730    IF A>=B THEN
17740      C=A
17750    ELSE
17760      C=B
17770    END IF
17780    RETURN C
17790    !
17800 FNEND
17810!
17820'*******************************************************************************
```

CHAPTER 6

GROUND COMBAT

1. PURPOSE.

The purpose of the DIME ground combat program (P4) is to determine the number of systems destroyed and ammunition expended during combat between ground forces. The program also displays a battle chronology describing significant maneuver events and designates either the Blue or Red force as terminating the battle through disengagement of direct-fire forces.

2. GENERAL.

A. The survival and effectiveness of a lightly armored force is dependent on its mobility. The force must be able to locate the enemy, move quickly to contact, strike, and then break contact before the enemy is able to respond by tailoring his mission to bring effective firepower into play against the more agile force. In a general sense, this is true of any force attempting to use surprise as a tactical multiplier. However, one usually thinks of a heavily armored force with a simpler tactical objective of containing the enemy and thus developing sufficient firepower to engage and defeat him.

B. This problem faced the DIME development team: how does one structure a division combat model which will fairly represent the light force tactics which seek to avoid decisive engagement, while at the same time fairly representing the advantages and disadvantages of a Soviet force seeking decisive engagement? Low-resolution corps/division models have classically played attrition scenarios representing an attacking force assaulting a defender in a prepared defense. They have not represented various missions and resulting force postures available to both attacker and defender. Nor have they represented the time-phased transition of a force from one posture to another (i.e., from a hasty to a prepared defense).

C. The DIME combat program attempts to represent this change in posture as a surprised unit, attacked in a vulnerable posture, which moves to a hardened posture. This process is simulated using the DIME missions available to each unit. The missions are as follows:

Mission Number	Description
0	Meeting engagement
1	Indirect fire
2	Movement
3	Frontal attack
4	Envelopmental attack
5	Delay

Mission Number	Description
6	Hasty defense
7	Prepared defense
8	Reserve/rear area
9	Ambush

Associated with each mission is a set of templates. The templates are used to represent the lethality and vulnerability of the unit and contain the following information:

(1) The percent of unit available to fire.

(2) The percent of unit available as targets.

(3) The percent of targetable unit fully exposed.

(4) The percent of targetable unit in hull defilade.

As the battle progresses, these templates are modified by a time-phased increment representing the change in unit vulnerability and lethality. The template structure gives the low-resolution DIME program the flexibility to represent units in various missions/postures engaged by attacking units in various postures. Although the templates represent the natural tendency for a unit to harden its position under fire, it was also necessary to structure the ground combat program to provide the gamer with the flexibility to select conditions which he believes will favorably initialize and terminate the battle.

D. The ground combat program simulates three phases of battle: phase I, movement to contact; phase II, the direct fire battle; and phase III, the withdrawal from direct fire contact. The program requires the gamer to input a set of parameters which essentially represent his battle plan or scenario. The program then executes the plan of both Blue and Red gamers determining the resulting attrition and also determining which side is forced to initiate and sustain a more vulnerable withdrawal posture in phase III. Figures 6-1 through 6-3 show a stylized depiction of the DIME battle geometry in each of the three phases.

(1) *Phase I, Movement to Contact.* This is essentially an artillery battle. During this period, the forces are beyond the opening range for direct fire (3000 m) and closing toward their objective. The principal scenario parameters input by the Red and Blue gamers describing this phase of the battle are:

(a) Opening range for artillery (beginning of phase I).

(b) Percent of force forward in covering action.

(c) Opening range for direct fire (ending of phase I).

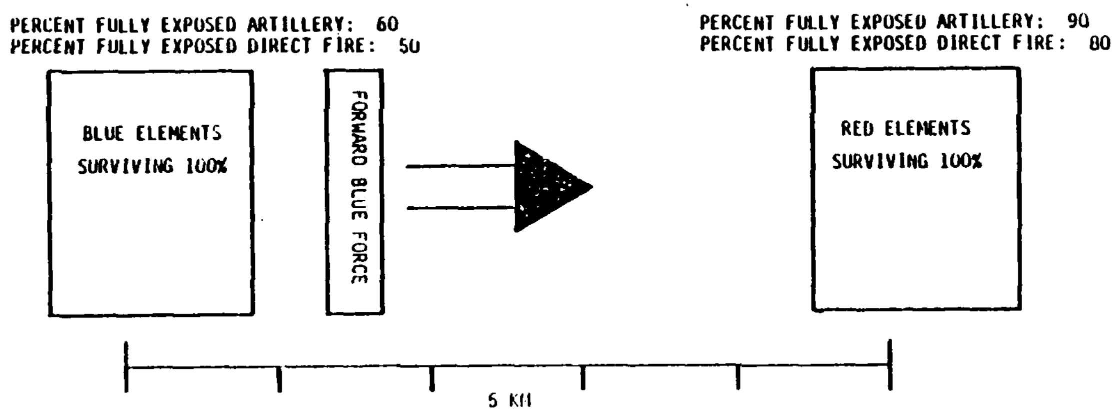
BLUE ATTACKER SCENARIO

MISSION: ASSAULT
OPEN RANGE ARTILLERY: 5 KM
OPEN RANGE DIRECT FIRE: 3 KM
PERCENT FORCES COVERING: 10
WITHDRAWAL RANGE: 0
PERCENT LOSS WITHDRAWAL: 40

RED DEFENDER SCENARIO

MISSION*: RESERVE
OPEN RANGE ARTILLERY: 4 KM
OPEN RANGE DIRECT FIRE: 3 KM
PERCENT FORCES COVERING: 0
WITHDRAWAL RANGE: 1 KM
PERCENT LOSS WITHDRAWAL: 60

BATTLE TIME: 8:30



*Red regiment in Reserve caught by flanking attack by Blue Battalion.

Figure 6-1. Phase I, Blue force movement to contact.

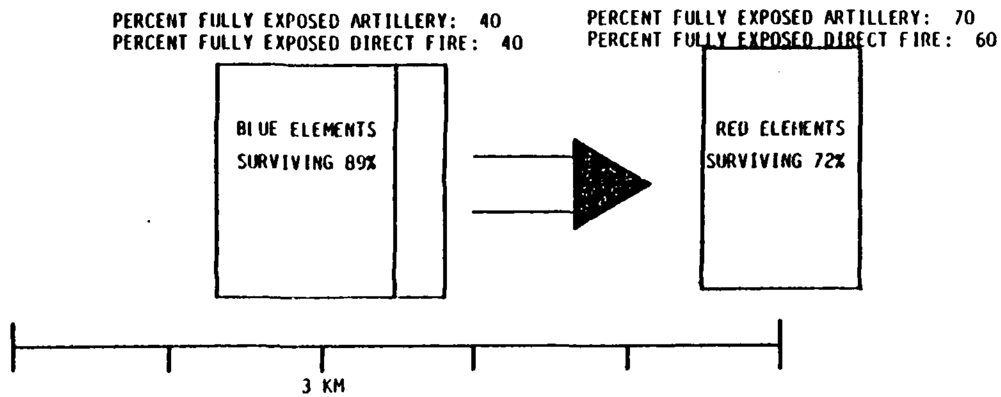
BLUE ATTACKER SCENARIO

MISSION: ASSAULT
OPEN RANGE ARTILLERY: 5 KM
OPEN RANGE DIRECT FIRE: 3 KM
PERCENT FORCES COVERING: 10
WITHDRAWAL RANGE: 0
PERCENT LOSS WITHDRAWAL: 40

RED DEFENDER SCENARIO

MISSION*: RESERVE
OPEN RANGE ARTILLERY: 4 KM
OPEN RANGE DIRECT FIRE: 3 KM
PERCENT FORCES COVERING: 0
WITHDRAWAL RANGE: 1 KM
PERCENT LOSS WITHDRAWAL: 60

BATTLE TIME: 9:30



*Although forces tend to harden, UIME does not allow Red to change mission throughout battle.

Figure 6-2. Phase II, direct fire battle.

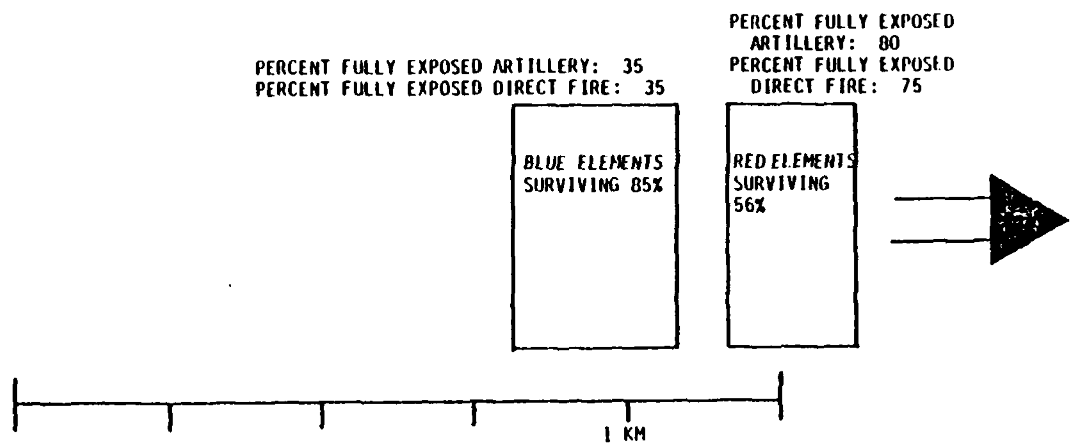
BLUE ATTACKER SCENARIO

MISSION: ASSAULT
OPEN RANGE ARTILLERY: 5 KM
OPEN RANGE DIRECT FIRE: 3 KM
PERCENT FORCES COVERING: 10
WITHDRAWAL RANGE: 0
PERCENT LOSS WITHDRAWAL: 40

RED DEFENDER SCENARIO

MISSION*: RESERVE
OPEN RANGE ARTILLERY: 4 KM
OPEN RANGE DIRECT FIRE: 3 KM
PERCENT FORCES COVERING: 0
WITHDRAWAL RANGE: 1 KM
PERCENT LOSS FORCING WITHDRAWAL: 60

BATTLE TIME: 10:30



*Although forced to withdraw, Red's overall mission remains constant.

Figure 6-3. Phase III, Red force withdrawal to new reserve position.

It should be noted Blue has placed 10 percent of his forces forward in a covering action in Figure 6-1.

(2) Phase II, The Direct Fire Battle. In this phase, direct fire systems of both forces engage available targets. In the example scenario (Figure 6-2), both Blue and Red have chosen 3 km as an opening range. Hence, all available firing systems which can perform at that range will begin to fire. Note that the DIME ground combat program has inflicted attrition on elements in both forces and moved the elements into less exposed positions to artillery and direct fire weapons. The principal gamer inputs affecting phase II are:

(a) Opening range for direct fire (beginning of phase II).

(b) Withdrawal range (ending of phase II).

(c) Percent of losses units will incur before withdrawing.

(d) Mission (determines percent of forces fully exposed to artillery/direct fire).

(3) Phase III, The Withdrawal Phase. In this phase (Figure 6-3) Blue and/or Red forces break contact with enemy direct fire weapons. The combat program also increases the vulnerability of the withdrawing force by increasing the percent of fully exposed targets. This increase is dependent on the mission of the withdrawing force. The gamer inputs which impact this phase are:

(a) Withdrawal range, if violated by either of the forces will cause the beginning of phase III.

(b) Withdrawal percent loss, if sustained by either force, will cause the beginning of phase III.

(c) Withdrawal time, time required to move out of direct fire range, determines the ending of phase III.

(d) Mission, determines the percent of forces fully exposed during withdrawal.

(4) Omission of battle phases. It should be noted that the DIME program does not require the gamer to use all of the battle phases. By carefully structuring the scenario input parameters, the gamer can delete any battle phase(s). For example, by making the "opening range for direct fire" equal to the "withdrawal range for direct fire," the DIME ground combat program will move directly from phase I to phase III, thereby simulating a standoff battle consisting primarily of artillery with one force withdrawing to avoid direct contact.

3. DATA FLOW.

The data flow structure for the ground combat program is shown in Figure 6-4. The data are divided into four categories: game parameters, unit status file ("UNITFILE") entries, weapon system parameters, and logistic parameters.

A. Game parameters. The game parameters are input through the terminal. These parameters describe both the tactical scenario and the environmental scenario.

(1) Tactical scenario. These inputs describe which units will fight and their initial missions and deployment postures. Tactical parameters also determine the criteria for shifting from one battle phase to the next and the planned allocation of available artillery and mortar tubes to the missions of prep/counterprep, close support, counterfire, suppression of air defense (SEAD), interdiction, and defensive smoke. DIME will attempt to honor the artillery allocations until it becomes obvious that the tactical situation demands a reallocation of resources. At this point, the program will automatically redistribute the indirect fire resources. The dimensions of the area occupied by both forces are also part of the tactical scenario. This area serves as a basis for computing target density in the artillery and mortar effectiveness algorithms and has significant impact on these algorithms. The tactical scenario represents the pace of battle desired by both Blue and Red gamers. The weapon system parameters, logistic parameters, and attrition algorithms ultimately drive the game in a favorable direction for one of the players.

(2) Environmental scenario. These parameters describe the battle terrain, weather, and the battle initiation time. The terrain type consists of one of four types:

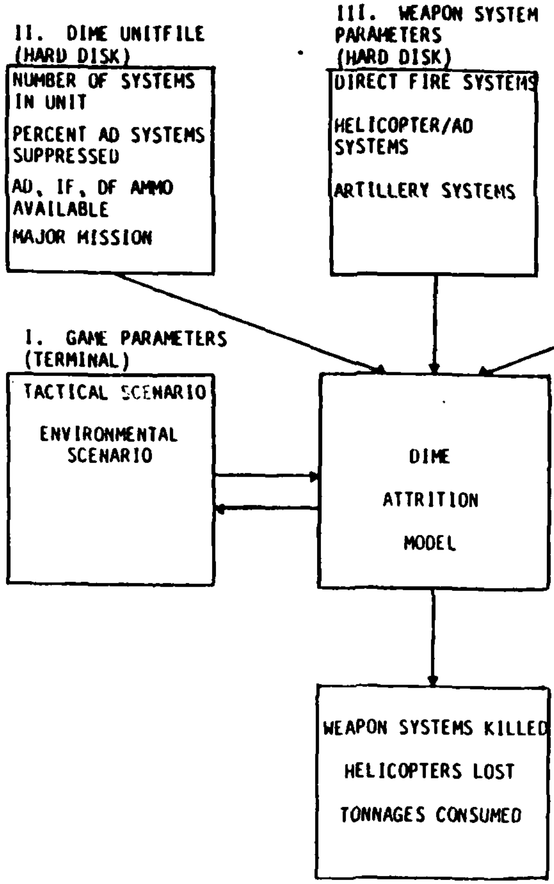
(a) Open.

(b) Rolling.

(c) Hilly.

(d) Mountainous.

Both the exposure rates of vehicles and personnel to direct fire, and the movement rates for vehicles and personnel, are also keyed on the terrain. The meteorological visibility affects the ranges at which direct fire systems can detect and engage ground targets. Cloud height has a similar degrading effect on laser-seeking artillery projectiles. Battle start time determines the ambient light (day or night) conditions under which the battle will be played with night conditions resulting in degraded movement rates and detection ranges.



I. Game Parameters

A. Tactical Scenario Parameters

1. Participating units
2. Specific units and percent of participating unit
3. Attacker/Defender
4. Initial indirect fire range (meters)
5. Direct fire range (meters)
6. Number of minefields
7. Mounted (0)/Dismounted (1)
8. Helicopter parameters
9. Artillery allocation parameters
10. Battle area

B. Environmental Scenario Parameters

1. Beginning Battle time (day/night)
2. Visibility (1- 5km, 2-5km, 3-2km, 4-1km)
3. Cloud height (meters)
4. Terrain (1 - open, 2 - rolling, 3 - hilly, 4 - mountainous)

II. DIME UNIT FILE

III. Weapon System Parameters

A. Mine Systems

1. Losses per element (breached)
2. Losses per element (bull)

B. Indirect Fire Systems

1. Delivery errors
2. Lethal areas
3. Smoke parameters
4. Fire delivery rates

C. Direct Fire Systems

1. Expected number of completed firings (ECF)
2. Probability of kill (P_K)
3. Fire distribution
4. Sensors available

D. Helicopter/AD Systems

1. Helo characteristics: basic load, mast/non-mast mounted, type of munition
2. Probability of detection by helo, AD, & DF elements
3. Probability of kill by helo, AD, & DF elements
4. Time exposed and masked per popup
5. Fire distribution
6. Number of missiles/guns fired per engagement

E. PGM Systems

1. Single shot kill probability
2. Cloud height degradation

F. Infantry/Small Arms

1. Losses to small arms (0-500m)

IV. Logistics

A. Logistic Parameters

1. Ammo weight
2. Basic load
3. Fuel capacity

B. Element weights

1. Direct fire battle target mask
2. Indirect fire battle target mask
3. Helicopter target preference
4. Direct fire target preference
5. Indirect fire target preference
6. PGM target preferences
7. Air defense target preferences

Figure 6-4. Information flow for the ground combat program.

B. DIME "UNITFILE". Following the entry of the tactical scenario, the ground combat program structures the forces for battle from the DIME "UNITFILE". Selecting the units indicated in the tactical scenario, the program retrieves the number of weapons from each unit record (located on the hard disk) and forms both the Blue and the Red forces. The tons of ammunition for air defense, direct fire, and indirect fire are also obtained from the "UNITFILE". The attrition program then moves through 30-minute battle time steps calculating element losses and ammunition consumption to both forces. When direct fire contact between the forces has been broken, thus completing the battle, the attrition program distributes the element and ammunition losses among each unit and updates their records on the DIME "UNITFILE."

C. Weapon system parameters. The weapon system parameter files contain the data describing the lethality and vulnerability characteristics of each weapon system being played in the program. The files are randomly accessed by the combat program as engagements occur during the battle. As noted in Figure 6-4, the files contain system information of six types:

(1) Mine systems. The ground combat mine module considers one type of minefield. The data representing the attrition effects against the 70 weapon systems are maintained in data statements as part of the mine module. The data describe kills per armored column and are based on the tactic (bull or breach) being used to cross the minefield. Refer to Chapter 6, Section I for more information.

(2) Indirect fire systems. The DIME ground combat artillery module uses algorithms from the Super Quickie II model to assess kills through artillery, mortars, or multiple launch rocket systems (MLRS for Blue)/multiple rocket launchers (MRL for Red). Data supporting these algorithms are found on the hard disk and consist of artillery delivery errors and lethal areas for each munition against the 70 target elements. Section II of this chapter describes the indirect fire module in greater detail.

(3) Smoke Parameters. Artillery-delivered white phosphorus smoke can be employed by a withdrawing unit during battle phase III. The smoke module calculates the percent of both forces screened by the smoke and then calculates the associated reductions in direct fire kills. Data describing the percent of targets visible to various direct fire sensor systems are found on the hard disk. The smoke file is accessed when the tactical scenario indicates that a retreating force wishes to deploy screening smoke. For more on smoke, see Section III of this chapter.

(4) Direct fire parameters. These files are located on the hard disk and accessed as targets close from 3000 meters. The records are structured to represent engagements in each 500-meter band and contain firing rates and probabilities of kill for each weapon engaging twenty target categories.

Other information describing individual weapon fire control systems (principal sensor, basic load) are also contained in these files. See Section IV of this chapter for greater detail of direct fire data flow and file structures.

(4) Helicopter/air defense systems. This file is also located on the hard disk and is accessed during helicopter engagements of ground targets. The ground combat program assumes that helicopters engaging forces will attempt to stand off at maximum range while still employing their weapons effectively. Consequently, the file contains records describing the helicopter's ability to detect, engage, and kill ground targets depending on the optimal range for various delivery (missile or gun) profiles. Also contained in the record is the ability of each air defense system to engage the helicopter as it stands off at its engagement range. See Section V of this chapter for a more explicit description on helicopter data flow and file structures.

(5) Precision-guided munitions. The ground combat program plays two types of precision-guided munitions (PGM). These consist of cannon-launched guided projectiles (CLGP) and guided antiarmor mortar projectiles (GAMP). Data necessary for these include single shot kill probabilities toward each of the 70 targets and designator degradation factors for each PGM. These data are held internally within the PGM attrition module. For greater detail, see Section VI of this chapter.

(6) Dismounted infantry personnel losses. If a final assault by dismounted infantry is part of the tactical scenario, the ground combat program will allow infantry to dismount during the last 500 meters of closure. Data describing losses to infantry personnel during this phase are maintained as data statements in the infantry module. See Section VII of this chapter for more information.

D. Logistic parameters. The ground combat program calculates the tonnages of ammunition consumed by both Blue and Red forces during the battle. The data base supporting these calculations consists of basic loads and the packaged round weights for each of the 70 weapon systems on the "UNITFILE". This data base is held in the ground combat program in the form of data statements.

E. DIME attrition output.

(1) The principal output from the combat program is the number of weapon systems lost in battle. Figure 6-5 shows an example of Blue system losses following a 2 1/2-hour battle. The losses are not provided in the classic weapon-to-weapon killer/victim sense; rather, the program provides losses inflicted by a functional group of weapons, i.e., direct fire (DF), indirect fire (IF), attack helicopters (A/H), infantry (INF), precision-guided munitions (PGMs), and mines (MIN). If it is necessary to obtain killer/victim tables by weapon type, this can be done for direct fire systems by applying the individual firer/target probabilities of kills developed by subroutine Df_cbt. However, the user is reminded that DIME is a low-resolution model designed primarily for analysis of brigade/battalion tactics and not for analysis of individual weapon systems.

(2) Other output from the program consists of a report produced every 30 minutes describing the location of the forces, the artillery and mortar volleys fired, and the current strengths of both forces. Figure 6-6 shows an example of the 30-minute battle updates produced by the program.

4. FILE STRUCTURE.

The DIME ground combat program (P4) accesses the following external files on the hard disk.

A. Sys_eff(I,J) contains numerical effectiveness values assigned to each of the 70 weapon elements. The effectiveness is determined by the element's firepower capability.

I is 1 = Blue
2 = Red
J is 1-70 for the weapon types.

B. Wpn_type(I,J) contains a number (1-3) which places the weapon elements into the ammunition categories of:

1 = Direct fire
2 = Indirect fire
3 = Air defense.

I and J represent the same as above.

RED KILLER--BLUE VICTIM TABLE

VICTIM SYS	START	D/F	I/F	PGM	A/H	INF	MIN	END
		<------	------	KILLER	------	------	------>	
M551	12.3	.3	.4	0.0	.1	0.0	0.0	11.5
FAV40	24.8	.4	.7	0.0	.6	0.0	0.0	23.1
HMV-G	2.2	0.0	.1	0.0	.1	0.0	0.0	2.0
DRAGN	5.3	0.0	.8	0.0	.1	0.0	0.0	4.3
CMD-V	20.4	.2	1.2	0.0	.5	0.0	0.0	18.4
HMV40	44.7	.5	2.2	0.0	1.1	0.0	0.0	40.9
ARTY	21.5	0.0	.8	0.0	0.0	0.0	0.0	20.7
MORTR	.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
INF	8.3	0.0	2.1	0.0	.2	0.0	0.0	6.0
VULCN	5.0	0.0	.6	0.0	.1	0.0	0.0	4.3
AVNGR	9.5	0.0	.2	0.0	0.0	0.0	0.0	9.3
STING	1.6	0.0	.2	0.0	0.0	0.0	0.0	1.3
F-TRK	5.7	0.0	.1	0.0	0.0	0.0	0.0	5.6
CGO-T	98.3	0.0	2.6	0.0	0.0	0.0	0.0	95.7

BLUE KILLER--RED VICTIM TABLE

VICTIM SYS	START	D/F	I/F	PGM	A/H	INF	MIN	END
		<------	------	KILLER	------	------	------>	
T55	9.2	.2	0.0	1.5	1.0	0.0	0.0	6.5
BMP73	2.5	0.0	0.0	.6	.3	0.0	0.0	1.6
BRDM3	6.3	.1	0.0	1.3	.5	0.0	0.0	4.3
AT-75	15.3	.6	.3	0.0	0.0	0.0	0.0	14.3
AGS17	10.3	.4	.1	0.0	0.0	0.0	0.0	9.7
CMD-V	17.7	.4	.3	0.0	.3	0.0	0.0	16.7
BTR	111.5	2.4	.9	4.3	1.8	0.0	0.0	102.1
ARTY	138.9	0.0	.3	0.0	0.0	0.0	0.0	138.6
MORTR	17.6	0.0	.5	0.0	0.0	0.0	0.0	17.1
MRL	15.3	0.0	.1	0.0	0.0	0.0	0.0	15.2
INF	180.6	3.7	2.2	7.0	2.7	0.0	0.0	165.0
ZSU-X	3.2	.1	0.0	0.0	.3	0.0	0.0	2.8
SA-13	3.9	0.0	0.0	0.0	0.0	0.0	0.0	3.9
SA-6	11.2	0.0	0.0	0.0	0.0	0.0	0.0	11.2
SA-14	37.0	1.4	1.2	0.0	0.0	0.0	0.0	34.4
F-TRK	132.9	0.0	.2	0.0	0.0	0.0	0.0	132.7
CGO-T	512.1	0.0	1.0	0.0	0.0	0.0	0.0	511.1
ENGR	26.8	.6	0.0	0.0	.4	0.0	0.0	25.8

ATTACK HELICOPTER RESULTS:

TYPE	#COMMITTED	#KILLED	#SORTIES
LCH	3.0	1.5	3.0
AH1S	5.0	.1	5.0
SCTS	0.0	0.0	0.0
HIND	12.0	3.6	10.4
HIP	0.0	0.0	0.0
SCTS	0.0	0.0	0.0

Figure 6-5. An example of system losses following a 2.5 hour battle.

```
-----------------------------------------------
: SIGNIFICANT BATTLE EVENTS FROM 1500 TO 1530 :
-----------------------------------------------
```

```
BLUE HELICOPTERS ATTACKING RED FORCE
    LCH     3.0
   AH1S     5.0
    SCT     0.0

RED HELICOPTERS ATTACKING BLUE FORCE
   HIND     6.0
    HIP     0.0
    SCT     0.0

RED  WITHIN BLUE MORT CS RANGE : REMAINING BLUE PREP AMMO AVAILABLE FOR CS

BLUE WITHIN RED  MORT CS RANGE : REMAINING RED  PREP AMMO AVAILABLE FOR CS

RED  WITHIN BLUE ARTY CS RANGE : REMAINING BLUE PREP AMMO AVAILABLE FOR CS

BLUE WITHIN RED  ARTY CS RANGE : REMAINING RED  PREP AMMO AVAILABLE FOR CS
```

BLUE ARTILLERY VOLLEYS FIRED

	P/CP	CS	SEAD	CF	INT	TOTAL	Tons CONSUMED	Tons AVAILABLE
ARTY	0	21	0	3	3	26	11.6	19.3
MORT	0	0	0	0	0	0	0.0	11.2

BLUE PGM ROUNDS:	CLGP	GAMP
FIRED	20.0	0.0
AVAILABLE	0.0	0.0

RED ARTILLERY VOLLEYS FIRED

	P/CP	CS	SEAD	CF	INT	TOTAL	Tons CONSUMED	Tons AVAILABLE
ARTY	0	116	49	97	97	360	175.4	391.6
MORT	0	27	0	0	0	27	4.1	59.7
MRL	0	0	0	4	4	9	38.5	213.6

```
CLOSE OF BATTLE UNDERWAY    1530 HRS    B/EFF: 1.00    R/EFF: 1.00
              INITIAL RANGE:    4000     CURRENT RANGE:  2500

BOTH FORCES IN CONTACT FOR 30 MINUTE PERIOD.
```

	BLUE HELO 1	BLUE HELO 2	BLUE HELO 3	RED HELO 1	RED HELO 2	RED HELO 3
S1 NDOFF RANGE:	3000	2500	0	3000	0	0
HELO MSN:	1	2	0	3	0	0
ATK_PROF:	5	7	0	7	0	0
# HELOS:	3.00	5.00	0.00	6.00	0.00	0.00

```
BLUE HELICOPTER MISSION  1 ABORTED DUE TO EXCESSIVE LOSSES.

NO MORE AIR MISSILES AVAILABLE FOR BLUE HELO  2 .  WILL RETURN TO BASE.

RED HELICOPTER MISSION  1 ABORTED DUE TO EXCESSIVE LOSSES.
```

Figure 6-6. An example of the 30 minute battle updates.

C. Ammo_wt(I,J) contains the ammunition expenditure per engagement for each of the 70 elements. I and J are the same as above.

D. Basic_ld(I,J) contains the total number of engagements for each of the 70 elements. I and J represent the same as above.

E. Arty_30min_wt(I,J) contains the total ammunition in tons that can be allocated to artillery during a 30-minute timestep.

I is: 1 = Blue
2 = Red

J is: 1 = Arty
2 = MLRS/MRL
3 = Mortar.

F. A_wt(I,J) contains the weight in tons for six batteries firing one burst of artillery ammunition. I and J are the same as in E above.

G. Bf_mask(J) and Rf_mask(J) contain a one, meaning element J may take part in battle, or a zero indicating that element J may not be involved in the battle. J represents the 1-70 system elements.

H. Bdf_mask(I,J) and Rdf_mask(I,J) contain a one for being a part of the direct fire battle, or a zero meaning element J is not involved in the battle.

I is: 1 = Conventional battle
2 = Attack on a command post
3 = Attack on a forward arming and refueling point (FARP)
4 = Attack on a log point
I is: 5 = Attack on a field artillery.

J is (1-70) for the weapon elements. Note: If Blue is attacking a log point and Red is defending, the Bdf_mask(1,J) and Rdf_mask(4,J) would be used.

I. Ds_start(I) contains the starting range in meters for close support in terrain I.

I is: 1 = Open
2 = Rolling
3 = Hilly
4 = Mountainous

J. Barty_30min(I,J) and Rarty_30min(I,J) contain tonnages per tube delivered per 30 minutes by supporting artillery. The descriptions for I and J are as follows:

I is the mission:
1 = Movement to contact
2 = Indirect fire
3 = Movement
4 = Frontal attack
5 = Envelop attack
6 = Delay
7 = Hasty defense
8 = Preparatory defense
9 = Rear area
10 = Ambush.

J is:
1 = Arty
2 = MLRS/MRL
3 = Mortar.

(Note: I = The mission value input, plus one, for array accessing.)

K. "UNITFILE". This file consists of 400 records, each containing 150 elements. The assignment of records consists of records 1-191 for Blue units and records 192-400 for Red units. Elements used within each record consist of 1-70, 78, 131 - 133, 80, 139 - 141. A description of these elements may be found in Chapter 1, Table 1-1.

L. Advance rate file. This file contains the maximum rate of advance for a force in meters advanced per minute. The file consists of 16 records.

Record 1. Blue, day, open terrain
Record 2. Blue, day, rolling terrain
Record 3. Blue, day, hilly terrain
Record 4. Blue, day, mountainous terrain
Record 5. Blue, night, open terrain
Record 6. Blue, night, rolling terrain
Record 7. Blue, night, hilly terrain
Record 8. Blue, night, mountainous terrain
Record 9. Red, day, open terrain
Record 10. Red, day, rolling terrain
Record 11. Red, day, hilly terrain
Record 12. Red, day, mountainous terrain
Record 13. Red, night, open terrain
Record 14. Red, night, rolling terrain
Record 15. Red, night, hilly terrain
Record 16. Red, night, mountainous terrain.

The appropriate record is read into the array Advance_rate(I,J), where:

I is battle phase:

1 = Mounted in Phase I
2 = Dismounted in Phase I
3 = Mounted in Phase III
4 = Dismounted in Phase III

and

J is mission:

1 = Movement to contact
2 = Indirect fire
3 = Movement
4 = Frontal attack
5 = Envelopment attack
6 = Delay
7 = Hasty defense
8 = Preparatory defense
9 = Reserve/rear area
10 = Ambush

M. Operational mission templates. These files contain the percentage of participants in a battle and target vulnerabilities. There are eight files for the Blue force and eight files for Red.

1. The files defining the number of target participants are:

(a) "BIFTARG" - Blue targets vs. indirect fire.
(b) "BDFTARG" - Blue targets vs. direct fire.
(c) "RIFTARG" - Red targets vs. indirect fire.
(d) "RDFTARG" - Red targets vs. direct fire.

2. Delta files contain the rate of change representing the percent of elements that can be introduced/extracted from battle per a 30-minute interval, while maintaining the same mission. These files are:

(a) "BIFDT" - Delta Blue target vs. indirect fire.
(b) "BDFDT" - Delta Blue target vs. direct fire.
(c) "RIFDT" - Delta Red target vs. indirect fire.
(d) "RDFDT" - Delta Red target vs. direct fire.

3. Firer files for direct fire contain the percent of participating firers and the corresponding delta files show the firer's rate of change.

(a) "BFIRE" - Blue firers for direct fire.
(b) "RFIRE" - Red firers for direct fire.
(c) "BDFIRE" - Delta Blue firers for direct fire.
(d) "RDFIRE" - Delta Red firers for direct fire.

4. Vulnerability files exist to represent the fraction of fully exposed systems to direct fire. They, too, have corresponding delta files which represent the percent increase/decrease to a system's vulnerability.

(a) "BVUL" - Blue vulnerabilities to direct fire.
(b) "RVUL" - Red vulnerabilities to direct fire.
(c) "BDVUL" - Delta Blue vulnerabilities.
(d) "RDVUL" - Delta Red vulnerabilities.

5. All of the template files contain 10 records each. These records represent the possible missions:

1 = Movement to contact
2 = Indirect fire
3 = Movement
4 = Frontal attack
5 = Envelop attack
6 = Delay
7 = Hasty defense
8 = Preparatory defense
9 = Rear area
10 = Ambush

The array used in accessing data from a record is in the form:

M(J)

where J is unit type and unit echelon and where echelon = 0 (Blue battalion/Red regiment) or 1 (Blue company/Red battalion).

11/01 = Combat unit
12/02 = Artillery unit
13/03 = ADA unit
14/04 = Helicopter site/FARP
15/05 = Command post/headquarters (CP/HQ)
16/06 = Engineer unit
17/07 = Fuel/ammunition (POL/AMMO) supply point
18/08 = Maintenance point
19/09 = Surface-to-air missile (SAM) site
20/10 = Communication/radar/electronic warfare (EW) site

N. For more information on any data, refer to Volume III of the DIME documentation.

5. ALGORITHMS.

A. Logic flow. Figure 6-7 presents a generalized logic flow for the processes occurring in the DIME ground combat program. The purpose of this diagram is to provide a framework for discussion of the algorithms supporting each of these processes. The DIME ground combat program is a timestep, deterministic model using expected value techniques for calculating losses to both forces.

B. Force structuring/tactical template initialization. The ground combat program begins processing by reading both tactical and environmental scenarios and structuring the opposing forces. This initialization is done in subprogram Set_battle (see Figure 6-8 for a generalized logic flow of Set_battle and Table 6-5 for a description of its principal variables).

The initialization process continues by accumulating the total tonnages of indirect fire (IF), direct fire (DF), and air defense (AD) ammunition available to both forces. After accepting a gamer description of the environmental scenario, the program then establishes the exposure/lethality templates for each force using the following equations:

(1) Exposure to indirect fire. Initial percentages exposed to indirect fire are:

$$Ift_{jm} = \sum_{i=1}^{n} P_{ij} * If_{im} \qquad \text{(Eq. 6-1)}$$

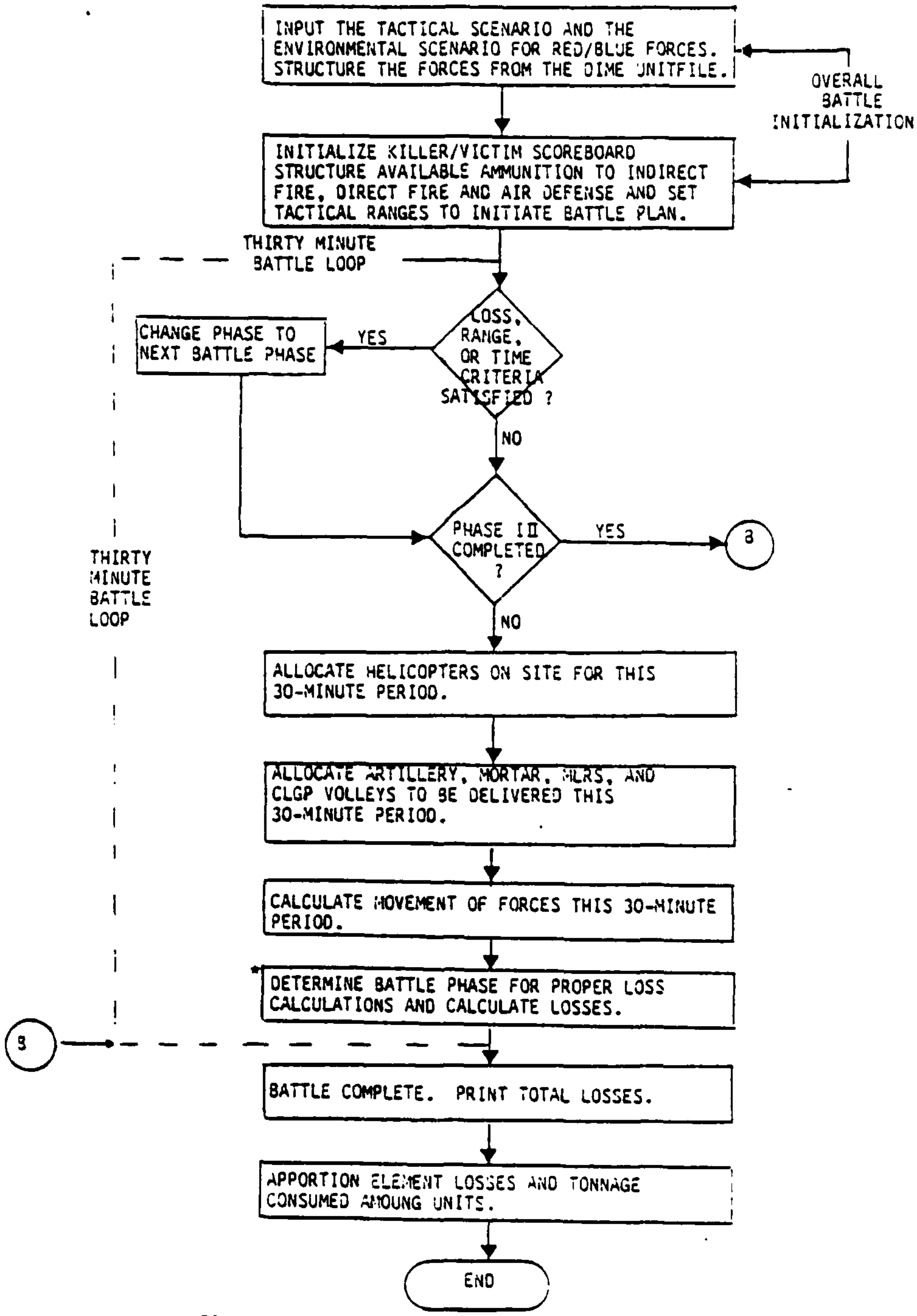
where:

Ift_{jm} = the percent of the j^{th} element type exposed to indirect fire at the beginning of the battle with the force in mission m (j = 1 to 70 element types in the force).

n = number of units in the entire force involved in the sector battle.

P_{ij} = percent of force from unit i and element type j.

If_{im} = percent of elements exposed to indirect fire for unit i in mission m. Note that unit i is of a specific unit type such as combat, artillery, ADA, etc.



*Block is given in greater detail on the following page.

Figure 6-7. DIME ground combat generalized logic flow.

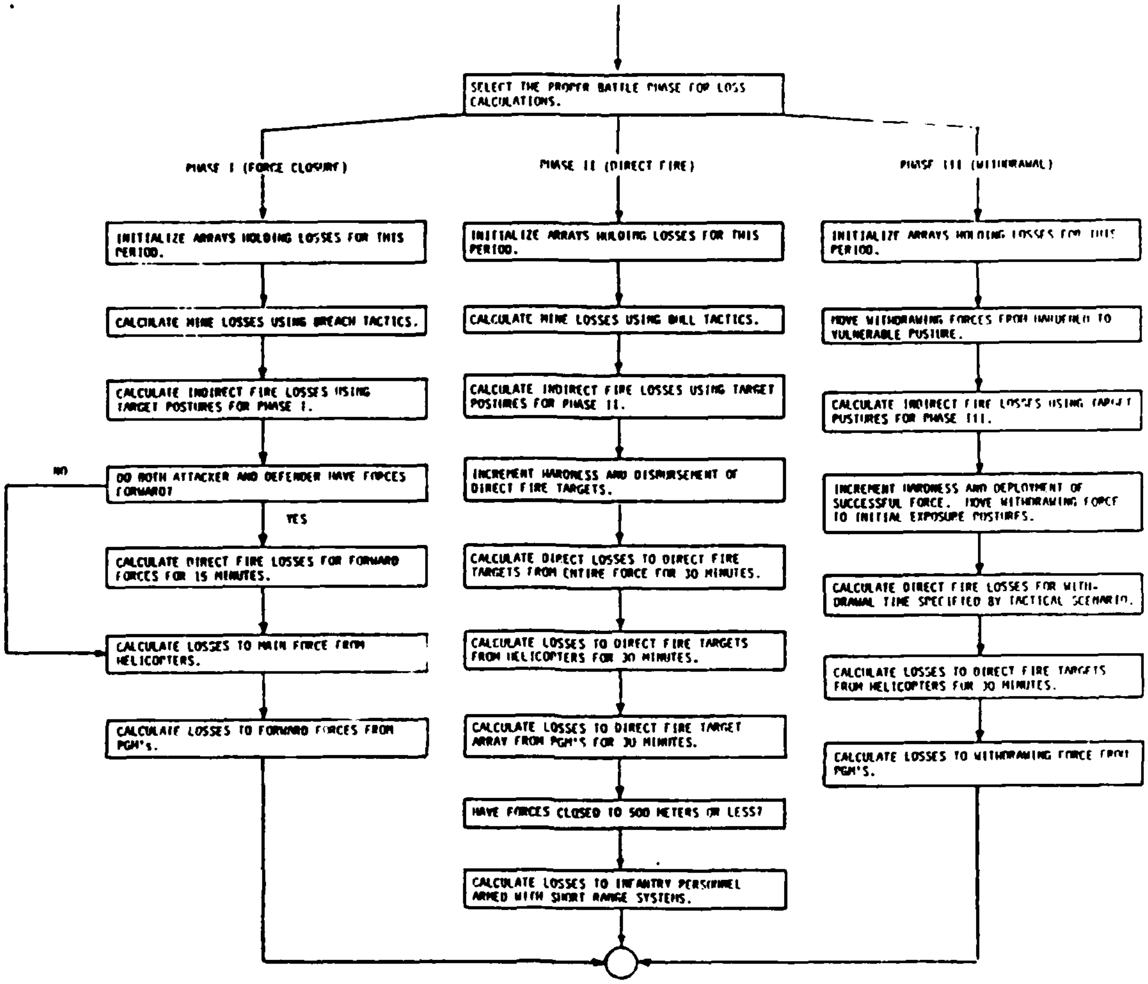


Figure 6-7. DIME ground combat generalized logic flow (concluded).

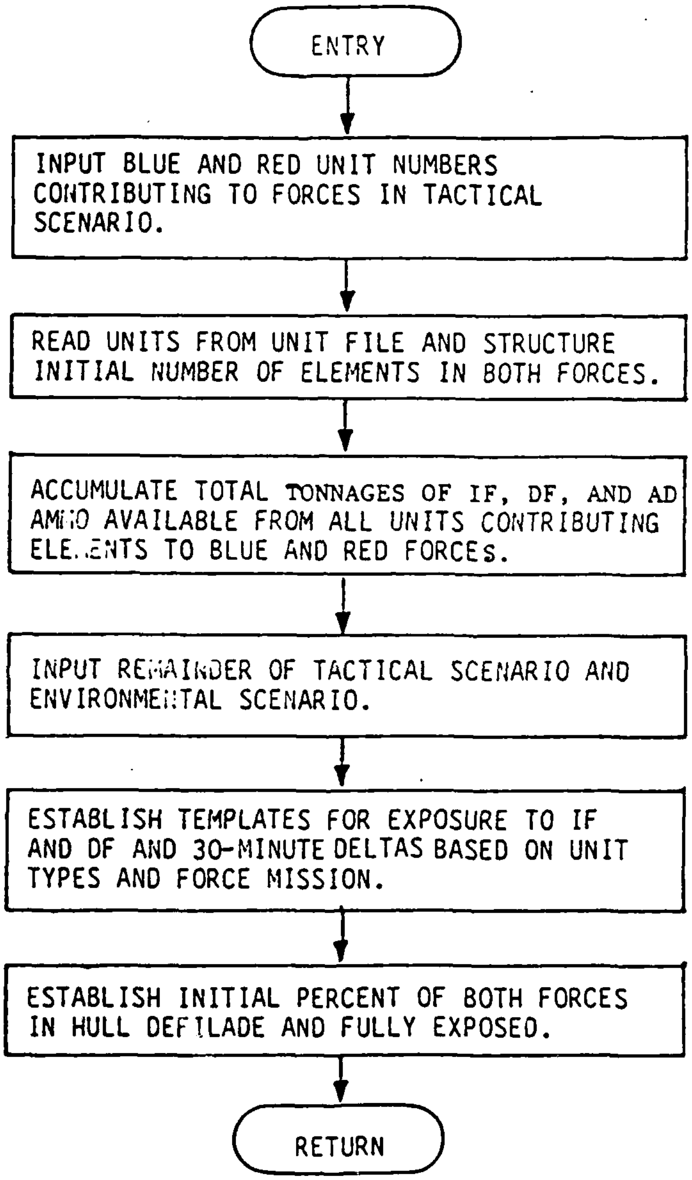


Figure 6-8. Generalized logic flow for initialization of force structure in subroutine Set_battle.

The delta changes which occur to the percentage each 30 minutes is also calculated.

$$Dift_{jm} = \sum_{i=1}^{n} P_{ij} * Dif_{im} \qquad \text{(Eq. 6-2)}$$

where:

$Dift_{jm}$ = the change in the percent of elements of type j exposed to indirect fire every 30 minutes with force in mission m. This increment is applied to Ift_{jm} each 30 minutes by DIME.

Dif_{im} = the change in the percent of elements of unit i exposing themselves to indirect fire each 30 minutes when the unit is operating under mission m. Note that unit i is of a specific unit type such as combat, artillery, ADA, etc.

(2) Exposure/vulnerability to direct fire. The following variables describing exposure and vulnerability to direct fire are also initialized in subroutine Set_battle using weighing procedures identical to those shown in equations 6-1 and 6-2.

Dft_{jm} = the percent of elements of type j in the force initially exposed to direct fire with force in mission m.

$Ddft_{jm}$ = the change in the percent of elements of type j exposed to direct fire each 30-minute period with force in mission m.

Dff_{jm} = the percent of elements of type j in the force which can initially fire in a direct fire battle with force in mission m.

$Ddff_{jm}$ = the change in the percent of elements of type j able to fire each 30-minute period with force in mission m.

Fet_{jm} = the percent of target and firer elements of type j which are fully exposed with force in mission m.

Dfetjm = the percent of targets and firers of type j which move from fully exposed to hull defilade each 30 minutes with the force in mission m.

It should be noted that the values If_{im} and Dif_{im} are taken from the DIME vulnerability/lethality templates discussed in paragraphs 2C and 4M(1 - 5) of this chapter. One of the primary functions of Set_battle is to initialize the vulnerability templates based on the unit echelon, type and mission for use by the attrition program. An example of the values used by DIME for If_{im} and Dif_{im} are shown in Table 6-1. For a complete listing of all values used in DIME, see C apter 4, Volume III, of the data documentation. The option exists to change the template values based on the level of deployment the gamer wants to portray.

C. Initialization of battle plan.

(1) Following the structuring of both forces, the combat program continues preparation for the battle by allocating indirect fire resources among eight tasks. The allocation occurs in subroutine Set_conditions. See Figure 6-9 for a logic flow of Set_conditions and Table 6-5 for a list of principal variables. The combat program allows allocation of indirect fire resources among eight tasks which are shown in Table 6-2 with possible allocations for artillery, mortars, and MLRS/MRL. The actual allocations made by DIME are dependent on gamer inputs from the tactical scenario. These inputs consist of the tactical fire plan describing the percent of available indirect firing systems which are to be applied to each task. Subroutine Set_conditions responds to this plan by earmarking the indirect fire ammunition for use in each task using the percentages input in the tactical fire plan. The earmarked tonnages serve as an upper bound to the ground combat program as it attempts to execute the fire plan during each 30-minute step in the battle. As an example, consider a unit having 50 tons of artillery ammunition to fire during a battle. Note that a 20 percent allocation for prep/counterprep will cause no more than 10 tons to be fired in this activity. This will be fired only if sufficient tubes survive in the 30-minute battle steps to fire it.

(2) Subroutine Set_conditions completes the development of the indirect fire plan by establishing tactical range lines. These lines represent the locations at which the indirect fire tasks will be initiated or cancelled by the program. The execution of the indirect fire tasks (i.e., the shift from prep/counterprep to close support) is governed by these phase lines and the battle range between the forces. Although the actual lines are established using inputs from the tactical scenario, the limits on these ranges for each task and the rules causing the program to reallocate resources from one task to another is shown in Table 6-3. The tactical scenario thresholds bounding the DIME ground combat battle phases are also initialized in Set_conditions.

D. Battle timestep loop. Following the initial allocations of maximum battle tonnages for indirect fire, the program moves into a timestep loop representing 30 minutes of battle time. The program will move through this loop until the tactical scenario criteria for phases I, II, and III have been satisfied. The 30-minute step is begun by comparing loss and range criteria for both forces with the tactical scenario threshold. If it is necessary to change phases, the program increments the phase counter and allocates the helicopters and indirect fire rounds for both forces for this 30-minute period. Principal variables affecting the process are listed in Table 6-5 under battle phase criteria.

Table 6-1. Blue target indirect fire templates (If_{im}) and indirect fire delta templates (Dif_{im}).

Unit type	Mission 0	1	2	3	4	5	6	7	8	9
Combat	.400	.160	.333	.600	.280	.200	.600	.800	.750	.050
Artillery	1.000	.160	.350	1.000	1.000	.400	1.000	1.000	1.000	1.000
Air defense	.400	.160	.333	.600	.280	.200	.600	.800	.750	.050
FARP	1.000	1.000	.500	1.000	1.000	1.000	1.000	1.000	.750	1.000
CP/HQ	1.000	1.000	.500	1.000	1.000	1.000	1.000	1.000	.750	1.000
Engineer	.133	.053	.111	.200	.093	.067	.200	.267	.250	.050
Supply	1.000	1.000	.500	1.000	1.000	1.000	1.000	1.000	.750	1.000
Maintenance	1.000	1.000	.500	1.000	1.000	1.000	1.000	1.000	.750	1.000
SAM site	1.000	1.000	.500	1.000	1.000	1.000	1.000	1.000	.750	1.000
Commo/radar	1.000	1.000	.500	1.000	1.000	1.000	1.000	1.000	.750	1.000

Unit type	Mission 0	1	2	3	4	5	6	7	8	9
Combat	.100	.140	.111	.067	.120	.133	.067	.033	.042	.010
Artillery	0.000	.140	.108	0.000	0.000	.100	0.000	0.000	0.000	0.000
Air defense	.100	.140	.111	.067	.120	.133	.067	.033	.042	.158
FARP	0.000	0.000	.083	0.000	0.000	0.000	0.000	0.000	.042	0.000
CP/HQ	0.000	0.000	.083	0.000	0.000	0.000	0.000	0.000	.042	0.000
Engineer	.144	.158	.148	.133	.151	.156	.133	.122	.125	.158
Supply	0.000	0.000	.083	0.000	0.000	0.000	0.000	0.000	.042	0.000
Maintenance	0.000	0.000	.083	0.000	0.000	0.000	0.000	0.000	.042	0.000
SAM site	0.000	0.000	.083	0.000	0.000	0.000	0.000	0.000	.042	0.000
Commo/radar	0.000	0.000	.083	0.000	0.000	0.000	0.000	0.000	.042	0.000

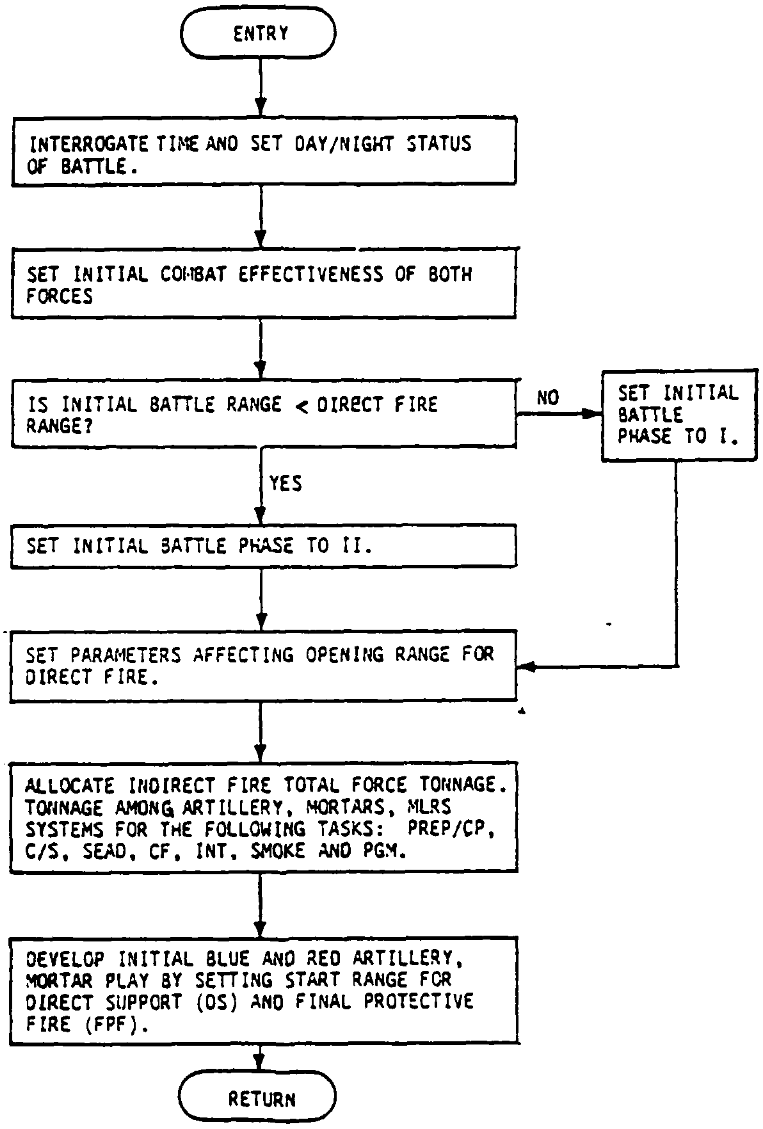


Figure 6-9. Generalized logic flow for initialization of battle parameters and 6-hour artillery ammunition allocation in subroutine Set_conditions.

Table 6-2. Possible allocations for blue and red indirect fire (IF) volleys.

	TASKS*							
Blue IF	Prep/CP	C/S	SEAD	CF	Int	Smoke	PGM	FPF
Artillery	X	X	X	X	X	X	X	X
MLRS	X	0	X	X	X	0	0	0
Mortar	X	X	X	0	0	X	X	X
Red IF								
Artillery	X	X	X	X	X	X	0	X
MRL	X	0	X	X	X	0	0	0
Mortar	X	X	X	0	0	X	0	X

X = possibility of allocation.

0 = cannot allocate in this task.

*TASKS	Description
Prep/CP	Preparatory/counterpreparatory
C/S	Close Support
SEAD	Suppression and engagement of air defense
CF	Counterfire
Int	Interdiction
Smoke	White phosphorous defensive smoke
PGM	Precision-guided munitions
FPF	Final protective fire

Table 6-3. Firing range conditions for various tasks in DIME indirect fire allocation module.

Task	Blue Artillery Begin	Blue Artillery End	Blue MLRS Begin	Blue MLRS End	Blue Mortar Begin	Blue Mortar End
Prep/CP	Blue prep time* and 12.0 km	Blue CS begins	Blue prep time* and 25.0 km	Blue CS begins	Blue prep time and 5.7 km	Blue CS begins
CS	5.0 km	Blue break less 100m	Does not fire		5.0 km**	Blue break
SEAD	Allocated at all ranges		Allocated at all ranges		Allocated at all ranges	
CF	12.0 km*	Blue FPF begins	25.0 km*	Blue break	Does not fire	
INT	12.0 km*	Blue FPF begins	25.0 km*	Blue break	Does not fire	
CLGP	Blue CS begins less 500m	Blue break	Does not fire		Does not fire	
GAMP	Does not fire		Does not fire		Blue CS begins less 500m	Blue break
FPF	Blue break plus 1.5 km	Blue break	Does not fire		Blue break plus 1.5 km	Blue break
SMOKE	Blue break and range < 3.2 km	Blue break	Does not fire		Blue break and range < 3.2 km	Blue break

* This is max range; gamer specification may allow to begin at closer ranges. However, battle time must have passed the attackers opening prep time.

** Terrain dependent: open - 6.0 km; rolling - 5.0 km; hilly - 4.00 km; mountainous - 4.0 km.

Table 6-3. Firing range conditions for various tasks in DIME indirect fire allocation module (concluded).

Task	Red Artillery		Red MRL		Red Mortar	
	Begin	End	Begin	End	Begin	End
Prep/CP	Red prep time* and 14.0 km	Red CS begins	Red prep time* and 14.0 km	Red CS begins	Red prep time* and 14.0 km	Red CS begins
CS	5.0 km	Red break less 100m	Does not fire		5.0 km**	Red break
SEAD	Allocated at all ranges		Allocated at all ranges		Allocated at all ranges	
CF	14.0 km*	Red FPF begins	25.0 km	Red break	Does not fire	
INT	14.0 km*	Red FPF begins	25.0 km	Red break	Does not fire	
FPF	Red break plus 1.5 km	Red break	Does not fire		Red break plus 1.5 km	Red break
SMOKE	Red break and range < 3.2 km	Red break	Does not fire		Red break and range < 3.2 km	Red break

* This is max range; gamer specification may allow to begin at closer ranges. However, battle time must have passed the attackers opening prep time.

** Terrain dependent: open - 6.0 km; rolling - 5.0 km; hilly - 4.00 km; mountainous - 4.0 km.

E. Allocation of helicopters (Helo arrive). This ground combat subroutine allocates the number of helicopters to be flown against the opposing force for any 30-minute period using a cell methodology. Figure 6-10 provides a logic flow of this methodology contained in subroutine Helo_arrive.

(1) Helicopter battle entry. The subroutine checks three criteria to determine if Blue helicopters can be scheduled for on battle site action.

(a) Are current force ranges less than Blue helicopter tactical entry range?

(b) Is current time less than Blue helicopter tactical entry time?

(c) Is meterological visibility greater than or equal to 2 km?

The meteorological criteria must always be satisfied. Helicopters will not fly on less than 2-km/days. Satisfying (c), if either (a) or (b) is satisfied, then the subroutine proceeds to calculate the number of helicopters available at the battle site. A similar check is made for Red helicopters.

(2) Helicopters available on site. The subroutine divides all surviving helicopters into battle cells. The number of battle cells are specified by the gamer and may vary from one to four. The cells allow the gamer to apply constant pressure on the opposing ground forces by using four cells or a 30-minute period of heavy helicopter pressure followed by 90 minutes of no helicopters on site using one cell. Two or three cells may also be selected. The cells are also used to simulate the periodic vacating of the battle site for refuel and rearmament by helicopters. The subroutine considers a cell to be on site for 30 minutes and off site for 90 minutes to refuel and rearm. Consequently, the subroutine schedules the presence of a cell based on Table 6-4. This scheduling table shows the number of cells at the battle site over five consecutive 30-minute periods for various cell configurations. A force of 12 helicopters is used in this table. For example, with three cells, the helicopters are divided into groups of four per cell. Cell one arrives on site at the time designated by the gamer with four helicopters. After 30 minutes, it returns to refuel and rearm and is followed by cell two with four new helicopters. Cell three relieves cell two and then, as it moves off site, there are no helicopters in the battle from 91 to 120 minutes past helicopter entry time until cell one returns at 121 minutes past helicopter entry time.

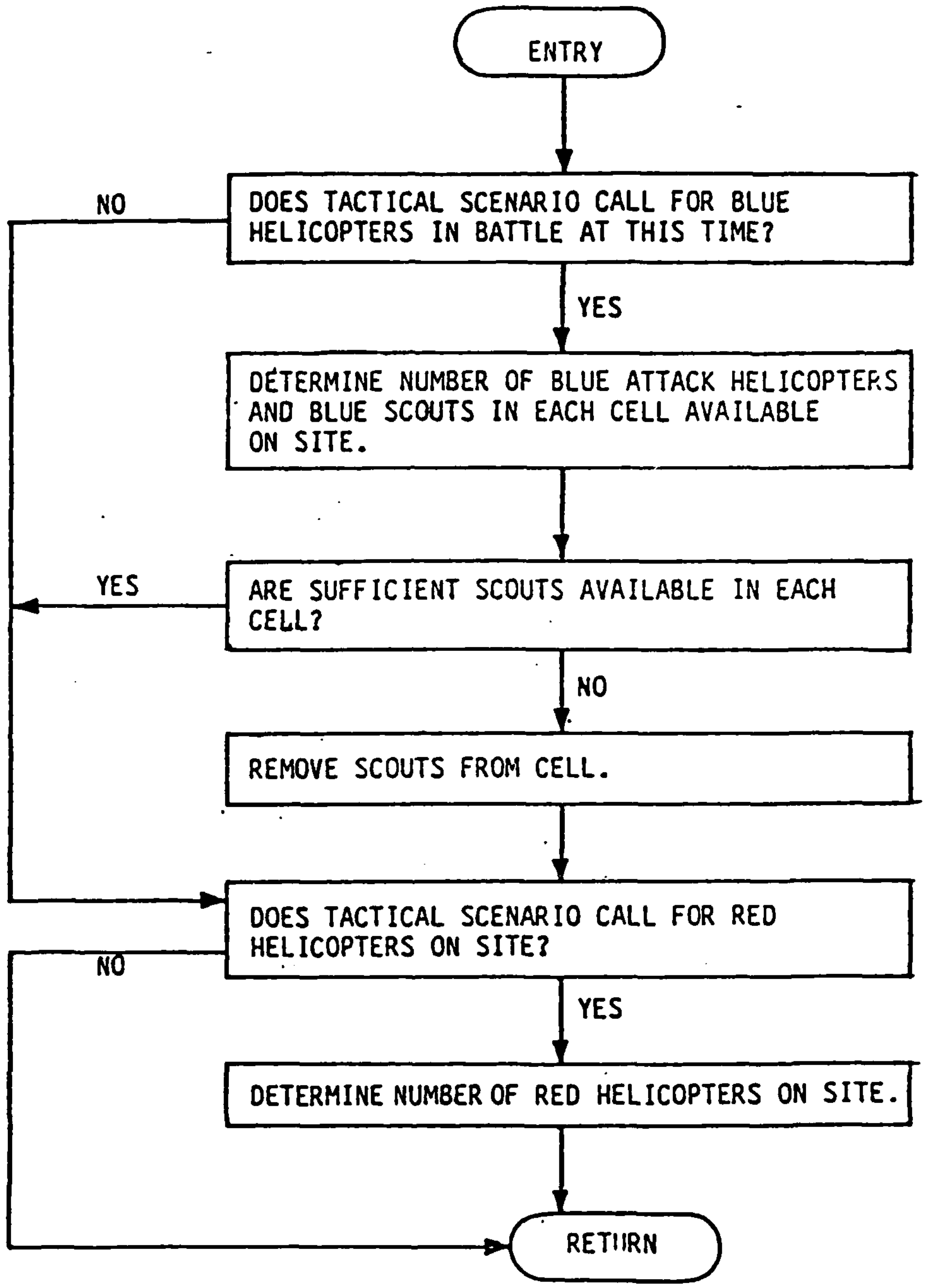


Figure 6-10. DIME logic flow for helicopter allocation found in subprogram Helo_arrive.

Table 6-4. On-site scheduling table for various numbers of helicopter cells.

Cells desired	Minutes following helicopter battle entry time 0-30	31-60	61-90	91-120	121-150
1	1/12	0/0	0/0	0/0	1/12
2	1/6	0/0	2/6	0/0	1/6
3	1/4	2/4	3/4	0/0	1/4
4	1/3	2/3	3/3	4/3	1/3

Table contains X/Y where:

X = the helicopter cell on side (example: number 1 of 3, number 2 of 3, etc.).

Y = the number of helicopters on site assuming 12 helicopters were allocated for this sector battle.

(3) Scouts available for Blue attack helicopters (AHs). If scouts are available, they are also divided among the number of cells. The actual number of scouts accompanying a Blue AH cell is given by:

$$Sc = \begin{cases} \text{MIN}(0.6 * Ah,\ S * Ro/C); \text{ if } S > 0.4 * Ah \\ 0; \text{ otherwise} \end{cases} \qquad \text{(Eq. 6-3)}$$

where:

Sc = number of scouts per cell.
Ah = number of attack helicopters per cell.
Ro = an operational reliability factor for scouts (currently set to 0.83).
C = number of cells desired by the gamer.
S = total number of scouts in the force.

Note that this essentially models the allocation of scouts along a criteria which will allow no more than three scouts to five AHs (a 3/5 mix) while assuring that the scout mix will be at least 2/5. If the 2/5 mix cannot be achieved, the subroutine discards the scouts (see Figure 6-8) and the Blue attack helicopters proceed alone.

(4) Following allocation of Blue helicopters, the subroutine then allocates Red helicopters using the cell methodology described for Blue. Note that the subroutine does not allocate scouts for the Red helicopter cells. Principal variables for the helicopter allocation process are described under Helo_arrive in Table 6-5.

F. Allocation of indirect fire volleys for 30 minutes.

(1) The number of volleys of artillery, mortars, and MLRS/MLR for both Blue and Red to be fired during the 30-minute period is calculated in subroutine Arty_arrive. The logic flow diagram for Arty_arrive is in Figure 6-11 with the subroutine's principal variables found in Table 6-5. The flow diagram shows the subroutine moving through four areas of allocation each 30 minutes.

(a) Allocation of smoke.

(b) Allocation of prep/counter prep, CLGP, and GAMP.

(c) Allocation of close support volleys.

(d) Allocation of counterfire and interdictive volleys.

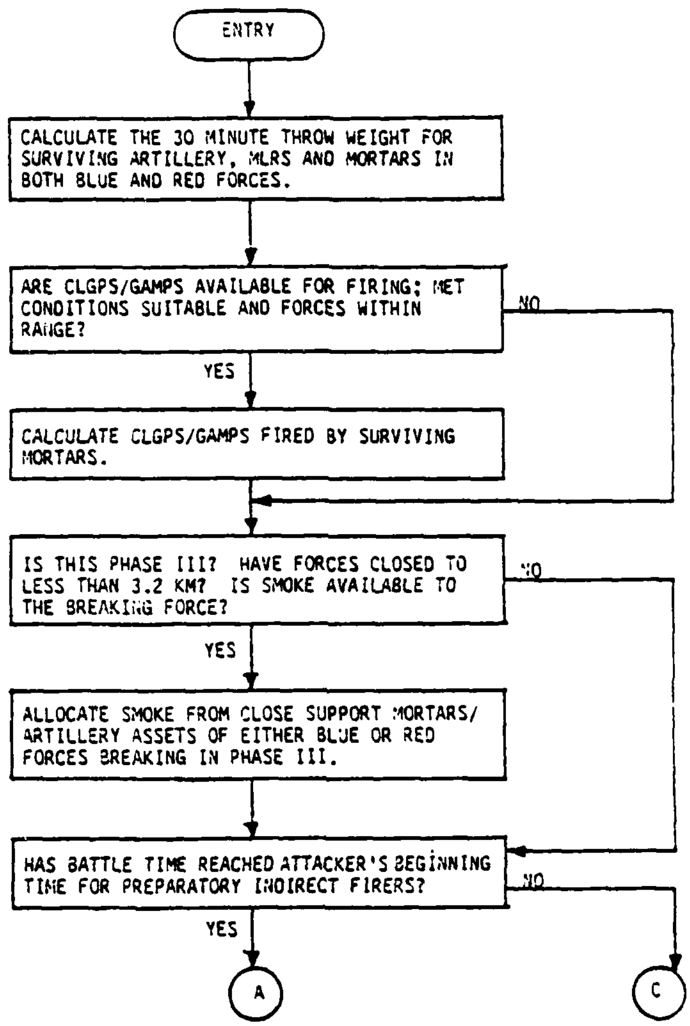


Figure 6-11. Logic flow for 30-minute allocation of artillery, MLRS/MRL and mortar volleys in subroutine Arty_arrive.

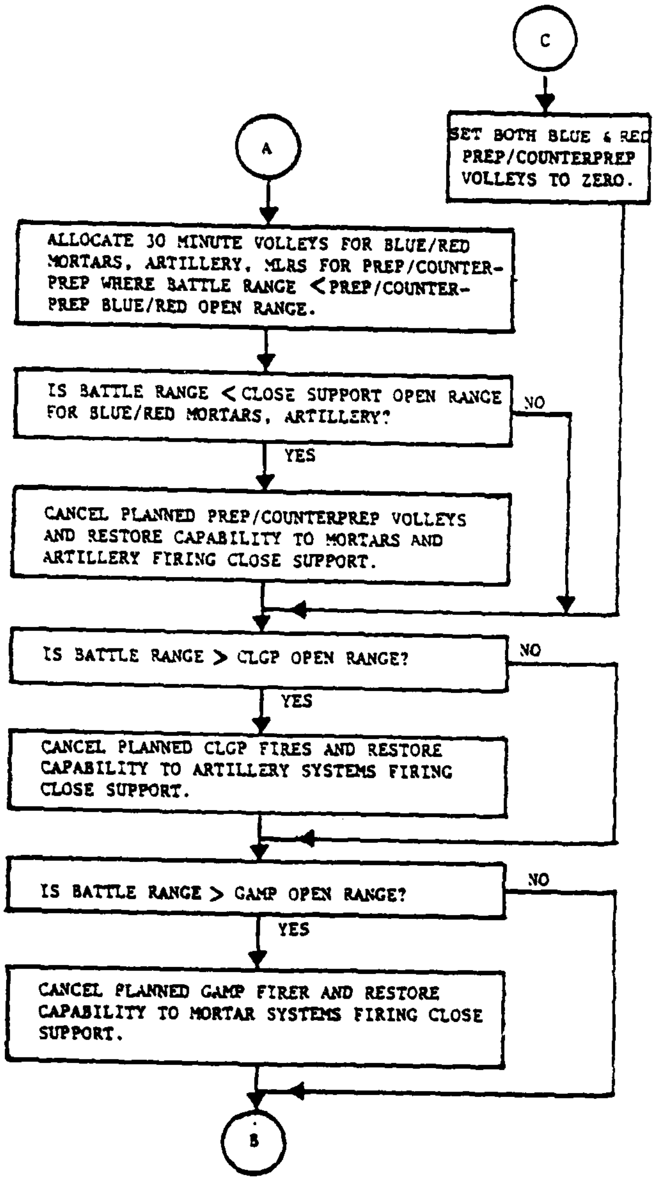


Figure 6-11. Logic flow for 30-minute allocation of artillery, MLRS/MRL and mortar volleys in subroutine Arty_arrive (continued).

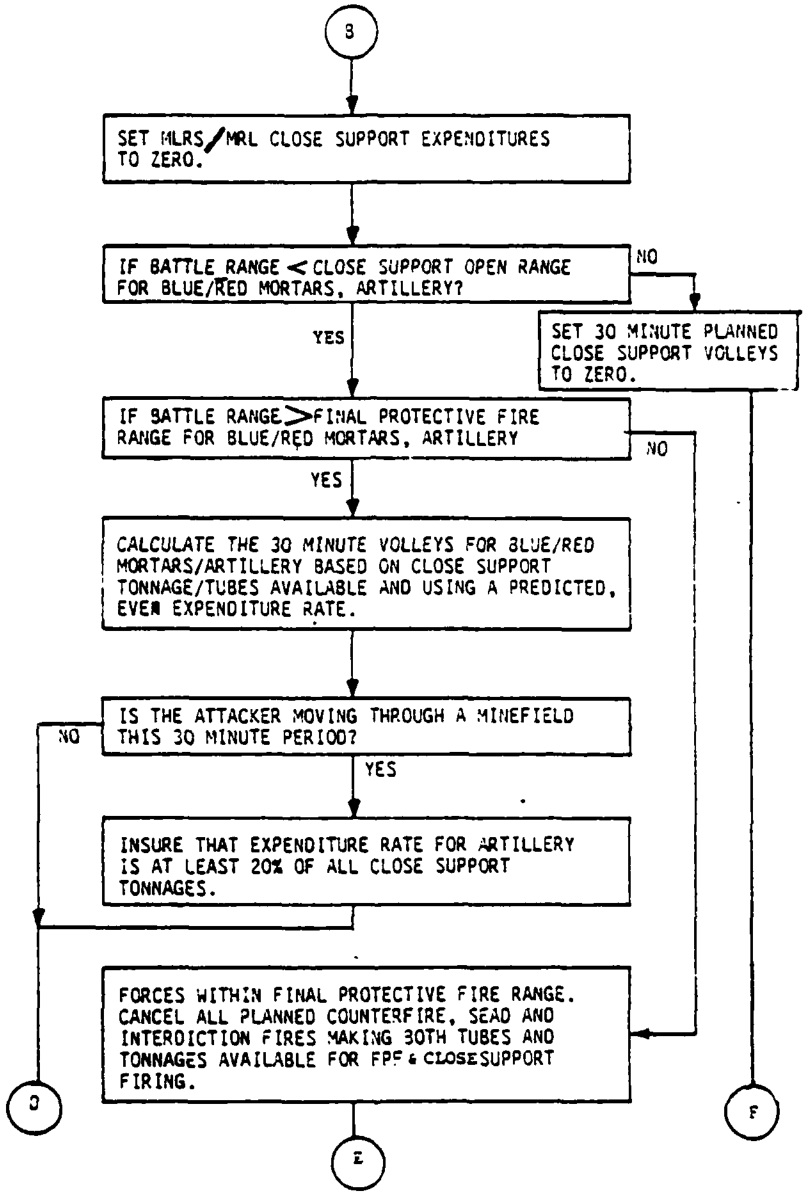


Figure 6-11. Logic flow for 30-minute allocation of artillery, MLRS/MRL and mortar volleys in subroutine Arty_arrive (continued).

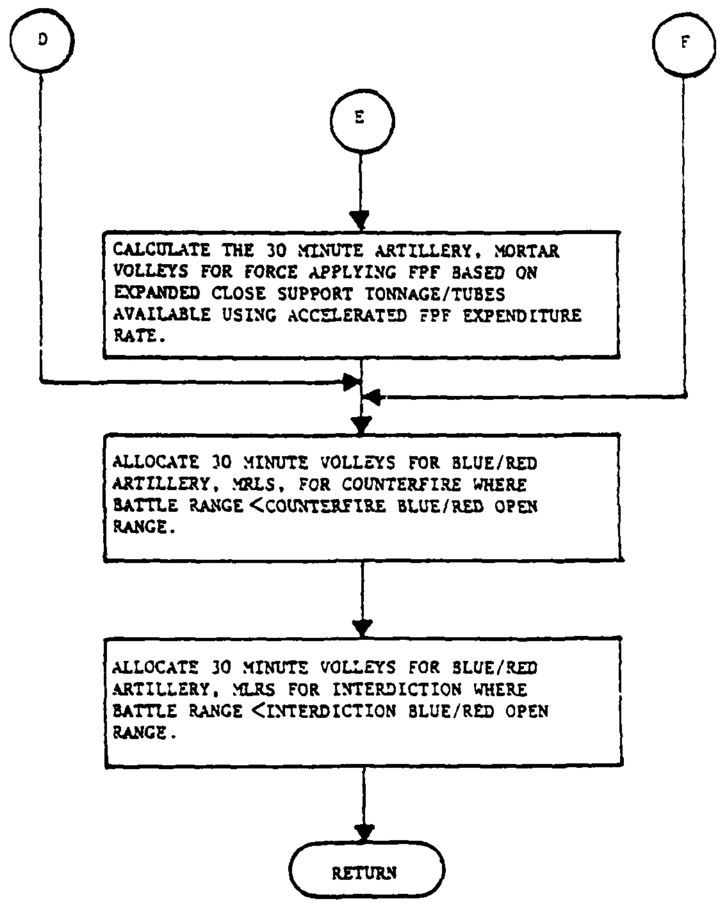


Figure 6-11. Logic flow for 30-minute allocation of artillery, MLRS/MRL and mortar volleys in subroutine Arty_arrive (concluded).

(2) The number of volleys allocated for each task depends upon the most restrictive of the following constraints:

(a) The number of tubes surviving to fire a particular task.

(b) The amount of ammunition available for use in a particular task for this 30-minute period.

(3) The subroutine uses the following equation for allocations by artillery, mortars, and MLRS/MRL:

$$V_{ij} = MIN\ (A_{ij}/W_{tj},\ T_{ij} * R_j/W_{tj}) \qquad \text{(Eq. 6-4)}$$

where:

V_{ij} = the number of volleys from indirect fire system type j fired in response to requests for task i support.
T_{ij} = surviving tubes of type j assigned to task i.
R_j = operational 30-minute firing rate of system type j (tons per 30 minutes).
Wt_j = weight (tons) of one round from indirect system of type j.
A_{ij} = tons of ammunition of type j available for task i.

For all tasks other than close support, the A_{ij} value is simply the amount of ammo initially allocated by subroutine Set_conditions. This amount has been decremented by consumptions which occurred during all previous 30-minute periods. However, for close support, the subroutine attempts to limit the amount of ammunition available each 30-minute period in an effort to keep constant pressure on the enemy. Figure 6-12 provides a graphic description of this process.

As the enemy advances at a constant rate, each 30-minute time step, the subroutine allocates a constant 10 percent of all ammunition available for close support. The subroutine maintains a reserve of 50 percent of all close support ammunition for use in the final protective fire (FPF) phase of the battle beginning at 1.5 km. Normally, a force does not advance at a constant rate. The force will move forward and then, suffering the effects of suppression from attrition and incoming fire, will slow during subsequent 30-minute advance periods. In this case, the indirect fire module sets as its goal the even allocation of all close support ammunition against an advancing force while maintaining a significant percent of the ammunition as an FPF reserve.

Referring again to Figure 6-12, it will be noted that the program allocates the close support ammunition for the 30-minute period before calculating force movement for that period. Consequently, the movement distance of the previous 30 minutes is used assuming that the advancing force will continue

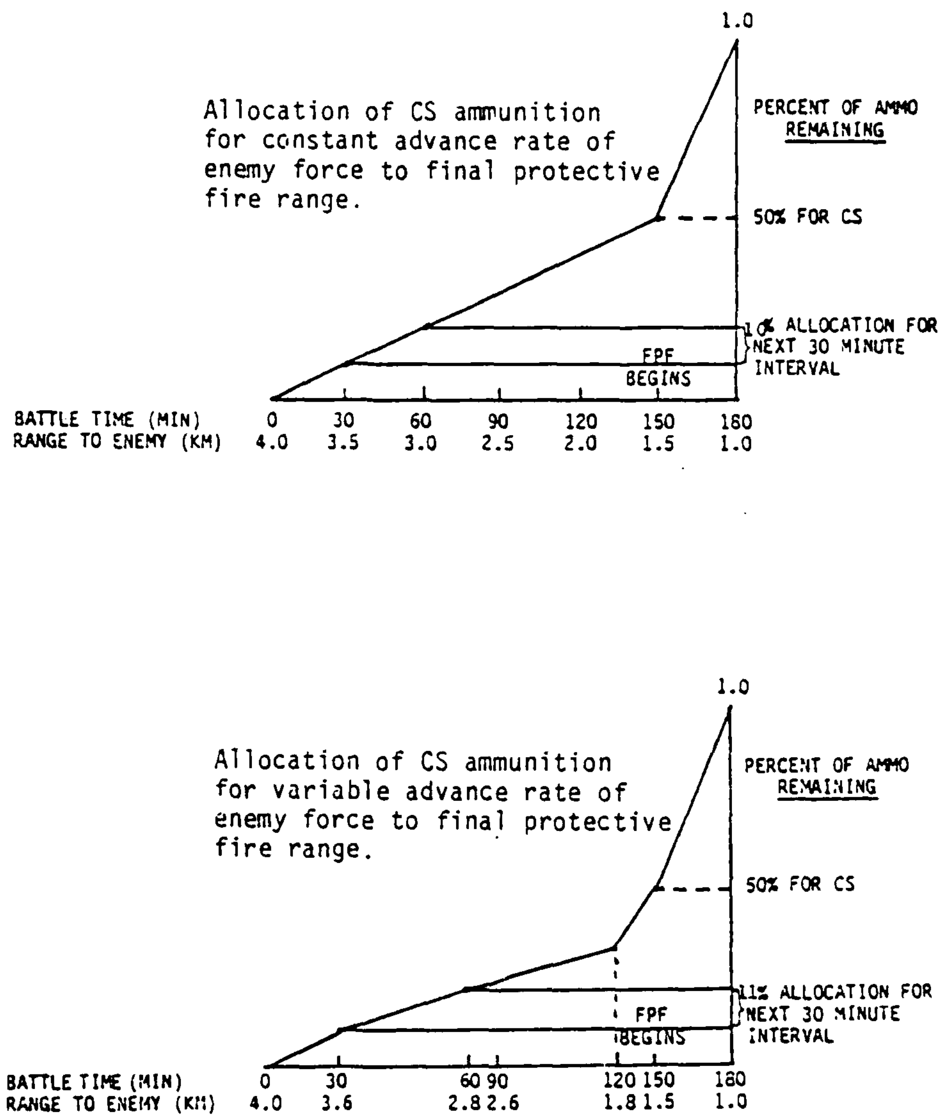


Figure 6-12. Graphical depiction of procedure for close support (CS) indirect fires. Graphs represent allocations against a force advancing at constant and variable rates.

at that rate during this 30-minute period. The following simple linear approximation is used to guide the uniform allocation of close support ammunition against the enemy as it moves to FPF range:

$$F_{ij} = \begin{cases} \dfrac{(1 - Fp)(Rs - Rt)}{Rs - Rp} - \dfrac{Xt_{ij}}{Xa_{ij}} & ; \text{ if } Rt > Rp \\ \left(1 - \dfrac{Xt_{ij}}{Xa_{ij}}\right) * \left(\dfrac{Rt' - Rt}{Rt - Rb}\right) & ; \text{ if } Rb < Rt < Rp \\ 1 - \dfrac{Xt_{ij}}{Xa_{ij}} & ; \text{ if } Rt < Rb \end{cases} \qquad \text{(Eq. 6-5)}$$

$$Fa_{ij} = MAX\ (Fmin,\ F_{ij}) \qquad \text{(Eq. 6-6)}$$

$$A_{ij} = MIN\ (Fa_{ij} * Xa_{ij},\ Xa_{ij} - Xt_{ij}) \qquad \text{(Eq. 6-7)}$$

where:

- F_{ij} = the fraction of indirect fire ammunition tonnage for system j allocated for close support (i) for this 30-minute timestep with strict maintenance of FPF reserves.
- Fp = the fraction of close support ammo held in reserve for use in final protective fires. Values currently used for both artillery and mortars are Blue 0.5 and Red 0.3.
- Rs = tactical phase line open fire range of artillery close support.
- Rt = range of target at beginning of this 30-minute period.
- Rp = phase line range at which final protective fire (FPF) will begin.
- Rb = range at which defender direct fire forces will break.

Xt_{ij} = total tonnage of type j consumed in close support (i) in all previous 30-minute timesteps.

Xa_{ij} = total tonnage of type j available for close support (i) including FPF tonnage.

Rt' = range of target at beginning of 30-minute period just prior to current period.

Fa_{ij} = fraction of ammunition type j available for close support task (i).

Fmin = factor representing lower bound for close support allocations on targets of opportunity. Fmin=0.083 for lower bound or 0.20 if target is in minefield.

A_{ij} = is as described in equation 6-4 with i as the close support tasks.

G. Force movement.

(1) The DIME ground combat program calculates the movement of the attacker force every 30 minutes. The program assumes that only the attacker force moves with the defender remaining fixed. Figure 6-13 provides a flow diagram of the procedure used to determine movement in subroutines Calc_movement and Mine_encounter. Principal variables for these subroutines can be found in Table 6-5.

(2) Subroutine Calc_movement selects the unsuppressed velocity for the attacking force based on the terrain, mission, day/night, and whether the force is primarily mounted or dismounted. It then calculates three suppression factors affecting the movement.

(a) Suppression from Calc_movement for a 30-minute period can be suppressed up to 40 percent based on casualties from artillery, direct fire, and helicopters sustained in the previous 30 minutes. The following equations are used to represent movement suppression from casualties:

$$Psc = MIN \left(.4,\ .4 * \left[\frac{Efp - Efc}{Eff} \right] \Big/ .06 \right) \qquad \text{(Eq. 6-8)}$$

where:

Psc = the fraction of force movement suppressed this 30 minutes due to casualties.

Efp = the force effectiveness score at the beginning of the previous 30-minute period.

Efc = the force effectiveness score at the end of the previous 30-minute period (i.e., current score at the beginning of this period).

Eff = the force effectiveness score at the beginning of this battle.

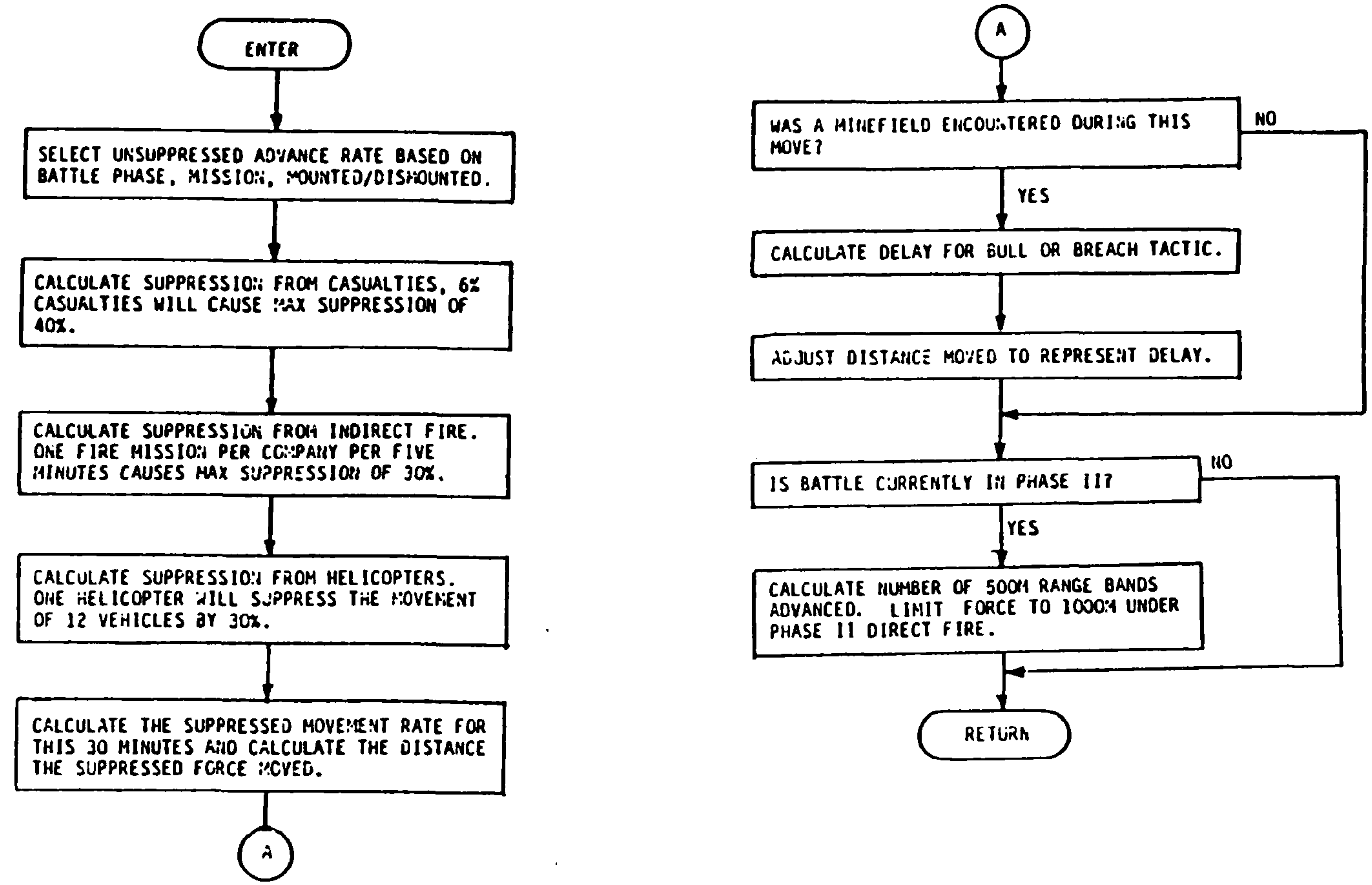


Figure 6-13. Logic flow for 30-minute movement calculation found in subroutines Calc_movement and Mine_encounter.

Note that the force effectiveness (Eft) at any time in the battle is calculated by:

$$Eft = \sum_{i=1}^{n} N_i * W_i \qquad \text{(Eq. 6-9)}$$

where:

N_i = the number of surviving elements of type i.
W_i = the effectiveness weight assigned to element i.
n = 70; the number of different system elements.

(b) Movement suppression from indirect fire. Calc_movement also considers suppression from indirect fire. The suppression is directly proportional to the average number of indirect fire missions directed toward each company of the attacking force. The subroutine calculates the number of companies surviving in the attacking force and then determines the average number of indirect fire (artillery, mortar, and MLRS/MRL) missions being sustained by the advancing companies. A maximum suppression level of 40 percent degradation in movement can be achieved by an average of six missions in 30 minutes falling on a maneuver company. The following equation is used to calculate movement suppression from indirect fire:

$$Ceqt = \left(\sum_{i=1}^{20} M_i * Nt_i + \sum_{i=36}^{47} M_i * Nt_i/8 \right) / 28 \qquad \text{(Eq. 6-10)}$$

where:

Ceqt = the number of company equivalents in the attacking force at a particular time.
M_i = mask set to 1, if the ith element is found in a maneuver company; 0 otherwise.
Nt_i = the number of attacker elements of type i surviving at a particular time.
28 = the number of direct fire elements (i = 1 to 20) plus infantry (i = 36–47) approximating a DIME company unit.

The number of indirect fire missions is given by:

$$Idm = \frac{\sum_{i=1}^{7} (A_{i,1} + A_{i,2})ARTY}{1.8} + \frac{\sum_{i=8}^{11} (A_{i,1} + A_{i,2})MORT}{1.2} + \frac{\sum_{i=12}^{15} (A_{i,1})M}{1.8}$$

(Eq. 6-11)

where:

- Idm = the number of indirect fire missions falling on the attacking force this 30 minutes.
- $(A_{i,1} + A_{i,2})ARTY$ = the total tonnage for artillery missions of prep and close support falling on the attacker during these 30 minutes. Note that 1.8 tons fired represents one artillery mission.
- $(A_{i,1})M$ = the total tonnage for multiple rocket launcher missions of prep fired at the attacker during these 30 minutes. Note that 1.8 tons fired represents one MRL mission. To calculate Idm for Blue, replace tonnages with Blue amounts.
- $(A_{i,1} + A_{i,2})MORT$ = the total tonnage for mortar missions of prep and close support fired at the attacker during these 30 minutes. Note that 1.2 tons represents one mortar mission.
- A_{ij} = is as defined by Eq. 6-7.

The suppressive effects on movement by indirect fire is given by:

$$Psid = Rbf * MIN (Idm/Ceqt * 6, 1) \qquad \text{(Eq 6-12)}$$

where:

- Psid = the fraction of force movement suppressed these 30 minutes due to indirect fire.
- Rbf = the maximum fraction of movement suppression due to indirect fire for the attacking force (Blue attacker 0.4; Red attacker 0.3).

(c) Movement suppression from attack helicopter. Movement is also suppressed by attack helicopters. The subroutine considers a maximum suppression of 30 percent of the force movement rate if one helicopter has been allocated to each 12 vehicles of the attacking force. The movement degradation is determined by the following expression:

$$Va = \sum_{i=1}^{20} Nt_i \qquad \text{(Eq. 6-13)}$$

where:

- Va = the number of maneuver vehicles in the attacking force which are potential targets for helicopter engagement.
- Nt_i = number of elements i in the attacking force surviving at a particular time.

$$Psh = 0.3 * MIN\ (12 * Hn/Va,\ 1) \qquad \text{(Eq. 6-14)}$$

where:

- Psh = the fraction of the advance rate suppressed by the helicopters.
- Hn = The number of defender helicopters currently attacking the defensive force.

The effective movement rate for this 30-minute period then becomes:

$$R_{30} = (1 - Psc - Psid - Psh) * Rm \qquad \text{(Eq. 6-15)}$$

where:

- R_{30} = the movement rate for these 30 minutes (meters per 30 minutes).
- Rm = the unsuppressed movement rate. This rate is a function of mounted/dismounted, day/night, mission, and terrain. Given in meters per 30 minutes.

(3) The actual distance moved by the force during the next 30-minute period is determined in subroutine Mine_encounter. This code projects the unimpeded force movement and then determines which, if any, minefields will be encountered by the attacking force as it advances this 30-minute period. If a minefield is encountered, the subroutine delays the force at the edge

of the minefield until it has opened the minefield. The force uses two tactics to open the minefield. The more cautious breach tactic is used when the trapped force finds itself under the following conditions:

(a) The attacking unit is beyond direct fire range, i.e., still in the phase I artillery battle (30-minute maximum delay).

(b) The attacking unit has suffered attrition to within five percent of its tactical break point (45-minute maximum delay).

The bolder bull tactic occurs only if the attacker finds himself delayed within direct fire range of the defender. In this case, the maximum delay time is only 10 minutes. However, the choice of the bull tactic penalizes the attacker more heavily in systems lost to mines. After selecting the proper clearing tactic, subroutine Mine_encounter assesses the actual delay using the following expression:

$$Dm = Md * Mw * Fd / Sw \qquad \text{(Eq. 6-16)}$$

where:

- Dm = the delay (minutes) of the attacking force as it clears the minefield.
- Md = maximum delay to clear a minefield affecting the entire force (minutes).
- Mw = width of the minefield (km).
- Sw = width of the maneuverable sector channelizing the attack (km).
- Fd = fraction of the force entering the minefield.

After assessing the mine delay in Mine_encounter, subroutine Calc_movement then determines the actual advancement for this 30-minute period using the following set of equations:

$$D = 30 - MIN(Dm + Dm', 30) \qquad \text{(Eq. 6-17a)}$$

$$Mm = \begin{cases} R_{30} * D \text{ ; for Phases I or III} \\ MAX\left(2, \left[\frac{R_{30} * D + 250}{500}\right]_I\right) \text{ ; For Phase II} \end{cases} \qquad \text{(Eq. 6-17b)}$$

$$M_{30} = Mm * 500 \qquad \text{(Eq. 6-17c)}$$

where:

M_{30} = the movement for this 30-minute period (meters).
Dm' = any mine delay from the previous 30 minutes extending into this 30-minute period (minutes)
$(R_{30} * D + 250)/500_I$ = indicates truncation to integer form.

Note that equations 6-17 have two implications for the attacking force finding itself in phase II:

(1) The force can advance no more than 1 km during a 30 minute period.

(2) If the force cannot move 250 meters or more, it is considered stationary.

6. "UNITFILE" IMPACT.

The ground combat program takes the kills associated with mines, indirect fire, direct fire, helicopters, PGMs, and infantry and deducts them from "UNITFILE" positions 1 to 70. Associated ammunition/fuel consumption is updated within the "UNITFILE". Ground combat attrition modules (discussed in following sections) return kills and ammunition consumption to this ground combat driver and are then updated to the "UNITFILE".

7. CODE.

A. Following the allocation and movement calculations discussed in paragraph 5 of this chapter, the ground combat program moves to one of three subroutines which drive the attrition algorithms for each phase. Although the methodology used in the code for the selection of a battle phase is not complex, it should be noted that this point represents a critical transfer of control in the DIME code structure. Once the phase is selected, the phase drivers (subroutines Phase1_btl, Phase2_btl, and Phase3_btl) are structured as shown in Figure 6-7. They have been written with considerable redundancy which was designed for ease of phase modification and debugging. The phase driver subroutines are as follows:

(1) Phase1_btl. The movement to contact phase, found in the code in Table 6-14, drives the attrition algorithms for this part of the battle. See paragraph 2D(1) for a description of phase I. Attrition available to this phase is as follows:

(a) Minefield attrition.
(b) Indirect fire (artillery) attrition.
(c) Direct fire attrition (15-minute contact).
(d) Helicopter attrition.
(e) Precision-guided munition (PGM) attrition.

The order of attrition does not change. If a type of attrition does not take place, it is skipped and attrition proceeds in order.

(2) Phase2_btl. The direct fire phase drives the attrition algorithms for this portion of the battle. See paragraph 2D(2) for a description of phase II. Attrition available to this phase is similar to phase I with the following exceptions:

(a) Direct fire occurs for 30 minutes in phase II.

(b) The infantry attrition module is engaged after PGMs if forces are within 500 meters of each other.

(3) Phase3_btl. The withdrawal phase drives the attrition algorithms for this part of the battle. See paragraph 2D(3) for a description of the withdrawal phase. Attrition available to this phase is as follows:

(a) Indirect fire.
(b) Delivery of smoke for withdrawal.
(c) Direct fire.
(d) Helicopter attrition.
(e) Precision-guided munitions (PGMs).
(f) Infantry attrition if within 500 meters.

B. Each type of attrition exists as separate modules within the ground combat code.

(1) In order to prepare for the minefield attrition module, subroutine Run_mine is called by the phase driver (Phase1_btl or Phase2_btl). Subroutine Run_mine mounts or dismounts troops, corrects the number of dismounted infantry, and keeps track of infantry kills on carriers as it sends needed parameters to the minefield attrition module (Mines). For more information on the minefield module, see Section I of this chapter.

(2) Indirect fire module preparation begins with the calling of subroutine Arty_sub by the phase driver (Phase1_btl, Phase2_btl, or Phase3_btl). Arty_sub prepares and sends needed parameters to the artillery attrition module while keeping track of current mounted and dismounted troops. The artillery module, Arty_atrit, is a module of Arty_sub. A complete description of the module and subroutine Arty_sub may be found in Section II of this chapter.

(3) Artillery/mortar-delivered white phosphorous smoke may be allocated to degrade the visibility of the withdrawing force. Degradation of visibility reduces the amount the withdrawing force can be seen as targets in direct fire combat. The smoke is allocated during the artillery allocation discussed in paragraph 5. The percent of frontage visible through a smoke screen is calculated by the smoke module and later used by the direct fire module during withdrawal. For a complete discussion of the smoke allocation, smoke module, and its effects in the direct fire module, refer to Section III of this chapter.

(4) Section IV of this chapter is devoted to direct fire attrition. The phase driver, after preparing some of the parameters, calls subroutine Df_cbt. This subroutine prepares more parameters, including the current number of mounted and dismounted troops, and sends them to the Df_attrition module.

(5) The helicopter module is called directly from the phase driver. This means all parameter preparation (including mounted/dismounted troops) is included in the phase driver subroutines. The duplication in each phase should be noted and remembered in case of code changes. The helicopter module, itself, is discussed thoroughly in Section V of this chapter.

(6) In order to prepare for the PGM attrition module, subroutine Clgp_gamp_atrit is called by the phase driver (Phase1_btl, Phase2_btl, or Phase3_btl). Subroutine Clgp_gamp_atrit mounts or dismounts troops, corrects the number of dismounted infantry, and keeps track of infantry kills on carriers as it sends needed parameters to the Pgm_atrit module. A complete description of the PGM module may be found in Section VI of this chapter.

(7) Preparing for the infantry attrition module, subroutine Infantry_cbt is called by the phase driver (Phase2_btl, or Phase3_btl). Subroutine Infantry_cbt helps keep track of the current number of dismounted infantry and sends needed parameters to the infantry module. For more information on the infantry module, see Section VII of this chapter.

C. During the preparation for each module, excluding smoke, the current number of mounted and dismounted troops is calculated. To do this, two other modules are accessed. The Dismounted module is called during the preparatory phase of attrition module calling. This module, in turn, calls the Load_infantry module.

(1) Dismounted. This module checks the force's mission and distance from opponent. When in a defensive mission (hasty, prepared, reserve, or ambush), all troops are dismounted. A frontal attack mission within 600 meters of the enemy also calls for dismounted troops. If mounted troops are possible, the Load_infantry module is called to determine the number of infantry personnel per carrier. The number of mounted troops is calculated by multiplying the total number of carriers times the personnel per carrier. The remaining troops (the beginning infantry minus the number of mounted) are left to be dismounted.

(2) Load_infantry. In order to calculate the number of personnel per carrier, a check is made for the mission. A defensive mission has a load factor of zero, so that all the troops remain dismounted. If the force has an attacking mission, the load factor is the minimum of the total number of infantry divided by the number of carriers, or eight, which is the maximum load factor.

(3) The subroutines which prepare the parameters for the attrition modules save the original number of infantry. This is then replaced by the number of dismounted troops before the number of system elements are sent to the modules. Upon return from the modules, the load factor times the number of personnel carriers killed is added to the total number of infantry killed. The saved number of infantry minus the total number of infantry killed now becomes the current number of infantry personnel.

D. The ground combat driver's subroutines and their primary variables are contained in Table 6-5. A listing of P4 code appears in Table 6-14.

Table 6-5. Subroutine table for ground combat/driver routine

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_data	Reads small arrays into programs from data statements.	A. Sys_eff (I,J)	Firepower score for system J (1-70) of side I (1 = Blue, 2 = Red)
		B. Wpn_typ (I)	Contains the weapon type for weapon I (1-70). (1 = DF, 2 = IF, 3 = AD)
		C. Ammo_wt (I,J)	Packaged weight (in tons) of individual round /burst of ammo I - 1 = Blue, 2 = Red J - 1 - 70 elements
		D. Arty_30min_wt (I,J)	Contains the weight in short tons that one gun can deliver firing at a sustained rate for 30 minutes (packaged ammo). I - 1 = Blue, 2 = Red J - 1 - 7 = artillery, 8 - 11 = mortars, 12 - 15 = MLRS/MLR
		E. A_wt (I,J)	Weight of artillery round/ packaged weight I - 1 = Blue, 2 = Red J - 1 - 7 = artillery, 8 - 11 = mortars, 12 - 15 = MLRS/MLR
		F. Bf_mask (I) Rf_mask (I)	Mask for selecting forward elements. I = 1-70; If value is 1, element used; if 0, not used
		G. Bdf_mask (I) Rdf_mask (I)	Mask for elements in DF battle. I = 1 - 70, value is 1 or 0.
		H. Ds_start	Start range for artillery DS (close SPT). I = 1 - 4.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_data (concluded)		I. Barty_30min (I,J) Rarty_30min (I,J)	Artillery allocated for 30 minutes for mission I (1-10), artillery type J (1-15).
		J. Basic_ld (I)	Total number of engagements for each element (I = 1 - 70).
		K. B_veh$ R_veh$	Weapon system labels (5 characters each).
Zero_out	Initializes variables used in the combat module.	A. B_unit_no (I) R_unit_no (I)	Unit number of each unit played (I = 1-12).
		B. B_unit_pct (I) R_unit_pct (I)	Percent of unit I committed to sector (I = 1-12).
		C. Init_b_eff Init_r_eff	Initial firepower score.
		D. B_df_ammo R_df_ammo	Direct fire ammunition available to fire.
		E. B_ad_ammo R_ad_ammo	Air defense ammunition available to fire.
		F. Bif_ammo (I) Rif_ammo (I)	Indirect fire ammunition available in sector I. (I = 1-15).
		G. Phase_ct (I)	Holds the number of 30 minute increments we have spent in phase I.
		H. B_unit (I,J) R_unit (I,J)	Number of systems J alive in I. (I = 1-12; J = 1-74.)
		I. Kv_r (I,J) Kv_b (I,J)	Number of losses in system J (1-70) due to killer category I (1-6).

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Zero_out (continued)		J. Sys_tot (I,J)	Keeps the current status of the forces in combat for element J. If I = 1 - Initial Blue system. = 2 - Final Blue system. = 3 - Initial Red system. = 4 - Final Red system.
		K. B_init (I,J) R_init (I,J)	Initial unit systems by unit I, system J. (I = 1-12; J = 1-70.)
		L. B_init (13,J) R_init (13,J)	Holds total weapons for this flight. (J = 1-70.)
		M. Minefield (I,J)	I = 1 to 3 minefields. J = 1 - minefield range = 2 - width of minefield (km) = 3 - width of sector (km) = 4 - % bypassed = 5 - % entering minefield = 6 - minefield assessment 0 = unused minefield 1 = used minefield
		N. Time_seg	Number of the current 30 minute time segment.
		O. Last_bah1_seg Last_bah2_seg Last_bsct_seg Last_rah1_seg Last_rah2_seg Last_rsct_seg	Time segment when helicopter was last flown.
		P. Bah1_seg Bah2_seg Bsct_seg Rah1_seg Rah2_seg Rsct_seg	Number of helicopter missions flown when 3 cells are used.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Zero_out (concluded)		Q. B_dsarty_fire (I) R_dsarty_fire (I)	Number of tons of artillery ammo fired in CS mission (I = 1 - 7).
		R. B_dsmort_fire (I) R_dsmort_fire (I)	Number of tons of mortar ammo fired in CS mission (I = 1 - 4).
		S. Barty_fire Rarty_fire	Number of 30 minute segments Red has been under Blue artillery fire.
		T. Mine_delay	Time delay due to mines.
		U. Bif_msn_tons (I,J) Rif_msn_tons (I,J)	Contains tons of ammo available this 30 minute segment by weapon type I (1 - 15) on mission type J (1 - 5).
		V. B_con (I,J) R_con (I,J)	Percentage of total number of elements of system J (1 - 70) provided by unit I (1 - 12).
Set_battle	Allows input of the sector worksheet Determines Blue element percentages. Calculates Red target parameters.	A. B_msn (I) R_msn (i)	Blue mission Red mission.
		B. Mift (I)	Percent of systems in unit I that can be engaged by combat systems in the indirect fire battle (I = 1 - 20).
		C. Mdft (I)	Percent of systems in unit I that can be engaged by combat systems in the direct fire battle (I = 1 - 20).
		D. Mifdt (I)	Rate of change representing percent of elements of unit I that can be introduced/extracted from the IF battle per 30 minute interval while maintaining the same mission (I=1-20)

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Set_battle (continued)		E. Mdfdt (I)	Rate of change representing the percent of elements of unit I that can be introduced/extracted from the direct fire battle per 30 minute interval, while maintaining the same mission (I = 1 - 20).
		F. Mfire (I)	Percent of systems in unit I that can effectively fire in battle (I = 1-20).
		G. Mdfire (I)	Rate of change representing the percent of elements of unit I that can effectively fire and which can be introduced into the battle every 30 minutes (I = 1-20).
		H. Mvul (I)	The fraction (0 - 1.0) of a system in unit I that is fully exposed to direct fire. (I = 1-20).
		I. Mdvul (I)	Percent increase/decrease in the vulnerability of a system in unit I on a particular mission (I = 1-20).
		J. B_if_t (I,E) R_if_t (I,E)	The percent of systems that can be engaged by combat systems in the indirect fire battle for unit E on a mission for 3 hour segment I.
		K. B_df_t (I,E) R_df_t (I,E)	The percent of systems that can be engaged by combat systems in the direct fire battle for unit E on a mission for 3 hour segment I.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Set_battle (concluded)		L. B_if_dt (I,E) R_if_dt (I,E)	The rate of change representing the number of elements engaged in combat and under indirect fire for unit E on a mission for 3 hour segment I.
		M. B_df_dt (I,E) R_df_dt (I,E)	The rate of change representing the number of elements engaged in combat and under direct fire for unit E on a mission for 3 hour segment I.
		N. B_f (I,E) R_f (I,E)	Percent of systems of unit E on mission I that can effectively fire in battle.
		O. B_df (I,E) R_df (I,E)	Rate of change representing the percent of elements of unit E on mission I that can effectively fire in battle.
		P. B_v (I,E) R_v (I,E)	Percent of unit E fully exposed while on mission I.
		Q. B_dv (I,E) R_dv (I,E)	Change in percent of unit E fully exposed while on mission I.
		R. B_type (I) R_type (I)	Unit type (I = 1-12).
Ready_load	Ready infantry and direct fire loads.	A. Sum_inf	Sum of infantry which will be mounted.
		B. Sum_df	Sum of the direct fire carriers designated as infantry carriers.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
L1	Enters the first line of data for starting the battle.		
L2	Sets up the Blue units fighting the battle; calculates Blue ammo available.	A. Bada_hnd (I)	Fraction of Blue hand-held AD elements suppressed in unit I.
		B. Bada_veh (I)	Fraction of Blue vehicular AD elements suppressed in unit I.
		C. If_ammo (J)	Number of tons of ammo available for current battle for: J = 1 - 7 artillery 8 - 11 mortar 12 - 15 MLRS/MRL
L3	Sets up the Red units committed to battle; calculates Red ammo available.	A. Rada_hnd (I)	Fraction of Red hand-held AD elements suppressed in unit I.
		B. Rada_veh (I)	Fraction of Red vehicular AD elements suppressed in unit I.
Ad_sup	Calculates percentages of air defense elements suppressed; calculates Red and Blue target target parameters.	A. Bhnd_sup Rhnd_sup	Fraction of all Blue/Red hand-held AD elements suppressed.
		B. Bveh_sup Rveh_sup	Fraction of all Blue/Red vehicular AD elements suppressed.
		C. Tot_init_i	Initial total number of vehicle type AD weapons for a certain sector unit.
		D. Tot_init_13	Initial total number of vehicle type AD weapons for the battle.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

'Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
L4	Enter battle parameters	A. Init_rg	Initial range (in meters).
		B. Df_rg	Direct fire range (in meters).
		C. No_minefields	Number of minefields.
L5	Enter weather conditions		
L6	Enter side attacking		
L7	Enter Blue mission data	A. B_terr	Blue terrain type (1-5)
		B. B_rg_break	Range (m) at which Blue must break.
		C. B_pct_fwd	Percent of Blue force forward.
		D. B_mopp	Mopp fatigue degradation.
		E. T_length (I)	Length of sector
		F. T_width (I)	Width of sector.
		G. B_break_t (I)	Amount of time Blue will fight before breaking.
		H. B_cas_break	Percent of casualties at which Blue will break.
L8	Enter Red mission data	A. R_terr	Red terrain type (1-5).
		B. R_rg_break	Range (m) at which Red must break.
		C. R_pct_fwd	Percent of Red force forward.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
L8 (concluded)		D. R_mopp	Mopp fatigue degradation.
		E. T_length (I)	Length of sector
		F. T_width (I)	Width of sector.
		G. R_break_t (I)	Amount of time Red will fight before breaking.
		H. R_cas_break	Percent of casualties at which Red will break.
L9	Enter Blue helicopter data	A. B_helo (I,J)	Data of type J = 1 # of helo = 3 # of cells for helicopter I = 1 Blue type 1 attack helo = 2 Blue type 2 attack helos = 3 Blue Scouts
		B. B_helo_atkprof(I)	Blue helo type I (I=1-2) attacker profile where 1 = missiles 2 = missiles and guns 3 = guns 4 = air-to-air missiles 5 = air-to-air and ground missiles 6 = air-to-air, ground missiles & guns 7 = air-to-air missiles and guns
		C. B_helo_delay	Time helicopter is at large from start of battle until it enters the battle.
		D. B_helo_rg_delay	Range at which helicopters are to enter battle (0= none, otherwise input in m).

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

L9 (Concluded)		E. B_helo_msn(I)	Mission for Blue helo of type I (I = 1-2): 1 = air-to-ground mission 2 = air-to-air mission 3 = SEAD mission
		F. B_atk_rg(I)	Standoff range (meters) of Blue helicopter type I (I=1-2)
L10	Enter Red helicopter data	A. R_helo (I,J)	Data of type J = 1 - # of helos = 2 - # of helo cells. for helicopter I = 1 - Red type 1 attack helo = 2 - Red type 2 attack helos = 3 - Red Scouts
		B. R_helo_atkprof(I)	Red helo type I (I=1-2) attacker profile
		C. R_helo_delay	Time helicopter is at large from start of battle until it enters the battle.
		D. R_helo_rg_delay	Range at which helicopters are to enter battle (0= none, otherwise input in m).
		E. R_helo_msn(I)	Mission for Blue helo of type I (I = 1-2): 1 = air-to-ground mission 2 = air-to-air mission 3 = SEAD mission
		F. R_atk_rg(I)	Standoff range (meters) of Blue helicopter type I (I=1-2)

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
L11	Enter Blue artillery data	A. Bif_msn (I)	Contains the percent of IF tonnage to be allocated to individual mission I.
		B. B_prep_time	Time Blue will begin preparation for battle. Input as hhmm, with hh = the hour and mm - the minutes after start time the preparation begin.
		C. No_gamp	Number of GAMP available.
		D. No_clgp	Number of CLGP available.
		E. Perc_gamp	Percentage of all IF tonnage filled by GAMP.
		F. Perc_clgp	Percentage of all IF tonnage filled by CLGP.
L12	Enter Red artillery data	A. Rif_msn (I)	Contains the percent of IF tonnage to be allocated to individual mission I.
		B. R_prep_time	Amount of time after start time that Red will begin preparation for attack.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Lmines	Enter minefield data	Minefield (I,J)	Contains minefield information where: I: (1-3) minefields J: 1 = minefield range 2 = minefield width 3 = sector width 4 = percent of forces entering minefields.
Set_print	Prints out initial battle data, including who is attacking, the type of mission, the type of terrain, helicopter information and other pertinent data.		
Set_conditions	Set up battlefield conditions. Set indirect fire mission tonnages for 6 hour battle. Set direct support artillery/mortar parameters.	A. B_cbt_eff R_cbt_eff	Current Blue/Red firepower score.
		B. Btl_time	Current time on the battlefield (in 30 minute intervals).
		C. Max_btl_time	The latest possible time for the battle to stop.
		D. Btl_rg	Current range between forces on the battlefield.
		E. First_bnd	Initial range band (1-6) for DF.
		F. Gamp_avail	Number of GAMP available for the current battle.
		G. Clgp_avail	Number of CLGP available for the current battle.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Set_conditions (concluded)		H. B_dsmort_brkrg R_dsmort_brkrg	End range for CS mortar support.
		I. B_dsarty_brkrg R_dsarty_brkrg	End range for CS artillery support.
		J. B_dsmort_start R_dsmort_start	Start range for CS mortar support.
		K. B_dsarty_start R_dsarty_start	Start range for CS artillery support.
		L. B_ds_conc_pt R_ds_conc_pt	Point (meters) to begin concentrated artillery fire prior to break
		M. B_mo_conc_pt R_mo_conc_pt	X-point at knee of CS curve
		N. B_mo_conc_level B_ds_conc_level R_mo_conc_level R_ds_conc_level	Y-point at knee of CS curve.
		O. B_p30_artyrg R_p30_artyrg B_p30_mortrg R_p30_mortrg	Battle range at end of previous 30 minute period.
		P. Advance_rate (I,J)	Maximum rate of advance (meters per minute for force on mission J and advance type: I = 1 advance mounted during phase I. = 2 advance dismounted during phase I. = 3 withdraw mounted during phase III. = 4 withdraw dismounted during phase III.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Print_sys_out	Prints out data on specified system	S (I)	Array used to print out data on weapon system elements (I = 1-70).
Control_battle	Calculates attrition assessments for ground battle. Update battle time. Set attack helicopter arrivals.	A. Saty (I)	Tons of ammo available this 30 minute segment by weapon type I (1-15) on all missions.
		B. Tot_volley (I)	Total number of rounds fired by weapon type I for all missions. (I = 1-15).
		C. Volley (I,J)	Number of rounds fired by weapon type I on mission J. (I = 1-15).
		D. Amt_of_advance	Distance moved in meters.
Print_eff	Prints the current effectiveness of Blue and Red forces.	A. B_eff_pt R_eff_Pt	The current effectiveness of the entire Blue/Red force.
Print_volleys	Prints data on amount of ammunition used.		
Print_init_res	Prints out initial unit status.		
Print_fin_res	Prints out final battle results.		

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Chk_brk_pt	Checks for battle breakpoints.		
Tally_cbt_eff	Calculates the adjusted firepower score.	A. Prev30_b_eff Prev30_r_eff	Firepower scores for the previous 30 minute segment for entire Blue/Red force in combat.
Apport_wri_loss	Writes unit status to history file.	A. Bif_left Rif_left	Contains total tons of ammo left available for all weapon types and missions.
		B. Bif_ammo_used Rif_ammo_used	Total tons of ammo used this 30 minute segment by all weapon types for all missions.
		C. *N* (*)	Represents the 150 elements from 1 record of the *UNITFILE*.
		D. B_df_ammo_used R_df_ammo_used	Tons of direct fire ammo used in this 30 minute segment of battle.
		E. B_ad_used R_ad_used	Tons of air defense ammo used in this 30 minute segment of battle.
		F. Ci_kv_b (I,J) Ci_kv_r (I,J)	Cumulative killer/victim matrix for entire game with killer J and victim I.
		G. Ci_helo_b (I,J) Ci_helo_r (I,J)	Cumulative data on helicopters for entire game (same format as B_helo).

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Helo_arrive	Calculates the attack helicopters available this period.	A. Bah1	Number of Blue type 1 attack helos attacking this segment.
		B. Bah2	Number of Blue type 2 attack helos attacking this segment.
		C. Bsct	Number of Blue scouts attacking this segment.
		D. Rah1	Number of Red attack helicopters in cell 1 attacking this segment.
		E. Rah2	Number of Red attack helicopters in cell 2 attacking this segment.
		F. Rsct	Number of Red scouts attacking this segment.
		G. Earliest_time	Earliest possible arrival time of helicopters.
		H. Helo	The number of helicopters which will arrive on station.
Chk_delay_time	Calculates delay times for helicopters.		
Ammo_breakdown	Apportions ammunition for use by weapon systems.	A. Sys_ammo (I)	Ammo weight for each indirect element per 30 minutes. J = 1 - 7 artillery 8 - 11 mortar 12 - 15 MLRS/MRL

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Arty_arrive	Calculates incoming artillery. Updates artillery capacity for tonnage not used. Schedule incoming prep fires. Allocate mortar fires for prep. Allocate prep fires for artillery pieces. Allocate prep fires for MLRS pieces. Allocate CLGP missions. Schedule incoming direct support fires. Schedule incoming counter fire missions. Schedule incoming interdiction missions.	A. B_vis (I) R_vis (I)	Visibility percent (0-1.0) for weapon type I.
		B. B_arty_cap (I,J) R_arty_cap (I,J)	Total weight of ammo which can be shot by artillery (I) in the current 30 minute segment for for mission J (J=1-5; I=1-4).
		C. B_mlrs_cap (I,J) R_mlrs_cap (I,J)	Total weight of ammo that can be shot by MLRS/MLR (I) in the current 30 minute segment for mission J (J=1-5; I=1-4).
		D. B_mort_cap (I,J) R_mort_cap (I,J)	Total weight of ammo which can be shot by mortars (I) in the current 30 minute segment for for mission J (J=1-5; I=1-4).
		E. Int_bmort Int_rmort	Tons of ammunition available to mortars at beginning of segment.
		F. Tot_arty (I,J)	Subtotal of amount of ammo delivered per 30 minutes (I=1-2; J=1-15).
		G. B_clgp_cap (I)	Weight of ammo available to CLGP in this 30 minute segment (I = 1-7).
		H. B_gamp_cap (I)	Weight of ammo available to GAMP in this 30 minute segment (I = 1-4).

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Arty_arrive (continued)		I. Tot_clgp	Total weight of ammo available to CLGP in current 30 minute segment.
		J. Tot_gamp	Total weight of ammo available to GAMP in current 30 minute segment.
		K. Cloud_ht	Cloud height over battle area (in meters).
		L. B_smk_cap (I) R_smk_cap (I)	Total weight of ammo which can be shot for smoke in current 30 minute for weapon type I where I = 1 - 7 artillery 8 - 11 mortar
		M. B_smok_tons(I) R_smok_tons(I)	Total weight of ammo which can be shot for smoke in current 30 minute for weapon type I where I = 1 - 7 artillery 8 - 11 mortar
		N. A_ammo_ton	Total artillery ammo tonnage for smoke.
		O. M_ammo_ton	Total mortar ammo tonnage for smoke.
		P. B_asmk_used (I) B_msmk_used (J)	Tons of artillery/mortar ammo actually fired for smoke in current 30 minute segment by Blue (I=1-7; J=1-4).
		Q. R_asmk_used (I) R_msmk_used (J)	Tons of artillery/mortar ammo actually fired for smoke in current 30 minute segment by Red (I=1-7; J=1-4).

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Arty_arrive (continued)		R. Smoke_used	Total arty or mortar tonnage actually used to emplace smoke.
		S. Prep_time	The amount of time after start time that the attacker will begin preparation.
		T. Int_fire_time	Time the fire can begin in the battle.
		U. Prep_fire_time	The Int_fire_time, rounded off to the nearest 30 minutes.
		V. Bif_fired (I,J) Rif_fired (I,J)	Tons of ammo fired this 30 minute segment by weapon type I on mission J.
		W. B_dsarty_avail (I) R_dsarty_avail (I)	Tons of arty ammo available for CS arty support (I=1-7).
		X. B_dsmort_avail (I) R_dsmort_avail (I)	Tons of mortar ammo available for CS mortar support (I=1-4).
		Y. Tot_ds_avail	Total tons of artillery ammo available for CS artillery support.
		Z. Clgp_msns	Number of CLGP missions flown.
		AA. Gamp_msns	Number of GAMP missions flown.
		BB. Frac_arty (I)	Fraction of artillery ammo available that was fired (I = 1-7).
		CC. Tons_avail	Total tons of ammo available this 30 minute segment.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Arty_arrive (concluded)		DD. Arty_bound	Maximum percentage of artillery ammo that can be fired on this mission.
		EE. Mine_hit	Number of minefields encountered that had not been assessed before.
		FF. Ds_attempted (I)	Tons of artillery fired in an attempt to provide direct support (I=1-7).
		GG. Frac_mort (I)	Fraction of mortars available which were fired (I=1-4).
		HH. Mort_bound	Maximum percentage of mortar ammo that can be fired on this mission.
		II. Mo_attempted (I)	Tons of mortar fired in an attempt to provide mortar support (I=1-4).
Calc_movement	Calculates attacker movement distances during the designated 30 minute period.	A. Move_minutes	Minutes of unsuppressed advance movement.
		B. Mission	Pointer for which side is attacking.
		C. Phase	Phase of the battle.
		D. M_per_minute	Unsuppressed advance rate for this mission.
		E. Unsupp_advance	Distance of unsuppressed advance (= Move_minutes times M_per_minute).

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Calc_movement (continued)		F. Casualty_level	Level of casualties for previous 30 minutes.
		G. Prev30_r_eff	Level of Red effectiveness for previous 30 minutes.
		H. Prev30_b_eff	Level of Blue effectiveness for previous 30 minutes.
		I. Cas_suppr	Suppression from casualties.
		J. Tot_systems	The total number from 10 elements, of forces in contact for one side.
		K. Incoming_arty (I)	Amount of incoming artillery (I=1-7).
		L. Incoming_mlrs (I)	Amount of incoming MLRS (I=1-4).
		M. Incoming_mort (I)	Amount of incoming mortars (I=1-4).
		N. Arty_equiv	The 155mm mission equivalent of incoming artillery.
		O. Mort_equiv	The 155mm mission equivalent of incoming mortars.
		P. Mlrs_equiv	The 155mm mission equivalent of incoming MLRS.
		Q. Tot_incom_msn	Total amount of incoming mission (sum of Arty_equiv, Mort_equiv Mlrs_equiv).
		R. Arty_level	Level of the artillery.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Calc_movement (concluded)		S. Company_equiv	Number of company equivalents.
		T. Arty_suppr	Amount of artillery suppression.
		U. Tot_vehicles	Total number of vehicles.
		V. Atk_helos	Number of attack helicopters attacking this segment.
		W. Atk_helo_level	Attack level of helicopters.
		X. Atkhelo_suppr	Suppression level of helicopters.
		Y. Tot_move_suppr	Total level of suppression of movement (the sum of Las_suppr, Arty_suppr and Atk_helo_suppr).
Mine_encounter	Checks for mine activation	A. Bul_bch	= 1 bull tactic, pushing through forcefully, minimizing time. = 2 breach tactic, clearing passage-way, minimizing losses.
		B. Max_delay	Maximum time of delay.
Phasel_btl	Conducts the attrition assessment for the Phase 1 battle.	A. Blue_aty	Fireflag for Blue artillery (0= no Blue arty this 30 minutes; (1= Blue arty this 30 minutes).
		B. Red_aty	Fireflag for Red artillery.
		C. A_pct_fwd	Percentage of attackers forward.
		D. D_pct_fwd	Percentage of defenders forward.
		E. D_fp	Defender's percent of effectiveness.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Phasel_btl (continued)		F. Sys(I,J)	A work array passing J = 1-70 elements to the subroutines. Sys (1,J) = Initial number of Blue targets. Sys (2,J) = Final number of Blue targets. Sys (3,J) = Initial number of Red targets. Sys (4,J) = Final number of Red targets. Sys (5,J) = Initial number of Blue fires. Sys (6,J) = Final number of Red fires.
		G. Blue_vul (I)	Array containing Blue vulnerability.
		H. Red_vul (I)	Array containing Red vulnerability.
		I. Rng_band	Range band: = 1 engagement at 1 - 500 m. = 2 engagement at 1000 m. = 3 engagement at 1500 m. = 4 engagement at 2000 m.
		J. Terrain	Terrain type; dependent on the attacker's terrain.
		K. Num_bands	Number of range bands.
		L. P_def (I)	Array containing percent of elements defending (I = 1-70).

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Phasel_btl (continued)		M. Cell (1,J)	Array containing initial number of element J helicopters.
		Cell (2,J)	Array containing remaining number of element J helicopters.
		N. Arty	Pointer for artillery type.
		O. Time_step	Interval of time in this phase in minutes.
		P. Adside	Pointer to AD side (1 = Blue; 2 = Red).
		Q. Ad_ammo	Total tons of ammo for AD.
		R. Ad_helo	Number of AD helicopters.
		S. Sided	Pointer to side (1 = Blue; 2 = Red).
		T. Target (1,J)	Array containing initial number of target systems of type J.
		Target (2,J)	Array containing remaining number of target systems of type J.
		U. H_targ (I,J)	Array containing initial number of helicopter targets of type J (J = 1-70).

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Phase1_btl (concluded)		V. R_dmount	Number of Red dismounted infantry.
		W. Sys_helo (I,J)	Array containing element losses to helicopters (I = 1 - Blue killed; 2 - Red killed).
		X. R_ld_fact	Number of Red personnel in carrier.
		Y. B_ld_fact	Number of Blue personnel in carrier.
		Z. Inf_surv (I)	Number of mounted infantry which survived (I = 1-5).
		AA. B_ms	Pointer to current Blue mission.
		BB. B_dmount	Number of Blue dismounted infantry.
		CC. Gamp_fact (I)	Array containing percent of the 70 elements at which GAMP may fire.
		DD. Clgp_fact (I)	Array containing percent of the 70 elements at which CLGP may fire.
Phase2_btl	Assesses attrition rate in Phase 2, direct fire	A. Del_30	Delay flag for 30 minute delay.
		B. Red_fct	Red factor for element I, involving the percent of Red forces which are firers, and the change in that percent every 30 minutes.
		C. Blue_fct	Same as Red_fct, but for Blue.
		D. Red_f_t(*)	Array containing fraction of Red force which is a target for this 30 minutes.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Phase2_btl (concluded)		E. Blue_f_t(*)	Array containing fraction of Blue force which is a target for this 30 minutes.
		F. Rdf_mask(*)	Mask for Red elements in DF battle.
		G. R_ms	Pointer to current Red mission (value 1 or 2).
		H. Cur_bnd	Current band entering.
		I. Num_bands	Number of range bands.
		J. T_conflict	Infantry conflict time (hours).
Phase3_btl	Conducts attrition assessments for Phase 3, withdrawal.		
Phase_int	Initializes systems to zero	A. Sys_direct (I,J)	Array containing participants and results of direct fire battle. (I = 1 - Blue killed; 2 - Red killed. J = 1-70 elements.)
		B. Sys_pgm (I,J)	Array containing elements and results of pgm battle (J= 1 - 70 elements).
		C. Sys_inf(*)	Array containing elements of infantry battle.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Phase_int (concluded)		D. Sys_mine (I,J)	Array containing elements entering and surviving minefields. I = 1 - Blue elements under attack. 2 - Blue elements surviving 3 - Red elements under attack. 4 - Red elements surviving. J = 1 - 70 elements.
		E. Sys_arty (I,J)	Array containing elements in artillery battle. I = 1 - Blue elements under attack. 2 - Blue elements surviving 3 - Red elements under attack. 4 - Red elements surviving. J = 1 - 70 elements.
Updatek_v	Updates the killer-victim arrays for Blue and Red forces.		
Run_mine	Calculates the percent of force in the minefield	A. Mounted	Number of mounted artillery.
		B. Dismounted	Number of dismounted artillery.
Arty_sub	Calculates Blue and Red elements targetable. Calculates artillery losses for Red and Blue.	A. R_dismounted	Number of Red dismounted infantry.
		B. B_dismounted	Number of Blue dismounted infantry.
Df_cbt	Calculates Red and Blue losses due to DF combat. Updates amount of ammunition for Red and Blue DF.	A. Prnt_rg	Range at which present attackers must break.
		B. R_ammo_1st	Amount of Red ammo used.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Dbf_cbt (concluded)		C. B_ammo_1st	Amount of Blue ammo used.
		D. B_ammo_load	The load of ammo for Blue, involving the packed weight of the ammo and the number of Blue engagements.
		E. R_ammo_load	The load of ammo for Red, involving the packed weight of the ammo and the number of Red engagements.
		F. B_engagements (I)	Array containing the number of Blue engagements for direct fire element I (I = 1 - 20).
		G. R_engagements (I)	Array containing the number of Red engagements for direct fire element I (I = 1 - 20).
		H. B_inf_save (I) R_inf_save (I)	Array containing the number of infantry on direct fire carriers. (I= 1-5).
		I. B_fire_sv (I) R_fire_sv (I)	Contains initial number of fires from the work array Sys. (I = 1-70).
W_smoke	Establishes the dimensions of the desired screen. Calls the Smk_emp routine. Returns the visibility through the screen and the amount of ammo used to emplace the screen.	A. S_time	Amount of time the screen will last. It equals 1.1 times the maximum amount of time before the side will break.
		B. B_msmk_left	Amount of smoke left fired by Blue mortars.
		C. B_asmk_left	Amount of smoke left fired by Blue artillery.

Table 6-5. Subroutine table for ground combat/driver routine *(continued)*

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
W_smoke (concluded)		D. R_msmk_left	Amount of smoke left fired by Red mortars.
		C. R_asmk_left	Amount of smoke left fired by Red artillery.
Infantry_cbt	Calculates losses to Red and Blue infantry due to infantry combat.	A. Bstat (I)	1 = Blue mission; 2 = Red mission.
		B. Attacker	Pointer to attacker. 1 = Blue; 2 = Red.
		C. Lossblue	Blue infantry losses in this combat.
		D. LossRed	Red infantry losses in this combat.
		E. Sum_inf_b Sum_inf_r	Total number of small arms.
Clgp_gamp_atrit	Attrition of CLGP and GAMP elements.	A. N_rnds (I)	Array containing the number of rounds. I = 1 - number CLGP rounds. = 2 - number GAMP rounds.
		B. Sens_type (I)	Array containing sensor type. I = 1 - CLGP = 2 - GAMP
		C. Fir_type (I)	Array containing fire type. I = 1 - CLGP = 2 - GAMP
		D. C_t (I,J) C_targ (I,J)	Array containing the number of targets at which CLGP and GAMP may fire.

Table 6-5. Subroutine table for ground combat/driver routine (continued)

Functional area(s): Ground combat attrition program (P4)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Dump_input	Prints out information on Red and Blue units.		
Close_files	Closes the files to the main program.		

Section I. Minefield Attrition

1. PURPOSE.

The purpose of the DIME ground combat minefield attrition module is to calculate losses to 70 types of elements due to an encounter with a minefield in a sector battle.

2. GENERAL.

A. Minefield attrition is assessed for an attacking force encountering a minefield during phase I (pre-closure) and phase II (direct fire) of the sector battle.

B. The defender may emplace a maximum of three minefields per sector battle, simply by indicating the location (battle range) and the percent of the attacking force entering the minefield.

C. The minefield is emplaced using a ground-emplaced mine-scattering system (GEMSS) of two strips of mines each with a depth of 60 meters and a density of 0.007.

D. Two tactics may be used by the attacker in crossing the minefield in which the percentage of expected kills varies accordingly.

(1) Bull tactic, pushing through forcefully, minimizing time. This tactic occurs if the minefield is within direct fire range.

(2) Breach tactic, clearing passageway minimizing losses. This tactic occurs if the minefield is outside of direct fire range.

E. The formation of systems varies according to the phase of battle.

(1) Phase I (pre-closure). The systems entering have time to disperse and become less dense, forming more columns upon entering the minefield.

(2) Phase II (direct fire). The systems are more densely situated, forming fewer columns upon entering the minefield.

(3) Phase III (withdrawal). Minefield attrition does not occur in this phase.

3. **DATA FLOW.**

All needed information is received from the ground combat mainline except for the internal data of expected kills. The information flow is represented in Figure 6-14.

4. **FILE STRUCTURE.**

The only files associated with the minefield subroutine are held as internal data statements. These contain the percentage of expected losses to the attacker according to the bull/breach tactic and whether Red or Blue units are attacking.

5. **ALGORITHMS.**

A. Figure 6-15 shows a generalized logic flow of the processes in the ground combat mine attrition subroutine. Minefield attrition is figured by calculating the minefield coverage fraction, a column number for the force density, and then calculating the losses accordingly using the percentage of expected kills. The following paragraphs provide a more detailed description of the algorithms used in the attrition process.

(1) Calculate minefield coverage fraction. The width of the minefield and the sector width, as input at the beginning of the ground combat module, are used to calculate the coverage fraction as follows:

$$Mcf = Mw/Sw \qquad \text{(Eq. 6-18)}$$

where:

Mcf = minefield coverage fraction.
Mw = minefield width.
Sw = sector width.

(2) Calculate columns. Columns represent the deployment density of systems according to the current battle phase as discussed in paragraph 2E.

$$C = Totsys/N \qquad \text{(Eq. 6-19)}$$

where:

C = columns.
Totsys = total number of all systems entering minefield.
N = 3 for battle phase I .
6 for battle phases II & III .

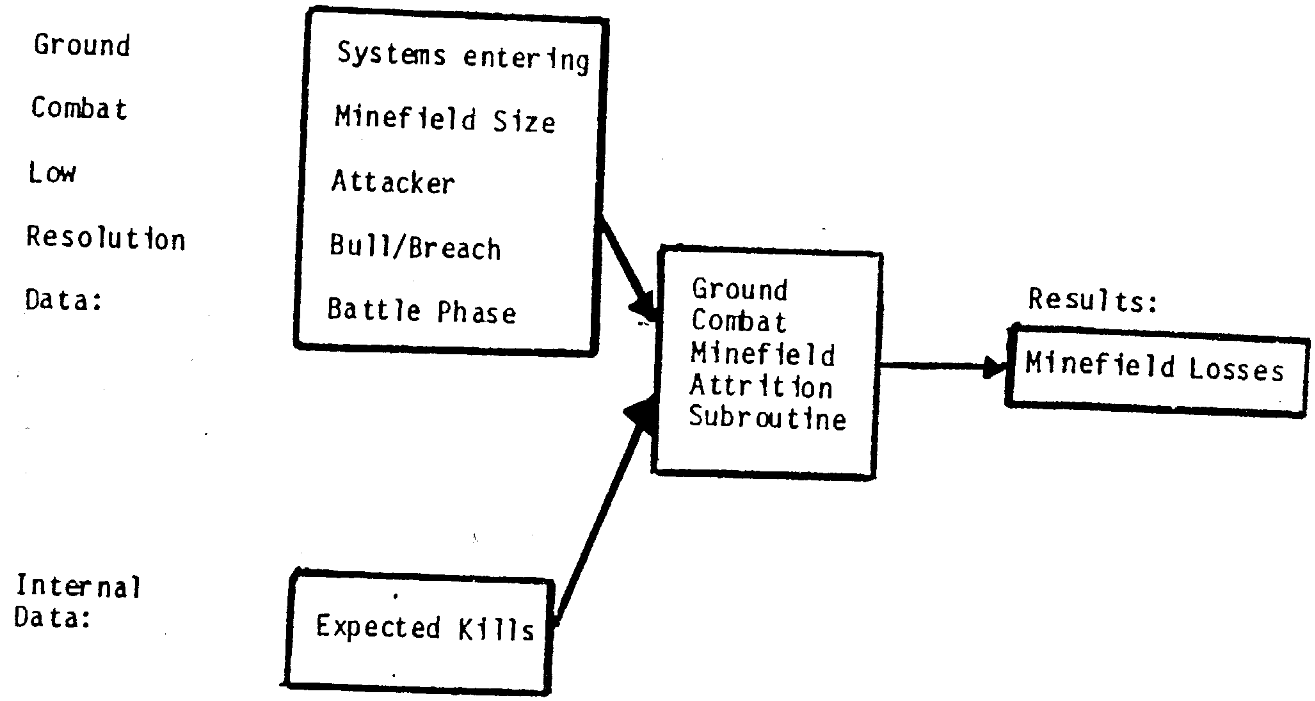


Figure 6-14. Minefield attrition information flow.

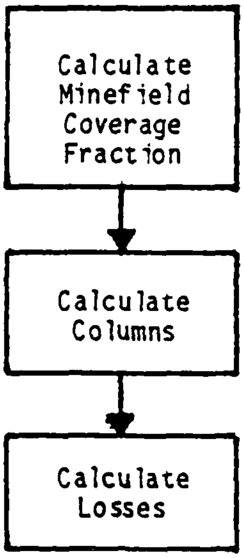


Figure 6-15. Minefield attrition logic flow.

(3) Calculate losses.

$$L_i = \frac{Mcf * Sys_i}{Totsys} * C * Eki \qquad \text{(Eq. 6-20)}$$

where:

L_i = number of losses.
Sysi = the number of type systems entering the minefield.
Eki = percentage of expected kills per system.

6. "UNITFILE" IMPACT.

This subroutine does not directly impact the unit status file ("UNITFILE"). Kills calculated in this subroutine are returned to the ground combat driver which in turn decrements all kills from the "UNITFILE".

7. CODE.

A. Introduction. This section contains information on the minefield attrition code. The functional areas discussed in the following paragraph are represented in Figure 6-16.

B. Mine attrition functional areas.

(1) The low resolution data are received from the ground combat mainline and consist of the systems entering the minefield, the minefield width and sector width, the attacker (Red/Blue force), whether the attacker will bull or breach the minefield, and the current battle phase.

(2) The array containing the number of kills is initialized to zero before any attrition is made.

(3) The low-resolution data and other minefield characteristics are checked. If inappropriate information has been passed, attrition calculations will not take place and an array of zero kills will be returned to the mainline. The appropriate characteristics should consist of:

(a) Minefield width not equal to zero.
(b) Sector width not equal to zero.
(c) Percentage of forces entering minefield not equal to zero.
(d) Minefield must be unused.
(e) Bull/breach flag equals 1 or 2.

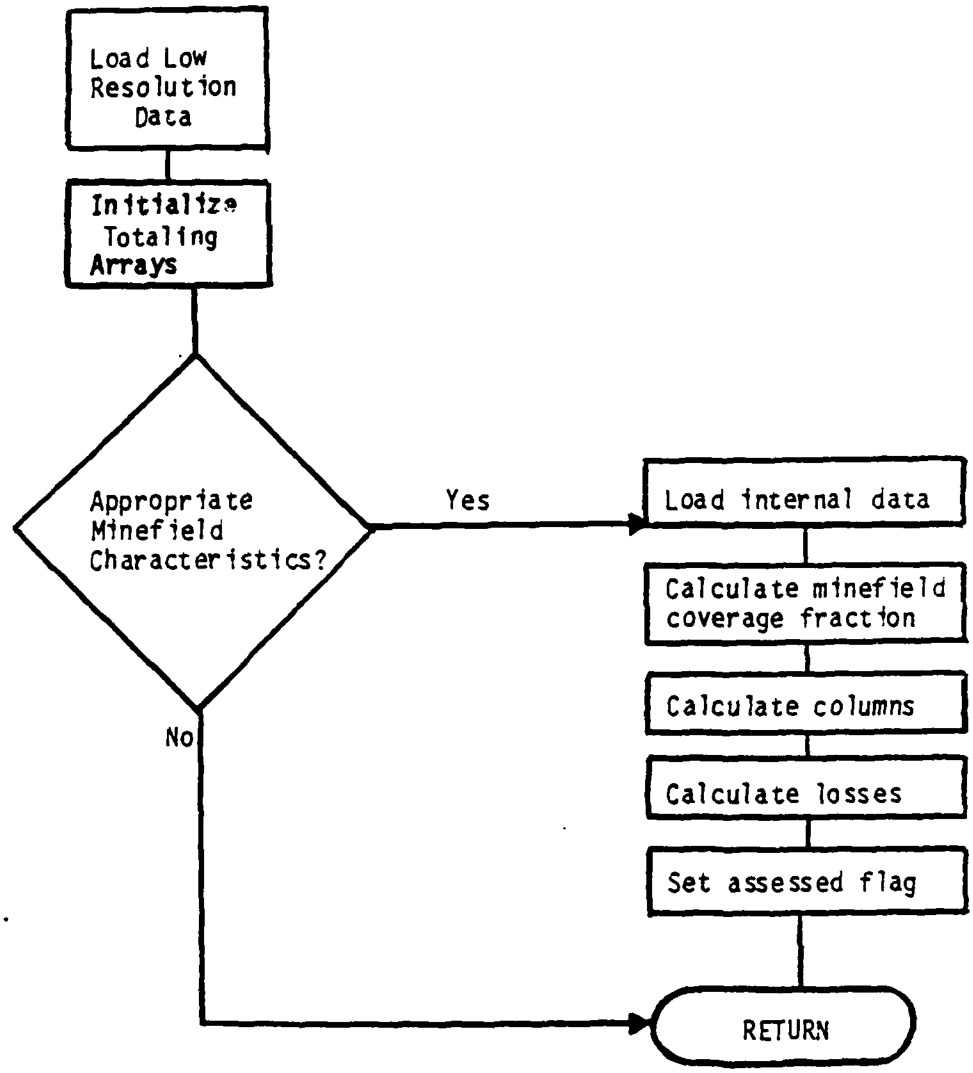


Figure 6-16. Minefield attrition functional flow.

(4) If the minefield characteristics are appropriate, the expected kill data are then loaded from internal data statements.

(5) The minefield coverage fraction is now calculated from the low-resolution data.

(6) The density of the forces is calculated according to the current battle phase which is referred to as the columns.

(7) Losses to the attacking force are then calculated using the expected kills, minefield coverage fraction, and columns per the 70 systems.

(8) A flag is then set to show that the minefield has been assessed, so as not to be used again.

(9) The number of kills per system is then returned to the ground combat mainline.

C. Primary variables. The primary variables of each functional area of the minefield attrition subroutine are shown in Table 6-6. Each variable is accompanied with a short description. Table 6-14 contains a code listing of the ground combat program.

Table 6-6. Minefield attrition subroutine table.

Functional area(s): Minefield Attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Mines	Mine subroutine.	A. Atk_def	Represents attacker: 0 = Red 1 = Blue
		B. Bul_bch	Bull/breach minefield tactic flag: 1 = Bull 2 = Breach
		C. Columns	Deployment density of systems according to current battle phase
		D. Mcf	Minefield coverage fraction
		E. Mine_frct (I,J)	Minefield expected kill fraction where: I = 1 - Blue bull 2 - Blue breach 3 - Red bull 4 - Red breach J = 1 - 70 weapon systems
		F. Mine_hit	Represents one of the possible three minefields currently being assessed

Table 6-6. Minefield attrition subroutine table (continued).

Functional area(s): Minefield Attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Mines (concluded)		G. Minefield (I,J)	Minefield information containing: I = 1 to 3, where 1 - First minefield 2 - Second minefield 3 - Third minefield J = 1 to 6, where 1 - Location of minefield in meters (range) 2 - Minefield width 3 - Sector width 4 - Percent of force entering minefield 5 - Vacant 6 - Assessed flag; 0 is not assessed, 1 is assessed
		H. Phase	Current battle phase 1 - Preclosure 2 - Direct fire 3 - Withdrawal
		I. Sys_mine (I,J)	Contains systems entering minefield and attrited systems. I = 1 to 4 where, 1 - Blue systems entering 2 - Blue systems attrited 3 - Red systems entering 4 - Red systems attrited J = 1 to 70 are the elements on the unit file

Section II. Artillery Attrition

1. PURPOSE.

The purpose of the DIME indirect fire subroutine, Arty_atrit, is to determine the losses to each of the 70 element types resulting from the delivery of artillery, rockets, multiple launch rocket systems (MLRS for Blue)/multiple rocket launchers (MRL for Red), and mortar munitions during the 30-minute interval.

2. GENERAL.

A. Interface with the main ground combat program. Arty_atrit is called from each of the three battle phase drivers discussed in the introduction of this chapter. Figure 6-17 provides an overview of the interface between the phase drivers and Arty_atrit. The interface subroutine, Arty_sub, prepares both the vehicular and personnel target arrays. It also maintains a counter of the successive number of 30-minute intervals that the targeted force has been subjected to indirect fire. Following the attrition calculations by Arty_atrit, Arty_sub posts the vehicular and personnel losses before returning to the battle phase driver.

B. Arty atrit structure. Since DIME is low-resolution, target acquisition is represented implicitly in two ways. First, indirect fire use is structured around five fire support tasks (prep, counterprep, close support, SEAD, and interdiction). Associated with each task is an implied target acquisition. Second, there are, in essence, four steps used in the assessment of attrition to indirect fire: set-up target area, determine elements targeted, determine ammunition fired, and assess damages.

(1) Set-up target area. The total area of the targeted force, input by the gamer, is reduced based upon the associated fire support task. This represents that portion of the total area occupied by the elements targeted for a specific fire support task, as well as the likelihood of acquiring them. This area is then further modified based upon the ability of the targeted force to disperse. The ability of the targeted force to disperse is determined by two factors: dispersion mask and whether the force is mounted (R=1) or dismounted (R=0). The dispersion mask is a 3 X 10 boolean array consisting of three battle phases and 10 missions. The dispersion mask may permit the force to increase its battle area radius if the given battle phase and mission are 1. If the force can disperse, the increase in the radius of the equivalent circle is 1200m if mounted (R=1) and 200m if dismounted (R=0) for a 30-minute timestep, thus diluting the density of the targeted elements. See Figure 6-18 for a graphical representation.

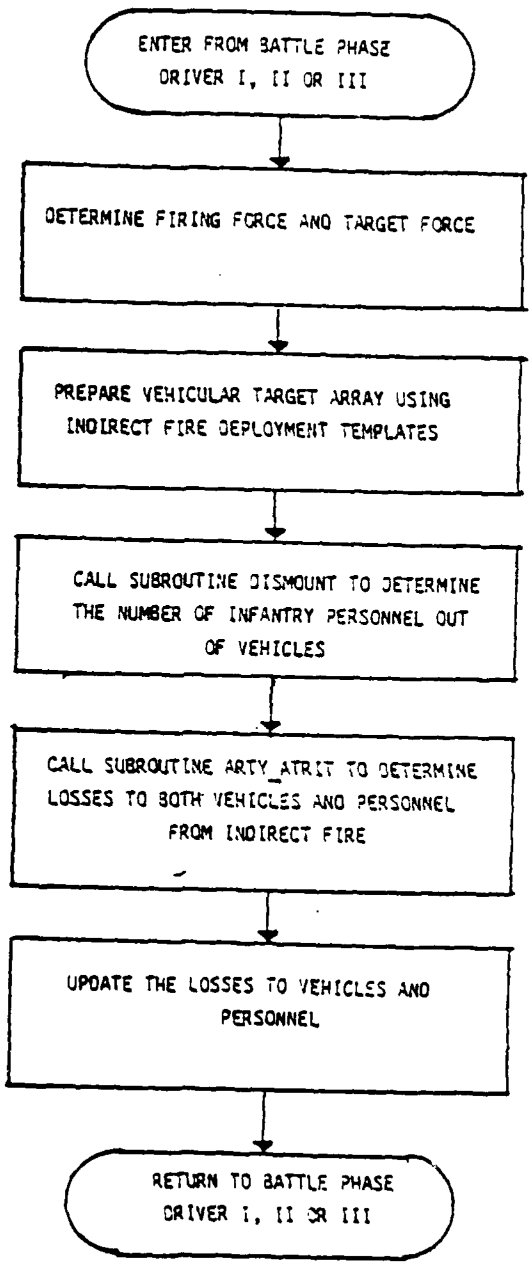


Figure 6-17. Logic flow of indirect fire subroutine Arty_sub (showing principal functions in calling the indirect fire attrition module Arty_atrit).

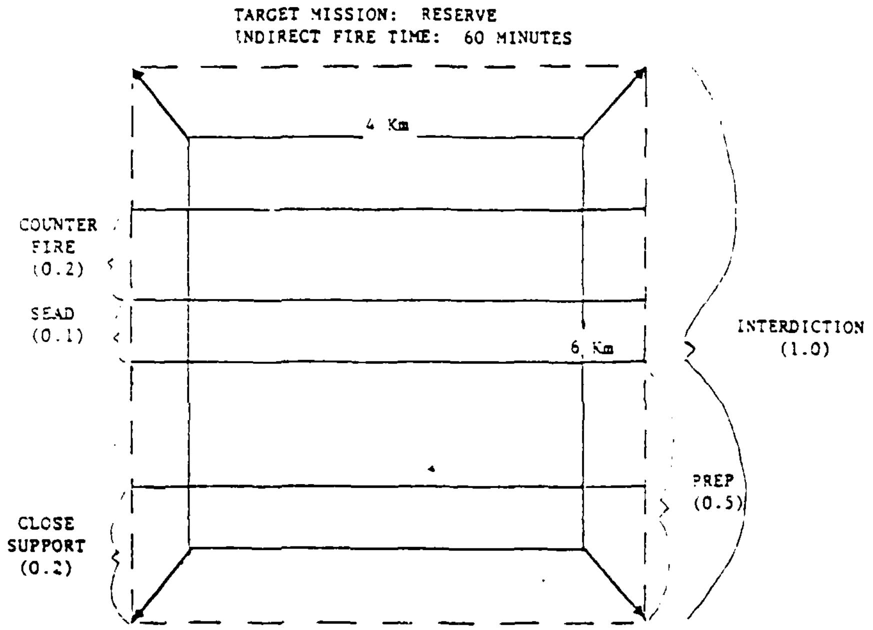


Figure 6-18. Schematic of notionalized artillery target. Shows bands for concentrating indirect fires in mission roles of prep, interdiction, close support, suppression of air defense and counterfire.

(2) Determine elements targeted. Though the quantity of each target element is passed into Arty_atrit in the Sys_arty(*) array, it is further modified to allow for selective targeting for each fire support task.

(3) Determine ammunition fired. The weight of ammunition fired is passed into Arty_atrit in the Bif_fired(*) and Rif_fired(*) arrays. It is then converted into the number of standard fire unit vollies fired.

(4) Assess damages. The majority of Arty_atrit deals with this function. Given the items discussed above, it determines the losses to the targeted force in each of the 70 elements using the methodology given in the Joint Technical Coordinating Group for Munitions Effectiveness (JTCG/ME) document and the Super Quickie II model.

3. DATA FLOW.

The Arty_atrit subroutine has three basic sources of data: the DIME host, munitions description from external files and target profile from external data structures. See Figure 6-19 for a graphical presentation.

A. DIME host data. These data are passed to Arty_atrit as calling arguments. They include:

(1) Tonnages of ammunition expended by artillery, rockets, and mortars for both Red and Blue.

(2) Target elements potentially available to be attacked by indirect fire for both Red and Blue.

(3) Target dimensions of entire force engaged in the battle for both Red and Blue.

(4) Mission and battle phase of the target unit for both Red and Blue.

(5) Cumulative time under indirect fire attack for both Red and Blue.

B. Target profile. These data are structured externally and include:

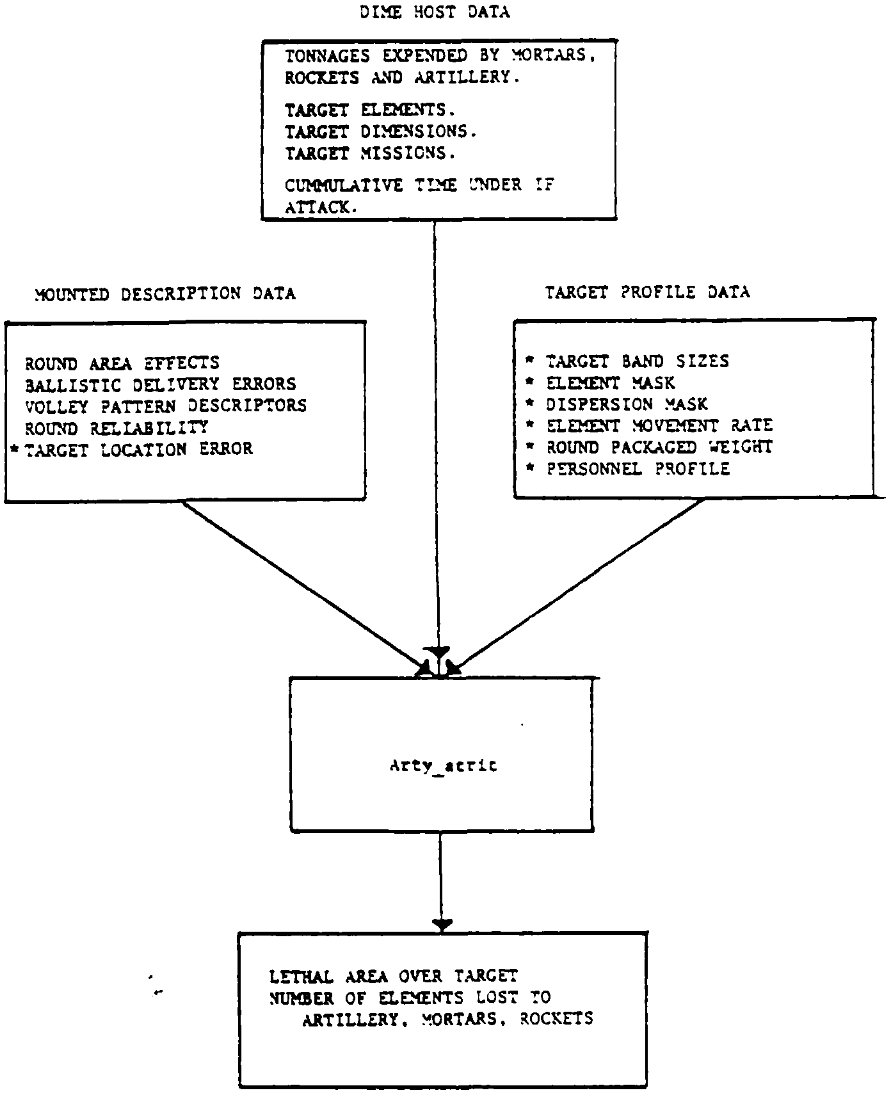
(1) Target band sizes for each fire support task.

(2) Dispersion mask for each phase and mission of the targeted force.

(3) Target mask array for each fire support task and targeted element, thus allowing selective targeting for each task.

(4) Round packaged weight (in short tons).

(5) Personnel (infantry) profile which permits the hardening of this element over time.



* Indicates internal data structure.

Figure 6-19. Data flow for the indirect fire attrition module, Arty_atrit.

C. Munitions descriptions. These are accessed from external files and include:

(1) Artillery, rocket, and mortar round (or submunition) effects against each of 72 target elements. It should be noted that personnel (infantry) are further divided into prone and prone-protected postures.

(2) Ballistic delivery error for each type of fire support and system (artillery, rockets, and mortars) for both Red and Blue at some fixed range (normally 1/2 or 2/3 maximum range). These include precision mean point of impact (MPI) errors.

(3) Volley pattern descriptors which describe the standard volley pattern of the fire unit for each type of fire support system for Red and Blue.

(4) Round reliability.

(5) Target location error (TLE) which represents the error associated with a generic target acquisition device for each fire support task (e.g., forward observer for close support).

4. FILE STRUCTURE.

Two sets of external files are used by Arty_atrit: lethal areas and fire delivery parameters.

A. Fire delivery files. Each file consists of fifteen records, seven for artillery, four for rockets and four for mortars. Each record includes:

Index	Description
1	Length of damage pattern.
2	Length of damage pattern factor which is determined by the angle of fall for the range used.
3	Probability of damage factor determined by number of submunitions, submunition reliability, length, and width of submunition pattern.
4	Precision error range for range used.
5	Precision error in deflection for range used.
6	Length of volley factor for one fire unit volley in range.

Index	Description
7	Width of volley factor for one fire unit volley in deflection.
8	Individual round reliability.
9	MPI error ir deflection for range used.
10	MPI error in range for range used.

B. Lethal area files. Each file, dependent on Red or Blue, consists of fifteen records, seven for artillery, four for rockets, and four for mortars. Each record contains 72 items, one for each of 70 weapon elements plus one for prone personnel and one for protected prone personnel.

5. ALGORITHMS.

The primary algorithm used in Arty_atrit is the Super Quickie II. This algorithm requires a definition of the target area elements targeted, and volleys fired. These are used to calculate the losses to targeted elements due to indirect fire.

$$Ps_i = \prod_{j=1}^{5} (1 - Fd_{ij}) \qquad \text{(Eq. 6-21a)}$$

where:

Ps_i = probability of survival of element is from direct fire.
i = target element
j = fire support task
Fd_{ij} = calculated as follows:

$$Fd_{ij} = Ecr_{ij} * Ecd_{ij} * Pnv_{ij} \qquad \text{(Eq. 6-21b)}$$

where:

Fd_{ij} = expected fractional damage to target element i from indirect fire support task j.
Ecr_{ij} = expected fraction of target covered by pattern in range (see Eq. 6-23).

Ecd_{ij} = expected fraction of target covered by pattern in deflection (see Eq. 6-25).
Pnv_{ij} = calculated as follows:

$$Pnv_{ij} = 1 - \left[1 - (Al_i * Nr * Rr) / (Avp_i * Of_i) \right]^{Vollies_j * Of_j} \quad \text{(Eq. 6-22)}$$

where:

Pnv_{ij} = the probability of damage within pattern for vollies; fired by an indirect fire type for a specific task.
Al_i = the lethal area of the complete round against target element i.
Nr = the number of rounds in the volley.
Rr = the reliability of the round.
Avpi = volley damage pattern area.
Of_i = the volley overlap factor.

The expected fraction of target covered by pattern in range (used in Eq. 6-21b) is calculated as follows:

$$Ecr_{ij} = 2.96 * Reptm_j * Fl_{ij} / Lt_j \quad \text{(Eq. 6-23)}$$

where:

$Reptm_j$ = total mean point of impact error in range for fire support task j.
Lt_j = target length for task j.
Fl_{ij} = calculated as follows:

$$Fl_{ij} = Fla_{ij} - Flb_{ij} \quad \text{(Eq. 6-24a)}$$

where:

$$Fla_{ij} = (Al_{ij}/2) \left[1 - e^{-.63 * Al_{ij}^2} \right]^{1/2} +$$

$$\left[e^{-Al_{ij}^2 / 2} \Big/ (2\pi)^{1/2} \right] \quad \text{(Eq. 6-24b)}$$

$$Flb_{ij} = (A2_{ij}/2)\left[1 - e^{-.63 * A2_{ij}^{2}}\right]^{1/2} +$$

$$\left[e^{- A2_{ij}^{2} / 2} \Big/ (2\pi)^{1/2}\right] \qquad \text{(Eq. 6-24c)}$$

and,

$$Al_{ij} = (Lvp_i + Lt_j) / (2.96 * Reptm_j) \qquad \text{(Eq. 6-24d)}$$

$$A2_{ij} = |Lvp_i - Lt_j| / (2.96 * Reptm_j) \qquad \text{(Eq. 6-24e)}$$

where:

Lvpj = the length of the volley pattern of the standard fire unit.

The expected fraction of target covered by pattern in deflection (used in Eq. 6-21b) is calculated as follows:

$$Ecd_{ij} = 2.96 * Deptm_j * Fw_{ij}/Wt_j \qquad \text{(Eq. 6-25)}$$

where:

$Deptm_j$ = total mean point of impact error in deflection.
Wt_j = target width for task j.
Fw_{ij} = calculated as follows:

$$Fw_{ij} = Fwa_{ij} - Fwb_{ij} \qquad \text{(Eq. 6-26a)}$$

where:

$$Fwa_{ij} = (B1_{ij}/2)\left[1 - e^{-.63 * B1_{ij}^{2}}\right] + \left[e^{-B1_{ij}^{2}/2} \Big/ (2\pi)^{1/2}\right] \qquad \text{(Eq. 6-26b)}$$

$$Fwb_{ij} = (B2_{ij}/2)\left[1 - e^{-.63 * B2_{ij}^{2}}\right] + \left[e^{-B2_{ij}^{2}/2} \Big/ (2\pi)^{1/2}\right] \qquad \text{(Eq. 6-26c)}$$

and,

$$B1_{ij} = (Wvp_i + Wt_j) / (2.96 * Deptm_j) \qquad \text{(Eq. 6-26d)}$$

$$B2_{ij} = | Wvp_i - Wt_j | / (2.96 * Deptm_j) \qquad \text{(Eq. 6-26e)}$$

where:

$Wvpj$ = the width of the volley pattern of the standard fire unit.

6. "UNITFILE" IMPACT.

Arty_atrit does not directly affect the "UNITFILE"; rather it returns to the main battle devices the losses to each of the 70 elements due to indirect fire.

7. CODE.

Figure 6-20 represents the functional flow of Arty_atrit. The subroutine is divided into two separate (but identical) sections: Blue firing on Red and Red firing on Blue. Each section has the four portions discussed in paragraph 2B above.

A. Calculate target area. Arty_atrit begins processing by calculating the total area that each of five fire support tasks are targeted against. These tasks are: preparatory fires, close support, suppression of air defense (SEAD), counterfire, and interdiction. One of two equations is used to determine target area. The equation selected is dependent upon the target's ability to disperse. The ability to disperse is in turn contingent upon the battle phase and the mission of the target.

(1) Target area excluding dispersion factor.

$$A = L * W * B \qquad \text{(Eq. 6-27)}$$

where:

A = total area that the indirect fire task is targeted against.
L = total length in meters of the target area in the range direction.
W = dimension (total width in meters) of the target area in the deflection direction.
B = percentage of total area that indirect fire task is targeted against.

(2) Target area including dispersion factor. When a target is allowed to disperse, it is necessary to first calculate the increase in target area due to dispersion and the original radius of the target area before determining the final target area for each of the indirect fire tasks.

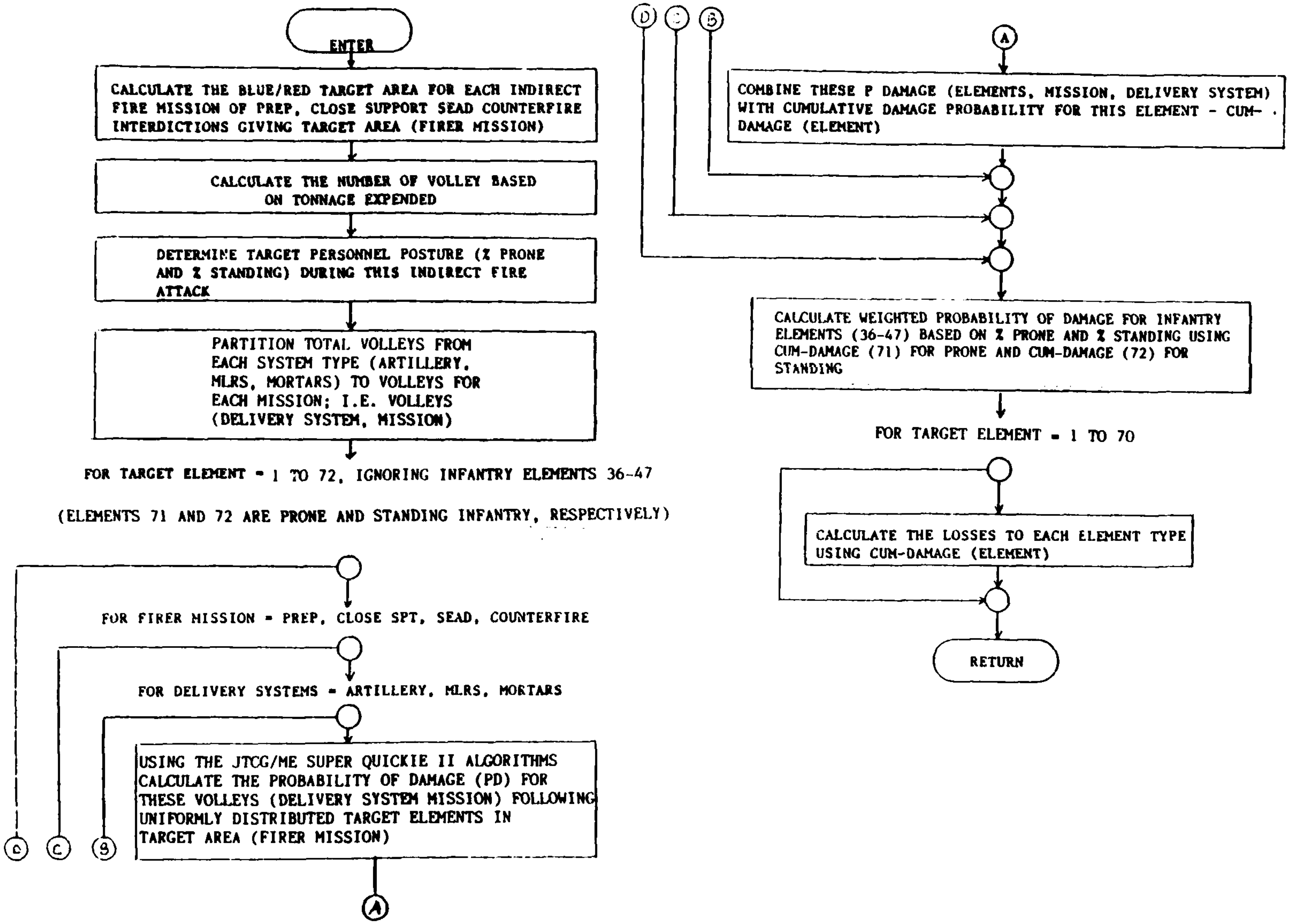


Figure 6-20. Generalized functional flow of Arty_atrit

(a) Calculate the increase in the target radius.

$$Ir = M * (F - 1) \qquad \text{(Eq. 6-28)}$$

where:

Ir = increase in target area due to target dispersion in meters.

M = meters moved per 30-minute interval where:
1000 if a Red target is mounted and attacking
1200 if a Blue target is mounted and attacking
200 if either a Red or Blue target is dismounted or defending.

F-1 = number of consecutive 30-minute intervals less one in which the target has been receiving indirect fire.

(b) Calculate radius of original target area.

$$R = (A/PI)^{.5} \qquad \text{(Eq. 6-29)}$$

where:

R = radius of original target area.
A = area that indirect fire task is targeted against excluding the dispersion factor (see Eq. 6-27).
PI = the symbol designating the ratio of the circumference of a circle to its diameter; approximately 3.1415927.

(c) Calculate target area including the dispersion factor.

$$Ad = PI * (R + Ir)^2 \qquad \text{(Eq. 6-30)}$$

where:

Ad = area that the indirect fire task is targeted against including the dispersion factor.
R = radius of original target area in meters, as calculated in Eq. 6-29 above.
Ir = increase in target area due to target dispersion, in meters, as calculated in Eq. 6-28 above.

B. <u>Calculate volleys to fire.</u> The number of volleys fired is calculated for each combination of the three indirect fire types (artillery, MLRS/MRL, mortars) and the five indirect fire tasks (preparatory fire, close support, SEAD, counterfire, interdiction). The type of indirect fire determines the number of tubes fired in the volley as well as the individual round or launcher load packaged weight (in tons).

(1) The weight of all ammunition fired in a single volley is determined by:

$$W_f = T * Rw_f \qquad \text{(Eq. 6-31)}$$

where:

- W_f = weight of ammunition, in tons, fired in a single volley by weapon type f.
- T = number of tubes firing in each volley.
- Rw_f = individual round or launcher load packaged weight, in tons,of weapon type f.

(2) The number of volleys to fire is computed from the following equations.

$$V_f = A_f/W_f \qquad \text{(Eq. 6-32)}$$

where:

- V_f = number of volleys fired by weapon type f under a specific firing task where:
  f = Type:
  1 Artillery
  2 MLRS/MRL
  3 Mortars
- A_f = total number of tons of ammunition fired by weapon type f under a specific firing task.
- W_f = weight of ammunition, in tons, fired in a single volley as calculated in Eq. 6-31.

C. Calculate fractional damage and total probability of survival indirect fire losses.

(1) The expected fractional damage for each DIME element, excluding the VIPER/infantry and for personnel in both standing and prone positions, is computed by Arty_atrit. The equations determining fractional damage are set forth in paragraph 5. above.

(2) The fractional damages computed by Arty_atrit are conditioned by the targeted force (Red or Blue) and the type of indirect fire (artillery, rockets, mortars). The fractional damage is used to determine losses to both Red and Blue from indirect fire and is passed back to the battle drivers.

D. Subroutine table and primary variables are listed in Table 6-7. Table 6-14 contains a listing of the ground combat code.

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main	Sets up targets for artillery fire.	A. Area_band (I)	Percentage of total area that task I (1-5) is targeted against.
		B. T_length(I)	The dimension (total length in meters) of the target area in the deflection direction.
		C. T_width(I)	The dimension (total width in meters) of the target area in the deflection direction.
		D. Area (I)	Area that task I is targeted against (square meters).
		E. Dispersion_mask (I,J)	A switch (0 or 1) determining if target can disperse when receiving fire on mission J, phase I.
		F. Increase_radius	Amount of radius the target area is increased (meters) due to target dispersion.
		G. Radius	Radius (meters) of original target area.
		H. Length (I)	Length (meters) of new target area after possible dispersion for task I.
		I. Width (I)	Width (meters) of new target area after possible dispersion for task I.

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main (continued)		J. B_tgt_mask (I,J) R_tgt_mask (I,J)	Target mask (0 or 1) for task I and target element J (1-72).
		K. Al (I,J)	Lethal area (square meters) of weapon type I (1-15) against target type J: 1 - 70 = DIME elements 71 = Infantry standing 72 = Infantry prone
		L. Ps (I)	Joint probability of survival of element I (1-72) against all indirect fire elements.
		M. Tubes_per_vol	Number of tubes firing per volley.
		N. Throw_wt	Weight (tons) of ammo fired per volley.
		O. Volleys (I,J)	Number of vollies to fire by weapon type I (1-15) under task J (1-5).
		P. Bif_fired (I,J)	Total number of tons of ammo fired by Blue weapon type I on task J.
		Q. Psnl_posture (I,J)	Percent of personnel in prone position for: I = 1 - Defense = 2 - Attack J = 1 - First volley = 2 - Subsequent volley.
		R. Posture	Percent of personnel currently in prone position.
		S. Barty_fire	Number of consecutive 30-minute intervals in which Blue uses indirect fire on Red.

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main (continued)		T. Rarty_fire	Number of consecutive 30-minute intervals in which Red uses indirect fire on Blue.
		U. Rd_wt (I)	Individual round or launcher load packaged weight (tons) of weapon type I (1-15).
		V. Sys_tot (I,J)	I = 1 - initial number of Blue element J (1-70) = 2 - current number of Blue element J = 3 - initial number of Red element J = 4 - current number of Red element J
		W. Sys_arty(I,J)	I = 1 - initial number of Blue element J (1-70) targetable = 2 - current number of Blue element J targetable = 3 - initial number of Red element J targetable = 4 - current number of Red element J targetable
		X. B_afire	Number of consecutive 30-minute intervals in which Blue uses indirect fire on Red.
		Y. R_afire	Number of consecutive 30-minute intervals in which Red uses indirect fire on Blue.
		Z. R	Flag for whether infantry is mounted.
		AA. Type	Loop counter indexing type of IF. 1 = Artillery 2 = MLRS 3 = Mortars.

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main (concluded)		BB. Task	Loop counter indexing one of 5 tasks.
		CC. Atk_def	0 = Blue attacking, Red defending 1 = Red attacking, Blue defending
		DD. Attacker	1 = Blue attacking 2 = Red attacking.
		EE. B_msn	Pointer to Blue mission.
		FF. R_msn	Pointer to Red mission.
		GG. B_phase	Pointer to Blue phase.
		HH. Element	Loop counter for target type.
		II. Index	Loop counter for target type.
		JJ. L_vp	Total length (meters) of the adjusted volley pattern in the range direction.
		KK. Rif_fired(I,J)	Array containing total number of tons of ammo fired by Red weapon type I for task J.
		LL. Del_par (I,J)	Array containing information from an external file which is assigned to fire delivery variables.

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Compute	Computes the probability of survival for a weapon system to go into the Ps(*) array. Computes for Blue firing on Red.	A. F_d (I,J)	Designed or expected fractional damage to element J (1-72) under task I (1-5).
		B. N_r	Number of rounds per volley.
		C. L_v	Total length (meters) of the volley pattern in the range direction.
		D. W_v	Total width (meters) of the volley pattern in the deflection direction.
		E. W_{dp}	Width (meters) of the damage pattern of a single round in the deflection direction.
		F. L_{dp}	Length (meters) of the damage pattern of a single round in the range direction.
		G. L_{dp_F}	L_{dp} factor. Determined by the angle of fall for the range selected.
		H. P_{dp}	Probability of damage within a single round pattern.
		I. P_{dp_f}	P_{dp} factor. Determined by the number of submunitions, submunitions reliability and single round submunition pattern in the range direction.
		J. A_{el}	Lethal area (square meters) of target damage per single round. Determined by length and width of damage pattern and probability of damage within single round pattern.

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Compute (concluded)		K. L_v_fac	Length of the volley pattern in the range direction for a single volley pattern.
		L. W_v_fac	Width of the volley pattern in the deflection direction of a single volley round pattern.
Compute_red	Computes the probability of survival for a weapon system to go into the Ps(*) array. Computes for Red firing on Blue.		
Calculate_fd	Calculates the expected fractional damage (used to compute the probability of survival).	A. L_ap	Adjusted damage pattern (meters) of a single round in the range direction.
		B. W_ap	Adjusted damage pattern (meters) of a single round in the deflection direction.
		C. A_vp	Volley pattern damage area.
		D. A_ap	Single round adjusted damage area.
		E. Of	Overlap factor. Determined by the number of rounds in each volley, the adjusted damage pattern and the volley damage pattern.
		F. Tle(I)	Target location error for task I.

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Calculate_fd (continued)		G. Rep_m	Mean point of impact probable error in range direction (excluding target location error).
		H. Dep_mac	Mean point of impact probable error the deflection direction (excluding target location error).
		I. P_nv	Probability of damage for a number of vollies. Used as a factor in the expected fractional damage, F_d(*.*).
		J. Rep_tm	Mean point of impact probable error in range direction (including target location error).
		K. Dep_tm	Mean point of impact probable error in deflection direction (including target location error).
		L. R_r	Reliability of the round.
		M. A_1	Factor used to determine F_1a. Determined by the length of volley damage, the length of the new target area after dispersion and the probable error in range.
		N. A_2	Factor used to determine F_1b. Determined by the length of volley damage, the length of the new target area after dispersion and the probable error in range.
		O. F_1a	Positive factor in F_1. Determined by A_1.

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Calculate_fd (continued)		P. F_1b	Negative factor in F_1. Determined by A_2.
		Q. F_1	Factor in the expected fraction of target covered by pattern in range direction, Ec_r.
		R. Ec_r	Expected fraction of target covered by pattern in range direction. Used to determine the expected fractional damage, F_d (*.*).
		S. B_1	Factor used to determine F_wa. Determined by the width of volley damage, the width of the new target area after possible dispersion and probable error in dispersion.
		T. B_2	Factor used to determine F_wb. Determined by the width of volley damage, the width of the new target area after possible dispersion and probable error in dispersion.
		U. F_wa	Positive factor in F_w. Determined by B_1.
		V. F_wb	Positive factor in F_w. Determined by B_2.
		W. F_w	Factor in the expected fraction of target covered by pattern in the deflection direction.
		X. Ec_d	Expected fraction of target covered by pattern in deflection direction. Used to determine the expected fractional damage, F_d (*.*).

Table 6-7. Artillery subroutine table.

Functional area(s): Arty atrit: Artillery attrition

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Calculate_fd (concluded)		Y. Kdep_p	The pattern adjustment factor. The precision probable error in deflection (excluding target location error).
		Z. Krep_p	The pattern adjustment factor. The precision probable error in range (excluding target location error).
		AA. W_vp	Total width (meters) of the adjusted volley pattern in the deflection direction.
Re_read	Assigns values to fire delivery variables.		
Sub_end	Closes the files for the subroutine.		

Section III. Smoke

1. PURPOSE.

The purpose of the smoke module is to return the percent of frontage visible through a screen to a firer using optic, crew-served, and thermal sensors.

2. GENERAL.

The DIME smoke module provides coverage for the withdrawing force. The withdrawing force can request mortar and/or artillery-delivered smoke, provided smoke ammunition has been allocated. The smoke module returns the percent of frontage visible through a screen to a firer using optic, crew-served, and thermal sensors and the tonnage used to emplace the screen which is subtracted from the withdrawing force's stockpile. The percent of frontage visible is used by the direct fire attrition module of DIME to determine the total number of systems available to be engaged as targets for the withdrawal phase (phase III) of the DIME battle. In turn, these targets are used to determine the attrition suffered by both forces.

3. DATA FLOW.

The smoke module is a portion of the ground combat attrition program. The data flow and the logic flow involve passing information through four routines. Figure 6-21 indicates the data flow through the four smoke subroutines. In addition to the input and/or output parameters, smoke ammunition weight in pounds per one round and the rate of fire used to build the screen are contained in external data files. Additional data is accessed from an auxiliary file containing the probabilities of detection through specific screens. This file will be discussed in greater detail in paragraph 4. The following discussion will include the data flow as it pertains to the general flow of the DIME smoke module.

A. The DIME smoke module employs a four-step methodology:

(1) Provides coverage for the withdrawing force.

(2) Provide mortar and/or artillery-emplaced smokescreen.

(3) Returns the percent of frontage visible to a firer using optics, crew-served, and thermal sensors.

(4) Depletes the ammunition stockpile of the force emplacing the screen.

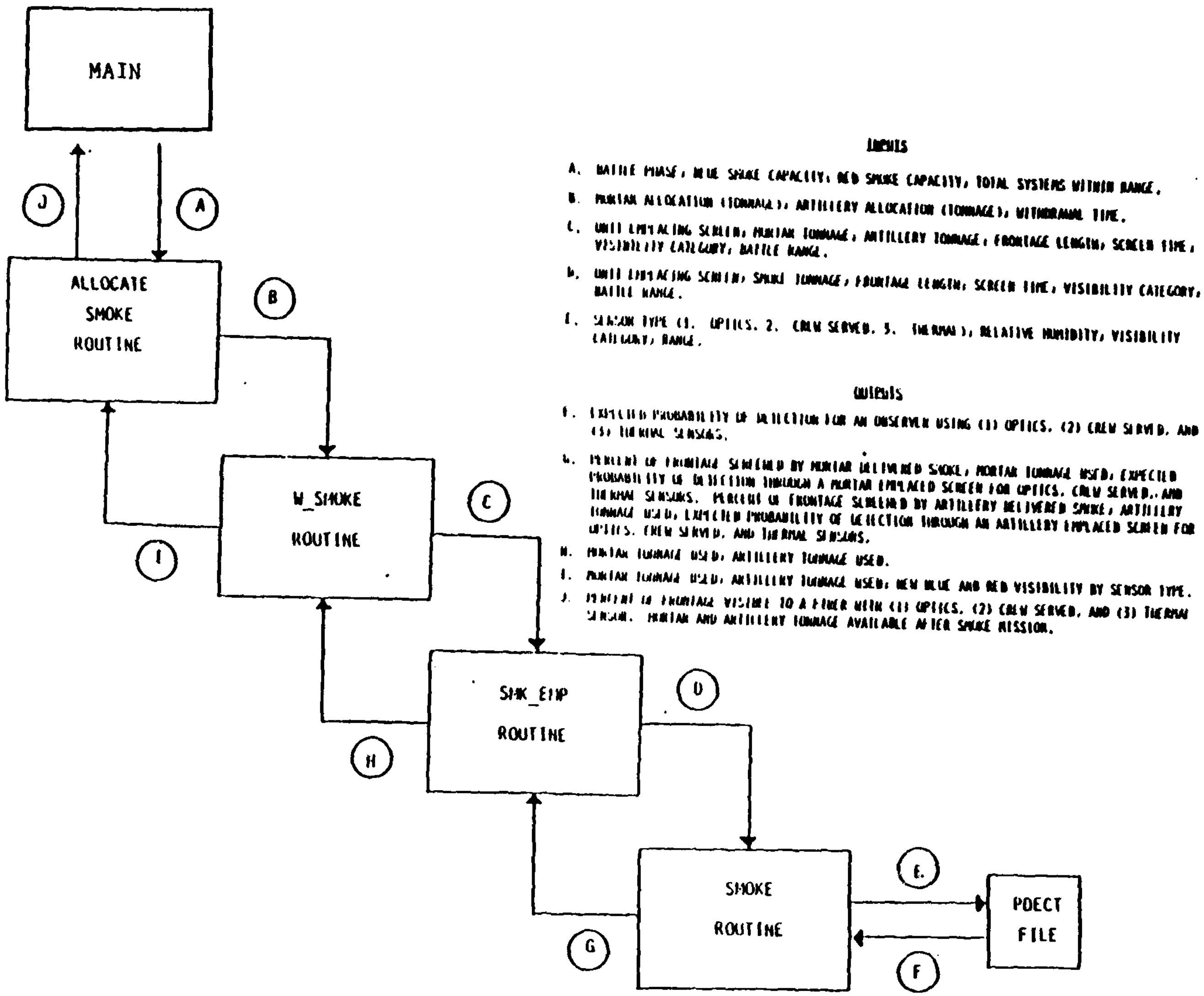


Figure 6-21. Data flow for smoke routines

B. The DIME smoke module divides these four steps into four routines: Allocate_smoke, W_smoke, Smk_emp, and Smoke.

(1) During the withdrawal phase of the battle, DIME calls the Allocate_smoke routine to determine if a percent of the close support ammunition has been allocated for smoke. If an allocation has been determined, the tonnage of ammunition is forwarded to the W_smoke routine as mortar tonnage and artillery tonnage designated for smoke.

(2) These tonnages, along with the range between the units, the visibility category, the unit's width, and the withdrawal time, are used by the W_smoke routine to determine the dimensions of the smokescreen.

(3) Once the dimensions of the screen have been determined, this information is passed to the Smk_emp routine which will attempt to emplace a screen with mortar-delivered smoke. The screen's length is determined from the unit's width passed in by the W_smoke routine. If the mortars cannot produce a screen which completely covers the unit, then the Smk_emp routine will complete the desired screen by emplacing the remainder of the screen with artillery-delivered_smoke.

(4) Once the screen is emplaced, the Smk_emp routine calls the Smoke routine to determine the probability of detection through "holes" in the screen as a function of the smoke round, relative humidity, visibility category, and the range between units for an observer using an optic, crew-served, or thermal sensor.

(5) Using these probabilities of detection, the Smk_emp routine determines the percent of frontage a firer can see through "holes" in the screen, plus any unsmoked frontage, as a function of sensor type. In addition, Smk_emp calculates the mortar tonnage and/or artillery tonnage used to emplace the screen.

(6) The direct fire attrition module uses the percent of frontage visible by sensor type to determine the total number of targets available to be engaged during the withdrawal phase.

4. FILE STRUCTURE.

The smoke module accesses an auxiliary file containing the probabilities of detection through specific screens.

A. The probability of detection values are stored on a file consisting of six records.

(1) Record 1 contains the probability of detection values for an observer using an optic sensor in relative humidity of less than 50 percent for four visibility categories (1 km, 2 km, 4 km, > 4 km) and in seven range categories (0 - .5 km, .5 - 1.0 km, 1.0 - 1.5 km, 1.5 - 2.0 km, 2.0 - 2.5 km, 2.5 - 3.0 km, 3.0 - 3.5 km).

(2) Record 2 contains the probability of detection values for an observer using an optic sensor in relative humidity of 50 percent or greater for four visibility categories and seven range categories.

(3) Record 3 and record 4 contain the probability of detection values for an observer using a crew-served sensor.

(4) Record 5 and record 6 contain the probability of detection values for an observer using a thermal sensor.

B. The probability of detection values are stored on five files: three for mortar-delivered and two for artillery-delivered screens emplaced using white phosphorous (WP) smoke.

(1) Blue mortars,

(a) 107mm

(b) 181mm.

(2) Blue artillery, 155mm.

(3) Red mortar, 120mm.

(4) Red artillery, 152mm.

5. **ALGORITHMS.**

The smoke module uses two major algorithms: calculation of the percent of frontage visible to a firer, as a function of sensor type and smoke round; and calculation of total number of systems available to be engaged as targets in the battle.

A. The algorithm used to calculate the percent of frontage visible to a firer, as a function of sensor type and smoke round, uses the following formula:

$$Psee_s = (1 - Tcov) + \left[\sum_{r=1}^{nr} (Pdect_{sr}) * Pftcov_r \right] \qquad \text{(Eq. 6-33)}$$

where:

Psees = percent of frontage visible to firer using sensor type s.
$Pdect_{sr}$ = expected probability of detection through a screen, as a function of sensor type and smoke round.

$Pftcov_r$ = percent of frontage covered by a screen, as a function of the smoke round used to emplace the screen.
Tcov = total percent of frontage covered by all smoke rounds.
nr = number of smoke rounds used to emplace the screen.

B. The algorithm used in the direct fire module to calculate the total number of systems available to be engaged as targets in the battle uses the following formula:

$$Tsystems = \sum_{s=1}^{70} Twpns_s * Psee_s \qquad \text{(Eq. 6-34)}$$

where:

Tsystems = total number of systems available to be engaged as targets in battle.
$Twpns_s$ = total number of systems within range.
$Psee_s$ = percent of frontage visible to a firer, as a function of sensor type and smoke round.

6. "UNITFILE" IMPACT.

The smoke module has no direct impact on the "UNITFILE". For the indirect impact discussion, refer to the ammunition allocation discussion in the introduction to this chapter.

7. CODE.

The smoke module code is contained as a submodule in the ground combat attrition program. The smoke module code is divided into four subroutines: Allocate_smoke, W_smoke, Smk_emp and Smoke. Note that Allocate_smoke and W_smoke are held as a portion of the ground combat mainline whereas Smoke_emp and Smoke are entirely separate modules in the ground combat code.

A. During artillery allocation in the ground combat driver, Allocate_smoke determines if a percent of the close support ammunition has been allocated for smoke. If a smoke allocation has been indicated, Allocate_smoke forwards this smoke tonnage to the W_smoke routine.

B. W_smoke uses the tonnages from Allocate_smoke combined with the range between units, the visibility category, the unit's width, and the withdrawal time to determine the dimensions of the desired smokescreen. W_smoke then sends necessary parameters to the Smk_emp module.

C. Smk_emp uses the dimensions received from W_smoke to emplace a smoke screen which will provide coverage for the withdrawing force. Smk_emp will attempt to emplace a screen over the entire withdrawing force (defined by the unit's length) by firing mortar ammunition. If the mortar smoke allocation is insufficient to cover the entire frontage, then the artillery smoke allocation is fired to cover any remaining frontage. The total mortar and/or artillery smoke tonnage used to emplace the screen is returned to the W_smoke routine where it is subtracted from the stockpile of the force that requested the smoke. Once the screen is emplaced, Smk_emp determines the total percent of frontage visible through the screen using the Equation 6-33. This total percent of frontage visible is sent to the direct fire attrition module routine where it is used in Equation 6-34 to determine the total number of systems available to be engaged as targets in the withdrawal phase of the DIME battle.

D. The Smoke routine determines the probability of detection through "holes" in the screen by accessing the off-line data file containing the expected probabilities of detection for a screen emplaced by a specific smoke round under certain meteorological conditions for a specified range. Smoke returns this probability of detection for an observer using an optics, crew-served, and a thermal sensor.

E. Figure 6-22 shows the general flow of the smoke module as it relates to the ground combat attrition module. Figures 6-23, 6-24, 6-25, and 6-26 indicate the specific flow diagrams for the four routines involved with smoke.

F. Table 6-8 indicates the primary variables and their functions as they relate to each subroutine. See Table 6-14 for a listing of the ground combat code.

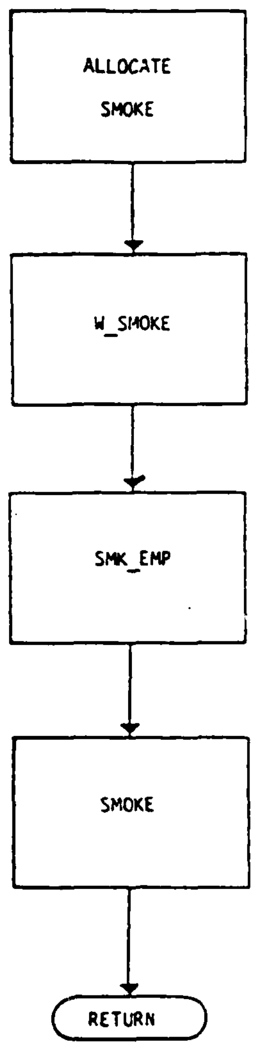


Figure 6-22. Generalized flow of smoke module.

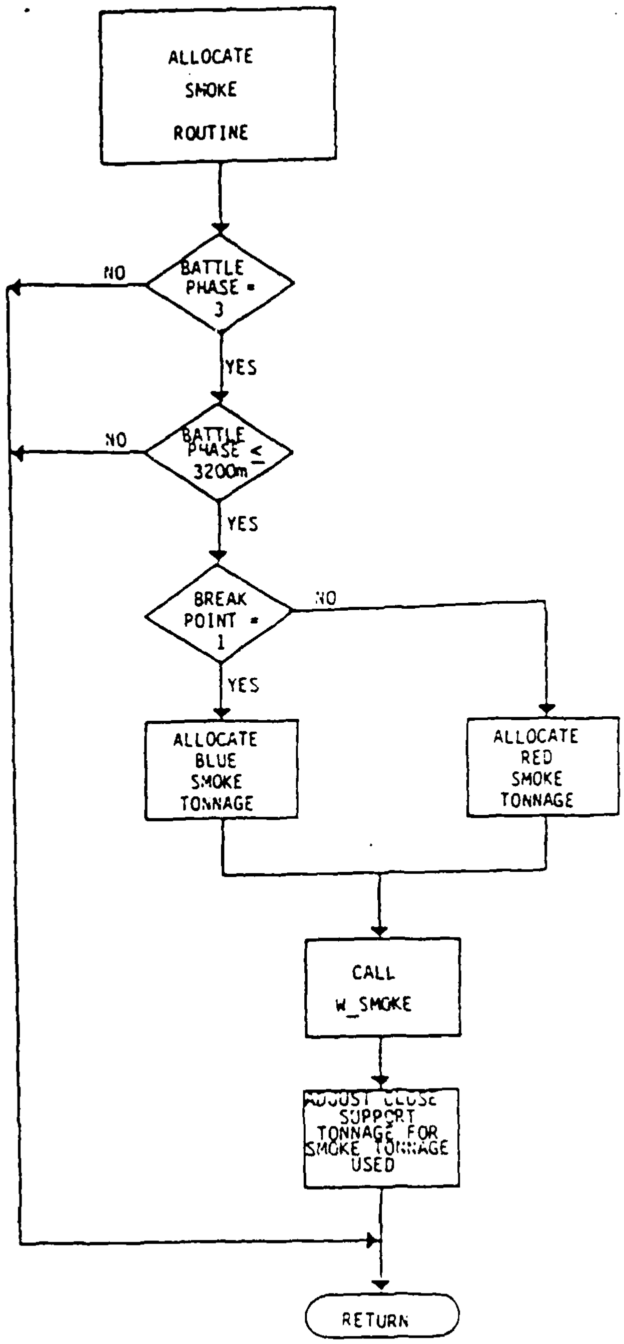


Figure 6-23. Generalized flow of Allocate_smoke routine.

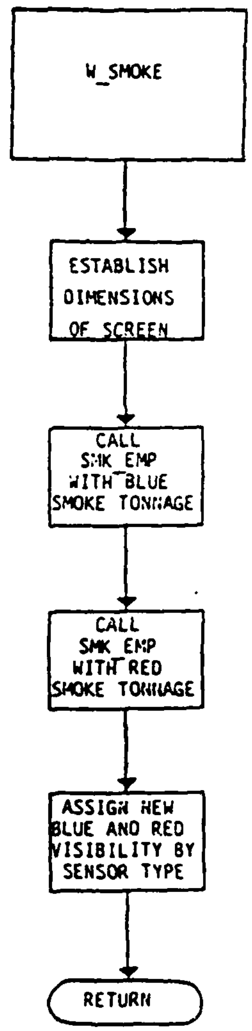


Figure 6-24. Generalized flow of W_smoke routine.

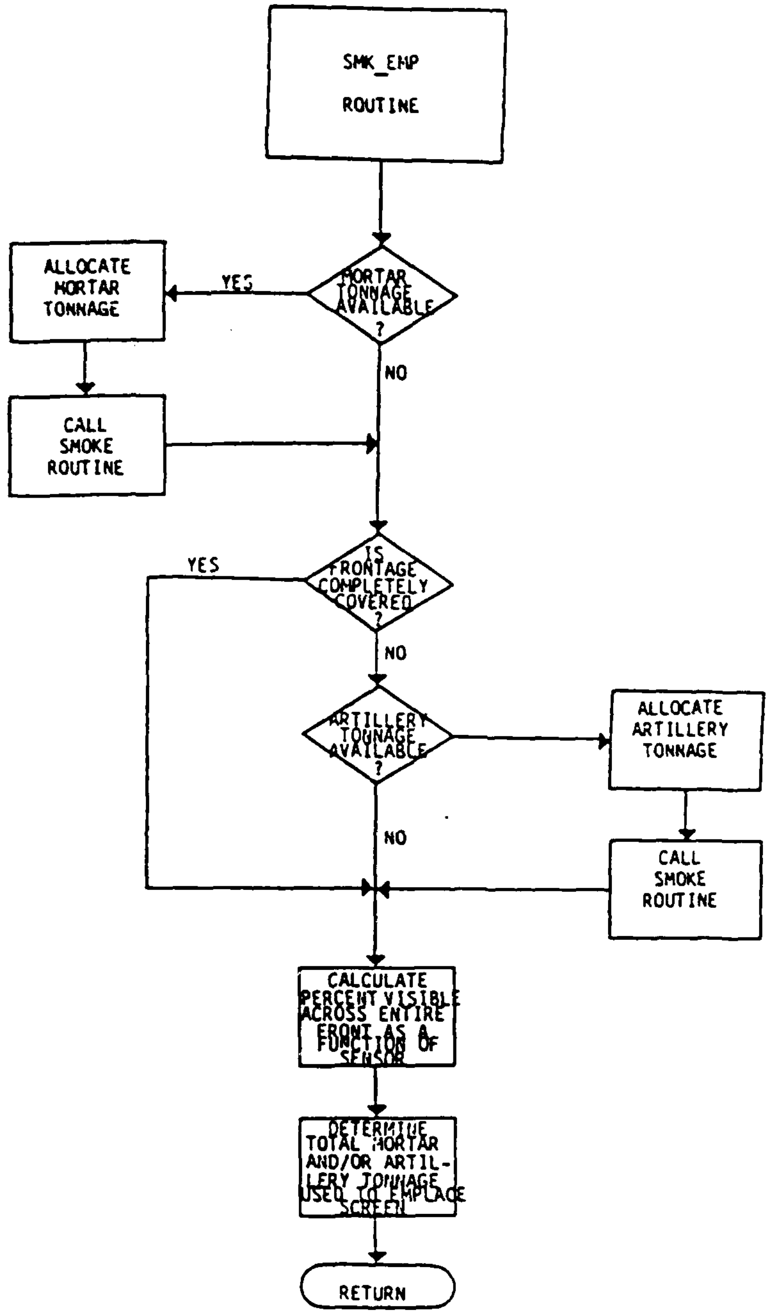


Figure 6-25. Generalize flow of Smk_emp routine.

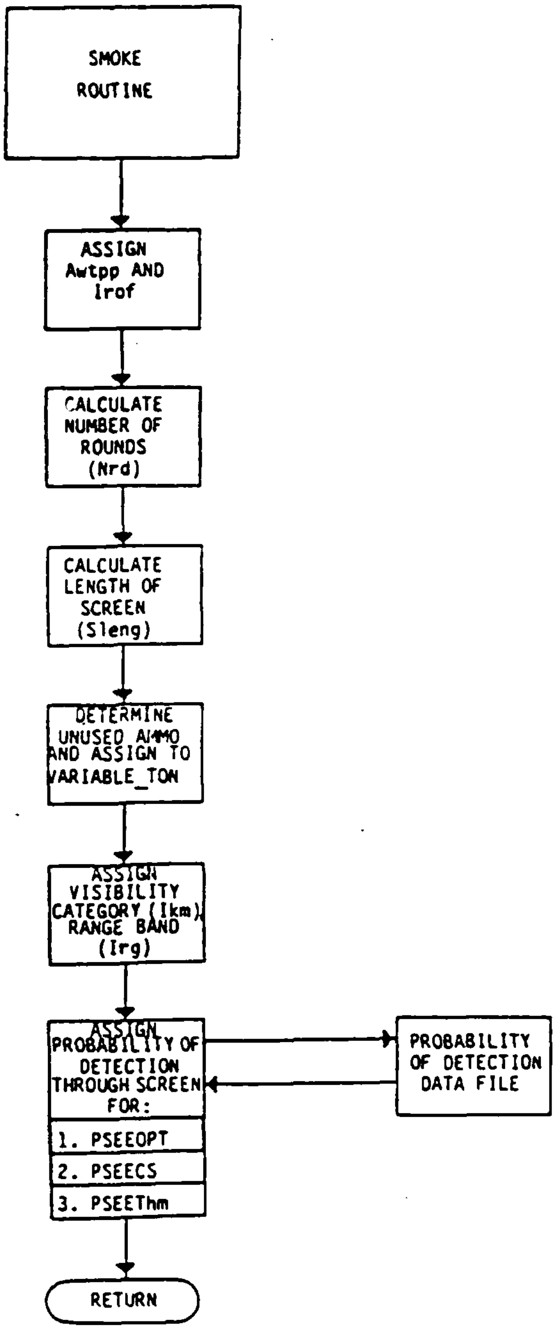


Figure 6-26. Generalized flow of routine Smoke.

Table 6-8. Smoke subroutine table.

Functional area(s): A. Provides coverage for withdrawing force; depletes the requesting unit's ammunition stockpile.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Allocate_smoke	Determines if a percent of close support ammunition was allocated to smoke by the withdrawing force. Forewards smoke tonnage to W_smoke routine. Updates ammo stockpile for smoke tonnage used to emplace screen.	A. Break_point	An integer value which indicates which force requested smoke during withdrawal phase.
		B. B_smok_tons (I)	An array containing the Blue allocated ammunition tonnage: I = 1-7 - Artillery capacity 8-11 - Mortar capacity.
		C. R_smok_tons (I)	An array containing the Red allocated ammunition tonnage: I = 1-7 - Artillery capacity 8-11 - Mortar capacity.
		D. Bif_msn_tons (I,J)	An array containing the Blue indirect fire ammunition allocation (I) as a function of mission (J). Contains adjusted tonnage after smoke mission. I = 1-11; J = 1-5.
		E. Rif_msn_tons (I,J)	An array containing the Red indirect fire ammunition allocation (I) as a function of mission (J). Contains adjusted tonnage after smoke mission. I = 1-11; J = 1-5.

Table 6-8. Smoke subroutine table.

Functional area(s): B. Prepares the dimensions of the smoke screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
W_smoke	Establishes the dimensions of the desired screen. Calls the Smkemp routine to emplace screen. Returns the percent of frontage visible through a screen as a function of sensor type.	A. Iunt	Flag which indicates the unit which emplaced the screen 1 = Blue - Light force 2 = Blue - Heavy force 3 = Red
		B. Ielem	Flag indicating whether the the screen is emplaced by artillery or mortars. 1 = artillery emplacement 8 = mortar emplacement.
		C. T_width (I)	An array containing the width of the units in contact: I = 1 - Blue unit's width = 2 - Red unit's width.
		D. S_time	Total screen time base on withdrawal time indicated by gamer.
		E. Vis	Visibility category flag. 1 = >4 km day 2 = 4 km day 3 = 2 km day 4 = 1 km day.
		F. Btl_rg	Range (kilometers) between units in contact.
		G. B_msmk_used (I)	Blue mortar smoke tonnage used to emplace screen (I = 1 - 4).
		H. B_asmk_used (I)	Blue artillery smoke tonnage used to emplace screen (I = 1 - 7).

Table 6-8. Smoke subroutine table.

Functional area(s): B. Prepares the dimensions of the smoke screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
W_smoke (concluded)		I. R_msmk_used (I)	Red mortar smoke tonnage used to emplace screen (I = 1 - 4).
		J. R_asmk_used (I)	Red artillery smoke tonnage used to emplace screen (I = 1 - 7).
		K. B_vis (I)	The percent of frontage visible through a smoke screen emplaced by the Blue force as a function of sensor type I. I = 1 - Optics = 2 - Crew served = 3 - Thermal.
		L. R_vis (I)	The percent of frontage visible through a smoke screen emplaced by the Red force as a function of sensor type I. I = 1 - Optics = 2 - Crew served = 3 - Thermal.

Functional area(s): C. Emplaces desired screen. Determines total percent of frontage visible through screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Smkemp	Attempts to cover entire frontage with mortar allocated tonnage. Covers any unsmoked frontage with artillery allocated tonnage. Calculates	A. Amowt (I)	Mortar and artillery tonnage allocated for smoke. I = 1-7 - artillery = 8-11 - mortar.
		B. Pnotsmok	Percent of frontage not covered by smoke.

Table 6-8. Smoke subroutine table.

Functional area(s): C. Emplaces desired screen. Determines total percent of frontage visible through screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Smkemp (continued)	percent of frontage visible through a screen as a function of sensor type. Calls Smoke routine to determine percent of frontage visible through "holes" in the screen.	C. Pmorto	Percent of frontage visible through mortar emplaced smoke when viewer is using an optic sensor.
		D. Amwt	Total weight of mortar/artillery ammunition available to fire for smoke.
		E. Partyo	Percentage of frontage visible through artillery-emplaced smoke when viewer is using an optic sensor.
		F. Pvopt	Total percent of frontage visible through mortar and/or artillery emplaced smoke when viewer is using an optic sensor.
		G. Pmortc	Percent of frontage visible through mortar emplaced smoke when viewer is using a crew served sensor.
		H. Partyc	Percent of frontage visible through artillery emplaced smoke when viewer is using a crew served sensor.
		I. Pvcs	Total percent of frontage visible through mortar and/or artillery emplaced smoke when viewer is using an crew served sensor.
		J. Pmortt	Percent of frontage visible through mortar emplaced smoke when viewer is using a thermal sensor.

Table 6-8. Smoke subroutine table.

Functional area(s): C. Emplaces desired screen. Determines total percent of frontage visible through screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Smkemp (continued)		K. Partyt	Percent of frontage visible through artillery emplaced smoke when viewer is using a thermal sensor.
		L. Pvth	Total percent of frontage visible through mortar and/or artillery emplaced smoke when viewer is using a thermal sensor.
		M. Mleng (I)	Percent of frontage covered by mortar emplaced smoke. (I = 1 - 4).
		N. Aleng (I)	Percent of frontage covered by artillery emplaced smoke. (I = 1 - 7).
		O. Tot_mleng	Total length of mortar emplaced smoke.
		P. Tot_aleng	Total length of artillery emplaced smoke.
		Q. Fwidt	Total frontage to be screened.
		R. Mton (I)	Total mortar tonnage used to emplace screen (I = 1 - 4).
		S. Aton (I)	Total artillery tonnage used to emplace screen (I = 1 - 7).
		T. Flwidt	Percent of frontage remaining unscreened after mortar emplaced smoke tonnage is expended.

Table 6-8. Smoke subroutine table.

Functional area(s): C. Emplaces desired screen. Determines total percent of frontage visible through screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Smkemp (concluded)		U. Apseeopt (I) Apseecs (I) Apseeth (I)	Percent of artillery emplaced smoke frontage visible through "holes" in the screen as a function of optics, crew served or thermal sensors respectively (I=1-7).
		V. Mpseeopt (I) Mpseecs (I) Mpseeth (I)	Percent of mortar emplaced smoke frontage visible through "holes" in the screen as a function of optics, crew served or thermal sensors respectively (I=1-4).

Functional area(s): D. Determines the probability of detection through desired screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Smoke	Accesses the probability of detection files for screens emplaced by mortar or artillery rounds. Returns percent of frontage visible through "holes" in an mortar or artillery emplaced screen.	A. Amowt (I)	The total amount of ammunition (tons) allocated to emplace screen.
		B. Amwtpp (I,J)	A real value containing the weight (pounds) of one round. The weight is a function of the unit (I) and the smoke round (J) used to emplace the screen (I = 1-3; J = 1-11).
		C. Irof (I,J)	A real value containing the rate of fire used to build the screen (I=1-3; J=1-11).
		D. Amwtp	A real value containing the amount of ammunition (pounds) allocated to emplace the smoke screen.
		E. Nrd	An integer value containing the number of rounds used to emplace.the screen.

Table 6-8. Smoke subroutine table.

Functional area(s): D. Determines the probability of detection through desired screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Smoke (continued)		F. Snrd (I)	The number of rounds used to emplace the screen for each type of indirect fire (I).
		G. Ton (I)	Ammunition tonnage allocated, but not used to emplace the screen.
		H. Sleng (I)	Length of screen actually produced by smoke tonnage allocated.
		I. Tot_sleng	Total length of screen which was produced by artillery or mortar.
		J. Smkdat(*)	Real array containing the probability of detection.
		K. Irh	A flag indicating the relative humidity category. 1 = Rel. humid. less than 50% 2 = Rel. humid. 50% or greater.
		L. Fwidt	A real value containing the total width (battle frontage) between units.
		M. Pseeopt (I)	Percent of smoked frontage visible through "holes" in the screen by an optics sensor.
		N. Pseecs (I)	Percent of smoked frontage visible through "holes" in the screen by a crew served sensor.
		O. Pseethm (I)	Percent of smoked frontage visible through "holes" in the screen by a thermal sensor.

Table 6-8. Smoke subroutine table.

Functional area(s): D. Determines the probability of detection through desired screen.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Smoke (concluded)		P. Irg	The range band flag, an integer from 1 - 7 where: 1 = 0 - .5 km 2 = .5 - 1.0 km 3 = 1.0 - 1.5 km 4 = 1.5 - 2.0 km 5 = 2.0 - 2.5 km 6 = 2.5 - 3.0 km 7 = 3.0 - 3.5 km

Section IV. Direct Fire Attrition

1. PURPOSE

The purpose of the direct fire module is to portray opposing ground forces exchanging volleys during direct fire combat.

2. GENERAL

A. The direct fire module is a time step deterministic model. It calculates attrition for a 30 minute period of direct fire contact. However, during the pre-closure phase of battle (phase I) forces are only in contact for a 15 minute period. To represent this pre-closure, the expected number of completed firings are multiplied by one half.

B. Attrition is calculated using 20 firers per side (elements 1 through 20 of the unit status file or "UNITFILE"). There are 70 target elements per side which are grouped into 17 out of 20 target categories for calculations. The first 17 target categories consist of ground elements, and target categories 18-20 consist of helicopter elements.

3. DATA FLOW

A. Information is received from the ground combat mainline program. This information consists of:

(1). Variables used to read appropriate data such as flags for terrain, day/night, visibility, attacker and rangeband.

(2). Ammunition, in tons, available for Red and Blue.

(3). Percent of systems vulnerable and percent of systems visible through smoke.

(4). The number of 500 meter range bands the forces move within the 15 or 30 minute period.

(5). An array containing the status for ground systems entering the battle. It contains the number of targets entering the battle, the number of firers entering the battle, and blank positions which will later contain the losses for the direct fire battle.

(6). An array containing the number of target helicopters entering the battle.

(7). Ranges, in meters, between target helicopters and direct fire systems.

B. External data consisting of sensor files, fire distributions, ammunition weights, expected number of completed firings (ECF), and probability of kill (PK) files are necessary. If more than one 500 meter range band is moved during a 30 minute period, then new ECF and PK files are needed for each new range band. This data flow is represented in Figure 6-27.

4. FILE STRUCTURE

A. External Files.

(1). Sensor files hold the sensor type values 1-3: 1 = optics, 2 = crew served and 3 = thermal. The arrays B_sen_d(J), B_sen_n(J), R_sen_d(J), and R_sen_n(J) represent the sensors during the day and night where J = 1 - 70 for each of the 70 target elements. B_vis(I) and R_vis(I) are internal data arrays which contain the percentage of targets visible through smoke for each of the three sensor types (I = 1 to 3). B_vis(I) and R_vis(I) are accessed through the values found in the sensor arrays.

(2) Category files are needed to place the 70 target elements into the target categories for attrition calculations. The B_cat(I) and R_cat(I) arrays contain a value between 1 and 17, which represent the target category they fall into. (Categories 18-20 are reserved for the 3 types of enemy helicopters.)

(3) Fire distribution files contain weighted factors (a value from 1 to 10) which each firer applies to one of the twenty target categories when firing. The higher value indicates a higher preference in firing at that target. The arrays B_fire_d(I,J) and R_fire_d(I,J) represent the weighted factor in the following manner:

I = (1-20) is the firer
J = (1-20) is the target category.

(4) Weight, engagement, and pointer files. Ammunition weights are kept within the B_ammo_wt(I) and R_ammo_wt(I) arrays for each firer. B_engagements(I) and R_engagements(I) are the number of engagements for each firer. Array Df_sen_ptr(S,I) contains the direct fire sensor pointers (value 1 = optical ground or 2 = thermal ground) and array Df_muni_ptr(S,I) contains the munitions pointers (value 1 = ground missile or 2 = ground kinetic energy round), where:

S = (1-2) for each side (1 = blue, 2 = red)
I = (1-20) for each firer

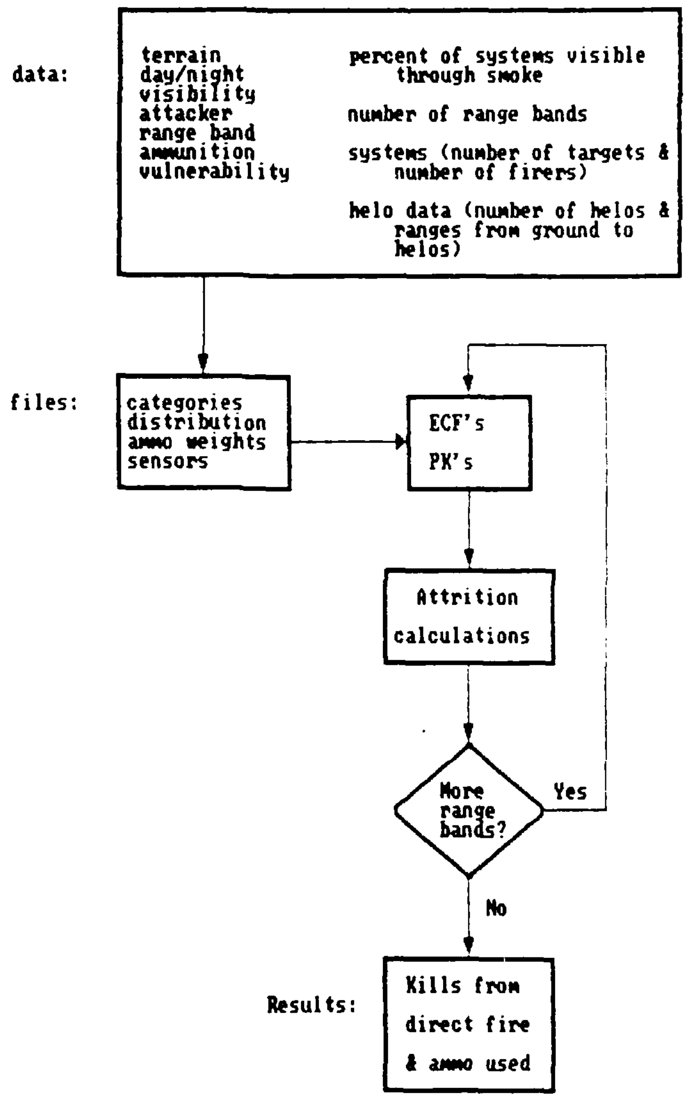


Figure 6-27. Direct fire data flow.

(5) Expected number of completed firing (ECF) files contain the expected number of completed firings for each of the twenty firers during a 30 minute period. ECF's are dependent upon day/night, defending/attacking and terrain represented within the file name for both Blue and Red. Dependent also on the range, the 6 different direct fire range bands are represented by different records within each file. Weather visibility is also an important factor for ECF's and is designated in the arrays B_ecf_vis(I,J) and R_ecf_vis(I,J) where:

I = (1-20) the twenty firers
J = (1-4) the visibility categories
(1) greater than 5 km
(2) 5 km
(3) 2 km
(4) 1 km.

(6) Probability of kill (PK) files are available for two types of target postures, hull defilade and fully exposed, which are represented by the file name. Dependent on range, the 6 records of each file represent the range band for the direct fire combat. The arrays containing the PK's are:

B_pk_fe(I,J), B_pk_hd(I,J), R_pk_fe(I,J), R_pk_hd(I,J)

where: I = (1-20) the firers
J = (1-20) the target categories.
J = (18-20) are helicopter target categories

5. ALGORITHMS

A. Figure 6-28 presents a generalized logic flow of the processes in the ground combat direct fire subroutine. Direct fire attrition involves calculating the fire distribution factor, rounds available and the attrition calculations. The following paragraphs provide a more detailed description of the algorithms used in the attrition process.

(1) Calculate the fire distribution factor (Fdf_{ij}).

$$Fdf_{ij} = (T_j * D_{ij} * Pkfe_{ij}) / \sum_{j=1}^{20} (T_j * D_{ij} * Pkfe_{ij}) \quad \text{(Eq. 6-35)}$$

where:

T_j = number of targets in category j being fired upon.
D_{ij} = a weighted factor to represent preferred distribution of fire for firer i vs. target j.
$Pkfe_{ij}$ = probability of kill for fully exposed targets (j) being fired on by i and where: i = (1-20) the firers and j = (1-20) target categories.

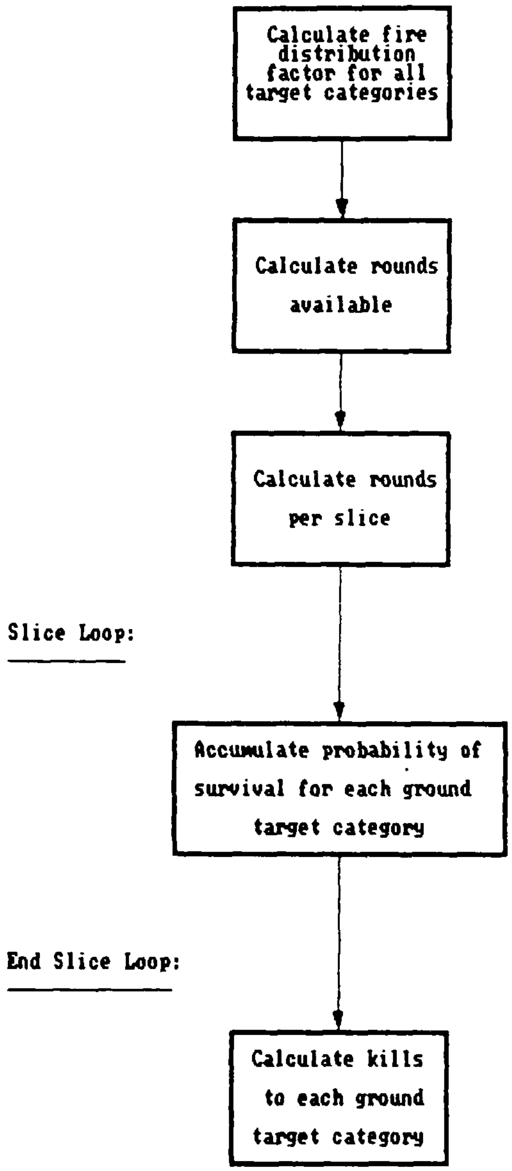


Figure 6-28. Direct fire logic flow.

At this point the Fdf_{ij} and Pk_{ij} of helicopter target categories i = 18- 20 are saved and used in the helicopter attrition portion (see section V of this chapter) of P4.

The following direct fire calculations are based on the ground target categories (j = 1-17); the helicopter target categories (j = 18-20) are ignored.

(2) Calculate rounds available. The rounds available to each firer for each of its target categories is calculated in the following manner:

$$Rnds_{ij} = Fdf_{ij} * Nf_i * MIN(Ecf_i, Ammo_i) \qquad \text{(Eq. 6-36)}$$

where:

$Rnds_{ij}$ = rounds available to each firer (i) per ground target category (j).
Nf_i = number of firers of type i.
Ecf_i = expected number of completed firings for firer i.
$Ammo_i$ = ammunition available for firer i.

(3) Slice methodology. Attrition during the 30 minute period is broken into 15 slices of attrition calculations. This methodology brings about a more accurate representation of exchanging fire during battle. It allows the attrition to occur in 15 sections during a 30 minute period rather than one large mass of fire in one moment for a 30 minute period. Before each slice a new value for firers and targets is used which represents deducted kills from the previous slice.

Before this slice methodology may be used to calculate attrition, the rounds available must be broken down into the number of rounds fired per slice. The calculation is done as follows:

$$Er_{ij} = Rnds_{ij} / Nf_i / Ns \qquad \text{(Eq. 6-37)}$$

where:

Er_{ij} = expected number of rounds per slice for each firer at each ground target category.
Ns = number of slices.
$Rnds_{ij}$ = rounds available to each firer per ground target category.
Nf_i = number of firers.

For each slice k, the attrition calculations involve the probability of survival for each target category being fired on by each firer:

$$P_{ijk} = ((1-(Pv_j * Pkfe_{ij} + Pnv_j * Pkhd_{ij})) / Nt_j) \uparrow (Nf_i * Er_{ij}) \qquad \text{(Eq. 6-38)}$$

where:

- P_{ijk} = the probability of survival for the jth ground target category being fired upon by the ith firer.
- Pv_j = percent of vulnerable targets for the jth ground target category.
- $Pkfe_{ij}$ = probability of kill for fully exposed targets in the jth ground target category being fired upon by the ith firer.
- Pnv_j = percent of targets not vulnerable for the jth ground target category.
- $Pkhd_{ij}$ = probability of kill for hull defilade targets in the jth ground target category being fired upon by the ith firer.
- Nt_j = total number of targets of type j.

The value of P_{ijk} is accumulated for all firers firing on each ground target category for slice k by:

$$P_{jk} = \prod_{i=1}^{17} P_{ijk} \qquad \text{(Eq. 6-38a)}$$

The value P_{jk} is also accumulated for all slices by:

$$Ps_j = \prod_{k=1}^{Ns} P_{jk} \qquad \text{(Eq. 6-38b)}$$

giving one probability of survival per ground target category over all slices and over all firers.

(4) *Losses per target category are then calculated using the following:*

$$Lj = Tj * Vj * (1-Ps_j) \qquad \text{(Eq. 6-39)}$$

where:

- L_j = losses in the jth ground target category.
- T_j = number of targets being fired upon in the jth target category.
- V_j = percent of visible targets through smoke in the jth ground target category.

6. "UNITFILE" IMPACT

This subroutine does not directly impact the "UNITFILE". Kills calculated in this subroutine along with the amount of ammunition used is returned to the ground combat mainline and then decremented from the "UNITFILE".

7. CODE

A. Introduction. This section contains information on the direct fire attrition code. The functional areas discussed in the following paragraph are represented in Figure 6-29.

B. Direct fire attrition functional areas.

(1) The data received from the ground combat mainline consists of terrain, day/night, weather visibility, the percent of targets fully exposed, the number of 500 meter range bands the force is to move, the beginning range, whether red or blue is the attacker, sensor visibilities through smoke, ammunition available and the number of targets and fires on both sides.

(2) Set_call. This portion of the code saves the original range and initial targets. If the number of range bands is 1/2, one attrition loop with half the ECF's are utilized to represent pre-closure battle.

(3) Init_reads. Reads external sensor data, category files, fire distributions and ammunition weights which do not change by range.

(4) The following calculations are necessary for each 500 meter range covered by the forces:

(a) Read_files. The PK and ECF files are read for a specific range.

(b) Categorize. This code calculates the total number of targets per target category and the percent in which each target is fully exposed.

(c) Smokes. This code calculates the total number of systems available to be engaged as targets in the battle.

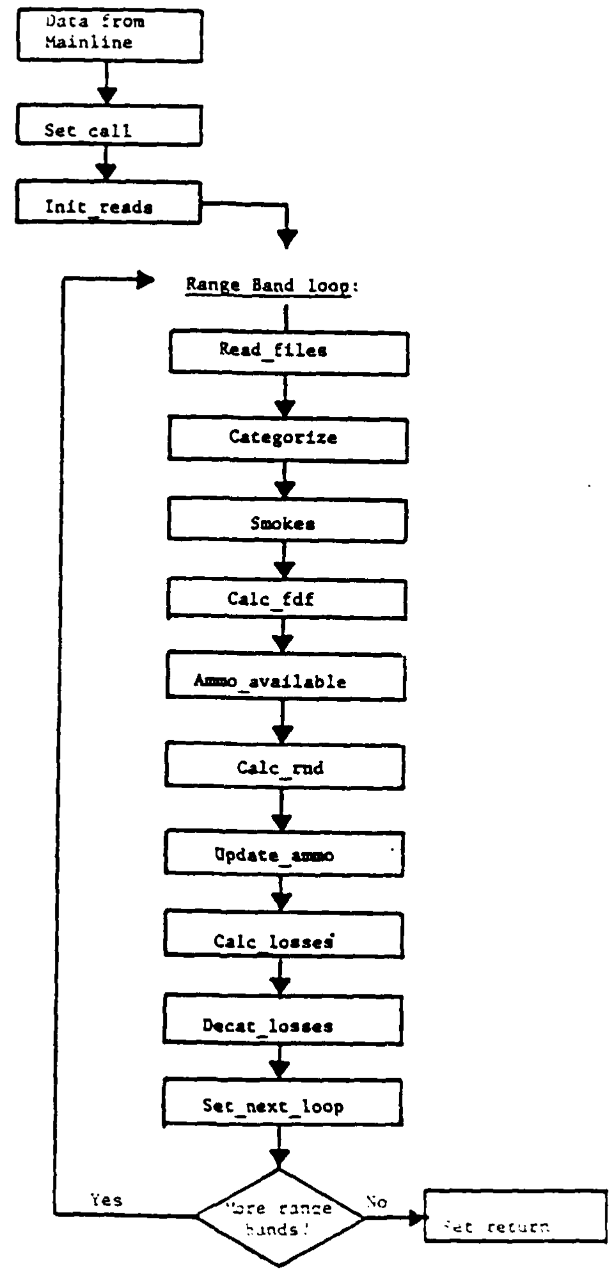


Figure 6-29. Direct fire functional flow.

(d) Calc_fdf. This portion of the code calculates the fire distribution factor being the distribution of rounds fired at the twenty target categories.

(e) Ammo_available. Determines the number of rounds available for firing per single weapon from the tons available sent from the mainline.

(f) Calc_rnd. Rounds fired by one of each of the twenty firers is determined by the minimum of rounds available and expected number of completed firings. These rounds per single weapon must then be multiplied by the total of each system type firing. These rounds must then be distributed between each of the ground target categories by using the fire distribution factor.

(g) Update_ammo. Accumulates the amount of ammo used and is returned to the mainline to be decremented from "UNITFILE" ammo.

(h) Calc_loss. Using the slice methodology discussed in the algorithm portion of direct fire attrition, the number of slices are set to 15 in this portion of the code. Before actual attrition is calculated, the number of rounds fired per slice must be calculated from the previous rounds per category. Occurring within the actual slice loop calculations are the following.

1. Firers are recalculated at the beginning of each slice to discard any killed during the previous slice(s).

2. Probability of survival is accumulated for all firers firing upon each ground target category. This, in turn, is accumulated for all slices as discussed in the algorithm portion of the direct fire documentation.

Having calculated the overall probability of survival for each ground target category, the losses are then determined by multiplying the number of targets in each category by their probability of kill (one minus the probability of survival) and their percent visible through smoke. This portion of the code is more explicitly displayed in Figure 6-30.

(i) Decat_losses. This apportions the losses within the first 17 target categories to losses within the 70 systems.

(j) Set_next_loop. This portion of the code decrements the appropriate losses from the firers, and the targets available along with adding in losses all contained in the array Sys(*). The range band is decremented to prepare for the following attrition calculations in another range.

(5) Set_return. Restores the range band and initial targets before returning the direct fire losses and ammunition used to the ground combat mainline.

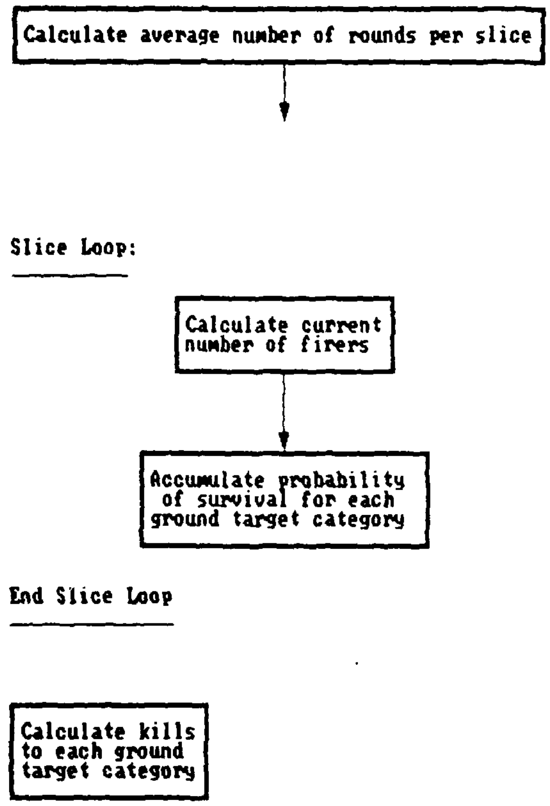


Figure 6-30. Direct fire functional flow of Calc_losses.

C. The primary variables of each functional area of the direct fire attrition subroutine are shown in Table 6-9. Each variable is accompanied by a short description. Table 6-14 contains a listing of the ground combat code.

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Main	Main driver for Df attrition.	A. Band_loop	Loop counter; counts number of bands
		B. Loops	Number of attrition loops necessary for the number of range bands and length of battle.
Set_call	Saves the original range and initial targets.	A. Rng_band	The current range band
		B. Rng_band_save	Save value of Rng_band
		C. Init_b_targets(I)	Array containing the initial number of Blue type I targets; I = 1 - 70
		D. Init_r_targets(I)	Array containing the initial number of Red type I targets; I = 1 - 70
		E. Num_bands	Number of 500 meter range bands
		F. Partial	If the number of range bands is less than 1, Partial = number of bands; otherwise, Partial = 1
		G. Sys(1,I)	Array containing the remaining number of Blue targets of type I for the next range band
		Sys(3,I)	Array containing the remaining number of Red targets of type I for the next range band; I from 1 to 70
		Sys(2,I)	Array containing the number of losses to Blue targets of type I; I = 1 - 70

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Set_call (concluded)		Sys(4,I)	Array containing the number of losses to Red targets of type I; I = 1 - 70
		Sys(5,I)	Array containing the number of remaining Blue firers of type I; I = 1 - 70
		Sys(6,I)	Array containing the number of remaining Red firers of type I; I = 1 - 70
Init_reads	Reads internal sensor data, external category files, fire distributions, and arms weights which do not change by range.	A. Day_night	0 = Day 1 = Night
		B. B_sen(I)	Array containing sensor data for Blue target I visible through smoke; I = 1 - 70
		C. R_sen(I)	Array containing sensor data for Red target I visible through smoke; I = 1 - 70
		D. Atk_def	0 = Blue attacking/Red defending 1 = Red attacking/Blue defending
		E. Terrain	1 = Open 2 = Rolling 3 = Hilly 4 = Mountainous

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Read_files	The probability of kill (PK) files and the expected number of completed firing (ECF) files are read for a specific range.	A. B_ecf_vis(I,J)	Array containing the expected number of completed firings for Blue firer I; I = 1 - 20; J = 1 for >5 km = 2 for 5 km = 3 for 2 km = 4 for 1 km
		B. R_ecf_vis(I,J)	Array containing the expected number of completed firings for Red firer I; I = 1 - 20; *J same as for B_ecf_vis(I,J)*
		C. B_ecf(I)	Array containing the expected number of completed firings for a fixed visibility category for Blue firer type I; I = 1 - 20
		D. R_ecf(I)	Array containing the expected number of completed firings for a fixed visibility category for Red firer type I; I = 1 - 20
		E. B_pk_fe (I,J) R_pk_fe (I,J)	Probability of kill for fully exposed target J being fired on by firer I. (I=1-20; J=1-20)
		F. B_pk_hd (I,J) R_pk_hd (I,J)	Probability of kill for hull defilade target J being fired on by firer I. (I=1-20; J=1-20)

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Categorize	Calculates the total number of targets per target category and the percent in which each category is fully exposed.	A. B_cat(I)	Array containing the type of ground category for Blue element I. I = 1 - 70. There are 17 types of ground categories.
		B. R_cat(I)	Array containing the type of ground category for Red element I. I = 1 - 70. Types of categories are the same as for B_cat(I)
		C. B_targ(I)	Array containing the number of Blue targets of category I; I = 1 - 20
		D. R_targ(I)	Array containing the number of Red targets of category I; I = 1 - 20
		E. B_vul_t(I)	Array containing the total vulnerability of Blue targets of category I. I = 1 - 20. Determined by the number of remaining type I elements and their vulnerability, as given by B_vua(I)
		F. B_vua(I)	Array containing the vulnerability of Blue target I; I = 1 - 70

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Categorize (continued)		G. R_vul_t(I)	Array containing the total vulnerability of Red targets of category I, I = 1 - 20. Determined by the number of remaining type I elements and their vulnerability, as given by R_vua(I).
		H. R_vua(I)	Array containing the vulnerability of Red target I; I = 1 - 70
		I. B_vul(I)	Array containing the vulnerability of Blue category I. I = 1 - 20. Determined by dividing B_vul_t(I) by the number of Blue targets of category I.
		J. R_vul(I)	Array containing the vulnerability of Red category I. I = 1 - 20. Determined by dividing R_vul_t(I) by the number of Red targets of category I.

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Smokes	Calculates the total number of systems available to be engaged as targets in the battle.	A. R_vis(I)	Array containing the percent of frontage visible through a smoke screen emplaced by the Blue forces as a function of sensor type I: I = 1 Optics = 2 Crew served = 3 Thermal
		B. B_vis(I)	Array containing the percent of frontage visible through a smoke screen emplaced by the Red force as a function of sensor type I
		C. B_targ_vis(I)	Array containing the number of Blue type I targets which are visible to Red; I = 1 - 20. Determined by the number of Red firers and the percent of frontage visible through a smoke screen emplaced by Blue forces. This is divided by the number of Blue firers of type I.
		D. R_targ_vis(I)	Array containing the number of Red type I targets which are visible to Blue; I = 1 - 20
		E. B_look(I)	Array containing the number of Blue firers of type I (I=1-20).
		F. R_look(I)	Array containing the number of Red firers of type I (I=1-20).

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Calc_fdf	Calculates the fire distribution factor; i.e., the distribution of rounds fired at the 20 target categories.	A. B_fire_d(I,J)	Array containing the fire distribution factor of Blue firer I on Red target J. Data accessed from data file. I = 1 - 20; J = 1 - 20.
		B. R_fire_d(I,J)	Array containing the fire distribution factor of Red firer I on Blue target J. Data accessed from data file. I = 1 - 20; J = 1 - 20.
		C. B_sum(I)	Array containing the sum of the fire of Blue firer of type I. I = 1 - 20. Determined by the number of Red targets, the preference factor of firer I against the Red target, and the probability of Blue firer I killing the Red target.
		D. R_sum(I)	Array containing the sum of the fire of Red firer of type I
		E. B_fdf(I,J)	Array containing the fire distribution factor of Blue firer I on Red target category J (I = 1 - 20; J = 1 - 20.) Determined by the number of Red targets of category J, the preference factor of Blue firer I against Red target J and the probability of Blue firer I killing fully exposed target J. This is divided by B_sum(I).

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Calc_fdf (concluded)		F. R_fdf(I,J)	Array containing the fire distribution factor of Red firer I on Blue target category J. I = 1 - 20; J = 1 - 20.
Ammo_available	Determine the number of rounds available for firing per single weapon from the tons available sent from the mainline.	A. B_ammo_wt(I)	Array containing the ammunition weight in pounds for firing per single weapon for Blue type I firer; I = 1 - 20
		B. R_ammo_wt(I)	Array containing the ammunition weight in pounds for firing per single weapon for Red type i firer; I = 1 - 20
		C. B_tons(1,I)	Array containing tons of ammunition initially available for Blue firer I sent from the mainline; I = 1 - 20
		B_tons(2,I)	Array containing tons of ammunition used by Blue firer I
		D. R_tons(1,I)	Array containing tons of ammunition initially available for Red firer I sent from the mainline; I = 1 - 20
		R_tons(2,I)	Array containing tons of ammunition used by Red firer I; I = 1 - 20
		E. B_ammo(I)	Array containing number of rounds available for firing per single weapon for Blue type I firer; I = 1 - 20

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Ammo_available (concluded)		F. R_ammo(I)	Array containing number of rounds available for firing per single weapon for Red type I firer; I = 1 - 20
Calc_rnd	Calculates the number of rounds fired by each of the twenty firers.	A. B_rnds_wep(I)	Array containing the number of rounds fired by Blue firer I; I = 1 - 20. Determine by taking the minimum value of the number of rounds available and the expected number of completed firings.
		B. R_rnds_wep(I)	Array containing the number of rounds fired by Red firer I; I = 1 - 20
		C. Tot_b_rnds(I)	Array containing the total number of rounds fired by all of the Blue firers of type I; I = 1 - 20. Determined by the number of Blue firers of type I and B_rnds_wep(I).
		D. Tot_r_rnds(I)	Array containing the total number of rounds fired by all of the Red firers of type I; I = 1 - 20. Determined by the number of Red firers of type I and R_rnds_wep(I).

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Calc_rnd (concluded)		E. B_rnds_cat(I,J)	Array containing the number of rounds fired from Blue firer I against Red target category J. I = 1 - 20; J = 1 - 20. Determined by the fire distribution factor of firer I against target J, and the total number of rounds fired by I.
		F. R_rnds_cat(I,J)	Array containing the number of rounds fired from Red firer I against Blue target category J. I = 1 - 20; J = 1 - 20.
Update_ammo	Accumulates the amount of ammunition used, in tons.		
Calc_loss	Calculates the average number of rounds fired per slice. Calculates the number of firers which survive for Red and Blue.	A. B_ex_r(I,J)	Array containing the average number of rounds fired per slice by Blue firer I against Red target category J. I = 1 - 20; J = 1 - 20.
	Calculates the losses for target categories for Red and Blue.	B. R_ex_r(I,J)	Array containing the average number of rounds fired per slice by Red firer I against Blue target category J. I = 1 - 20; J = 1 - 20.

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Calc_loss (continued)		C. B_firers	Number of Blue firers surviving. Determined by the probability of survival of type I firer, and the number of type I firers.
		D. R_firers	Number of Red firers surviving
		E. Ps_r_ps	Probability of Red target J being killed by Blue firer I. Determined by the vulnerability of target J, the probability of firer I killing the fully exposed target J, and the probability of firer I killing the hull defilade target J.
		F. Ps_b_ps	Probability of Blue target J being killed by Red firer I
		G. R_ps(I,J)	Array containing the probability of Red target category J surviving against Blue firer I. I = 1 - 20; J = 1 - 20. Determined by Ps_r_ps, the number of Blue firers surviving, and the average number of rounds fired per slice by Blue firer I against Red target category J.
		H. B_ps(I,J)	Array containing the probability of Blue target category J surviving against Red firer I. I = 1 - 20; J = 1 - 20.

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Calc_loss (continued)		I. B_psuv (I)	Accumulated probability of Blue target categories surviving against all Red firers (I=1-20).
		J. R_psuv (I)	Accumulated probability of Red target categories surviving against all Blue firers (I=1-20).
		K. Lossr	Number of losses to Red category J targets. Determined by the number of J type targets which are visible to Blue, and R_ps(I,J).
		L. Lossb	Number of losses to Blue category J targets
		M. Lo_r_cat(I,J)	Array containing the number of losses of Red type J targets, due to Blue type I firers. I = 1 - 20; J = 1 - 20.
		N. Lo_b_cat(I,J)	Array containing the number of losses of Blue type J targets, due to Red type I firers. I = 1 - 20; J = 1 - 20.
		O. Lo_r_cats(J)	Number of losses of Red type J targets; J = 1 - 20
		P. Lo_b_cats(J)	Number of losses of Blue type J targets; J = 1 - 20
		Q. Loss_r_cat(J)	Array containing the number of losses of Red category J targets

Table 6-9. Direct fire subroutine table.

Functional area(s): Df attrition; Direct fire attrition.

Subroutine called	Subroutine function(s)	Primary variables	Variable description
Calc_loss (concluded)		R. Loss_b_cat(J)	Array containing the number of losses of Blue category J targets
		S. Red_target	A weighting factor for Ps_r_ps
		T. Blue_target	A weighting factor for Ps_b_ps
Decat_losses	Apportions the losses within the 17 target categories to losses within the 70 target categories.	A. Loss_blue (I) Loss_red (I)	Number of losses to each element (I=1-70).
Set_next_loop	Decrements the appropriate losses from the firers and targets available.		
Set_return	Restores the range band and initial targets. Returns the direct fire losses and ammunition used to the ground combat mainline.		

Section V. Helicopter Operations

1. PURPOSE

The purpose of the helicopter operations section of the DIME ground combat program is to realistically game air-to-ground, ground-to-air, and air-to-air interactions at variable ranges and missions between helicopters and the 20 target categories described in Section IV. These interactions take place during ground combat operations and are separate from air operations (ingress, egress, or strike) as in the DIME air defense program.

2. GENERAL

A. Both Red and Blue forces are played with capabilities and limitations which can mirror current and future equipment, vehicles, weapons, munitions, and helicopters.

B. Program characteristics include:

(1) Two types of attack helicopters and one scout helicopter are played for the Red and Blue forces.

(2) Seven possible air defense (AD) types may be played.

(3) Twenty possible direct fire types may be played.

(4) Losses to ground targets and helicopters are calculated by using various algorithms available within the module. These losses are a function of:

(a) Visibility (day or night)

(b) Terrain (mountainous, hilly, rolling, open)

(c) Target profile (fully exposed or hull defilade)

(d) Helicopter munitions (missiles, guns and missiles, guns/rockets or air to air missiles).

3. DATA FLOW

The helicopter module receives data from the main driver routine and auxiliary stored files. These data are used by the Helo_kills module to calculate the total elements killed by the helicopters and the total helicopters killed by the air defense and direct fire elements. See the data flow shown in Figure 6-31.

A. Inputs. The helicopter module receives the following data through a call statement from the main driver routine.

(1) Cell (S,I). An array which contains the cell size where:

S = 1 = Blue side
2 = Red side

I = 1 = the number of helicopter type 1
2 = the number of helicopter type 2
3 = the number of helicopter type 3

(2) Target (S,I,J). An array which contains the target matrix where:

S = Pointer to side

I = 1 = original number of systems
2 = surviving number of systems

J = 1 to 70 systems

(3) Ad_ammo(S). The ammunition available in short tons (2000 lbs) where:

S = Pointer to side

(4) Terr. An integer value of 1 to 4 which identifies the defender's terrain where:

1 = Open
2 = Rolling
3 = Hilly
4 = Mountainous

(5) Atk_prof (S,I). An integer value of 1 to 7 which identifies the helicopter's attacker profile:

1 = Missiles only
2 = Missiles and guns
3 = Guns and rockets
4 = Air to Air Missile

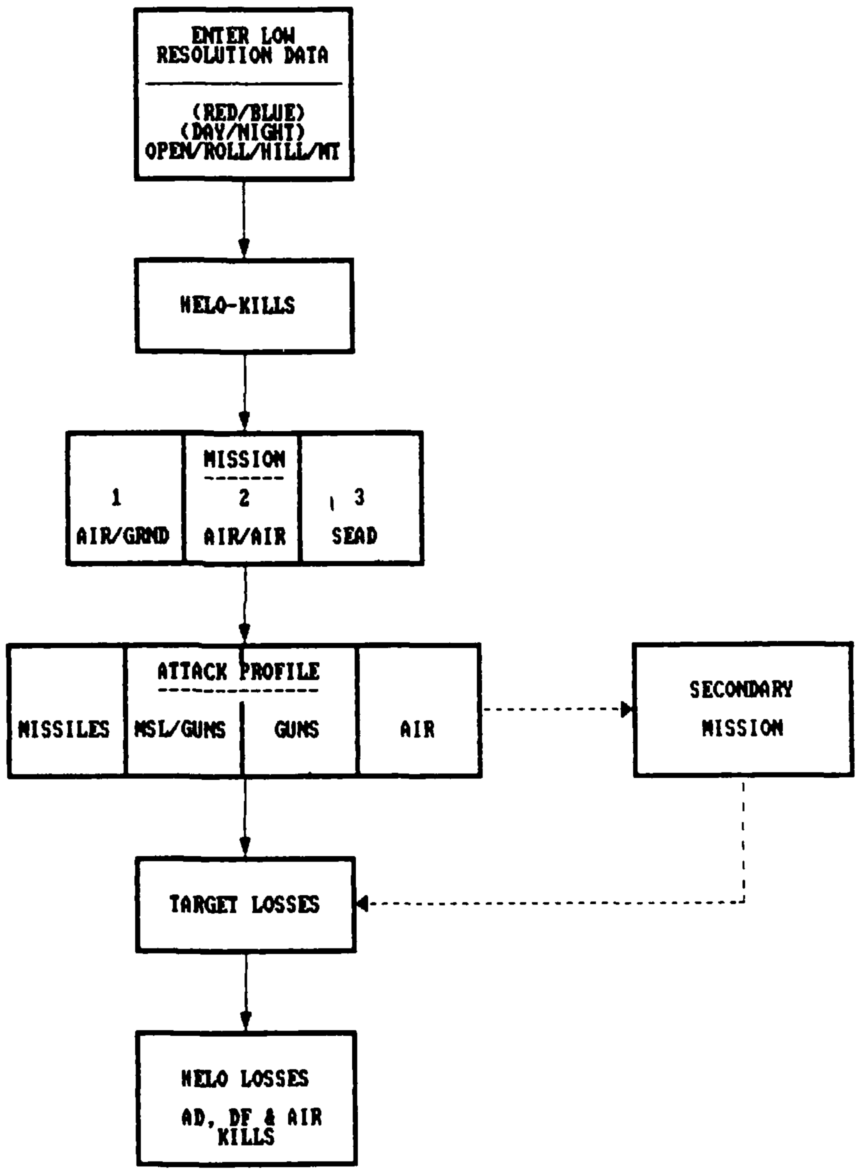


Figure 6-31. Data flow of helicopter attrition module.

5 = Air to Air Missile with Missiles
6 = Air to Air Missile with Missiles and Guns
7 = Air to Air Missile with Guns and Rockets

where:

S = pointer to side
I = helicopter type

(6) Helo_mis (S,I). An integer value of 1 to 3 which identifies the attacker's mission:

1 = Air to ground
2 = Air to air
3 = Suppression of Enemy Air Defense (SEAD)

where:

S = pointer to side
I = helicopter type

(7) Day_nite. An integer value representing the light visibility category, where:

0 = Day
1 = Night

(8) Time_step. The on-site time of a helicopter cell in minutes.

(9) P_def (S,I). Percent of the targets in hull defilade. The remaining targets are in the fully exposed posture where:

S = pointer to side
I = 1 to 70 systems

(10) Arty (S). An integer value of 1 or 2 which identifies the presence or absence of artillery fires, where:

1 = Yes, under artillery attack
2 = No, not under artillery attack

(11) Veh_ada (S). Percentage of vehicular AD Systems suppressed where:

S = pointer to side

(12) Hnd_ada(S). Percentage of hand-held AD systems suppressed where:

S = pointer to side

(13) Stnd_off_rg (S,I). A variable integer value indicating the attacker's standoff range to the opposing target in meters where:

S = pointer to side
I = helicopter type

(14) Vis. An integer value of 1-4 indicating the atmospheric visibility where:

1 = > 5 km
2 = 5 km
3 = 2 km
4 = 1 km

B. The helicopter module also receives input data through the common block area called "Helo_info".

(1) Btl_rg. The current range between the two forces on the battle field.

(2) Rg_avg (S,I,J). The range between the helicopter I (I = 1 - 3) and the target category J (J = 1 - 20) for side S (S = 1 - 2) in meters. The ranges are calculated in subprogram Helo_range and are illustrated in Tables 6-Va, 6-Vb, and 6-Vc.

(3) Df_ammo (S). Direct fire ammunition available to shoot at helicopter targets for side S.

(4) Df_fire_dist (S,I,J). Fire distribution factor of each direct fire type I shooting at target helicopter J (J = 1-3).

(5) Df_pk_helo (S,I,J,M). Direct fire probability of kill of helicopters J, mast mounted (M=1) and non-mast mounted (M=2).

(6) Df_sen_ptr (S,I). Pointer to type of sensor (value 1 = optical ground, 2 = thermal ground) for direct fire type I (I = 1-20) on side S (S = 1-2).

(7) Df_muni_ptr (S,I). Pointer to type of munition (value 1 = ground missile, 2 = ground kinetic energy round) for direct fire type I (I = 1- 20) on side S (S = 1-2).

C. Other inputs required by the helicopter module are read in from auxiliary storage files, such as the: helicopter characteristics files; helicopter sensor files; helicopter munitions files; helicopter vulnerability files; AD vulnerability files; AD ammunition files; helicopter target preference file; direct fire sensor files; direct fire munitions files. This data is stored in common block area "Helo_attrite" for later retrieval in the Helo_kills module.

D. Outputs. The helicopter module returns to the main driver routine the number of elements in a target category killed by the helicopters and the number of helicopters killed by AD elements, ground elements and other helicopters.

4. FILE STRUCTURE

DIME helicopter files are used by the ground combat (P4) program to supply statistical performance/preference data and physical characteristics data (basic load, weight/round, etc.). Figure 6-32 is an example of the interaction between the file structures.

A. Helicopter characteristics files.

There is one helicopter characteristics file for each force. Each file contains three records, one for each helicopter type. Each helicopter record contains eight elements. The structure of each record is as follows:

Index	Description
1	Pointer to unique sensor type (value 1-10).
2	Pointer to unique missile type (value 1-15).
3	Pointer to unique gun type (value 1-15).
4	Pointer to unique air to air missile (value 1-15).
5	Basic load for the unique missile above
6	Basic load for the unique gun above
7	Basic load for the unique air to air missile above
8	Sensor location (0= non-mast mounted, 1= mast mounted)

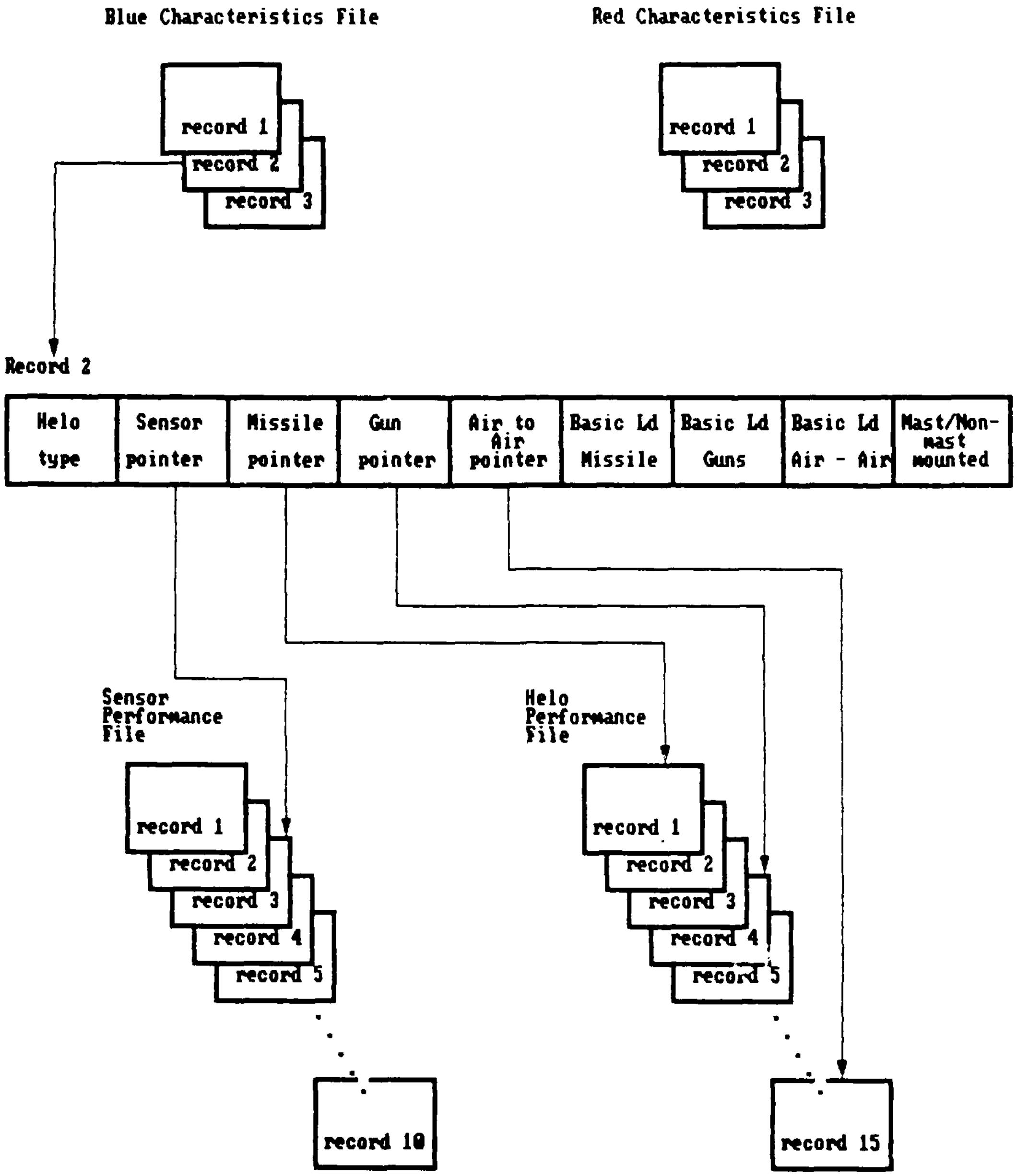


Figure 6-32. Helicopter file hierarchy.

In this figure, record 2 of the Blue Characteristics file is shown in detail. Each record describes a helicopter type plus sensor and munition pointers. The sensor pointer points to record 3 of the Sensor Performance File which describes a unique type of sensor (contains probability of detection data). The missiles, gun and air-to-air pointers respectively point to records 1, 4 and 15 of the Helicopter Performance File. These records describe a munition type (contain probability of kill data, tactical number of rounds, fire and guide time and time masked and exposed.

B. <u>Helicopter sensor performance files.</u>

There is one helicopter sensor performance file for each force. Each file contains ten records for unique types of sensors. Each sensor record contains seventy-seven elements used to represent the probability of detection ($P\infty$) and the average time to detect ($\overline{T}$) for each target. Record entries represent fitted coefficients in the equation

$P\infty = ar^2 + br + c$ and $\overline{T} = a'r^2 + b'r + c'$ where

r = range in meters to target.

The structure of each record is as follows:

Index	Description
1	Sensor description string (eight characters).
2	Probability of detection ($P\infty$) coefficient [a] of target category personnel, fully exposed
3	Probability of detection ($P\infty$) coefficient [a] of target category personnel, hull defilade
4	Probability of detection ($P\infty$) coefficient [a] of target category light vehicles, fully exposed
5	Probability of detection ($P\infty$) coefficient [a] of target category light vehicles, hull defilade
6	Probability of detection ($P\infty$) coefficient [a] of target category heavy vehicles, fully exposed
7	Probability of detection ($P\infty$) coefficient [a] of target category heavy vehicles, hull defilade
8	Probability of detection ($P\infty$) coefficient [a] of target category artillery, fully exposed
9	Probability of detection ($P\infty$) coefficient [a] of target category artillery, hull defilade
10	Probability of detection ($P\infty$) coefficient [a] of target category helicopters, fully exposed
11	Probability of detection ($P\infty$) coefficient [a] of target category helicopters, hull defilade

12-21 Probability of detection (P_∞) coefficient [b] with the same format as 2-11.

22-31 Probability of detection (P_∞) coefficient [c] with the same format as 2-11.

32-41 Average time to detect ($\overline{T}$) coefficient [a'] with the same format as 2-11.

42-51 Average time to detect ($\overline{T}$) coefficient [b'] with the same format as 2-11.

52-61 Average time to detect ($\overline{T}$) coefficient [c'] with the same format as 2-11.

62-69 Probability of detection range minimum [$rmin_a$] for each atmospheric condition. (Day/Night and Visibility) (e.g., 62-Day, >5 KM, 63-Day, 5 KM, 64-Day, 2 KM, 65-Day, 1 KM, 66-Night, >5 KM, etc)

70-77 Probability of detection range maximum [$rmax_a$] for each atmospheric condition. (Day/Night and Visibility) (e.g., 70-Day, >5 KM, 71-Day, 5 KM, 72-Day, 2 KM, 73-Day, 1 KM, 74-Night, >5 KM, etc)

C. Helicopter performance against targets files.

There is one helicopter performance file for each force. Each file contains fifteen records for unique types of munitions. These records contain data describing munition lethalities (Pk's) and the performance parameters of the helicopters used in the tactics of munition delivery. Record entries for Pk's represent fitted coefficients in the equation

$$Pk = ar^2 + br + c \text{ where } r = \text{range in meters}$$

Each munition record contains 131 elements. The structure of each record is as follows:

Index	Description
1	Munition description string (eight characters).
2-41	Probability of kill (Pk) coefficient [a] for each of the 20 target categories, fully exposed and hull defilade. (e.g., 2-Target Category 1 Fully Exposed, 3-Target Category 1 Hull Defilade, 4-Target Category 2 Fully Exposed, etc.)

Index	Description
42-81	Probability of kill (Pk) coefficient [b] for each of the 20 target categories, fully exposed and hull defilade. (e.g., 42-Target Category 1 Fully Exposed, 43-Target Category 1 Hull Defilade, 44-Target Category 2 Fully Exposed, etc.)
82-121	Probability of kill (Pk) coefficient [c] for each of the 20 target categories, fully exposed and hull defilade. (e.g., 82-Target Category 1 Fully Exposed, 83-Target Category 1 Hull Defilade, 84-Target Category 2 Fully Exposed, etc.)
122	Probability of kill (Pk) range minimum [rmin]
123	Probability of kill (Pk) range maximum [rmax]
124	Tactical number of rounds fired per pop up
125	Fire and guide time [fm].
126-128	Time masked [Tm_j] for each mission in the pop up / pop down cycle. (e.g., 126-air to ground, 127-air to air, 128-SEAD)
129-131	Time exposed [Te_j] for each mission in the pop up / pop down cycle. (e.g., 129-air to ground, 130-air to air, 131-SEAD)

D. Helicopter target preference files.

There is one helicopter target preference file for each force. Each file contains three records, one for each mission, with record 1 = air to ground, record 2 = air to air, and record 3 = SEAD. Each mission record contains 28 elements. The files contain numeric weights representing helicopter preferences for targets and the parameters α, β used to approximate line of sight between helicopters and their targets. The parameters α, β are used in the following form

$$Plos = \alpha(e^{-\beta r})$$

The structure of each record is as follows:

Index	Description
1-20	Preferences for each target category [D_{ijk}].
21-24	Probability of line of sight alpha [α] for each terrain (open, rolling, hilly, mountainous)

Index	Description
25-28	Probability of line of sight beta [β] for each terrain. (open, rolling, hilly, mountainous)

E. Helicopter vulnerability against air defense files.

There is one helicopter vulnerability file for each force. Each file contains seven records; one for each air defense weapon. Each air defense weapon record contains thirty-nine elements describing the ability of the AD weapon to detect and kill the helicopters. Detection parameters include probability of detection and time to detect. The lethality parameters represent PKs. The parameters in the file represent fitting coefficients and are as described under the helicopter lethalities file. The structure of each record is as follows:

Index	Description
1-2	Probability of detection (P_∞) coefficient [a] for mast mounted and non-mast mounted.
3-4	Probability of detection (P_∞) coefficient [b] for mast mounted and non-mast mounted.
5-6	Probability of detection (P_∞) coefficient [c] for mast mounted and non-mast mounted.
7-8	Average time to detect $\overline{T}$ coefficient [a'] for mast mounted and non-mast mounted.
9-10	Average time to detect $\overline{T}$ coefficient [b'] for mast mounted and non-mast mounted.
11-12	Average time to detect $\overline{T}$ coefficient [c'] for mast mounted and non-mast mounted.
13-20	Probability of detection range minimum [$rmin_a$] for each atmospheric condition. (Day/Night and Visibility) (e.g., 13-Day, >5 KM, 14-Day, 5 KM, 15-Day, 2 KM, 16-Day, 1 KM, 17-Night, >5 KM, etc)
21-28	Probability of detection range maximum [$rmax_a$] for each atmospheric condition. (Day/Night and Visibility) (e.g., 21-Day, >5 KM, 22-Day, 5 KM, 23-Day, 2 KM, 24-Day, 1 KM, 25-Night, >5 KM, etc)
29-30	Mast mounted and non-mast mounted probability of kill coefficient [a].

Index	Description
31-32	Mast mounted and non-mast mounted probability of kill coefficient [b].
33-34	Mast mounted and non-mast mounted probability of kill coefficient [c].
35	Probability of kill range minimum [rmin].
36	Probability of kill range maximum [rmax].
37-39	Preferences of AD weapon for each enemy helicopter [H].

F. Air defense miscellaneous files.

There is one air defense miscellaneous file for each force. Each file contains seven records, one for each air defense weapon. Each air defense weapon record contains 3 elements. The structure for each record is as follows:

Index	Description
1	Weight per round (lbs).
2	Rounds per engagement.
3	Flyout velocity of the munition [Fad] in meters/sec.

G. Direct fire sensor performance files.

There is one direct fire sensor performance file for each force. Each file contains two records, the first for optical ground sensors and the second for thermal ground sensors. Each sensor record contains 28 elements representing the probability of detection P and the average time to detect T. The parameters in the file represent fitting coefficients and are as described under the helicopter sensor performance files. The structure of each file is as follows.

Index	Description
1-2	Probability of detection (P_∞) coefficient [a] for mast mounted and non-mast mounted helicopters.

Index	Description
3–4	Probability of detection ($P\infty$) coefficient [b] for mast mounted and non-mast mounted helicopters.
5–6	Probability of detection ($P\infty$) coefficient [c] for mast mounted and non-mast mounted helicopters.
7–8	Average time to detect $\overline{T}$ coefficient [a'] for mast mounted and non-mast mounted helicopters.
9–10	Average time to detect $\overline{T}$ coefficient [b'] for mast mounted and non-mast mounted helicopters.
11–12	Average time to detect $\overline{T}$ coefficient [c'] for mast mounted and non-mast mounted helicopters.
13–20	Probability of detection range minimum [$rmin_a$] for each atmospheric condition (Day/Night and Visibility - e.g., 13-day, > 5 km, 14-day, 5 km, 15-day, 2 km, 16-day, 1 km, 17-night, > 5 km, etc.)
21–28	Probability of detection range maximum [$rmax_a$] for each atmospheric condition (same format as probability of detection range minimum above).

I. Direct fire miscellaneous files.

There is one direct fire miscellaneous file for each force. Each file contains two records, the first for ground missile data, and the second for ground kinetic energy round data. Each record contains two elements. The structure for each record is as follows:

Index	Description
1	Rounds fired per engagement
2	Flyout velocity of the munition [Fvdf] in meters/second

5. ALGORITHMS

A. Figure 6-33 presents a generalized logic flow of the processes occurring in the P4 helicopter module. The diagram provides a framework for the algorithms used in the module.

B. The geometry for the helicopter module is assumed to be as follows:

(1) For air to ground missions, the helicopters will play in-line with the ground forces.

(2) For SEAD missions, the attacking helicopters will flank the ground forces.

(3) For air to air missions, the attacking helicopter will maximize the range to opposing ground forces.

The following definitions are used:

R_R = The input range by the Red Helicopter player

R_B = The input range by the Blue Helicopter player

R_f = The range between the forces

B = Centroid of the Blue Force

R = Centroid of the Red Force

r = range used by the model when calculating Pk, P_∞, $\overline{T}$, etc. for helicopter to helicopter engagements (Table 6-Va) and helicopter vs ground engagements (Tables 6-Vb & 6-Vc).

The range calculations used in the helicopter module are illustrated in tables 6-Va through 6-Vc.

C. The following paragraphs provide a detailed description of the algorithms used for the attrition of target elements due to helicopters. Attrition to helicopters from air defense and direct fire elements is also calculated.

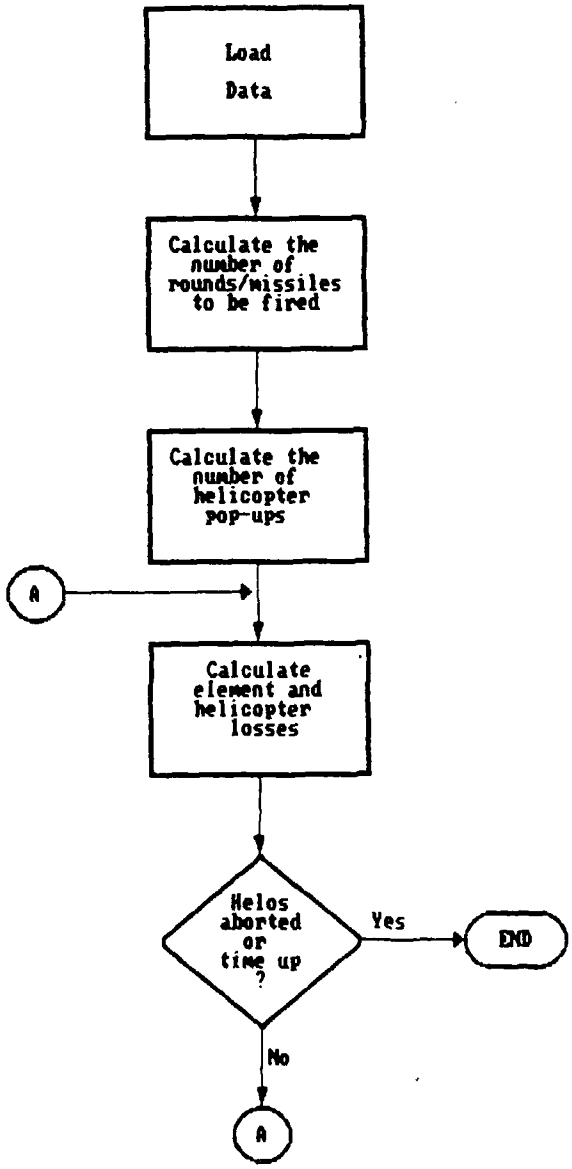


Figure 6-33. Generalized logic flow of helicopter attrition module.

Helicopter to Helicopter (r)

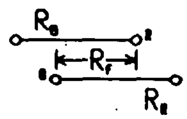
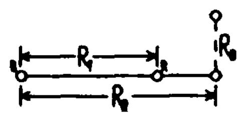
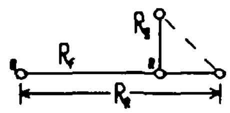
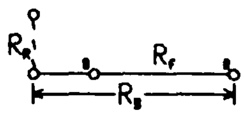
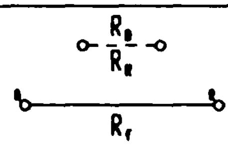
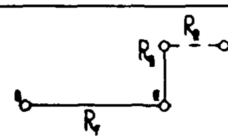
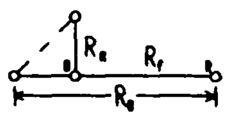
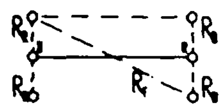
RED MISSION \ BLUE MISSION	Direct Support	Helo to Helo	SEAD
Direct Support	 $ABS(R_B + R_R - R_r)$	 R_B	 $\sqrt{(R_R - R_r)^2 + R_B^2}$
Helo to Helo	 R_R	 $\frac{R_R + R_B}{2}$	 R_B
SEAD	 $\sqrt{(R_B - R_r)^2 + R_R^2}$	 R_B	 $\frac{1}{2}R_r + \frac{1}{2}\sqrt{R_r^2 + (R_B + R_R)^2}$

TABLE 6-Va

Blue Ground to Red Helo Ranges (r

RED MISSION

BLUE MISSION

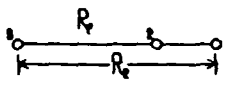
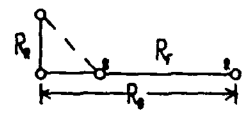
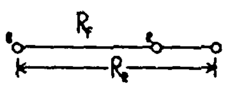
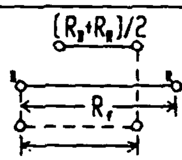
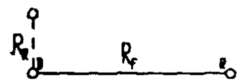
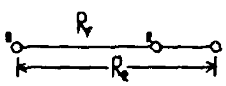
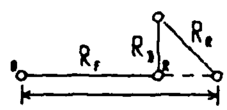
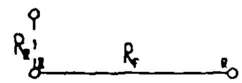
	Direct Support	Helo to Helo	SEAD
Direct Support	 R_e	 $\sqrt{(R_s-R_f)^2+R_e^2}$	 R_e
Helo to Helo	 R_e	 $\frac{R_f}{2}+\frac{R_s+R_e}{4}$	 R_e
SEAD	 R_e	 $R_f+\sqrt{R_e^2-R_s^2}$	 R_e

TABLE 6-Vb

Red Ground to Blue Helo Ranges(r

BLUE MISSION (columns); RED MISSION (rows)

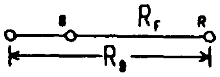
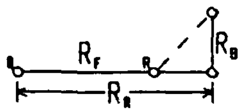
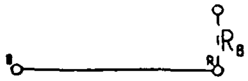
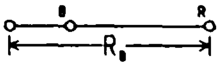
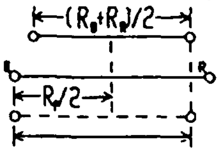
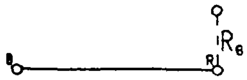
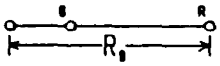
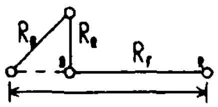
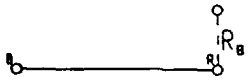
RED MISSION	Direct Support	Helo to Helo	SEAD
Direct Support	 R_B	 $\sqrt{(R_B-R_f)^2+R_B^2}$	 R_B
Helo to Helo	 R_B	 $\frac{R_f}{2}+\frac{R_B+R_R}{4}$	 R_B
SEAD	 R_B	 $R_f+\sqrt{R_B^2-R_R^2}$	 R_B

TABLE 6-Vc

(1) Number of helicopter munitions fired during the time step.

The number of rounds fired during the timestep per helicopter are:

$$R_{ij} = \overline{R}p_{ij} * Put_i * Fdf_{ij} \qquad \text{(Eq. 6-40)}$$

where:

$\overline{R}p_{ij}$ = Average rounds fired at target category j per pop-up by helicopter i

Put_i = Number of pop-ups by helicopter i during the assessment interval

Fdf_{ij} = Fire distribution factor of rounds from firer i to target j

(i) For air to ground:

$$\overline{R}p_{ij} = Pdt(r,t)_{ij} * Plos(r,ga)_k * Np \qquad \text{(Eq. 6-41)}$$

(ii) For air to air:

$$\overline{R}p_{ij} = P(engage)_{ij} * Np \qquad \text{(Eq. 6-41a)}$$

where:

$Pdt(r,t)_{ij}$ = Probability of detection of target j which falls into detection category t by the firer i. The detection category is described under equation 6-42.

$Plos(r,ga)_k$ = Probability that the helicopter has line of sight to the opposing force in terrain type k at range r. The target may be either ground or air (ga).

Np = Tactical number of rounds fired by a helicopter when engaging a target.

$P(engage)_{ij}$ = Probability that helicopter i will engage target helicopter j

(i) Both $Pdt(r,t)_{ij}$ and $Plos(r,ga)_k$ are functions of range. They have the following forms.

$$Pdt(r,t)_{ij} = Pdfe(r,t)_a * Pfe + Pdhd(r,t)_a * Phd \qquad \text{(Eq. 6-42)}$$

where:

$Pdfe(r,t)_a$ = Probability of helicopter i detecting a fully exposed target of type t at range r under atmospheric conditions a. The target categories are personnel, light vehicles, heavy vehicles, artillery, and helicopters. The subscript a represents the atmospheric conditions. Eight conditions can be played: 1 km, 2 km, 5 km, >5 km for both day and night.

If $rmin_a \leq r \leq rmax_a$:

$$Pdfe(r,t)_a = P_\infty(r,t)\left(1 - e^{(-Ut(r)/\overline{T}(r,t))}\right) \quad \text{(Eq. 6-43)}$$

Otherwise, $Pdfe(r,t)_a = 0$.

where $P\infty(r,t)$ = $a_t r^2 + b_t r + c_t$ is the probability of detecting the target of category t at range r when searching infinite time. a_t, b_t, c_t are fit parameters for target type t.

$T(r,t)$ = $a_t r^2 + b_t r + c_t$ is the average time to detect a target. a_t, b_t, c_t are fit parameters for target type t.

$Ut(r)$ = the time the helicopter is exposed (te_{ik}) minus its fire and guide time. $te_{ik} = r / fm$ (fm is munition dependent for a missile, gun or air defense)

$rmin_a$ and $rmax_a$ are bounds based on the day, night and atmospheric type.

$Pdhd(r,t)_a$ = probability of detection for hull defilade targets. The definition of the structure is analogous to $Pdfe(r,t)_a$.

Pfe = percent of the targeted force fully exposed.

Phd = percent of the targeted force in hull defilade.

$$Plos(r,ga)_k = \alpha\left(e^{-\beta r}\right) \quad \text{(Eq. 6-44)}$$

where:

α and β are fitting factors for probability of line of sight in four different types of terrain k for helicopters searching for ground targets and helicopters searching for air targets and in SEAD missions.

Popups for this time interval can be described as

$$Put_i = \frac{ts}{tm_{ik} + te_{ik}} \qquad \text{(Eq. 6-45)}$$

where:

Put_i = number of pop-ups this time interval for firer i
ts = seconds represented in the attrition step
tm_{ik} = the average time masked per firing cycle for helicopter i for mission k

te_{ik} = the average time exposed for helicopter i for mission k

k = 1 air to ground
2 air to air
3 SEAD for helicopter type j

The tm_{ik} and te_{ik} values that are used in calculations are determined by helicopter i's primary mission. If helicopter i is on an air to ground or SEAD mission, the tm_{ik} and te_{ik} of the conventional munitions (guns or missiles) are used. If it is on an air to air mission, the tm_{ik} and te_{ik} of the air munitions are used.

(ii) Equation 6-41a is basically the same as equation 6-41 with

$$P(\text{engage})_{ij} = P(\text{Exposure time}_i > \phi(r)) * P(\text{Detection time}_{ij} < \text{Exposure time}_i - \phi(r)) \qquad \text{(Eq. 6-46)}$$

where:

$$\phi(r) = r/fi \qquad \text{(Eq. 6-47)}$$

r is the target range and fi is the firer munition pinpoint and flyout time in meters/second.

The difference in the equations arises because the target and firer are popping up and down with their own frequencies. To solve the equations, the following assumptions are made:

(a) The pop-up, pop-down process is an alternating Markov process with the durations of the alternating up, down states negative exponential random variables.

(b) The time to acquire a target given it is continuously visible is an exponentially distributed random variable.

(c) The line of sight process between each target-firer pair is independent of all other pairs.

(d) Once a firer helicopter i has begun an engagement of a helicopter target j, it will continue until the engagement is complete, regardless if any other helicopter is firing at the same target.

P(Exposure) then becomes the exponentially distributed density function:

$$P(t) = \xi_i * e^{-\xi_i t} \quad \text{where } 0<t \qquad \text{(Eq. 6-48)}$$

and

$$\xi_i = 1 / \overline{te}_{ik} \qquad \text{(Eq. 6-49)}$$

where $\overline{te}_{ik}$ = the exposure time for firer i while it is performing mission k.

$DET_{ij}(t)$ is the probability that firer i will have detected target helicopter j after searching (t) seconds. $DET_{ij}(t)$ has the following form under the assumption of completed firing engagement (from "Vector-2 System of Theater Level Combat" DDC number ADB037799):

$$DET_{ij}(t) = 1 - \left[1 - (Z_{ij} / (\mu_j - Z_{ij}))(e^{-Z_{ij}t} - e^{-\mu_i t}) \right]^{N_j} \qquad \text{(Eq. 6-50)}$$

where:

N_j = number of target helicopters j on site
Z_{ij} = the rate firer i detects target j, with j moving in and out of the line of sight of i.

$$Z_{ij} = \frac{\lambda_{ij} * \eta_{ij}}{\eta_j + \mu_j} \qquad \text{(Eq. 6-51)}$$

$$\eta_j = (1 / \overline{tm}_{jk}) \qquad \text{(Eq. 6-52)}$$

$$\mu_j = (1 / \overline{te}_{jk}) \qquad \text{(Eq. 6-53)}$$

Note that $\bar{t}mjk$ and $\bar{t}ejk$ are the mask and exposure times of the target helicopter j and are determined by the same rules as in equation 6-45.

λ_{ij} = the rate at which helicopter i detects helicopter j

$$\lambda_{ij} = \frac{P_{\infty}(r,t)\left[1 - e^{-(te_{ik} - r / fm)/\overline{T}(r,t)}\right]}{\overline{T}(r,t)} \quad \text{(Eq. 6-54)}$$

Where $P\infty(r,t)$ and $\overline{T}(r,t)$ are as described in equation 6-43.

Also note that the dividend of λ_{ij} is the same calculation as $Pdfe(r,t)_a$ (equation 6-43).

Then:

$$P(engage)_{ij} = \int_{\phi(r)}^{\infty} \xi_i * e^{-\xi_i t} * DET_{ij}(t - \phi(r))dt$$

Substituting in equation 6-50 we get:

$$P(engage)_{ij} = \int_{\phi(r)}^{\infty} \xi_i * e^{-\xi_i t} \left[1 - \left[1 - (z_{ij} / (\mu_j - z_{ij})) * \left(e^{-z_{ij}(t - \phi(r))} - e^{-\mu_j(t - \phi(r))}\right)\right]\right]^{N_j}$$

The approximation used to numerically solve this equation is:

$$P(\text{engage})_{ij} = \sum_{t=\phi(r)}^{-\frac{\ln .1}{\xi_i}} \xi_i * e^{-\xi_i t} * DET_{ij}(t - \phi(r))$$

(Eq. 6-55)

(This will account for at least 90% of the density of the firer exposure time.)

(2) Probability of kill for Target j by Helicopter i.

The probability of kill for this timestep is given by:

$$Pk_{ij} = Pkd_{ij}(r) * Phd + Pke_{ij}(r) * Pfe \qquad \text{(Eq. 6-56)}$$

where:

$Pkd_{ij}(r)$ = the probability of kill for target category j by helicopter i for defilade targets at range r. Note that the target category (1-20) has been used.

$Pke_{ij}(r)$ = probability of kill for target category j by helicopter i for fully exposed targets at range r. Again the target category (1-20) has been used.

$Pkd_{ij}(r)$ and $Pke_{ij}(r)$ have the following forms: If $rmin \leq r \leq rmax$, then

$$Pkd_{ij}(r) \text{ and } Pke_{ij}(r) = ar^2 + br + c \qquad \text{(Eq. 6-57)}$$

Otherwise they equal 0.

The variables rmin and rmax are a function of target category and helicopter munition combination. (Each helicopter can be equipped with a maximum load of two air to ground munitions, and one air to air munitions. It can be loaded explicitly with fewer munitions.)

Phd and Pfe are respectively the percent of the target force in hull defilade or fully exposed. Note that these are the same percentages as described under probability of detection.

(3) Fire Distribution Factor for Target j by Helicopter i.

The fire distribution factor, Fdf_{ij}, of rounds fired by helicopter i at target category j can be calculated using the following:

$$Fdf_{ij} = \frac{Pdt(r,t)_{ij} * tgt_j * Pke_{ij}(r) * D_{ijk}}{\sum_{j=1}^{20} Pdt(r,t)_{ij} * tgt_j * Pke_{ij}(r) * D_{ijk}} \qquad \text{(Eq. 6-58)}$$

(sum over all target categories)

where:

D_{ijk} = a weighted factor representing the preferred distribution of firer i vs category j for mission k.
tgt_j = number of target elements in category j being engaged by firer i.

$Pdt(r,t)_{ij}$ and $Pke_{ij}(r)$ are as described in previous paragraphs.

(4) Determine Actual Number of Air and Conventional Rounds Fired Based on Ammunition Constraints.

(a) $$Pg_i = \sum_{j=1}^{17} Fdf_{ij} \qquad \text{(Eq. 6-59)}$$

$$Pa_i = \sum_{j=18}^{20} Fdf_{ij}$$

where:

Pg_i = desired percent of time firing at ground targets
Pa_i = desired percent of time firing at air targets.
Fdf_{ij} is as described in Equation 6-58.

(b)

$$Cnvpop_i = \sum_{j=1}^{17} (\overline{R}p_{ij} * Fdf_{ij}) / Pg_i \qquad \text{(Eq. 6-60)}$$

$$Airpop_i = \sum_{j=18}^{20} (\overline{R}p_{ij} * Fdf_{ij}) / Pa_i$$

where:

$Cnvpop_i$ = total number of conventional munitions fired per popup by firer i at all ground targets.
$Airpop_i$ = total number of air missiles fired per popup by firer i at all air targets.

$\overline{R}p_{ij}$ and Fdf_{ij} are as described in Equations 6-41 and 6-58, respectively.

Note that the tradeoff rate of air to air/ground is $Airpop_i/Cnvpop_i$.

(c)

$$Rg_i = \sum_{j=1}^{17} \overline{R}p_{ij} * Put_i * Fdf_{ij} \qquad \text{(Eq. 6-61)}$$

$$Rh_i = \sum_{J=18}^{20} \overline{R}p_{ij} * Put_i * Fdf_{ij}$$

where:

Rg_i = total number of rounds fired at ground targets by firer i.
Rh_i = total number of air missiles fired at air targets by firer i.

$\overline{R}p_{ij}$, Put_i, and Fdf_{ij} are as described in Equations 6-41, 6-45, and 6-58, respectively.

These equations can also be written as:

$$Rg_i = \sum_{j=1}^{17} R_{ij} \qquad \text{(Eq. 6-61a)}$$

$$Rh_i = \sum_{j=18}^{20} R_{ij}$$

Where R_{ij} is as described in Equation 6-40.

(d) Determine rounds based on ammunition constraints.

Rab_i = basic load of air munitions for firer i.
Rcb_i = basic load of conventional munitions for firer i.

Case I: Enough air and ground rounds on board to fire the desired number of rounds:

$$Rh_i \leq Rab_i \text{ and } Rg_i \leq Rcb_i$$

then:

$$Rair_i = Rh_i \text{ and } Rconv_i = Rg_i$$

where:

$Rair_i$ = actual total number of air rounds fired based on ammo constraints.
$Rconv_i$ = actual total number of ground rounds fired based on ammo constraints.

Case II: Not enough air rounds on board:

$$Rh_i > Rab_i$$

$$R'g_i = (Rh_i - Rab_i) * (Cnvpop_i / Airpop_i) \qquad \text{(Eq. 6-62)}$$

where $R'g_i$ = amount of conventional rounds needed to make up insufficient air rounds.

IIa.

If there are enough conventional rounds on board:

$$R'g_i + Rg_i \leq Rcb_i$$

then:

$$Rair_i = Rab_i$$
$$Rconv_i = R'g_i + Rg_i$$

IIb.

If there are not enough conventional rounds on board:

$$R'g_i + Rg_i > Rcb_i$$

then

$$Rair_i = Rab_i$$
$$Rconv_i = Rcb_i$$

and the number of popups is reduced:

$$Put_i = Put_i * ((Rair_i + Rconv_i * (Airpop_i/Cnvpop_i)) / (Rh_i + Rg_i * Airpop_i/Cnvpop_i))) \quad \text{(Eq. 6-63)}$$

Case III: Not enough ground rounds on board:

$$Rg_i > Rcb_i$$

$$R'h_i = (Rg_i - Rcb_i) * (Airpop_i / Cnvpop_i) \quad \text{(Eq. 6-62a)}$$

where $R'h_i$ = amount of air missiles needed to make up insufficient ground rounds.

IIIa.

If there are enough air munitions on board:

$$R'h_i + Rh_i \leq Rab_i$$

then

$$Rair_i = R'h_i + Rh_i$$
$$Rconv_i = Rcb_i$$

IIIb.

If there are not enough air munitions on board:

$$R'h_i + Rh_i > Rab_i$$

then

$$Rair_i = Rab_i$$
$$Rconv_i = Rcb_i$$

and the number of popups is reduced.

$$Put_i = Put_i * ((Rair_i * (Cnvpop_i / Airpop_i) + Rconv_i)) / (Rh_i * (Cnvpop_i / Airpop_i) + Rg_i)) \quad \text{(Eq. 6-63a)}$$

$Rair_i$ and $Rconv_i$ are then used to scale down the actual number of rounds fixed at each target j, if needed.

(5) Number of Rounds Fired by the Air Defense Elements During the Time Step (R_{ij}).

The rounds fired during the time step per air defense element are

$$R_{ij} = \overline{Rap}_{ij} * Put_j * Fdad_{ij} \quad \text{(Eq. 6-64)}$$

where:

$\overline{Rap}_{ij}$ = the rounds fired by the AD element at helicopter j each time the helicopter pops up
Put_j = the number of pop ups for helicopter j during this time period
$Fdad_{ij}$ = fire distribution factor of rounds from AD firer i to target type j.

Note that:

$$\overline{Rap}_{ij} = Pad(r,t)_{ij} * Plos(r,ga)_k \quad \text{(Eq. 6-65)}$$

$Pad(r,t)_{ij}$ = the probability that the air defense element i can detect and engage a helicopter of type j in time T with mission ga.
$Plos(r,ga)_k$ = probability that AD element at range r will have line of sight to the helicopter in mission ga in terrain k

If $rmin_a \leq r \leq rmax_a$:

$$Pad(r,t)_{ij} = P_{\infty}(r,j)\left(1 - e^{-(T(r) / \overline{T}(r,t))}\right) \qquad \text{(Eq. 6-66)}$$

where:

$P\infty(r,j)$ = the probability that air defense weapon n can detect helicopter j at range r given it can search for an infinite period under atmospheric conditions a.

Note that:

$$P\infty(r,j) = ar^2 + br + c \qquad \text{(Eq. 6-67)}$$

where a, b, c are fitting parameters and r is the range in meters

$\overline{T}(r,j)$ = the average time to detect helicopter type j at range r.

$$\overline{T}(r,j) = a\overline{T}r^2 + b\overline{T}r + c\overline{T}$$

where $a\overline{T}$, $b\overline{T}$, $c\overline{T}$ are fitting parameters and are as described above.

T(r) is the time the AD system has to detect the helicopter given an engagement will occur following detection.

$$T(r) = te_j - r / fad_i \qquad \text{(Eq. 6-68)}$$

where:

te_j = the exposure time for helicopter j
fad_i = the flyout velocity of the air defense munition (m/sec).

$$Plos(r,ga)_k = \alpha\left(e^{-\beta * r}\right) \quad \text{(Eq. 6-69)}$$

where:

α, β are fitting factors for probability of line of sight in 4 different terrains.

(6) Probability of Kill for Helicopter j by AD element i.

The probability of kill against helicopter j if rmin $\leq$ r $\leq$ rmax is

$$Pk_{ij}(r) = a_{ij}r^2 + b_{ij}r + c_{ij} \quad \text{(Eq. 6-70)}$$

Otherwise, it is 0.

Variables a_{ij}, b_{ij} and c_{ij} are fitting parameters for Pk's of an AD element. Variables rmin and rmax also represent the effective range envelope of the munition.

(7) Fire Distribution Factor for Helicopter j by AD element i.

$$Fdad_{ij} = \frac{Pad(r,t)_{ij} * Tgt_j * Pk_{ij}(r) * H_{ij}}{\sum_{\text{all } j}^{\text{Helicopter types}} Pad(r,t)_{ij} * Tgt_j * Pk_{ij}(r) * H_{ij}} \quad \text{(Eq. 6-71)}$$

where:

Tgt_j = number of helicopter j targets being engaged by AD i.

H_{ij} = a preference of AD element i for helicopter type j.

$Pad(r,t)_{ij}$ and $Pk_{ij}(r)$ are as described in previous paragraphs.

(8) Number of Rounds Fired by the Direct Fire Elements During the Time Step (R_{ij}).

The rounds fired during the time step per direct fire element are:

$$R_{ij} = \overline{Rdp}_{ij} * Put_j * Fdf_{ij} \qquad \text{(Eq. 6-72)}$$

where:

$\overline{Rdp}_{ij}$ = the rounds fired by the direct fire element i at helicopter j per popup.

Put_j = the number of popups for helicopter j during this time step.

Fdf_{ij} = fire distribution factor from direct firer i to helicopter target j. These factors are calculated in the direct fire portion of ground combat (P4) and passed to the helicopter attrition module (Helo_kills) through the common block Helo_info.

Note that:

$$\overline{Rdp}_{ij} = Pdfdt_{ij} * Plos(r,ga)_k \qquad \text{(Eq. 6-73)}$$

where:

$Pdfdt_{ij}$ = the probability that direct fire element i can detect and engage helicopter j.

$Plos(r,ga)_k$ is as described in previous paragraphs.

If $rmin_a \leq r \leq rmax_a$:

$$Pdfdt_{ij} = P_\infty(r,j)\ (1 - e^{-(Ut(r)/\overline{T}(r,t))}) \qquad \text{(Eq. 6-74)}$$

where:

$P_\infty(r,j)$ = the probability that direct fire elements can detect helicopter j at range r given it can search for an infinite period under atmospheric conditions a.

$$P_{\infty}(r,j) = ar^2 + br + c \qquad \text{(Eq. 6-75)}$$

where:

Variables a, b, c are fitting factors for a sensor type and r is the range between direct fire elements and helicopter j in meters.

Ut(r) = the time the direct fire system has to detect the helicopter given an engagement does occur.

$$Ut(r) = te_j - r/fvdf \qquad \text{(Eq. 6-76)}$$

where:

te_j = helicopter j's exposure time.
fvdf = flyout velocity of the direct fire munition (m/sec).

(9) Probability of Kill for Helicopter j by Direct Fire i.

The probability of kill (Pk_{ij}) against helicopter j is retrieved from the direct firer's PK files. The PK values are dependent on range, with the 6 records in the PK files representing 500 meter range bands. The values in the range band within which the helicopter to ground range falls are the ones stored and used in the helicopter attrition section.

(10) Fire Distribution Factor for Helicopter j by DF element i.

The fire distribution factor, Fdf_{ij} from direct firer i to helicopter target j is calculated in the direct fire portion of the ground combat (P4) routine. See Equation 6-35 for more details. These values are then stored in the common block Helo_info and used in the helicopter attrition module.

(11) Attrition of Helicopters, Air Defense, and Direct Fire.

To determine the helicopter losses, first the probabilities of survival for helicopter target j being fired upon by enemy helicopters, air defense and direct fire elements are calculated separately:

$$P_{ij} = (1 - (Pk_{ij} / Tgt_j))^{R_{ij}} \qquad \text{(Eq. 6-77)}$$

where:

P_{ij} = the probability of survival for the jth target helicopter being fired upon by the ith firer, where firers are AD elements, DF elements and attack helicopters.

Pk_{ij}, Tgt_i, and R_{ij} are as described in previous paragraphs.

P_{ij} is then accumulated for all firers.

$$Ps_j = \underbrace{\prod_{i=1}^{7} P_{ij}}_{\text{all AD elements}} * \underbrace{\prod_{i=1}^{20} P_{ij}}_{\text{all DF elements}} * \underbrace{\prod_{i=1}^{3} P_{ij}}_{\text{all attack helos}} \qquad \text{(Eq. 6-78)}$$

where:

Ps_j = the probability of survival for the jth target helicopter over all AD elements, DF elements and attack helicopters of type i.

Losses per target category are then calculated.

$$Loss_j = (1-Ps_j) * Tgt_j \qquad \text{(Eq. 6-79)}$$

where

$Loss_j$ = losses of target helicopter j.

The losses to the ground targets are calculated by:

$$Lossg_j = (1- (1- (Pk_{ij} / Tgt_j))^{R_{ij}})* Tgt_j \qquad \text{(Eq. 6-80)}$$

where:

$Lossg_j$ = losses of ground target category j.
Tgt_j = number of target elements in category j.
Pk_{ij} = probability of kill of target category j by firer type i
R_{ij} = rounds fired by i against target category j during this timestep

6. "UNITFILE" IMPACT

The unit status file ("UNITFILE") is not directly affected by the helicopter module. The information is returned to the ground combat mainline where it then affects the "UNITFILE".

7. CODE.

The helicopter subroutine code is explicitly depicted in the flow diagram in Figure 6-34. Notice that the routine contains numerous loops, checks and subroutines.

A listing of major variables by subroutine is found in Table 6-10. Table 6-14 contains a listing of the ground combat code,

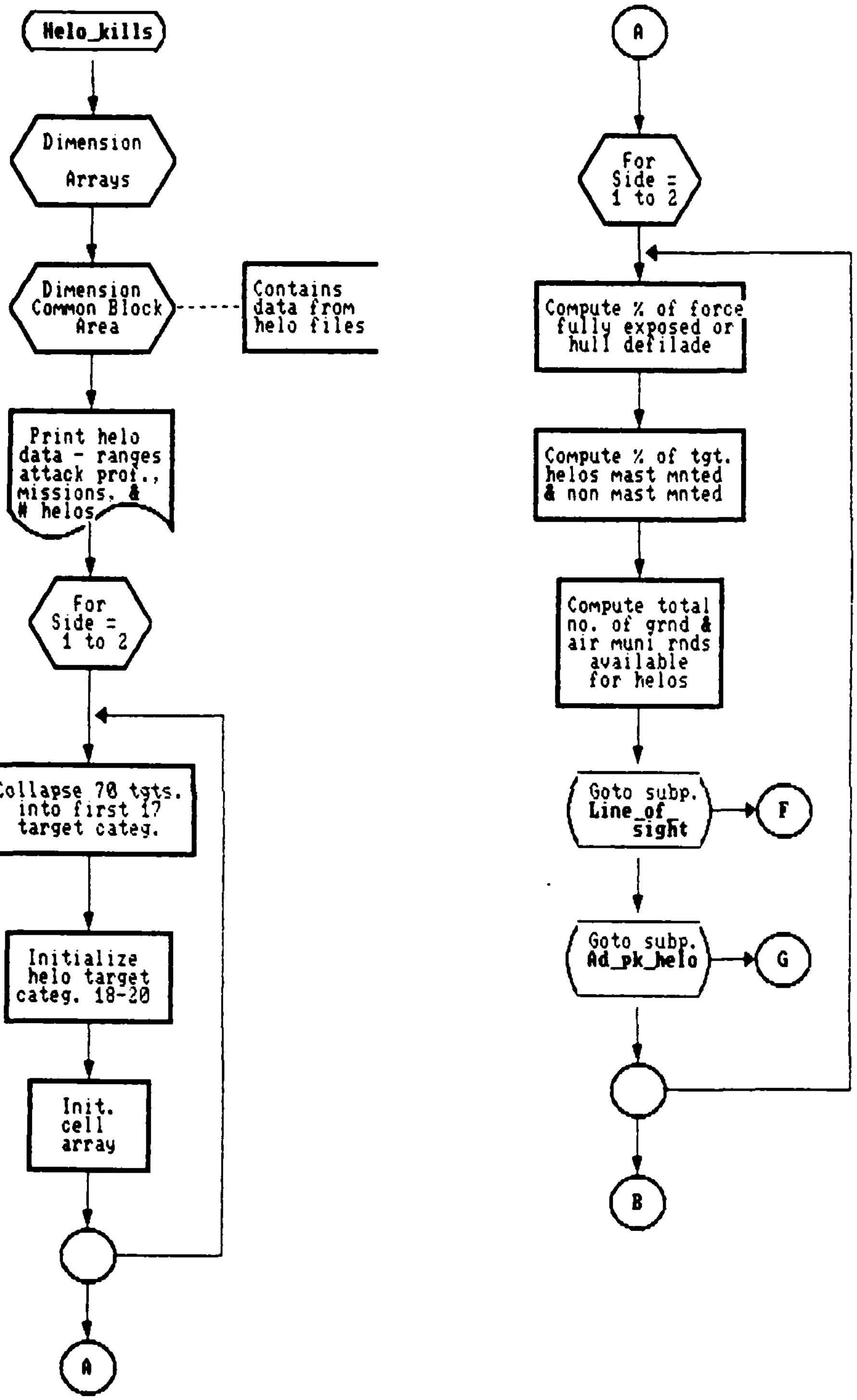


Figure 6-34. Functional flow of helicopter attrition module.

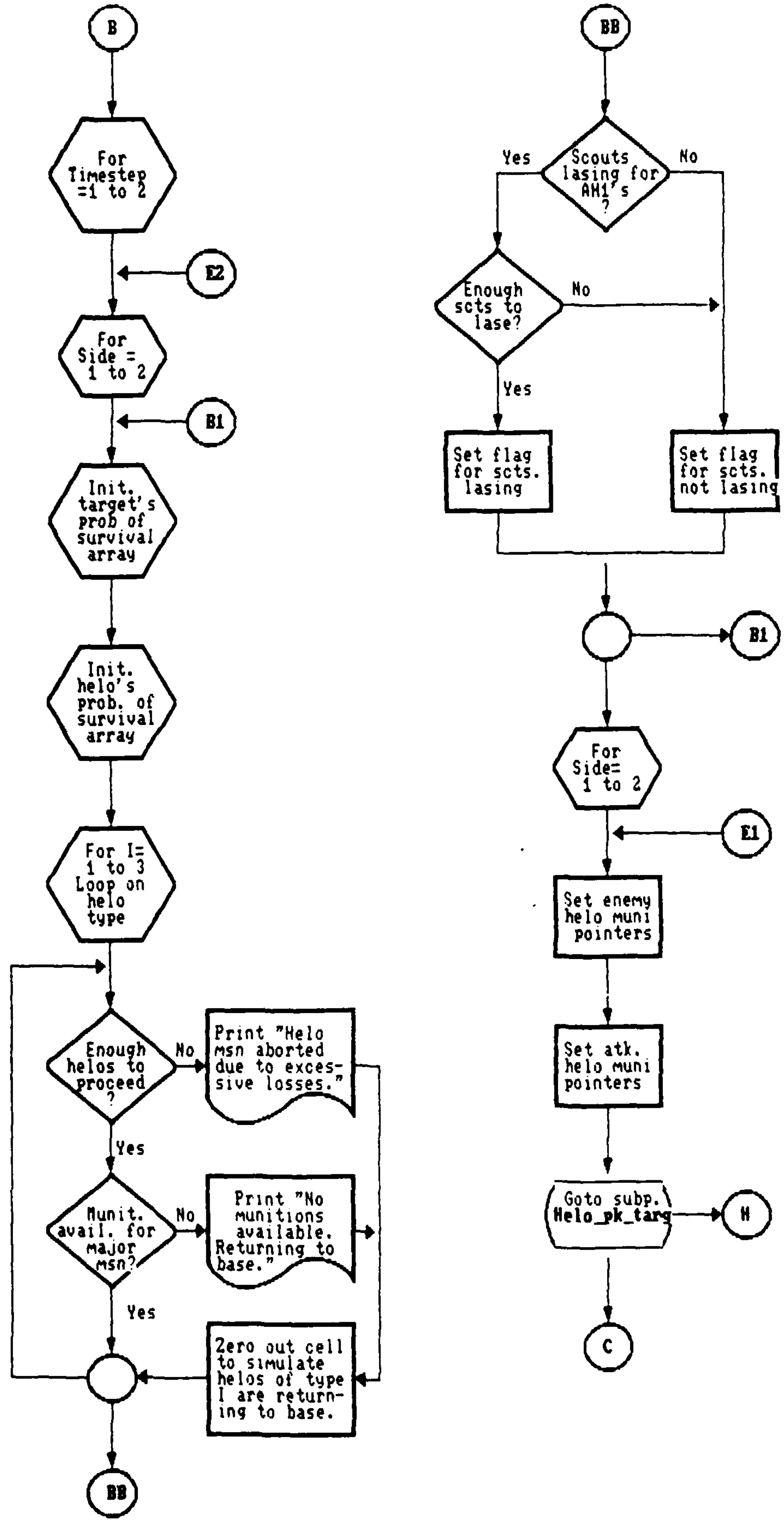


Figure 6-34. Functional flow of helicopter attrition module (continued).

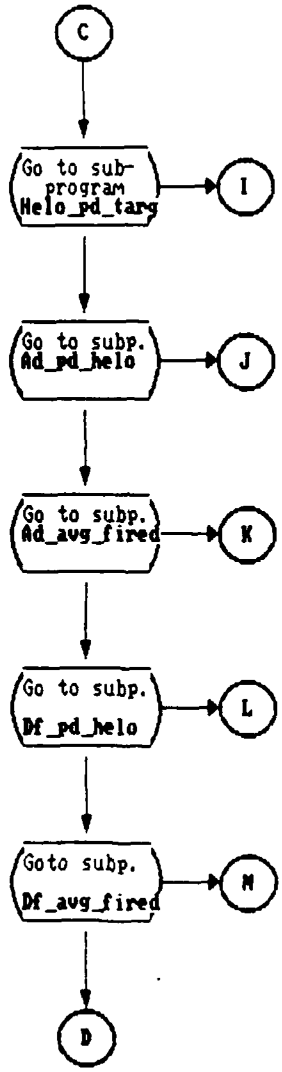


Figure 6-34. Functional flow of helicopter attrition module (continued).

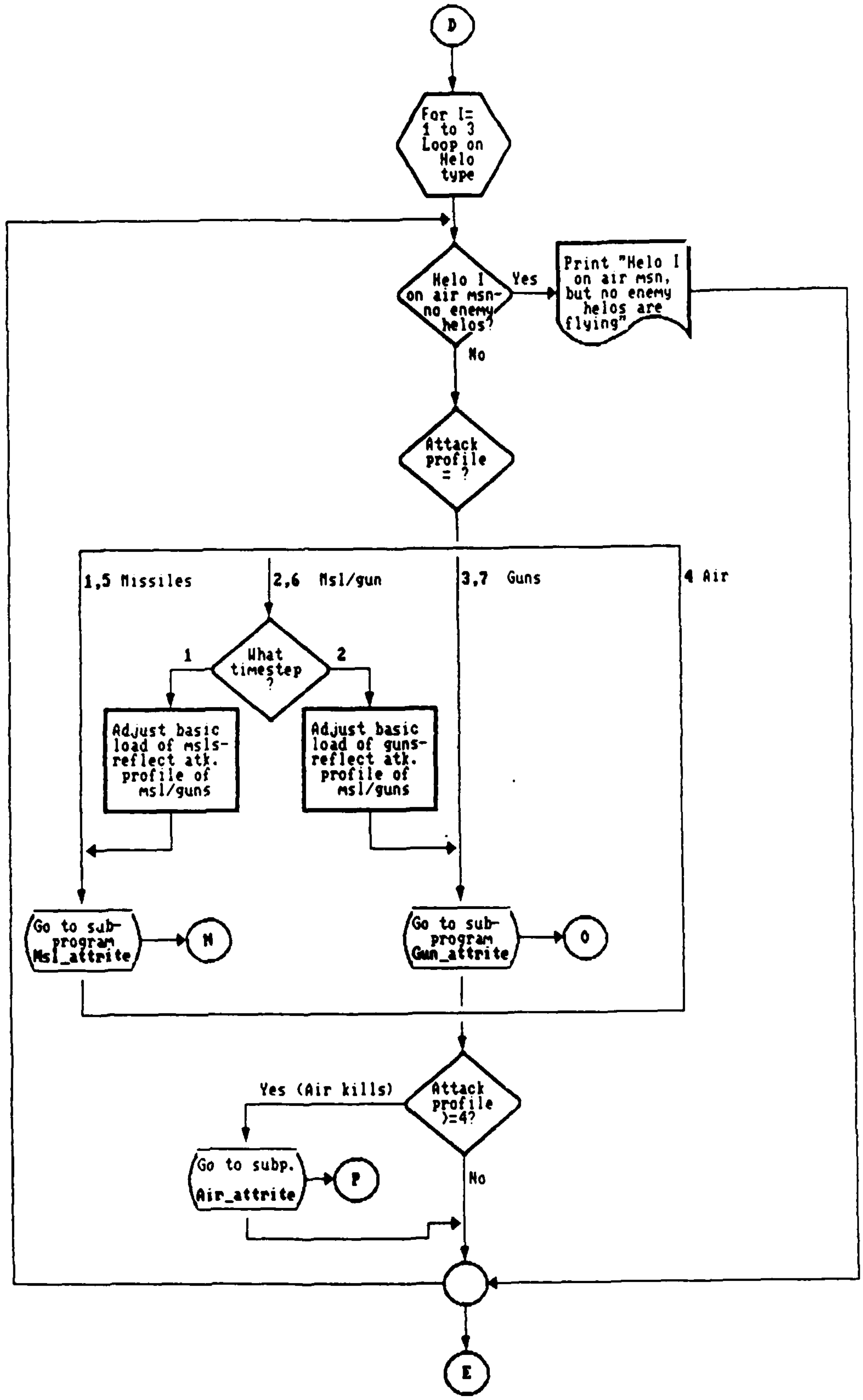


Figure 6-34. Functional flow of helicopter attrition module (continued).

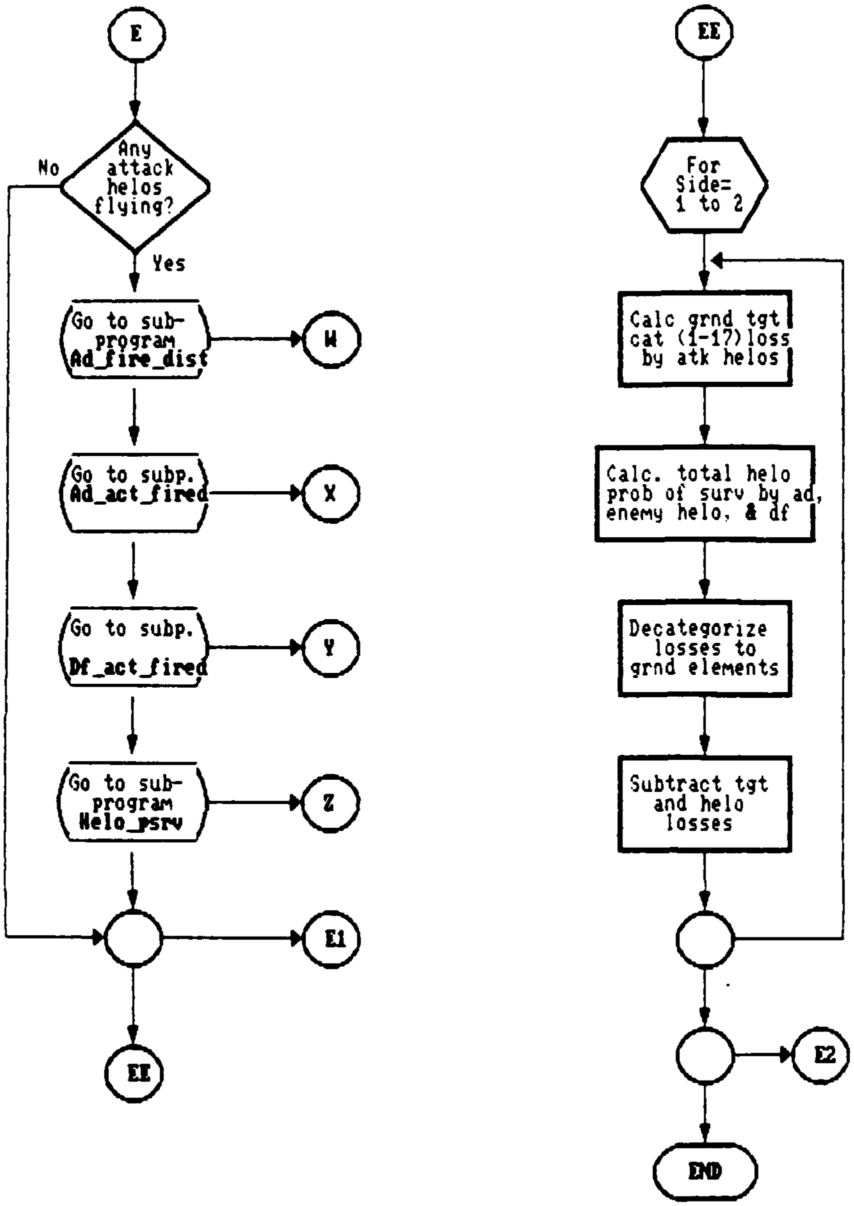


Figure 6-34. Functional flow of helicopter attrition module (continued).

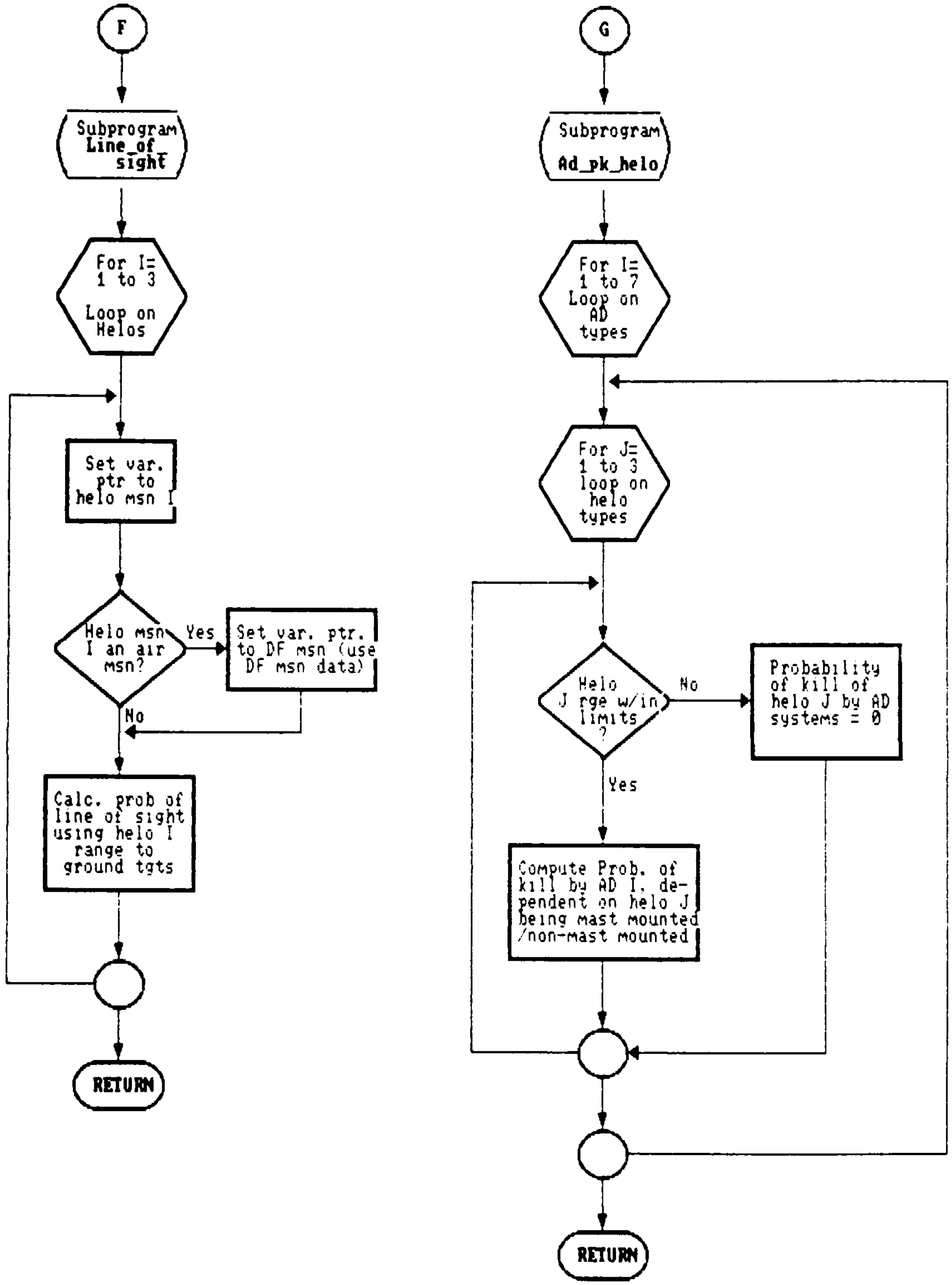


Figure 6-34. Functional flow of helicopter attrition module (continued).

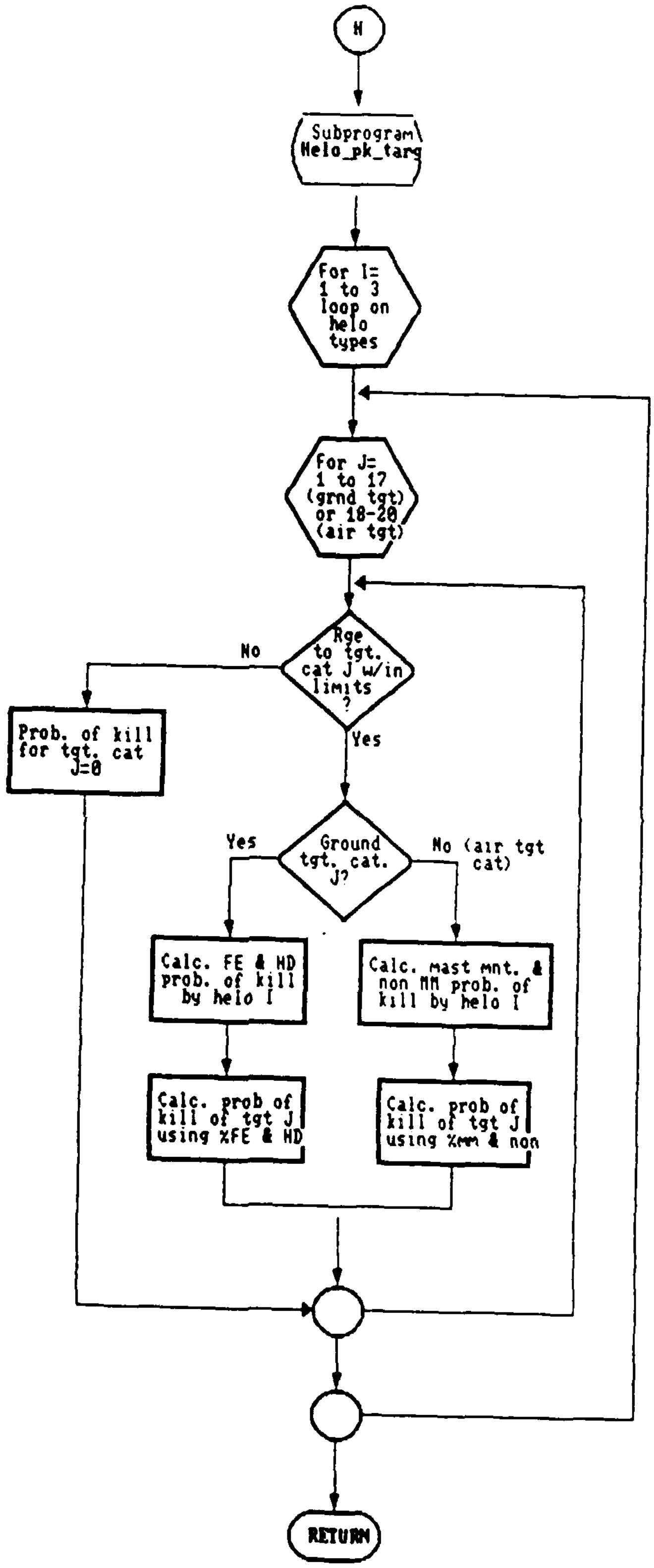


Figure 6-34. Functional flow of helicopter attrition module (continued).

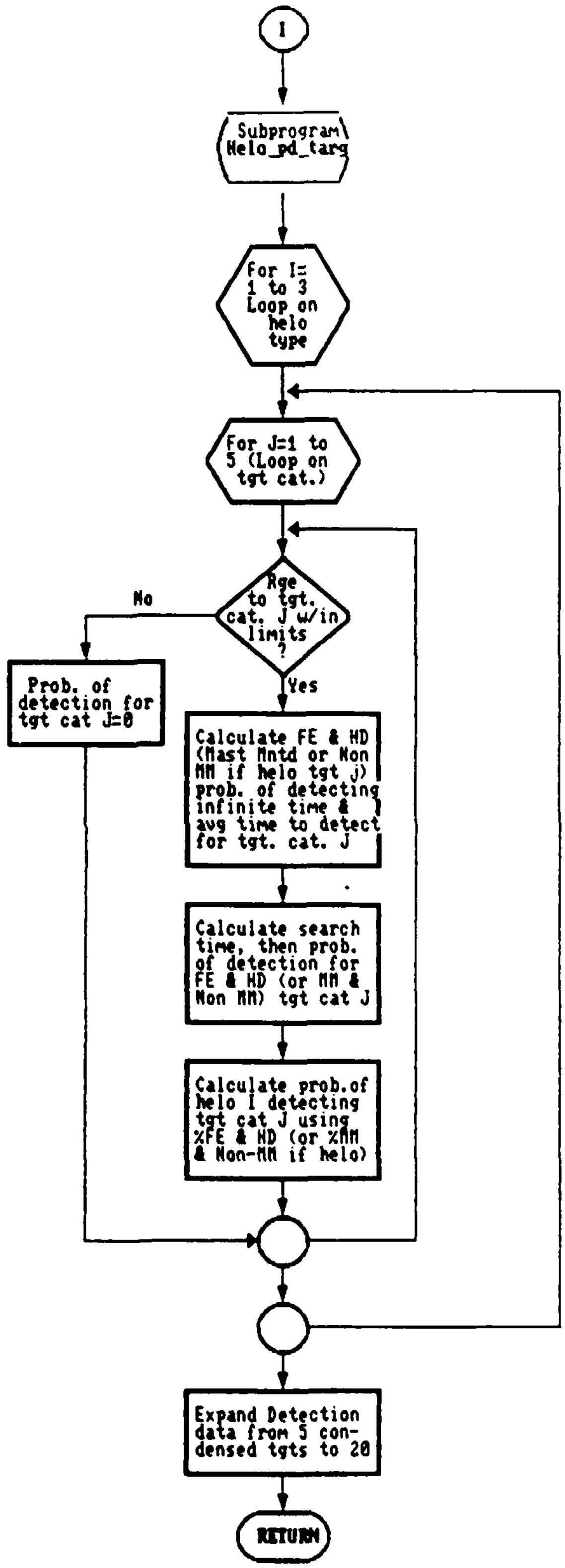


Figure 6-34. Functional flow of helicopter attrition module (continued).

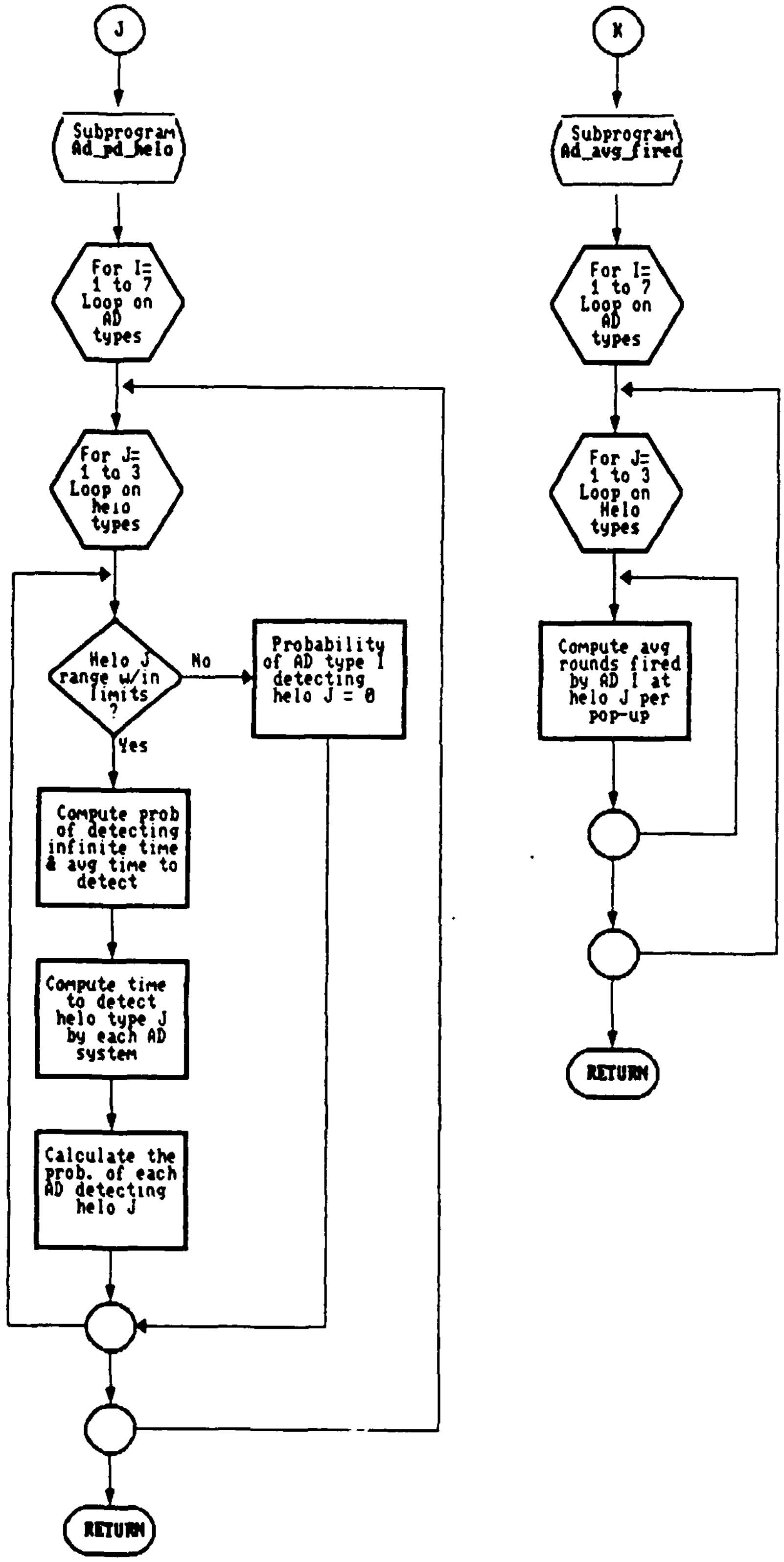


Figure 6-34. Functional flow of helicopter attrition module (continued).

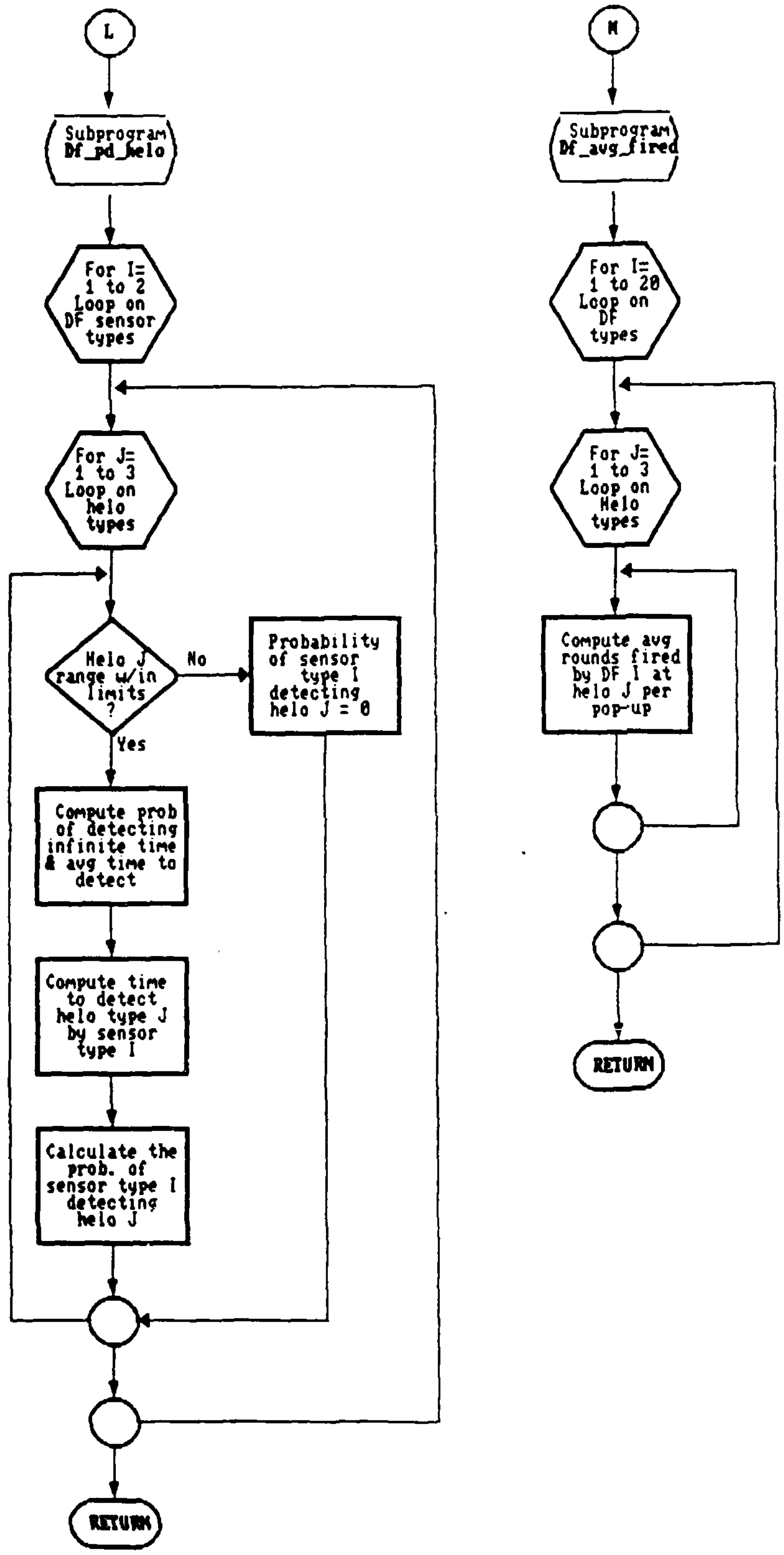


Figure 6-34. Functional flow of helicopter attrition module (continued).

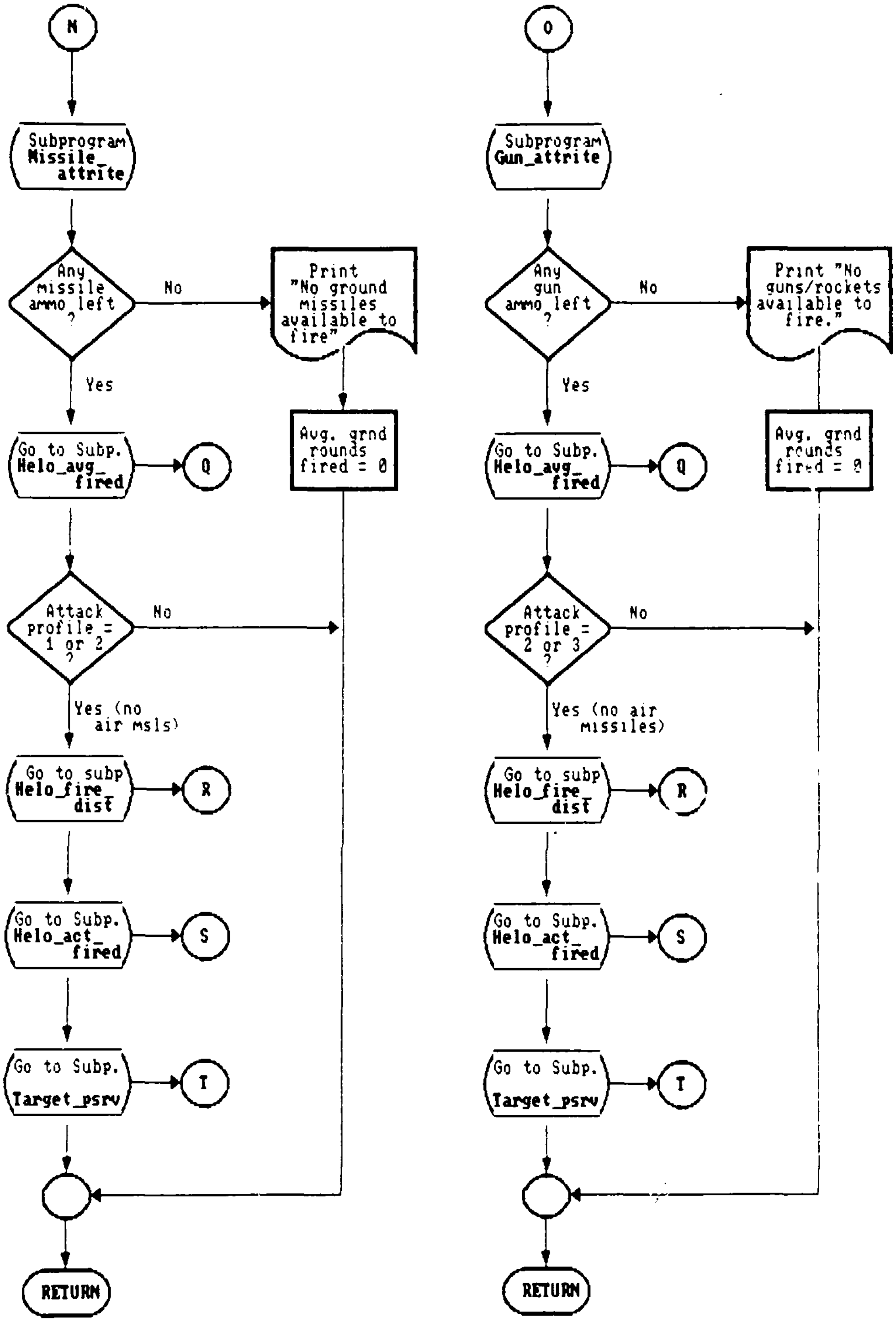


Figure 6-34. Functional flow of helicopter attrition module (continued).

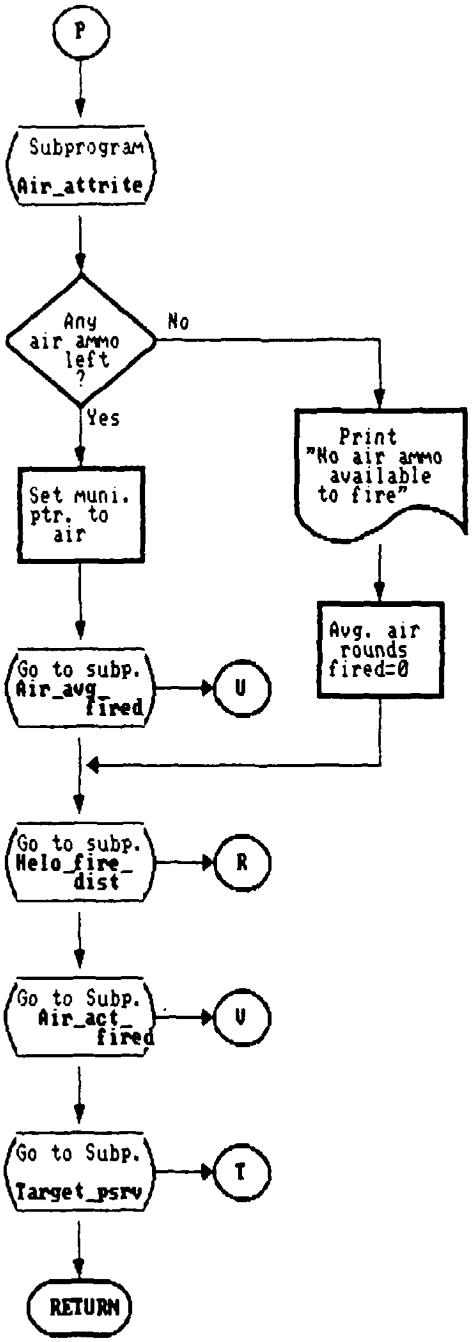


Figure 6-34. Functional flow of helicopter attrition module (continued).

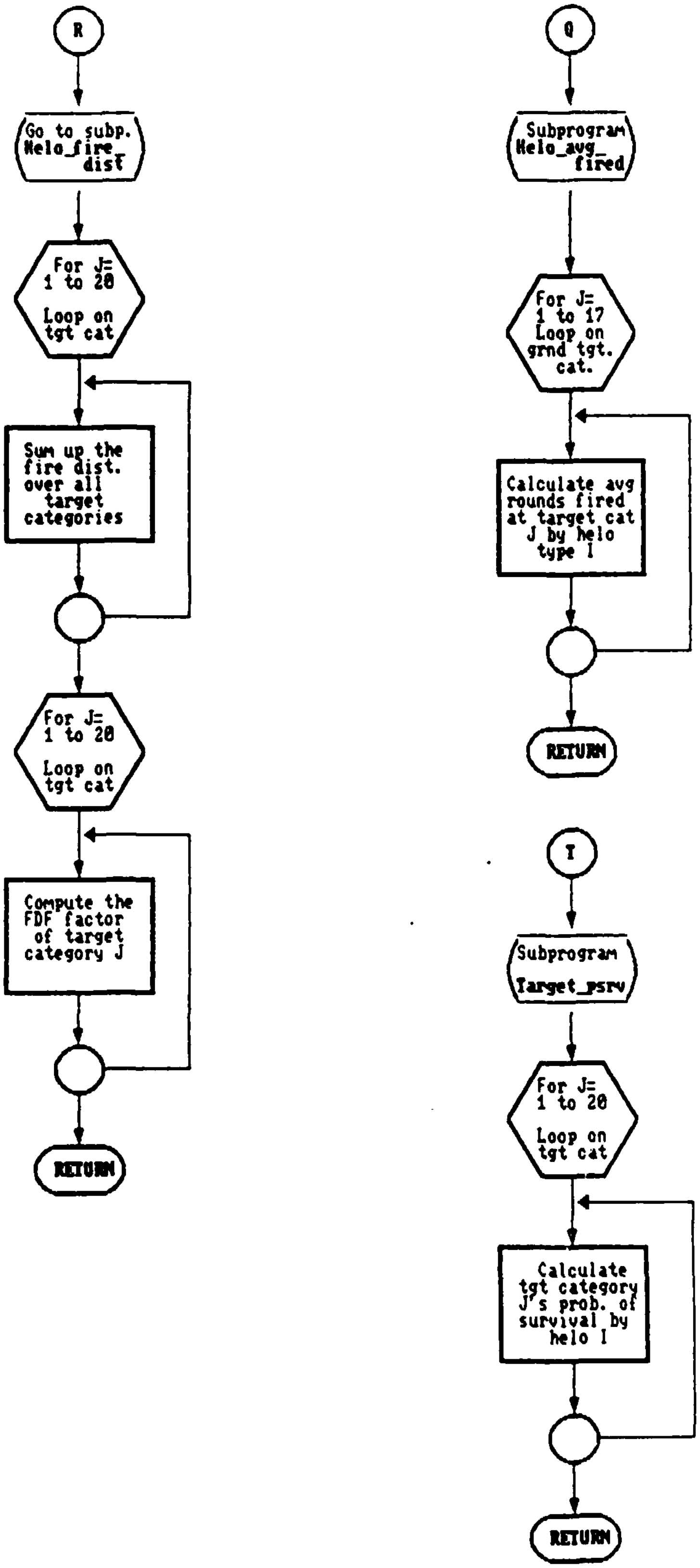


Figure 6-34. Functional flow of helicopter attrition module (continued).

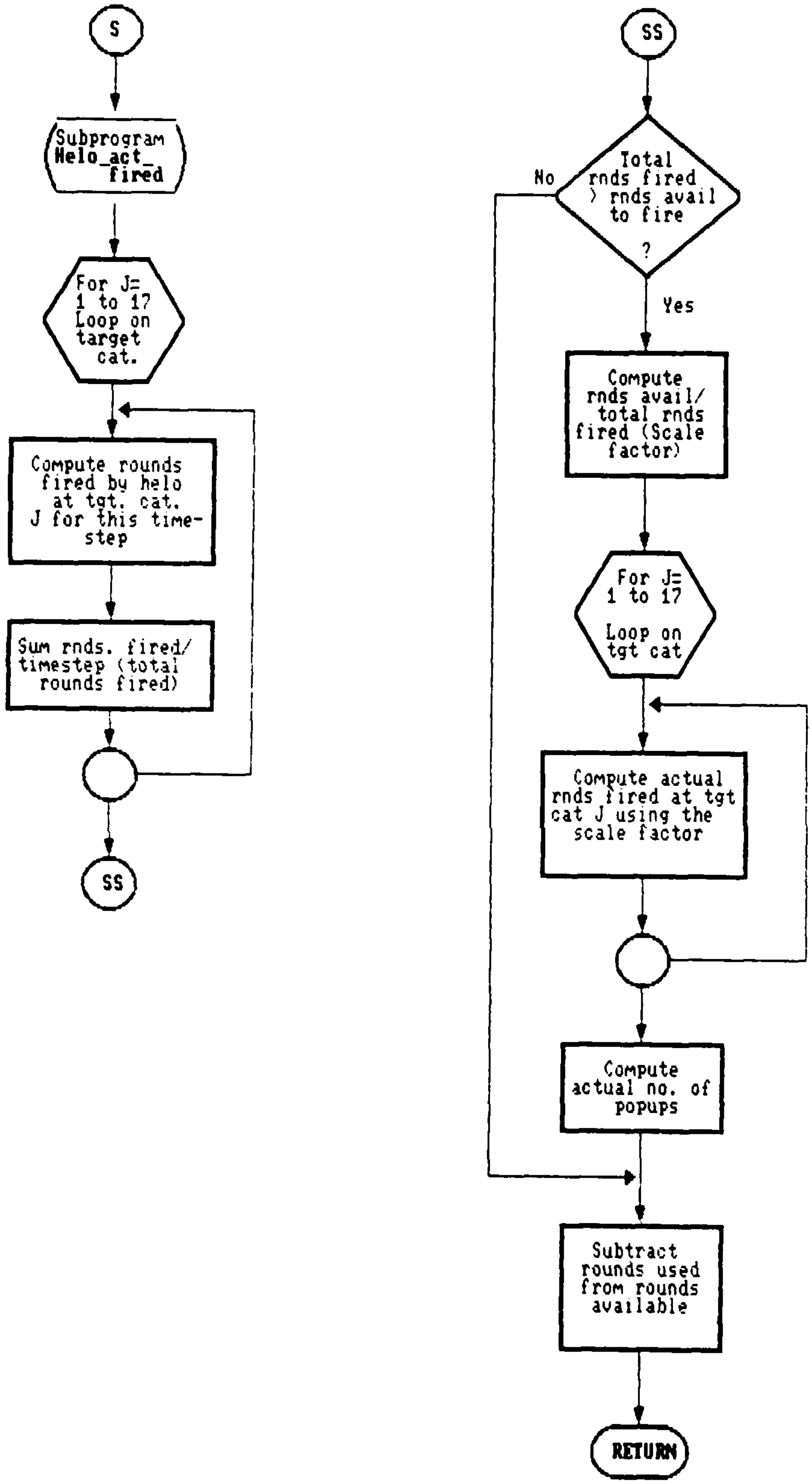


Figure 6-34. Functional flow of helicopter attrition module (continued).

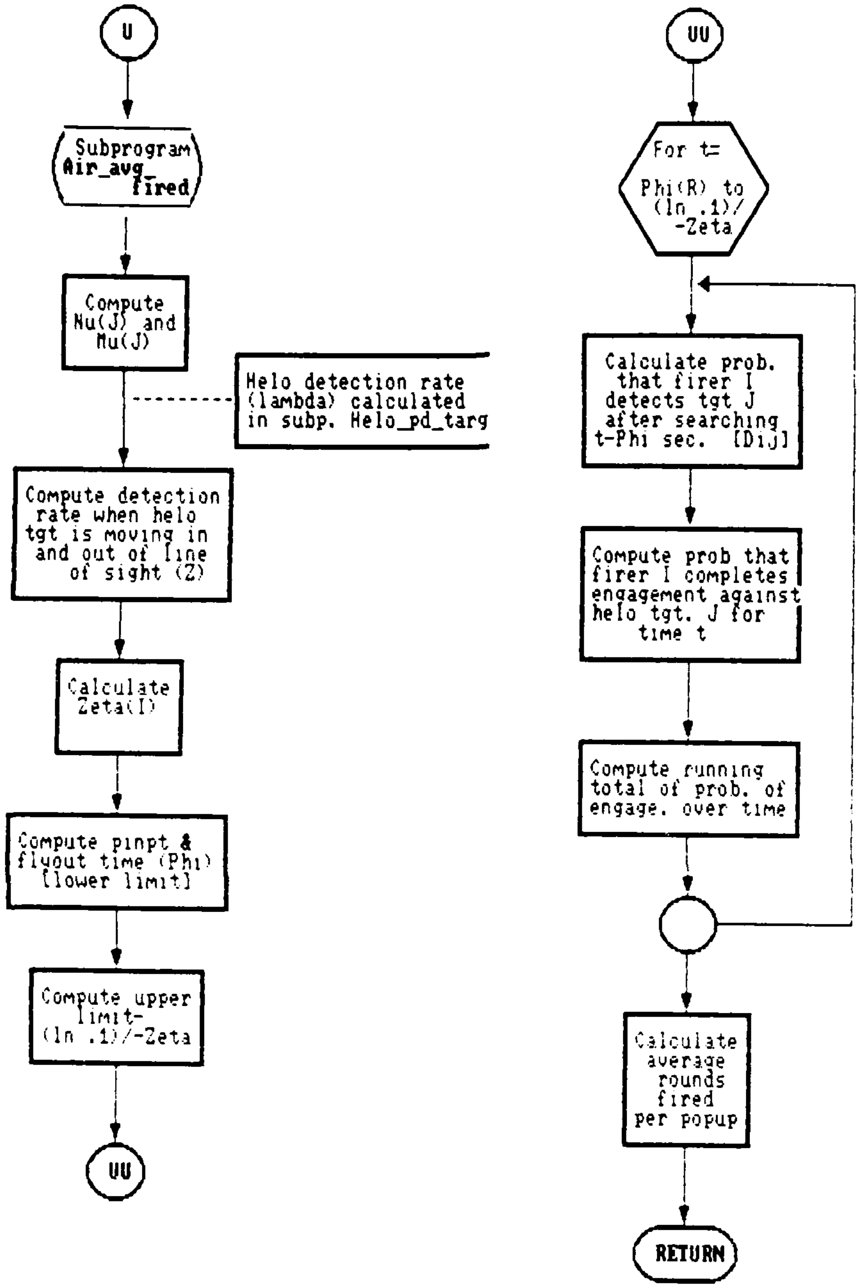


Figure 6-34. Functional flow of helicopter attrition module (continued).

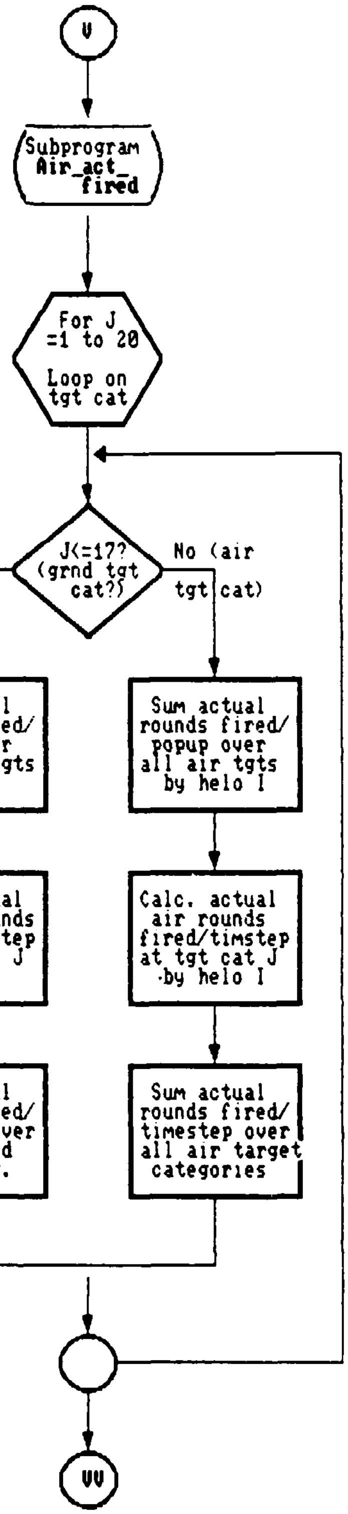


Figure 6-34. Functional flow of helicopter attrition module (continued).

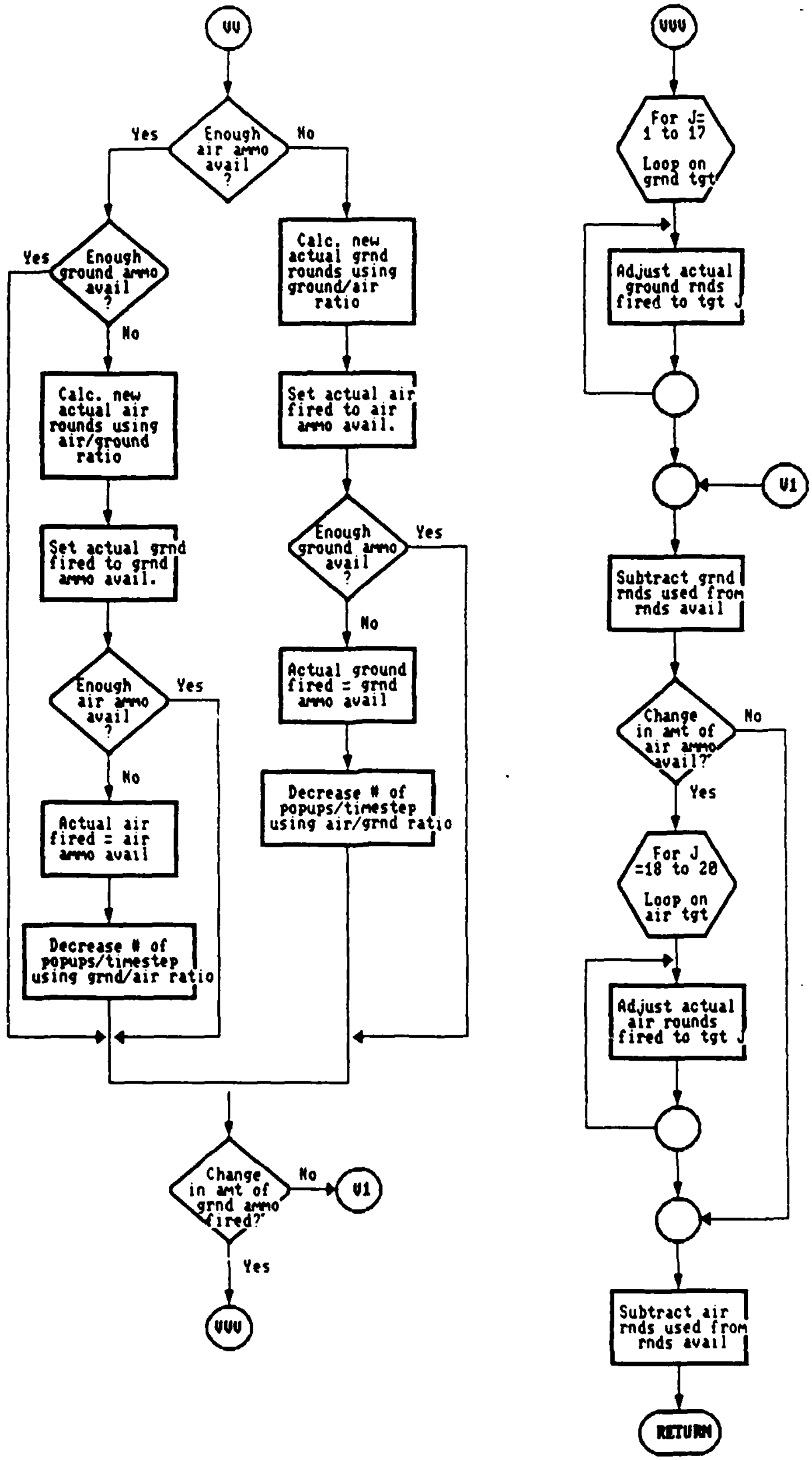


Figure 6-34. Functional flow of helicopter attrition module (continued).

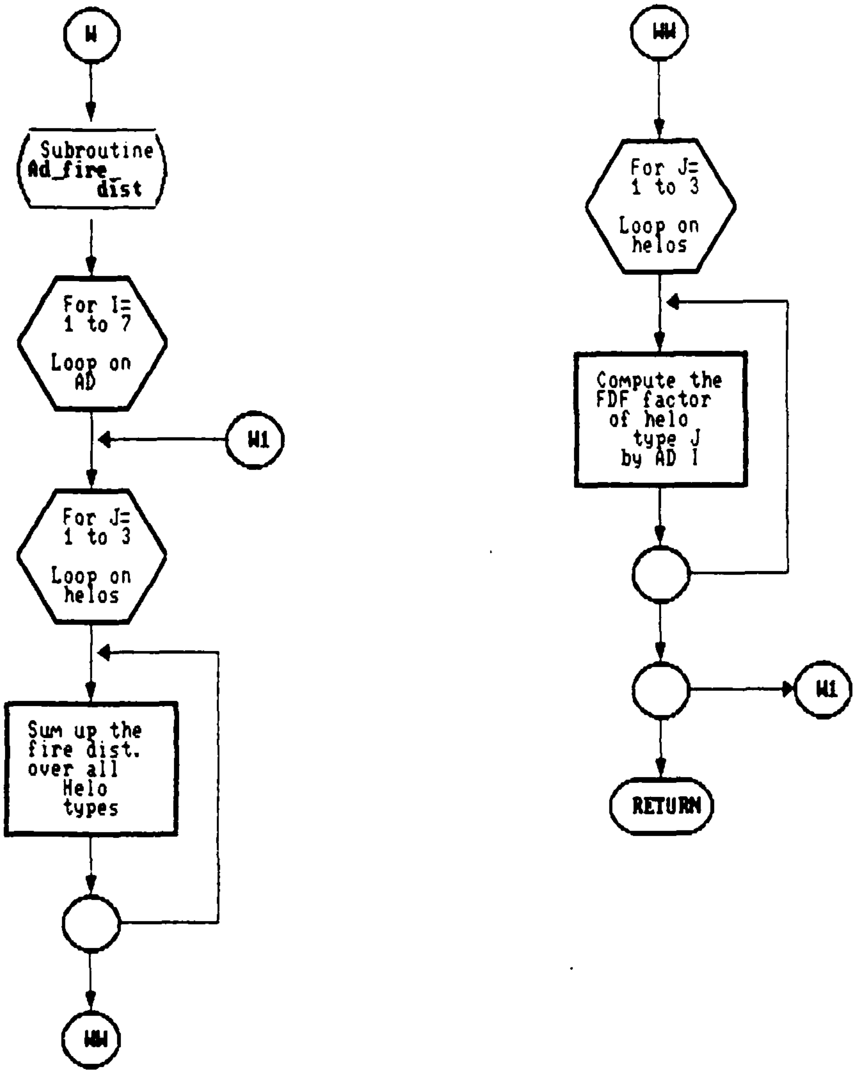


Figure 6-34. Functional flow of helicopter attrition module (continued).

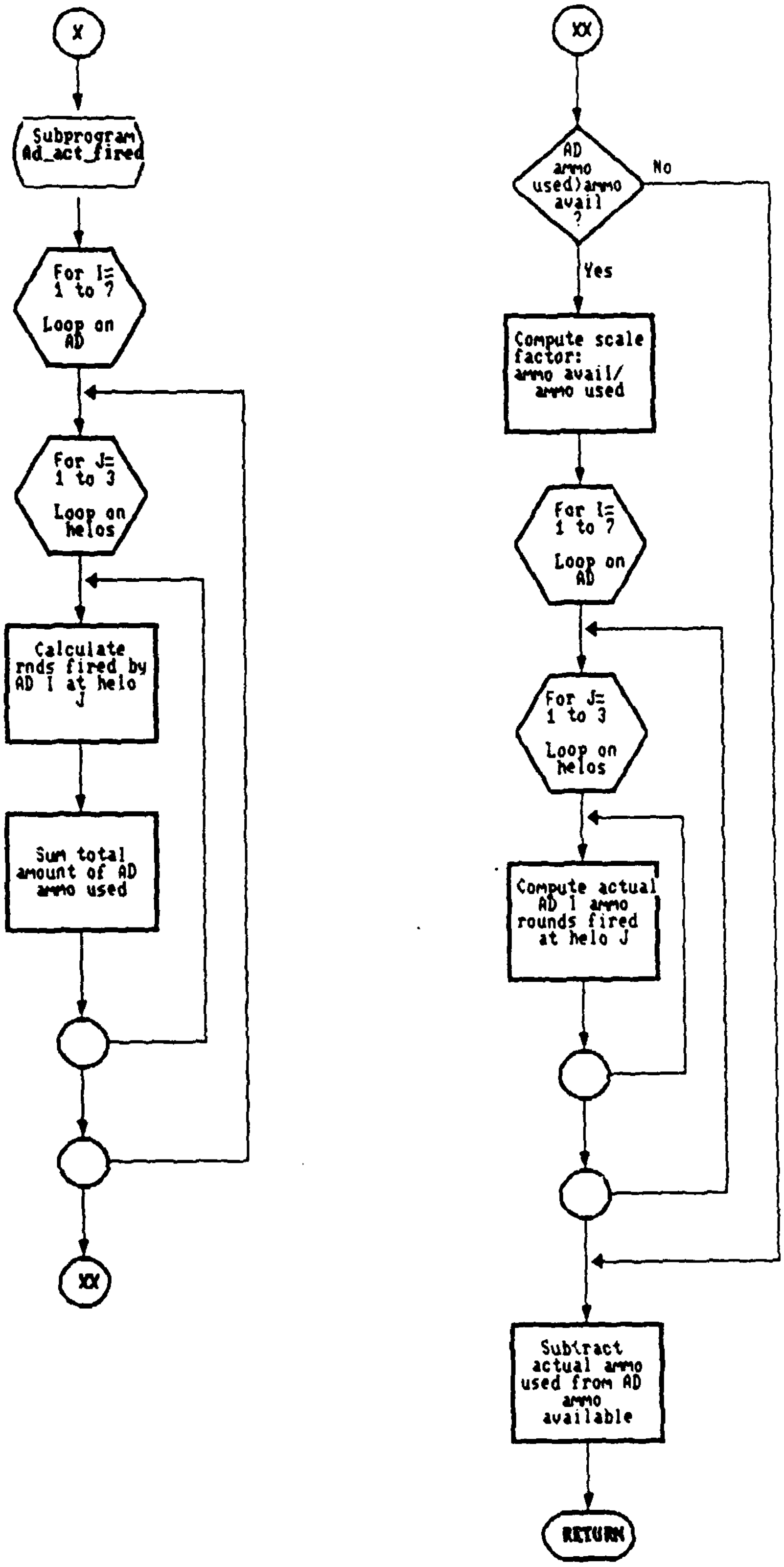


Figure 6-34. Functional flow of helicopter attrition module (continued).

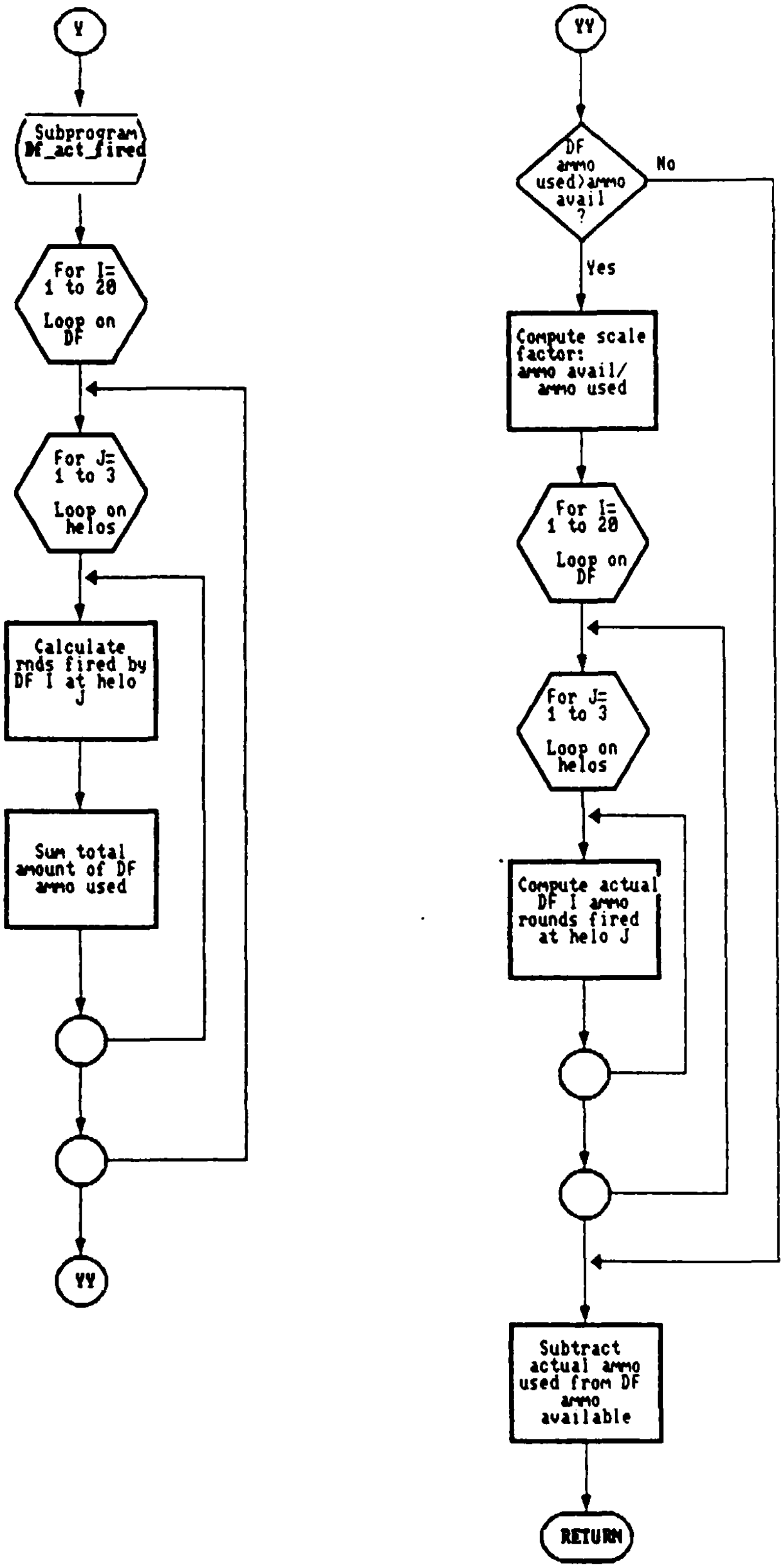


Figure 6-34. Functional flow of helicopter attrition module (continued).

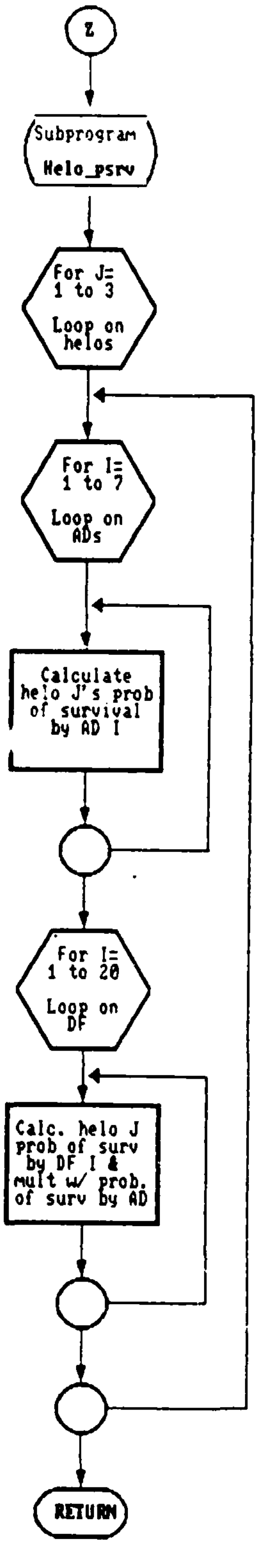


Figure 6-34. Functional flow of helicopter attrition module (concluded).

Table 6-10. Helicopter Subroutine table.

Functional area(s): A. Helo attrition variables held in common block Helo attrite.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_files	Reads in data from files. Variables used in subroutine Helo_kills	A. Helo_load (S,I,M)	Basic load of helicopters I (1-3) on side S (1=Blue; 2=Red) with munition type M (1=missile; 2=gun; 3=air-to-air).
		B. Mast_mnt (S,I)	Integer value which identifies whether helicopter on side S is mast mounted or not (0=mast mounted; 1=not mast mounted).
		C. Pd_fe_inf_a(S,I,J)/ Pd_fe_inf_b(S,I,J)/ Pd_fe_inf_c(S,I,J)	A, B, and C coefficients of a quadratic equation used to calculate the probability of detecting a fully exposed target of category J (1-20) when searching infinite time (S=side; I=helicopter type).
		D. Pd_hd_inf_a(S,I,J)/ Pd_hd_inf_b(S,I,J)/ Pd_hd_inf_c(S,I,J)	Same as above for hull defilade targets.
		E. Pd_fe_tbar_a(S,I,J)/ Pd_fe_tbar_b(S,I,J)/ Pd_fe_tbar_c(S,I,J)	A, B, and C coefficients of a quadratic equation used to calculate the average time to detect a fully exposed target of category J (1-20) helicopter I of side S.
		F. Pd_hd_tbar_a(S,I,J)/ Pd_hd_tbar_b(S,I,J)/ Pd_hd_tbar_c(S,I,J)	Same as above for hull defilade targets.
		G. Pd_rmin(S,I,A)/ Pd_rmax(S,I,A)	Minimum and maximum probability of of detection ranges based on helo type I and atmospheric conditions A (1-8) by side S.
		H. Pk_fe_a(S,I,M,J)/ Pk_fe_b(S,I,M,J)/ Pk_fe_c(S,I,M,J)	A, B, and C coefficients of a quadratic equation used to calculate the probability of kill of fully exposed targets of category J based on munition M, Side S, Helo type I.

Table 6-10. Helicopter Subroutine table.

Functional area(s): A. Helo attrition variables held in common block Helo attrite. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_files (continued)		I. Pk_hd_a(S,I,M,J)/ Pk_hd_b(S,I,M,J)/ Pk_hd_c(S,I,M,J)	Same as above for hull defilade targets.
		J. Pk_rmin(S.I,M)/ Pk_rmax(S,I,M)	Minimum and maximum probability of of kill ranges based on helo type I and munition type M by side S.
		K. Np(S,I,M)	Tactical number of rounds per engagement fired by helo type I using munition M.
		L. Fm(S,I,M)	Fire and guide time (m/sec) of munition M and helo type I.
		M. Tm(S,I,M)/ Te(S,I,M)	The average time masked and exposed per firing cycle for helo I firing munition M based on helicopter's mission.
		N. Tgt_pref(S,I,J)	Preference for targets of category J by helicopters flying mission I. Values 1 - 10, with 10 the highest preference.
		O. Plos_alpha(S,M)/ Plos_beta(S,M)	Alpha and beta values of an exponential equation used to calculate the probability of line of sight for mission M (1-3) based on terrain.

Table 6-10. Helicopter Subroutine table.

Functional area(s): A. Helo attrition variables held in common block Helo attrite. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_files (continued)		P. Pd_inf_ad_a(S,I,J)/ Pd_inf_ad_b(S,I,J)/ Pd_inf_ad_c(S,I,J)	A, B and C coefficients of a quadratic equation used to calculate the probability of Ad type I (1-7) detecting a mast mounted (J=1) or non-mast mounted (J=2) helicopter when searching infinite time.
		Q. Pd_tbar_ad_a(S,I,J)/ Pd_tbar_ad_b(S,I,J)/ Pd_tbar_ad_c(S,I,J)	A, B and C coefficients of a quadratic equation used to calculate the average time for Ad type I (1-7) to detect a mast mounted (J=1) or non-mast mounted (J=2) helicopter.
		R. Pd_ad_rmin(S,I,A)/ Pd_ad_rmax(S,I,A)	Minimum and maximum AD type I probability of detection ranges based on atmospheric conditi... A (1-8), (S=side 1-blue; 2-red).
		S. Pk_ad_a(S,I,A)/ Pk_ad_b(S,I,A)/ Pk_ad_c(S,I,A)	A, B and C coefficients used to calculate AD I's probability of kill of mast mounted (J=1) and non-mast mounted (J=2) helicopters.
		T. Pk_ad_rmin(S,I)/ Pk_ad_rmax(S,I)	Minimum and maximum AD type I probability of kill ranges for side S.
		U. Ad_pref(S,I,J)	Preference of AD element I for helicopter type J.
		V. Rnd_wt(S,I)	Weight of AD element I's munition.
		W. Rnds(S,I)	Number of rounds fired per engagement by AD element I.
		X. Fad(S,I)	Flyout velocity of AD I's munition (m/sec).

Table 6-10. Helicopter Subroutine table.

Functional area(s): A. Helo attrition variables held in common block Helo attrite. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_files (continued)		Y. Pd_cat(S,I)	A value from 1-5 representing the probability of detection category to which target category I (1-20, S=1 or 2) belongs.
		Z. Pd_inf_df_a(S,I,M)/ Pd_inf_df_b(S,I,M)/ Pd_inf_df_c(S,I,M)	A, B, and C coefficients of a quadratic equation used to calculate the probability of detecting a mast mounted (M=1) or non-mast mounted (M=2) helicopter of side S when searching infinite time. I - direct fire sensor type (1=optical; 2=thermal).
		AA. Pd_tbar_df_a(S,I,M)/ Pd_tbar_df_b(S,I,M)/ Pd_tbar_df_c(S,I,M)	A, B, and C coefficients of a quadratic equation used to calculate the average time to detect a mast mounted (M=1) or non-mast mounted (M=2) helicopter of side S. I - direct fire sensor type (1=optical; 2=thermal).
		BB. Pd_df_rmin(S,I,A)/ Pd_df_rmax(S,I,A)	Minimum and maximum probability of of detection ranges based on sensor type I and atmospheric conditions A (1-8) by side S.
		CC. Df_rnd_wt(S,M)	Weight of direct fire munition type M's round (M=1 - ground missile; M=2 - ground kinetic energy round).
		DD. Df_rnds_eng(S,M)	Number of rounds fired per engagement for munition type M (1-2) on side S (1-2).
		EE. F_df(S,M)	Flight velocity of DF munition type M (in m/sec).
		FF. Df_sen_ptr(S,I)	Sensor (1=optical; 2=thermal) that direct fire type I (1-20) of side S(1-2) uses.

Table 6-10. Helicopter Subroutine table.

Functional area(s): A. Helo attrition variables held in common block Helo attrite. (concluded)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_files (concluded)		GG. Df_muni_ptr(S,I)	Munitions (1=ground missile; 2=kinetic energy round) that direct fire type I (1-20) of side S(1-2) uses.

Functional area(s): B. Helo attrition variables held in common block Direct fire

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_files	Reads in data from files	A. B_cat (I) R_cat (I)	A value (1-17) which represents the ground target category to which weapon system I (1-70) belongs.

Functional area(s): C. Helo attrition variables held in common block Helo info

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main	Main driver	A. Btl_rg	Current range between two forces on the battlefield.
		B. Df_ammo (S)	Amount of direct fire ammo (short tons) available for side S.
Helo_range	Calculates the ranges from helo to helo and helo to ground	A. Rg_avg(S,I,J)	Average stand-off range between helo type I (1-3) and target category J (1-20) for Blue and Red helos.
		B. Rg_avg_pd(S,I,J)	Average stand-off range between helicopter I on side S and the condensed probability of detection category J (1-5).

Table 6-10. Helicopter Subroutine table.

Functional area(s): C. Helo attrition variables held in common block Helo info (concluded)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Df_attrition	Direct fire attrition	A. Df_fire_dist(S,I,J)	Fire distribution factor of each direct fire type I on side S shooting at target helicopter J (1-3).
		B. Df_pk_helo(S,I,J,M)	Direct fire I's probability of kill of helicopter J, which is mast mounted (M=1) or not (M=2).

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Helo_kills (driver)	Inputs from driver and initialization	A. Helo_mis(S,I)	Mission for helicopter type I 1 = air to ground 2 = air to air 3 = SEAD
		B. Stnd_off_rg(S,I)	Stand-off range (meters) of helicopter type I on side S.
		C. Side	An integer which identifies the attacking helicopter's side. 1 = Blue 2 = Red
		D. Side_def	An integer which identifies the defending helicopter's side. 1 = Blue 2 = Red
		E. Day_night	An integer representing the light visibility category. 0 = Day 1 = Night
		F. Time_step	On-site time of helicopter cells.

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Helo_kills (continued)		G. Cell(S,I,J)	Initial (I=1) and remaining (I=2) number of helicopters of type J on side S.
		H. Atk_prof(S,I)	An integer (1-7) representing the attacker profile of helicopter I on side S. 1 = missiles 2 = missiles and guns 3 = guns 4 = air to air missiles 5 = air to air missiles and ground missiles 6 = air to air missiles with ground missiles and guns 7 = air to air missiles and guns
		I. Target(S,I,J)	Initial (I=1) and remaining (I=2) number of targets J (1-70) on side S.
		J. Vis	Visibility index. 1 = >5 km 2 = 5 km 3 = 2 km 4 = 1 km
		K. Atmos	Atmospheric conditions based on day/night and visibility.
		L. Ad_ammo (S)	Amount (short tons) of AD ammo available for side S.
		M. Ad_wt_avl(S)	Amount (pounds) of AD ammo available for side S.
		N. Df_wt_avl	Amount (pounds) of DF ammo available.
		O. Totden	Total number of targets remaining.

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Helo_kills (continued)		P. Totnum	Total number of fully exposed targets remaining.
		Q. P_def_ray(S,J)	An array containing the percentage of each target element J on side S that is fully exposed.
		R. Pct_force_fe(S)/ Pct_force_hd(S)	Percent of side S's force which is fully exposed and hull defilade.
		S. Pct_mm(S)/ Pct_non_mm(S)	Percent of side S's force which is mast mounted and non-mast mounted.
		T. Tot_en_helos	Total number of enemy helicopters flying this timestep.
		U. Helo_rnds_avl(S,I)	Number of rounds available for helicopter type I, side S.
		V. Air_rnds_avl(S,I)	Number of air missiles available for helicopter type I, side S.
		W. No_targets(S,I)	Number of elements on target category I (1-20) for side S.
		X. Pop_tstep(I)	Number of popups per timestep for helicopter type I.
		Y. Muni(I)	Integer representing the munitions type used by helicopter I this timestep. 1 = ground missiles 2 = guns 3 = air to air missiles
		Z. En_muni(I)	Integer representing the munition type used by enemy helicopter I this timestep.
		AA. Lase	Integer representing lasing helicopter. 1, 2 = self 3 = scout

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Helo_kills (concluded)		BB. Target_psrv(S,J)	Probability of survival of target category J on side S.
		CC. Helo_psrv(S,I)	Probability of survival of helicopter I on side S.
		DD. Target_loss(S,J)	Number of targets of category J (1-20) lost for side S (1=Blue; 2=Red).
		EE. Helo_loss(S,I)	Number of helicopters of type I lost for side S.
Line_of_sight	Probability of line of sight between helicopters and ground elements.	A. Plos(S,I)	Probability of line of sight for helicopter I on side S to enemy ground elements.
Helo_pd_targ	Calculates the probability of a helicopter detecting a target	A. Hdinf	Probability of detecting hull defilade targets given infinite search time.
		B. Feinf	Probability of detecting fully exposed targets given infinite search time.
		C. Hdtbar	Average time to detect a hull defilade target.
		D. Fetbar	Average time to detect a fully exposed target.

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Helo_pd_targ (concluded)		E. Ut(I)	Actual time available to helicopter I to search for a target.
		F. Pdt_fe	Actual probability of detecting a fully exposed target.
		G. Pdt_hd	Actual probability of detecting a hull defilade target.
		H. Helo_pd_tgt(J5)	Actual probability of the helicopter's detecting collapsed target category J5 (1-5).
		I. Helo_pd_targ(I,J)	Probability of helicopter I detecting target category J.
Helo_pk_targ	Calculates the probability of a helicopter killing a target	A. Pkhd	Probability of killing a hull defilade target.
		B. Pkfe	Probability of killing a fully exposed target.
		C. Helo_pk_targ(I,J)	Probability of helicopter I killing a target in category J.

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Ad_pd_helo	Calculate the probability of Air Defense systems to detect helicopters.	A. Adinf	Probability of detecting targets given infinite time.
		B. Adtbar	Average time to detect helicopter.
		C. Adteng	Time available to detect helicopter assuming an engagement will follow (munitions sensitive).
		D. Ad_pd_helo(I,J)	Actual probability of Air Defense System I detecting helicopter J.
Ad_avg_fired	Calculate the amount of ammunition fired by the Air Defense systems as if each helicopter is by itself.	A. Ad_avg_fired(I,J)	Amount of ammunition fired by AD system I at at helicopter J.
Ad_pk_helo	Calculate the probability of killing a helicopter at a given range.	A. Mt	Variable used as a subscript in probability of kill data. 1 = helicopter has mast mounted detection sensor (little exposure) 2 = helicopter must be exposed to accomplish detection.
		B. Ad_pk_helo(I,J)	Probability of killing helicopter J by AD system I.

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Helo_avg_fired	Calculate the amount of ammunition fired by a helicopter at a target (as if only one target is available).	A. Helo_avg_fired(I,J)	The amount of ammunition fired by helicopter I at target category J.
Helo_fire_dist	Calculate the distribution of fire against all target categories.	A. Helo_tot_dist	The sum of all fire distribution for each target.
		B. Helo_grd_dist	The sum of all fire distribution for ground targets only (J=1-17).
		C. Helo_file_dist (I,J)	The percentage of fire that helicopter I directs at target category J (total equals 1.0).
Helo_act_fired	Calculate the actual amount of ammunition fired by a helicopter against a target category.	A. Helo_act_fired (I,J)	Actual amount of ammunition fired by helicopter I at target category J.
		B. Tot_rnds_fired	Total rounds fired at all targets.
		C. Helo_act_pp	Rounds fired at all targets during 1 popup.
Target_psrv	Calculate the losses to targets as a result of helicopter fire.	A. Target_psrv(S,J)	Probability of survival of target J after all helicopters have fired.
Ad_fire_dist	Calculate the distribution of fire against all helicopters by Air Defense systems.	A. Ad_tot_dist	The sum of all fire distribution for each helicopter.
		B. Ad_fire_dist (I,J)	The percentage of fire that Air Defense system I directs at helicopter J (total equals 1.0).

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Ad_act_fired	Calculate the actual amount of ammunition fired by an Air Defense system against a helicopter.	A. Ad_act_fired(I,J)	Actual amount of ammunition fired by AD system I against helicopter J.
		B. Tot_wt_fired	Total weight of rounds fired at all helicopters.
Helo_psrv	Calculate the losses to helicopters as a result of Air Defense system fire.	A. Mt	Mast type 1 = mast mounted 2 = non-mast mounted
Air_avg_fired	Calculate the average rounds fired by one helicopter against another.	A. Pdt_rate_los	The rate that firing helicopter I detects target helicopter J when when both are popping up and down.
		B. P_engage	The probability of engagement, given the flight time of the appropriate munitions.
		C. Helo_avg_fired (I,J)	The average number of rounds fired by helicopter I at target helicopter J.
Air_act_fired	Calculate the actual rounds fired by one helicopter at another.	A. Helo_air_dist	The sum of all fire distribution for air targets only (J=18-20).
		B. Grd_pp	Actual rounds fired per popup at all ground targets.
		C. Helo_act_fired (I,J)	Actual rounds fired by helicopter I at target helicopter J.
		D. Helo_actual	Actual rounds fired by helicopter I at all ground targets.
		E. Air_pp	Actual rounds fired per popup at all air targets.

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (continued)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Air_act_fired (concluded)		F. Air_act_fired(I,J)	Actual rounds fired by helicopter I at helicopter J.
		G. Air_actual	Actual rounds fired by helicopter I at all helicopter targets.
		H. A_g_ratio	The ratio of actual rounds fired at air targets vs. ground targets per popup.
Df_pd_helo	Calculate the probability of Direct Fire (DF) systems detecting the target helicopter.	A. Dfinf	The probability of detecting the the target given infinite time.
		B. Dftbar	The average time to detect a target.
		C. Mt	Variable used as a subscript in probability of kill data. 1 = helicopter has mast mounted detection sensor (little exposure) 2 = helicopter must be exposed to accomplish detection.
		D. Dfteng	Time available to detect helicopter assuming an engagement will follow (munitions sensitive).
Df_avg_fired	Calculate the average ammunition fired by Direct Fire systems at helicopters.	A. Df_avg_fired(I,J)	Average number of rounds fired by DF system I at at helicopter J.

Table 6-10. Helicopter Subroutine table.

Functional area(s): D. Calculate kills between helicopters and ground elements, AD Elements and enemy helicopters. (concluded)

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Df_act_fired	Calculate the actual amount of ammunition fired by a Direct Fire system against helicopters.	A. Df_act_fired(I,J)	Actual number of rounds fired by DF system I against helicopter J.
		B. Tot_rnds_fired	Total number of rounds fired at all helicopters.
		C. Tot_wt_fired	Total weight of rounds fired at all helicopters.

Section VI. Precision Guided Munitions Attrition

1. PURPOSE.

The precision-guided munitions (PGM) attrition module calculates losses to target elements due to cannon-launched guided projectiles (CLGP) and guided antiarmor mortar projectiles (GAMP).

2. GENERAL.

A. CLGP are Blue indirect fire weapon systems that fire at point targets. Guidance for the CLGP rounds is assumed to be provided by a ground locator laser designator (GLLD) or may be specified to have remotely piloted vehicle (RPV) guidance. Smoke does not degrade the allocation of targets to CLGP. GAMP attrition is applied in a similar manner, but without the choice of guidance. Smoke is also assumed not to degrade the allocation of targets to GAMP. Both PGMs fire only under close support artillery munitions.

B. Currently, only Blue PGMs are used against Red targets in DIME.

3. DATA FLOW.

A. Data received from the ground combat driver program include which PGMs fire, number of rounds to be fired, visibility range, terrain type, PGM sensor types, cloud height in meters, atmospheric (dust) degradation, and the targets available.

B. All other data are specified in external data files. This information includes weighted values for target preference, single-shot kill probabilities, masks displaying which targets may be fired upon, and designator degradation factors.

C. The data flow is represented in Figure 6-35.

4. FILE STRUCTURE.

External files are as follows:

A. The Tgt_value (I,J) file contains numbers from 0 to 10 (10 being highest preference) which represent the preference of the Ith PGM firing on the J^{th} target where:

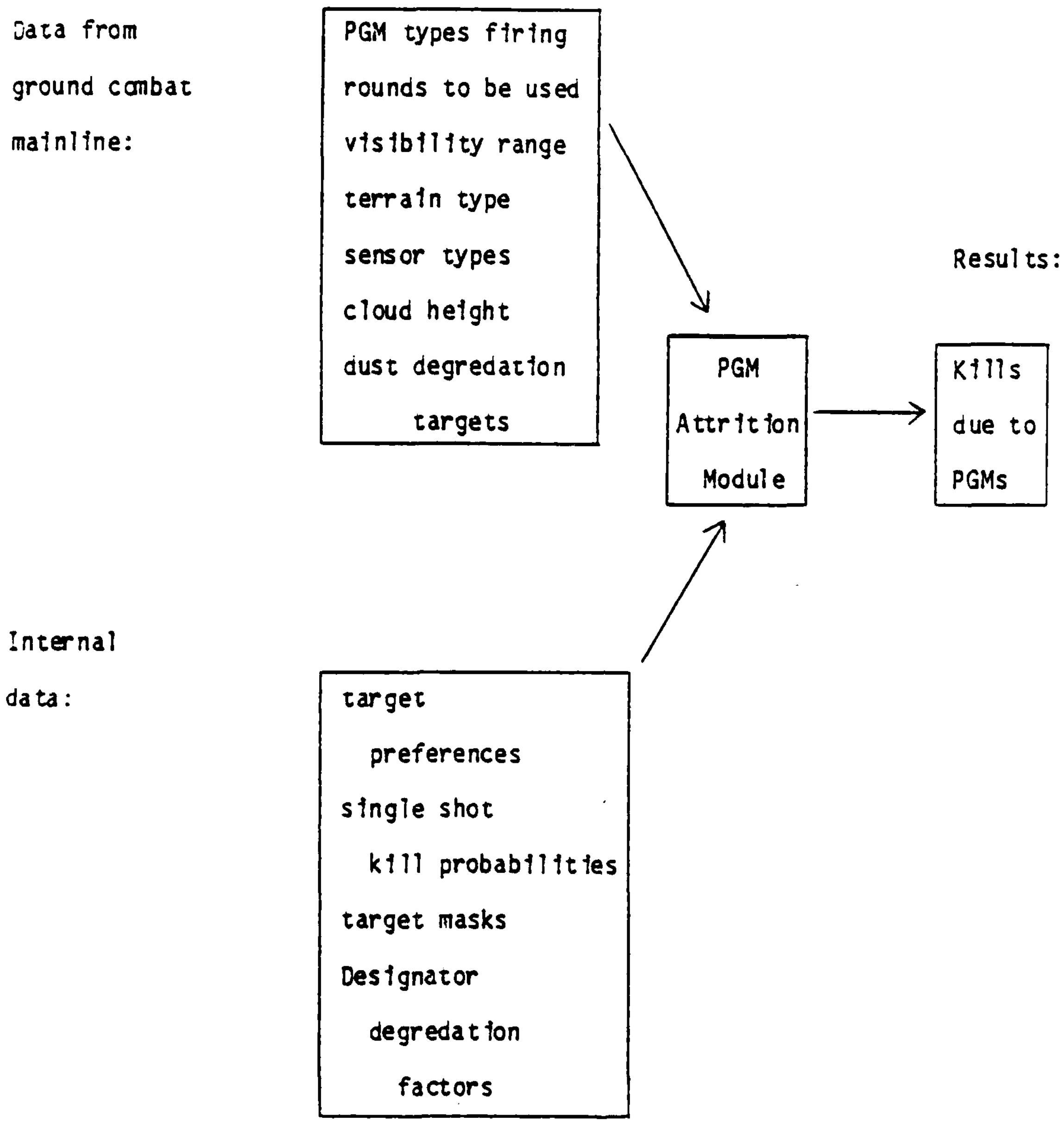


Figure 6-35. PGM data flow.

I = 1 - CLGP
    2 - GAMP
J = 1 to 70 system elements

B. The Sskp (I,J) file contains the single-shot kill probability (SSKP) for the I^{th} PGM firing on the J^{th} target. All CLGP SSKPs depend on the guidance system. Therefore, Sskp(1,J), for J=1 to 70, contain zeros. The array Sskp_clgp(J) is selected according to the guidance system.

C. Tgt_mask1(I,J) represents the firing ability of the I^{th} PGM to fire upon the J^{th} target. A zero means the target may not be fired upon; a 1 means the target may be fired upon.

D. The following data refers only to CLGP:

(1) Terr_factor(I) is CLGP's designator degradation factor based on terrain type where:

I = 1 - Open
    2 - Rolling
    3 - Hilly
    4 - Mountainous.

(2) Prob_dustabort(I,J) is CLGP's designator degradation factor where:

I = 1 - Light atmospheric (dust) obscuration
    2 - Medium atmospheric (dust) obscuration
    3 - Heavy atmospheric (dust) obscuration

J = 1 - 7 km visibility
    2 - 5 km visibility
    3 - 2 km visibility
    4 - 1 km visibility.

(3) Clgp_mask(J) partitions the combined mask of Tgt_mask1(*) into a mask which specifies the sensor designator. Clgp_mask(J) may be chosen to represent GLLD or RPV accordingly. J represents the 70 target types.

(4) Prob_desg(I,J) is the CLGP designator degradation factor where:

I = 1 - Up to 1500 feet
    2 - 1500-2000 feet
    3 - 2000-2500 feet
    4 - 2500-3000 feet
    5 - 3000-4500 feet
    6 - Over 4500 feet

J = 1 - 7 km visibility
    2 - 5 km visibility
    3 - 2 km visibility
    4 - 1 km visibility

5. **ALGORITHMS.**

A. Figure 6-36 represents a generalized logic flow of the processes occurring in the PGM attrition module. This module involves a slice methodology like that in Chapter 6, section IV, which is the direct fire attrition. This slice methodology is used to bring about a more accurate representation of fire during battle. It allows the attrition to occur in 15 slices during a 30-minute period rather than one large mass of fire in one moment for a 30-minute period. PGMs do not use range bands as does the direct fire algorithm.

B. Calculations occurring during each slice consist of the following:

(1) Calculate weighted sum.

$$W_p = \sum_{t=1}^{T} (Tv_{pt} * Ps_t * Nt_t * Tm_{pt} * Des) \qquad \text{(Eq. 6-81)}$$

where:

- W_p = the weighted target sum for the p^{th} PGM.
- T = total number of target systems.
- Tv_{pt} = the target value of the p^{th} PGM firing on the t^{th} target.
- Ps_t = the current probability of survival for the t^{th} target during the current slice.
- Nt_t = the number of targets in the t^{th} target element.
- Tm_{pt} = the target mask for the p^{th} PGM firing on the t^{th} target.
- Des = the designator discriminating factor for discriminating between high and low probability of kill (PK) targets.

(2) Calculate rounds fired per slice. These rounds are calculated per slice for each firer firing on one type target system. The formula below is used only if $Tm_{pt} * Ct_t$ is greater than zero:

$$R_{pt} = [(Ct_t * Tv_{pt} * Des)/W_p] * (Nr_p / Ns) \qquad \text{(Eq. 6-82)}$$

where:

- R_{pt} = the rounds fired by a PGM firing on one type target system.
- Ct_t = current number of targets available to fire upon for this slice.
- Nr_p = the number of rounds the p^{th} PGM is to fire.
- Ns = the number of slices; set to 15.

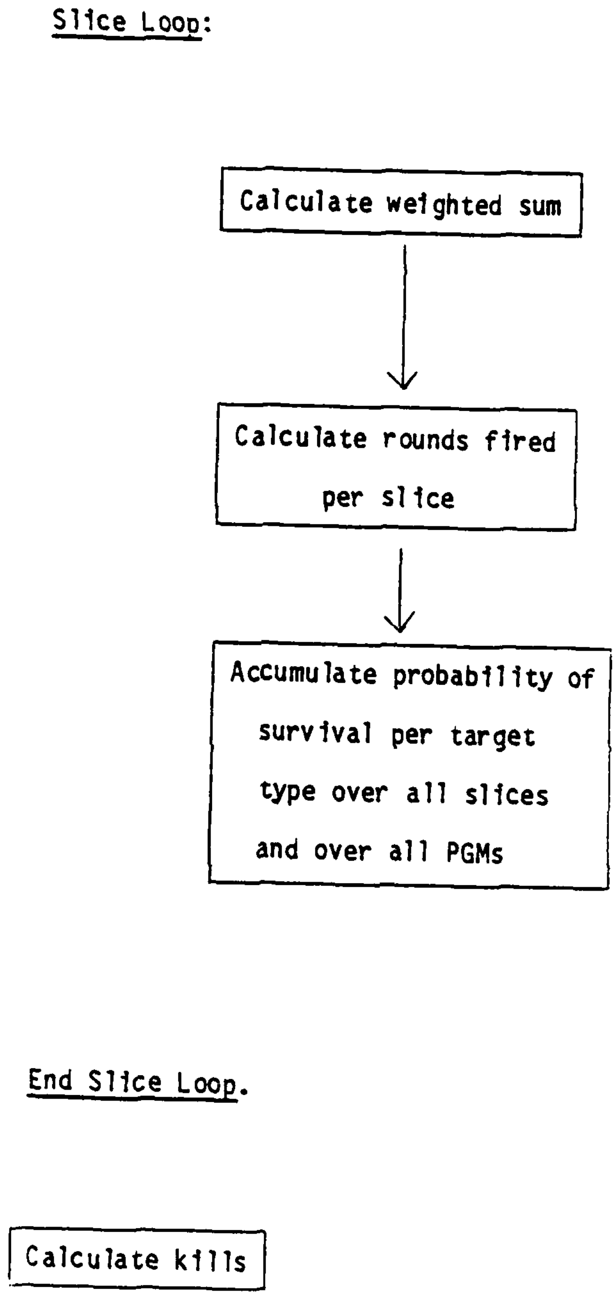


Figure 6-36. PGM logic flow.

These rounds are then multiplied by all degradation factors to represent the number of rounds which hit the targets. The number of rounds which hit the targets is shown as Rh below.

$$Rh_{pt} = R_{pt} * Td * Pd * Pda \qquad \text{(Eq. 6-82a)}$$

where:

Td = degradation due to terrain.
Pd = designator degradation due to clouds and visibility.
Pda = designator degradation due to dust and visibility.

(3) The probability of survival for each target being fired upon by each PGM is:

$$P_{pt} = (1 - Pk_{pt} * Lr/D) \uparrow Rh_{pt} \qquad \text{(Eq. 6-83)}$$

where:

P_{pt} = the probability of survival for the t^{th} target being fired upon by the p^{th} PGM.
Pk_{pt} = the single-shot kill probability of the p^{th} PGM firing on the t^{th} target.
Lr = the laser designator reliability factor.
D = the maximum of 1 and the current number of available targets for this slice (CT) multiplied by the smoke degradation factor which is set to 1.
Rh_{pt} = number of rounds which hit the targets.

The value P_{tk} is the probability of target t surviving the lethality of all PGMs for slice K.

$$P_{tk} = \prod_{p=1}^{Np} P_{ptk} \qquad \text{(Eq. 6-84)}$$

The value P_{tk} is accumulated across all slices by:

$$Ps_t = \prod_{k=1}^{Ns} P_{tk} \qquad \text{(Eq. 6-84a)}$$

giving the probability of survival (Ps_t) per target over all slices (Ns) and over all PGMs.

(4) Now it is possible to calculate the losses to each target system. The formula is as follows:

$$L_t = (1 - Ps_t) * Nt_t \qquad \text{(Eq. 6-85)}$$

where:

L_t = losses to target elements due to PGMs.
Ps_t = the probability of survival per target element over all slices and over all PGMs.
Nt_t = the number of targets in the t^{th} target element.

6. "UNITFILE" IMPACT.

This module does not directly impact the unit status file ("UNITFILE"). Kills calculated in this routine are returned to the ground combat mainline and then decremented from the "UNITFILE".

7. CODE.

A. Introduction. This section contains information on the PGM attrition code. The functional areas discussed in the following paragraphs are represented in Figure 6-37.

B. PGM attrition functional areas.

(1) The data received from the ground combat mainline include PGMs which fire, number of rounds to be fired, visibility range, terrain type, PGM sensor types, cloud height in meters, atmospheric (dust) degradation flag, and available targets.

(2) Data files are read for both CLGP and GAMP.

(3) Initialization of variables, such as the number of slices, the number of possible firers, and the number of possible target elements, occurs. Before accumulation of the probability of survival, it must be initialized to 1.

(4) Files specifically set up for CLGP are now read. To do this, flags are set to determine the guidance system so the appropriate data may be used.

(5) Due to the two sets of mask files and SSKP files set up for CLGP, it is necessary to combine the two sets of files into one set of usable files:

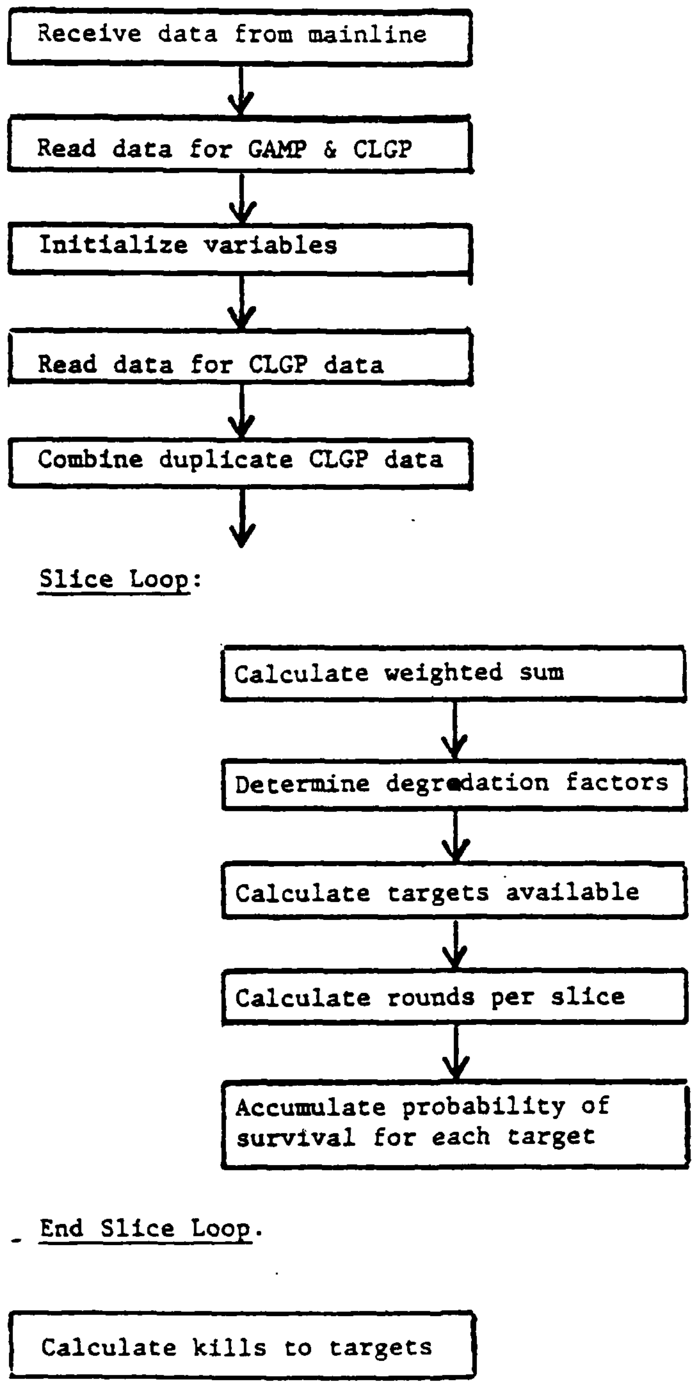


Figure 6-37. PGM functional flow.

The Tgt_mask1(1,J) and the Clgp_mask(J) files are combined giving Tgt_mask(1,J). At the same time, Tgt_mask1(2,J) is placed into Tgt_mask(2,J). Then the Sskp_clgp(J) replaces the zeros originally in Sskp(1,J). The J represents the 70 weapon elements.

(6) Using the slice methodology discussed in the algorithm portion of PGM attrition, the calculations occurring within the slice loop consist of the following:

(a) Weighted sum of current targets for each firer.

(b) Specific degradation factors determined for both CLGP and GAMP.

(c) Current available targets. This represents the slice losses to all targets.

(d) Rounds fired per slice. These are calculated within the slice loop. Degradation factors are then multiplied to the rounds fired. This is done to represent the number of rounds which hit targets. Actual rounds used are calculated in the ground combat mainline.

(e) Probability of survival per slice for each firer firing on each target. This probability is then accumulated for all firers firing on each target over all slices.

(7) The accumulated probability of survival for each target is used to calculate losses due to PGMs.

C. The primary variables for the PGM attrition module are shown in Table 6-11. Each variable is accompanied by a short description. See Table 6-14 for the ground combat code listing.

Table 6-11. PGM subroutine table.

Functional area(s): A. PGM induced losses.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Pgm_atrit	Calculates attrition due to PGM's.	A. Cld_ndx	Cloud height category: 1 = up to 1500 ft. 2 = 1500 - 2000 ft. 3 = 2000 - 2500 ft. 4 = 2000 - 3000 ft. 5 = 3000 - 4500 ft. 6 = over 4500 ft.
		B. Clgp_mask (I)	Represents the firing ability of CLGP on the Ith target (I = 1-70): 1 = Can fire 2 = Can not fire.
		C. Cloud	Cloud height (feet).
		D. Cloud_ht	Cloud height (meters).
		E. Des	The designator discriminator, used to discriminate between high and low Pk targets. Used in calculation of rounds which hit targets.
		F. Dust_index	Atmospheric obscuration is: 1 = Light 2 = Medium 3 = Heavy.
		G. Fir_typ (I)	Flag, whose value: 1 = PGM is being fired 2 = PGM not fired indicates the play of PGM I, I = 1 - CLGP = 2 - GAMP.

Table 6-11. PGM subroutine table.

Functional area(s): A. PGM induced losses.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Pgm_atrit (continued)		H. Lase_reliab	Designator reliability factor.
		I. N_rnds (I)	The number of rounds available to be fired by the Ith firer where I = 1 - CLGP = 2 - GAMP.
		J. N_rnds_fired (I)	The number of rounds fired by the Ith PGM where I = 1 - CLGP = 2 - GAMP.
		K. N_tgts	Number of surviving targets within this attrition loop.
		L. Nfirers	Number of possible firers (set to 2 for CLGP and GAMP).
		M. N_rnds	Rounds available after degradation factors.
		N. Nslices	Number of slice loops (set to 15).
		O. Ntargets	Number of system types which are targets (set to 70).
		P. P_desg	Specific designator degradation factor according to the dust-index and the visibility-index.
		Q. P_dustabort	Specific designator degradation factor according to the dust-index and the visibility-index.

Table 6-11. PGM subroutine table.

Functional area(s): A. PGM induced losses.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Pgm_atrit (continued)		R. P_surv (I)	Cumulative probability of survival for the Ith target type.
		S. Prob_desg (I,J)	CLGP designator degradation factor. I is the cloud index and J is the visibility index.
		T. Prob_dustabort (I,J)	CLGP designator degradation factor. I is the dust index and J is the visibility index.
		U. Psurv_tf (I,J)	Probability of survival for the Jth target being fired on by the Ith PGM.
		V. Sens_typ (I)	Flag, with the following values for the Ith PGM. Flag = 0 - no sensors = 1 - for GLLD = 2 - RPV I = 1 - CLGP = 2 - GAMP.
		W. Smk	Smoke degradation factor (set to 1 for PGMs).
		X. Sskp (I,J)	Single shot kill probability for the Ith PGM firing on the Jth target. Note: CLGP SSPKs done separately.
		Y. Sskp_clgp (I)	Single shot kill probability for Clgp firing on the Ith target.

Table 6-11. PGM subroutine table.

Functional area(s): A. PGM induced losses.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Pgm_atrit (continued)		Z. Targets (I,J)	Number of targets where: I = 1 - Red targets available = 2 - Red targets killed and J is the 70 target elements.
		AA. Terr_degrd	Specific terrain degradation factor, depending on the current terrain.
		BB. Terr_factor (I)	CLGP degradation factors for terrain type I, where: I = 1 - Open = 2 - Rolling = 3 - Hilly = 4 - Mountainous.
		CC. Terr_typ	The current terrain 1 - Open 2 - Rolling 3 - Hilly 4 - Mountainous.
		DD. Tgt_mask (I,J)	Final target mask, containing Tgt_mask1 for GAMP and a combination of Tgt_mask1 and Clgp_mask for CLGP. I is the PGM; J the target.
		EE. Tgt_mask1 (I,J)	The firing ability of the Ith PGM on the Jth target.
		FF. Tgt_value (I,J)	Contains numbers from 0 to 10 representing the preference of the Ith PGM firing on the Jth target. The higher the value, the greater the preference.

Table 6-11. PGM subroutine table.

Functional area(s): A. PGM induced losses.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Pgm_atrit (concluded)		GG. Vis_ndx	Current visibility category: 1 - 7 km day 2 - 5 km day 3 - 2 km day 4 - 1 km day.
		HH. Vis_rng	Same as Vis_ndx.
		II. W_sum (I)	A weighted sum of all targets on which the Ith PGm is firing.

Section VII. Infantry

1. PURPOSE.

The purpose of the infantry module is to calculate the Red and Blue direct fire infantry losses during a 30-minute interval.

2. GENERAL.

A. The infantry module combines gamer inputs passed in through the main driver routine with firepower scores and a force multiplier to compute a firepower ratio for each force.

B. The firepower ratio is used to compute the total infantry attrition suffered by each force during a 30-minute interval.

C. For each 30-minute interval, the main driver routine combines the infantry losses with all other losses to determine a total ground combat attrition suffered by each force for that period.

3. DATA FLOW.

A. The infantry module uses driver inputs passed from the ground combat driver program to access the appropriate firepower scores and force multipliers for each force.

(1) Driver inputs. The main driver routine passes the following data into the infantry module.

(a) Sys(I,J). A 6x70 array which contains the current status of all units in conflict, where:

I = 1 - Blue targets
2 - Blue losses
3 - Red targets
4 - Red losses
5 - Blue direct fire elements
6 - Red direct fire elements

J = 1 - 70 - weapon systems

NOTE: The only J used in this subroutine is when J = 36 - 47, since these elements on the weapons list contain the infantry position.

(b) Force. An integer value of 1 or 2 which identifies the force type, where:

1 - Light force
2 - Heavy force.

(c) Cstat(I). An integer value of 1 to 10 which identifies the mission for force I where:

I = 1 - Blue force
2 - Red force

Mission = 1 - Movement to contact
2 - Indirect fire
3 - Movement
4 - Frontal attack
5 - Envelopmental attack
6 - Delay
7 - Hasty defense
8 - Prepared defense
9 - Rear area
10 - Ambush.

(d) Attacker. An integer value of 1 or 2 which identifies the attacking force, where: 1 - Blue attacking 2 - Red attacking.

(e) Hr_conflict. A real value which contains the time, in hours, for assessment of infantry battle.

(2) Firepower scores. The firepower scores are simply numerical values assigned to weapon systems to quantify their potential to inflict damage.

(a) The firepower scores used in the DIME game were derived from the Concepts Analysis Agency's (CAA) Weapon Effectiveness Indices/Weighted Unit Values (WEI/WUV) methodology. The DIME firepower scores are classified and may be found in volume III of this report.

(b) The firepower scores file consists of two records containing the firepower score for the Blue (record 1) and the Red (record 2) forces. Each record is a three dimensional array, where:

I = 1 to 70 weapon systems

J = 1 - light force
2 - heavy force

K = 1 - Red attacking/Blue defending
2 - Blue attacking/Red defending.

(3) Force multiplier. A real value which reflects an adjustment factor for maneuver unit weapons is either an attack or defend posture. Table 6-12 contains a list of the force multipliers by battle posture and mission. These multipliers, dependent on attacker/defender and mission status, are established as equations within the code.

Table 6-12. Infantry force multipliers.

Tactical Situation	Battle posture: Attacker	Battle posture: Defender
Movement to contact	1.5	1.0
Indirect fire	1.5	1.0
Movement	1.5	1.0
Frontal attack	1.5	1.0
Envelopmental attack	1.5	1.0
Delay	1.5	1.0
Hasty defense	1.5	1.2
Prepared defense	1.3	1.5
Rear area	2.0	0.5
Ambush	1.0	4.5

B. The infantry module returns the total losses for both Blue and Red forces.

C. Figure 6-38 indicates the generalized data flow for the infantry module.

Table 6-14. Ground combat code (continued).

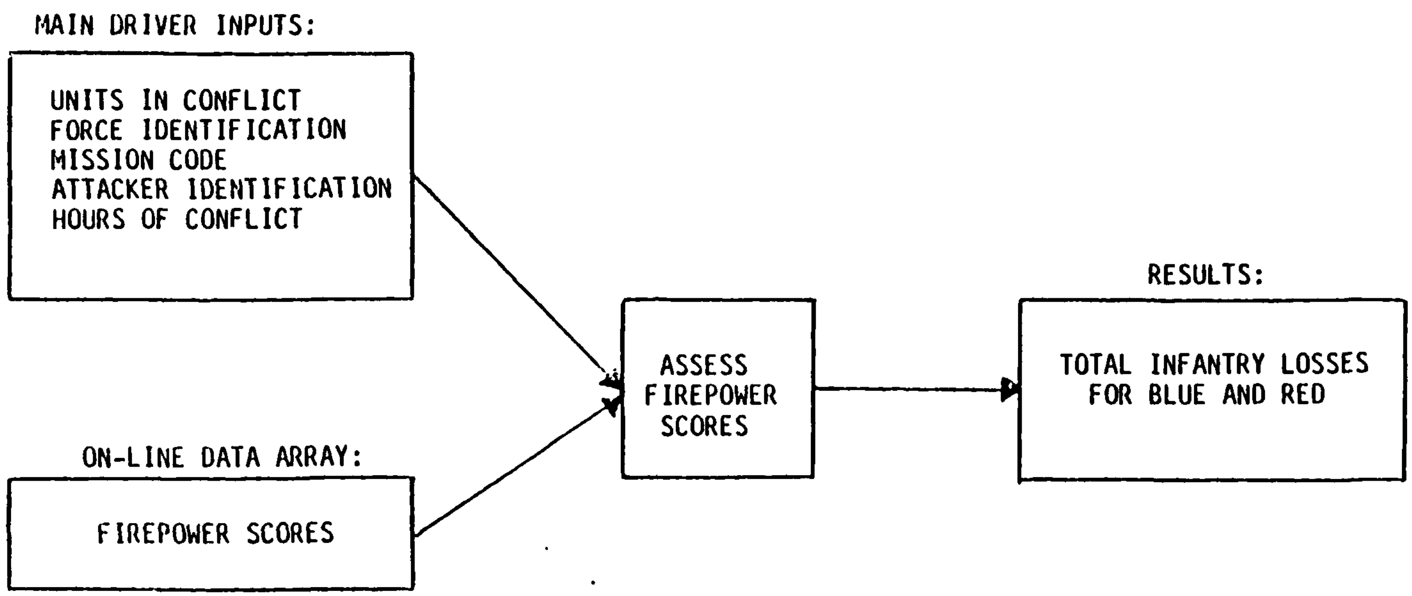


Figure 6-38. Infantry data flow.

4. FILE STRUCTURE.

The only data stored in an auxiliary file are the firepower scores. The firepower scores are read into two on-line data arrays, Fpsb(*) and Fpsr(*).

A. Fpsb (I,J,K). An array with dimensions (70,2,2) which contains the Blue firepower scores for the 70 weapon elements assigned to one of two force types, either attacking or defending.

I = 1 to 70 weapon systems

J = 1 - light force
    2 - heavy force

K = 1 - Red attacking/Blue defending
    2 - Blue attacking/Red defending.

B. Fpsr (I,J,K). An array containing the Red firepower scores. The definitions of I, J, and K are the same as above.

5. ALGORITHMS.

A. The infantry module combines the gamer inputs, force multiplier, and firepower scores discussed in paragraphs 3 and 4 with the generalized logic flow shown in Figure 6-39 to calculate the total losses suffered by each force.

(1) The program begins by calculating a firepower ratio.

(a) A firepower ratio is a measure of one force's capability to inflict damage relative to the capability of another force. In forming such a ratio, the tactical situation of the maneuver units of both the attacking and defending forces are considered, and the firepower scores are adjusted accordingly. For instance, a defending force would expect to be less vulnerable if it were occupying a fortified defensive position than if it were engaging the enemy in the open. Likewise, an attacking force would expect to inflict greater damage executing a frontal attack against a unit in a hasty defense, as a unit in a prepared defense.

(b) The unadjusted total firepower score for each force is multiplied by the appropriate tactical situation adjustment factor (see Table 6-12, infantry force multipliers). The attacker-to-defender firepower ratio is then calculated. The firepower ratio calculation is expressed algebraically as:

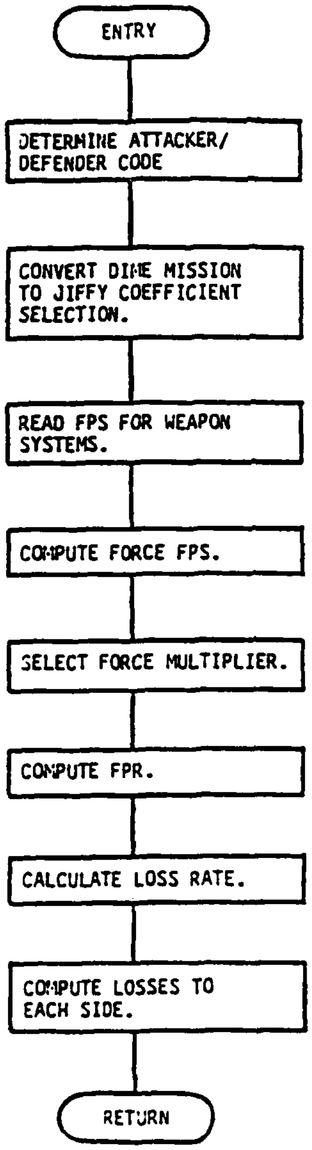


Figure 6-39. Generalized infantry logic flow.

$$Fpr = \frac{\left(\sum_{i=1}^{70} (As_i * Afps_i)\right) * Atsaf}{\left(\sum_{i=1}^{70} (Ds_i * Dfps_i)\right) * Dtsaf} \qquad \text{(Eq. 6-86)}$$

where:

- Fpr = the firepower ratio.
- Atsaf = the attacker tactical situation adjustment factor (see Table 6-12).
- Dtsaf = the defender tactical situation adjustment factor (see Table 6-12).
- $Dfps_i$ = the firepower score of the defender's i^{th} system.
- $Afps_i$ = the firepower score of the attacker's i^{th} system.
- As_i = the number of attacking systems (i)
- Ds_i = the number of defending systems (i)

(2) This calculated firepower ratio is used to compute the casualty rates for defending and attacking forces.

(a) Defending forces. The casualty rate for a defending force is a function of the combat force ratio (FPR) and the mission of the defending force. Figure 6-40 shows a graphical representation of the casualty rate for ground combat personnel in one of six defense missions. Combining the graphs with a curve fitting equation using the FPR, the casualty rate is calculated by:

$$Drate = A + B * Fpr + C * Fpr^2 \qquad \text{(Eq. 6-87)}$$

where:

- Drate = defending force casualty rate.
- Fpr = firepower ratio.
- A = Y-intercept.
- B,C = constants determined from curve fitting (see Figure 6-40).

Defending forc' coefficients have been calculated as follows:

1. Defending from prepared positions.

A = .00919
B = .004085
C = .000097

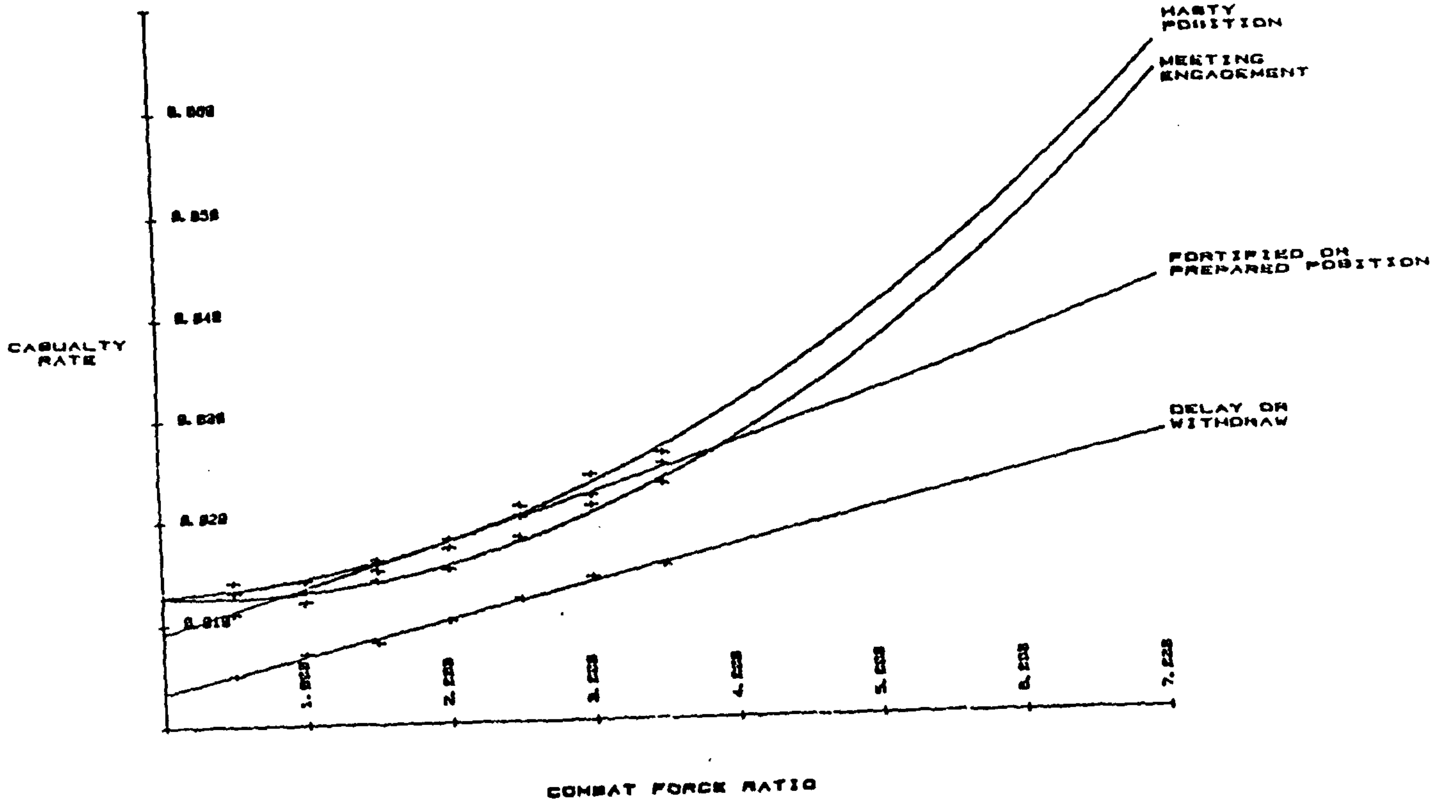


Figure 6-40. Ground combat personnel casualty rate for defending forces.

2. Defending from hasty positions.

A = .01274
B = .0005
C = .001

3. Defending against an enemy fighting a meeting engagement.

A = .001257
B = .000857
C = .001143

4. Defending force delaying or withdrawing against attackers.

A = .003286
B = .0034286
C = 0.0

(b) Attacking forces. The casualty rate for an attacking force is a function of the combat force ratio (FPR) and the mission of the attacking force. Figure 6-41 shows the graphical representation of the attacking forces. Again, rate is calculated by combining the FPR with a curve-fitting equation:

$$Arate = A * Fpr^{-B} \qquad \text{(Eq. 6-88)}$$

where:

Arate = attack force casualty note.
Fpr = firepower ratio.
A,B = constants determined from fitting curve (see Figure 6-41).

Attacking force coefficients have been calculated as follows:

1. Against a fortified or prepared position:

A = .0483
B = .251

2. Against a hasty position:

A = .0401
B = .237

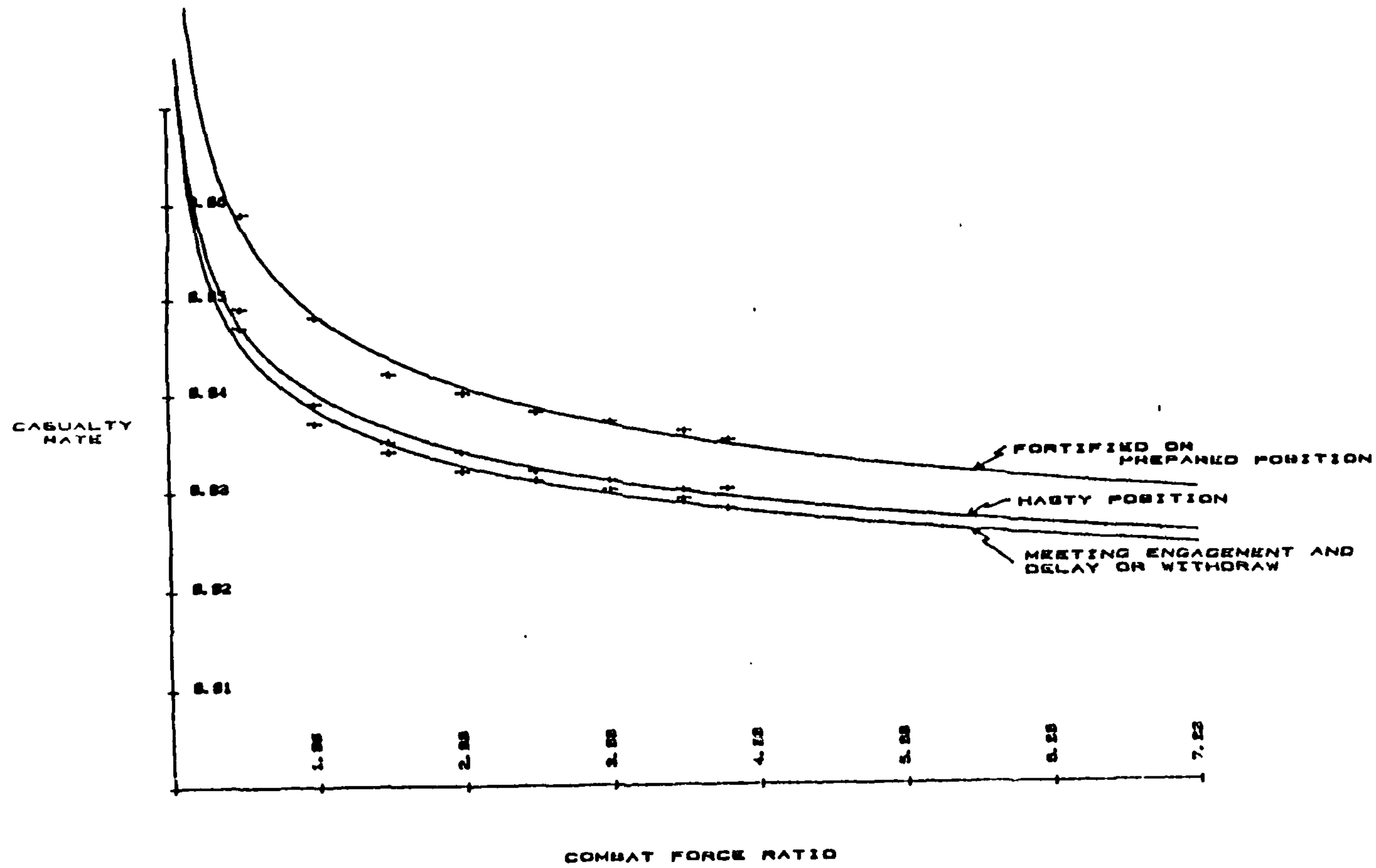


Figure 6-41. Ground combat personnel casualty rate for attacking forces.

3. Against a delaying or withdrawing enemy or fighting a meeting engagement:

$$A = .0384$$
$$B = .2383$$

(3) Finally, the total losses suffered by each force are calculated using the following formula:

$$Loss_i = Pers_i * Frac\_comtd * Rate_i * Hr\_conflict \qquad \text{(Eq. 6-89)}$$

where:

- i = 1 - Blue force
  2 - Red force.
- $Loss_i$ = total losses per battle.
- $Pers_i$ = total personnel assigned to unit.
- Frac_comtd = percent of unit committed to fight battle, where value is defaulted to 1.0.
- $Rate_i$ = casualty rate loss.
- Hr_conflict = time, in hours, for assessment of infantry battle.

B. These losses are returned to the main driver routine where they are combined with other losses to determine the total ground combat losses suffered by each force.

6. "UNITFILE" IMPACT.

The infantry module returns the infantry losses to the ground combat driver. The driver then deducts the personnel losses from the infantry elements of the "UNITFILE".

7. CODE.

A. The infantry module consists of a driver routine and two subroutines.

(1) Main driver. The main driver establishes the initial parameters and pointers for reading in the appropriate data files.

(2) Once the correct data has been read into the program, the main driver routine calculates the firepower ratio by multiplying the firepower scores by the force multiplier.

(3) The main driver then calls two subroutines: Rate and Losses.

(a) Rate. The Rate routine uses the firepower ratio to compute the casualty loss rate.

(b) Losses. The Loss routine uses the casualty loss rate to compute the total personnel losses by a force for a specified battle time.

(4) The personnel losses are returned to the ground combat attrition module where they are used to compute the total kills for the 30-minute battle phase being evaluated.

B. The primary variables for the infantry module are shown in Table 6-13. Each variable is accompanied by a short description. See Table 6-14 for the ground combat code listing.

Table 6-13. Infantry subroutine table.

Functional area(s): A. Game initialization.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main Driver	Establishes initial pointers and parameters for reading in data.	A. Sys (*)	A 6 x 70 array containing the current status of all units in conflict.
		B. Force	In integer value of 1 or 2 which identifies the force type. 1 = light force 2 = heavy force.
		C. Cstat (I)	An integer value of 1 to 10 which identifies the mission for force I.
		D. Attacker	An integer value identifying the attacking force. 1 = Blue attacking 2 = Red attacking.
		E. Hr_conflict	A real value containing the time (hours) used to assess the infantry battle.
		F. Converta (*)	A 10 x 10 array used to select the appropriate loss coefficient equation for the attacker.
		G. Convertd (*)	A 10 x 10 array used to select the appropriate loss coefficient equation for the defender.
		H. Defender	Flag identifying the defending force. 1 = Blue defending 2 = Red defending.

Table 6-13. Infantry subroutine table.

Functional area(s): A. Game initialization.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main Driver (continued)	Accesses and reads in appropriate data files.	A. FPS1	A real file containing the firepower scores for Blue and Red forces.
		B. What (I)	An integer value of 11-16 or 21-23 which identifies the casualty loss equation/coefficient to use in calculating the casualty loss rate for each force. I = 1 - Attacking force = 2 - Defending force.
		C. Fpsb (I,J,K) Fpsr (I,J,K)	A 70 x 2 x 2 array containing the Blue/Red firepower score for the 70 elements (I). J = 1 - Light force = 2 - Heavy force K = 1 - Attacking = 2 - Defending.
	Compute adjusted firepower score for the forces.	A. Fratio (I)	The total unadjusted firepower scores for the forces. I = 1 - Blue = 2 - Red.
		B. Top	The total adjusted firepower scores for the attacking force.
		C. Bottom	The total adjusted firepower scores for the defending force.

Table 6-13. Infantry subroutine table.

Functional area(s): A. Game initialization.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Main Driver (concluded)		D. Dmultiply	Adjustment factor multiplier for a defending force.
		E. Amultiply	Adjustment factor multiplier for an attacking force.
	Compute firepower ratio.	A. Fpr	A real value, containing the ratio of the adjusted firepower scores of the attacker to the defender.

Functional area(s): B. Calculate rate of casualty losses.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Rate	Accesses correct equation/coefficient for computing rate of casualty losses.	A. Brate (I)	The rate of casualty losses for forces on the attack (I=1) and defense (I=2).

Functional area(s): C. Calculate total losses per force.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Losses	Calculates loss to attacking and defending forces.	A. Pers (I)	Number of infantry personnel for the Blue (I=1) and Red (I=2) forces.
		B. Fract_comtd	Fraction of unit committed to infantry (default = 1.0).

Table 6-14. Ground combat code.

```
3    !!!  "P4" IS THE GROUND ATTRITION MODULE FOR THE DIVISION MAP
6    ! DATA CHANGED ON 21 MARCH 1985, ROB BELFLOWER, BDM
9    ! EXPANDED VERSION -- JUNE 9, 1986 -- BY OAO CORP.
12   !      DECLASSIFIED -- AUG 7, 1986 -- BY OAO CORP.   ** DC **
15   ! HELICOPTER METHODOLOGY CHANGED -- JUNE 1987 -- BY OAO CORP.
18     OPTION BASE 1
21     DIM Sys_dfl(7),Sys_dfs(7),Sys_if(7),Sys_ad(7),Sys_sum(7)
24     DIM Sys_tot(4,70),Kv_r(6,70),Kv_b(6,70),Rif_ammo(15)
27     DIM Ci_kv_r(8,70),Ci_kv_b(8,70),Sys(6,70),B_helo(3,6),R_helo(3,6)
30     DIM Ci_helo_b(3,6),Ci_helo_r(3,6),Ammo_wt(2,70),If_ammo(15),Bif_ammo(15)
33     DIM B_unit(12,74),R_unit(12,74),S(70),Wpn_type(70)
36     DIM Sys_eff(2,70),B_init(13,70),R_init(13,70),Sys_ammo(15),Bif_msn(6)
39     DIM Rif_msn(6),Bif_msn_tons(15,5),Rif_msn_tons(15,5),Arty_30min_wt(2,15)
42     DIM Bif_fired(15,5),Rif_fired(15,5),Tot_arty(2,15),Ds_start(4),N(150)
45     DIM Advance_rate(4,10),Minefield(3,6),Mf(4),R_veh$[350],B_veh$[350]
48     DIM B_if_t(2,70),R_if_t(2,70),B_df_t(2,70),R_df_t(2,70)
51     DIM B_if_dt(2,70),R_if_dt(2,70),B_df_dt(2,70),R_df_dt(2,70)
54     DIM B_f(2,70),R_f(2,70),B_v(2,70),R_v(2,70),B_dv(2,70),R_dv(2,70)
57     DIM B_con(12,70),R_con(12,70),B_type(12),R_type(12)
60     DIM Mift(20),Mdft(20),Mifdt(20),Mdfdt(20),T_length(2),T_width(2)
63     DIM Mfire(20),Mdfire(20),Mvul(20),Mdvul(20),Target(2,70),Red_f_t(70)
66     DIM Phase_ct(3),Sys_mine(4,70),Sys_arty(4,70),Basic_ld(2,70),Blue_f_t(70)
69     DIM Sys_direct(2,70),Bf_mask(70),Rf_mask(70),Bdf_mask(5,70),Rdf_mask(5,70
72     DIM Blue_vul(70),Red_vul(70),B_ammo(2,70),R_ammo(2,70),Bstat(2)
75     DIM Sys_helo(2,70),Sys_pgm(2,70),Sys_inf(2,70),B_fire_sv(70),R_fire_sv(70
78     DIM B_break_t(12),R_break_t(12),B_df(2,70),R_df(2,70),B_msn(1),R_msn(1)
81     DIM Volley(15,5),Saty(15),A_wt(2,15),Tot_volley(15),Label$[75]
84     DIM B_arty_cap(7,5),B_mlrs_cap(4,5),B_mort_cap(4,5),H_targ(4,70)
87     DIM R_arty_cap(7,5),R_mlrs_cap(4,5),R_mort_cap(4,5)
90     DIM Clgp_fact(70),Gamp_fact(70),B_clgp_cap(7),B_gamp_cap(4)
93     DIM C_targ(2,70),C_t(4,70),R_vis(3),B_vis(3),R_inf_save(5),B_inf_save(5)
96     DIM B_engagements(20),R_engagements(20),Barty_30min(10,15)
99     DIM Rarty_30min(10,15),B_smok_tons(11),R_smok_tons(11),Inf_surv(5)
102    DIM B_smk_cap(11),R_smk_cap(11),Bada_hnd(12),Bada_veh(12),Rada_hnd(12)
105    DIM B_dsarty_avail(7),B_dsarty_fire(7),B_dsmort_avail(4),B_dsmort_fire(4)
108    DIM R_dsarty_avail(7),R_dsarty_fire(7),R_dsmort_avail(4),R_dsmort_fire(4)
111    DIM B_asmk_used(7),B_msmk_used(4),R_asmk_used(7),R_msmk_used(4)
114    DIM Frac_arty(7),Frac_mort(4),Ds_attempted(7),Mo_attempted(4)
117    DIM Fir_typ(2),Sens_typ(2),N_rnds(2),Rada_veh(12),B_inf(1,5),R_inf(1,5)
120    DIM Incoming_arty(7),Incoming_mlrs(4),Incoming_mort(4),B_ech(12),R_ech(1
123    DIM B_unit_no(12),R_unit_no(12),Hfile(2,12,10)
126    DIM B_helo_atkprof(2),R_helo_atkprof(2),B_helo_msn(2),R_helo_msn(2)
129    DIM B_atk_rg(2),R_atk_rg(2),H_d_$[64],H_msn$[32],Stnd_off_rg(2,3)
132    DIM Side$(2)[2],Helo_char(3,8),Desc$[8],P_det_inf(3,5,2),P_det_tbar(3,5,2
135    DIM Rmin(8),Rmax(8),Pk(3,20,2),Tim_me(6),Pref(20),Plos(8),Pd_inf_ad(3,2)
138    DIM Pd_tbar_ad(3,2),Pk_ad(3,2),Pref_ad(3),Cat20(20),Rg_msn$(3)[10]
141    DIM Atk_prof(2,3),Helo_mis(2,3),P_def(2,70),Arty(2),Ad_helo(2),Ad_sv(2)
144    DIM Veh_ada(2),Hnd_ada(2),Helo_tgt(2,2,70),Sen_ptr(20),Mun_ptr(20)
147    DIM Df_det_inf(3,2),Df_det_tbar(3,2)
150  !
153    INTEGER I,J,K,B_unit_pct(12),R_unit_pct(12),St_time,Minute,Earliest_time
```

Table 6-14. Ground combat code (continued).

```
156  INTEGER Delay_minute,R_minute,B_minute,R_prep_time,B_prep_time
159  INTEGER H_side,Muni,Jtarg,M,Iad,Jmast
162  '
165  COM /Mines/ Mine_frct(4,70)
168  COM /Infantry/ Convertd(10),Converta(10,10),Fpsb(70,2,2),Fpsr(70,2,2)
171  COM /Artv/ B_area_band(5),R_area_band(5),B_disprsn_mask(3,10),R_disprsn_m
k(3,10),B_tgt_mask(5,72),R_tgt_mask(5,72),B_rd_wt(15),R_rd_wt(15)
174  COM /Arty/ B_psnl_posture(2,2),R_psnl_posture(2,2),Tle(5)
177  COM /Smoke/ Amwtpp(3,11),Irof(3,11)
180  COM /Pgm/ Tgt_value(2,70),Tgt_mask1(2,70),Terr_factor(4),Prob_dustabort(
),Clgp_msk_ns(70),Clgp_msk_gl(70),Clgp_msk_rp(70)
183  COM /Pgm/ Prob_dsg_ns(6,4),Prob_dsg_gl(6,4),Prob_dsg_rp(6,4),Sskp_ns(70),
kp_gl(70),Sskp_rp(70)
186  COM /Direct_fire/ B_cat(70),R_cat(70),B_sen_d(70),B_sen_n(70),R_sen_d(70)
_sen_n(70),B_ammo_wt(20),R_ammo_wt(20)
189  COM /No_helos/ Cell(2,2,3)
192  '
195  COM /Helo_attrite/ Helo_load(2,3,3),Pd_fe_inf_a(2,3,5),Pd_fe_inf_b(2,3,5)
d_fe_inf_c(2,3,5),Pd_hd_inf_a(2,3,5),Pd_hd_inf_b(2,3,5)
198  COM /Helo_attrite/ Pd_hd_inf_c(2,3,5),Pd_fe_tbar_a(2,3,5),Pd_fe_tbar_b(2,
5),Pd_fe_tbar_c(2,3,5),Pd_hd_tbar_a(2,3,5),Pd_hd_tbar_b(2,3,5)
201  COM /Helo_attrite/ Pd_hd_tbar_c(2,3,5),Pd_rmin(2,3,8),Pd_rmax(2,3,8),Pk_f
a(2,3,3,20),Pk_fe_b(2,3,3,20),Pk_fe_c(2,3,3,20),Pk_hd_a(2,3,3,20)
204  COM /Helo_attrite/ Pk_hd_b(2,3,3,20),Pk_hd_c(2,3,3,20),Pk_rmin(2,3,3),Pk_
ax(2,3,3),Np(2,3,3),Fm(2,3,3),Tm(2,3,3),Te(2,3,3)
207  COM /Helo_attrite/ Plos_alpha(2,3),Plos_beta(2,3),Pd_inf_ad_a(2,7,2),Pd_i
_ad_b(2,7,2),Pd_inf_ad_c(2,7,2),Pd_tbar_ad_a(2,7,2),Pd_tbar_ad_b(2,7,2)
210  COM /Helo_attrite/ Pd_tbar_ad_c(2,7,2),Pd_ad_rmin(2,7,8),Pd_ad_rmax(2,7,8
Pk_ad_a(2,7,2),Pk_ad_b(2,7,2),Pk_ad_c(2,7,2),Pk_ad_rmin(2,7),Pk_ad_rmax(2,7)
213   COM /Helo_attrite/ Rnd_wt(2,7),Rnds(2,7),Fad(2,7),Pd_inf_df_a(2,2,2),Pd_
f_df_b(2,2,2),Pd_inf_df_c(2,2,2),Pd_tbar_df_a(2,2,2),Pd_tbar_df_b(2,2,2)
216   COM /Helo_attrite/ Pd_tbar_df_c(2,2,2),Pd_df_rmin(2,2,8),Pd_df_rmax(2,2,
,Df_rnds_eng(2,2),F_df(2,2)
219   COM /Helo_attrite/ INTEGER Mast_mnt(2,3),Tgt_pref(2,3,20),Ad_pref(2,7,3)
d_cat(2,20)
222  '
225  COM /Helo_info/ Btl_rg,Rg_avg(2,3,20),Rg_avg_pd(2,3,5),Df_ammo(2),Df_fire
ist(2,20,3),Df_pk_helo(2,20,3,2),INTEGER Df_sen_ptr(2,20),Df_muni_ptr(2,20)
228  !
231  DIM Disk$[50],Disk3$[50]
234  Disk$=":9134,704,0"
237  DIM M$[16],N$[16]    !ROB
240  Disk3$=":9134,704,0"
243  Dcdisk$=":9134,704,0"    ! ** DC **
246  ASSIGN @Unitpath TO "UNITFILE"&Disk$
249  ASSIGN @Kvpath TO "KVFILE"&Disk$
252  ASSIGN @Helopath TO "HELOFILE"&Disk$
255  'ASSIGN @Ammopath TO "AMMOFILE"&Disk3$
258  ASSIGN @Advanpath TO "ADVANFILE"&Disk$
261  ASSIGN @Fname TO "NAMEFILE:9134,704,0"   ! ROB
264  ASSIGN @Rname TO "NAMEFILE:9134,704,0"   ! ROB
267   '
```

Table 6-14. Ground combat code (continued).

```
270   ! READ IN SMALL PERMANENT DATA FILES
273   GOSUB Read_data
276   !
279  Start_battle:     !    START OF THE ATTRITION MODULE
282   PRINTER IS 1
285   PRINT USING "@,#"
288   PRINT TABXY(30,17),"GROUND COMBAT MODULE"
291   !
294   PRINTER IS 702
297   ! CONDUCT SECTOR ATTRITION ASSESSMENTS
300   !
303   GOSUB Zero_out                              !  INITIALIZE VARIABLES
306   GOSUB Set_battle                            !  INPUT BATTLE CONDITIONS
309   GOSUB Set_conditions                        !  SET ALL BATTLE CONDITIONS
312   GOSUB Control_battle                        !  CONTROLS BATTLE FLOW
315   GOSUB Print_fin_res                         !  PRINTS SECTOR BATTLE RESUL
318   !
321   PRINTER IS 1
324   ! LET GAMER ANALYZE RESULTS FOR PROPER BATTLE PORTRAYAL
327   INPUT "UPDATE THE HISTORY FILE WITH THESE RESULTS?  (Y or N)",Q$
330   IF Q$<>"Y" AND Q$<>"N" THEN 327
333   !
336   ! WRITE SECTOR RESULTS TO THE HISTORY FILE
339   IF Q$="Y" THEN
342     GOSUB Apport_wri_loss
345   END IF
348   !
351   GOSUB Close_files
354   !
357   ! CHECK FOR MORE COMBAT
360   INPUT "MORE SECTORS TO PROCESS?  (Y or N)",Q$
363   IF Q$<>"Y" AND Q$<>"N" THEN 360
366   IF Q$="Y" THEN LOAD "NEW_F4"&Disk$
369   !
372   ! IF COMBAT IS DONE, MAKE A FILE COPY.  IF NOT, GO BACK TO MAIN MENU
375   INPUT "IS ALL COMBAT ASSESSED FOR THIS TURN?  (Y or N)",Q$
378   IF Q$<>"Y" AND Q$<>"N" THEN 375
381   IF Q$="N" THEN
384     LOAD "DIME:9134,704,0"
387     DISP "GOING BACK TO DIME MENU"
390     GOTO Halt
393   ELSE
396      !OPTIONAL UNITFILE BACKUP MAY BE PLACED HERE.
399     LOAD "DIME:9134,704,0"
402     GOTO Halt
405   END IF
408   !
411   !********************* END OF MAIN PROGRAM ****************************
414   !
417  Read_data:   !   THIS SBR READS SMALL ARRAYS INTO THE PROGRAM
420  !
423  !   ** DC **
```

Table 6-14. Ground combat code (continued).

```
426 !
429  ASSIGN @Psyseff TO "SYS_EFF"&Dcdisk$
432  ENTER @Psyseff,1;Sys_eff(*)          ! FIREPOWER SCORE OF RED/BLUE WEAPO
435  ASSIGN @Psyseff TO *
438 !
441  ASSIGN @Pwpntyp TO "WPN_TYPE"&Dcdisk$
444  ENTER @Pwpntyp,1;Wpn_type(*)              ! 1=DF  2=IF  3=AD
447  ASSIGN @Pwpntyp TO *
450 !
453  ASSIGN @Pammowt TO "AMMO_WT"&Dcdisk$
456  ENTER @Pammowt,1;Ammo_wt(*)          ! PACKED WT OF INDIV RD/BURST OF AMM
459  ASSIGN @Pammowt TO *
462 !
465  ASSIGN @Pbasld TO "BASIC_LD"&Dcdisk$
468  ENTER @Pbasld,1;Basic_ld(*)
471  ASSIGN @Pbasld TO *
474 !
477  ASSIGN @Partrt TO "ARTY_RATE"&Dcdisk$
480  ENTER @Partrt,1;Arty_30min_wt(*)
483  ASSIGN @Partrt TO *   ! RED & BLUE 30 MIN WT DEFINED AT THIS POINT FOR A
 DSTRBTN
486 !
489  ASSIGN @Partwt TO "ARTY_WT"&Dcdisk$
492  ENTER @Partwt,1;A_wt(*)      ! WT OF IF ROUND/PACKAGED WT - VOLLEY WT. FO
 BATTERIES
495  ASSIGN @Partwt TO *
498 !
501  ASSIGN @Pifmask TO "BFMASK"&Dcdisk$
504  ENTER @Pifmask,1;Bf_mask(*)
507  ASSIGN @Pifmask TO *
510 !
513  ASSIGN @Pifmask TO "RFMASK"&Dcdisk$
516  ENTER @Pifmask,1;Rf_mask(*)
519  ASSIGN @Pifmask TO *
522 !
525  ASSIGN @Pdfmask TO "BDF_MASK"&Dcdisk$
528  ENTER @Pdfmask,1;Bdf_mask(*)
531  ASSIGN @Pdfmask TO *
534   !
537  ASSIGN @Pdfmask TO "RDF_MASK"&Dcdisk$
540  ENTER @Pdfmask,1;Rdf_mask(*)
543  ASSIGN @Pdfmask TO *
546 !  ** END DC **
549  B_veh$[1,125]="DF   FAV-TM551 FAV4OHMV-GDF   DRAGNLAW  DF   CMD-VDF   DF
DF   DF   DF   HMV4ODF-ICDF-ICDF-ICDF-ICARTY ARTY ARTY ARTY ARTY "
552  B_veh$[126,250]="ARTY ARTY MORTRMORTRMORTRMORTRMLRSTMLRSTMLRSTMLRSTINF  I
  INF  INF  INF  SARMSSARMSSARMSSARMSSARMSSARMSSARMSVULCNAVNGRIHAWK"
555  B_veh$[251,350]="ADA  ADA  STINGADAHHF-TRKJ4TRKWATERCGO-TNATRKEWTRKEWTRKE
R OBSCEAVLB PONBRENGEOENGFOMATHEMATHEAATHE"
558  R_veh$[1,125]="T55  DF   BMP73DF   BRDM3BRDM5AT-75AGS17T12  CMD-VDF   DF
DF   DF   DF   BMPATBTR  DF-ICDF-ICDF-ICARTY ARTY ARTY ARTY ARTY "
561  R_veh$[126,250]="ARTY ARTY MORTRMORTRMORTRMORTRMRL  MRL  MRL  MRL  INF  I
```

Table 6-14. Ground combat code (continued).

```
  INF   INF   INF   SARMSSARMSSARMSSARMSSARMSSARMSSARMSZSU-XSA-13SA-6 "
564  R_veh$[251,350]="ADA   ADA   SA-14ADAHHF-TRKJ4TRKWATERCGO-TNATRKEWTRKEWTRKE
R OBSCEAVLB FONBRENGEOENGEOMATHEMATHEAATHE"
567 !
570  !   ** DC **
573  !
576  ASSIGN @Pdsst TO "DS_START"&Dcdisk$
579  ENTER @Pdsst,1;Ds_start(*)           !START RANGE FOR ARTY DS (CLOSE SPT)
582  ASSIGN @Pdsst TO *
585 !
588  ASSIGN @Partaloc TO "BARTYALLOC"&Dcdisk$
591  ENTER @Partaloc,1;Barty_30min(*)
594  ASSIGN @Partaloc TO *
597 !
600  ASSIGN @Partaloc TO "RARTYALLOC"&Dcdisk$
603  ENTER @Partaloc,1;Rarty_30min(*)
606  ASSIGN @Partaloc TO *
609 !  ** END DC **
612  RETURN
615    !
618    !-------------------------------------------------------------------
621    !
624 Zero_out:    !  THIS SBR INITIALIZES VARIABLES USED IN THE COMBAT MODULE
627    !
630  Dc=0
633    !
636  FOR I=1 TO 12
639    B_unit_no(I)=0                ! UNIT# OF SECTOR UNITS
642    R_unit_no(I)=0
645    B_unit_pct(I)=0               ! % OF UNIT COMMITTED TO SECTOR
648    R_unit_pct(I)=0
651  NEXT I
654  Init_b_eff=0                    ! BLUE INITIAL FIREPOWER SCORE
657  B_df_ammo=0                     ! DIRECT FIRE AMMO AVAILABLE TO FIRE
660  B_ad_ammo=0                     ! AIR DEFENSE AMMO AVAILABLE TO FIRE
663  Init_r_eff=0
666  R_df_ammo=0
669  R_ad_ammo=0
672  FOR I=1 TO 15
675    Bif_ammo(I)=0                 ! INDIRECT FIRE AMMO AVAILABLE IN THIS SECT
678    Rif_ammo(I)=0
681  NEXT I
684  FOR I=1 TO 3
687    Phase_ct(I)=0
690  NEXT I
693  FOR I=1 TO 12
696    FOR J=1 TO 74
699      R_unit(I,J)=0               ! CURRENT SYSTEMS ALIVE BY UNIT# AND SYSTEM
702      B_unit(I,J)=0
705    NEXT J
708  NEXT I
711  FOR I=1 TO 70
```

Table 6-14. Ground combat code (continued).

```
714     FOR J=1 TO 6
717       Kv_r(J,I)=0                    ! KV TABLES FOR SECTOR BATTLE
720       Kv_b(J,I)=0
723       Sys(J,I)=0                     ! SUBPROGRAM PASSING ARRAY FOR SYSTEMS
726     NEXT J
729     FOR J=1 TO 4
732       Sys_tot(J,I)=0                 ! CUMULATIVE SYSTEM STATUS
735     NEXT J
738     FOR J=1 TO 12
741       B_init(J,I)=0                  ! INITIAL UNIT SYSTEMS BY UNIT (J), SYS (I
744       R_init(J,I)=0
747       R_con(J,I)=0
750       B_con(J,I)=0
753     NEXT J
756     B_init(13,I)=0
759     R_init(13,I)=0
762   NEXT I
765   FOR I=1 TO 3
768     FOR J=1 TO 6
771       R_helo(I,J)=0                  ! HELICOPTER RESULTS
774       B_helo(I,J)=0
777       Minefield(I,J)=0               ! SECTOR MINEFIELD INFO
780     NEXT J
783   NEXT I
786   Time_seg=0                         ! # OF THE CURRENT 30 MIN TIME SEGMENT
789   Last_bah1_seg=0
792   Last_bah2_seg=0                    ! TIME SEGMENT WHEN HELICOPTER LAST FLOWN
795   Last_bsct_seg=0
798   Last_rah1_seg=0
801   Last_rah2_seg=0
804   Last_rsct_seg=0
807   Rah1_seg=0
810   Rah2_seg=0
813   Rsct_seg=0
816   Bah1_seg=0                         ! # OF HELO MISSIONS FLOWN IN CELL=3 STATUS
819   Bah2_seg=0
822   Bsct_seg=0
825   FOR I=1 TO 7
828     B_dsarty_fire(I)=0
831     R_dsarty_fire(I)=0                  ! # OF TONS OF AMMO FIRED FIRED IN CS M
ION
834   NEXT I
837   FOR I=1 TO 4
840     B_dsmort_fire(I)=0
843     R_dsmort_fire(I)=0
846   NEXT I
849   Barty_fire=0
852   Rarty_fire=0
855   Mine_delay=0                       ! TIME DELAY DUE TO MINES
858   First_df=0                         ! POINTER FOR 1ST ENTRY INTO DIRECT FIRE B
861     !
864   FOR I=1 TO 15
```

Table 6-14. Ground combat code (continued).

```
867    FOR J=1 TO 5
870      Bif_msn_tons(I,J)=0
873      Rif_msn_tons(I,J)=0
876    NEXT J
879  NEXT I
882   !
885  RETURN
888   !
891   !---------------------------------------------------------------------
894   !
897 Set_battle:    !  THIS SBR ALLOWS INPUT OF THE SECTOR WORK SKEET
900   !
903  PRINTER IS 1
906 Start_input:  '
909  GOSUB L1
912  GOSUB L2
915  GOSUB L3
918  GOSUB L4
921  GOSUB L5
924  GOSUB L6
927  GOSUB L7
930  GOSUB L8
933  GOSUB L9
936  GOSUB L10
939  GOSUB L11
942  GOSUB L12
945  Beg=1
948  End=No_minefields
951  GOSUB Lmines
954 Auto:PRINTER IS 1
957  PRINT "GROUND COMBAT INPUT:"
960  GOSUB Dump_input
963  REPEAT
966    INPUT "IS INPUT CORRECT? (Y/N)",Answer$
969  UNTIL Answer$="Y" OR Answer$="N"
972  IF Answer$="N" THEN
975    REPEAT
978      INPUT "NUMBER OF INCORRECT LINES?",Num
981    UNTIL Num>=0 AND Num<=15
984    IF Num=0 THEN Auto
987    FOR Nums=1 TO Num
990      INPUT "LINE TO BE CORRECTED? Note: line 2 is blue units. line 3 is rec
",Nul
993      ON Nul GOSUB L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15
996    NEXT Nums
999    GOTO Auto
1002 END IF
1005 GOSUB Ad_sup
1008 REPEAT
1011   INPUT "DO YOU WANT A HARD COPY OF THE INPUT? (Y/N)",Answer$
1014 UNTIL Answer$="Y" OR Answer$="N"
1017 IF Answer$="Y" THEN
```

Table 6-14. Ground combat code (continued).

```
1020   PRINTER IS 702
1023   PRINT USING "@,#"
1026   PRINT "GROUND COMBAT INPUT:"
1029   GOSUB Dump_input
1032   PRINTER IS 1
1035 END IF
1038 PRINTER IS 1
1041 PRINT USING "@,#"
1044 PRINT TABXY(30,17),"GROUND COMBAT MODULE"
1047 PRINT
1050 PRINT "                                   ** BATTLE HISTORY **"
1053 PRINTER IS 702
1056 FOR I=1 TO 70
1059   Sys_tot(1,I)=B_init(13,I)
1062   Sys_tot(3,I)=R_init(13,I)
1065 NEXT I
1068 !
1071 ! DETERMINE INFANTRY LOAD FACTORS
1074 Side=1
1077 GOSUB Ready_load
1080 CALL Load_infantry(Sys_tot(*),B_msn(1),Side_pt,Sum_inf,Sum_df,B_ld_fact
1083 Side=2
1086 GOSUB Ready_load
1089 CALL Load_infantry(Sys_tot(*),R_msn(1),Side_pt,Sum_inf,Sum_df,R_ld_fact
1092    ! DETERMINE BLUE ELEMENT PERCENTAGES
1095 INPUT "DO YOU WANT TO CHANGE BLUE MISSION TEMPLATE FILES?",Chgbmt$
1098 Listop=0
1101 IF Chgbmt$="Y" THEN
1104   INPUT "DISPLAY VALUES TO 1=SCREEN ONLY, 2=SCREEN & PRINTER ?",Listop
1107   IF Listop=2 THEN
1110     PRINTER IS 702
1113     PRINT USING "@"
1116   END IF
1119 END IF
1122 Side=1
1125! FOR I=1 TO 2
1128 J=B_msn(1)
1131 ASSIGN @P TO "BIFTARG"&Disk3$
1134 ENTER @P,J;Mift(*)
1137 ASSIGN @P TO "BDFTARG"&Disk3$
1140 ENTER @P,J;Mdft(*)
1143 ASSIGN @P TO "BIFDT"&Disk3$
1146 ENTER @P,J;Mifdt(*)
1149 ASSIGN @P TO "BDFDT"&Disk3$
1152 ENTER @P,J;Mdfdt(*)
1155 ASSIGN @P TO "BFIRE"&Disk3$
1158 ENTER @P,J;Mfire(*)
1161 ASSIGN @P TO "BDFIRE"&Disk3$
1164 ENTER @P,J;Mdfire(*)
1167 ASSIGN @P TO "BVUL"&Disk3$
1170 ENTER @P,J;Mvul(*)
1173 ASSIGN @P TO "BDVUL"&Disk3$
```

Table 6-14. Ground combat code (continued).

```
1176 ENTER @P,J;Mdvul(*)
1179      !
1182 FOR I=1 TO 2
1185   FOR K=1 TO No_b_unit
1188     T=B_type(K)+B_ech(K)*10
1191     Hfile(I,K,1)=Mift(T)
1194     Hfile(I,K,2)=Mdft(T)
1197     Hfile(I,K,3)=Mifdt(T)
1200     Hfile(I,K,4)=Mdfdt(T)
1203     Hfile(I,K,5)=Mfire(T)
1206     Hfile(I,K,6)=Mdfire(T)
1209     Hfile(I,K,7)=Mvul(T)
1212     Hfile(I,K,8)=Mdvul(T)
1215     IF Chgbmt$="Y" THEN
1218       CALL Missn_tmpls(Side,(I),(T),(K),B_unit_no(*),Hfile(*),Listop)
1221     END IF
1224     FOR E=1 TO 70
1227       B_if_t(I,E)=Hfile(I,K,1)*B_con(K,E)+B_if_t(I,E)
1230       B_df_t(I,E)=Hfile(I,K,2)*B_con(K,E)+B_df_t(I,E)
1233       B_if_dt(I,E)=Hfile(I,K,3)*B_con(K,E)+B_if_dt(I,E)
1236       B_df_dt(I,E)=Hfile(I,K,4)*B_con(K,E)+B_df_dt(I,E)
1239       B_f(I,E)=Hfile(I,K,5)*B_con(K,E)+B_f(I,E)
1242       B_df(I,E)=Hfile(I,K,6)*B_con(K,E)+B_df(I,E)
1245       B_v(I,E)=Hfile(I,K,7)*B_con(K,E)+B_v(I,E)
1248       B_dv(I,E)=Hfile(I,K,8)*B_con(K,E)+B_dv(I,E)
1251     NEXT E
1254   NEXT K
1257 NEXT I
1260     ! CALCULATE RED TARGET PARAMETERS
1263 INPUT "DO YOU WISH TO CHANGE RED MISSION TEMPLATE FILES?",Chgrmt$
1266 IF Chgrmt$="Y" AND Listop=0 THEN
1269   INPUT "DISPLAY VALUES TO 1=SCREEN ONLY, 2=SCREEN & PRINTER ?",Listop
1272   IF Listop=2 THEN
1275     PRINTER IS 702
1278     PRINT USING "@"
1281   END IF
1284 END IF
1287 Side=2
1290! FOR I=1 TO 2
1293 J=R_msn(1)
1296 ASSIGN @P TO "RIFTARG"&Disk3$
1299 ENTER @P,J;Mift(*)
1302 ASSIGN @P TO "RDFTARG"&Disk3$
1305 ENTER @P,J;Mdft(*)
1308 ASSIGN @P TO "RIFDT"&Disk3$
1311 ENTER @P,J;Mifdt(*)
1314 ASSIGN @P TO "RDFDT"&Disk3$
1317 ENTER @P,J;Mdfdt(*)
1320 ASSIGN @P TO "RFIRE"&Disk3$
1323 ENTER @P,J;Mfire(*)
1326 ASSIGN @P TO "RDFIRE"&Disk3$
1329 ENTER @P,J;Mdfire(*)
```

Table 6-14. Ground combat code (continued).

```
1332 ASSIGN @P TO "RVUL"&Disk3$
1335 ENTER @P,J;Mvul(*)
1338 ASSIGN @P TO "RDVUL"&Disk3$
1341 ENTER @P,J;Mdvul(*)
1344 ASSIGN @P TO *
1347      !
1350 FOR I=1 TO 2
1353   FOR K=1 TO No_r_unit
1356     T=R_type(K)+R_ech(K)*10
1359     Hfile(I,K,1)=Mift(T)
1362     Hfile(I,K,2)=Mdft(T)
1365     Hfile(I,K,3)=Mifdt(T)
1368     Hfile(I,K,4)=Mdfdt(T)
1371     Hfile(I,K,5)=Mfire(T)
1374     Hfile(I,K,6)=Mdfire(T)
1377     Hfile(I,K,7)=Mvul(T)
1380     Hfile(I,K,8)=Mdvul(T)
1383     IF Chgrmt$="Y" THEN
1386       CALL Missn_tmpls(Side,(I),(T),(K),R_unit_no(*),Hfile(*),Listop)
1389     END IF
1392     FOR E=1 TO 70
1395       R_if_t(I,E)=Hfile(I,K,1)*R_con(K,E)+R_if_t(I,E)
1398       R_df_t(I,E)=Hfile(I,K,2)*R_con(K,E)+R_df_t(I,E)
1401       R_if_dt(I,E)=Hfile(I,K,3)*R_con(K,E)+R_if_dt(I,E)
1404       R_df_dt(I,E)=Hfile(I,K,4)*R_con(K,E)+R_df_dt(I,E)
1407       R_f(I,E)=Hfile(I,K,5)*R_con(K,E)+R_f(I,E)
1410       R_df(I,E)=Hfile(I,K,6)*R_con(K,E)+R_df(I,E)
1413       R_v(I,E)=Hfile(I,K,7)*R_con(K,E)+R_v(I,E)
1416       R_dv(I,E)=Hfile(I,K,8)*R_con(K,E)+R_dv(I,E)
1419     NEXT E
1422   NEXT K
1425 NEXT I
1428 End_input:RETURN
1431 !----------------------------------------------------------------------
1434 Ready_load: ! READY INFANTRY AND DIRECT FIRE LOADS
1437 Side_pt=2*Side-1
1440 Sum_inf=Sys_tot(Side_pt,36)+Sys_tot(Side_pt,37)+Sys_tot(Side_pt,38)+Sys_t(
(Side_pt,39)+Sys_tot(Side_pt,40)
1443 Sum_df=Sys_tot(Side_pt,16)+Sys_tot(Side_pt,17)+Sys_tot(Side_pt,18)+Sys_tot
Side_pt,19)+Sys_tot(Side_pt,20)
1446 RETURN
1449  !----------------------------------------------------------------------
1452  !
1455 L1: !
1458 INPUT "ENTER LINE 1:",Turn,Sector,No_b_unit,No_r_unit,St_time,End_time
1461 CALL Ck_var("# BLUE UNITS","TO",No_b_unit,0,12)
1464 CALL Ck_var("# RED UNITS","TO",No_r_unit,0,12)
1467 Minute=St_time MOD 100
1470 IF St_time<0 OR St_time>2345 OR Minute>45 THEN
1473   PRINT
1476   PRINT "** ERROR: START TIME MUST BE BETWEEN 00-2345 HRS : RE-ENTER LINE
"
```

Table 6-14. Ground combat code (continued).

```
1479   GOTO Start_input
1482 END IF
1485 IF End_time<St_time THEN End_time=End_time+2400
1488 RETURN
1491   !
1494   !-------------------------------------------------------------------
1497   !
1500 L2:   !
1503 FOR Ij=1 TO 12
1506   B_unit_no(Ij)=0
1509   B_unit_pct(Ij)=0
1512 NEXT Ij
1515 FOR Ii=1 TO 70
1518   B_init(13,Ii)=0
1521 NEXT Ii
1524 B_ad_ammo=0
1527 B_df_ammo=0
1530 FOR Jj=1 TO 15
1533   Bif_ammo(Jj)=0
1536 NEXT Jj
1539  ! INPUT BLUE UNITS
1542  !
1545 FOR I=1 TO No_b_unit
1548   INPUT "ENTER BLUE UNIT, PERCENT COMMITTED",B_unit_no(I),B_unit_pct(I)
1551   IF B_unit_no(I)<0 OR B_unit_no(I)>191 THEN
1554     PRINT
1557     PRINT "** ERROR: UNIT # ",B_unit_no(I)," NOT ALLOWED, IT MUST BE 1-:
"
1560     GOTO 1548
1563   END IF
1566   IF B_unit_pct(I)<0 OR B_unit_pct(I)>100 THEN
1569     PRINT
1572     PRINT I,"** ERROR: PERCENT NOT ALLOWED, IT MUST BE 0-100. "
1575     GOTO 1548
1578   END IF
1581 NEXT I
1584 FOR I=1 TO No_b_unit                              !  READ BLUE UNITS
1587   ENTER @Unitpath,B_unit_no(I);N(*)
1590   B_ech(I)=N(76)
1593    !RESET DETECTION STATUS ROB
1596   N(91)=.2
1599   IF N(92)<3 THEN N(92)=3
1602   OUTPUT @Unitpath,B_unit_no(I);N(*)
1605   FOR J=1 TO 70
1608     B_unit(I,J)=N(J)*B_unit_pct(I)/100
1611     B_init(I,J)=B_unit(I,J)
1614     B_init(13,J)=B_init(13,J)+B_init(I,J)
1617   NEXT J
1620   B_unit(I,71)=INT(((N(78)-INT(N(78)))*10)+.01)
1623   B_unit(I,72)=N(131)*B_unit_pct(I)/100
1626   B_unit(I,73)=N(132)*B_unit_pct(I)/100
1629   B_unit(I,74)=N(133)*B_unit_pct(I)/100
```

Table 6-14. Ground combat code (continued).

```
1632    B_type(I)=B_unit(I,71)+1
1635    B_df_ammo=B_df_ammo+B_unit(I,72)
1638    B_ad_ammo=B_ad_ammo+B_unit(I,74)
1641    Bada_hnd(I)=INT(N(80))/100
1644    Bada_veh(I)=N(80)-(Bada_hnd(I)*100)
1647    Side=1
1650     !
1653     ! CALCULATE AMMO AVAILABLE
1656    GOSUB Ammo_breakdown
1659    FOR J=1 TO 15
1662      Bif_ammo(J)=Bif_ammo(J)+If_ammo(J)*B_unit_pct(I)/100
1665    NEXT J
1668 NEXT I
1671 Bdf_ammo_sv=B_df_ammo
1674 Bad_ammo_sv=B_ad_ammo
1677 RETURN
1680   !
1683   !----------------------------------------------------------------------
1686   !
1689 L3:  !
1692 FOR Ij=1 TO 12
1695    R_unit_no(Ij)=0
1698    R_unit_pct(Ij)=0
1701 NEXT Ij
1704 FOR Ii=1 TO 70
1707    R_init(13,Ii)=0
1710 NEXT Ii
1713 R_ad_ammo=0
1716 R_df_ammo=0
1719 FOR Jj=1 TO 15
1722    Rif_ammo(Jj)=0
1725 NEXT Jj
1728   ! INPUT RED UNITS
1731   !
1734 FOR I=1 TO No_r_unit
1737    INPUT "ENTER RED UNIT, PERCENT COMMITTED",R_unit_no(I),R_unit_pct(I)
1740    IF R_unit_no(I)<192 OR R_unit_no(I)>400 THEN
1743      PRINT
1746      PRINT "** ERROR: UNIT # ",R_unit_no(I)," NOT ALLOWED, IT MUST BE 192-
0."
1749      GOTO 1737
1752    END IF
1755    IF R_unit_pct(I)<0 OR R_unit_pct(I)>100 THEN
1758      PRINT I,"** ERROR: PERCENT NOT ALLOWED, IT MUST BE 0-100."
1761      GOTO 1737
1764    END IF
1767 NEXT I
1770   !
1773 FOR I=1 TO No_r_unit                              ! READ RED UNITS
1776    ENTER @Unitpath,R_unit_no(I);N(*)
1779     !RESET DETECTION STATUS ROB
1782    N(91)=.2
```

Table 6-14. Ground combat code (continued).

```
1785    R_ech(I)=N(76)
1788     'RESET DETECTION STATUS ROB
1791    N(91)=.2
1794    IF N(92)<3 THEN N(92)=3
1797    OUTPUT @Unitpath,R_unit_no(I);N(*)
1800    FOR J=1 TO 70
1803      R_unit(I,J)=N(J)*R_unit_pct(I)/100
1806      R_init(I,J)=R_unit(I,J)
1809      R_init(13,J)=R_init(13,J)+R_init(I,J)
1812    NEXT J
1815    R_unit(I,71)=INT(((N(78)-INT(N(78)))*10)+.01)
1818    R_unit(I,72)=N(131)*R_unit_pct(I)/100
1821    R_unit(I,73)=N(132)*R_unit_pct(I)/100
1824    R_unit(I,74)=N(133)*R_unit_pct(I)/100
1827    R_type(I)=R_unit(I,71)+1
1830    R_df_ammo=R_df_ammo+R_unit(I,72)
1833    R_ad_ammo=R_ad_ammo+R_unit(I,74)
1836    Rada_hnd(I)=INT(N(80))/100
1839    Rada_veh(I)=N(80)-(Rada_hnd(I)*100)
1842    Side=2
1845     ! CALCULATE RED AMMO AVAILABLE
1848    GOSUB Ammo_breakdown
1851    FOR J=1 TO 15
1854      Rif_ammo(J)=Rif_ammo(J)+If_ammo(J)*R_unit_pct(I)/100
1857    NEXT J
1860 NEXT I
1863     !
1866 RETURN
1869  '
1872  '-----------------------------------------------------------------
1875  '
1878 Ad_sup:  !
1881 ! CALCULATE PERCENTAGE OF AIR DEFENSE SUPPRESSED
1884 Bhnd_sup=0
1887 Bveh_sup=0
1890 Rhnd_sup=0
1893 Rveh_sup=0
1896 FOR I=1 TO No_b_unit
1899    IF B_init(13,53)+B_init(13,54)<=.1 THEN Bveh_ada
1902    Bhnd_sup=Bhnd_sup+Bada_hnd(I)*(B_init(I,53)+B_init(I,54))/(B_init(13,53
B_init(13,54))
1905 Bveh_ada: '
1908    Tot_init_13=B_init(13,48)+B_init(13,49)+B_init(13,50)+B_init(13,51)+B_i
t(13,52)
1911    IF Tot_init_13<=.1 THEN End_supb
1914    Tot_init_i=B_init(I,48)+B_init(I,49)+B_init(I,50)+B_init(I,51)+B_init(I
2)
1917    Bveh_sup=Bveh_sup+Bada_veh(I)*Tot_init_i/Tot_init_13
1920 End_supb:NEXT I
1923 FOR I=1 TO No_r_unit
1926 Rhnd_ada:IF R_init(13,53)+R_init(13,54)<=.1 THEN Rveh_ada
1929    Rhnd_sup=Rhnd_sup+Rada_hnd(I)*(R_init(I,53)+R_init(I,54))/(R_init(13,5
```

Table 6-14. Ground combat code (continued).

```
R_init(13,54))
1932 Rveh_ada: '
1935   Tot_init_13=R_init(13,48)+R_init(13.49)+R_init(13,50)+R_init(13,51)+R_
t(13,52)
1938   IF Tot_init_13<=.1 THEN End_supr
1941   Tot_init_i=R_init(I,48)+R_init(I,49)+R_init(I,50)+R_init(I,51)+R_init(
2)
1944   Rveh_sup=Rveh_sup+Rada_veh(I)*Tot_init_i/Tot_init_13
1947 End_supr:NEXT I
1950 !
1953 Rdf_ammo_sv=R_df_ammo
1956 Rad_ammo_sv=R_ad_ammo
1959    ! CALCULATE BLUE TARGET PARAMETERS
1962 FOR I=1 TO No_b_unit
1965   FOR J=1 TO 70
1968     IF B_init(13,J)=0 THEN B_dem_0
1971     B_con(I,J)=B_unit(I,J)/B_init(13,J)
1974 B_dem_0:NEXT J
1977 NEXT I
1980    ! CALCULATE RED TARGET PARAMETERS
1983 FOR I=1 TO No_r_unit
1986   FOR J=1 TO 70
1989     IF R_init(13,J)=0 THEN R_dem_0
1992     R_con(I,J)=R_unit(I,J)/R_init(13,J)
1995 R_dem_0:NEXT J
1998 NEXT I
2001 RETURN
2004  !
2007  !-----------------------------------------------------------------------
2010  !
2013 L4: !
2016  !    ENTER BATTLE PARAMETERS
2019 INPUT "ENTER LINE 4:",Atk_def,Init_rg,Df_rg,No_minefields,Ride,Dis_inf
2022 CALL Ck_var("BL/R ATKR","OR",Atk_def,0,1)
2025 CALL Ck_var("INIT RG(m)","THRU",Init_rg,0,40000)
2028 IF Init_rg<=3000 THEN
2031   Limit_df_rg=Init_rg
2034 ELSE
2037   Limit_df_rg=3000
2040 END IF
2043 CALL Ck_var("DF RG(m)","THRU",Df_rg,0,Limit_df_rg)
2046 CALL Ck_var("# OF MINEFIELDS","TO",No_minefields,0,3)
2049 CALL Ck_var("MTD/DISM","OR",Ride,0,1)
2052 CALL Ck_var("INF DISM","OR",Dis_inf,0,1)
2055 FOR Ii=No_minefields+1 TO 3
2058   FOR Jj=1 TO 6
2061     Minefield(Ii,Jj)=0
2064   NEXT Jj
2067 NEXT Ii
2070 RETURN
2073  !
2076  !-----------------------------------------------------------------------
```

Table 6-14. Ground combat code (continued).

```
2079  !
2082 L5:  !
2085 INPUT "ENTER LINE 5:",Vis,Cloud_ht,Irh
2088 CALL Ck_var("VISIBILITY","TO",Vis,1,4)
2091 CALL Ck_var("CLOUD HT(m)","THRU",Cloud_ht,0,999999999)
2094 CALL Ck_var("REL HUMID","OR",Irh,1,2)
2097 Vis_bound$=" "
2100 SELECT Vis
2103 CASE 1
2106 !  Visib=7
2109   Visibility=5
2112   Vis_bound$=">"
2115 CASE 2
2118 !  Visib=5
2121   Visibility=5
2124 CASE 3
2127 !  Visib=2
2130   Visibility=2
2133 CASE 4
2136 !  Visib=1
2139   Visibility=1
2142 CASE ELSE
2145   PRINT " NO VISIBILITY "
2148   STOP
2151 END SELECT
2154 IF Vis=3 OR Vis=4 THEN   !SUPPRESS ALL HAND HELD ADA
2157   Bhnd_sup=1
2160   Rhnd_sup=1
2163 END IF
2166 RETURN
2169  !
2172  !------------------------------------------------------------------
2175  !
2178 L6:  !
2181 INPUT "ENTER LINE 6:",Ialb
2184 CALL Ck_var("ATTACKER SPECIAL TASK","TO",Ialb,0,4)
2187 Balb=1
2190 Ralb=1
2193 SELECT Atk_def
2196 CASE 0    !RED ATTACKER
2199   Balb=Ialb+1
2202 CASE 1    !BLUE ATTACKER
2205   Ralb=Ialb+1
2208 END SELECT
2211  !
2214 RETURN
2217  !
2220  !------------------------------------------------------------------
2223  !
2226 L7:  !
2229 INPUT "ENTER LINE 7:",B_msn(1),B_terr,B_rg_break,B_pct_fwd,B_mopp,T_lengt
1),T_width(1),B_break_t(1),B_cas_break
```

Table 6-14. Ground combat code (continued).

```
2232 CALL Ck_var("BLUE MISSION","TO",B_msn(1),0,9)
2235 CALL Ck_var("BLUE TERRAIN","TO",B_terr,1,4)
2238 CALL Ck_var("BL RG BRK/PT","THRU",B_rg_break,0,Init_rg-.001)
2241 CALL Ck_var("BLUE ADV. GUARD","THRU",B_pct_fwd,0,1)
2244 CALL Ck_var("BLUE MOPP/FATIGUE","THRU",B_mopp,0,100)
2247 !CALL Ck_var("BLUE SECTOR LENGTH","THRU",T_length(1),0,25)
2250 !CALL Ck_var("BLUE SECTOR WIDTH","THRU",T_width(1),0,25)
2253 CALL Ck_var("BLUE W/DRAW TIME","THRU",B_break_t(1),0,30)
2256 CALL Ck_var("BLUE CAS BRK/PT","THRU",B_cas_break,0,1)
2259 !
2262 !T_length(1)=T_length(1)*1000
2265 !T_width(1)=T_width(1)*1000
2268 B_msn(1)=B_msn(1)+1
2271 !
2274 RETURN
2277 !
2280 !------------------------------------------------------------------------
2283 !
2286 L8: !
2289 INPUT "ENTER LINE 8:",R_msn(1),R_terr,R_rg_break,R_pct_fwd,R_mopp,T_lengt
2),T_width(2),R_break_t(1),R_cas_break
2292 CALL Ck_var("RED MISSION","TO",R_msn(1),0,9)
2295 CALL Ck_var("RED TERRAIN","TO",R_terr,1,4)
2298 CALL Ck_var("RD RG BRK/PT","THRU",R_rg_break,0,Init_rg-.001)
2301 CALL Ck_var("RED ADV. GUARD","THRU",R_pct_fwd,0,1)
2304 CALL Ck_var("RED MOPP/FATIGUE","THRU",R_mopp,0,100)
2307 !CALL Ck_var("RED SECTOR LENGTH","THRU",T_length(2),0,25)
2310 !CALL Ck_var("RED SECTOR WIDTH","THRU",T_width(2),0,25)
2313 CALL Ck_var("RED W/DRAW TIME","THRU",R_break_t(1),0,30)
2316 CALL Ck_var("RED CAS BRK/PT","THRU",R_cas_break,0,1)
2319 !
2322 !T_width(2)=T_width(2)*1000
2325 !T_length(2)=T_length(2)*1000
2328 R_msn(1)=R_msn(1)+1
2331 !
2334 RETURN
2337 !
2340 !------------------------------------------------------------------------
2343 !
2346 L9: !
2349 PRINT TABXY(1,14),"** NOTE: Helicopter standoff ranges input on lines 9 &
0 are ranges from"
2352 PRINT TABXY(1,15),"** attack helicopters to primary mission targets."
2355 INPUT "ENTER LINE 9:",B_helo(1,1),B_helo(2,1),B_helo(3,1),B_helo(1,3),B_h
o(2,3),B_helo_atkprof(*),B_helo_delay,B_helo_rg_delay,B_helo_msn(*),B_atk_rg(*
2358 IF B_helo(1,1)>0 AND B_helo(1,3)=0 THEN
2361    PRINT "** ERROR: CANNOT HAVE AH-64'S WITHOUT CELLS; RE-ENTER LINE 9"
2364    GOTO L9
2367 END IF
2370 IF B_helo(1,1)>0 AND B_helo(3,1)>0 AND B_helo_msn(1)=2 THEN
2373    !SCOUTS ARE LASING FOR AH1'S AND ON AN AIR MISSION?  CAN'T DO THAT
2376    PRINT "** ERROR: SCOUTS CANNOT LASE ON AN AIR TO AIR MISSION. WILL IGN
```

Table 6-14. Ground combat code (continued).

```
 SCOUTS"
2379    B_helo(3,1)=0    !ZERO OUT SCOUTS
2382 END IF
2385! IF B_helo(1,1)>0 THEN B_helo_atkprof(1)=1
2388 CALL Ck_var("# OF   LCH","THRU",B_helo(1,1),0,99)
2391 CALL Ck_var("# OF AH1S","THRU",B_helo(2,1),0,99)
2394 CALL Ck_var("# OF SCOUTS","THRU",B_helo(3,1),0,99)
2397 CALL Ck_var("  LCH CELLS","TO",B_helo(1,3),0,4)
2400 CALL Ck_var("AH1S CELLS","TO",B_helo(2,3),0,4)
2403 CALL Ck_var("BLUE ATK PROF","TO",B_helo_atkprof(1),0,7)
2406 CALL Ck_var("BLUE ATK PROF","TO",B_helo_atkprof(2),0,7)
2409 CALL Ck_var("BLUE TIME DELAY","THRU",B_helo_delay,0,530)
2412 CALL Ck_var("BLUE ATK RG","THRU",B_helo_rg_delay,0,Init_rg)
2415 CALL Ck_var("BLUE HELO MSN","TO",B_helo_msn(1),0,3)
2418 CALL Ck_var("BLUE HELO MSN","TO",B_helo_msn(2),0,3)
2421 CALL Ck_var("BLUE STANDOFF RG","THRU",B_atk_rg(1),0,Init_rg)
2424 CALL Ck_var("BLUE STANDOFF RG","THRU",B_atk_rg(2),0,Init_rg)
2427 IF B_helo(3,1)>0 THEN
2430    B_helo(3,3)=B_helo(1,3)
2433 END IF
2436 RETURN
2439  !
2442  !----------------------------------------------------------------
2445  !
2448 L10:  !
2451 INPUT "ENTER LINE 10:",R_helo(1,1),R_helo(2,1),R_helo(3,1),R_helo(1,3),R_t
lo(2,3),R_helo_atkprof(*),R_helo_delay,R_helo_rg_delay,R_helo_msn(*),R_atk_rg(
2454 IF R_helo(1,1)>0 AND R_helo(1,3)=0 THEN
2457    PRINT "** ERROR: MUST SPECIFY # OF CELLS: RE-ENTER LINE 10"
2460    GOTO L10
2463 END IF
2466 IF R_helo(1,1)>0 AND R_helo(3,1)>0 AND R_helo_msn(1)=2 THEN
2469    !SCOUTS ARE LASING FOR AH1'S AND ON AN AIR MISSION?  CAN'T DO THAT
2472    PRINT "** ERROR: SCOUTS CANNOT LASE ON AN AIR TO AIR MISSION. WILL IGNOR
 SCOUTS"
2475    R_helo(3,1)=0    !ZERO OUT SCOUTS
2478 END IF
2481! IF R_helo(2,1)>0 THEN R_helo_atkprof(2)=3
2484 CALL Ck_var("# OF HIND","THRU",R_helo(1,1),0,99)
2487 CALL Ck_var("HIND CELLS","TO",R_helo(1,3),0,4)
2490 CALL Ck_var("# OF AH2S","THRU",R_helo(2,1),0,99)
2493 CALL Ck_var("AH2 CELLS","TO",R_helo(2,3),0,4)
2496 CALL Ck_var("RED ATK PROF","TO",R_helo_atkprof(1),0,7)
2499 CALL Ck_var("RED ATK PROF","TO",R_helo_atkprof(2),0,7)
2502 CALL Ck_var("RED TIME DELAY","THRU",R_helo_delay,0,530)
2505 CALL Ck_var("RED ATK RG","THRU",R_helo_rg_delay,0,Init_rg)
2508 CALL Ck_var("RED HELO MSN","TO",R_helo_msn(1),0,3)
2511 CALL Ck_var("RED HELO MSN","TO",R_helo_msn(2),0,3)
2514 CALL Ck_var("RED STANDOFF RG","THRU",R_atk_rg(1),0,Init_rg)
2517 CALL Ck_var("RED STANDOFF RG","THRU",R_atk_rg(2),0,Init_rg)
2520 IF R_helo(3,1)>0 THEN R_helo(3,3)=R_helo(1,3)
2523 PRINT TABXY(1,14),"
```

Table 6-14. Ground combat code (continued).

```
2526 PRINT TABXY(1,15),"                                                    "
2529 RETURN
2532  !
2535  !----------------------------------------------------------------------
2538  !
2541 L11:  !
2544 INPUT "ENTER LINE 11:",Bif_msn(*),B_prep_time,No_gamp,Perc_gamp,No_clgp,P
c_clgp,Clgp_rpv
2547 Total_pct=0
2550 FOR Tot_pct=1 TO 5        ! SMOKE WILL COME OUT OF CLOSE SUPPORT
2553   Total_pct=Bif_msn(Tot_pct)+Total_pct
2556 NEXT Tot_pct
2559 Total_pct=Total_pct+Perc_gamp+Perc_clgp
2562 IF Total_pct=100 OR Total_pct=0 THEN
2565   GOTO End_tot_pct_ch1
2568 ELSE
2571   PRINTER IS 1
2574   PRINT
2577   PRINT "** ERROR: %'S DO NOT EQUAL 100 OR 0 : RE-ENTER LINE 11"
2580   GOTO L11
2583 END IF
2586 End_tot_pct_ch1: !
2589 CALL Ck_var("SMOKE % ","THRU",Bif_msn(6),0,100)
2592 !CALL Ck_var("TIME DELAY","THRU",B_prep_time,0,2400)
2595 CALL Ck_var("# GAMP","THRU",No_gamp,0,999)
2598 CALL Ck_var("# CLGP","THRU",No_clgp,0,999)
2601 CALL Ck_var("RVP","OR",Clgp_rpv,0,1)
2604 B_prep_time=B_prep_time+30
2607 B_minute=B_prep_time MOD 100
2610 SELECT B_minute
2613 CASE 16 TO 45
2616   B_minute=30
2619 CASE <16
2622   B_minute=0
2625 CASE >45
2628   B_minute=100
2631 END SELECT
2634 B_prep_time=INT(B_prep_time/100)*100+B_minute
2637 RETURN
2640  !
2643  !----------------------------------------------------------------------
2646  !
2649 L12:  !
2652 INPUT "ENTER LINE 12:",Rif_msn(*),R_prep_time
2655 Total_pct=0
2658 FOR Tot_pct=1 TO 5        ! SMOKE WILL COME OUT OF CLOSE SUPPORT
2661   Total_pct=Rif_msn(Tot_pct)+Total_pct
2664 NEXT Tot_pct
2667 IF Total_pct=100 OR Total_pct=0 THEN
2670   GOTO End_tot_pct_ch2
2673 ELSE
```

Table 6-14. Ground combat code (continued).

```
2676   PRINTER IS 1
2679   PRINT
2682   PRINT "** ERROR: %'S DO NOT EQUAL 100 OR 0 ; RE-ENTER LINE 12"
2685   GOTO L12
2688 END IF
2691 End_tot_pct_ch2: !
2694 CALL Ck_var("SMOKE % ","THRU",Rif_msn(6),0,100)
2697 !CALL Ck_var("TIME DELAY","THRU",R_prep_time,0,2400)
2700 R_prep_time=R_prep_time+30
2703 R_minute=R_prep_time MOD 100
2706 SELECT R_minute
2709 CASE 16 TO 45
2712   R_minute=30
2715 CASE <16
2718   R_minute=0
2721 CASE >45
2724   R_minute=100
2727 END SELECT
2730 R_prep_time=INT(R_prep_time/100)*100+R_minute
2733 RETURN
2736   !
2739   !----------------------------------------------------------------------
2742   !
2745 Lmines:  !
2748 IF No_minefields>0 THEN
2751   FOR I=Beg TO End
2754     INPUT "ENTER MINEFIELD DATA: ",Mf(*)
2757     FOR J=1 TO 4
2760       Minefield(I,J)=Mf(J)
2763     NEXT J
2766     Minefield(I,5)=100  !******* NO LONGER USED
2769     CALL Ck_var("RANGE","THRU",Minefield(I,1),0,Init_rg-.001)
2772     CALL Ck_var("WIDTH","THRU",Minefield(I,2),1,999)
2775     CALL Ck_var("SECTOR WIDTH","THRU",Minefield(I,3),1,999)
2778     CALL Ck_var("% ENTERED","THRU",Minefield(I,4),0,100)
2781   NEXT I
2784 END IF
2787 RETURN
2790   !
2793   !----------------------------------------------------------------------
2796   !
2799 L13:  !
2802 IF No_minefields>=1 THEN
2805   Beg=1
2808   End=1
2811   GOSUB Lmines
2814 END IF
2817 RETURN
2820   !
2823   !----------------------------------------------------------------------
2826   !
2829 L14:  !
```

Table 6-14. Ground combat code (continued).

```
2832 IF No_minefields=2 THEN
2835   Beg=2
2838   End=2
2841   GOSUB Lmines
2844 END IF
2847 RETURN
2850  !
2853  !----------------------------------------------------------------------
2856  !
2859 L15:  !
2862 IF No_minefields=3 THEN
2865   Beg=3
2868   End=3
2871   GOSUB Lmines
2874 END IF
2877 RETURN
2880 !
2883 !------------------------------------------------------------------------
2886 !
2889 Set_print: !
2892 PRINT USING "//////,21A,4Z,2X,25A,6D";"BATTLE WILL BEGIN at ";St_time;"wi
 an INITIAL RANGE of ";Init_rg
2895 IF End_time<2400 THEN
2898   PRINT USING "19A,4Z";"BATTLE WILL END at ";End_time
2901 ELSE
2904   PRINT USING "19A,4Z";"BATTLE WILL END at ";End_time-2400
2907 END IF
2910 PRINT USING "/,6X,32A,6D";"DIRECT FIRE will begin at range ";Df_rg
2913 PRINT USING "/,6X,14A,1A,2D,22A,6D,1A";"Visibility is ";Vis_bound$;Visibi
ty;"km  ; Cloud height is ";Cloud_ht;"m"
2916 PRINT USING "//"
2919 SELECT Atk_def
2922 CASE 0
2925   B_atk_def$="DEFENDER"
2928   R_atk_def$="ATTACKER"
2931 CASE 1
2934   B_atk_def$="ATTACKER"
2937   R_atk_def$="DEFENDER"
2940 END SELECT
2943 PRINT
2946 PRINT
2949 PRINT "              BLUE ";B_atk_def$;"              :              RED  ";R_a
tk_def$
2952 PRINT USING "11X,13A,15X,1A,12X,13A";"-------------";":";"-------------"
2955 PRINT USING "39X,1A";":"
2958 SELECT B_msn(1)
2961 CASE 1
2964   B_msn$="MvContact"
2967 CASE 2
2970   B_msn$="Indr Fire"
2973 CASE 3
2976   B_msn$=" Movement"
```

Table 6-14. Ground combat code (continued).

```
2979 CASE 4
2982   B_msn$=" Frnt Atk"
2985 CASE 5
2988   B_msn$="  Env Atk"
2991 CASE 6
2994   B_msn$="    Delay"
2997 CASE 7
3000   B_msn$=" Hast Def"
3003 CASE 8
3006   B_msn$=" Prep Def"
3009 CASE 9
3012   B_msn$="Rear Area"
3015 CASE 10
3018   B_msn$="   Ambush"
3021 END SELECT
3024 SELECT R_msn(1)
3027 CASE 1
3030   R_msn$="MvContact"
3033 CASE 2
3036   R_msn$="Indr Fire"
3039 CASE 3
3042   R_msn$=" Movement"
3045 CASE 4
3048   R_msn$=" Frnt Atk"
3051 CASE 5
3054   R_msn$="  Env Atk"
3057 CASE 6
3060   R_msn$="    Delay"
3063 CASE 7
3066   R_msn$=" Hast Def"
3069 CASE 8
3072   R_msn$=" Prep Def"
3075 CASE 9
3078   R_msn$="Rear Area"
3081 CASE 10
3084   R_msn$="   Ambush"
3087 END SELECT
3090 PRINT USING "6X,8A,6X,9A,10X,1A,7X,8A,6X,9A";"Mission ";B_msn$;"|";"Missi
 ";R_msn$
3093 SELECT B_terr
3096 CASE 1
3099   B_terr$=" OPEN"
3102 CASE 2
3105   B_terr$=" ROLL"
3108 CASE 3
3111   B_terr$=" HILL"
3114 CASE 4
3117   B_terr$="MOUNT"
3120 END SELECT
3123 SELECT R_terr
3126 CASE 1
3129   R_terr$=" OPEN"
```

Table 6-14. Ground combat code (continued).

```
3132 CASE 2
3135   R_terr$=" ROLL"
3138 CASE 3
3141   R_terr$=" HILL"
3144 CASE 4
3147   R_terr$="MOUNT"
3150 END SELECT
3153 PRINT USING "6X.7A,11X.5A,10X.1A.7X.7A,11X,5A":"Terrain":B_terr$:"|":"Ter
in":R_terr$
3156 PRINT USING "6X.11A.6X .6D.10X,1A, 7X.11A.6X .6D":"Break Range":B_rg_brea
"|":"Break Range":R_rg_break
3159 PRINT USING "6X,16A.4X .3D.10X,1A, 7X,16A.4X .3D":"% Casualty Break":B_ca
break*100:"|":"% Casualty Break":R_cas_break*100
3162 PRINT USING "6X,9A.11X,3D.10X,1A, 7X,9A,11X.3D":"% Forward":B_pct_fwd*100
|":"% Forward":R_pct_fwd*100
3165 Set_prnt_fmt1:IMAGE 6X.10A.10X.3D,10X,1A. 7X.10A.10X.3D
3168    !
3171    !                   PRINT HELICOPTER INFORMATION
3174    !
3177 PRINT USING "6X.12A.21X.1A.7X,12A":"Helicopters:":"|":"Helicopters:"
3180 H_d_$="          MisslesMsl&guns    Guns Air-air Msl&AirMslGnAir Gun&Air"
 !ATTACK PROFILE
3183 H_msn$="         Air-grnd Air-air    SEAD"           !HELO MISSION
3186 Rg_msn$(1)="Rg to D.F."
3189 Rg_msn$(2)="Rg to Helo"
3192 Rg_msn$(3)="Rg to SEAD"
3195 PRINT USING "9X,11A,  X.8A.10X.1A.10X.11A,  X.8A":"Helo msn   ":H_msn$[B_h
o_msn(1)*8+1;8];"|":"Helo msn   ";H_msn$[R_helo_msn(1)*8+1;8]
3198 PRINT USING "21X        .8A.10X,1A,22X        .8A":H_msn$[B_helo_msn(2)*8+1
];"|";H_msn$[R_helo_msn(2)*8+1;8]
3201 PRINT USING "9X,11A,  X,8A,10X,1A.10X,11A,  X.8A":"Atk profile":H_d_$[B_he
_atkprof(1)*8+1;8];"|":"Atk profile":H_d_$[R_helo_atkprof(1)*8+1:8]
3204 PRINT USING "21X        .8A.10X.1A,22X        .8A":H_d_$[B_helo_atkprof(2)*
1;8]:"|";H_d_$[R_helo_atkprof(2)*8+1;8]
3207 FOR I=1 TO 2
3210   IF B_helo_msn(I)>0 AND R_helo_msn(I)>0 THEN
3213     PRINT USING "9X,10A,6X,4Z,10X,1A,10X,10A,6X,4Z":Rg_msn$(B_helo_msn(I)
B_atk_rg(I):"|":Rg_msn$(R_helo_msn(I)):R_atk_rg(I)
3216   ELSE
3219     IF B_helo_msn(I)>0 AND R_helo_msn(I)<=0 THEN
3222       PRINT USING "9X,10A,6X,4Z,10X,1A.10X.10A.6X.4Z":Rg_msn$(B_helo_msn(
):B_atk_rg(I):"|":"Stndoff Rg":R_atk_rg(I)
3225     ELSE
3228       IF B_helo_msn(I)<=0 AND R_helo_msn(I)>0 THEN
3231         PRINT USING "9X,10A,6X,4Z,10X,1A,10X,10A,6X,4Z":"Stndoff Rg":B_at
rg(I);"|":Rg_msn$(R_helo_msn(I)):R_atk_rg(I)
3234       ELSE
3237         PRINT USING "9X,10A,6X,4Z,10X,1A,10X,10A,6X,4Z":"Stndoff Rg":B_at
rg(I):"|":"Stndoff Rg":R_atk_rg(I)
3240       END IF
3243     END IF
3246   END IF
```

Table 6-14. Ground combat code (continued).

```
3249 NEXT I
3252 PRINT USING "9X,5A,14X,D,10X,1A,10X,5A,14X,D";"Cells";B_helo(1,3);"¦";"Ce
s";R_helo(1,3)
3255 PRINT USING "28X      ,D,10X,1A,29X       ,D";B_helo(2,3);"¦";R_helo(2,3)
3258 PRINT USING "9X,11A, 5X,4Z,10X,1A,10X,11A,5X,4Z";"Entry time";B_helo_dela
"¦";"Entry time";R_helo_delay
3261 PRINT USING "9X,11A, 3X,6D,10X,1A,10X,11A,3X,6D";"Entry range";B_helo_rg_
lay;"¦";"Entry range";R_helo_rg_delay
3264 PRINT USING "9X,6A,12X,2D,10X,1A,10X,6A,12X,2D";"#  LCH";B_helo(1,1);"¦";
 HIND";R_helo(1,1)
3267 PRINT USING "9X,6A,12X,2D,10X,1A,10X,6A,12X,2D";"# AH1S";B_helo(2,1);"¦";
  HIP";R_helo(2,1)
3270 PRINT USING "9X,6A,12X,2D,10X,1A,10X,6A,12X,2D";"#  SCT";B_helo(3,1);"¦";
  SCT";R_helo(3,1)                 !END OF CHANGES   DWS
3273 !PRINT USING "9X,6A,12X,2D,10X,1A";"# AHIP";B_helo(3,1);"¦"   !ROB
3276    !
3279 PRINT USING "39X,1A";"¦"
3282 IF B_atk_def$="DEFENDER" THEN
3285   PRINT USING "6X,12A,10X,D,10X,1A";"# Minefields";No_minefields;"¦"
3288   IF No_minefields>0 THEN
3291     PRINT USING " 9X,6A,8X,6D,10X,1A";"Range:";Minefield(1,1);"¦"
3294     IF No_minefields>1 THEN
3297       FOR Minflds=2 TO No_minefields
3300         PRINT USING "23X,6D,10X,1A";Minefield(Minflds,1);"¦"
3303       NEXT Minflds
3306     END IF
3309   END IF
3312 ELSE
3315   PRINT USING "39X,1A, 7X,12A,10X,D";"¦";"# Minefields";No_minefields
3318   IF No_minefields>0 THEN
3321     PRINT USING "39X,1A,10X,6A,8X,6D";"¦";"Range:";Minefield(1,1)
3324     IF No_minefields>1 THEN
3327       FOR Minflds=2 TO No_minefields
3330         PRINT USING "39X,1A,24X,6D";"¦";Minefield(Minflds,1)
3333       NEXT Minflds
3336     END IF
3339   END IF
3342 END IF
3345 PRINT " "
3348 PRINT " "
3351 FOR I=1 TO 12     !TEMP EXPERIMENT TO PRINT NAMES ROB
3354   M$=" "
3357   N$=" "
3360   IF I>No_b_unit THEN 3372
3363   ENTER @Fname,B_unit_no(I);M$
3366   ENTER @Unitpath,B_unit_no(I);N(*)
3369   Beff=N(79)
3372   IF I>No_r_unit THEN 3384
3375   ENTER @Rname,R_unit_no(I);N$
3378   ENTER @Unitpath,R_unit_no(I);N(*)
3381   Reff=N(79)
3384   IF I>No_b_unit AND I>No_r_unit THEN 3414
```

Table 6-14. Ground combat code (continued).

```
3387    IF I>No_b_unit THEN
3390      PRINT USING "47X,3D,2X,16A,2X,3D":R_unit_no(I):N$:R_unit_pct(I)*Reff
3393      GOTO 3414
3396    END IF
3399    IF I>No_r_unit THEN
3402      PRINT USING "6X,3D,2X,16A,2X,3D":B_unit_no(I):M$:R_unit_pct(I)*Beff
3405      GOTO 3414
3408    END IF
3411    PRINT USING "6X,3D,2X,16A,2X,3D,15X,3D,2X,16A,2X,3D":B_unit_no(I);M$;B_
it_pct(I)*Beff;R_unit_no(I);N$;R_unit_pct(I)*Reff
3414 NEXT I
3417 RETURN
3420   !
3423   !------------------------------------------------------------------
3426   !
3429 Set_conditions:    !     SETUP BATTLEFIELD CONDITIONS
3432   !
3435   ! SET BATTLE START TIME TO NEAREST 30 MINUTE BLOCK
3438 Minute=St_time MOD 100
3441 End_minute=End_time MOD 100
3444 SELECT Minute
3447 CASE 16 TO 45
3450   Minute=30
3453 CASE <16
3456   Minute=0
3459 CASE >45
3462   Minute=100
3465 END SELECT
3468 St_time=INT(St_time/100)*100+Minute
3471 End_time=INT(End_time/100)*100+End_minute
3474 IF St_time=2400 THEN St_time=0
3477   !
3480   !SET DAY/NIGHT STATUS
3483 IF St_time<600 OR St_time>2100 THEN
3486   Day_nite=1                                 !    NIGHT=1
3489 ELSE
3492   Day_nite=0                                 !    DAY=0
3495 END IF
3498   !
3501   !   SET FORCE EFFECTIVENESS
3504 FOR I=1 TO 70
3507   Init_b_eff=Init_b_eff+Sys_tot(1,I)*Sys_eff(1,I)
3510   Init_r_eff=Init_r_eff+Sys_tot(3,I)*Sys_eff(2,I)
3513 NEXT I
3516   !
3519   ! SET CURRENT FORCE EFFECTIVENESS
3522 B_cbt_eff=Init_b_eff
3525 R_cbt_eff=Init_r_eff
3528   !
3531   ! SET UP FORCE MATRIX
3534 FOR I=1 TO 70
3537   Sys_tot(2,I)=Sys_tot(1,I)
```

Table 6-14. Ground combat code (continued).

```
3540   Sys_tot(4,I)=Sys_tot(3,I)
3543 NEXT I
3546  !
3549  ! SET BATTLE TIME
3552 Btl_time=St_time
3555  !
3558 Max_btl_time=End_time
3561  ! SET BATTLE TERMINATION TIME
3564 !SELECT Btl_time
3567 !CASE 0 TO 599
3570   !Max_btl_time=600
3573 !CASE 600 TO 1199
3576   !Max_btl_time=1200
3579 !CASE 1200 TO 1799
3582   !Max_btl_time=1800
3585 !CASE 1800 TO 2400
3588   !Max_btl_time=2400
3591 !END SELECT
3594  !
3597  ! SET HOUR COUNTER
3600 St_hour=INT(Max_btl_time/100+.1)-6
3603 Battle_hour=INT(St_time/100+.1)
3606 Battle_min=St_time MOD 100
3609 Hh=(Battle_hour-St_hour)*2
3612 IF Battle_min>.1 THEN Hh=Hh+1
3615  !
3618  ! SET BATTLE RANGE
3621 Btl_rg=Init_rg
3624  !
3627  ! SET BATTLE PHASE
3630 IF Btl_rg>Df_rg THEN
3633   Btl_phase=1
3636 ELSE
3639   Btl_phase=2
3642 END IF
3645  !
3648  !SET FIRST BAND FOR DIRECT FIRE
3651 First_bnd=INT(Df_rg/500+.5)
3654 IF First_bnd>6 THEN First_bnd=6
3657 IF First_bnd<=0 THEN First_bnd=1
3660  !
3663  !SET UP HELO ATTACK PROFILE, MISSION & STANDOFF RANGE ARRAYS
3666 FOR I=1 TO 3
3669   J=I
3672   IF I=3 THEN
3675     IF B_helo(3,1)>0 THEN      !SCOUTS DO EXIST
3678       J=1                   !SET SCT'S ATK PROF. MSN AND RANGE FROM HELO 1'S
3681     ELSE                    !ZERO OUT SCT'S ARRAYS
3684       Atk_prof(1,3)=0
3687       Helo_mis(1,3)=0
3690       Stnd_off_rg(1,3)=0
3693       GOTO Rd_sct
```

Table 6-14. Ground combat code (continued).

```
3696       END IF
3699     END IF
3702     Atk_prof(1,I)=B_helo_atkprof(J)
3705     Helo_mis(1,I)=B_helo_msn(J)
3708     Stnd_off_rg(1,I)=B_atk_rg(J)
3711 Rd_sct:IF I=3 THEN
3714       IF R_helo(3,1)>0 THEN       !SCOUTS DO EXIST
3717         J=1                  !SET SCT'S ATK PROF. MSN AND RANGE FROM HELO 1'S
3720       ELSE                   !ZERO OUT SCT'S ARRAYS
3723         Atk_prof(2,3)=0
3726         Helo_mis(2,3)=0
3729         Stnd_off_rg(2,3)=0
3732         GOTO Nxt_setup
3735       END IF
3738     END IF
3741     Atk_prof(2,I)=R_helo_atkprof(J)
3744     Helo_mis(2,I)=R_helo_msn(J)
3747     Stnd_off_rg(2,I)=R_atk_rg(J)
3750 Nxt_setup:!
3753 NEXT I
3756   !
3759   ! SET INDIRECT FIRE MISSION TONNAGES FOR 6-HOUR BATTLE
3762   !
3765   !SET FOR GAMP&CLGP
3768 Gamp_avail=No_gamp*.11
3771 Clgp_avail=No_clgp*.11
3774   !
3777   ! SET FOR ARTILLERY
3780 FOR J=1 TO 7
3783     FOR I=1 TO 5
3786       Bif_msn_tons(J,I)=Bif_ammo(J)*Bif_msn(I)/100
3789       Rif_msn_tons(J,I)=Rif_ammo(J)*Rif_msn(I)/100
3792     NEXT I
3795     B_dsarty_avail(J)=Bif_msn_tons(J,2)
3798     R_dsarty_avail(J)=Rif_msn_tons(J,2)
3801 NEXT J
3804   !
3807   ! SET FOR MLRS
3810 FOR I=1 TO 5
3813     IF I=2 THEN Next_mlrs          ! MLRS NOT A CS WEAPON
3816     IF Bif_msn(2)>=100 THEN Rif_mt
3819     FOR J=12 TO 15
3822       Bif_msn_tons(J,I)=Bif_ammo(J)*Bif_msn(I)/(100-Bif_msn(2))
3825     NEXT J
3828 Rif_mt:IF Rif_msn(2)>=100 THEN Next_mlrs
3831     FOR J=12 TO 15
3834       Rif_msn_tons(J,I)=Rif_ammo(J)*Rif_msn(I)/(100-Rif_msn(2))
3837     NEXT J
3840 Next_mlrs:NEXT I
3843   !
3846   ! SET FOR MORTARS
3849 Int_bmort=Bif_msn(1)+Bif_msn(2)+Bif_msn(3)    ! NO MORTARS IN CFIRE/INT
```

Table 6-14. Ground combat code (continued).

```
3852 Int_rmort=Rif_msn(1)+Rif_msn(2)+Rif_msn(3)
3855 FOR I=1 TO 5
3858   IF I=4 OR I=5 THEN Next_mort
3861   IF Int_bmort=0 THEN Next_mort_r
3864   FOR J=8 TO 11
3867     Bif_msn_tons(J,I)=Bif_ammo(J)*Bif_msn(I)/Int_bmort
3870   NEXT J
3873 Next_mort_r: !
3876   IF Int_rmort=0 THEN Next_mort
3879   FOR J=8 TO 11
3882     Rif_msn_tons(J,I)=Rif_ammo(J)*Rif_msn(I)/Int_rmort
3885   NEXT J
3888 Next_mort:NEXT I
3891 FOR J=8 TO 11
3894   B_dsmort_avail(J-7)=Bif_msn_tons(J,2)
3897   R_dsmort_avail(J-7)=Rif_msn_tons(J,2)
3900 NEXT J
3903  !
3906  ! ZERO SEAD AMMO.  NOT EXPLICITLY USED
3909 !FOR I=1 TO 15                    !ROB
3912   !Rif_fired(I,3)=0
3915   !Bif_fired(I,3)=0
3918 !NEXT I
3921  !
3924 Rif_ammo_sv=0
3927 Bif_ammo_sv=0
3930 FOR I=1 TO 15
3933   FOR J=1 TO 5
3936     Rif_ammo_sv=Rif_ammo_sv+Rif_msn_tons(I,J)
3939     Bif_ammo_sv=Bif_ammo_sv+Bif_msn_tons(I,J)
3942   NEXT J
3945 NEXT I
3948  !
3951  !SET ARTY MLRS AND MORT CAP FOR MISSION
3954  ! MAX FIRE RATES FOR THIS MISSION
3957 FOR I=1 TO 15
3960   Arty_30min_wt(1,I)=Barty_30min(B_msn(1),I)
3963   Arty_30min_wt(2,I)=Rarty_30min(R_msn(1),I)
3966 NEXT I
3969  !
3972  ! SET DIRECT SUPPORT ARTILLERY/MORTAR PARAMETERS
3975 B_dsmort_brkrg=B_rg_break-100
3978 R_dsmort_brkrg=R_rg_break-100
3981 B_dsarty_brkrg=B_rg_break            ! END RANGE FOR CS ARTY SUPPORT
3984 R_dsarty_brkrg=R_rg_break
3987 B_dsarty_start=Ds_start(R_terr)      ! START RANGE FOR CS ARTY SUPPORT
3990 R_dsarty_start=Ds_start(B_terr)
3993  !
3996 IF Ds_start(R_terr)<5700 THEN
3999   B_dsmort_start=Ds_start(R_terr)
4002 ELSE
4005   B_dsmort_start=5700
```

Table 6-14. Ground combat code (continued).

```
4008 END IF
4011  !
4014 IF Ds_start(B_terr)<5700 THEN
4017   R_dsmort_start=Ds_start(B_terr)
4020 ELSE
4023   R_dsmort_start=5700
4026 END IF
4029 B_ds_shift=0                         ! STATUS OF RE-DEFINE OF FIRING LEVEL
4032 R_ds_shift=0                         !       0=NOT RE-DEFINED
4035 B_mo_shift=0                         !       1=RE-DEFINED
4038 R_mo_shift=0
4041 R_ds_conc_pt=R_dsarty_brkrg+1500
4044 B_ds_conc_pt=B_dsarty_brkrg+1500       ! X-PT AT KNEE OF CS CURVE
4047 B_mo_conc_pt=B_dsarty_brkrg+1500
4050 R_mo_conc_pt=R_dsarty_brkrg+1500
4053 B_ds_conc_level=.5
4056 R_ds_conc_level=.7                     ! Y-PT AT KNEE OF CS CURVE
4059 B_mo_conc_level=.5
4062 R_mo_conc_level=.7
4065 B_p30_artyrg=B_dsarty_start
4068 R_p30_artyrg=R_dsarty_start     ! BATTLE RANGE AT END OF PREVIOUS 30 MIN F
4071 B_p30_mortrg=B_dsmort_start
4074 R_p30_mortrg=R_dsmort_start
4077  !
4080 IF Bt_time<600 OR Bt_time>2100 THEN
4083   Day_nite=1      ! CHECK THIS
4086 ELSE
4089   Day_nite=0
4092 END IF
4095  ! READ ADVANCE RATE DATA
4098 IF Atk_def=0 THEN
4101   File=8+4*Day_nite+R_terr
4104 ELSE
4107   File=4*Day_nite+B_terr
4110 END IF
4113 ENTER @Advanpath,File;Advance_rate(*)
4116  !
4119 RETURN
4122  !
4125  !-----------------------------------------------------------------
4128  !
4131 Print_sys_out:  !  THIS SBR PRINTS OUT SPECIFIED SYSTEMS
4134  !
4137 FOR I=7 TO 70 STEP 7
4140   PRINT USING Fmt1;I-6,":",S(I-6),I-5,":",S(I-5),I-4,":",S(I-4),I-3,":",S
-3),I-2,":",S(I-2),I-1,":",S(I-1),I,":",S(I)
4143 NEXT I
4146 Fmt1:IMAGE 7(2D,1A,4D.1D,2X)
4149 RETURN
4152  !
4155  !-----------------------------------------------------------------
4158  !
```

Table 6-14. Ground combat code (continued).

```
4161 Control_battle:  !  THIS SBR CONTROLS THE GROUND BATTLE
4164  !
4167  !!PRINT OUT INITIAL SECTOR CONDITIONS
4170 GOSUB Print_init_res
4173  !
4176  !  READ DATA FILES TO BE USED IN 30 MINUTE BATTLE
4179  !
4182 GOSUB Read_files
4185  !
4188  ! START ATTRITION ASSESSMENTS
4191 Start_30min_btl: !  START ATTRITION ASSESSMENTS FOR A 30 MINUTE TIME PERI
4194  !
4197  ! COMPUTE COMBAT EFFECTIVENESS
4200 GOSUB Tally_cbt_eff
4203  !
4206  ! CHECK BREAKPOINT CRITERIA
4209 GOSUB Check_brk_pt
4212  !
4215  ! SET PHASE FOR NEXT 30 MIN BATTLE
4218 IF Break_point=1 OR Break_point=2 THEN Btl_phase=3
4221 IF Break_point=3 THEN GOTO End_battle
4224  !
4227 Continue_btl:  ! UPDATE BATTLE TIME.
4230  !                TIME IS SET TO END OF THE 30 MIN PERIOD.
4233  ! SETTING MISSIONS FOR RED AND BLUE
4236 B_ms=1
4239 R_ms=1
4242 !
4245 B_time_print=Btl_time
4248 !
4251 Btl_time=Btl_time+30
4254 Int_minute=Btl_time MOD 100
4257 IF Int_minute=60 THEN
4260   Btl_time=Btl_time+40        ! SET TIME TO NEXT WHOLE HOUR AT 60 MINS
4263 END IF
4266 IF Btl_time<600 OR Btl_time>2100 THEN
4269   Day_nite=1
4272 ELSE
4275   Day_nite=0
4278 END IF
4281 Time_seg=Time_seg+1
4284  !
4287 PRINT USING "@,#"
4290 PRINT USING "1X,45A";"---------------------------------------------"
4293 IF Btl_time<2430 THEN
4296   PRINT USING "33A,4Z,4A,4Z,2A";"! SIGNIFICANT BATTLE EVENTS FROM ";B_tim
print;" TO ";Btl_time;" :"
4299 ELSE
4302   PRINT USING "33A,4Z,4A,4Z,2A";": SIGNIFICANT BATTLE EVENTS FROM ";B_tim
print-2400;" TO ";Btl_time-2400;":"
4305 END IF
4308 PRINT USING "1X,45A";"---------------------------------------------"
```

Table 6-14. Ground combat code (continued).

```
4311  '
4314  ! SET ATTACK HELICOPTER ARRIVALS FOR RED/BLUE
4317 GOSUB Helo_arrive
4320  !
4323 PRINT USING "/"
4326 IF Vis=4 THEN
4329   PRINT "   HELICOPTERS NOT FLYING ON 1km DAY"
4332 ELSE
4335   PRINT "   BLUE HELICOPTERS ATTACKING RED FORCE"
4338   PRINT USING Fmt80;" LCH";Bah1
4341   PRINT USING Fmt80;"AH1S";Bah2
4344   PRINT USING Fmt80;" SCT";Bsct
4347  ! PRINT USING Fmt80;"AHIP";Bsct
4350   PRINT
4353   PRINT "   RED HELICOPTERS ATTACKING BLUE FORCE"
4356   PRINT USING Fmt80;"HIND";Rah1
4359   PRINT USING Fmt80;" HIP";Rah2
4362   PRINT USING Fmt80;" SCT";Rsct
4365 END IF
4368 Fmt80:IMAGE 6X,4A,2X,3D.D
4371  ! SET INCOMING INDIRECT FIRE FOR RED/BLUE
4374 GOSUB Arty_arrive
4377  !
4380  !
4383 FOR I=1 TO 15
4386   Saty(I)=0
4389   Tot_volley(I)=0
4392   FOR J=1 TO 5
4395     Volley(I,J)=Bif_fired(I,J)/A_wt(1,I)
4398     Saty(I)=Saty(I)+Bif_msn_tons(I,J)
4401   NEXT J
4404   Tot_volley(I)=Tot_arty(1,I)/A_wt(1,I)
4407 NEXT I
4410 B_or_r_$="BLUE"
4413 B_or_r=1
4416 Label$="ARTY ARTY ARTY ARTY ARTY ARTY ARTY MORT MORT MORT MORT MLRS MLRS
RS MLRS"
4419 GOSUB Print_volleys
4422 PRINT
4425 PRINT "   BLUE PGM ROUNDS:       CLGP                  GAMP"
4428 PRINT USING "6X,9A,6X,6D.D,11X,6D.D";"FIRED     ";Clgp_msns/.11;Gamp_msns/
mo_wt(1,28)     !?? STILL USE 28
4431 PRINT USING "6X,9A,6X,6D.D,11X,6D.D";"AVAILABLE";Clgp_avail/.11;Gamp_avai
Ammo_wt(1,28)  !??STILL USE 28
4434  !
4437 IF B_ds_shift=1 AND Atk_def=0 THEN PRINT USING "/,70A";"   BLUE FINAL PRO
CTIVE ARTILLERY BEING APPLIED"
4440 IF B_ds_shift=1 AND Atk_def=1 THEN PRINT USING "/,70A";"   BLUE FINAL ART
LERY BARRAGE ON OBJECTIVE BEING APPLIED"
4443 IF B_mo_shift=1 AND Atk_def=0 THEN PRINT USING "/,67A";"   BLUE FINAL PRO
CTIVE MORTAR BEING APPLIED"
4446 IF B_mo_shift=1 AND Atk_def=1 THEN PRINT USING "/,67A";"   BLUE FINAL MOR
```

Table 6-14. Ground combat code (continued).

```
R BARRAGE ON OBJECTIVE BEING APPLIED"
4449 FOR I=1 TO 15
4452   Saty(I)=0
4455   Tot_volley(I)=0
4458   FOR J=1 TO 5
4461     Volley(I,J)=Rif_fired(I,J)/A_wt(2,I)
4464     Saty(I)=Saty(I)+Rif_msn_tons I,J)
4467   NEXT J
4470   Tot_volley(I)=Tot_arty(2,I)/A_wt(2,I)
4473 NEXT I
4476 B_or_r_$="RED"
4479 B_or_r=2
4482 Label$="ARTY ARTY ARTY ARTY ARTY ARTY ARTY MORT MORT MORT MORT MRL  MRL
L  MRL  "
4485 GOSUB Print_volleys
4488 IF R_ds_shift=1 AND Atk_def=1 THEN PRINT USING "/,70A";"   RED  FINAL PRC
CTIVE ARTILLERY BEING APPLIED"
4491 IF R_ds_shift=1 AND Atk_def=0 THEN PRINT USING "/,70A";"   RED  FINAL ART
LERY BARRAGE ON OBJECTIVE BEING APPLIED"
4494 IF R_mo_shift=1 AND Atk_def=1 THEN PRINT USING "/,67A";"   RED  FINAL PRC
CTIVE MORTAR BEING APPLIED"
4497 IF R_mo_shift=1 AND Atk_def=0 THEN PRINT USING "/,67A";"   RED  FINAL MOF
R BARRAGE ON OBJECTIVE BEING APPLIED"
4500  ! CALCULATE AMOUNT OF ADVANCE BY ATTACKER
4503 GOSUB Calc_movement
4506  ! CHECK FOR MINEFIELDS
4509 GOSUB Mine_encounter
4512  !ADJUST MOVEMENT FOR MINES
4515 IF Mine_hit<>0 AND Amt_of_advance<>0 THEN
4518   !CALCULATE MOVEMENT TIME TO EDGE OF MINEFIELD
4521   Edge_time=(Btl_rg-Minefield(Mine_hit,1))/Amt_of_advance*30
4524   IF Edge_time+Mine_delay<30 THEN
4527     Amt_of_advance=(1-Mine_delay/30)*Amt_of_advance
4530     Mine_delay=0
4533   ELSE
4536     Amt_of_advance=Btl_rg-Minefield(Mine_hit,1)
4539     Mine_delay=Mine_delay-(30-Edge_time)
4542   END IF
4545 END IF
4548  !
4551  ! CHECK FOR DIRECT FIRE
4554 Check_dir_fire: !
4557 IF Btl_phase=3 THEN Calc_btl_rg
4560 IF Btl_rg-Amt_of_advance<=Df_rg THEN
4563   Btl_phase=2
4566   IF First_df=0 THEN
4569     Int_advance=Btl_rg-Df_rg
4572     Int_btl_rg=(Amt_of_advance-Int_advance)
4575     Cur_bnd=First_bnd
4578     SELECT Int_btl_rg
4581     CASE <250
4584       First_df=1
```

Table 6-14. Ground combat code (continued).

```
4587        Df_500_bds=1
4590        Amt_of_advance=Int_advance
4593      CASE 250 TO 750
4596        First_df=1
4599        Df_500_bds=1
4602        Amt_of_advance=Int_advance+500
4605      CASE >750
4608        First_df=1
4611        Df_500_bds=2
4614        Amt_of_advance=Int_advance+1000
4617      END SELECT
4620    ELSE
4623      Int_btl_rg=Amt_of_advance
4626      SELECT Int_btl_rg
4629      CASE <250
4632        Amt_of_advance=0
4635        Df_500_bds=1
4638      CASE 250 TO 750
4641        Amt_of_advance=500
4644        Df_500_bds=1
4647      CASE >750
4650        Amt_of_advance=1000
4653        Df_500_bds=2
4656      END SELECT
4659    END IF
4662 ELSE
4665    Btl_phase=1
4668 END IF
4671   !
4674 Calc_btl_rg: !
4677 IF Btl_phase=3 THEN
4680    SELECT Atk_def
4683    CASE 0                    ! RED ATK
4686      IF Break_point=2 THEN Amt_of_advance=0
4689    CASE 1                    ! BLUE ATK
4692      IF Break_point=1 THEN Amt_of_advance=0
4695    END SELECT
4698 END IF
4701                 !
4704 Btl_rg=MAX(Btl_rg-Amt_of_advance,0)
4707   !
4710 GOSUB Print_eff
4713   !
4716   !
4719 SELECT Btl_phase
4722 CASE 1
4725    IF Btl_time<2430 THEN
4728      PRINT USING Fmt11;"ATTACKER FORCES CLOSING",Btl_time," HRS","B/EFF: "
urr_b_pt,"R/EFF: ",Curr_r_pt
4731    ELSE
4734      PRINT USING Fmt11;"ATTACKER FORCES CLOSING",Btl_time-2400," HRS","B/F
: ",Curr_b_pt,"R/EFF: ",Curr_r_pt
```

Table 6-14. Ground combat code (continued).

```
4737   END IF
4740   PRINT USING "19X,15A,6D,5X,14A,6D";"INITIAL RANGE: ",Btl_rg+Amt_of_adva
e,"CURRENT RANGE: ",Btl_rg
4743   GOSUB Phase1_btl
4746   Flag_30min=1
4749 CASE 2
4752   IF Btl_time<2430 THEN
4755     PRINT USING Fmt11;"CLOSE DF BATTLE UNDERWAY",Btl_time," HRS","B/EFF:
Curr_b_pt,"R/EFF: ",Curr_r_pt
4758   ELSE
4761     PRINT USING Fmt11;"CLOSE DF BATTLE UNDERWAY",Btl_time-2400," HRS","B/
F: ",Curr_b_pt,"R/EFF: ",Curr_r_pt
4764   END IF
4767   PRINT USING "19X,15A,6D,5X,14A,6D";"INITIAL RANGE: ",Btl_rg+Amt_of_adva
e,"CURRENT RANGE: ",Btl_rg
4770 Fmt11:IMAGE /,3X,25A,2X,4Z,4A,4X,7A,1D.2D,3X,7A,1D.2D
4773   GOSUB Phase2_btl
4776   Flag_30min=1
4779 CASE 3
4782   GOSUB Phase3_btl
4785   Flag_30min=2
4788 END SELECT
4791   ! COMPUTE COMBAT EFFECTIVENESS
4794   !GOSUB Tally_cbt_eff
4797   !GOSUB Print_eff
4800   !PRINT "    ";"BEFF: ";INT((Curr_b_pt+.005)*100);"%","REFF: ";INT((Curr_r
t+.005)*100);"%"
4803 GOTO Start_30min_btl
4806   !
4809   !
4812 End_battle:RETURN
4815   !
4818   !--------------------------------------------------------------------
4821 Print_eff:   !
4824 R_eff_pt=0
4827 B_eff_pt=0
4830 FOR I=1 TO 70
4833   B_eff_pt=B_eff_pt+Sys_tot(2,I)*Sys_eff(1,I)
4836   R_eff_pt=R_eff_pt+Sys_tot(4,I)*Sys_eff(2,I)
4839 NEXT I
4842 IF Init_b_eff=0 THEN
4845   Curr_b_pt=0
4848 ELSE
4851   Curr_b_pt=B_eff_pt/Init_b_eff
4854 END IF
4857 IF Init_r_eff=0 THEN
4860   Curr_r_pt=0
4863 ELSE
4866   Curr_r_pt=R_eff_pt/Init_r_eff
4869 END IF
4872 RETURN
4875   !
```

Table 6-14. Ground combat code (continued).

```
4878  !-------------------------------------------------------------------
4881  !
4884 Print_volleys:  !
4887 PRINT
4890 PRINT
4893 PRINT USING Pv_f3;B_or_r_$;" ARTILLERY VOLLEYS FIRED ":"Tons":"Tons"
4896 Pv_f3:IMAGE 3X,4A,25A,29X,4A,8X,4A
4899 PRINT USING Pv_f1;"P/CP";" CS ";"SEAD";" CF ";" INT";"TOTAL":"CONSUMED":'
AILABLE"
4902 Pv_f1:IMAGE 9X,5(4A,3X),2X,5A,8X,8A,4X,9A
4905 FOR Wv=1 TO 15
4908   IF Saty(Wv)+Tot_arty(B_or_r,Wv)>0 THEN  !If tons consumed or avail not
 then print out line
4911     PRINT USING Pv_f2;Label$[Wv*5-4,Wv*5],Volley(Wv,1),Volley(Wv,2),Volle
Wv,3),Volley(Wv,4),Volley(Wv,5),Tot_volley(Wv),Tot_arty(B_or_r,Wv),Saty(Wv)
4914   END IF
4917 NEXT Wv
4920 Pv_f2:IMAGE 3X,5A,5(5D,2X),2X,6D,7X,6D.D,4X,8D.D
4923 RETURN
4926  !
4929  !-------------------------------------------------------------------
4932  !
4935 Print_init_res:  !  THIS SBR PRINTS OUT INITIAL UNIT STATUS
4938  !
4941 PRINT USING "@,#,13A,5X,11A,2D,6X,8A,3D";"GROUND COMBAT","GAME TURN: ",T
,"SECTOR: ",Sector
4944 GOSUB Set_print
4947 RETURN  ! TEMPORARY DISABLING OF THIS SUB-ROUTINE
4950 PRINT USING "//,11A,29X,10A";"BLUE UNITS:","RED UNITS:"
4953 FOR I=0 TO 9 STEP 3
4956   FOR J=1 TO 3
4959     PRINT USING Fmt2;B_unit_no(I+J),B_unit_pct(I+J)
4962   NEXT J
4965 Fmt2:IMAGE 3D,1X,3D,3X,#
4968   PRINT USING "7X,#"
4971   FOR J=1 TO 3
4974     PRINT USING Fmt3;R_unit_no(I+J),R_unit_pct(I+J)
4977   NEXT J
4980 Fmt3:IMAGE 3X,3D,1X,3D,#
4983   PRINT USING "/"
4986 NEXT I
4989 PRINT USING "////,13A";"BLUE SYSTEMS:"
4992 FOR I=1 TO 70
4995   S(I)=Sys_tot(2,I)
4998 NEXT I
5001 GOSUB Print_sys_out
5004 PRINT USING "//,12A";"RED SYSTEMS:"
5007 FOR I=1 TO 70
5010   S(I)=Sys_tot(4,I)
5013 NEXT I
5016 GOSUB Print_sys_out
5019 GOSUB Set_print
```

Table 6-14. Ground combat code (continued).

```
5022 PRINT USING "@,#,2X,26A,//";"* * * BATTLE HISTORY * * *"
5025 RETURN
5028  !
5031  !------------------------------------- ----------------------------
5034  !
5037 Print_fin_res:  ! THIS SBR PRINTS OUT FINAL BATTLE RESULTS
5040 PRINT
5043 PRINT
5046 PRINT
5049 PRINT " -------------------- "
5052 IF Btl_time<2430 THEN
5055   PRINT USING "17A,4Z,2A";"! BATTLE ENDS AT ";Btl_time;" !"
5058 ELSE
5061   PRINT USING "17A,4Z,2A";": BATTLE ENDS AT ";Btl_time-2400;":"
5064 END IF
5067 PRINT " -------------------- "
5070  !
5073 PRINT USING "@,#,29A";"RED KILLER--BLUE VICTIM TABLE"
5076 PRINT USING "//,6A,14X,48A";"VICTIM","<------------------ KILLER --------
----------->"
5079 PRINT USING Fmt30;" SYS","START","D/F","I/F","PGM","A/H","INF","MIN","END
5082 Fmt30:IMAGE 4A,5X,5A,7(6X,3A)
5085 FOR I=1 TO 70
5088   IF Sys_tot(1,I)<=0 THEN Skip_prt1
5091   PRINT USING Fmt31;B_veh$[(I-1)*5+1;5],Sys_tot(1,I),Kv_b(1,I),Kv_b(2,I),
_b(3,I),Kv_b(4,I),Kv_b(5,I),Kv_b(6,I),Sys_tot(2,I)
5094 Skip_prt1:NEXT I
5097 Fmt31:IMAGE 5A,8(3X,4D.1D)
5100 PRINT USING "////,29A";"BLUE KILLER--RED VICTIM TABLE"
5103 PRINT USING "//,6A,14X,48A";"VICTIM","<------------------ KILLER --------
----------->"
5106 PRINT USING Fmt30;" SYS","START","D/F","I/F","PGM","A/H","INF","MIN","END
5109 FOR I=1 TO 70
5112   IF Sys_tot(3,I)<=0 THEN Skip_prt2
5115   PRINT USING Fmt31;R_veh$[(I-1)*5+1;5],Sys_tot(3,I),Kv_r(1,I),Kv_r(2,I),
_r(3,I),Kv_r(4,I),Kv_r(5,I),Kv_r(6,I),Sys_tot(4,I)
5118 Skip_prt2:NEXT I
5121  !
5124  !!PRINT OUT HELICOPTER RESULTS (SORTIES,LOSSES,AMMO/FUEL EXPENDED)
5127  !
5130 PRINT USING "@,#,27A";"ATTACK HELICOPTER RESULTS:"
5133 PRINT USING Fmt32;"TYPE","#COMMITTED","#KILLED","#SORTIES"
5136 PRINT USING Fmt33;" LCH",B_helo(1,1),B_helo(1,2),B_helo(1,6)
5139 PRINT USING Fmt33;"AH1S",B_helo(2,1),B_helo(2,2),B_helo(2,6)
5142 PRINT USING Fmt33;"SCTS",B_helo(3,1),B_helo(3,2),B_helo(3,6)
5145 PRINT USING Fmt34;"HIND",R_helo(1,1),R_helo(1,2),R_helo(1,6)
5148 PRINT USING Fmt33;"HIP ",R_helo(2,1),R_helo(2,2),R_helo(2,6)
5151 PRINT USING Fmt33;"SCTS",R_helo(3,1),R_helo(3,2),R_helo(3,6)
5154 Fmt32:IMAGE ///,4A,3X,10A,4X,7A,(4X,8A),//
5157 Fmt33:IMAGE 4A,6X,3D.1D,2(7X,3D.1D)
5160 Fmt34:IMAGE //,4A,6X,3D.1D,2(7X,3D.1D)
5163 RETURN  !DISABLE LER PRINTOUT
```

Table 6-14. Ground combat code (continued).

```
5166 FOR J=1 TO 2
5169   Sys_dfl(J)=Sys_tot(J,5)
5172   FOR I=1 TO 3
5175     Sys_dfl(J)=Sys_dfl(J)+Sys_tot(J,I)
5178   NEXT I
5181   Sys_dfs(J)=Sys_tot(J,4)
5184   FOR I=7 TO 9
5187     Sys_dfs(J)=Sys_dfs(J)+Sys_tot(J,I)
5190   NEXT I
5193 NEXT J
5196 FOR J=3 TO 4
5199   Sys_dfl(J)=Sys_tot(J,6)+Sys_tot(J,7)+Sys_tot(J,9)
5202   FOR I=1 TO 3
5205     Sys_dfl(J)=Sys_dfl(J)+Sys_tot(J,I)
5208   NEXT I
5211   Sys_dfs(J)=Sys_tot(J,4)+Sys_tot(J,5)+Sys_tot(J,8)
5214 NEXT J
5217 FOR J=1 TO 4
5220   Sys_if(J)=Sys_tot(J,11)+Sys_tot(J,13)+Sys_tot(J,14)
5223   Sys_ad(J)=Sys_tot(J,12)+Sys_tot(J,15)+Sys_tot(J,16)+Sys_tot(J,17)
5226   Sys_sum(J)=Sys_dfl(J)+Sys_dfs(J)+Sys_if(J)+Sys_ad(J)
5229 NEXT J
5232 Sys_dfl(7)=Kv_b(4,1)+Kv_b(4,2)+Kv_b(4,3)+Kv_b(4,5)
5235 Sys_dfs(7)=Kv_b(4,4)+Kv_b(4,7)+Kv_b(4,8)+Kv_b(4,9)
5238 Sys_if(7)=Kv_b(4,11)+Kv_b(4,13)+Kv_b(4,14)
5241 Sys_ad(7)=Kv_b(4,12)+Kv_b(4,15)+Kv_b(4,16)+Kv_b(4,17)
5244 Sys_sum(7)=Sys_dfl(7)+Sys_dfs(7)+Sys_if(7)+Sys_ad(7)
5247 Sys_dfl(5)=Sys_dfl(1)-Sys_dfl(2)
5250 Sys_dfs(5)=Sys_dfs(1)-Sys_dfs(2)
5253 Sys_if(5)=Sys_if(1)-Sys_if(2)
5256 Sys_ad(5)=Sys_ad(1)-Sys_ad(2)
5259 Sys_sum(5)=Sys_sum(1)-Sys_sum(2)
5262 Sys_dfl(6)=Sys_dfl(3)-Sys_dfl(4)
5265 Sys_dfs(6)=Sys_dfs(3)-Sys_dfs(4)
5268 Sys_if(6)=Sys_if(3)-Sys_if(4)
5271 Sys_ad(6)=Sys_ad(3)-Sys_ad(4)
5274 Sys_sum(6)=Sys_sum(3)-Sys_sum(4)
5277 IF Sys_sum(1)=0 THEN
5280   Cfr_b=1
5283 ELSE
5286   IF Sys_sum(3)=0 THEN
5289     Cfr_b=1
5292   ELSE
5295     Cfr_b=Sys_sum(3)/Sys_sum(1)
5298   END IF
5301 END IF
5304 Cfr_r=1/Cfr_b
5307 Cfr_bwo=Cfr_b
5310 IF Sys_dfl(1)=0 THEN
5313   Cfr_bdfl=0
5316   Ler_bdfl=0
5319   Fer_bdfl=0
```

Table 6-14. Ground combat code (continued).

```
5322 ELSE
5325   Cfr_bdfl=Sys_dfl(3)/Sys_dfl(1)
5328   IF Sys_dfl(5)=0 THEN
5331     Ler_bdfl=0
5334   ELSE
5337     Ler_bdfl=Sys_dfl(6)/Sys_dfl(5)
5340   END IF
5343   IF Cfr_bdfl=0 THEN
5346     Fer_bdfl=0
5349   ELSE
5352     Fer_bdfl=Ler_bdfl/Cfr_bdfl
5355   END IF
5358 END IF
5361 IF Sys_dfs(1)=0 THEN
5364   Cfr_bdfs=0
5367   Ler_bdfs=0
5370   Fer_bdfs=0
5373 ELSE
5376   Cfr_bdfs=Sys_dfs(3)/Sys_dfs(1)
5379   IF Sys_dfs(5)=0 THEN
5382     Ler_bdfs=0
5385   ELSE
5388     Ler_bdfs=Sys_dfs(6)/Sys_dfs(5)
5391   END IF
5394   IF Cfr_bdfs=0 THEN
5397     Fer_bdfs=0
5400   ELSE
5403     Fer_bdfs=Ler_bdfs/Cfr_bdfs
5406   END IF
5409 END IF
5412 IF Sys_sum(5)=0 THEN
5415   Ler_b=0
5418 ELSE
5421   Ler_b=Sys_sum(6)/Sys_sum(5)
5424 END IF
5427 IF Ler_b=0 THEN
5430   Ler_r=0
5433 ELSE
5436   Ler_r=1/Ler_b
5439 END IF
5442 Sysum57=Sys_sum(5)-Sys_sum(7)
5445 IF Sysum57=0 THEN
5448   Ler_bwo=0
5451 ELSE
5454   Ler_bwo=Sys_sum(6)/(Sys_sum(5)-Sys_sum(7))
5457 END IF
5460 Fer_b=Ler_b/Cfr_b
5463 Fer_r=Ler_r/Cfr_r
5466 Fer_bwo=Ler_bwo/Cfr_bwo
5469 PRINT
5472 PRINT
5475 PRINT
```

Table 6-14. Ground combat code (continued).

```
5478 PRINT
5481 PRINT USING "20X,27A";"DIME SECTOR-BATTLE ANALYSIS"
5484 PRINT USING "/,22X,10A,2D,8A,3D";"GAME 2, CI",Turn,". SECTOR",Sector
5487 PRINT USING "/,24X,4A,19X,3A";"BLUE","RED"
5490 IF Sys_sum(7)>0 THEN
5493   PRINT USING "34X,2A";"BY"
5496   PRINT USING "16X,5A,3X,3A,3X,10A,5X,18A";"START","END","LOSS  HIND","ST
T   END   LOSS"
5499   PRINT USING Fmt35;"DFLR",Sys_dfl(1),Sys_dfl(2),Sys_dfl(5),Sys_dfl(7),Sy
dfl(3),Sys_dfl(4),Sys_dfl(6)
5502   PRINT USING Fmt35;"DFSR",Sys_dfs(1),Sys_dfs(2),Sys_dfs(5),Sys_dfs(7),Sy
dfs(3),Sys_dfs(4),Sys_dfs(6)
5505   PRINT USING Fmt35;" IF ",Sys_if(1),Sys_if(2),Sys_if(5),Sys_if(7),Sys_if
),Sys_if(4),Sys_if(6)
5508   PRINT USING Fmt35;" AD ",Sys_ad(1),Sys_ad(2),Sys_ad(5),Sys_ad(7),Sys_ad
),Sys_ad(4),Sys_ad(6)
5511   PRINT
5514   PRINT USING Fmt35;"SUM:",Sys_sum(1),Sys_sum(2),Sys_sum(5),Sys_sum(7),Sy
sum(3),Sys_sum(4),Sys_sum(6)
5517   PRINT USING "//,20X,8A,2X,10A";"W/O HIND","DFLR  DFSR"
5520   PRINT USING Fmt37;"CFR = ",Cfr_b,Cfr_bwo,Cfr_bdfl,Cfr_bdfs,"CFR = ",Cfr
5523   PRINT USING Fmt37;"LER = ",Ler_b,Ler_bwo,Ler_bdfl,Ler_bdfs,"LER = ",Ler
5526   PRINT USING Fmt37;"FER = ",Fer_b,Fer_bwo,Fer_bdfl,Fer_bdfs,"FER = ",Fer
5529 ELSE
5532   PRINT USING "/,16X,5A,3X,3A,3X,4A,11X,18A";"START","END","LOSS","START
END   LOSS"
5535   PRINT USING Fmt36;"DFLR",Sys_dfl(1),Sys_dfl(2),Sys_dfl(5),Sys_dfl(3),Sy
dfl(4),Sys_dfl(6)
5538   PRINT USING Fmt36;"DFSR",Sys_dfs(1),Sys_dfs(2),Sys_dfs(5),Sys_dfs(3),Sy
dfs(4),Sys_dfs(6)
5541   PRINT USING Fmt36;" IF ",Sys_if(1),Sys_if(2),Sys_if(5),Sys_if(3),Sys_if
),Sys_if(6)
5544   PRINT USING Fmt36;" AD ",Sys_ad(1),Sys_ad(2),Sys_ad(5),Sys_ad(3),Sys_ad
),Sys_ad(6)
5547   PRINT
5550   PRINT USING Fmt36;"SUM:",Sys_sum(1),Sys_sum(2),Sys_sum(5),Sys_sum(3),Sy
sum(4),Sys_sum(6)
5553   PRINT USING "//,30X,10A";"DFLR  DFSR"
5556   PRINT USING Fmt38;"CFR = ",Cfr_b,Cfr_bdfl,Cfr_bdfs,"CFR = ",Cfr_r
5559   PRINT USING Fmt38;"LER = ",Ler_b,Ler_bdfl,Ler_bdfs,"LER = ",Ler_r
5562   PRINT USING Fmt38;"FER = ",Fer_b,Fer_bdfl,Fer_bdfs,"FER = ",Fer_r
5565 END IF
5568 Fmt35:IMAGE 10X,4A,2(1X,4D.1D),2(1X,3D.1D),4X,2(4D.1D,1X),3D.1D
5571 Fmt36:IMAGE 10X,4A,2(1X,4D.1D),1X,3D.1D,10X,2(4D.1D,1X),3D.1D
5574 Fmt37:IMAGE 8X,6A,3D.2D,3X,3D.2D,1X,3D.2D,1X,3D.2D,8X,6A,3D.2D
5577 Fmt38:IMAGE 8X,6A,3D.2D,9X,3D.2D,1X,3D.2D,8X,6A,3D.2D
5580  !
5583 RETURN
5586  !
5589  !-------------------------------------------------------------------
5592 Read_files:    ! THIS SBR READS DATA FILES USED IN 30 MINUTE BATTLES
5595 Dcdisk$=":9134,704,0"
```

Table 6-14. Ground combat code (continued).

```
5598  !
5601  !********************!
5604  !   READ MINE FILE   !
5607  !********************!
5610 IF No_minefields>0 THEN
5613   ASSIGN @Pmine TO "MINE_FRCT"&Dcdisk$
5616   ENTER @Pmine,1;Mine_frct(*)
5619   ASSIGN @Pmine TO *
5622 END IF
5625  !***********************!
5628  !  READ INFANTRY FILES  !
5631  !***********************!
5634 ASSIGN @Pcond TO "CONVERT_D"&Dcdisk$
5637 ENTER @Pcond,1;Convertd(*)
5640 ASSIGN @Pcond TO *
5643  !
5646 ASSIGN @Pcona TO "CONVERT_A"&Dcdisk$
5649 ENTER @Pcona,1;Converta(*)
5652 ASSIGN @Pcona TO *
5655  !
5658  !LOAD CLASSIFIED FIREPOWER SCORES
5661 ASSIGN @P TO "FPS1"&Dcdisk$
5664 ENTER @P,1;Fpsb(*)
5667 ENTER @P,2;Fpsr(*)
5670 ASSIGN @P TO *
5673  !
5676  !************************!
5679  !  READ ARTILLERY FILES  !
5682  !************************!
5685  !  THIS REDUCES TOTAL AREA TO ONLY THAT AREA TASK IS TARGETED AGAINS
5688 ASSIGN @Parband TO "RD_AR_BAND"&Dcdisk$
5691 ENTER @Parband,1;R_area_band(*)     ! (task)
5694 ASSIGN @Parband TO *
5697  !
5700 ASSIGN @Parband TO "BL_AR_BAND"&Dcdisk$
5703 ENTER @Parband,1;B_area_band(*)     ! (task)
5706 ASSIGN @Parband TO *
5709  !
5712 ASSIGN @Pdspmsk TO "RD_DSP_MSK"&Dcdisk$
5715 ENTER @Pdspmsk,1;R_disprsn_mask(*)        ! (PHASE,MSN)
5718 ASSIGN @Pdspmsk TO *
5721  !
5724 ASSIGN @Pdspmsk TO "BL_DSP_MSK"&Dcdisk$
5727 ENTER @Pdspmsk,1;B_disprsn_mask(*)       ! (PHASE,MSN)
5730 ASSIGN @Pdspmsk TO *
5733  !
5736  !    MASK FOR TARGETS
5739  ! THIS ALLOWS SELECTIVE TARGETING FOR EACH TASK
5742 ASSIGN @Ptgtmsk TO "RD_TGT_MSK"&Dcdisk$
5745 ENTER @Ptgtmsk,1;R_tgt_mask(*)     ! (TASK,TGT ELM)
5748 ASSIGN @Ptgtmsk TO *
5751  !
```

Table 6-14. Ground combat code (continued).

```
5754 ASSIGN @Ptgtmsk TO "BL_TGT_MSK"&Dcdisk$
5757 ENTER @Ptgtmsk,1;B_tgt_mask(*)     ! (TASK,TGT ELM)
5760 ASSIGN @Ptgtmsk TO *
5763  !
5766  ! THIS IS INDIVIDUAL RD OR LCHR LOAD PKGD WT IN TONS
5769 ASSIGN @Pround TO "BL_ROUNDWT"&Dcdisk$
5772 ENTER @Pround,1;B_rd_wt(*)     ! (TYPE)
5775 ASSIGN @Pround TO *
5778  !
5781 ASSIGN @Pround TO "RD_ROUNDWT"&Dcdisk$
5784 ENTER @Pround,1;R_rd_wt(*)     ! (TYPE)
5787 ASSIGN @Pround TO *
5790  !
5793  ! PERCENT OF PERSONNEL PRONE
5796 ASSIGN @Ppsnpst TO "RD_PSN_PST"&Dcdisk$
5799 ENTER @Ppsnpst,1;R_psnl_posture(*)
5802 ASSIGN @Ppsnpst TO *
5805  !
5808 ASSIGN @Ppsnpst TO "BL_PSN_PST"&Dcdisk$
5811 ENTER @Ppsnpst,1;B_psnl_posture(*)      ! (%prone)(1st volley,later volley
5814 ASSIGN @Ppsnpst TO *
5817  !
5820 ASSIGN @Ptle TO "TLE"&Dcdisk$
5823 ENTER @Ptle,1;Tle(*)                ! Target Loc Error(task)
5826 ASSIGN @Ptle TO *
5829  !********************!
5832  !  READ SMOKE FILES  !
5835  !********************!
5838  !
5841 ASSIGN @Pawpp TO "AMWTPP"&Dcdisk$
5844.ENTER @Pawpp,1;Amwtpp(*)        !WT. IN LBS OF 1 ROUND FOR ARTY & MORT BY
IT TYPE
5847 ASSIGN @Pawpp TO *
5850  !
5853 ASSIGN @Prof TO "IROF"&Dcdisk$
5856 ENTER @Prof,1;Irof(*)        !  RATE OF FIRE FOR ARTY & MORT BY TYPE OF UN
5859 ASSIGN @Prof TO *
5862  !**************************!
5865  !  READ DIRECT FIRE FILES  !
5868  !**************************!
5871  !
5874  ! READS IN SENSOR DATA FOR 70 TARGETS VISIBLE THROUGH SMOKE
5877 ASSIGN @Psen TO "BL_SEN_DAY"&Dcdisk$
5880 ENTER @Psen,1;B_sen_d(*)
5883 ASSIGN @Psen TO *
5886  !
5889 ASSIGN @Psen TO "BL_SEN_NIT"&Dcdisk$
5892 ENTER @Psen,1;B_sen_n(*)
5895 ASSIGN @Psen TO *
5898  !
5901 ASSIGN @Psen TO "RD_SEN_DAY"&Dcdisk$
5904 ENTER @Psen,1;R_sen_d(*)
```

Table 6-14. Ground combat code (continued).

```
5907 ASSIGN @Psen TO *
5910  !
5913 ASSIGN @Psen TO "RD_SEN_NIT"&Dcdisk$
5916 ENTER @Psen,1;R_sen_n(*)
5919 ASSIGN @Psen TO *
5922  !
5925  !ENTER CATEGORY FILES
5928 ASSIGN @Path1 TO "BH_CAT"&Dcdisk$
5931 ASSIGN @Path2 TO "RD_CAT"&Dcdisk$
5934 ENTER @Path1,1;B_cat(*)
5937 ENTER @Path2,1;R_cat(*)
5940  !
5943  !ENTER AMMO WEIGHTS
5946 ASSIGN @Path1 TO "BH_DFAMO"&Dcdisk$
5949 ASSIGN @Path2 TO "RD_DFAMO"&Dcdisk$
5952 ENTER @Path1,1;B_ammo_wt(*)
5955 ENTER @Path1,3;Sen_ptr(*)      !READ BLUE SENSOR PTRS
5958 ENTER @Path1,4;Mun_ptr(*)      !READ BLUE MUNIT  PTRS
5961 FOR I=1 TO 20
5964   Df_sen_ptr(1,I)=Sen_ptr(I)
5967   Df_muni_ptr(1,I)=Mun_ptr(I)
5970 NEXT I
5973 ENTER @Path2,1;R_ammo_wt(*)
5976 ENTER @Path2,3;Sen_ptr(*)      !READ RED  SENSOR PTRS
5979 ENTER @Path2,4;Mun_ptr(*)      !READ RED  MUNIT  PTRS
5982 FOR I=1 TO 20
5985   Df_sen_ptr(2,I)=Sen_ptr(I)
5988   Df_muni_ptr(2,I)=Mun_ptr(I)
5991 NEXT I
5994 ASSIGN @Path1 TO *
5997 ASSIGN @Path2 TO *
6000  !******************!
6003  !  READ PGM FILES  !
6006  !******************!
6009  !
6012 ASSIGN @Ptgtval TO "TGT_VALS"&Dcdisk$
6015 ENTER @Ptgtval,1;Tgt_value(*)
6018 ASSIGN @Ptgtval TO *
6021  !
6024 ASSIGN @Ptgtmsk1 TO "TGT_MASK1"&Dcdisk$
6027 ENTER @Ptgtmsk1,1;Tgt_mask1(*)
6030 ASSIGN @Ptgtmsk1 TO *
6033  !
6036 ASSIGN @Pterr TO "TERR_FCT"&Dcdisk$
6039 ENTER @Pterr,1;Terr_factor(*)
6042 ASSIGN @Pterr TO *
6045  !
6048 ASSIGN @Pdust TO "DUST_ABRT"&Dcdisk$
6051 ENTER @Pdust,1;Prob_dustabort(*)
6054 ASSIGN @Pdust TO *
6057  !
6060 ASSIGN @Pclgpmsk TO "NS_CLGPMSK"&Dcdisk$
```

Table 6-14. Ground combat code (continued).

```
6063 ENTER @Pclgpmsk,1;Clgp_msk_ns(*)
6066 ASSIGN @Pclgpmsk TO *
6069  !
6072 ASSIGN @Pclgpmsk TO "GL_CLGPMSK"&Dcdisk$
6075 ENTER @Pclgpmsk,1;Clgp_msk_gl(*)
6078 ASSIGN @Pclgpmsk TO *
6081  !
6084 ASSIGN @Pclgpmsk TO "RV_CLGPMSK"&Dcdisk$
6087 ENTER @Pclgpmsk,1;Clgp_msk_rp(*)
6090 ASSIGN @Pclgpmsk TO *
6093  !
6096 ASSIGN @Pprbdes TO "NOS_PRBDES"&Dcdisk$
6099 ASSIGN @Pkpclgp TO "NOS_KPCLGP"&Dcdisk$
6102 ENTER @Pprbdes,1;Prob_dsg_ns(*)
6105 ENTER @Pkpclgp,1;Sskp_ns(*)
6108 ASSIGN @Pprbdes TO *
6111 ASSIGN @Pkpclgp TO *
6114  !
6117 ASSIGN @Pprbdes TO "GLL_PRBDES"&Dcdisk$
6120 ASSIGN @Pkpclgp TO "GLL_KPCLGP"&Dcdisk$
6123 ENTER @Pprbdes,1;Prob_dsg_gl(*)
6126 ENTER @Pkpclgp,1;Sskp_gl(*)
6129 ASSIGN @Pprbdes TO *
6132 ASSIGN @Pkpclgp TO *
6135  !
6138 ASSIGN @Pprbdes TO "RPV_PRBDES"&Dcdisk$
6141 ASSIGN @Pkpclgp TO "RPV_KPCLGP"&Dcdisk$
6144 ENTER @Pprbdes,1;Prob_dsg_rp(*)
6147 ENTER @Pkpclgp,1;Sskp_rp(*)
6150 ASSIGN @Pprbdes TO *
6153 ASSIGN @Pkpclgp TO *
6156  !**********************************************!
6159  !  READ HELICOPTER FILES AND STORE IN ARRAYS   !
6162  !**********************************************!
6165 Side$(1)="BL"
6168 Side$(2)="RD"
6171 FOR H_side=1 TO 2
6174   ASSIGN @Helo_file TO Side$(H_side)&"HELICHAR"&Dcdisk$
6177   ENTER @Helo_file;Helo_char(*)    !READ ALL 3 RECORDS AT THE SAME TIME
6180   ASSIGN @Helo_file TO *
6183   FOR I=1 TO 3
6186     FOR Muni=1 TO 3
6189       Helo_load(H_side,I,Muni)=Helo_char(I,Muni+4)     !BASIC LOAD
6192     NEXT Muni
6195     Mast_mount(H_side,I)=Helo_char(I,8)    ! 0-MAST MOUNT, 1-NON MAST MT
6198   NEXT I
6201 !    READ IN DATA FROM SENSOR FILE (PROB OF DETECTION DATA)
6204   ASSIGN @Helo_file TO Side$(H_side)&"HELISENS"&Dcdisk$
6207   FOR I=1 TO 3      !RETRIEVE SENSOR RECORD FOR HELO TYPE I
6210     IF Helo_char(I,1)>0 THEN
6213       ENTER @Helo_file,Helo_char(I,1);Desc$,P_det_inf(*),P_det_tbar(*),Rm
(*),Rmax(*)
```

Table 6-14. Ground combat code (continued).

```
6216        FOR J=1 TO 8
6219          Pd_rmin(H_side,I,J)=Rmin(J)
6222          Pd_rmax(H_side,I,J)=Rmax(J)
6225        NEXT J
6228        FOR Jtarg=1 TO 5     !LOOP ON DETECTION CATEGORIES
6231          Pd_fe_inf_a(H_side,I,Jtarg)=P_det_inf(1,Jtarg,1)
6234          Pd_fe_inf_b(H_side,I,Jtarg)=P_det_inf(2,Jtarg,1)
6237          Pd_fe_inf_c(H_side,I,Jtarg)=P_det_inf(3,Jtarg,1)
6240          Pd_hd_inf_a(H_side,I,Jtarg)=P_det_inf(1,Jtarg,2)
6243          Pd_hd_inf_b(H_side,I,Jtarg)=P_det_inf(2,Jtarg,2)
6246          Pd_hd_inf_c(H_side,I,Jtarg)=P_det_inf(3,Jtarg,2)
6249          Pd_fe_tbar_a(H_side,I,Jtarg)=P_det_tbar(1,Jtarg,1)
6252          Pd_fe_tbar_b(H_side,I,Jtarg)=P_det_tbar(2,Jtarg,1)
6255          Pd_fe_tbar_c(H_side,I,Jtarg)=P_det_tbar(3,Jtarg,1)
6258          Pd_hd_tbar_a(H_side,I,Jtarg)=P_det_tbar(1,Jtarg,2)
6261          Pd_hd_tbar_b(H_side,I,Jtarg)=P_det_tbar(2,Jtarg,2)
6264          Pd_hd_tbar_c(H_side,I,Jtarg)=P_det_tbar(3,Jtarg,2)
6267        NEXT Jtarg
6270      END IF
6273    NEXT I
6276    ASSIGN @Helo_file TO *
6279 ! READ HELICOPTER PERFORMANCE FILE (PROB OF KILL, TIME MASKED & EXPOSED)
6282    ASSIGN @Helo_file TO Side$(H_side)&"HELIPERF"&Dcdisk$
6285    FOR I=1 TO 2
6288      IF H_side=1 THEN Helo_mis(1,I)=B_helo_msn(I)
6291      IF H_side=2 THEN Helo_mis(2,I)=R_helo_msn(I)
6294    NEXT I
6297    Helo_mis(1,3)=0
6300    Helo_mis(2,3)=0
6303    IF H_side=1 AND B_helo(3,1)>0 THEN Helo_mis(1,3)=B_helo_msn(1)
6306    IF H_side=2 AND R_helo(3,1)>0 THEN Helo_mis(2,3)=R_helo_msn(1)
6309    FOR I=1 TO 3      !RETRIEVE MUNITION RECORD FOR HELO TYPE I
6312      FOR Muni=1 TO 3
6315        IF Helo_char(I,Muni+1)>0 THEN        !BE SURE THIS POINTS TO A RECOR
6318          ENTER @Helo_file,Helo_char(I,Muni+1);Desc$,Pk(*),Pk_rmin(H_side.I
uni),Pk_rmax(H_side,I,Muni),Np(H_side,I,Muni),Fm(H_side,I,Muni),Tim_me(*)
6321          FOR Jtarg=1 TO 20        !LOOP ON TARGET CATEGORIES
6324            Pk_fe_a(H_side,I,Muni,Jtarg)=Pk(1,Jtarg,1)
6327            Pk_fe_b(H_side,I,Muni,Jtarg)=Pk(2,Jtarg,1)
6330            Pk_fe_c(H_side,I,Muni,Jtarg)=Pk(3,Jtarg,1)
6333            Pk_hd_a(H_side,I,Muni,Jtarg)=Pk(1,Jtarg,2)
6336            Pk_hd_b(H_side,I,Muni,Jtarg)=Pk(2,Jtarg,2)
6339            Pk_hd_c(H_side,I,Muni,Jtarg)=Pk(3,Jtarg,2)
6342          NEXT Jtarg
6345          IF Helo_mis(H_side,I)>0 THEN
6348            Tm(H_side,I,Muni)=Tim_me(Helo_mis(H_side,I))    !TIME MASKED
6351            Te(H_side,I,Muni)=Tim_me(Helo_mis(H_side,I)+3)     !TIME EXPOSED
6354          END IF
6357        END IF
6360      NEXT Muni
6363    NEXT I
6366    ASSIGN @Helo_file TO *
```

Table 6-14. Ground combat code (continued).

```
6369 !   READ HELICOPTER PREFERENCE FILES
6372   ASSIGN @Helo_file TO Side$(H_side)&"HELIPREF"&Dcdisk$
6375   IF H_side=1 THEN Terr=R_terr
6378   IF H_side=2 THEN Terr=B_terr
6381   FOR M=1 TO 3       !LOOP ON MISSIONS
6384     ENTER @Helo_file,M;Pref(*),Plos(*)
6387     FOR Jtarg=1 TO 20
6390       Tgt_pref(H_side,M,Jtarg)=Pref(Jtarg)
6393     NEXT Jtarg
6396     Plos_alpha(H_side,M)=Plos(Terr)      !PROB OF LOS ALPHA.BETA FOR THIS
6399     Plos_beta(H_side,M)=Plos(Terr+4)     !TYPE OF TERRAIN INPUT
6402   NEXT M
6405   ASSIGN @Helo_file TO *
6408 !  READ AD PERFORMANCE FILES
6411   ASSIGN @Helo_file TO Side$(H_side)&"ADPERF"&Dcdisk$
6414   FOR Iad=1 TO 7
6417     ENTER @Helo_file,Iad;Pd_inf_ad(*),Pd_tbar_ad(*),Rmin(*),Rmax(*),Pk_ad
),Pk_ad_rmin(H_side,Iad),Pk_ad_rmax(H_side,Iad),Pref_ad(*)
6420     FOR Jmast=1 TO 2       !LOOP ON MAST/NON-MAST
6423       Pd_inf_ad_a(H_side,Iad,Jmast)=Pd_inf_ad(1,Jmast)
6426       Pd_inf_ad_b(H_side,Iad,Jmast)=Pd_inf_ad(2,Jmast)
6429       Pd_inf_ad_c(H_side,Iad,Jmast)=Pd_inf_ad(3,Jmast)
6432       Pd_tbar_ad_a(H_side,Iad,Jmast)=Pd_tbar_ad(1,Jmast)
6435       Pd_tbar_ad_b(H_side,Iad,Jmast)=Pd_tbar_ad(2,Jmast)
6438       Pd_tbar_ad_c(H_side,Iad,Jmast)=Pd_tbar_ad(3,Jmast)
6441       Pk_ad_a(H_side,Iad,Jmast)=Pk_ad(1,Jmast)
6444       Pk_ad_b(H_side,Iad,Jmast)=Pk_ad(2,Jmast)
6447       Pk_ad_c(H_side,Iad,Jmast)=Pk_ad(3,Jmast)
6450     NEXT Jmast
6453     FOR J=1 TO 8       !LOOP ON ATMOSPHERES
6456       Pd_ad_rmin(H_side,Iad,J)=Rmin(J)
6459       Pd_ad_rmax(H_side,Iad,J)=Rmax(J)
6462     NEXT J
6465     FOR J=1 TO 3       !LOOP ON NO. OF HELOS
6468       Ad_pref(H_side,Iad,J)=Pref_ad(J)
6471     NEXT J
6474   NEXT Iad
6477   ASSIGN @Helo_file TO *
6480 !READ AD MISCELLANEOUS FILES (HOLDS AMMO DESCRIPTION)
6483   ASSIGN @Helo_file TO Side$(H_side)&"_ADAMMO"&Dcdisk$
6486   FOR Iad=1 TO 7
6489     ENTER @Helo_file,Iad;Rnd_wt(H_side,Iad),Rnds(H_side,Iad),Fad(H_side,I
)
6492   NEXT Iad
6495   ASSIGN @Helo_file TO *
6498 !READ PROBABILITY OF DETECTION CATEGORY FILE
6501   ASSIGN @Helo_file TO Side$(H_side)&"PDCAT"&Dcdisk$
6504   ENTER @Helo_file,1;Cat20(*)
6507   FOR Jtarg=1 TO 20
6510     Pd_cat(H_side,Jtarg)=Cat20(Jtarg)
6513   NEXT Jtarg
6516   ASSIGN @Helo_file TO *
```

Table 6-14. Ground combat code (continued).

```
6519 !READ DIRECT FILE PROBABILITY OF DETECTION FILE
6522   ASSIGN @Helo_file TO Side$(H_side)&"_DF_SENS"&Dcdisk$
6525   FOR I=1 TO 2  !LOOP ON SENSOR TYPE  1-OPTICAL. 2-THERMAL
6528     ENTER @Helo_file.I;Df_det_inf(*),Df_det_tbar(*).Rmin(*).Rmax(*)
6531     FOR J=1 TO 2     !1-MAST MNT, 2-NON MAST MNT
6534       Pd_inf_df_a(H_side,I,J)=Df_det_inf(1,J)
6537       Pd_inf_df_b(H_side,I,J)=Df_det_inf(2,J)
6540       Pd_inf_df_c(H_side,I,J)=Df_det_inf(3,J)
6543       Pd_tbar_df_a(H_side,I,J)=Df_det_tbar(1,J)
6546       Pd_tbar_df_b(H_side,I,J)=Df_det_tbar(2.J)
6549       Pd_tbar_df_c(H_side,I,J)=Df_det_tbar(3,J)
6552     NEXT J
6555     FOR J=1 TO 8  !LOOP ON ATMOSPHERES
6558       Pd_df_rmin(H_side,I,J)=Rmin(J)
6561       Pd_df_rmax(H_side,I,J)=Rmax(J)
6564     NEXT J
6567   NEXT I
6570   ASSIGN @Helo_file TO *
6573 !DIRECT FIRE MISCELLANEOUS FILE (HOLDS MUNITION DESCRIPTION)
6576   ASSIGN @Helo_file TO Side$(H_side)&"_DF_MUNI"&Dcdisk$
6579   FOR I=1 TO 2   !LOOP ON MUNI TYPE  1-GRND MISSILE,2-GRND KINETIC ENRGY
6582     ENTER @Helo_file,I;Df_rnds_eng(H_side,I),F_df(H_side,I)
6585   NEXT I
6588   ASSIGN @Helo_file TO *
6591 NEXT H_side
6594 RETURN
6597 !-----------------------------------------------------------------------
6600  !
6603 Check_brk_pt:  ! THIS SBR CHECKS FOR BATTLE BREAKPOINTS
6606  !
6609 Break_point=0
6612  !
6615  ! CALCULATE CURRENT FORCE EFFECTIVENESS
6618 IF Init_b_eff=0 THEN
6621   Curr_b_eff=0
6624 ELSE
6627   Curr_b_eff=B_cbt_eff/Init_b_eff
6630 END IF
6633 IF Init_r_eff=0 THEN
6636   Curr_r_eff=0
6639 ELSE
6642   Curr_r_eff=R_cbt_eff/Init_r_eff
6645 END IF
6648  !
6651  ! CHECK TO SEE IF GAME TURN TIME IS EXCEEDED
6654 IF Btl_time>=Max_btl_time OR Btl_phase=3 THEN
6657   Break_point=3
6660   IF Btl_time<2430 THEN
6663     PRINT USING Fmt10;"SECTOR BATTLE ENDS".Btl_time," HRS"."B/EFF: ",Curr
_eff."R/EFF: ",Curr_r_eff."RANGE: ",Btl_rg
6666   ELSE
6669     PRINT USING Fmt10;"SECTOR BATTLE ENDS".Btl_time-2400." HRS","B/EFF: "
```

Table 6-14. Ground combat code (continued).

```
urr_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6672    END IF
6675    GOTO End_brk_pt
6678 END IF
6681 Fmt10:IMAGE /,3X,25A,2X,4Z,4A,2X,7A,1D.2D,2X,7A,1D.2D,3X,7A,6D
6684  !
6687  ! CHECK TO SEE IF CASUALTY BREAKPOINT IS EXCEEDED
6690 IF Curr_b_eff<=B_cas_break THEN
6693    Break_point=1
6696    IF Btl_time<2430 THEN
6699      PRINT USING Fmt10;"BLUE BREAKS FOR LOSSES",Btl_time," HRS","B/EFF: ",
rr_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6702    ELSE
6705      PRINT USING Fmt10;"BLUE BREAKS FOR LOSSES",Btl_time-2400," HRS","B/EF
 ",Curr_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6708    END IF
6711 END IF
6714 IF Curr_r_eff<=R_cas_break THEN
6717    Break_point=2
6720    IF Btl_time<2430 THEN
6723      PRINT USING Fmt10;"RED BREAKS FOR LOSSES",Btl_time," HRS","B/EFF: ",C
r_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6726    ELSE
6729      PRINT USING Fmt10;"RED BREAKS FOR LOSSESS",Btl_time-2400," HRS","B/EF
 ",Curr_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6732    END IF
6735 END IF
6738  !
6741  ! CHECK TO SEE IF RANGE BREAKPOINT IS EXCEEDED
6744 IF Btl_rg<=B_rg_break THEN
6747    Break_point=1
6750    IF Btl_time<2430 THEN
6753      PRINT USING Fmt10:"BLUE BREAKS FOR RANGE",Btl_time," HRS","B/EFF: ",C
r_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6756    ELSE
6759      PRINT USING Fmt10:"BLUE BREAKS FOR RANGE",Btl_time-2400," HRS","B/EFF
",Curr_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6762    END IF
6765 END IF
6768 IF Btl_rg<=R_rg_break THEN
6771    Break_point=2
6774    IF Btl_time<2430 THEN
6777      PRINT USING Fmt10;"RED BREAKS FOR RANGE",Btl_time," HRS","B/EFF: ",Cu
_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6780    ELSE
6783      PRINT USING Fmt10:"RED BREAKS FOR RANGE",Btl_time-2400," HRS","B/EFF:
,Curr_b_eff,"R/EFF: ",Curr_r_eff,"RANGE: ",Btl_rg
6786    END IF
6789 END IF
6792  !
6795 End_brk_pt:!
6798 RETURN
```

Table 6-14. Ground combat code (continued).

```
6801   !
6804   !-----------------------------------------------------------------
6807   !
6810 Apport_wri_loss:  ! THIS SBR WRITES UNIT STATUS TO HISTORY FILE
6813   !
6816 Rif_left=0
6819 Bif_left=0
6822 FOR I=1 TO 15
6825   FOR J=1 TO 5
6828     Rif_left=Rif_left+Rif_msn_tons(I,J)
6831     Bif_left=Bif_left+Bif_msn_tons(I,J)
6834   NEXT J
6837 NEXT I
6840   !
6843 IF Rif_left<=0 THEN Rif_left=0
6846 IF Bif_left<=0 THEN Bif_left=0
6849 Bif_ammo_used=Bif_ammo_sv-Bif_left
6852 Rif_ammo_used=Rif_ammo_sv-Rif_left
6855   !
6858   ! WRITE OUT BLUE UNIT DATA
6861   !
6864 FOR I=1 TO No_b_unit
6867   ENTER @Unitpath,B_unit_no(I);N(*)
6870   FOR J=1 TO 70
6873     N(J)=N(J)-(Sys_tot(1,J)-Sys_tot(2,J))*B_con(I,J)
6876   NEXT J
6879 !
6882   IF Bdf_ammo_sv=0 THEN
6885     GOTO B_if_ammo
6888   ELSE
6891     B_df_ammo_used=(Bdf_ammo_sv-B_df_ammo)*B_unit(I,72)/Bdf_ammo_sv
6894     N(131)=N(131)-B_df_ammo_used
6897     IF N(131)<0 THEN N(131)=0
6900     N(139)=N(139)+B_df_ammo_used
6903   END IF
6906 B_if_ammo: !
6909   IF Bif_ammo_sv<=0 THEN
6912     GOTO B_ad_ammo
6915   ELSE
6918     If_used=(Bif_ammo_used)*B_unit(I,73)/Bif_ammo_sv
6921     N(132)=N(132)-If_used
6924     IF N(132)<0 THEN N(132)=0
6927     N(140)=N(140)+If_used
6930   END IF
6933 B_ad_ammo: !
6936   IF Bad_ammo_sv<=0 THEN
6939     GOTO End_b_ammo
6942   ELSE
6945     B_ad_used=(Bad_ammo_sv-B_ad_ammo)*B_unit(I,74)/Bad_ammo_sv
6948     N(133)=N(133)-B_ad_used
6951     IF N(133)<0 THEN N(133)=0
6954     N(141)=N(141)+B_ad_used
```

Table 6-14. Ground combat code (continued).

```
6957   END IF
6960 End_b_ammo: !
6963   OUTPUT @Unitpath,B_unit_no(I);N(*)
6966 NEXT I
6969  !
6972  ! WRITE OUT RED UNIT DATA
6975  !
6978 FOR I=1 TO No_r_unit
6981   ENTER @Unitpath,R_unit_no(I);N(*)
6984   FOR J=1 TO 70
6987     N(J)=N(J)-(Sys_tot(3,J)-Sys_tot(4,J))*R_con(I,J)
6990   NEXT J
6993 !
6996   IF Rdf_ammo_sv<=0 THEN
6999     GOTO R_if_ammo
7002   ELSE
7005     R_df_ammo_used=(Rdf_ammo_sv-R_df_ammo)*R_unit(I,72)/Rdf_ammo_sv
7008     N(131)=N(131)-R_df_ammo_used
7011     IF N(131)<0 THEN N(131)=0
7014     N(139)=N(139)+R_df_ammo_used
7017   END IF
7020 R_if_ammo: !
7023   IF Rif_ammo_sv<=0 THEN
7026     GOTO R_ad_ammo
7029   ELSE
7032     If_used=(Rif_ammo_used)*R_unit(I,73)/Rif_ammo_sv
7035     N(132)=N(132)-If_used
7038     IF N(132)<0 THEN N(132)=0
7041     N(140)=N(140)+If_used
7044   END IF
7047 R_ad_ammo: !
7050   IF Rad_ammo_sv=0 THEN
7053     GOTO End_red_ammo
7056   ELSE
7059     R_ad_used=(Rad_ammo_sv-R_ad_ammo)*R_unit(I,74)/Rad_ammo_sv
7062     N(133)=N(133)-R_ad_used
7065     IF N(133)<0 THEN N(133)=0
7068     N(141)=N(141)+R_ad_used
7071   END IF
7074 End_red_ammo: !
7077   OUTPUT @Unitpath,R_unit_no(I);N(*)
7080 NEXT I
7083  !
7086  ! UPDATE KILLER-VICTIM TABLES
7089 ENTER @Kvpath,1;Ci_kv_b(*)
7092 ENTER @Kvpath,2;Ci_kv_r(*)
7095 FOR I=1 TO 6
7098   FOR J=1 TO 70
7101     Ci_kv_b(I,J)=Ci_kv_b(I,J)+Kv_b(I,J)
7104     Ci_kv_r(I,J)=Ci_kv_r(I,J)+Kv_r(I,J)
7107   NEXT J
7110 NEXT I
```

Table 6-14. Ground combat code (continued).

```
7113 OUTPUT @Kvpath,1;Ci_kv_b(*)
7116 OUTPUT @Kvpath,2;Ci_kv_r(*)
7119  !
7122  ! UPDATE HELICOPTER FILES
7125 ENTER @Helopath,1;Ci_helo_b(*)
7128 ENTER @Helopath,2;Ci_helo_r(*)
7131 FOR I=1 TO 3
7134   FOR J=1 TO 6
7137     Ci_helo_b(I,J)=Ci_helo_b(I,J)+B_helo(I,J)
7140     Ci_helo_r(I,J)=Ci_helo_r(I,J)+R_helo(I,J)
7143   NEXT J
7146 NEXT I
7149 OUTPUT @Helopath,1;Ci_helo_b(*)
7152 OUTPUT @Helopath,2;Ci_helo_r(*)
7155  !
7158 RETURN
7161  !
7164  !------------------------------------------------------------------
7167  !
7170 Tally_cbt_eff:  ! THIS SBR CALCULATES THE ADJUSTED FIREPOWER SCORE
7173  !
7176 Prev30_b_eff=B_cbt_eff
7179 Prev30_r_eff=R_cbt_eff
7182 B_cbt_eff=0
7185 R_cbt_eff=0
7188 FOR I=1 TO 70
7191   B_cbt_eff=B_cbt_eff+Sys_tot(2,I)*Sys_eff(1,I)
7194   R_cbt_eff=R_cbt_eff+Sys_tot(4,I)*Sys_eff(2,I)
7197 NEXT I
7200  !.
7203 RETURN
7206  !
7209  !------------------------------------------------------------------
7212  !
7215 Helo_arrive:  ! THIS SBR CALCULATES THE ATK HELOS AVAILABLE THIS PERIOD
7218  !
7221  !   THIS SBR ASSUMES AN ON-STATION TIME OF 30 MINUTES FOR ATK HELOS AND
7224  !   90 MINUTE CYCLE TIME.  A SINGLE AH CAN ATTACK ONLY ONCE EVERY 2 HOUR
7227  !
7230  !INITIALIZE BOTH RED AND BLUE HELOS
7233 Bah1=0
7236 Bah2=0
7239 Bsct=0
7242 Rah1=0
7245 Rah2=0
7248 Rsct=0
7251  !
7254 Set_blue_helos:  ! SCHEDULE BLUE ATTACK HELICOPTERS
7257  !
7260  ! CHECK DELAY FACTORS
7263 Earliest_time=0
7266 IF B_helo_delay>0 THEN
```

Table 6-14. Ground combat code (continued).

```
7269   Delay_time=B_helo_delay
7272   GOSUB Chk_delay_time
7275 END IF
7278  !
7281  ! IF DELAYS ARE IN EFFECT NO ATK HELOS ARRIVE
7284 IF Btl_time<=Earliest_time OR (Btl_rg>B_helo_rg_delay AND B_helo_rg_delav
0) OR Vis=4 THEN
7287   Bah1=0
7290   Bah2=0
7293   Bsct=0
7296   GOTO Set_red_helos
7299 END IF
7302  !
7305 Set_blue_ah1:  ! SCHEDULE BLUE #1 ATK HELOS
7308  !
7311 Helo_alive=B_helo(1,1)-B_helo(1,2)
7314 IF Helo_alive<=0 THEN GOTO Set_blue_ah2
7317 Set_bah1_cell:Helo=Helo_alive*1/B_helo(1,3)       ! OA RATE=0.83
7320 IF Helo<1 AND B_helo(1,3)>1 THEN  !DECREASE # OF CELLS & TRY SETTING AGAI
7323   B_helo(1,3)=B_helo(1,3)-1
7326   B_helo(3,3)=B_helo(1,3)        ! SCTS HAVE SAME CELL# AS AH#1
7329   GOTO Set_bah1_cell
7332 ELSE
7335   IF Helo<1 AND B_helo(1,3)=1 THEN    !NO USE, NOT ENOUGH HELOS
7338     Bah1=0
7341     GOTO Set_blue_ah2
7344   END IF
7347 END IF
7350  !
7353  ! SCHEDULE THE ARRIVAL TIME OF THE CELL
7356 SELECT B_helo(1,3)
7359 CASE =1
7362   IF Last_bah1_seg=0 OR Time_seg-Last_bah1_seg=4 THEN
7365     Bah1=Helo
7368     Last_bah1_seg=Time_seg
7371   ELSE
7374     Bah1=0
7377   END IF
7380 CASE =2
7383   IF Last_bah1_seg=0 OR Time_seg-Last_bah1_seg=2 THEN
7386     Bah1=Helo
7389     Last_bah1_seg=Time_seg
7392   ELSE
7395     Bah1=0
7398   END IF
7401 CASE =3
7404   IF Last_bah1_seg=0 THEN
7407     Bah1=Helo
7410     Last_bah1_seg=Time_seg
7413     Bah1_seg=1
7416     GOTO Set_blue_ah2
7419   END IF
```

Table 6-14. Ground combat code (continued).

```
7422   IF Bah1_seg<3 THEN
7425     Bah1=Helo
7428     Bah1_seg=Bah1_seg+1
7431     Last_bah1_seg=Time_seg
7434   ELSE
7437     Bah1=0
7440     Bah1_seg=0
7443   END IF
7446 CASE =4
7449   Bah1=Helo
7452   Last_bah1_seg=Time_seg
7455 END SELECT
7458  !
7461 Set_blue_ah2:  ! SCHEDULE BLUE #2 ATK HELOS
7464  !
7467 Helo_alive=B_helo(2,1)-B_helo(2,2)
7470 IF Helo_alive<=0 THEN GOTO Set_blue_sct
7473 Set_bah2_cell:Helo=Helo_alive*1/B_helo(2,3)       ! OA RATE=0.83
7476 IF Helo<1 AND B_helo(2,3)>1 THEN
7479   B_helo(2,3)=B_helo(2,3)-1
7482   GOTO Set_bah2_cell
7485 ELSE
7488   IF Helo<1 AND B_helo(2,3)=1 THEN
7491     Bah2=0
7494     GOTO Set_blue_sct
7497   END IF
7500 END IF
7503  !
7506  ! SCHEDULE THE ARRIVAL TIME OF THE CELL
7509 SELECT B_helo(2,3)
7512 CASE =1
7515   IF Last_bah2_seg=0 OR Time_seg-Last_bah2_seg=4 THEN
7518     Bah2=Helo
7521     Last_bah2_seg=Time_seg
7524   ELSE
7527     Bah2=0
7530   END IF
7533 CASE =2
7536   IF Last_bah2_seg=0 OR Time_seg-Last_bah2_seg=2 THEN
7539     Bah2=Helo
7542     Last_bah2_seg=Time_seg
7545   ELSE
7548     Bah2=0
7551   END IF
7554 CASE =3
7557   IF Last_bah2_seg=0 THEN
7560     Bah2=Helo
7563     Last_bah2_seg=Time_seg
7566     Bah2_seg=1
7569     GOTO Set_blue_sct
7572   END IF
7575   IF Bah2_seg<3 THEN
```

Table 6-14. Ground combat code (continued).

```
7578      Bah2=Helo
7581      Bah2_seg=Bah2_seg+1
7584      Last_bah2_seg=Time_seg
7587    ELSE
7590      Bah2=0
7593      Bah2_seg=0
7596    END IF
7599 CASE =4
7602    Bah2=Helo
7605    Last_bah2_seg=Time_seg
7608 END SELECT
7611  !
7614 Set_blue_sct:  ! SCHEDULE BLUE SCOUT HELOS     CHANGED 12/11/86 DWS
7617 !GOTO 9188
7620  !
7623 !IF Bah1=0 THEN                         ! NO SCOUTS FLOWN IF NO BLUE AH1 FLOWI
7626 !  Bsct=0
7629 !  GOTO Set_red_helos
7632 !END IF
7635 Helo_alive=B_helo(3,1)-B_helo(3,2)
7638 IF Helo_alive=0 THEN Set_red_helos
7641 Set_bsct_cell:Helo=Helo_alive*1/B_helo(3,3)      ! OA RATE=0.83
7644 IF Bah1>0 THEN      !SCOUTS LASING FOR AH1
7647    IF Helo<1 OR Helo<.4*Bah1 THEN
7650      Bsct=0
7653      PRINT
7656      PRINT "   INSUFFICIENT SCOUTS TO CONTINUE,  REMAINING AH64 WILL OPER
AUTONOMOUS"
7659      GOTO Set_red_helos
7662    ELSE
7665      IF Helo>Bah1/1.667 THEN Helo=Bah1/1.667       ! USE 3:5 MIX FOR SCT:
7668      Bsct=Helo
7671    END IF
7674 ELSE        !SCOUTS WILL FIRE ON ITS OWN
7677    IF B_helo(1,1)>0 THEN      !SCOUTS ARE LASING, BUT NOT THIS 30 MIN PERI
7680      Bsct=0
7683      GOTO Set_red_helos
7686    END IF
7689    IF Helo<1 AND B_helo(3,3)>1 THEN
7692      B_helo(3,3)=B_helo(3,3)-1       !DECREASE # OF CELLS BY 1
7695      GOTO Set_bsct_cell
7698    ELSE
7701      IF Helo<1 AND B_helo(3,3)=1 THEN
7704        Bsct=0
7707        GOTO Set_red_helos
7710      END IF
7713    END IF
7716  ! SCHEDULE THE ARRIVAL TIME OF THE CELL
7719    SELECT B_helo(3,3)
7722    CASE =1
7725      IF Last_bsct_seg=0 OR Time_seg-Last_bsct_seg=4 THEN
7728        Bsct=Helo
```

Table 6-14. Ground combat code (continued).

```
7731        Last_bsct_seg=Time_seg
7734      ELSE
7737        Bsct=0
7740      END IF
7743    CASE =2
7746      IF Last_bsct_seg=0 OR Time_seg-Last_bsct_seg=2 THEN
7749        Bsct=Helo
7752        Last_bsct_seg=Time_seg
7755      ELSE
7758        Bsct=0
7761      END IF
7764    CASE =3
7767      IF Last_bsct_seg=0 THEN
7770        Bsct=Helo
7773        Last_bsct_seg=Time_seg
7776        Bsct_seg=1
7779        GOTO Set_red_helos
7782      END IF
7785      IF Bsct_seg<3 THEN
7788        Bsct=Helo
7791        Bsct_seg=Bsct_seg+1
7794        Last_bsct_seg=Time_seg
7797      ELSE
7800        Bsct=0
7803        Bsct_seg=0
7806      END IF
7809    CASE =4
7812      Bsct=Helo
7815      Last_bsct_seg=Time_seg
7818    END SELECT
7821 END IF                              !END OF CHANGES 12/11/86 DWS
7824 Set_red_helos:  ! SCHEDULE RED ATTACK HELICOPTERS
7827   !
7830   ! CHECK DELAY FACTORS
7833 Earliest_time=0
7836 IF R_helo_delay>0 THEN
7839   Delay_time=R_helo_delay
7842   GOSUB Chk_delay_time
7845 END IF
7848   !
7851   ! IF DELAYS ARE IN EFFECT NO ATK HELOS ARRIVE
7854 IF Btl_time<=Earliest_time OR (Btl_rg>R_helo_rg_delay AND R_helo_rg_delay
0) OR Vis=4 THEN
7857   Rah1=0
7860   GOTO End_helo_arrive
7863 END IF
7866   !
7869 Set_red_ah1:  ! SCHEDULE RED #1 ATK HELOS
7872   !
7875 Helo_alive=R_helo(1,1)-R_helo(1,2)
7878 IF Helo_alive<=0 THEN GOTO Set_red_ah2
7881 Set_rah1_cell:Helo=Helo_alive*1/R_helo(1,3)      ! OA RATE=0.75
```

Table 6-14. Ground combat code (continued).

```
7884 IF Helo<1 AND R_helo(1,3)>1 THEN
7887   R_helo(1,3)=R_helo(1,3)-1
7890   GOTO Set_rah1_cell
7893 ELSE
7896   IF Helo<1 AND R_helo(1,3)=1 THEN
7899     Rah1=0
7902     GOTO End_helo_arrive
7905   END IF
7908 END IF
7911  !
7914  ! SCHEDULE THE ARRIVAL TIME OF THE CELL
7917 SELECT R_helo(1,3)
7920 CASE =1
7923   IF Last_rah1_seg=0 OR Time_seg-Last_rah1_seg=4 THEN
7926     Rah1=Helo
7929     Last_rah1_seg=Time_seg
7932   ELSE
7935     Rah1=0
7938   END IF
7941 CASE =2
7944   IF Last_rah1_seg=0 OR Time_seg-Last_rah1_seg=2 THEN
7947     Rah1=Helo
7950     Last_rah1_seg=Time_seg
7953   ELSE
7956     Rah1=0
7959   END IF
7962 CASE =3
7965   IF Last_rah1_seg=0 THEN
7968     Rah1=Helo
7971     Last_rah1_seg=Time_seg
7974     Rah1_seg=1
7977     GOTO Set_red_ah2
7980   END IF
7983   IF Rah1_seg<3 THEN
7986     Rah1=Helo
7989     Rah1_seg=Rah1_seg+1
7992     Last_rah1_seg=Time_seg
7995   ELSE
7998     Rah1=0
8001     Rah1_seg=0
8004   END IF
8007 CASE =4
8010   Rah1=Helo
8013   Last_rah1_seg=Time_seg
8016 END SELECT
8019  !
8022 Set_red_ah2:    ! SCHEDULE RED #2 ATK HELOS    ROB
8025  !
8028 Helo_alive=R_helo(2,1)-R_helo(2,2)
8031 IF Helo_alive<=0 THEN Set_red_sct
8034 Set_rah2 ell:Helo=Helo_alive*1/R_helo(2,3)
8037 IF Helo<  AND R_helo(2,3)>1 THEN
```

Table 6-14. Ground combat code (continued).

```
8040   R_helo(2,3)=R_helo(2 3)-1
8043   GOTO Set_rah2_cell
8046 ELSE
8049   IF Helo<1 AND R_helo(2,3)=1 THEN
8052     Rah2=0
8055     GOTO Set_red_sct
8058   END IF
8061 END IF
8064 SELECT R_helo(2,3)
8067 CASE =1
8070   IF Last_rah2_seg=0 OR Time_seg-Last_rah2_seg=4 THEN
8073     Rah2=Helo
8076     Last_rah2_seg=Time_seg
8079   ELSE
8082     Rah2=0
8085   END IF
8088 CASE =2
8091   IF Last_rah2_seg=0 OR Time_seg-Last_rah2_seg=2 THEN
8094     Rah2=Helo
8097     Last_rah2_seg=Time_seg
8100   ELSE
8103     Rah2=0
8106   END IF
8109 CASE =3
8112   IF Last_rah2_seg=0 THEN
8115     Rah2=Helo
8118     Last_rah2_seg=Time_seg
8121     Rah2_seg=1
8124     GOTO Set_red_sct
8127   END IF
8130   IF Rah2_seg<3 THEN
8133     Rah2=Helo
8136     Rah2_seg=Rah2_seg+1
8139     Last_rah2_seg=Time_seg
8142   ELSE
8145     Rah2=0
8148     Rah2_seg=0
8151   END IF
8154 CASE =4
8157   Rah2=Helo
8160   Last_rah2_seg=Time_seg
8163 END SELECT
8166!
8169 Set_red_sct:  ! SCHEDULE RED SCOUT HELOS     NEW CODE 12/11/86 DWS
8172 Helo_alive=R_helo(3,1)-R_helo(3,2)
8175 IF Helo_alive=0 THEN End_helo_arrive
8178 Set_rsct_cell:Helo=Helo_alive*1/R_helo(3,3)      ! OA RATE=0.83
8181 IF Rah1>0 THEN     !SCOUTS LASING FOR AH1
8184   IF Helo<1 OR Helo<.4*Rah1 THEN
8187     Rsct=0
8190     PRINT
8193     PRINT "   INSUFFICIENT SCOUTS TO CONTINUE.  REMAINING RED AH1S WILL
```

Table 6-14. Ground combat code (continued).

```
RATE AUTONOMOUS"
8196       GOTO End_helo_arrive
8199     ELSE
8202       IF Helo>Rah1/1.667 THEN Helo=Rah1/1.667          ! USE 3:5 MIX FOR SCT
8205       Rsct=Helo
8208     END IF
8211 ELSE         !SCOUTS WILL FIRE ON ITS OWN
8214     IF Helo<1 AND R_helo(3,3)>1 THEN
8217       R_helo(3,3)=R_helo(3,3)-1        !DECREASE # OF CELLS BY 1
8220       GOTO Set_rsct_cell
8223     ELSE
8226       IF Helo<1 AND R_helo(3,3)=1 THEN
8229         Rsct=0
8232         GOTO End_helo_arrive
8235       END IF
8238     END IF
8241   ! SCHEDULE THE ARRIVAL TIME OF THE CELL
8244     SELECT R_helo(3,3)
8247     CASE =1
8250       IF Last_rsct_seg=0 OR Time_seg-Last_rsct_seg=4 THEN
8253         Rsct=Helo
8256         Last_rsct_seg=Time_seg
8259       ELSE
8262         Rsct=0
8265       END IF
8268     CASE =2
8271       IF Last_rsct_seg=0 OR Time_seg-Last_rsct_seg=2 THEN
8274         Rsct=Helo
8277         Last_rsct_seg=Time_seg
8280       ELSE
8283         Rsct=0
8286       END IF
8289     CASE =3
8292       IF Last_rsct_seg=0 THEN
8295         Rsct=Helo
8298         Last_rsct_seg=Time_seg
8301         Rsct_seg=1
8304         GOTO End_helo_arrive
8307       END IF
8310       IF Rsct_seg<3 THEN
8313         Rsct=Helo
8316         Rsct_seg=Rsct_seg+1
8319         Last_rsct_seg=Time_seg
8322       ELSE
8325         Rsct=0
8328         Rsct_seg=0
8331       END IF
8334     CASE =4
8337       Rsct=Helo
8340       Last_rsct_seg=Time_seg
8343     END SELECT
8346 END IF                                   !END OF NEW CODE 12/11/86 DWS
```

Table 6-14. Ground combat code (continued).

```
8349 End_helo_arrive: ! UPDATE SORTIE NUMBERS
8352 B_helo(1,6)=B_helo(1,6)+Bah1
8355 B_helo(2,6)=B_helo(2,6)+Bah2
8358 B_helo(3,6)=B_helo(3,6)+Bsct
8361 R_helo(1,6)=R_helo(1,6)+Rah1
8364 R_helo(2,6)=R_helo(2,6)+Rah2
8367 R_helo(2,6)=R_helo(2,6)+Rsct
8370  !
8373 RETURN
8376  !
8379  !-------------------------------------------------------------------
8382  !
8385 Chk_delay_time:  ! THIS SBR CALCULATES DELAY TIMES
8388  !
8391 Int_time=INT(St_time/100)*100
8394 Int_delay=INT(Delay_time/100)*100
8397 Minute=(St_time MOD 100)+(Delay_time MOD 100)
8400 SELECT Minute
8403 CASE =0
8406   Delay_minute=0
8409 CASE =30
8412   Delay_minute=30
8415 CASE =60
8418   Delay_minute=100
8421 END SELECT
8424    !
8427    ! SET EARLIEST ARRIVAL TIME
8430 Earliest_time=Int_time+Int_delay+Delay_minute
8433    !
8436 RETURN
8439    !
8442    ! ----------------------------------------------------------------
8445    !
8448 Ammo_breakdown:     ! THIS SBR APPORTIONS AMMO FOR USE BY WEAPONS SYSTEM
8451    !
8454 FOR Ikp=1 TO 15
8457   Sys_ammo(Ikp)=0
8460 NEXT Ikp
8463 If_ammo_divisor=0
8466    ! APPORTION IF AMMO AMONG ARTY, MORTARS, AND MLRS
8469 FOR J1=1 TO 15
8472   Sys_ammo(J1)=N(J1+20)*Arty_30min_wt(Side,J1)
8475   If_ammo_divisor=If_ammo_divisor+Sys_ammo(J1)
8478 NEXT J1
8481 FOR J1=1 TO 15
8484   IF If_ammo_divisor=0 THEN
8487     If_ammo(J1)=0
8490   ELSE
8493     If_ammo(J1)=N(132)*Sys_ammo(J1)/If_ammo_divisor
8496   END IF
8499 NEXT J1
8502    !
```

Table 6-14. Ground combat code (continued).

```
8505 RETURN
8508     !
8511     ! ----------------------------------------------------------------------
8514     !
8517 Arty_arrive:    ! THIS SBR CALCULATES THE AMOUNT OF INCOMING ARTILLERY F
8520     !
8523     !SET VISIBILITY FOR 100%-- WILL BE DEGRADED IF SMOKE IS FIRED
8526 FOR I=1 TO 3
8529   R_vis(I)=1
8532   B_vis(I)=1
8535 NEXT I
8538  !SET CAP FOR ARTY
8541 FOR I=1 TO 5
8544   FOR J=1 TO 7
8547     B_arty_cap(J,I)=Sys_tot(2,J+20)*Arty_30min_wt(1,J)*Bif_msn(I)/100
8550     R_arty_cap(J,I)=Sys_tot(4,J+20)*Arty_30min_wt(2,J)*Rif_msn(I)/100
8553   NEXT J
8556     !SET CAP FOR MLRS
8559   IF I=2 THEN Mort_cap
8562   FOR J=1 TO 4
8565     IF Bif_msn(2)>=100 THEN
8568       B_mlrs_cap(J,I)=0
8571     ELSE
8574       B_mlrs_cap(J,I)=Sys_tot(2,J+31)*Arty_30min_wt(1,J+11)*Bif_msn(I)/(1
-Bif_msn(2))
8577     END IF
8580     IF Rif_msn(2)>=100 THEN
8583       R_mlrs_cap(J,I)=0
8586     ELSE
8589       R_mlrs_cap(J,I)=Sys_tot(4,J+31)*Arty_30min_wt(2,J+11)*Rif_msn(I)/(1
-Rif_msn(2))
8592     END IF
8595   NEXT J
8598 Mort_cap:  !
8601   FOR J=1 TO 4
8604     B_mort_cap(J,I)=0
8607     R_mort_cap(J,I)=0
8610   NEXT J
8613   IF I=4 OR I=5 THEN Next_cap
8616   IF Int_bmort=0 THEN Next_mort_cr
8619   FOR J=1 TO 4
8622     B_mort_cap(J,I)=Sys_tot(2,J+27)*Arty_30min_wt(1,J+7)*Bif_msn(I)/Int_b
rt
8625   NEXT J
8628 Next_mort_cr: !
8631   IF Int_rmort=0 THEN Next_cap
8634   FOR J=1 TO 4
8637     R_mort_cap(J,I)=Sys_tot(4,J+27)*Arty_30min_wt(2,J+7)*Rif_msn(I)/Int_r
rt
8640   NEXT J
8643 Next_cap:NEXT I
8646 FOR I=1 TO 15
```

Table 6-14. Ground combat code (continued).

```
8649    Tot_arty(1,I)=0                              ' SUB-TOTAL OF AMT DELIVERED PER 30 MI
8652    Tot_arty(2,I)=0
8655    FOR J=1 TO 5
8658      Bif_fired(I,J)=0
8661      Rif_fired(I,J)=0
8664    NEXT J
8667 NEXT I
8670    !
8673 Clgp_msns=0
8676 Gamp_msns=0
8679 SELECT Clgp_rpv
8682 CASE 0       ! NO RPV
8685    FOR I=1 TO 7
8688      IF Sys_tot(2,2)+Sys_tot(2,3)<=0 OR Clgp_avail=0 THEN   'SET CLGP TO
8691        B_clgp_cap(I)=0
8694      ELSE
8697        B_clgp_cap(I)=Sys_tot(2,I+20)*Arty_30min_wt(1,I)*Perc_clgp*(Sys_tot
,2)+Sys_tot(2,3))/((Sys_tot(1,2)+Sys_tot(1,3))*100)   !?? STILL 2 & 3
8700      END IF
8703    NEXT I
8706 CASE 1       ! RPV AVAILABLE
8709    FOR I=1 TO 7
8712      IF Clgp_avail=0 THEN
8715        B_clgp_cap(I)=0
8718      ELSE
8721        B_clgp_cap(I)=Sys_tot(2,I+20)*Arty_30min_wt(1,I)*Perc_clgp/100
8724      END IF
8727    NEXT I
8730 END SELECT
8733 FOR I=1 TO 4
8736    B_gamp_cap(I)=Sys_tot(2,I+31)*Arty_30min_wt(1,I+11)*Perc_gamp/100
8739 NEXT I
8742    !
8745 IF Cloud_ht*3.28<=1500 THEN No_clgp_gamp
8748 IF Vis=4 THEN No_clgp
8751 IF Btl_rg>Ds_start(B_terr) THEN No_clgp_gamp
8754 IF Btl_rg-150<=Ds_start(B_terr) AND Btl_rg>Ds_start(B_terr)-500 AND Clgp
v=0 THEN No_clgp
8757 GOTO Allocate_smoke
8760 No_clgp_gamp:     !
8763 FOR I=1 TO 4
8766    B_mort_cap(I,2)=B_mort_cap(I,2)+B_gamp_cap(I)
8769    B_gamp_cap(I)=0
8772 NEXT I
8775 No_clgp: !
8778 FOR I=1 TO 7
8781    B_arty_cap(I,2)=B_arty_cap(I,2)+B_clgp_cap(I)
8784    B_clgp_cap(I)=0
8787 NEXT I
8790    !
8793 Allocate_smoke:!
8796 IF Btl_phase<>3 OR Btl_rg>3200 THEN Allocate_prep
```

Table 6-14. Ground combat code (continued).

```
8799 Fire_smoke:  !
8802 IF Break_point=1 THEN
8805  ! BLUE BREAKS...HENCE IT WILL SMOKE
8808   FOR I=1 TO 7
8811     B_smk_cap(I)=B_arty_cap(I,2)*Bif_msn(6)/100
8814     IF B_smk_cap(I)>Bif_msn_tons(I,2) THEN
8817       B_smok_tons(I)=Bif_msn_tons(I,2)
8820     ELSE
8823       B_smok_tons(I)=B_smk_cap(I)
8826     END IF
8829   NEXT I
8832   FOR I=8 TO 11
8835     B_smk_cap(I)=B_mort_cap(I-7,2)*Bif_msn(6)/100
8838     IF B_smk_cap(I)>Bif_msn_tons(I,2) THEN
8841       B_smok_tons(I)=Bif_msn_tons(I,2)
8844     ELSE
8847       B_smok_tons(I)=B_smk_cap(I)
8850     END IF
8853   NEXT I
8856   A_ammo_ton=0
8859   M_ammo_ton=0
8862   FOR I=1 TO 7          !TOTAL ARTY AMMO TONNAGE
8865     A_ammo_ton=B_smok_tons(I)+A_ammo_ton
8868   NEXT I
8871   FOR I=8 TO 11         !TOTAL MORT AMMO TONNAGE
8874     M_ammo_ton=B_smok_tons(I)+M_ammo_ton
8877   NEXT I
8880   IF A_ammo_ton<.1 AND M_ammo_ton<.1 THEN Allocate_prep
8883   GOSUB W_smoke
8886 ! UPDATE ARTILLERY CAPACITY FOR TONNAGE NOT USED
8889   FOR I=1 TO 7
8892     B_arty_cap(I,2)=B_arty_cap(I,2)-B_asmk_used(I)
8895     Bif_msn_tons(I,2)=Bif_msn_tons(I,2)-B_asmk_used(I)
8898   NEXT I
8901   FOR I=1 TO 4
8904     B_mort_cap(I,2)=B_mort_cap(I,2)-B_msmk_used(I)
8907     Bif_msn_tons(I+7,2)=Bif_msn_tons(I+7,2)-B_msmk_used(I)
8910   NEXT I
8913   Smoke_used=0
8916   FOR I=1 TO 4
8919     Smoke_used=Smoke_used+B_msmk_used(I)
8922   NEXT I
8925   IF Smoke_used>0 THEN
8928     PRINT
8931     PRINT USING Fmt_m_smk;"   TONS OF SMOKE FIRED BY BLUE MORTAR DURING
HDRAWAL",Smoke_used
8934   END IF
8937 Fmt_m_smk:IMAGE 55A,4D.1D
8940   Smoke_used=0
8943   FOR I=1 TO 7
8946     Smoke_used=Smoke_used+B_asmk_used(I)
8949   NEXT I
```

Table 6-14. Ground combat code (continued).

```
8952    IF Smoke_used>0 THEN
8955      PRINT
8958      PRINT USING Fmt_a_smk;"    TONS OF SMOKE FIRED BY BLUE ARTILLERY DURI
WITHDRAWAL",Smoke_used
8961    END IF
8964 ELSE
8967 ! RED FIRING SMOKE
8970    FOR I=1 TO 7
8973      R_smk_cap(I)=R_arty_cap(I,2)*Rif_msn(6)/100
8976      IF R_smk_cap(I)>Rif_msn_tons(I,2) THEN
8979        R_smok_tons(I)=Rif_msn_tons(I,2)
8982      ELSE
8985        R_smok_tons(I)=R_smk_cap(I)
8988      END IF
8991    NEXT I
8994    FOR I=8 TO 11
8997      R_smk_cap(I)=R_mort_cap(I-7,2)*Rif_msn(6)/100
9000      IF R_smk_cap(I)>Rif_msn_tons(I,2) THEN
9003        R_smok_tons(I)=Rif_msn_tons(I,2)
9006      ELSE
9009        R_smok_tons(I)=R_smk_cap(I)
9012      END IF
9015    NEXT I
9018    A_ammo_ton=0
9021    M_ammo_ton=0
9024    FOR I=1 TO 7       !TOTAL ARTY AMMO TONNAGE
9027      A_ammo_ton=R_smok_tons(I)+A_ammo_ton
9030    NEXT I
9033    FOR I=8 TO 11      !TOTAL MORT AMMO TONNAGE
9036      M_ammo_ton=R_smok_tons(I)+M_ammo_ton
9039    NEXT I
9042    IF A_ammo_ton<.1 AND M_ammo_ton<.1 THEN Allocate_prep
9045    GOSUB W_smoke
9048    ! UPDATE ARTILLERY CAPABILITY FOR AMMO NOT USED
9051    FOR I=1 TO 7
9054      R_arty_cap(I,2)=R_arty_cap(I,2)-R_asmk_used(I)
9057      Rif_msn_tons(I,2)=Rif_msn_tons(I,2)-R_asmk_used(I)
9060    NEXT I
9063    FOR I=1 TO 4
9066      R_mort_cap(I,2)=R_mort_cap(I,2)-R_msmk_used(I)
9069      Rif_msn_tons(I+7,2)=Rif_msn_tons(I+7,2)-R_msmk_used(I)
9072    NEXT I
9075    Smoke_used=0
9078    FOR I=1 TO 4
9081      Smoke_used=Smoke_used+R_msmk_used(I)
9084    NEXT I
9087    IF Smoke_used>0 THEN
9090      PRINT
9093      PRINT USING Fmt_m_smk;"    TONS OF SMOKE FIRED BY RED MORTARS DURING
HDRAWAL",Smoke_used
9096 Fmt_a_smk:IMAGE 58A,4D.1D
9099    END IF
```

Table 6-14. Ground combat code (continued).

```
9102   Smoke_used=0
9105   FOR I=1 TO 7
9108     Smoke_used=Smoke_used+R_asmk_used(I)
9111   NEXT I
9114   IF Smoke_used>0 THEN
9117     PRINT
9120     PRINT USING Fmt_a_smk;"   TONS OF SMOKE FIRED BY RED  ARTILLERY DUR:
WITHDRAWAL",Smoke_used
9123   END IF
9126 END IF
9129 Allocate_prep: ! SCHEDULE INCOMING PREP FIRES
9132  ! CHANGED TO ALLOW INDEPENDENT SCHEDULING OF FIRES ROB
9135 !IF Atk_def=0 THEN
9138   !Prep_time=R_prep_time
9141 !ELSE
9144   !Prep_time=B_prep_time
9147 !END IF
9150 !IF Btl_time<=Prep_fire_time THEN
9153   !FOR I=1 TO 15
9156     !Bif_fired(I,1)=0              ! TONS FIRED THIS 30 MIN
9159     !Rif_fired(I,1)=0
9162   !NEXT I
9165   !GOTO Allocate_clgp
9168 !END IF
9171     !
9174     ! ALLOCATE MORTAR FIRES FOR PREP
9177 IF Btl_rg>5700 THEN                       ! RANGE TOO GREAT FOR MORTARS
9180   FOR I=8 TO 11
9183     Bif_fired(I,1)=0
9186     Rif_fired(I,1)=0
9189   NEXT I
9192   GOTO Arty_prep
9195 END IF
9198 IF Btl_time<B_prep_time THEN 9294
9201 IF Btl_rg<B_dsmort_start THEN
9204   Tons_avail=0
9207   FOR I=8 TO 11         !MORTAR
9210     IF Bif_msn_tons(I,1)>0 THEN
9213       Bif_msn_tons(I,2)=Bif_msn_tons(I,1)+Bif_msn_tons(I,2)
9216       B_dsmort_avail(I-7)=B_dsmort_avail(I-7)+Bif_msn_tons(I,1)
9219       Tons_avail=Tons_avail+Bif_msn_tons(I,1)
9222       Bif_msn_tons(I,1)=0
9225     END IF
9228     B_mort_cap(I-7,2)=B_mort_cap(I-7,2)+B_mort_cap(I-7,1)
9231     Bif_fired(I,1)=0
9234   NEXT I
9237   IF Tons_avail>0 THEN
9240     PRINT
9243     PRINT "   RED  WITHIN BLUE MORT CS RANGE ; REMAINING BLUE PREP AMMO
ILABLE FOR CS"
9246   END IF
9249   GOTO Red_mort_prep
```

Table 6-14. Ground combat code (continued).

```
9252 END IF
9255 Blue_mort_prep:  !
9258 FOR I=8 TO 11
9261   IF Bif_msn_tons(I,1)<B_mort_cap(I-7,1) THEN
9264     Bif_fired(I,1)=Bif_msn_tons(I,1)
9267     Bif_msn_tons(I,1)=0
9270     Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,1)
9273   ELSE
9276     Bif_fired(I,1)=B_mort_cap(I-7,1)
9279     Bif_msn_tons(I,1)=Bif_msn_tons(I,1)-B_mort_cap(I-7,1)
9282     Tot_arty(1,I)=Tot_arty(1,I)+B_mort_cap(I-7,1)
9285   END IF
9288 NEXT I
9291     !
9294 Red_mort_prep: !
9297 IF Btl_time<R_prep_time THEN 9390
9300 IF Btl_rg<R_dsmort_start THEN
9303   Tons_avail=0
9306   FOR I=8 TO 11
9309     IF Rif_msn_tons(I,1)>0 THEN
9312       Rif_msn_tons(I,2)=Rif_msn_tons(I,1)+Rif_msn_tons(I,2)
9315       R_dsmort_avail(I-7)=R_dsmort_avail(I-7)+Rif_msn_tons(I,1)
9318       Tons_avail=Tons_avail+Rif_msn_tons(I,1)
9321       Rif_msn_tons(I,1)=0
9324     END IF
9327     R_mort_cap(I-7,2)=R_mort_cap(I-7,2)+R_mort_cap(I-7,1)
9330     Rif_fired(I,1)=0
9333   NEXT I
9336   IF Tons_avail>0 THEN
9339     PRINT
9342     PRINT "   BLUE WITHIN RED  MORT CS RANGE ; REMAINING RED  PREP AMMO
ILABLE FOR CS"
9345   END IF
9348   GOTO Arty_prep
9351 END IF
9354 FOR I=8 TO 11
9357   IF Rif_msn_tons(I,1)<R_mort_cap(I-7,1) THEN
9360     Rif_fired(I,1)=Rif_msn_tons(I,1)
9363     Rif_msn_tons(I,1)=0
9366     Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,1)
9369   ELSE
9372     Rif_fired(I,1)=R_mort_cap(I-7,1)
9375     Rif_msn_tons(I,1)=Rif_msn_tons(I,1)-R_mort_cap(I-7,1)
9378     Tot_arty(2,I)=Tot_arty(2,I)+R_mort_cap(I-7,1)
9381   END IF
9384 NEXT I
9387     !
9390 Arty_prep: ! ALLOCATE PREP FIRES FOR ARTILLERY PIECES
9393  !
9396 Blue_arty_prep:IF Btl_rg>12000 OR Btl_time<B_prep_time THEN
9399   FOR I=1 TO 7
9402     Bif_fired(I,1)=0
```

Table 6-14. Ground combat code (continued).

```
9405    NEXT I
9408    GOTO Red_arty_prep
9411 END IF
9414 IF Btl_rg<B_dsarty_start THEN
9417    Tons_avail=0
9420    FOR I=1 TO 7
9423      IF Bif_msn_tons(I,1)>0 THEN
9426        Bif_msn_tons(I,2)=Bif_msn_tons(I,1)+Bif_msn_tons(I,2)
9429        B_dsarty_avail(I)=B_dsarty_avail(I)+Bif_msn_tons(I,1)
9432        Tons_avail=Tons_avail+Bif_msn_tons(I,1)
9435        Bif_msn_tons(I,1)=0
9438      END IF
9441      B_arty_cap(I,2)=B_arty_cap(I,2)+B_arty_cap(I,1)
9444      Bif_fired(I,1)=0
9447    NEXT I
9450    IF Tons_avail>0 THEN
9453      PRINT
9456      PRINT "   RED  WITHIN BLUE ARTY CS RANGE ; REMAINING BLUE PREP AMMO
ILABLE FOR CS"
9459    END IF
9462    GOTO Red_arty_prep
9465 END IF
9468    !
9471 FOR I=1 TO 7
9474    IF Bif_msn_tons(I,1)<B_arty_cap(I,1) THEN
9477      Bif_fired(I,1)=Bif_msn_tons(I,1)
9480      Bif_msn_tons(I,1)=0
9483      Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,1)
9486    ELSE
9489      Bif_fired(I,1)=B_arty_cap(I,1)
9492      Bif_msn_tons(I,1)=Bif_msn_tons(I,1)-B_arty_cap(I,1)
9495      Tot_arty(1,I)=Tot_arty(1,I)+B_arty_cap(I,1)
9498    END IF
9501 NEXT I
9504    !
9507 Red_arty_prep: !
9510 IF Btl_rg>14000 OR Btl_time<R_prep_time THEN
9513    FOR I=1 TO 7
9516      Rif_fired(I,1)=0
9519    NEXT I
9522    GOTO Mlrs_prep
9525 END IF
9528 IF Btl_rg<R_dsarty_start THEN
9531    Tons_avail=0
9534    FOR I=1 TO 7
9537      IF Rif_msn_tons(I,1)>0 THEN
9540        Rif_msn_tons(I,2)=Rif_msn_tons(I,1)+Rif_msn_tons(I,2)
9543        R_dsarty_avail(I)=R_dsarty_avail(I)+Rif_msn_tons(I,1)
9546        Tons_avail=Tons_avail+Rif_msn_tons(I,1)
9549        Rif_msn_tons(I,1)=0
9552      END IF
9555      R_arty_cap(I,2)=R_arty_cap(I,2)+R_arty_cap(I,1)
```

Table 6-14. Ground combat code (continued).

```
9558   NEXT I
9561   IF Tons_avail>0 THEN
9564     PRINT
9567     PRINT "   BLUE WITHIN RED  ARTY CS RANGE ; REMAINING RED  PREP AMMO
ILABLE FOR CS"
9570   END IF
9573   GOTO Mlrs_prep
9576 END IF
9579   !
9582 FOR I=1 TO 7
9585   IF Rif_msn_tons(I,1)<R_arty_cap(I,1) THEN
9588     Rif_fired(I,1)=Rif_msn_tons(I,1)
9591     Rif_msn_tons(I,1)=0
9594     Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,1)
9597   ELSE
9600     Rif_fired(I,1)=R_arty_cap(I,1)
9603     Rif_msn_tons(I,1)=Rif_msn_tons(I,1)-R_arty_cap(I,1)
9606     Tot_arty(2,I)=Tot_arty(2,I)+R_arty_cap(I,1)
9609   END IF
9612 NEXT I
9615   !
9618 Mlrs_prep:   ! ALLOCATE PREP FIRES FOR MLRS SYSTEMS
9621   !
9624 FOR I=12 TO 15
9627   Bif_fired(I,1)=0
9630   Rif_fired(I,1)=0
9633 NEXT I
9636 IF Btl_rg>25000 THEN Red_mlrs_prep
9639 IF Blt_time<B_prep_time THEN Red_mlrs_prep
9642 IF Btl_rg<B_dsarty_start THEN Red_mlrs_prep
9645   !
9648 Blue_mlrs_prep: !
9651 FOR I=12 TO 15
9654   IF Bif_msn_tons(I,1)<B_mlrs_cap(I-11,1) THEN
9657     Bif_fired(I,1)=Bif_msn_tons(I,1)
9660     Bif_msn_tons(I,1)=0
9663     Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,1)
9666   ELSE
9669     Bif_fired(I,1)=B_mlrs_cap(I-11,1)
9672     Bif_msn_tons(I,1)=Bif_msn_tons(I,1)-B_mlrs_cap(I-11,1)
9675     Tot_arty(1,I)=Tot_arty(1,I)+B_mlrs_cap(I-11,1)
9678   END IF
9681 NEXT I
9684   !
9687 Red_mlrs_prep: !
9690 IF Btl_time<R_prep_time OR Btl_rg>15500 THEN Allocate_clgp
9693 IF Btl_rg<R_dsarty_start THEN Allocate_clgp
9696 FOR I=12 TO 15
9699   IF Rif_msn_tons(I,1)<R_mlrs_cap(I-11,1) THEN
9702     Rif_fired(I,1)=Rif_msn_tons(I,1)
9705     Rif_msn_tons(I,1)=0
9708     Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,1)
```

Table 6-14. Ground combat code (continued).

```
9711    ELSE
9714      Rif_fired(I,1)=R_mlrs_cap(I-11,1)
9717      Rif_msn_tons(I,1)=Rif_msn_tons(I,1)-R_mlrs_cap(I-11,1)
9720      Tot_arty(2,I)=Tot_arty(2,I)+R_mlrs_cap(I-11,1)
9723    END IF
9726  NEXT I
9729    !
9732    !
9735  Allocate_clgp:  ! ALLOCATE CLGP MISSIONS FOR THIS 30 MIN PD
9738    !
9741  IF Btl_rg>12000 OR Btl_time<B_prep_time THEN
9744    Clgp_msns=0
9747    Gamp_msns=0
9750    GOTO Allocate_ds
9753  END IF
9756  !
9759  IF Clgp_avail<=0 THEN
9762    FOR I=1 TO 7
9765      B_arty_cap(I,2)=B_arty_cap(I,2)+B_clgp_cap(I)
9768    NEXT I
9771    Clgp_msns=0
9774    GOTO Allocate_gamp
9777  END IF
9780  Tot_clgp=0
9783  FOR I=1 TO 7
9786    Tot_clgp=Tot_clgp+B_clgp_cap(I)
9789  NEXT I
9792  IF Tot_clgp>Clgp_avail THEN
9795    Clgp_msns=Clgp_avail
9798    Clgp_avail=0
9801  ELSE
9804    Clgp_msns=Tot_clgp
9807    Clgp_avail=Clgp_avail-Tot_clgp
9810  END IF
9813    !
9816    !
9819  Allocate_gamp:  ! ALLOCATE GAMP MISSIONS FOR THIS 30 MIN PERIOD
9822    !
9825  IF Btl_rg>5700 THEN
9828    Gamp_msns=0
9831    GOTO Allocate_ds
9834  END IF
9837  !
9840  IF Gamp_avail<=0 THEN
9843    FOR I=1 TO 4
9846      B_mort_cap(I,2)=B_mort_cap(I,2)+B_gamp_cap(I)
9849    NEXT I
9852    Gamp_msns=0
9855    GOTO Allocate_ds
9858  END IF
9861  Tot_gamp=0
9864  FOR I=1 TO 4
```

Table 6-14. Ground combat code (continued).

```
9867   Tot_gamp=Tot_gamp+B_gamp_cap(I)
9870 NEXT I
9873 IF Tot_gamp>Gamp_avail THEN
9876   Gamp_msns=Gamp_avail
9879   Gamp_avail=0
9882 ELSE
9885   Gamp_msns=Tot_gamp
9888   Gamp_avail=Gamp_avail-Tot_gamp
9891 END IF
9894   !
9897 Allocate_ds:   ! SCHEDULE INCOMING DIRECT SUPPORT FIRES
9900   !
9903 TRACE OFF
9906   ! MLRS DOES NOT FIRE IN DIRECT SUPPORT
9909 FOR I=12 TO 15
9912   Bif_msn_tons(I,2)=0
9915   Rif_msn_tons(I,2)=0
9918 NEXT I
9921   !
9924 Set_blue_ds:   !
9927 Tot_ds_avail=0
9930 FOR I=1 TO 7
9933   Tot_ds_avail=Tot_ds_avail+B_dsarty_avail(I)
9936 NEXT I
9939 IF Btl_rg>B_dsarty_start OR Tot_ds_avail=0 OR Btl_time<B_prep_time THEN
9942   FOR I=1 TO 11
9945     Bif_fired(I,2)=0
9948   NEXT I
9951   GOTO Set_b_dsmort
9954 END IF
9957   !
9960 Set_b_dsarty:IF Btl_rg<B_dsarty_start AND Btl_rg>=B_ds_conc_pt THEN
9963   X=B_dsarty_start-B_ds_conc_pt
9966   Y=B_ds_conc_level
9969   FOR I=1 TO 7
9972     IF B_dsarty_avail(I)>0 THEN
9975       Frac_arty(I)=(Y/X*(B_dsarty_start-Btl_rg))-(B_dsarty_fire(I)/B_dsart
_avail(I))
9978     ELSE
9981       Frac_arty(I)=0
9984     END IF
9987   NEXT I
9990   B_p30_artyrg=Btl_rg
9993   GOTO Set_b_dsarty_rd
9996 END IF
9999   !
10002 IF Btl_rg<B_ds_conc_pt AND B_p30_artyrg>=B_ds_conc_pt AND B_ds_shift=0 TH
N
10005  FOR I=3 TO 5
10008    FOR J=1 TO 7
10011      Bif_msn_tons(J,2)=Bif_msn_tons(J,2)+Bif_msn_tons(J,I)
10014      B_dsarty_avail(J)=B_dsarty_avail(J)+Bif_msn_tons(J,I)
```

Table 6-14. Ground combat code (continued).

```
10017       Bif_msn_tons(J,I)=0
10020     NEXT J
10023   NEXT I
10026   B_ds_shift=1
10029 END IF
10032   !
10035 IF B_ds_shift=1 THEN
10038   FOR I=3 TO 5
10041     FOR J=1 TO 7
10044       B_arty_cap(J,2)=B_arty_cap(J,2)+B_arty_cap(J,I)
10047       B_arty_cap(J,I)=0
10050     NEXT J
10053   NEXT I
10056   X=B_p30_artyrg-B_dsarty_brkrg
10059   IF X<=0 THEN
10062     X=500
10065     B_p30_artyrg=Btl_rg+500
10068   END IF
10071   FOR I=1 TO 7
10074     IF B_dsarty_avail(I)>0 THEN
10077       Y=1-(B_dsarty_fire(I)/B_dsarty_avail(I))
10080       Frac_arty(I)=(Y/X*(B_p30_artyrg-Btl_rg))
10083     ELSE
10086       Frac_arty(I)=0
10089     END IF
10092   NEXT I
10095   B_p30_artyrg=Btl_rg
10098 END IF
10101   !
10104 Set_b_dsarty_rd:IF B_ds_shift=0 THEN
10107   Arty_bound=B_ds_conc_level
10110 ELSE
10113   Arty_bound=1.0
10116 END IF
10119   !
10122 FOR I=1 TO 7
10125   IF Frac_arty(I)<.083 THEN Frac_arty(I)=.083
10128   IF Mine_hit<>0 AND Frac_arty(I)<.20 THEN Frac_arty(I)=.20
10131   IF B_dsarty_avail(I)>0 THEN
10134     IF Frac_arty(I)>Arty_bound-(B_dsarty_fire(I)/B_dsarty_avail(I)) THEN
10137       Frac_arty(I)=Arty_bound-(B_dsarty_fire(I)/B_dsarty_avail(I))
10140     END IF
10143   END IF
10146   !
10149   Ds_attempted(I)=Frac_arty(I)*B_dsarty_avail(I)
10152   IF Ds_attempted(I)>Bif_msn_tons(I,2) THEN Ds_attempted(I)=Bif_msn_tons(
2)
10155   IF Ds_attempted(I)<B_arty_cap(I,2) THEN
10158     Bif_fired(I,2)=Ds_attempted(I)
10161     Bif_msn_tons(I,2)=Bif_msn_tons(I,2)-Ds_attempted(I)
10164     Tot_arty(1,I)=Tot_arty(1,I)+Ds_attempted(I)
10167     B_dsarty_fire(I)=B_dsarty_fire(I)+Ds_attempted(I)
```

Table 6-14. Ground combat code (continued).

```
10170  ELSE
10173    Ds_attempted(I)=B_arty_cap(I,2)
10176    Bif_fired(I,2)=Ds_attempted(I)
10179    Bif_msn_tons(I,2)=Bif_msn_tons(I,2)-Ds_attempted(I)
10182    Tot_arty(1,I)=Tot_arty(1,I)+Ds_attempted(I)
10185    B_dsarty_fire(I)=B_dsarty_fire(I)+Ds_attempted(I)
10188  END IF
10191 NEXT I
10194  !
10197  !
10200 Set_b_dsmort: ! ALLOCATE BLUE MORTAR DS FIRES
10203 Tot_ds_avail=0
10206 FOR I=1 TO 4
10209  Tot_ds_avail=Tot_ds_avail+B_dsmort_avail(I)
10212 NEXT I
10215 IF Btl_rg>B_dsmort_start OR Tot_ds_avail=0 OR Btl_time<B_prep_time THEN
10218  FOR I=8 TO 11
10221    Bif_fired(I,2)=0
10224  NEXT I
10227  GOTO Set_red_ds
10230 END IF
10233  !
10236 IF Btl_rg<B_dsmort_start AND Btl_rg>=B_mo_conc_pt THEN
10239  X=B_dsmort_start-B_mo_conc_pt
10242  Y=B_mo_conc_level
10245  FOR I=1 TO 4
10248    IF B_dsmort_avail(I)>0 THEN
10251      Frac_mort(I)=(Y/X*(B_dsmort_start-Btl_rg))-(B_dsmort_fire(I)/B_dsmor
_avail(I))
10254    ELSE
10257      Frac_mort(I)=0
10260    END IF
10263  NEXT I
10266  B_p30_mortrg=Btl_rg
10269  GOTO Set_b_dsmort_rd
10272 END IF
10275  !
10278 IF Btl_rg<B_mo_conc_pt AND B_p30_mortrg>=B_mo_conc_pt AND B_mo_shift=0 TH
N
10281  B_mo_shift=1
10284 END IF
10287  !
10290 IF B_mo_shift=1 THEN
10293  X=B_p30_mortrg-B_dsmort_brkrg
10296  IF X<=0 THEN
10299    X=500
10302    B_p30_mortrg=Btl_rg+500
10305  END IF
10308  FOR I=1 TO 4
10311    IF B_dsmort_avail(I)>0 THEN
10314      Y=1-(B_dsmort_fire(I)/B_dsmort_avail(I))
10317      Frac_mort(I)=(Y/X*(B_p30_mortrg-Btl_rg))
```

Table 6-14. Ground combat code (continued).

```
10320     ELSE
10323       Frac_mort(I)=0
10326     END IF
10329   NEXT I
10332   B_p30_mortrg=Btl_rg
10335 END IF
10338   !
10341 Set_b_dsmort_rd:IF B_mo_shift=0 THEN
10344   Mort_bound=B_mo_conc_level
10347 ELSE
10350   Mort_bound=1.0
10353 END IF
10356   !
10359 FOR I=1 TO 4
10362   IF Frac_mort(I)<.083 THEN Frac_mort(I)=.083
10365   IF Mine_hit<>0 AND Frac_mort(I)<.20 THEN Frac_mort(I)=.20
10368   IF B_dsmort_avail(I)>0 THEN
10371     IF Frac_mort(I)>Mort_bound-(B_dsmort_fire(I)/B_dsmort_avail(I)) THEN
10374       Frac_mort(I)=Mort_bound-(B_dsmort_fire(I)/B_dsmort_avail(I))
10377     END IF
10380   END IF
10383   !
10386   Mo_attempted(I)=Frac_mort(I)*B_dsmort_avail(I)
10389   IF Mo_attempted(I)>Bif_msn_tons(I+7,2) THEN Mo_attempted(I)=Bif_msn_ton(
I+7,2)
10392   IF Mo_attempted(I)<B_mort_cap(I,2) THEN
10395     Bif_fired(I+7,2)=Mo_attempted(I)
10398     Bif_msn_tons(I+7,2)=Bif_msn_tons(I+7,2)-Mo_attempted(I)
10401     B_dsmort_fire(I)=B_dsmort_fire(I)+Mo_attempted(I)
10404     Tot_arty(1,I+7)=Tot_arty(1,I+7)+Mo_attempted(I)
10407   ELSE
10410     Mo_attempted(I)=B_mort_cap(I,2)
10413     Bif_fired(I+7,2)=Mo_attempted(I)
10416     Bif_msn_tons(I+7,2)=Bif_msn_tons(I+7,2)-Mo_attempted(I)
10419     B_dsmort_fire(I)=B_dsmort_fire(I)+Mo_attempted(I)
10422     Tot_arty(1,I+7)=Tot_arty(1,I+7)+Mo_attempted(I)
10425   END IF
10428 NEXT I
10431   !
10434 Set_red_ds:   !
10437 Tot_ds_avail=0
10440 FOR I=1 TO 7
10443   Tot_ds_avail=Tot_ds_avail+R_dsarty_avail(I)
10446 NEXT I
10449 IF Btl_rg>R_dsarty_start OR Tot_ds_avail=0 OR Btl_time<R_prep_time THEN
10452   FOR I=1 TO 11
10455     Rif_fired(I,2)=0
10458   NEXT I
10461   GOTO Set_r_dsmort
10464 END IF
10467   !
10470 Set_r_dsartv:IF Btl_rg<R_dsarty_start AND Btl_rg>=R_ds_conc_pt THEN
```

Table 6-14. Ground combat code (continued).

```
10473  X=R_dsarty_start-R_ds_conc_pt
10476  Y=R_ds_conc_level
10479  FOR I=1 TO 7
10482    IF R_dsarty_avail(I)>0 THEN
10485      Frac_arty(I)=(Y/X*(R_dsarty_start-Btl_rg))-(R_dsarty_fire(I)/R_dsar
_avail(I))
10488    ELSE
10491      Frac_arty(I)=0
10494    END IF
10497  NEXT I
10500  R_p30_artyrg=Btl_rg
10503  GOTO Set_r_dsarty_rd
10506 END IF
10509  !
10512 IF Btl_rg<R_ds_conc_pt AND R_p30_artyrg>=R_ds_conc_pt AND R_ds_shift=0 T
N
10515  FOR I=3 TO 5
10518    FOR J=1 TO 7
10521      Rif_msn_tons(J,2)=Rif_msn_tons(J,2)+Rif_msn_tons(J,I)
10524      R_dsarty_avail(J)=R_dsarty_avail(J)+Rif_msn_tons(J,I)
10527      Rif_msn_tons(J,I)=0
10530    NEXT J
10533  NEXT I
10536  R_ds_shift=1
10539 END IF
10542  !
10545 IF R_ds_shift=1 THEN
10548  FOR I=3 TO 5
10551    FOR J=1 TO 7
10554      R_arty_cap(J,2)=R_arty_cap(J,2)+R_arty_cap(J.I)
10557      R_arty_cap(J,I)=0
10560    NEXT J
10563  NEXT I
10566  X=R_p30_artyrg-R_dsarty_brkrg
10569  IF X<=0 THEN
10572    X=500
10575    B_p30_artyrg=Btl_rg+500
10578  END IF
10581  FOR I=1 TO 7
10584    IF R_dsarty_avail(I)>0 THEN
10587      Y=1-(R_dsarty_fire(I)/R_dsarty_avail(I))
10590      Frac_arty(I)=(Y/X*(R_p30_artyrg-Btl_rg))
10593    ELSE
10596      Frac_arty(I)=0
10599    END IF
10602  NEXT I
10605  R_p30_artyrg=Btl_rg
10608 END IF
10611  '
10614 Set_r_dsarty_rd:IF R_ds_shift=0 THEN
10617  Arty_bound=R_ds_conc_level
10620 ELSE
```

Table 6-14. Ground combat code (continued).

```
10623  Arty_bound=1.0
10626 END IF
10629  !
10632 FOR I=1 TO 7
10635  IF Frac_arty(I)<.083 THEN Frac_arty(I)=.083
10638  IF Mine_hit<>0 AND Frac_arty(I)<.20 THEN Frac_arty(I)=.20
10641  IF R_dsarty_avail(I)>0 THEN
10644    IF Frac_arty(I)>Arty_bound-(R_dsarty_fire(I)/R_dsarty_avail(I)) THEN
10647      Frac_arty(I)=Arty_bound-(R_dsarty_fire(I)/R_dsarty_avail(I))
10650    END IF
10653  END IF
10656  !
10659  Ds_attempted(I)=Frac_arty(I)*R_dsarty_avail(I)
10662  IF Ds_attempted(I)>Rif_msn_tons(I,2) THEN Ds_attempted(I)=Rif_msn_tons(
2)
10665  IF Ds_attempted(I)<R_arty_cap(I,2) THEN
10668    Rif_fired(I,2)=Ds_attempted(I)
10671    Rif_msn_tons(I,2)=Rif_msn_tons(I,2)-Ds_attempted(I)
10674    Tot_arty(2,I)=Tot_arty(2,I)+Ds_attempted(I)
10677    R_dsarty_fire(I)=R_dsarty_fire(I)+Ds_attempted(I)
10680  ELSE
10683    Ds_attempted(I)=R_arty_cap(I,2)
10686    Rif_fired(I,2)=Ds_attempted(I)
10689    Rif_msn_tons(I,2)=Rif_msn_tons(I,2)-Ds_attempted(I)
10692    Tot_arty(2,I)=Tot_arty(2,I)+Ds_attempted(I)
10695    R_dsarty_fire(I)=R_dsarty_fire(I)+Ds_attempted(I)
10698  END IF
10701 NEXT I
10704  !
10707  !
10710 Set_r_dsmort: ! ALLOCATE RED MORTAR DS FIRES
10713 Tot_ds_avail=0
10716 FOR I=1 TO 4
10719  Tot_ds_avail=Tot_ds_avail+R_dsmort_avail(I)
10722 NEXT I
10725 IF Btl_rg>R_dsmort_start OR Tot_ds_avail=0 OR Btl_time<R_prep_time THEN
10728  FOR I=8 TO 11
10731    Rif_fired(I,2)=0
10734  NEXT I
10737  GOTO 10944
10740 END IF
10743  !
10746 IF Btl_rg<R_dsmort_start AND Btl_rg>=R_mo_conc_pt THEN
10749  X=R_dsmort_start-R_mo_conc_pt
10752  Y=R_mo_conc_level
10755  FOR I=1 TO 4
10758    IF R_dsmort_avail(I)>0 THEN
10761      Frac_mort(I)=(Y/X*(R_dsmort_start-Btl_rg))-(R_dsmort_fire(I)/R_dsmo
_avail(I))
10764    ELSE
10767      Frac_mort(I)=0
10770    END IF
```

Table 6-14. Ground combat code (continued).

```
10773  NEXT I
10776  R_p30_mortrg=Btl_rg
10779  GOTO Set_r_dsmort_rd
10782 END IF
10785  !
10788 IF Btl_rg<R_mo_conc_pt AND R_p30_mortrg>=R_mo_conc_pt AND R_mo_shift=0 TH
N
10791  R_mo_shift=1
10794 END IF
10797  !
10800 IF R_mo_shift=1 THEN
10803  X=R_p30_mortrg-R_dsmort_brkrg
10806  IF X<=0 THEN
10809    X=500
10812    R_p30_mortrg=Btl_rg+500
10815  END IF
10818  FOR I=1 TO 4
10821    IF R_dsmort_avail(I)>0 THEN
10824      Y=1-(R_dsmort_fire(I)/R_dsmort_avail(I))
10827      Frac_mort(I)=(Y/X*(R_p30_mortrg-Btl_rg))
10830    ELSE
10833      Frac_mort(I)=0
10836    END IF
10839  NEXT I
10842 END IF
10845  !
10848 Set_r_dsmort_rd:IF R_mo_shift=0 THEN
10851  Mort_bound=R_mo_conc_level
10854 ELSE
10857  Mort_bound=1.0
10860 END IF
10863  !
10866 FOR I=1 TO 4
10869  IF Frac_mort(I)<.083 THEN Frac_mort(I)=.083
10872  IF Mine_hit<>0 AND Frac_mort(I)<.20 THEN Frac_mort(I)=.20
10875  IF R_dsmort_avail(I)>0 THEN
10878    IF Frac_mort(I)>Mort_bound-(R_dsmort_fire(I)/R_dsmort_avail(I)) THEN
10881      Frac_mort(I)=Mort_bound-(R_dsmort_fire(I)/R_dsmort_avail(I))
10884    END IF
10887  END IF
10890  !
10893  Mo_attempted(I)=Frac_mort(I)*R_dsmort_avail(I)
10896  IF Mo_attempted(I)>Rif_msn_tons(I+7,2) THEN Mo_attempted(I)=Rif_msn_tons(
I+7,2)
10899  IF Mo_attempted(I)<R_mort_cap(I,2) THEN
10902    Rif_fired(I+7,2)=Mo_attempted(I)
10905    Rif_msn_tons(I+7,2)=Rif_msn_tons(I+7,2)-Mo_attempted(I)
10908    R_dsmort_fire(I)=R_dsmort_fire(I)+Mo_attempted(I)
10911    Tot_arty(2,I+7)=Tot_arty(2,I+7)+Mo_attempted(I)
10914  ELSE
10917    Mo_attempted(I)=R_mort_cap(I,2)
10920    Rif_fired(I+7,2)=Mo_attempted(I)
```

Table 6-14. Ground combat code (continued).

```
10923     Rif_msn_tons(I+7,2)=Rif_msn_tons(I+7,2)-Mo_attempted(I)
10926     R_dsmort_fire(I)=R_dsmort_fire(I)+Mo_attempted(I)
10929     Tot_arty(2,I+7)=Tot_arty(2,I+7)+Mo_attempted(I)
10932  END IF
10935 NEXT I
10938  !
10941  !
10944 !Allocate_sead !SCHEDULE INCOMING SEAD MISSIONS  ! ROB
10947  !
10950  ! MORTARS DO NOT FIRE SEAD
10953 FOR I=8 TO 11
10956  Bif_msn_tons(I,3)=0
10959  Rif_msn_tons(I,3)=0
10962 NEXT I
10965  !
10968 IF Btl_rg>12000 OR Btl_time<B_prep_time THEN
10971  FOR I=1 TO 7
10974    Bif_fired(I,3)=0
10977  NEXT I
10980  GOTO Red_arty_sead
10983 END IF
10986 Blue_arty_sead: !
10989 FOR I=1 TO 7
10992  IF Bif_msn_tons(I,3)<B_arty_cap(I,3) THEN
10995    Bif_fired(I,3)=Bif_msn_tons(I,3)
10998    Bif_msn_tons(I,3)=0
11001    Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,3)
11004  ELSE
11007    Bif_fired(I,3)=B_arty_cap(I,3)
11010    Bif_msn_tons(I,3)=Bif_msn_tons(I,3)-(B_arty_cap(I,3))
11013    Tot_arty(1,I)=Tot_arty(1,I)+(B_arty_cap(I,3))
11016  END IF
11019 NEXT I
11022 !
11025 Red_arty_sead:!
11028 IF Blt_rg>14000 OR Btl_time<R_prep_time THEN
11031  FOR I=1 TO 7
11034    Rif_fired(I,3)=0
11037  NEXT I
11040  GOTO Blue_mlrs_sead
11043 END IF
11046 FOR I=1 TO 7
11049  IF Rif_msn_tons(I,3)<R_arty_cap(I,3) THEN
11052    Rif_fired(I,3)=Rif_msn_tons(I,3)
11055    Rif_msn_tons(I,3)=0
11058    Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,3)
11061  ELSE
11064    Rif_fired(I,3)=R_arty_cap(I,3)
11067    Rif_msn_tons(I,3)=Rif_msn_tons(I,3)-(R_arty_cap(I,3))
11070    Tot_arty(2,I)=Tot_arty(2,I)+(R_arty_cap(I,3))
11073  END IF
11076 NEXT I
```

Table 6-14. Ground combat code (continued).

```
11079 !
11082 Blue_mlrs_sead:IF Btl_rg>25000 OR Blt_time<B_prep_time THEN
11085  FOR I=12 TO 15
11088    Bif_fired(I,3)=0
11091  NEXT I
11094  GOTO Red_mlrs_sead
11097 END IF
11100 !
11103 FOR I=12 TO 15
11106  IF Bif_msn_tons(I,3)<B_mlrs_cap(I-11,3) THEN
11109    Bif_fired(I,3)=Bif_msn_tons(I,3)
11112    Bif_msn_tons(I,3)=0
11115    Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,3)
11118  ELSE
11121    Bif_fired(I,3)=B_mlrs_cap(I-11,3)
11124    Bif_msn_tons(I,3)=Bif_msn_tons(I,3)-(B_mlrs_cap(I-11,3))
11127    Tot_arty(1,I)=Tot_arty(1,I)+(B_mlrs_cap(I-11,3))
11130  END IF
11133 NEXT I
11136 !
11139 Red_mlrs_sead:!
11142 IF Btl_rg>15500 OR Blt_time<R_prep_time THEN
11145  FOR I=12 TO 15
11148    Rif_fired(I,3)=0
11151  NEXT I
11154  GOTO Allocate_cfire
11157 END IF
11160 FOR I=12 TO 15
11163  IF Rif_msn_tons(I,3)<R_mlrs_cap(I-11,3) THEN
11166    Rif_fired(I,3)=Rif_msn_tons(I,3)
11169    Rif_msn_tons(I,3)=0
11172    Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,3)
11175  ELSE
11178    Rif_fired(I,3)=R_mlrs_cap(I-11,3)
11181    Rif_msn_tons(I,3)=Rif_msn_tons(I,3)-(R_mlrs_cap(I-11,3))
11184    Tot_arty(2,I)=Tot_arty(2,I)+(R_mlrs_cap(I-11,3))
11187  END IF
11190 NEXT I
11193 !
11196 !
11199 Allocate_cfire:  ! SCHEDULE INCOMING COUNTERFIRE MISSIONS
11202  !
11205  ! MORTARS DO NOT FIRE COUNTERFIRE
11208 FOR I=8 TO 11
11211  Bif_msn_tons(I,4)=0
11214  Rif_msn_tons(I,4)=0
11217 NEXT I
11220  !
11223 IF Btl_rg>12000 OR Btl_time<B_prep_time THEN
11226  FOR I=1 TO 7
11229    Bif_fired(I,4)=0
11232  NEXT I
```

Table 6-14. Ground combat code (continued).

```
11235  GOTO Red_arty_cfire
11238 END IF
11241 Blue_arty_cfire: !
11244 FOR I=1 TO 7
11247  IF Bif_msn_tons(I,4)<B_arty_cap(I,4) THEN
11250    Bif_fired(I,4)=Bif_msn_tons(I,4)
11253    Bif_msn_tons(I,4)=0
11256    Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,4)
11259  ELSE
11262    Bif_fired(I,4)=B_arty_cap(I,4)
11265    Bif_msn_tons(I,4)=Bif_msn_tons(I,4)-(B_arty_cap(I,4))
11268    Tot_arty(1,I)=Tot_arty(1,I)+(B_arty_cap(I,4))
11271  END IF
11274 NEXT I
11277  !
11280 Red_arty_cfire: !
11283 IF Btl_rg>14000 OR Btl_time<R_prep_time THEN
11286  FOR I=1 TO 7
11289    Rif_fired(I,4)=0
11292  NEXT I
11295  GOTO Blue_mlrs_cfire
11298 END IF
11301 FOR I=1 TO 7
11304  IF Rif_msn_tons(I,4)<R_arty_cap(I,4) THEN
11307    Rif_fired(I,4)=Rif_msn_tons(I,4)
11310    Rif_msn_tons(I,4)=0
11313    Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,4)
11316  ELSE
11319    Rif_fired(I,4)=R_arty_cap(I,4)
11322    Rif_msn_tons(I,4)=Rif_msn_tons(I,4)-(R_arty_cap(I,4))
11325    Tot_arty(2,I)=Tot_arty(2,I)+(R_arty_cap(I,4))
11328  END IF
11331 NEXT I
11334  !
11337 Blue_mlrs_cfire:IF Btl_rg>25000 OR Btl_time<B_prep_time THEN
11340  FOR I=12 TO 15
11343    Bif_fired(I,4)=0
11346  NEXT I
11349  GOTO Red_mlrs_cfire
11352 END IF
11355  !
11358 FOR I=12 TO 15
11361  IF Bif_msn_tons(I,4)<B_mlrs_cap(I-11,4) THEN
11364    Bif_fired(I,4)=Bif_msn_tons(I,4)
11367    Bif_msn_tons(I,4)=0
11370    Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,4)
11373  ELSE
11376    Bif_fired(I,4)=B_mlrs_cap(I-11,4)
11379    Bif_msn_tons(I,4)=Bif_msn_tons(I,4)-(B_mlrs_cap(I-11,4))
11382    Tot_arty(1,I)=Tot_arty(1,I)+(B_mlrs_cap(I-11,4))
11385  END IF
11388 NEXT I
```

Table 6-14. Ground combat code (continued).

```
11391  !
11394 Red_mlrs_cfire: ! CALCULATE RED MRL COUNTERFIRE
11397  !
11400 IF Btl_rg>15500 OR Btl_time<R_prep_time THEN
11403  FOR I=12 TO 15
11406    Rif_fired(I,4)=0
11409  NEXT I
11412  GOTO Allocate_intd
11415 END IF
11418 FOR I=12 TO 15
11421  IF Rif_msn_tons(I,4)<R_mlrs_cap(I-11,4) THEN
11424    Rif_fired(I,4)=Rif_msn_tons(I,4)
11427    Rif_msn_tons(I,4)=0
11430    Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,4)
11433  ELSE
11436    Rif_fired(I,4)=R_mlrs_cap(I-11,4)
11439    Rif_msn_tons(I,4)=Rif_msn_tons(I,4)-(R_mlrs_cap(I-11,4))
11442    Tot_arty(2,I)=Tot_arty(2,I)+(R_mlrs_cap(I-11,4))
11445  END IF
11448 NEXT I
11451  !
11454  !
11457 Allocate_intd:  ! SCHEDULE INCOMING INTERDICTION MISSIONS
11460  !
11463  ! MORTARS DO NOT FIRE INTERDICTION
11466 FOR I=8 TO 11
11469  Bif_msn_tons(I,5)=0
11472  Rif_msn_tons(I,5)=0
11475 NEXT I
11478  !
11481 IF Btl_rg>12000 OR Btl_time<B_prep_time THEN
11484  FOR I=1 TO 7
11487    Bif_fired(I,5)=0
11490  NEXT I
11493  GOTO Red_arty_intd
11496 END IF
11499 Blue_arty_intd:!
11502  !
11505 FOR I=1 TO 7
11508  IF Bif_msn_tons(I,5)<B_arty_cap(I,5) THEN
11511    Bif_fired(I,5)=Bif_msn_tons(I,5)
11514    Bif_msn_tons(I,5)=0
11517    Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,5)
11520  ELSE
11523    Bif_fired(I,5)=B_arty_cap(I,5)
11526    Bif_msn_tons(I,5)=Bif_msn_tons(I,5)-(B_arty_cap(I,5))
11529    Tot_arty(1,I)=Tot_arty(1,I)+(B_arty_cap(I,5))
11532  END IF
11535 NEXT I
11538  !
11541 Red_arty_intd: !
11544 IF Btl_rg>14000 OR Btl_time<R_prep_time THEN
```

Table 6-14. Ground combat code (continued).

```
11547  FOR I=1 TO 7
11550    Rif_fired(I,5)=0
11553  NEXT I
11556  GOTO Blue_mlrs_intd
11559 END IF
11562  !
11565 FOR I=1 TO 7
11568  IF Rif_msn_tons(I,5)<R_arty_cap(I,5) THEN
11571    Rif_fired(I,5)=Rif_msn_tons(I,5)
11574    Rif_msn_tons(I,5)=0
11577    Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,5)
11580  ELSE
11583    Rif_fired(I,5)=R_arty_cap(I,5)
11586    Rif_msn_tons(I,5)=Rif_msn_tons(I,5)-(R_arty_cap(I,5))
11589    Tot_arty(2,I)=Tot_arty(2,I)+(R_arty_cap(I,5))
11592  END IF
11595 NEXT I
11598  !
11601 Blue_mlrs_intd:IF Btl_rg>25000 OR Btl_time<B_prep_time THEN
11604  FOR I=12 TO 15
11607    Bif_fired(I,5)=0
11610  NEXT I
11613  GOTO Red_mlrs_intd
11616 END IF
11619  !
11622 FOR I=12 TO 15
11625  IF Bif_msn_tons(I,5)<B_mlrs_cap(I-11,5) THEN
11628    Bif_fired(I,5)=Bif_msn_tons(I,5)
11631    Bif_msn_tons(I,5)=0
11634    Tot_arty(1,I)=Tot_arty(1,I)+Bif_fired(I,5)
11637  ELSE
11640    Bif_fired(I,5)=B_mlrs_cap(I-11,5)
11643    Bif_msn_tons(I,5)=Bif_msn_tons(I,5)-(B_mlrs_cap(I-11,5))
11646    Tot_arty(1,I)=Tot_arty(1,I)+(B_mlrs_cap(I-11,5))
11649  END IF
11652 NEXT I
11655  !
11658 Red_mlrs_intd:  ! CALCULATE RED MRL INTERDICTION
11661  !
11664 IF Btl_rg>15500 OR Btl_time<R_prep_time THEN
11667  FOR I=12 TO 15
11670    Rif_fired(I,5)=0
11673  NEXT I
11676  GOTO End_arty_arrive
11679 END IF
11682 FOR I=12 TO 15
11685  IF Rif_msn_tons(I,5)<R_mlrs_cap(I-11,5) THEN
11688    Rif_fired(I,5)=Rif_msn_tons(I,5)
11691    Rif_msn_tons(I,5)=0
11694    Tot_arty(2,I)=Tot_arty(2,I)+Rif_fired(I,5)
11697  ELSE
11700    Rif_fired(I,5)=R_mlrs_cap(I-11,5)
```

Table 6-14. Ground combat code (continued).

```
11703    Rif_msn_tons(I,5)=Rif_msn_tons(I,5)-(R_mlrs_cap(I-11,5))
11706    Tot_arty(2,I)=Tot_arty(2,I)+(R_mlrs_cap(I-11,5))
11709  END IF
11712 NEXT I
11715  !
11718  !
11721 End_arty_arrive:RETURN
11724  !
11727  ! -------------------------------------------------------------------
11730  !
11733 Calc_movement:  ! THIS SBR CALCULATES ATTACKER MOVEMENT DISTANCES DURIN
11736  !                  THE DESIGNATED 30 MINUTE PERIOD
11739  !
11742 Prev30_btl_rg=Btl_rg
11745  !
11748  ! SET NUMBER OF MINUTES ADVANCED
11751 IF Mine_delay<30 THEN
11754  Move_minutes=30-Mine_delay
11757 ELSE
11760  Move_minutes=0
11763  Mine_delay=Mine_delay-30
11766 END IF
11769  !
11772  ! SET UNSUPPRESSED ADVANCE RATE PER MINUTE
11775 IF Atk_def=0 THEN
11778  Mission=R_msn(1)
11781 ELSE
11784  Mission=B_msn(1)
11787 END IF
11790  !
11793 IF Btl_phase=1 THEN
11796  Phase=1
11799 ELSE
11802  Phase=3
11805 END IF
11808  !
11811 M_per_minute=Advance_rate(Phase+Ride,Mission)
11814 Unsupp_advance=M_per_minute*Move_minutes
11817  !
11820  ! CALCULATE SUPPRESSION FROM CASUALTIES (UP TO 40%)
11823  !     6% CASUALTIES IN THE PREVIOUS GAME TURN WILL CAUSE MAX
11826  !     MOVEMENT SUPPRESSION
11829  !
11832 IF Atk_def=0 THEN
11835  Casualty_level=(Prev30_r_eff/Init_r_eff)-Curr_r_eff
11838 ELSE
11841  Casualty_level=(Prev30_b_eff/Init_b_eff)-Curr_b_eff
11844 END IF
11847 IF Casualty_level>.06 THEN Casualty_level=.06
11850 Cas_suppr=.40*Casualty_level/.06
11853  !
11856  ! CALCULATE SUPPRESSION FROM ARTILLERY (UP TO 30%)
```

Table 6-14. Ground combat code (continued).

```
11859  !      DS AND COUNTERPREP SUPPRESS ATTACKER MOVEMENT
11862  !      A SINGLE 155mm EQUIV MISSION SUPPRESSES A CO-
11865  !      SIZED ELEMENT FOR 5 MINUTES
11868  !
11871 IF Atk_def=0 THEN
11874  Side=4
11877 ELSE
11880  Side=2
11883 END IF
11886  !
11889  ! TALLY SYSTEMS TO BE SUPPRESSED
11892 Tot_systems=0
11895 FOR I=1 TO 20    ! TALLY DIRECT FIRE PLATFORMS
11898  Tot_systems=Tot_systems+Sys_tot(Side,I)
11901 NEXT I
11904 FOR I=36 TO 47             !  TALLY SMALL ARMS (INFANTRY)
11907  Tot_systems=Tot_systems+Sys_tot(Side,I)/8
11910 NEXT I
11913  !
11916  ! TALLY AMOUNT OF INCOMING FIRE
11919 IF Atk_def=0 THEN
11922  FOR I=1 TO 7
11925    Incoming_arty(I)=Bif_fired(I,1)+Bif_fired(I,2)
11928  NEXT I
11931  FOR I=1 TO 4
11934    Incoming_mlrs(I)=Bif_fired(I+11,1)
11937  NEXT I
11940  FOR I=1 TO 4
11943    Incoming_mort(I)=Bif_fired(I+7,1)+Bif_fired(I+7,2)
11946  NEXT I
11949 ELSE
11952  FOR I=1 TO 7
11955    Incoming_arty(I)=Rif_fired(I,1)+Rif_fired(I,2)
11958  NEXT I
11961  FOR I=1 TO 4
11964    Incoming_mlrs(I)=Rif_fired(I+11,1)
11967  NEXT I
11970  FOR I=1 TO 4
11973    Incoming_mort(I)=Rif_fired(I+7,1)+Rif_fired(I+7,2)
11976  NEXT I
11979 END IF
11982  !
11985  ! TALLY 155mm MISSION EQUIVALENTS
11988 Arty_equiv=0
11991 Mort_equiv=0
11994 Mlrs_equiv=0
11997 FOR I=1 TO 7        !NEED TO CHANGE NO. OF TONS W/IN TYPES??
12000  Arty_equiv=Arty_equiv+Incoming_arty(I)/1.8   ! 1.8=TONS IN 1 FA 155mm MS
12003 NEXT I
12006 FOR I=1 TO 4
12009  Mort_equiv=Mort_equiv+Incoming_mort(I)/1.2 ! 1.2=TONS IN A MORTAR->155r
EQUIV
```

Table 6-14. Ground combat code (continued).

```
12012 NEXT I
12015 FOR I=1 TO 4
12018  Mlrs_equiv=Mlrs_equiv+Incoming_mlrs(I)/1.8
12021 NEXT I
12024 Tot_incom_msn=Arty_equiv+Mort_equiv+Mlrs_equiv
12027  !
12030  ! TALLY NUMBER OF COMPANY EQUIVALENTS
12033 Arty_level=0
12036 IF Tot_incom_msn=0 THEN GOTO Helo_suppress
12039 Company_equiv=Tot_systems/28
12042 Arty_level=Tot_incom_msn/Company_equiv
12045  !
12048  ! COMPUTE ARTY LEVEL
12051 IF Arty_level>6 THEN Arty_level=6
12054  !
12057  ! COMPUTE ARTILLERY SUPPRESSION
12060 IF Atk_def=0 THEN
12063  Arty_suppr=Arty_level*.30/6
12066 ELSE
12069  Arty_suppr=Arty_level*.40/6
12072 END IF
12075  !
12078  ! CALCULATE SUPPRESSION FROM ATK HELICOPTERS (UP TO 20%)
12081  !     ONE ATTACK HELICOPTER WILL SUPPRESS THE MOVEMENT
12084  !     OF 12 VEHICLES
12087 Helo_suppress:  !
12090 Tot_vehicles=0
12093 IF Atk_def=0 THEN
12096  Side=4
12099 ELSE
12102  Side=2
12105 END IF
12108 FOR I=1 TO 20
12111  Tot_vehicles=Tot_vehicles+Sys_tot(Side,I)
12114 NEXT I
12117!Tot_vehicles=Tot_vehicles+.5*Sys_tot(Side,48)+.20*Sys_tot(Side,10)
12120 IF Atk_def=0 THEN
12123  Atk_helos=Bah1+Bah2
12126 ELSE
12129  Atk_helos=Rah1+Rah2
12132 END IF
12135 Atk_helo_level=0
12138 IF Tot_vehicles=0 THEN GOTO Atkhelo_s
12141 Atk_helo_level=12*Atk_helos/Tot_vehicles
12144 IF Atk_helo_level>1 THEN Atk_helo_level=1
12147 Atkhelo_s:  !
12150 Atkhelo_suppr=Atk_helo_level*.30
12153  !
12156 Tot_move_suppr=Cas_suppr+Arty_suppr+Atkhelo_suppr
12159  !
12162  ! CALCULATE AMOUNT OF ADVANCE
12165  !
```

Table 6-14. Ground combat code (continued).

```
12168 Amt_of_advance=Unsupp_advance*(1-Tot_move_suppr)
12171  !
12174 RETURN
12177  !
12180  ! ------------------------------------------------------------------
12183  !
12186 Mine_encounter:  ! THIS SBR CHECKS FOR MINEFIELD ACTIVATION
12189  !
12192 Mine_hit=0
12195 IF No_minefields=0 THEN End_mine_encoun
12198 FOR I=1 TO No_minefields
12201  IF Btl_rg-Amt_of_advance<Minefield(I,1) AND Minefield(I,6)=0 THEN
12204     ! MINEFIELD ENCOUNTERED WHICH HAS NOT BEEN ASSESSED BEFORE
12207    Mine_hit=I
12210 !
12213 ! CALCULATE MINE DELAY AND SET MINE TACTIC
12216    SELECT Atk_def
12219    CASE 0 !RED IS ENTERING MINEFIELD
12222      Mine_vic$="RED"
12225     ! CHECK FOR OVER KILL BY MINES
12228      IF (Curr_r_eff-R_cas_break)<.05 THEN
12231        IF Minefield(I,1)>Df_rg THEN
12234          Mine_delay=30
12237        ELSE
12240          Mine_delay=45
12243        END IF
12246        Bul_bch=0
12249        GOTO Mine_clear
12252      END IF
12255    CASE 1 ! BLUE IS ENTERING MINEFIELD
12258      Mine_vic$="BLUE"
12261      ! CHECK FOR OVERKILL BY MINES
12264      IF (Curr_b_eff-B_cas_break)<.05 THEN
12267        SELECT Btl_phase
12270        CASE 1
12273          Mine_delay=30
12276        CASE 2,3
12279          Mine_delay=45
12282        END SELECT
12285        Bul_bch=0
12288        GOTO Mine_clear
12291      END IF
12294    END SELECT
12297    GOTO Bull_or_breach
12300 Mine_clear:     !
12303    PRINT
12306    PRINT "    ";Mine_vic$;" ENCOUNTERS MINEFIELD WITH CASUALTIES NEAR BR
POINT"
12309    PRINT
12312    PRINT "    ";Mine_vic$;" STOPS TO CLEAR MINEFIELD"
12315    GOTO End_delay
12318 Bull_or_breach:  !
```

Table 6-14. Ground combat code (continued).

```
12321     IF Minefield(I,1)>Df_rg THEN
12324       Max_delay=30
12327       Bul_bch=2
12330       Mine_tac$="BREACH"
12333     ELSE
12336       Max_delay=10
12339       Bul_bch=1
12342       Mine_tac$="BULL THROUGH"
12345     END IF
12348     Mine_delay=Max_delay*Minefield(Mine_hit,2)/Minefield(Mine_hit,3)*Mine
eld(Mine_hit,4)/100
12351     PRINT
12354     PRINT "    ";Mine_vic$;" ENCOUNTERS MINEFIELD"
12357     PRINT
12360     PRINT "    ";Mine_vic$;" ELECTS TO ";Mine_tac$;" MINEFIELD"
12363 End_delay:  !
12366     Prnt_mn_dlay=Mine_delay
12369     GOTO End_mine_encoun
12372   END IF
12375 NEXT I
12378 End_mine_encoun: !
12381 RETURN
12384   !
12387   ! -------------------------------------------------------------------
12390   !
12393 Phase1_btl: ! THIS SBR CONDUCTS THE ATTRITION ASSESSMENTS FOR THE
12396            !   PHASE 1 (PRE-CLOSURE FIRE) BATTLE
12399   !
12402 Phase_ct(1)=Phase_ct(1)+1 !COUNTS NUMBER OF 30MIN. INTERVALS OF THIS PHA
12405 Del_blue=0
12408 Del_red=0
12411 GOSUB Phase_int
12414   ! ** MINEFILD PORTION OF PHASE I **
12417   ! CHECK FOR MINEFIELD ASSESSMENTS
12420 IF Mine_hit=0 THEN GOTO Run_arty
12423 IF Minefield(Mine_hit,6)=1 THEN GOTO Run_arty
12426   !
12429 GOSUB Run_mine
12432   !
12435    !
12438 Run_arty: !** ARTILLERY ATTRITION OF PHASE I  **
12441    !
12444    !CHECK ON INDIRECT FIRE USED THIS PHASE
12447 Blue_aty=0      !BLUE FIRE FLAG
12450 Red_aty=0       !RED  FIRE FLAG
12453 FOR I=1 TO 15
12456  FOR J=1 TO 5
12459     IF Bif_fired(I,J)>0 THEN Blue_aty=1
12462     IF Rif_fired(I,J)>0 THEN Red_aty=1
12465  NEXT J
12468 NEXT I
12471    !
```

Table 6-14. Ground combat code (continued).

```
12474    !SET PARAMETERS FOR ARTY.
12477 IF Blue_aty=0 AND Red_aty=0 THEN GOTO Direct_fire
12480    !
12483 GOSUB Arty_sub
12486    !
12489 Direct_fire: ! ** DIRECT FIRE PORTION OF PHASE I **
12492 IF Atk_def=0 THEN
12495  A_pct_fwd=R_pct_fwd        !RED ATTACKER SET ATTACKER FORWARD
12498  D_pct_fwd=B_pct_fwd        !BLUE IS DEFENDER
12501 ELSE
12504  A_pct_fwd=B_pct_fwd        !BLUE ATTACKER
12507  D_pct_fwd=R_pct_fwd        !RED DEFENDER
12510 END IF
12513 FOR I=1 TO 2
12516  FOR J=1 TO 3
12519    Cell(I,2,J)=0           !ZERO OUT RED & BLUE HELO'S CELLS
12522  NEXT J
12525 NEXT I
12528 Cell(1,1,1)=Bah1           !LOAD UP CELLS
12531 Cell(1,1,2)=Bah2
12534 Cell(1,1,3)=Bsct
12537 Cell(2,1,1)=Rah1
12540 Cell(2,1,2)=Rah2
12543 Cell(2,1,3)=Rsct
12546 CALL Helo_range(Cell(*),Helo_mis(*),Stnd_off_rg(*))   !CALCULATE RANGES
12549    !
12552    !IF DEFENDER HAS NO FORCES FORWARD THEN THERE WILL BE NO DIRECT FIRE
12555    !BATTLE IN PHASE I.
12558 IF D_pct_fwd=0 THEN GOTO Helo_atrit
12561    !
12564    !DEFENDER IS FORWARD.  DETERMINE IF DEFENDER PULLED BACK FOR ATTRITIO
12567 IF Atk_def=0 THEN     !RED IS DEFENDER
12570       !CALCULATE FP SCORE
12573  CALL Fwdfp(Sys_tot(*),Bf_mask(*),Atk_def,D_pct_fwd,D_fp,Sys_eff(*))
12576 ELSE      !BLUE IS DEFENDER
12579       !CALCULATE FP SCORE
12582  CALL Fwdfp(Sys_tot(*),Rf_mask(*),Atk_def,D_pct_fwd,D_fp,Sys_eff(*))
12585 END IF
12588 !
12591    !
12594 IF D_fp>.25 THEN GOTO Start_direct
12597 PRINT
12600 PRINT "   FORWARD FORCES ATTRITED TO ";D_fp
12603 PRINT
12606 PRINT "   NO DIRECT FIRE PHASE I BATTLE IN STEP ";Phase_ct(1)
12609 GOTO Helo_atrit
12612    !
12615 Start_direct: !
12618 IF A_pct_fwd<>0 THEN GOTO Attack_fwd
12621    !
12624    !ATTACKING MAIN BODY ; SET UP ARRAY
12627 IF Atk_def=0 THEN GOTO Red_attck
```

Table 6-14. Ground combat code (continued).

```
12630 FOR I=1 TO 70       !SET UP BLUE MAIN BODY
12633  Sys(1,I)=Sys_tot(2,I)*Bdf_mask(1,I)*B_df_t(B_ms,I)
12636  Sys(5,I)=Sys_tot(2,I)*Bdf_mask(1,I)*B_f(B_ms,I)
12639 NEXT I                                          !******* ANTI-ARMOR
12642 FOR I=1 TO 70      !SET UP RED ELEMENTS
12645  Sys(3,I)=Sys_tot(4,I)*Rf_mask(I)*D_pct_fwd
12648  Sys(6,I)=Sys(3,I)
12651 NEXT I
12654   !SET UP VULNERABILITY--ASSUME MAIN BODY WILL HAVE GIVEN VULNERABILITY
12657 FOR I=1 TO 70
12660  Blue_vul(I)=B_v(B_ms,I)
12663  Red_vul(I)=.25                 !RED DEFENDERS FORWARD
12666 NEXT I
12669 GOTO Call_df_cbt
12672 Red_attck: ! RED ATTACKING WITH NO FORCES FORWARD
12675 FOR I=1 TO 70       !SET UP RED MAIN BODY
12678  Sys(3,I)=Sys_tot(4,I)*Rdf_mask(1,I)*R_df_t(R_ms,I)
12681  Sys(6,I)=Sys_tot(4,I)*Rdf_mask(1,I)*R_df_t(R_ms,I)
12684 NEXT I
12687 FOR I=1 TO 70       !SET UP BLUE ELEMENTS
12690  Sys(1,I)=Sys_tot(2,I)*Bf_mask(I)*D_pct_fwd
12693  Sys(5,I)=Sys(1,I)
12696 NEXT I
12699 Sys(1,5)=0  !TEMPORARY CODE FOR HTLE/ADEA ONLY
12702 FOR I=1 TO 70
12705  Red_vul(I)=R_v(R_ms,I)
12708  Blue_vul(I)=.25
12711 NEXT I
12714 GOTO Call_df_cbt
12717 Attack_fwd:  ! BOTH ATTACKER AND DEFENDER HAVE FORCES FORWARD
12720 FOR I=1 TO 70
12723  Sys(1,I)=Sys_tot(2,I)*Bf_mask(I)*B_pct_fwd
12726  Sys(5,I)=Sys(1,I)
12729  Sys(3,I)=Sys_tot(4,I)*Rf_mask(I)*R_pct_fwd
12732  Sys(6,I)=Sys(3,I)
12735 NEXT I
12738 Sys(1,5)=0  !TEMPORARY CODE FOR HTLD/ADEA  ONLY
12741 FOR I=1 TO 70
12744  Red_vul(I)=.25          !ASSUME BOTH RED AND BLUE
12747  Blue_vul(I)=.25           !ELEMENTS FORWARD HAVE ONLY 25% EXOPSED
12750 NEXT I
12753    !
12756    !NOW CAL GROUND ATTRITION ; NOTE-ONLY HANDLING 1 RANGE BAND AT 2000M
12759 Call_df_cbt: !
12762 Btl_phase=1      !BATTLE PHASE 1
12765 !SET RANGE BAND
12768 SELECT Vis
12771 CASE 1 TO 2
12774  Rng_band=4      !ENGAGEMENTS AT 2000m
12777 CASE 3
12780  Rng_band=3      !ENGAGEMENTS AT 1500m
12783 CASE 4
```

Table 6-14. Ground combat code (continued).

```
12786  Rng_band=1       !ENGAGEMENTS AT 0 TO 500m FOR 1km DAY
12789 END SELECT
12792 Terrain=B_terr      !FIGHT ON DEFENDERS TERRAIN
12795 IF Atk_def=1 THEN Terrain=R_terr
12798 Num_bands=.5       !ONLY HALF A RANGE BAND
12801 GOSUB Df_cbt
12804!
12807 Helo_atrit: ! ** HELICOPTER ATTRITION FOR PHASE I **
12810 IF Bah1+Bah2+Bsct<=0 AND Rah1+Rah2+Rsct<=0 THEN GOTO No_helo1
12813     !LOAD CELL SIZES
12816 Arty(1)=2                !ARTILLERY NOT SHOOTING
12819 Arty(2)=2
12822 Veh_ada(1)=Bveh_sup       !SUPPRESSION OF VEHICULAR & HAND-HELD ADA
12825 Veh_ada(2)=Rveh_sup
12828 Hnd_ada(1)=Bhnd_sup
12831 Hnd_ada(2)=Rhnd_sup
12834 Time_step=15
12837     !SET VULNERABILITY OF RED TARGETS
12840 IF L_blue_helos=Phase_ct(1)-1 THEN
12843  Del_red=Del_red+1
12846 ELSE
12849  Del_red=0
12852 END IF
12855 L_blue_helos=Phase_ct(1)
12858 FOR I=1 TO 70
12861  P_def(2,I)=R_v(R_ms,I)+Del_red*R_dv(R_ms,I)
12864  IF P_def(2,I)<.1 THEN P_def(2,I)=.1
12867  IF P_def(2,I)>.9 THEN P_def(2,I)=.9
12870 NEXT I
12873     !SET VULNERABILITY OF BLUE TARGETS
12876 IF L_red_helos=Phase_ct(1)-1 THEN
12879  Del_blue=Del_blue+1
12882 ELSE
12885  Del_blue=0
12888 END IF
12891 L_red_helos=Phase_ct(1)
12894 FOR I=1 TO 70
12897  P_def(1,I)=B_v(B_ms,I)+Del_blue*B_dv(B_ms,I)
12900  IF P_def(1,I)<.1 THEN P_def(1,I)=.1
12903  IF P_def(1,I)>.9 THEN P_def(1,I)=.9
12906 NEXT I
12909     ! ---PREPARE FORCES
12912 FOR I=1 TO 70     !RED GROUND TARGETS
12915  Target(1,I)=Sys_tot(4,I)*R_df_t(R_ms,I)
12918  Target(2,I)=0
12921  Helo_tgt(2,1,I)=Target(1,I)     !PASSED TO HELO_KILLS  (INITIAL NO.)
12924  Helo_tgt(2,2,I)=0               !REMAINING NO OF TGTS
12927 NEXT I
12930     !
12933     !APPORTION RED AD AMMO BASED ON RED TARGET ELEMENTS
12936     ! RED AD SYSTEMS
12939 Adside=2
```

Table 6-14. Ground combat code (continued).

```
12942 Ad_ammo=R_ad_ammo
12945 CALL Helo_ammo(Sys_tot(*),Target(*),Adside,Ad_ammo,Ad_helo(2),Basic_ld(*
Ammo_wt(*))
12948 Ad_sv(2)=Ad_helo(2)
12951 Sided=2
12954 FOR I=1 TO 70
12957  H_targ(3,I)=Target(1,I)
12960 NEXT I
12963 CALL Dismount(H_targ(*),R_ld_fact,R_msn(1),Sided,Btl_rg,R_mount,R_dmount
12966 CALL Apport_inf(Target(*),1,R_dmount)
12969     !RED HELOS--PREPARE BLUE FORCES
12972 FOR I=1 TO 70
12975  Target(1,I)=Sys_tot(2,I)*B_df_t(B_ms,I)
12978  Target(2,I)=0
12981  Helo_tgt(1,1,I)=Target(1,I)      !PASSED TO HELO_KILLS  (INITIAL NO.)
12984  Helo_tgt(1,2,I)=0                !REMAINING NO OF TGTS
12987 NEXT I
12990     !APPORTION BLUE  AD AMMO
12993     ! BLUE AD SYSTEMS
12996 Adside=1
12999 Ad_ammo=B_ad_ammo
13002 CALL Helo_ammo(Sys_tot(*),Target(*),Adside,Ad_ammo,Ad_helo(1),Basic_ld(*
Ammo_wt(*))
13005 Ad_sv(1)=Ad_helo(1)
13008 Df_ammo(1)=B_df_ammo
13011 Df_ammo(2)=R_df_ammo
13014 Sided=1
13017 FOR I=1 TO 70
13020  H_targ(1,I)=Target(1,I)
13023 NEXT I
13026 CALL Dismount(H_targ(*),B_ld_fact,B_msn(1),Sided,Btl_rg,B_mount,B_dmount
13029 CALL Apport_inf(Target(*),1,B_dmqunt)
13032!
13035 CALL Helo_kills(Cell(*),Helo_tgt(*),Ad_helo(*),Terr,Atk_prof(*),Helo_mis
),Day_nite,Time_step,P_def(*),Arty(*),Veh_ada(*),Hnd_ada(*),Stnd_off_rg(*),Vis
13038!
13041 R_ad_ammo=R_ad_ammo-(Ad_sv(2)-Ad_helo(2))
13044 B_ad_ammo=B_ad_ammo-(Ad_sv(1)-Ad_helo(1))
13047 B_df_ammo=Df_ammo(1)
13050 R_df_ammo=Df_ammo(2)
13053     !
13056     !STORE KILLS AND SUBTRACT BLUE HELICOPTERS
13059 FOR Side=1 TO 2
13062  FOR I=1 TO 70
13065     Sys_helo(Side,I)=Helo_tgt(Side,1,I)-Helo_tgt(Side,2,I)
13068  NEXT I
13071  IF Side=1 THEN CALL Inf_survive(H_targ(*),1,Sys_helo(*),1,B_ld_fact,Inf
urv(*))
13074  IF Side=2 THEN CALL Inf_survive(H_targ(*),3,Sys_helo(*),2,R_ld_fact,Inf
urv(*))
13077  FOR I=36 TO 40
13080     Sys_helo(Side,I)=Sys_helo(Side,I)+Inf_surv(I-35)
```

Table 6-14. Ground combat code (continued).

```
13083  NEXT I
13086 NEXT Side
13089    !REMOVE HELOS  (ATTACKING AND DEFENDING)
13092 FOR I=1 TO 3
13095  B_helo(I,2)=B_helo(I,2)+Cell(1,1,I)-Cell(1,2,I)
13098  R_helo(I,2)=R_helo(I,2)+Cell(2,1,I)-Cell(2,2,I)
13101 NEXT I
13104    !SUBTRACT SYSTEM LOSSES
13107 FOR I=1 TO 70
13110  Sys_tot(2,I)=Sys_tot(2,I)-Sys_helo(1,I)
13113  Sys_tot(4,I)=Sys_tot(4,I)-Sys_helo(2,I)
13116 NEXT I
13119 No_helo1:  !
13122    !
13125 ! ** PGM PORTION OF PHASE I **
13128 !SET UP CLGP AND GAMP ARRAY FACTORS
13131 FOR I=1 TO 70
13134  Gamp_fact(I)=R_df_t(R_ms,I)
13137  Clgp_fact(I)=2*R_pct_fwd
13140  IF R_pct_fwd=0 THEN Clgp_fact(I)=R_df_t(R_ms,I)*2
13143  IF Clgp_fact(I)>1 OR Clgp_rpv=1 THEN Clgp_fact(I)=1
13146 NEXT I
13149 GOSUB Clgp_gamp_atrit
13152    !
13155 GOSUB Updatek_v
13158    !
13161 End_phase1_btl:RETURN   ! ** END BATTLE PHASE I **
13164    !
13167    !-----------------------------------------------------------------
13170    !
13173 Phase2_btl:  ! THIS SBR ASSESSES ATTRITION IN THE PHASE 2 (DIRECT FIRE)
13176  !
13179 Phase_ct(2)=Phase_ct(2)+1    !COUNT NUMBER OF 30 MIN INTERVALS IN PHASE
13182 Btl_phase=2
13185 GOSUB Phase_int   !ZERO ALL KILL ARRAYS FOR THIS 30 MINUTES
13188  !
13191  ! ** MINEFIELD PORTION OF PHASE II **
13194 IF Mine_hit=0 THEN GOTO Arty_phase2
13197 IF Minefield(Mine_hit,6)=1 THEN GOTO Arty_phase2
13200 GOSUB Run_mine
13203 Arty_phase2: ! ** ARTILLERY PORTION OF PHASE II **
13206   !
13209   !CHECK ON INDIRECT FIRE USED IN THIS PHASE
13212 Blue_aty=0          !BLUE FIRE FLAG
13215 Red_aty=0           !RED  FIRE FLAG
13218 FOR I=1 TO 15
13221  FOR J=1 TO 5
13224    IF Bif_fired(I,J)>0 THEN Blue_aty=1
13227    IF Rif_fired(I,J)>0 THEN Red_aty=1
13230  NEXT J
13233 NEXT I
13236 IF Blue_aty=0 AND Red_aty=0 THEN GOTO Dir_fir2
```

Table 6-14. Ground combat code (continued).

```
13239 GOSUB Arty_sub
13242 Dir_fir2: ! ** DIRECT FIRE PORTION OF PHASE II **
13245   !SET UP FORCES FOR TARGETS IN DIRECT FIRE
13248 Del_30=Phase_ct(2)-1
13251 GOSUB Collect_trucks  !TEMPORARY CODE FOR HTLE/ADEA ONLY
13254 FOR I=1 TO 70
13257  Red_fct=R_f(R_ms,I)+Del_30*R_df(R_ms,I)
13260  Blue_fct=B_f(B_ms,I)+Del_30*B_df(B_ms,I)
13263  IF Red_fct>.95 THEN Red_fct=.95
13266  IF Blue_fct>.95 THEN Blue_fct=.95     !ADJUST FIRERS FOR THIS 30 MIN
13269  IF Blue_fct<.05 THEN Blue_fct=.05
13272  IF Red_fct<.05 THEN Red_fct=.05
13275  Sys(5,I)=Sys_tot(2,I)*Bdf_mask(Balb,I)*Blue_fct       !SET # BLUE FIRE
13278  Sys(6,I)=Sys_tot(4,I)*Rdf_mask(Ralb,I)*Red_fct        !SET # RED  FIRE
13281       !CALCULATE # OF SYSTEMS WHICH ARE TARGETS
13284  Red_f_t(I)=R_df_t(R_ms,I)+Del_30*R_df_dt(R_ms,I)
13287  Blue_f_t(I)=B_df_t(B_ms,I)+Del_30*B_df_dt(B_ms,I)
13290  IF Red_f_t(I)>1 THEN Red_f_t(I)=1
13293  IF Blue_f_t(I)>1 THEN Blue_f_t(I)=1
13296  IF Red_f_t(I)<.05 THEN Red_f_t(I)=.05
13299  IF Blue_f_t(I)<.05 THEN Blue_f_t(I)=.05
13302  Sys(1,I)=Sys_tot(2,I)*Blue_f_t(I)*Bdf_mask(Balb,I)
13305  Sys(3,I)=Sys_tot(4,I)*Red_f_t(I)*Rdf_mask(Ralb,I)
13308       !CALCULATE SYSTEM VULNERABILITIES
13311  Red_fct=R_v(R_ms,I)+R_dv(R_ms,I)*Del_30
13314  Blue_fct=B_v(B_ms,I)+B_dv(B_ms,I)*Del_30
13317  IF Red_fct>1 THEN Red_fct=1
13320  IF Blue_fct>1 THEN Blue_fct=1
13323  IF Red_fct<.05 THEN Red_fct=.05
13326  IF Blue_fct<.05 THEN Blue_fct=.05
13329  Red_vul(I)=Red_fct
13332  Blue_vul(I)=Blue_fct
13335 NEXT I
13338   !
13341 Sys(1,5)=0    !TEMPORARY CODE FOR HTLE/ADEA ONLY
13344   !SET UP GROUND ATRITION PARAMETERS
13347 Btl_phase=2
13350   !DETERMINE PROPER RANGE BAND
13353 Rng_band=Cur_bnd
13356 Num_bands=Df_500_bds
13359 IF Cur_bnd-Num_bands<0 THEN Num_bands=1
13362 IF Amt_of_advance=0 AND Cur_bnd<>0 THEN
13365  Cur_bnd=Cur_bnd      !NO ADVANCE
13368 ELSE
13371  Cur_bnd=Cur_bnd-Num_bands       !SET UP CURRENT BAND FOR NEXT CALL
13374 END IF
13377 IF Cur_bnd<0 THEN Rng_band=1   !DON'T ALLOW FORCES TOGET BEHIND EACH OTH
13380 Terrain=B_terr      !FIGHT ON DEFENDERS TERRAIN
13383 IF Atk_def=1 THEN Terrain=R_terr
13386 !
13389 FOR I=1 TO 2
13392  FOR J=1 TO 3
```

Table 6-14. Ground combat code (continued).

```
13395    Cell(I,2,J)=0           !ZERO OUT RED & BLUE HELO'S CELLS
13398  NEXT J
13401 NEXT I
13404 Cell(1,1,1)=Bah1           !LOAD UP CELLS
13407 Cell(1,1,2)=Bah2
13410 Cell(1,1,3)=Bsct
13413 Cell(2,1,1)=Rah1
13416 Cell(2,1,2)=Rah2
13419 Cell(2,1,3)=Rsct
13422 CALL Helo_range(Cell(*),Helo_mis(*),Stnd_off_rg(*))   !CALCULATE RANGES
                                                            !FOR HELICOPTERS
13425 GOSUB Df_cbt     !PERFORM DIRECT FIRE COMBAT
13428    !
13431    ! ** HELICOPTER ATTRITION OF PHASE II **
13434 IF Bah1+Bah2+Bsct<=0 AND Rah1+Rah2+Rsct<=0 THEN GOTO No_helo2
13437   !
13440 Arty(1)=2
13443 Arty(2)=2
13446 Veh_ada(1)=Bveh_sup         !SUPPRESSION OF VEHICULAR & HAND-HELD ADA
13449 Veh_ada(2)=Rveh_sup
13452 Hnd_ada(1)=Bhnd_sup
13455 Hnd_ada(2)=Rhnd_sup
13458 Time_step=15
13461   !BLUE HELOS ---- PREPARE RED FORCES
13464 FOR I=1 TO 70
13467  Target(1,I)=Sys_tot(4,I)*Red_f_t(I)*Rdf_mask(Ralb,I)
13470  Target(2,I)=0
13473  P_def(2,I)=Red_vul(I)          !SET VULNERABILITY
13476  Helo_tgt(2,1,I)=Target(1,I)   !PASSED TO HELO_KILLS (INITIAL NO OF TGTS
13479  Helo_tgt(2,2,I)=0             !REMAINING NO. OF TARGETS
13482 NEXT I
13485   !
13488   !APPORTION RED AD AMMO BASED ON RED TARGETS
13491   ! RED AD SYSTEMS
13494 Adside=2
13497 Ad_ammo=R_ad_ammo
13500 CALL Helo_ammo(Sys_tot(*),Target(*),Adside,Ad_ammo,Ad_helo(2),Basic_ld(
Ammo_wt(*))
13503 Ad_sv(2)=Ad_helo(2)
13506 Sided=2
13509 FOR I=1 TO 70
13512  H_targ(3,I)=Target(1,I)
13515 NEXT I
13518 CALL Dismount(H_targ(*),R_ld_fact,R_msn(1),Sided,Btl_rg,R_mount,R_dmoun
13521 CALL Apport_inf(Target(*),1,R_dmount)
13524 !
13527   !RED HELOS -- PREPARE FORCES
13530 FOR I=1 TO 70
13533  Target(1,I)=Sys_tot(2,I)*Blue_f_t(I)*Bdf_mask(Balb,I)
13536  Target(2,I)=0
13539  P_def(1,I)=Blue_vul(I)         !SET VULNERABILITY
13542  Helo_tgt(1,1,I)=Target(1,I)   !PASSED TO HELO_KILLS (INITIAL NO.)
```

Table 6-14. Ground combat code (continued).

```
13545  Helo_tgt(1,2,I)=0                 !SET REMAINING NO. OF TGTS TO 0
13548 NEXT I
13551    !APPORTION AD_AMMO TO BLUE AD SYSTEMS
13554 Adside=1
13557 Ad_ammo=B_ad_ammo
13560 CALL Helo_ammo(Sys_tot(*),Target(*),Adside,Ad_ammo,Ad_helo(1),Basic_ld(*
Ammo_wt(*))
13563 Ad_sv(1)=Ad_helo(1)
13566 Df_ammo(1)=B_df_ammo
13569 Df_ammo(2)=R_df_ammo
13572 Sided=1
13575 FOR I=1 TO 70
13578  H_targ(1,I)=Target(1,I)
13581 NEXT I
13584 CALL Dismount(H_targ(*),B_ld_fact,B_msn(1),Sided,Btl_rg,B_mount,B_dmount
13587 CALL Apport_inf(Target(*),1,B_dmount)
13590 !
13593 CALL Helo_kills(Cell(*),Helo_tgt(*),Ad_helo(*),Terr,Atk_prof(*),Helo_mis
),Day_nite,Time_step,P_def(*),Arty(*),Veh_ada(*),Hnd_ada(*),Stnd_off_rg(*),Vis
13596 !
13599 R_ad_ammo=R_ad_ammo-(Ad_sv(2)-Ad_helo(2))
13602 B_ad_ammo=B_ad_ammo-(Ad_sv(1)-Ad_helo(1))
13605 B_df_ammo=Df_ammo(1)
13608 R_df_ammo=Df_ammo(2)
13611    !STORE KILLS AND SUBTRACT BLUE HELICOPTERS
13614 FOR Side=1 TO 2
13617  FOR I=1 TO 70
13620    Sys_helo(Side,I)=Helo_tgt(Side,1,I)-Helo_tgt(Side,2,I)
13623  NEXT I
13626  IF Side=1 THEN CALL Inf_survive(H_targ(*),1,Sys_helo(*),1,B_ld_fact,Inf
urv(*))
13629  IF Side=2 THEN CALL Inf_survive(H_targ(*),3,Sys_helo(*),2,R_ld_fact,Inf
urv(*))
13632  FOR I=36 TO 40
13635    Sys_helo(Side,I)=Sys_helo(Side,I)+Inf_surv(I-35)
13638  NEXT I
13641 NEXT Side
13644    !REMOVE HELOS
13647 FOR I=1 TO 3
13650  B_helo(I,2)=B_helo(I,2)+Cell(1,1,I)-Cell(1,2,I)
13653  R_helo(I,2)=R_helo(I,2)+Cell(2,1,I)-Cell(2,2,I)
13656 NEXT I
13659    !
13662    !ADJUST BLUE AD AMMO AND DELETE SYSTEMS
13665    !SUBTRACT SYSTEM LOSSES FOR HELOS
13668 FOR I=1 TO 70
13671  Sys_tot(2,I)=Sys_tot(2,I)-Sys_helo(1,I)
13674  Sys_tot(4,I)=Sys_tot(4,I)-Sys_helo(2,I)
13677 NEXT I
13680 No_helo2:!
13683 !
13686 ! ** PGM PORTION OF PHASE II **
```

Table 6-14. Ground combat code (continued).

```
13689 !SET UP CLGP AND GAMP ARRAY FACTORS
13692 FOR I=1 TO 70
13695  Gamp_fact(I)=Red_f_t(I)
13698  Clgp_fact(I)=Red_f_t(I)*2
13701  IF Clgp_fact(I)>1 OR Clgp_rpv=1 THEN Clgp_fact(I)=1
13704 NEXT I
13707 GOSUB Clgp_gamp_atrit
13710 !
13713 ! ** INFANTRY PORTION OF PHASE II **
13716 ! NOTE: RNG_BAND HOLDS THE BAND IN WHICH THE FIGHT HAS BEGUN
13719 !      IF RNG_BAND<=2 THEN, AT LESS THAN 1000M , INFANTRY WILL DISMOUN
13722 IF Rng_band>=2 THEN GOTO Finish2
13725 !
13728 T_conflict=.5  !INFANTRY CONFLICT TIME IN HOURS
13731 GOSUB Infantry_cbt
13734 !
13737 Finish2: !
13740 GOSUB Updatek_v
13743 End_phase2_btl:RETURN  ! ** END BATTLE PHASE II **
13746 !
13749 !-----------------------------------------------------------------------
13752 !
13755 Phase3_btl: ! THIS SUBROUTINE CONDUCTS ATTRITION ASSESSMENTS FOR
13758             ! PHASE III (WITHDRAWAL)
13761 !
13764 !
13767 Phase_ct(3)=Phase_ct(3)+1    !COUNT # OF 30 MINUTE SEGMENTS IN PHASE II
13770                               !(SHOULD ONLY BE ONE)
13773 Btl_phase=3
13776 GOSUB Phase_int   !ZERO ALL KILL ARRAYS FOR THIS 30 MIN.
13779 !
13782 ! ** NO MINES IN PHASE III **
13785 !
13788 ! MINES ADDED TO PHASE III ! ROB
13791 IF Mine_hit=0 THEN GOTO 13803
13794 IF Minefield(Mine_hit,6)=1 THEN GOTO 13803
13797 GOSUB Run_mine
13800 ! ** ARTILLERY PORTION OF PHASE III **
13803 !  CHECK ON INDIRECT FIRE USED IN THIS PHASE
13806 Blue_aty=0  !BLUE FIRE FLAG
13809 Red_aty=0   !RED  FIRE FLAG
13812 FOR I=1 TO 15
13815  FOR J=1 TO 5
13818    IF Bif_fired(I,J)>0 THEN Blue_aty=1
13821    IF Rif_fired(I,J)>0 THEN Red_aty=1
13824  NEXT J
13827 NEXT I
13830 IF Blue_aty=0 AND Red_aty=0 THEN GOTO Direct_3
13833 GOSUB Arty_sub
13836 !
13839 Direct_3: ! ** DIRECT FIRE PORTION OF PHASE III **
13842 FOR I=1 TO 2
```

Table 6-14. Ground combat code (continued).

```
13845  FOR J=1 TO 3
13848    Cell(I,2,J)=0           !ZERO OUT RED & BLUE HELO'S CELLS
13851  NEXT J
13854 NEXT I
13857 Cell(1,1,1)=Bah1           !LOAD UP CELLS
13860 Cell(1,1,2)=Bah2
13863 Cell(1,1,3)=Bsct
13866 Cell(2,1,1)=Rah1
13869 Cell(2,1,2)=Rah2
13872 Cell(2,1,3)=Rsct
13875 CALL Helo_range(Cell(*),Helo_mis(*),Stnd_off_rg(*))  !CALCULATE RANGES
13878    !TEST FOR ENTERING PHASE II
13881 IF Phase_ct(2)=0 THEN GOTO Helos3    !NO PHASE II, NO DIRECT FIRE PULLC
13884    !
13887 GOSUB Collect_trucks    !TEMPORARY CODE FOR HTLD/ADEA ONLY
13890 FOR I=1 TO 70
13893  SELECT Break_point
13896  CASE 1        !BLUE BREAK
13899    Blue_fct=B_f(B_ms,I)*.9        !SET BLUE FIRERS TO 90%
13902    Red_fct=R_f(R_ms,I)+Phase_ct(2)*R_df(R_ms,I)
13905    IF Red_fct<.05 THEN Red_fct=.05
13908  CASE 2        !RED BREAK
13911    Blue_fct=B_f(B_ms,I)+Phase_ct(2)*B_df(B_ms,I)
13914    Red_fct=R_f(R_ms,I)*.9         !SET RED FIRERS TO 90%
13917    IF Blue_fct<.05 THEN Blue_fct=.05
13920  END SELECT
13923  IF Red_fct>1 THEN Red_fct=1
13926  IF Blue_fct>1 THEN Blue_fct=1
13929  Sys(5,I)=Sys_tot(2,I)*Bdf_mask(Balb,I)*Blue_fct      !SET BLUE FIRERS
13932  Sys(6,I)=Sys_tot(4,I)*Rdf_mask(Ralb,I)*Red_fct       !SET RED FIRERS
13935      !
13938      !CALCULATE SYSTEMS WHICH ARE TARGETS
13941  SELECT Break_point
13944  CASE 1          !BLUE BREAK
13947    Blue_f_t(I)=B_df_t(B_ms,I)*.9
13950    Red_f_t(I)=R_df_t(R_ms,I)+Phase_ct(2)*R_df_dt(R_ms.I)
13953  CASE 2          !RED BREAK
13956    Blue_f_t(I)=B_df_t(B_ms,I)+Phase_ct(2)*B_df_dt(B_ms,I)
13959    Red_f_t(I)=R_df_t(R_ms,I)*.9
13962  END SELECT
13965  IF Red_f_t(I)>1 THEN Red_f_t(I)=1
13968  IF Blue_f_t(I)>1 THEN Blue_f_t(I)=1
13971  IF Red_f_t(I)<.05 THEN Red_f_t(I)=.05
13974  IF Blue_f_t(I)<.05 THEN Blue_f_t(I)=.05
13977  Sys(1,I)=Sys_tot(2,I)*Blue_f_t(I)*Bdf_mask(Balb,I)
13980  Sys(3,I)=Sys_tot(4,I)*Red_f_t(I)*Rdf_mask(Ralb,I)
13983      !CALCULATE SYSTEM VULNERABILITIES
13986  SELECT Break_point
13989  CASE 1           !BLUE BREAK
13992    Red_fct=R_v(R_ms,I)+R_dv(R_ms,I)*Phase_ct(2)
13995    Blue_fct=B_v(B_ms,I)*.9
13998  CASE 2           !RED BREAK
```

Table 6-14. Ground combat code (continued).

```
14001     Red_fct=R_v(R_ms,I)*.9
14004     Blue_fct=B_v(B_ms,I)+B_dv(B_ms,I)*Phase_ct(2)
14007   END SELECT
14010   IF Red_fct>1 THEN Red_fct=1
14013   IF Blue_fct>1 THEN Blue_fct=1
14016   IF Red_fct<.05 THEN Red_fct=.05
14019   IF Blue_fct<.05 THEN Blue_fct=.05
14022   Red_vul(I)=Red_fct
14025   Blue_vul(I)=Blue_fct
14028 NEXT I
14031 IF Break_point<>1 THEN Sys(1,5)=0    !TEMPORARY CODE FOR HTLD/ADEA ONLY
14034   !SET UP GROUND ATTRITION PARAMETERS
14037 Btl_phase=3
14040 Rng_band=Cur_bnd
14043 SELECT Break_point        !SET BREAK TIME
14046 CASE 1         !BLUE BREAK
14049   Num_bands=B_break_t(B_ms)/30
14052 CASE 2         !RED BREAK
14055   Num_bands=R_break_t(R_ms)/30
14058 END SELECT
14061 IF Cur_bnd<=0 THEN Rng_band=1
14064 Terrain=B_terr     !FIGHT ON DEFENDERS TERRAIN
14067 IF Atk_def=1 THEN Terrain=R_terr
14070 GOSUB Df_cbt     !PERFORM DIRECT FIRE COMBAT
14073 !
14076 Helos3: ! ** HELICOPTER ATTRITION FOR PHASE III **
14079 IF Bah1+Bah2+Bsct<=0 AND Rah1+Rah2+Rsct<=0 THEN GOTO No_helo3
14082 Arty(1)=2
14085 Arty(2)=2
14088 Veh_ada(1)=Bveh_sup       !SUPPRESSION OF VEHICULAR & HAND-HELD ADA
14091 Veh_ada(2)=Rveh_sup
14094 Hnd_ada(1)=Bhnd_sup
14097 Hnd_ada(2)=Rhnd_sup
14100 Time_step=15
14103   !BLUE HELOS -- PREPARE FORCES
14106 FOR I=1 TO 70
14109   IF Phase_ct(2)>0 THEN
14112     Target(1,I)=Sys_tot(4,I)*Red_f_t(I)*Rdf_mask(Ralb,I)
14115     P_def(2,I)=Red_vul(I)
14118   ELSE
14121     Target(1,I)=Sys_tot(4,I)*R_df_t(R_ms,I)
14124     SELECT Break_point
14127     CASE 1         !BLUE BREAK; RED VULNERABILITY NOT AFFECTED
14130       P_def(2,I)=R_v(R_ms,I)+Del_red*R_dv(R_ms,I)
14133       IF P_def(2,I)>1 THEN P_def(2,I)=1
14136       IF P_def(2,I)<.05 THEN P_def(2,I)=.05
14139     CASE 2         !RED BREAK
14142       P_def(2,I)=R_v(R_ms,I)*.9
14145       IF P_def(2,I)>1 THEN P_def(2,I)=1
14148       IF P_def(2,I)<.05 THEN P_def(2,I)=.05
14151     END SELECT
14154   END IF
```

Table 6-14. Ground combat code (continued).

```
14157  Target(2,I)=0
14160  Helo_tgt(2,1,I)=Target(1,I)     !THIS ARRAY IS PASSED TO HELO_KILLS
14163  Helo_tgt(2,2,I)=0
14166 NEXT I
14169   !APPORTION RED AD AMMO BASED ON RED TARGETS
14172   ! RED AD SYSTEMS
14175 Adside=2
14178 Ad_ammo=R_ad_ammo
14181 CALL Helo_ammo(Sys_tot(*),Target(*),Adside,Ad_ammo,Ad_helo(2),Basic_ld(*
Ammo_wt(*))
14184 Ad_sv(1)=Ad_helo(2)
14187 Sided=2
14190 FOR I=1 TO 70
14193  H_targ(3,I)=Target(1,I)
14196 NEXT I
14199 CALL Dismount(H_targ(*),R_ld_fact,R_msn(1),Sided,Btl_rg,R_mount,R_dmount
14202 CALL Apport_inf(Target(*),1,R_dmount)
14205 !---PREPARE BLUE FORCES
14208 FOR I=1 TO 70
14211  IF Phase_ct(2)>0 THEN
14214    Target(1,I)=Sys_tot(2,I)*Blue_f_t(I)*Bdf_mask(Balb,I)
14217    P_def(1,I)=Blue_vul(I)
14220  ELSE
14223    Target(1,I)=Sys_tot(2,I)*B_df_t(B_ms,I)
14226    SELECT Break_point
14229    CASE 1        !BLUE BREAK; BLUE VULNERABILITY AFFECTED
14232      P_def(1,I)=B_v(B_ms,I)*.9
14235      IF P_def(1,I)>1 THEN P_def(1,I)=1
14238      IF P_def(1,I)<.05 THEN P_def(1,I)=.05
14241    CASE 2        !RED BREAK; BLUE NOT AFFECTED
14244      P_def(1,I)=B_v(B_ms,I)+Del_blue*B_dv(B_ms,I)
14247      IF P_def(1,I)>1 THEN P_def(1,I)=1
14250      IF P_def(1,I)<.05 THEN P_def(1,I)=.05
14253    END SELECT
14256  END IF
14259  Target(2,I)=0
14262  Helo_tgt(1,1,I)=Target(1,I)     !THIS ARRAY IS PASSED TO HELO_KILLS
14265  Helo_tgt(1,2,I)=0
14268 NEXT I
14271   !APPORTION AD_AMMO TO BLUE AD SYSTEMS
14274 Adside=1
14277 Ad_ammo=B_ad_ammo
14280 CALL Helo_ammo(Sys_tot(*),Target(*),Adside,Ad_ammo,Ad_helo(1),Basic_ld(*
Ammo_wt(*))
14283 Ad_sv(1)=Ad_helo(1)
14286 Df_ammo(1)=B_df_ammo
14289 Df_ammo(2)=R_df_ammo
14292 Sided=1
14295 FOR I=1 TO 70
14298  H_targ(1,I)=Target(1,I)
14301 NEXT I
14304 CALL Dismount(H_targ(*),B_ld_fact,B_msn(1),Sided,Btl_rg,B_mount,B_dmount
```

Table 6-14. Ground combat code (continued).

```
14307 CALL Apport_inf(Target(*),1,B_dmount)
14310 !
14313 CALL Helo_kills(Cell(*),Helo_tgt(*),Ad_helo(*),Terr,Atk_prof(*),Helo_mis
),Day_nite,Time_step,P_def(*),Arty(*),Veh_ada(*),Hnd_ada(*),Stnd_off_rg(*),Vis
14316 !
14319 B_ad_ammo=B_ad_ammo-(Ad_sv(1)-Ad_helo(1))
14322 R_ad_ammo=R_ad_ammo-(Ad_sv(2)-Ad_helo(2))
14325 B_df_ammo=Df_ammo(1)
14328 R_df_ammo=Df_ammo(2)
14331    !STORE  KILLS AND SUBTRACT BLUE HELICOPTERS
14334 FOR Side=1 TO 2
14337  FOR I=1 TO 70
14340    Sys_helo(2,I)=Helo_tgt(Side,1,I)-Helo_tgt(Side,2,I)
14343  NEXT I
14346  IF Side=1 THEN CALL Inf_survive(H_targ(*),1,Sys_helo(*),1,B_ld_fact,Inf
urv(*))
14349  IF Side=2 THEN CALL Inf_survive(H_targ(*),3,Sys_helo(*),2,R_ld_fact,Inf
urv(*))
14352  FOR I=36 TO 40
14355    Sys_helo(Side,I)=Sys_helo(Side,I)+Inf_surv(I-35)
14358  NEXT I
14361 NEXT Side
14364    !REMOVE HELOS
14367 FOR I=1 TO 3
14370  B_helo(I,2)=B_helo(I,2)+Cell(1,1,I)-Cell(1,2,I)
14373  R_helo(I,2)=R_helo(I,2)+Cell(2,1,I)-Cell(2,2,I)
14376 NEXT I
14379 !
14382    !ADJUST BLUE AD AMMO AND DELETE SYSTEMS
14385    !SUBTRACT SYSTEM LOSSES FOR HELOS
14388 FOR I=1 TO 70
14391  Sys_tot(2,I)=Sys_tot(2,I)-Sys_helo(1,I)
14394  Sys_tot(4,I)=Sys_tot(4,I)-Sys_helo(2,I)
14397 NEXT I
14400 No_helo3:!
14403    ! ** PGM PORTION OF PHASE III **
14406    !IF BLUE IS BREAKING THEN ONLY GAMP
14409 FOR I=1 TO 70
14412  Gamp_fact(I)=Red_f_t(I)
14415  Clgp_fact(I)=Red_f_t(I)
14418  IF Break_point=1 AND Clgp_rpv=0 THEN Clgp_fact(I)=0      !BLUE BREAKS W
14421                                                           ! NO RPV'S
14424 NEXT I
14427 GOSUB Clgp_gamp_atrit
14430    ! ** INFANTRY PORTION OF PHASE III **
14433 IF Phase_ct(2)=0 THEN GOTO Finish3
14436    !If RNG_BAND<2 then infantry will play
14439 IF Rng_band>=2 THEN GOTO Finish3
14442 T_conflict=Time_step/60     !SET TIME IN HOUR UNITS
14445 GOSUB Infantry_cbt
14448    !
14451 Finish3: !
```

Table 6-14. Ground combat code (continued).

```
14454 GOSUB Updatek_v
14457 End_phase3_btl:RETURN    ! ** END BATTLE PHASE III **
14460 !
14463 !------------------------------------------------------------------------
14466 !
14469 Phase_int: !
14472 FOR I=1 TO 2
14475  FOR K=1 TO 70
14478    Sys_direct(I,K)=0          !ZERO DIRECT FIRE KILLS
14481  NEXT K
14484 NEXT I
14487 FOR J=1 TO 70
14490  FOR I=1 TO 2
14493    Sys_helo(I,J)=0
14496    Sys_pgm(I,J)=0
14499    Sys_inf(I,J)=0
14502  NEXT I
14505  FOR I=1 TO 4
14508    Sys_mine(I,J)=0
14511    Sys_arty(I,J)=0
14514  NEXT I
14517  FOR I=1 TO 6
14520    Sys(I,J)=0
14523  NEXT I
14526 NEXT J
14529 RETURN
14532 !
14535 !------------------------------------------------------------------------
14538 !
14541 Updatek_v:   !  UPDATES THE KV
14544 Hh=Hh+1
14547 FOR J=1 TO 70
14550  Kv_b(1,J)=Kv_b(1,J)+Sys_direct(1,J)
14553  Kv_r(1,J)=Kv_r(1,J)+Sys_direct(2,J)
14556  Kv_b(2,J)=Kv_b(2,J)+Sys_arty(2,J)
14559  Kv_r(2,J)=Kv_r(2,J)+Sys_arty(4,J)
14562  Kv_b(3,J)=Kv_b(3,J)+Sys_pgm(1,J)
14565  Kv_r(3,J)=Kv_r(3,J)+Sys_pgm(2,J)
14568  Kv_b(4,J)=Kv_b(4,J)+Sys_helo(1,J)
14571  Kv_r(4,J)=Kv_r(4,J)+Sys_helo(2,J)
14574  Kv_b(5,J)=Kv_b(5,J)+Sys_inf(1,J)
14577  Kv_r(5,J)=Kv_r(5,J)+Sys_inf(2,J)
14580  Kv_b(6,J)=Kv_b(6,J)+Sys_mine(2,J)
14583  Kv_r(6,J)=Kv_r(6,J)+Sys_mine(4,J)
14586 NEXT J
14589  !
14592 RETURN
14595  !
14598  ! ---------------------------------------------------------------------
14601  !
14604 Run_mine: !
14607  !CALCULATE THE % OF FORCE IN MINEFIELD
```

Table 6-14. Ground combat code (continued).

```
14610 SELECT Atk_def+1
14613 CASE 1     !RED FORCES IN THE MINEFIELD
14616  FOR I=1 TO 70
14619    Sys_mine(3,I)=Sys_tot(4,I)*R_df_t(R_ms,I)*Rf_mask(I)
14622  NEXT I
14625  FOR I=36 TO 40     !SAVE INFANTRY COUNT FOR SUB INF_SURVIVE
14628    R_inf(1,I-35)=Sys_mine(3,I)
14631  NEXT I
14634  Sided=2
14637  CALL Dismount(Sys_mine(*),R_ld_fact,R_msn(1),Sided,Btl_rg,Mounted,Dism
ted)
14640  CALL Apport_inf(Sys_mine(*),3,Dismounted)
14643  CALL Mines(Sys_mine(*),Minefield(*),Mine_hit,Atk_def,Bul_bch,Btl_phase
14646     !
14649     !ELEMENTS LOST DUE TO MINES ARE IN SYS_MINE(4,I)
14652  FOR I=1 TO 70
14655    Sys_tot(4,I)=Sys_tot(4,I)-Sys_mine(4,I)
14658  NEXT I
14661  IF Mounted<>0 THEN
14664    CALL Inf_survive(R_inf(*),0,Sys_mine(*),4,R_ld_fact,Inf_surv(*))
14667    FOR I=36 TO 40
14670      Sys_tot(4,I)=Sys_tot(4,I)-Inf_surv(I-35)
14673      Sys_mine(4,I)=Sys_mine(4,I)+Inf_surv(I-35)
14676    NEXT I
14679  END IF
14682  PRINT
14685  PRINT "   RED  FORCES IN MINEFIELD AT RANGE ";Minefield(Mine_hit,1);":
LAY ";Prnt_mn_dlay;" MINUTES"
14688 CASE 2     !BLUE FORCES IN THE MINEFIELD
14691  FOR I=1 TO 70
14694    Sys_mine(1,I)=Sys_tot(2,I)*B_df_t(B_ms,I)*Bf_mask(I)
14697  NEXT I
14700  FOR I=36 TO 40     !SAVE INFANTRY COUNT BEFORE MOUNTING
14703    B_inf(1,I-35)=Sys_mine(1,I)
14706  NEXT I
14709  Sided=1
14712  CALL Dismount(Sys_mine(*),B_ld_fact,B_msn(1),Sided,Btl_rg,Mounted,Dism
ted)
14715  CALL Apport_inf(Sys_mine(*),1,Dismounted)
14718  CALL Mines(Sys_mine(*),Minefield(*),Mine_hit,Atk_def,Bul_bch,Btl_phase
14721 !
14724  FOR I=1 TO 70
14727    Sys_tot(2,I)=Sys_tot(2,I)-Sys_mine(2,I)
14730  NEXT I
14733  IF Mounted<>0 THEN
14736    CALL Inf_survive(B_inf(*),0,Sys_mine(*),2,B_ld_fact,Inf_surv(*))
14739    FOR I=36 TO 40
14742      Sys_tot(2,I)=Sys_tot(2,I)-Inf_surv(I-35)
14745      Sys_mine(2,I)=Sys_mine(2,I)+Inf_surv(I-35)
14748    NEXT I
14751  END IF
14754  PRINT
```

Table 6-14. Ground combat code (continued).

```
14757  PRINT "   BLUE FORCES IN MINEFIELD AT RANGE ";Minefield(Mine_hit,1),";
LAY ";Prnt_mn_dlay;" MINUTES"
14760 END SELECT
14763   !
14766 RETURN
14769 !
14772 !------------------------------------------------------------------
14775 !
14778 Arty_sub: !
14781 IF Red_aty=0 THEN GOTO Bluearty
14784 Redarty: !
14787 Rarty_fire=Rarty_fire+1     !INCREASE RED ARTY COUNTER
14790 FOR I=1 TO 70
14793  Blue_fct=B_if_t(B_ms,I)+B_if_dt(B_ms,I)*Rarty_fire
14796  IF Blue_fct>1 THEN Blue_fct=1
14799  Sys_arty(1,I)=Sys_tot(2,I)*Blue_fct
14802 NEXT I
14805   !
14808   !CALCULATE RED SYSTEMS TARGETABLE
14811 Bluearty: !
14814 IF Blue_aty=0 THEN Barty_fire=Barty_fire+1     !INCREASE BLUE ARTY COUNTE
14817 FOR I=1 TO 70
14820  Red_fct=R_if_t(R_ms,I)+R_if_dt(R_ms,I)*Barty_fire
14823  IF Red_fct>1 THEN Red_fct=1
14826  Sys_arty(3,I)=Sys_tot(4,I)*Red_fct
14829 NEXT I
14832 FOR I=36 TO 40    !SAVE THE INFANTRY COUNT BEFORE MOUNTING
14835  B_inf(1,I-35)=Sys_arty(1,I)
14838  R_inf(1,I-35)=Sys_arty(3,I)
14841 NEXT I
14844   !
14847   !CALCULATE ARTILLERY LOSSES
14850 B_phase=Btl_phase
14853 R=Ride
14856 Sided=2
14859 CALL Dismount(Sys_arty(*),R_ld_fact,R_msn(1),Sided,Btl_rg,R_mounted,R_di
ounted)
14862 CALL Apport_inf(Sys_arty(*),3,R_dismounted)
14865 Sided=1
14868 CALL Dismount(Sys_arty(*),B_ld_fact,B_msn(1),Sided,Btl_rg,B_mounted,B_di
ounted)
14871 CALL Apport_inf(Sys_arty(*),1,B_dismounted)
14874 CALL Arty_atrit(Sys_arty(*),B_msn(B_ms),R_msn(R_ms),Bif_fired(*),Rif_fir
(*),T_length(*),T_width(*),Barty_fire,Rarty_fire,B_phase,Atk_def,Sys_tot(*),R)
14877 FOR I=1 TO 70
14880  Sys_tot(2,I)=Sys_tot(2,I)-Sys_arty(2,I)        !SUBTRACT BLUE KILLED
14883  Sys_tot(4,I)=Sys_tot(4,I)-Sys_arty(4,I)        !SUBTRACT RED  KILLED
14886 NEXT I
14889 IF R_mounted<>0 THEN
14892  CALL Inf_survive(R_inf(*),0,Sys_arty(*),4,R_ld_fact,Inf_surv(*))
14895  FOR I=36 TO 40
14898    Sys_arty(4,I)=Sys_arty(4,I)+Inf_surv(I-35)
```

Table 6-14. Ground combat code(continued).

```
14901      Sys_tot(4,I)=Sys_tot(4,I)-Inf_surv(I-35)
14904    NEXT I
14907 END IF
14910 IF B_mounted<>0 THEN
14913    CALL Inf_survive(B_inf(*),0,Sys_arty(*),2,B_ld_fact.Inf_surv(*))
14916    FOR I=36 TO 40
14919      Sys_arty(2,I)=Sys_arty(2,I)+Inf_surv(I-35)
14922      Sys_tot(2,I)=Sys_tot(2,I)-Inf_surv(I-35)
14925    NEXT I
14928 END IF
14931      !
14934 RETURN
14937 !
14940 !-----------------------------------------------------------------
14943 !
14946 Collect_trucks: !  TEMPORARY CODE FOR HTLD/ADEA ONLY
14949 IF Balb<>1 THEN
14952  Sys_tot(2,6)=Sys_tot(2,55)+Sys_tot(2,58)
14955  Pct_fuel_truck=Sys_tot(2,55)/Sys_tot(2,6)
14958 END IF
14961 RETURN            !ENDS TEMPORARY CODE FOR HTLD/ADEA
14964 !
14967 !-----------------------------------------------------------------
14970 !
14973 Df_cbt:  !
14976 PRINT
14979 SELECT Btl_phase
14982 CASE 1
14985  PRINT "   FORWARD FORCES IN CONTACT FOR 15 MINUTE PERIOD"
14988 CASE 2
14991  SELECT Atk_def
14994  CASE 0
14997    B_r_attack$="RED "
15000  CASE 1
15003    B_r_attack$="BLUE"
15006  END SELECT
15009  IF Cur_bnd=Rng_band THEN
15012    PRINT USING "30A,6D";"   STATIONERY FORCES AT RANGE ";Btl_rg
15015  ELSE
15018    PRINT "   BOTH FORCES IN CONTACT FOR 30 MINUTE PERIOD "
15021  END IF
15024 CASE 3
15027  Prnt_rg=B_rg_break
15030  IF Break_point=2 THEN Prnt_rg=R_rg_break
15033           !PRINT USING "54A,6D";"   FORCES DISENGAGING ; DIRECT FIRE OC
RING AT RANGE ";Btl_rg
15036 END SELECT
15039 FOR I=1 TO 20
15042  B_fire_sv(I)=Sys(5,I)
15045  R_fire_sv(I)=Sys(6,I)
15048 NEXT I
15051 CALL Df_ammo(Sys_tot(*),B_ammo(*),R_ammo(*),B df_ammo,R_df_ammo,B_engage
```

Table 6-14. Ground combat code (continued).

```
nts(*),R_engagements(*))
15054 FOR I=1 TO 5
15057  R_inf_save(I)=Sys(3,I+35)
15060  B_inf_save(I)=Sys(1,I+35)
15063 NEXT I
15066 Sided=2
15069 CALL Dismount(Sys(*),R_ld_fact,R_msn(1),Sided,Btl_rg,R_mounted,R_dismoun
d)
15072 CALL Apport_inf(Sys(*),3,R_dismounted)
15075 Sided=1
15078 CALL Dismount(Sys(*),B_ld_fact,B_msn(1),Sided,Btl_rg,B_mounted,B_dismoun
d)
15081 CALL Apport_inf(Sys(*),1,B_dismounted)
15084 T=Turn
15087 Sec=Sector
15090 Bu=No_b_unit
15093 Ru=No_r_unit
15096 St=St_time
15099 CALL Df_attrition(Blue_vul(*),Red_vul(*),Terrain,Day_nite,Vis,Num_bands,
g_band,Atk_def,Sys(*),B_ammo(*),R_ammo(*),B_vis(*),R_vis(*),T,Sec,Bu,Ru,St,Dc)
15102     !
15105 Aportion_trucks:     !  TEMPORARY CODE FOR HTLD/ADEA ONLY
15108 IF Balb<>1 AND Btl_phase<>1 THEN
15111  Sys(2,55)=Sys(2,6)*Pct_fuel_truck
15114  Sys(2,58)=Sys(2,6)-Sys(2,55)
15117  Sys(2,6)=0
15120  Sys_tot(2,6)=0
15123 END IF
15126             ! ENDS TEMPORARY CODE FOR HTLD/ADEA
15129    !CALCULATE LOSSES AND STORE IN SYS_DIRECT
15132 R_ammo_1st=0        !RED AMMO USED
15135 B_ammo_1st=0        !BLUE AMMO USED
15138 FOR I=1 TO 70
15141  Sys_direct(1,I)=Sys(2,I)  !UPDATE BLUE LOSSES
15144  Sys_direct(2,I)=Sys(4,I)  !UPDATE RED  LOSSES
15147  Sys_tot(2,I)=Sys_tot(2,I)-Sys(2,I)
15150  Sys_tot(4,I)=Sys_tot(4,I)-Sys(4,I)
15153 NEXT I
15156 FOR I=1 TO 5
15159  Sys(3,I+35)=R_inf_save(I)
15162  Sys(1,I+35)=B_inf_save(I)
15165 NEXT I
15168 IF B_mounted<>0 THEN
15171  CALL Inf_survive(Sys(*),1,Sys(*),2,B_ld_fact,Inf_surv(*))
15174  FOR I=36 TO 40
15177    Sys_tot(2,I)=Sys_tot(2,I)-Inf_surv(I-35)
15180    Sys_direct(1,I)=Sys_direct(1,I)+Inf_surv(I-35)
15183  NEXT I
15186 END IF
15189 IF R_mounted<>0 THEN
15192  CALL Inf_survive(Sys(*),3,Sys(*),4,R_ld_fact,Inf_surv(*))
15195  FOR I=36 TO 40
```

Table 6-14. Ground combat code (continued).

```
15198    Sys_tot(4,I)=Sys_tot(4,I)-Inf_surv(I-35)
15201    Sys_direct(2,I)=Sys_direct(2,I)+Inf_surv(I-35)
15204  NEXT I
15207 END IF
15210 '
15213 FOR I=1 TO 20
15216  B_ammo_load=Ammo_wt(1,I)*B_engagements(I)
15219  IF B_ammo_load>B_ammo(1,I) THEN B_ammo_load=B_ammo(1,J)
15222  B_ammo_lst=B_ammo_lst+Sys_direct(1,I)*B_ammo_load
15225  IF Sys_direct(1,I)<B_fire_sv(I) THEN
15228    B_ammo_lst=B_ammo_lst+(B_fire_sv(I)-Sys_direct(1,I))*B_ammo(2,I)
15231  END IF
15234  R_ammo_load=Ammo_wt(2,I)*R_engagements(I)
15237  IF R_ammo_load>R_ammo(1,I) THEN R_ammo_load=R_ammo(1,J)
15240  R_ammo_lst=R_ammo_lst+Sys_direct(2,I)*R_ammo_load
15243  IF Sys_direct(2,I)<R_fire_sv(I) THEN
15246    R_ammo_lst=R_ammo_lst+(R_fire_sv(I)-Sys_direct(2,I))*R_ammo(2,I)
15249  END IF
15252    !
15255 NEXT I
15258    !
15261    !UPDATE AMMO, BOTH RED AND BLUE
15264 B_df_ammo=B_df_ammo-B_ammo_lst
15267 R_df_ammo=R_df_ammo-R_ammo_lst
15270    !
15273 IF B_df_ammo<0 THEN B_df_ammo=0
15276 IF R_df_ammo<0 THEN R_df_ammo=0
15279    !
15282 RETURN
15285  !
15288  !------------------------------------------------------------------------
15291 W_smoke: !
15294  ! THIS ROUTINE WILL ESTABLISH DIMINISHES FOR SMOKE SCREEN AND THEN CAL
15297  ' SMOKE EMPLACE(SMKEMP) ROUTINE... THE SMKEMP ROUTINE RETURNS THE
15300  ! VISIBILITY THROUGH THE SCREEN AND THE AMOUNT OF AMMO USED TO EMPLACE
15303  ! THE SCREEN
15306  ! BLUE SMOKE
15309 Iunt=1    !***CHANGES ACCORDING TO HTLD(=1) OR C-SERIES(=2)?????????????
15312 Ielem=1
15315 S_time=B_break_t(1)*1.1
15318 IF S_time>30 THEN S_time=30
15321 FOR Vkp=1 TO 3
15324  B_vis(Vkp)=1
15327 NEXT Vkp
15330 CALL Smkemp(Iunt,Ielem,B_smok_tons(*),T_width(1),S_time,Vis,Btl_rg,B_ms
used(*),B_asmk_used(*),B_vis(1),B_vis(2),B_vis(3),Irh)
15333 FOR I=1 TO 4
15336  B_msmk_used(I)=B_smok_tons(I+7)-B_msmk_left(I)
15339 NEXT I
15342 FOR I=1 TO 7
15345  B_asmk_used(I)=B_smok_tons(I)-B_asmk_left(I)
15348 NEXT I
```

Table 6-14. Ground combat code (continued).

```
15351 !
15354  ! RED SMOKE
15357 Iunt=3 !***3 REPRESENTS THE RED FORCE
15360 Ielem=1
15363 S_time=R_break_t(1)*1.1
15366 IF S_time>30 THEN S_time=30
15369 FOR Vkp=1 TO 3
15372  R_vis(Vkp)=1
15375 NEXT Vkp
15378 CALL Smkemp(Iunt,Ielem,R_smok_tons(*),T_width(2),S_time,Vis,Btl_rg,R_msml
used(*),R_asmk_used(*),R_vis(1),R_vis(2),R_vis(3),Irh)
15381 FOR I=1 TO 4
15384  R_msmk_used(I)=R_smok_tons(I+7)-R_msmk_left(I)
15387 NEXT I
15390 FOR I=1 TO 7
15393  R_asmk_used(I)=R_smok_tons(I)-R_asmk_left(I)
15396 NEXT I
15399 !
15402 FOR Vkp=1 TO 3
15405  Viss=R_vis(Vkp)
15408  IF Viss>B_vis(Vkp) THEN Viss=B_vis(Vkp)
15411  B_vis(Vkp)=Viss
15414  R_vis(Vkp)=Viss
15417 NEXT Vkp
15420 RETURN
15423  !
15426  !-----------------------------------------------------------------------
15429 Infantry_cbt:  !
15432 Force=1  !***CHANGES ACCORDING TO HTLD(=1) OR C-SERIES(=2)?????????????
15435 Bstat(1)=B_msn(B_ms)       !SET BLUE MISSION
15438 Bstat(2)=R_msn(R_ms)       !SET RED  MISSION
15441 Attacker=2-Atk_def         !SET ATTACKER; 1 FOR BLUE, 2 FOR RED
15444 PRINT
15447 SELECT Attacker
15450 CASE 1
15453  B_r_attack$="BLUE"
15456 CASE 2
15459  B_r_attack$="RED"
15462 END SELECT
15465 PRINT "    ";B_r_attack$;" ATTACKER DISMOUNTS AND CONTINUES ATTACK"
15468 Lossblue=0
15471 Lossred=0
15474 Sided=2
15477 CALL Dismount(Sys(*),R_ld_fact,R_msn(1),Sided,Btl_rg,R_mounted,R_dismoun
d)
15480 CALL Apport_inf(Sys(*),3,R_dismounted)
15483 Sided=1
15486 CALL Dismount(Sys(*),B_ld_fact,B_msn(1),Sided,Btl_rg,B_mounted,B_dismour
d)
15489 CALL Apport_inf(Sys(*),1,B_dismounted)
15492 CALL Infantry(Sys(*),Force,Bstat(*),Attacker,T_conflict,Lossblue,Lossrec
15495 !APPORTION OUT LOSSES OVER SMALL ARMS ELEMENTS
```

Table 6-14. Ground combat code (continued).

```
15498 Sum_inf_b=0
15501 Sum_inf_r=0
15504 FOR I=36 TO 47    !SUM UP NO. OF SMALL ARMS
15507  Sum_inf_b=Sum_inf_b+Sys(1,I)
15510  Sum_inf_r=Sum_inf_r+Sys(3,I)
15513 NEXT I
15516 FOR I=36 TO 47    !APPORTION OUT LOSSES IN SMALL ARMS
15519  IF Sum_inf_b>0 THEN Sys_inf(1,I)=(Sys(1,I)/Sum_inf_b)*Lossblue
15522  IF Sum_inf_b<=0 THEN Sys_inf(1,I)=0
15525  IF Sum_inf_r>0 THEN Sys_inf(2,I)=(Sys(3,I)/Sum_inf_r)*Lossred
15528  IF Sum_inf_r<=0 THEN Sys_inf(2,I)=0
15531  Sys_tot(2,I)=Sys_tot(2,I)-Sys_inf(1,I)
15534  Sys_tot(4,I)=Sys_tot(4,I)-Sys_inf(2,I)
15537 NEXT I
15540      !
15543 RETURN
15546 !
15549 !-----------------------------------------------------------------
15552 !
15555 Clgp_gamp_atrit: !
15558 !
15561 IF Clgp_msns<=0 AND Gamp_msns<=0 THEN End_clgp
15564 FOR I=1 TO 2
15567  N_rnds(I)=0
15570  Sens_typ(I)=0
15573  Fir_typ(I)=0
15576 NEXT I
15579 S_clgp: !
15582 IF Clgp_msns<=0 THEN S_gamp
15585 Fir_typ(1)=1
15588 Sens_typ(1)=Clgp_rpv+1       !  SET RPV 1=no 2=yes
15591 N_rnds(1)=Clgp_msns/.11        !  CONVERT TONS TO ROUNDS
15594 S_gamp: !
15597 IF Gamp_msns<=0 THEN Set_pgm_tgt
15600 Fir_typ(2)=1
15603 Sens_typ(2)=0      ! SENSORS ASSUMED IN GAMP DATA
15606 N_rnds(2)=Gamp_msns/.11
15609 Set_pgm_tgt: !
15612   !
15615 FOR I=1 TO 70
15618  C_targ(1,I)=Sys_tot(4,I)*Clgp_fact(I)
15621  C_targ(2,I)=0
15624  C_t(3,I)=C_targ(1,I)
15627  C_t(4,I)=0
15630 NEXT I
15633    !
15636    !CALL DISMOUNT TO DETERMINE NUMBER OF INFANTRY IN CARRIERS
15639 Sided=2
15642 CALL Dismount(C_t(*),R_ld_fact,R_msn(1),Sided,Btl_rg,R_mount,R_dmount)
15645 CALL Apport_inf(C_targ(*),1,R_dmount)
15648 Dust=1     !DUST FACTOR   :  NO DUST
15651 CALL Pgm_atrit(Fir_typ(*),N_rnds(*),Vis,R_terr,Sens_typ(*),Cloud_ht,Dus
```

Table 6-14. Ground combat code (continued).

```
_targ(*))
15654    !STORE CLGP KILLS AND SUBTRACT LOSSES
15657 FOR I=1 TO 70
15660  IF I<36 OR I>40 THEN
15663    Sys_pgm(2,I)=Sys_pgm(2,I)+C_targ(2,I)
15666    Sys_tot(4,I)=Sys_tot(4,I)-C_targ(2,I)
15669  END IF
15672 NEXT I
15675 IF R_mount<>0 THEN
15678  CALL Inf_survive(C_t(*),3,C_targ(*),2,R_ld_fact,Inf_surv(*))
15681  FOR I=36 TO 40
15684    Sys_pgm(2,I)=Sys_pgm(2,I)+Inf_surv(I-35)
15687    Sys_tot(4,I)=Sys_tot(4,I)-Inf_surv(I-35)
15690  NEXT I
15693 END IF
15696    !
15699 End_clgp: !
15702 RETURN
15705  !
15708  !----------------------------------------------------------------------
15711  !
15714 Dump_input: !
15717 IF End_time<2400 THEN
15720  PRINT "LINE  1 : ",Turn,Sector,No_b_unit,No_r_unit,St_time,End_time
15723 ELSE
15726  PRINT "LINE  1 : ",Turn,Sector,No_b_unit,No_r_unit,St_time,End_time-240
15729 END IF
15732 PRINT USING "/,11A,29X,10A";"BLUE UNITS:","RED UNITS:"
15735 FOR I=0 TO 9 STEP 3
15738  FOR J=1 TO 3
15741    PRINT USING Fmtd;B_unit_no(I+J),B_unit_pct(I+J)
15744  NEXT J
15747 Fmtd:IMAGE 3D,1X,3D,3X,#
15750  PRINT USING "7X,#"
15753  FOR J=1 TO 3
15756    PRINT USING Fmtu;R_unit_no(I+J),R_unit_pct(I+J)
15759  NEXT J
15762 Fmtu:IMAGE 3X,3D,1X,3D,#
15765  PRINT USING "/"
15768 NEXT I
15771 PRINT
15774 PRINT
15777 PRINT "          ";"ATTACKER","INIT RG","DF RG","# MINFLDS","MTD/DSMT","
MT INF?"
15780 PRINT "LINE  4 : ";Atk_def,Init_rg,Df_rg,No_minefields,Ride,Dis_inf
15783 PRINT
15786 PRINT "LINE  5 : VISIBILITY, CLOUD HT, AND HUMIDITY: ";Vis,Cloud_ht,Irh
15789 PRINT
15792 PRINT "LINE  6 : DEEP ATTACK #";Ialb
15795 PRINT
15798 PRINT
15801 PRINT "BATTLE PARAMETERS:"
```

Table 6-14. Ground combat code (continued).

```
15804 PRINT
15807 PRINT "LINE  7 : ";B_msn(1)-1,B_terr,B_rg_break,B_pct_fwd,B_mopp,T_lengt
1)/1000,T_width(1)/1000,B_break_t(1),B_cas_break
15810 PRINT
15813 PRINT "LINE  8 : ";R_msn(1)-1,R_terr,R_rg_break,R_pct_fwd,R_mopp,T_lengt
2)/1000,T_width(2)/1000,R_break_t(1),R_cas_break
15816 PRINT
15819 PRINT
15822 PRINT "HELICOPTER DATA:"
15825 PRINT
15828 PRINT "LINE  9 : ";B_helo(1,1),B_helo(2,1),B_helo(3,1),B_helo(1,3),B_hel
2,3),B_helo_atkprof(*),B_helo_delay,B_helo_rg_delay,B_helo_msn(*),B_atk_rg(*)
15831 PRINT
15834 PRINT "LINE 10: ";R_helo(1,1),R_helo(2,1),R_helo(3,1),R_helo(1,3),R_helo
,3),R_helo_atkprof(*),R_helo_delay,R_helo_rg_delay,R_helo_msn(*),R_atk_rg(*)
15837 PRINT
15840 PRINT
15843 PRINT "ARTILLERY DATA:"
15846 PRINT
15849 PRINT "LINE 11: ";Bif_msn(*),B_prep_time-30,No_gamp,Perc_gamp,No_clgp,Pe
_clgp,Clgp_rpv
15852 PRINT
15855 PRINT "LINE 12: ";Rif_msn(*),R_prep_time-30
15858 PRINT
15861 PRINT
15864 PRINT "MINEFIELD DATA:"
15867 PRINT
15870 FOR I=13 TO 15
15873  PRINT "LINE ";I;": ";Minefield(I-12,1),Minefield(I-12,2),Minefield(I-12
),Minefield(I-12,4)
15876 NEXT I
15879 RETURN
15882  !
15885  !------------------------------------------------------------------
15888  !
15891 Close_files:  !
15894 ASSIGN @Unitpath TO *
15897 ASSIGN @Kvpath TO *
15900 ASSIGN @Helopath TO *
15903!ASSIGN @Ammopath TO *
15906 ASSIGN @Advanpath TO *
15909 RETURN
15912 !
15915 !-------------------------------------------------------------------
15918 !
15921 !
15924 Halt: !
15927 END
15930 !
15933 !
15936 !
15939 !*********************************************************************
```

Table 6-14. Ground combat code (continued).

```
**********************************************************************
15942 !
15945 SUB Fwdfp(Sys_tot(*),F_mask(*),At_df,D_pct_fwd,D_fp,Sys_eff(*))
15948  OPTION BASE 1
15951'
15954!SET POINTERS FOR EFFECTIVENESS
15957'
15960  IF At_df=0 THEN
15963    Spt=1 !BLUE DEFENDER
15966    Sef=1
15969  ELSE
15972    Spt=3 !RED DEFENDER
15975    Sef=2
15978  END IF
15981!
15984!CALCULATE INITIAL EFFECTIVENESS
15987!
15990  Init_eff=0
15993  Cur_eff=0
15996!
15999  FOR I=1 TO 10
16002    A=F_mask(I)*D_pct_fwd*Sys_eff(Sef,I)
16005    Init_eff=Init_eff+A*Sys_tot(Spt,I)
16008    Cur_eff=Cur_eff+A*Sys_tot(Spt+1,I)
16011  NEXT I
16014!
16017  D_fp=0
16020  IF Init_eff=0 THEN GOTO Ret
16023  D_fp=Cur_eff/Init_eff
16026 Ret:'
16029 SUBEND
16032 !
16035 !**********************************************************************
16038 !
16041 SUB Df_ammo(Sys_tot(*),B_ammo(*),R_ammo(*),B_df_ammo,R_df_ammo,B_engag
ts(*),R_engagements(*))
16044 !
16047  OPTION BASE 1
16050 !
16053  DIM B_ammo_wt(20),R_ammo_wt(20)
16056 !
16059 !
16062  Run$="BH_"
16065  DIM Disk3$[50]
16068  Disk3$=":9134,704,0"
16071 !
16074 !READ WEIGHT & ENGAGEMENT FILE
16077  ASSIGN @P1 TO Run$&"DFAMO"&Disk3$
16080  '
16083  Red$="RD_"
16086  ASSIGN @P2 TO Red$&"DFAMO"&Disk3$
16089  ENTER @P1,1;B_ammo_wt(*)
```

Table 6-14. Ground combat code (continued).

```
16092  ENTER @P1,2;B_engagements(*)
16095  ENTER @P2,1;R_ammo_wt(*)
16098  ENTER @P2,2;R_engagements(*)
16101  ASSIGN @P1 TO *
16104  ASSIGN @P2 TO *
16107 !
16110 !TOTAL CAPACITY
16113  B_capacity=0
16116  R_capacity=0
16119  FOR I=1 TO 20
16122    B_capacity=B_capacity+Sys_tot(2,I)*B_ammo_wt(I)*B_engagements(I)
16125    R_capacity=R_capacity+Sys_tot(4,I)*R_ammo_wt(I)*R_engagements(I)
16128  NEXT I
16131 !
16134 !RNDS FOR EACH WEAPON IN POUNDS
16137  FOR I=1 TO 20
16140    IF B_capacity<>0 THEN B_ammo(1,I)=B_engagements(I)*B_df_ammo*2000/B_c
acity
16143    IF R_capacity<>0 THEN R_ammo(1,I)=R_engagements(I)*R_df_ammo*2000/R_c
acity
16146  NEXT I
16149  FOR I=1 TO 20
16152    B_ammo(1,I)=B_ammo(1,I)*Sys_tot(2,I)
16155    R_ammo(1,I)=R_ammo(1,I)*Sys_tot(4,I)
16158  NEXT I
16161  FOR I=1 TO 20
16164    IF Sys_tot(2,I)<>0 THEN B_ammo(1,I)=B_ammo(1,I)/Sys_tot(2.I)
16167    IF Sys_tot(4,I)<>0 THEN R_ammo(1,I)=R_ammo(1,I)/Sys_tot(4,I)
16170  NEXT I
16173 !
16176 !WEIGHT IN TONS PER WEAPON
16179  FOR I=1 TO 20
16182    B_ammo(1,I)=B_ammo(1,I)*B_ammo_wt(I)/2000
16185    R_ammo(1,I)=R_ammo(1,I)*R_ammo_wt(I)/2000
16188  NEXT I
16191 !
16194 SUBEND
16197 !
16200 !************************************************************************
16203 !
16206 SUB Ammo(Rndse(*),Wrndse(*),Ad_ammo,Ad_ele(*),Ad_eng(*))
16209 !!!--THIS SUBROUTINE CALCULATES THE NUMBER OF ENGAGEMENTS AVAILABLE FOR
EACH AD ELEMENT BASED ON ITS SHARE OF THE AMMO.
16212  OPTION BASE 1
16215  DIM Wt_ad(7)
16218 !  CALCULATE THE PERCENTAGE OF AMMO DUE TO EACH ELEMENT TYPE
16221  Total=0
16224  FOR I=1 TO 7
16227    Wt_ad(I)=(Ad_ele(I)*(Ad_eng(I,1)+Ad_eng(I,2))*Wrndse(I))/2000
16230    Total=Wt_ad(I)+Total
16233  NEXT I
16236  IF Total<=Ad_ammo THEN 16269
```

Table 6-14. Ground combat code (continued).

```
16239 !COMPUTE SCALE FACTOR
16242  Scle=Ad_ammo/Total
16245 !RESCALE ALL
16248  FOR I=1 TO 7
16251    Wt_ad(I)=Scle*Wt_ad(I)
16254    Ad_eng(I,1)=Ad_eng(I,1)*Scle
16257    Ad_eng(I,2)=Ad_eng(I,2)*Scle
16260  NEXT I
16263  Total=Total*Scle
16266 !NOW SUBTRACT AMMO USED FROM AMOUNT AVAILABLE
16269  Ad_ammo=Ad_ammo-Total
16272 SUBEND
16275 !
16278 !**********************************************************************
16281 !
16284 SUB Ad_pri(Expos(*),N_systems(*),Ad_prior(*))
16287 !!! - THIS SUBROUTINE DISTRIBUTES THE AD FIRERS
16290  OPTION BASE 1
16293  Total=0.
16296  FOR I=1 TO 2
16299    Ad_prior(I)=0
16302    Total=Expos(I)*N_systems(I)+Total
16305  NEXT I
16308  IF Total=0 THEN End_sub
16311  FOR I=1 TO 2
16314    Ad_prior(I)=Expos(I)*N_systems(I)/Total
16317  NEXT I
16320 End_sub:!
16323 SUBEND
16326 !
16329 !**********************************************************************
16332 !
16335 SUB Firedst(P_pref(*),Target(*),Mtgprf(*),P_pep,P_veh,T_targets,Pkmse(*
16338 !!! - THIS SUBROUTINE CALCULATES THE DISTRIBUTION OF FIRE UNDER THE CUF
T TARGET CONFIGURATION. ADJUST THE TARGET ENTRY TO REPRESENT INFANTRY SQUADS
16341  OPTION BASE 1
16344  FOR I=36 TO 47
16347    Target(1,I)=Target(1,I)/8.
16350    Target(2,I)=Target(2,I)/8.
16353  NEXT I
16356 !!! - CALCULATE THE WEIGHTED TARGET SUM AND TARGET SUM
16359  W_sum=0
16362  T_targets=0.
16365  FOR I=1 TO 70
16368    Tgt=Target(2,I)
16371    W_sum=Mtgprf(I)*Tgt*Pkmse(I)+W_sum
16374    T_targets=T_targets+Tgt
16377  NEXT I
16380 !!! - CALCULATE THE WEIGHTS
16383  IF W_sum=0 THEN GOTO S_end
16386  FOR I=1 TO 70
16389    Tgt=Target(2,I)
```

Table 6-14. Ground combat code (continued).

```
16392    P_pref(I)=Mtgprf(I)*Tgt*Pkmse(I)/W_sum
16395  NEXT I
16398  IF T_targets=0 THEN
16401    P_pep=0
16404  ELSE
16407    Tot_init=0
16410    Tot_rem=0
16413    FOR I=36 TO 47
16416      Tot_init=Tot_init+Target(1,I)
16419      Tot_rem=Tot_rem+Target(2,I)
16422    NEXT I
16425    P_pep=(Tot_init-Tot_rem)/T_targets
16428  END IF
16431  P_veh=1.-P_pep
16434  FOR I=36 TO 47
16437    Target(1,I)=Target(1,I)*8.  ! RESTORE TARGET
16440    Target(2,I)=Target(2,I)*8.
16443  NEXT I
16446 S_end: !
16449 SUBEND
16452 !
16455 !**********************************************************************
16458 !
16461 SUB Helo_ammo(Sys_tot(*),Target(*),Side,Ad_ammo,Ad_helo,Basic_ld(*),Ammo
t(*))
16464  OPTION BASE 1
16467  DIM Adc(7)
16470 !
16473  Total=0   !SET TOTAL WEIGHT
16476!CALCULATE POINTER TO TOTAL
16479  Adpt=Side*2  !SET POINTER TO AD SIDE
16482  FOR I=1 TO 7
16485    Adc(I)=Sys_tot(Adpt,I+47)*Basic_ld(Side,I+47)*Ammo_wt(Side,I+47)
16488    Total=Total+Adc(I)
16491  NEXT I
16494  Ad_helo=0
16497  FOR I=1 TO 7
16500    IF Sys_tot(Adpt,I+47)<>0 AND Total<>0 THEN
16503      Ad_helo=Ad_helo+(Adc(I)/Total)*Ad_ammo/Sys_tot(Adpt,I+47)*Target(1,
47)
16506    END IF
16509  NEXT I
16512 SUBEND
16515 !
16518 !**********************************************************************
16521 !
16524 SUB Mines(Sys_mine(*),Minefield(*),Mine_hit,Atk_def,Bul_bch,Phase)
16527 !
16530  OPTION BASE 1
16533 !
16536  COM /Mines/ Mine_frct(4,70)
16539 !
```

Table 6-14. Ground combat code (continued).

```
16542 !INITIALIZE
16545  FOR I=1 TO 70
16548    Sys_mine(2,I)=0
16551    Sys_mine(4,I)=0
16554  NEXT I
16557 !
16560 !CHECKS FOR:
16563  !SECTOR WIDTH=0,MINEFIELD WIDTH=0,FORCE ENTERING=0,OR PREVIOUS ASSESSME.
16566  IF Minefield(Mine_hit,2)=0 OR Minefield(Mine_hit,3)=0 OR Minefield(Mine
it,4)=0 OR Minefield(Mine_hit,6)=1 OR Bul_bch=0 THEN GOTO Rtn
16569 !SET FOR ATTACK POSITION
16572  SELECT Atk_def
16575  CASE 0  !RED IS ENTERING MINEFIELD
16578    Entering=3
16581    Killed=4
16584    Ias=3
16587    IF Bul_bch=2 THEN Ias=4
16590  CASE 1
16593    Entering=1
16596    Killed=2
16599    Ias=1
16602    IF Bul_bch=2 THEN Ias=2
16605  END SELECT
16608 !
16611 !MINEFIELD COVERAGE FRACTION
16614  Mcf=(Minefield(Mine_hit,2)/Minefield(Mine_hit,3))
16617 !
16620  Tot_veh=0
16623  FOR I=1 TO 70
16626    Tot_veh=Tot_veh+Sys_mine(Entering,I)
16629  NEXT I
16632  IF Tot_veh<3 THEN Rtn
16635 !
16638  SELECT Phase
16641  CASE 0,1 !PHASE  1
16644    Col_no=3
16647  CASE ELSE !PHASE 2 OR 3
16650    Col_no=6
16653  END SELECT
16656 !
16659  Columns=Tot_veh/Col_no
16662 !
16665 !CALCULATE LOSSES
16668  FOR I=1 TO 70
16671    Sys_mine(Killed,I)=(Mcf*Sys_mine(Entering,I)/Tot_veh)*Columns*Mine_fr
(Ias,I)
16674  NEXT I
16677 !
16680 !SET ASSESSED FLAG
16683 Rtn: !
16686  Minefield(Mine_hit,6)=1
16689 !
```

Table 6-14. Ground combat code (continued).

```
16692 SUBEND
16695 !
16696 !**********************************************************************
16701 !
16704 SUB Infantry(Sys(*),Force,Cstat(*),Attacker,Hr_conflict,Lossblue,Lossr
16707  OPTION BASE 1
16710  DIM Brate(2),Loss(2),Pers(2),Bstat(2)
16713  DIM Fratio(2),What(2)
16716 !
16719  COM /Infantry/ Convertd(10),Converta(10,10),Fpsb(70,2,2),Fpsr(70,2,2)
16722 !
16725  Bstat(1)=Cstat(1)
16728  Bstat(2)=Cstat(2)
16731  Frac_comtd=1.
16734  IF Attacker=1 THEN
16737    Defender=2
16740  ELSE
16743    Defender=1
16746  END IF
16749 !
16752  Pers(1)=0
16755  Pers(2)=0
16758  FOR I=36 TO 47
16761    Pers(1)=Pers(1)+Sys(1,I)
16764    Pers(2)=Pers(2)+Sys(3,I)
16767  NEXT I
16770 !PERFORM CONVERSION TO SUB MISSION VALUES FOR RATE SUB USAGE
16773  What(Attacker)=Converta(Bstat(Attacker),Bstat(Defender))
16776  Defender=(Attacker MOD 2)+1
16779  What(Defender)=Convertd(Bstat(Defender))
16782 !
16785  Fratio(1)=0
16788  Fratio(2)=0
16791  !COMPUTE BLUE (1) AND RED (2) UNADJUSTED FIREPOWER SCORES
16794  FOR I=1 TO 70
16797    Fratio(1)=Fratio(1)+(Sys(1,I)*Fpsb(I,Force,Attacker))
16800    Fratio(2)=Fratio(2)+(Sys(3,I)*Fpsr(I,Force,Attacker))
16803  NEXT I
16806  !DETERMINE WHO IS ATTACKER/DEFENDER;GET APPROPRIATE FORCE MULTIPLIER
16809  !BASED ON MISSIONS OF EACH
16812  IF Attacker=1 THEN
16815    Top=Fratio(1)
16818    Bottom=Fratio(2)
16821  ELSE
16824    Top=Fratio(2)
16827    Bottom=Fratio(1)
16830  END IF
16833  SELECT What(Defender)
16836  CASE 21
16839    Dmultiply=1.0
16842  CASE 22
16845    Dmultiply=1.0
```

Table 6-14. Ground combat code (continued).

```
16848  CASE 23
16851    Dmultiply=.5
16854  CASE 24
16857    Dmultiply=2.0
16860  CASE 25
16863    Dmultiply=1.5
16866  CASE 26
16869    Dmultiply=1.2
16872  CASE 27
16875    Dmultiply=4.5
16878  END SELECT
16881  SELECT Bstat(Attacker)
16884  CASE 2
16887    Amultiply=1.5
16890  CASE 4
16893    SELECT Bstat(Defender)
16896    CASE 2
16899      Amultiply=1.5
16902    CASE 6
16905      Amultiply=1.5
16908    CASE 9
16911      Amultiply=2.
16914    CASE ELSE
16917      Amultiply=1
16920    END SELECT
16923  CASE 5
16926    SELECT Bstat(Defender)
16929    CASE 2
16932      Amultiply=1.5
16935    CASE 6
16938      Amultiply=1.5
16941    CASE 7
16944      Amultiply=1.5
16947    CASE 8
16950      Amultiply=1.3
16953    CASE 9
16956      Amultiply=2.0
16959    CASE ELSE
16962      Amultiply=1.
16965    END SELECT
16968  CASE 7
16971    PRINTER IS 1
16974    PRINT "ATTACKER HAS BEEN GIVEN DEFENSIVE MISSION."
16977    PRINT "INFANTRY ROUTINE WILL NOT ACCEPT THIS CASE."
16980    Subend$="SUBEND"
16983  CASE 8
16986    PRINTER IS 1
16989    PRINT "ATTACKER HAS BEEN GIVEN DEFENSIVE MISSION."
16992    PRINT "INFANTRY ROUTINE WILL NOT ACCEPT THIS CASE."
16995    Subend$="SUBEND"
16998  CASE 10
17001    Amultiply=4.5
```

Table 6-14. Ground combat code (continued).

```
17004  CASE ELSE
17007    Amultiply=1.
17010  END SELECT
17013  IF Subend$="SUBEND" THEN GOTO Subend
17016   ! COMPUTE FIREPOWER RATIO
17019  IF Amultiply=0 OR Dmultiply=0 THEN
17022    PRINTER IS 1
17025    PRINT "ERROR IN INFANTRY SUBROUTINE: FORCE MULTIPLIER IS 0"
17028  END IF
17031  Top=Top*Amultiply
17034  Bottom=Bottom*Dmultiply
17037  IF Bstat(Defender)=10 THEN
17040    X=Top
17043    Top=Bottom
17046    Bottom=X
17049  END IF
17052  Fpr=Top/Bottom
17055  GOSUB Losses
17058  Lossblue=Loss(1)
17061  Lossred=Loss(2)
17064  GOTO Subend
17067 !
17070 !-------------------------------------------------------------------
17073 !
17076 Losses:   !
17079  FOR I=1 TO 2
17082    GOSUB Rate
17085    X=Pers(I)*Frac_comtd*Brate(I)*Hr_conflict
17088    Loss(I)=X
17091  NEXT I
17094  RETURN
17097 !
17100 !-------------------------------------------------------------------
17103 !
17106 Rate:!
17109  SELECT What(I)
17112  CASE 11
17115    Brate(I)=.0384*(Fpr^(-.2383))
17118  CASE 12
17121    Brate(I)=.0384*(Fpr^(-.2383))
17124  CASE 13
17127    Brate(I)=.0384*(Fpr^(-.2383))
17130  CASE 14
17133    Brate(I)=.0483*(Fpr^(-.251))
17136  CASE 15
17139    Brate(I)=.0483*(Fpr^(-.251))
17142  CASE 16
17145    Brate(I)=.0401*(Fpr^(-.237))
17148  CASE 21
17151    Brate(I)=.0125714+(Fpr*.0005)+(.001143*(Fpr*Fpr))
17154  CASE 22
17157    Brate(I)=.003286+(.0034286*Fpr)
```

Table 6-14. Ground combat code (continued).

```
17160  CASE 23
17163    Brate(I)=.003286+(.0034286*Fpr)
17166  CASE 24
17169    Brate(I)=.00919+(.004085*Fpr)+(.000097*(Fpr*Fpr))
17172  CASE 25
17175    Brate(I)=.00919+(.004085*Fpr)+(.000097*(Fpr*Fpr))
17178  CASE 26
17181    Brate(I)=.012714+(.0005*Fpr)+(.001*(Fpr*Fpr))
17184  CASE 27
17187    Brate(I)=.0384*(Fpr^(-.2383))
17190  CASE ELSE
17193    PRINTER IS 1
17196    PRINT "BAD What VALUE IN SUBROUTINE Rate"
17199    STOP
17202  END SELECT
17205  RETURN
17208 Subend:SUBEND
17211 !
17214 !***********************************************************************
17217 !
17220 SUB Dismount(Sys(*),Load_factor,Mission,Side,Range,Mounted,Dismounted)
17223 ! PURPOSE TO CALCULATE THE NUMBER OF ELEMENTS MOUNTED AND DISMOUNTED
17226  OPTION BASE 1
17229  Side_pt=2*Side-1
17232  Sum_inf=0
17235  FOR I=36 TO 40
17238    Sum_inf=Sum_inf+Sys(Side_pt,I)    !TOTAL INFANTRY FOR DF
17241  NEXT I
17244  IF (Mission=4 AND Range<600) OR (Mission>=6 AND Mission<=10) THEN
17247    Mounted=0
17250    Dismounted=Sum_inf
17253  ELSE
17256    Sum_df=Sys(Side_pt,16)+Sys(Side_pt,17)+Sys(Side_pt,18)+Sys(Side_pt,19
Sys(Side_pt,20)
17259    CALL Load_infantry(Sys(*),Mission,Side_pt,Sum_inf,Sum_df,Load_factor)
17262    Mounted=Sum_df*Load_factor
17265    Dismounted=Sum_inf-Mounted
17268  END IF
17271 SUBEND
17274 !
17277 !***********************************************************************
17280 !
17283 SUB Load_infantry(Sys(*),Mission,Side_pt,Sum_inf,Sum_df,Load_factor)
17286 !
17289  OPTION BASE 1
17292  SELECT Mission
17295  CASE 1 TO 6
17298    IF Sum_df=0 THEN
17301      Load_factor=0
17304      GOTO End_rtn
17307    END IF
17310    Load_factor=Sum_inf/Sum_df
```

Table 6-14. Ground combat code (continued).

```
17313    IF Load_factor>8 THEN Load_factor=8
17316    GOTO End_rtn
17319  CASE 7 TO 10
17322    Load_factor=0
17325 End_rtn: !
17328  END SELECT
17331 SUBEND
17334 !
17337 !**********************************************************************
17340 SUB Apport_inf(Target(*),Position,Dismount)
17343  Tot_targ=0
17346  FOR I=36 TO 40  !TOTAL UP INFANTRY FOR DF CARRIER
17349    Tot_targ=Tot_targ+Target(Position,I)
17352  NEXT I
17355  FOR I=36 TO 40  !APPORTION INFANTRY
17358    IF Tot_targ=0 THEN
17361      Target(Position,I)=0
17364    ELSE
17367      Target(Position,I)=(Target(Position,I)/Tot_targ)*Dismount
17370    END IF
17373  NEXT I
17376 SUBEND
17379 !**********************************************************************
17382 SUB Inf_survive(Sys(*),Pos_init,Sys_rem(*),Pos_rem,Load_factor,Inf_surv
)
17385 !
17388  Sum_df=0
17391  FOR I=16 TO 20   !SUM INFANTRY CARRIERS THAT SURVIVED
17394    Sum_df=Sum_df+Sys_rem(Pos_rem,I)
17397  NEXT I
17400  Mount_inf=Load_factor*Sum_df  !NO. OF MOUNTED INFANTRY THAT SURVIVED
17403 !
17406  Sum_inf=0
17409  FOR I=1 TO 5   !SUM INITIAL NO. OF INFANTRY
17412    IF Pos_init>0 THEN Sum_inf=Sum_inf+Sys(Pos_init,I+35)
17415    IF Pos_init=0 THEN Sum_inf=Sum_inf+Sys(1,I)
17418  NEXT I
17421 !
17424  FOR I=1 TO 5    !APPORTION MOUNTED INFANTRY THAT SURVIVED
17427    IF Sum_inf=0 THEN
17430      Inf_surv(I)=0
17433    ELSE
17436      IF Pos_init>0 THEN Inf_surv(I)=(Sys(Pos_init,I+35)/Sum_inf)*Mount_
17439      IF Pos_init=0 THEN Inf_surv(I)=(Sys(1,I)/Sum_inf)*Mount_inf
17442    END IF
17445  NEXT I
17448 SUBEND
17451 !**********************************************************************
17454 !
17457 SUB Arty_atrit(Sys_arty(*),B_msn,R_msn,Bif_fired(*),Rif_fired(*),T_leng
*),T_width(*),B_afire,R_afire,B_phase,Atk_def,Sys_tot(*),R)
17460  OPTION BASE 1
```

Table 6-14. Ground combat code (continued).

```
17463!
17466  DIM Al(15,72),F_d(5,72),Ps(72),Del_par(15,10)
17469  DIM Area(5),Length(5),Width(5),Disk3$[50]
17472  DIM Vollies(15,5)
17475!
17478  COM /Arty/ B_area_band(5),R_area_band(5),B_disprsn_mask(3,10),R_disprsn_m
ask(3,10),B_tgt_mask(5,72),R_tgt_mask(5,72),B_rd_wt(15),R_rd_wt(15)
17481  COM /Arty/ B_psnl_posture(2,2),R_psnl_posture(2,2),Tle(5)
17484!
17487  Barty_fire=B_afire
17490  Rarty_fire=R_afire
17493  IF Barty_fire>4 THEN Barty_fire=4
17496  IF Rarty_fire>4 THEN Rarty_fire=4
17499!
17502! BLUE firing on RED
17505  FOR Task=1 TO 5
17508    Increase_radius=0
17511 ! Calc tgt area
17514 ! this reduces total area to only that area task is targeted against
17517    Area(Task)=T_length(2)*T_width(2)*R_area_band(Task)
17520 ! now handle dispersion
17523    IF Barty_fire>1 THEN
17526 ! IF THE TARGET IS RECEIVING FIRE, IT CAN DISPERSE DEPENDING ON MSN&PHAS
17529      IF R_disprsn_mask(B_phase,R_msn)=1 THEN
17532        Increase_radius=200*(Barty_fire-1)      ! 200 DISMT MOVE/30 MIN
17535        IF R=0 AND Atk_def=0 THEN
17538          Increase_radius=1000*(Barty_fire-1)   ! 1000 MOUNT MOVE/30 MIN
17541        END IF
17544        Radius=SQR(Area(Task)/PI)
17547        Area(Task)=PI*(Radius+Increase_radius)^2
17550      END IF
17553    END IF
17556 Calc_len_r: !
17559    Length(Task)=SQR(Area(Task))
17562    Width(Task)=Length(Task)
17565  NEXT Task
17568 !
17571 ! LOAD LETHAL AREAS
17574 !        CHANGE BARTLA$ AND DISK3$ TO READ NEW FILENAMES AND WINCHESTER
17577  Bartla$="BH_R_LA"
17580  Disk3$=":9134,704,0"
17583  ASSIGN @Areafile TO Bartla$&Disk3$
17586  ENTER @Areafile;Al(*)
17589 ! CONVERT TONS OF AMMO TO VOLLIES TO FIRE
17592  FOR Type=1 TO 15
17595    FOR Task=1 TO 5
17598      IF Type>=12 AND Type<=15 THEN   !MLRS
17601        Tubes_per_vol=1
17604      ELSE
17607        Tubes_per_vol=6
17610      END IF
17613      Throw_wt=Tubes_per_vol*B_rd_wt(Type)
```

Table 6-14. Ground combat code (continued).

```
17616       IF Sys_tot(2,Type+20)>0 THEN
17619         Vollies(Type,Task)=Bif_fired(Type,Task)/Throw_wt
17622       ELSE
17625         Vollies(Type,Task)=0   ! number of Btry/plt vollies or lchr loac
17628       END IF
17631     NEXT Task
17634   NEXT Type
17637 !
17640   Bfd$="FIDEL_H"
17643   Disk3$=":9134,704,0"
17646   ASSIGN @Readblue TO Bfd$&Disk3$
17649   ENTER @Readblue;Del_par(*)
17652 !
17655 ! calculate fd & total if losses
17658 !
17661 ! THIS CHANGE TO DEL_PAR(1,3) REFLECTS TWO CANNON ARTILLERY SYSTEM BEIN
17664 ! PLAYED FOR BLUE(155MM AND 8") ONLY ONE USED IN HTLD STUDY
17667 !
17670   FOR Element=1 TO 35
17673     Ps(Element)=1
17676     GOSUB Compute
17679 ! this computes total losses due to indirect fire
17682     Sys_arty(4,Element)=(1-Ps(Element))*Sys_arty(3,Element)
17685   NEXT Element
17688 ! this allows 2 postures for infantry
17691   Element=71  ! standing
17694   Ps(71)=1
17697   GOSUB Compute
17700 !
17703   Element=72  ! CROUCH
17706   Ps(72)=1
17709   GOSUB Compute
17712 !
17715   Attacker=Atk_def+1
17718   IF Barty_fire>1 THEN
17721     Posture=B_psnl_posture(Attacker,2)
17724   ELSE
17727     Posture=B_psnl_posture(Attacker,1)
17730   END IF
17733 !
17736   Ps_inf=(1-Posture)*Ps(71)+Posture*Ps(72)
17739   FOR Element=36 TO 47
17742     Ps(Element)=Ps_inf
17745     Sys_arty(4,Element)=(1-Ps(Element))*Sys_arty(3.Element)
17748   NEXT Element
17751   FOR Element=48 TO 70
17754     Ps(Element)=1
17757     GOSUB Compute
17760     Sys_arty(4,Element)=(1-Ps(Element))*Sys_arty(3,Element)
17763   NEXT Element
17766!
17769!-------------------------------------------------------------
```

Table 6-14. Ground combat code (continued).

```
17772!
17775! RED firing on BLUE
17778 FOR Task=1 TO 5
17781   Increase_radius=0
17784 ! Calc tgt area
17787 ! this reduces total area to only that area task is targeted against
17790 !
17793   Area(Task)=T_length(1)*T_width(1)*B_area_band(Task)
17796 ! now handle dispersion
17799   IF Rarty_fire>1 THEN
17802 ! IF THE TARGET IS RECEIVING FIRE, IT CAN DISPERSE DEPENDING ON MSN&PHA
17805     IF B_disprsn_mask(B_phase,B_msn)=1 THEN
17808       Increase_radius=200*(Rarty_fire-1)
17811       IF R=0 AND Atk_def=1 THEN
17814         Increase_radius=1200*(Rarty_fire-1)
17817       END IF
17820       Radius=SQR(Area(Task)/PI)
17823       Area(Task)=PI*(Radius+Increase_radius)^2
17826     END IF
17829   END IF
17832 Calc_len_b:   !
17835   Length(Task)=SQR(Area(Task))
17838   Width(Task)=Length(Task)
17841 NEXT Task
17844 ! LOAD LETHAL AREAS
17847 Rartla$="R_H_LA"
17850 Disk3$=":9134,704,0"
17853 ASSIGN @Areafile TO Rartla$&Disk3$
17856 !
17859 ENTER @Areafile;Al(*)
17862 ! CONVERT TONS OF AMMO TO VOLLIES TO FIRE
17865 !
17868 FOR Type=1 TO 15
17871   FOR Task=1 TO 5
17874     IF Type>=12 AND Type<=15 THEN    !MLRS
17877       Tubes_per_vol=1
17880     ELSE
17883       Tubes_per_vol=6
17886     END IF
17889     Throw_wt=Tubes_per_vol*R_rd_wt(Type)
17892     IF Sys_tot(4,Type+20)>0 THEN
17895       Vollies(Type,Task)=Rif_fired(Type,Task)/Throw_wt
17898     ELSE
17901       Vollies(Type,Task)=0    ! number of Btry/plt vollies or lchr load
17904     END IF
17907   NEXT Task
17910 NEXT Type
17913  !
17916 Rfd$="FIDEL_R"
17919 Disk3$=":9134,704,0"
17922 ASSIGN @Readred TO Rfd$&Disk3$
17925 ENTER @Readred;Del_par(*)
```

Table 6-14. Ground combat code (continued).

```
17928 !
17931 ! calculate fd & total if losses
17934  FOR Element=1 TO 35
17937    Ps(Element)=1
17940    GOSUB Compute_red
17943 !   this computes total losses due to indirect fire
17946    Sys_arty(2,Element)=(1-Ps(Element))*Sys_arty(1,Element)
17949    IF Element=1 THEN
17952    END IF
17955  NEXT Element
17958 ! this allows 2 postures for infantry
17961  Element=71   ! standing
17964  Ps(71)=1
17967  GOSUB Compute_red
17970 !
17973  Element=72   ! CROUCH
17976  Ps(72)=1
17979  GOSUB Compute_red
17982 !
17985  Attacker=Atk_def+1
17988  IF Rarty_fire>1 THEN
17991    Posture=R_psnl_posture(Attacker,2)
17994  ELSE
17997    Posture=R_psnl_posture(Attacker,1)
18000  END IF
18003 !
18006  Ps_inf=(1-Posture)*Ps(71)+Posture*Ps(72)
18009  FOR Element=36 TO 47
18012    Ps(Element)=Ps_inf
18015    Sys_arty(2,Element)=(1-Ps(Element))*Sys_arty(1,Element)
18018  NEXT Element
18021 !
18024  FOR Element=48 TO 70
18027    Ps(Element)=1
18030    GOSUB Compute_red
18033    Sys_arty(2,Element)=(1-Ps(Element))*Sys_arty(1,Element)
18036  NEXT Element
18039                     ! FOR Element=1 TO 70
18042                     !   NEXT Element
18045!
18048  GOTO Sub_end
18051!
18054!-----------------------------------------------------------------
18057!
18060 Compute:!
18063  FOR Task=1 TO 5
18066    F_d(Task,Element)=0
18069  NEXT Task
18072    !
18075  FOR Task=1 TO 5
18078    IF R_tgt_mask(Task,Element)=0 THEN End_loop
18081  !  Type=1 ! arty firing DP-ICM @ 10 K
```

Table 6-14. Ground combat code (continued).

```
18084     FOR Type=1 TO 7
18087       IF Vollies(Type,Task)>0 THEN
18090         GOSUB Re_read
18093         W_dp=L_dp
18096         P_dp=1-EXP(P_dp_f*Al(Type,Element))
18099         A_el=L_dp*W_dp*P_dp
18102         L_v=L_v_fac*(Sys_tot(2,Type+20)/6)
18105         W_v=W_v_fac*(Sys_tot(2,Type+20)/6)
18108         N_r=6
18111         GOSUB Calculate_fd
18114       END IF
18117     NEXT Type
18120   !
18123 Tpe2: !
18126   !
18129   !  Type=2 ! MLRS firing DP-ICM @ 25 Km
18132     FOR Type=12 TO 15
18135       IF Vollies(Type,Task)>0 THEN
18138         GOSUB Re_read
18141         W_dp=L_dp
18144         P_dp=1-EXP(P_dp_f*Al(Type,Element))
18147         A_el=L_dp*W_dp*P_dp
18150         L_v=L_v_fac*(Sys_tot(2,Type+20))
18153         W_v=W_v_fac*(Sys_tot(2,Type+20))
18156         N_r=12
18159         GOSUB Calculate_fd
18162       END IF
18165     NEXT Type
18168   !
18171 Tpe3: !
18174   !
18177   !  Type=3 ! MORTAR FIRING HE @ 3 Km
18180     FOR Type=8 TO 11
18183       IF Vollies(Type,Task)>0 THEN
18186         GOSUB Re_read
18189         L_dp=L_dp_f*SQR(Al(Type,Element))
18192         W_dp=L_dp/.564288772
18195         A_el=Al(Type,Element)
18198         L_v=L_v_fac*(Sys_tot(2,Type+20)/6)
18201         W_v=W_v_fac*(Sys_tot(2,Type+20)/6)
18204         N_r=6
18207         GOSUB Calculate_fd
18210       END IF
18213     NEXT Type
18216   !
18219   !
18222 End_loop:!
18225   NEXT Task
18228   RETURN
18231!
18234!------------------------------------------------------------------
18237!
```

Table 6-14. Ground combat code (continued).

```
18240 Compute_red:!
18243  FOR Task=1 TO 5
18246    F_d(Task,Element)=0
18249  NEXT Task
18252  FOR Task=1 TO 5
18255    IF B_tgt_mask(Task,Element)=0 THEN Skip
18258 !   Type=1  ! Arty firing DP-ICM @ 11.5 Km
18261    FOR Type=1 TO 7
18264      IF Vollies(Type,Task)>0 THEN
18267        GOSUB Re_read
18270        L_dp=L_dp_f*SQR(A1(Type,Element))
18273        W_dp=L_dp/.564288772
18276 !     Note: A1 is equivalent rd A1, not submunition
18279        A_el=A1(Type,Element)
18282        L_v=L_v_fac*(Sys_tot(4,Type+20)/6)
18285        W_v=W_v_fac*(Sys_tot(4,Type+20)/6)
18288        N_r=6
18291        GOSUB Calculate_fd
18294      END IF
18297    NEXT Type
18300 Rtpe2:     !
18303     !
18306  !  Type=2 ! mrl firing HE @ 13.7 Km
18309    FOR Type=12 TO 15
18312      IF Vollies(Type,Task)>0 THEN
18315        GOSUB Re_read
18318        L_dp=L_dp_f*SQR(A1(Type,Element))
18321        W_dp=L_dp/.564288772
18324        A_el=A1(Type,Element)
18327        L_v=L_v_fac*(Sys_tot(4,Type+20))
18330        W_v=W_v_fac*(Sys_tot(4,Type+20))
18333        N_r=40
18336        GOSUB Calculate_fd
18339      END IF
18342    NEXT Type
18345   !
18348 Rtpe3:  !
18351   !
18354  !  Type=3 ! mortar firing HE @ 3.8 Km
18357    FOR Type=8 TO 11
18360      IF Vollies(Type,Task)>0 THEN
18363        GOSUB Re_read
18366        L_dp=L_dp_f*SQR(A1(Type,Element))
18369        W_dp=L_dp/.564288772
18372        A_el=A1(Type,Element)
18375        L_v=L_v_fac*(Sys_tot(4,Type+20)/6)
18378        W_v=W_v_fac*(Sys_tot(4,Type+20)/6)
18381        N_r=6
18384        GOSUB Calculate_fd
18387      END IF
18390    NEXT Type
18393 Skip:!
```

Table 6-14. Ground combat code (continued).

```
18396  NEXT Task
18399  RETURN
18402!
18405!-----------------------------------------------------------------------
18408!
18411 Calculate_fd:!
18414  L_ap=L_dp+Krep_p        ! Single Round Adj
18417  W_ap=W_dp+Kdep_p        !  Damage Pattern
18420  L_vp=L_ap+L_v
18423  W_vp=W_ap+W_v
18426  A_vp=L_vp*W_vp          ! Volley Damage Pat.
18429  A_ap=L_ap*W_ap
18432  Of=N_r*A_ap/A_vp        ! Overlap Factor
18435  IF Of<1 THEN Of=1
18438  !P_nv=1-(1-(A_el*N_r*R_r)/(A_vp*Of))^(Vollies(Type,Task)*Of)   ! P(damage
18441  P_nv1=(A_el*N_r*R_r)/(A_vp*Of)
18444  IF P_nv1>1 THEN P_nv1=1
18447  P_nv=1-(1-P_nv1)^(Vollies(Type,Task)*Of)
18450   !
18453  Rep_tm=SQR((Rep_m^2)+(.5731*Tle(Task))^2)
18456  Dep_tm=SQR((Dep_m^2)+(.5731*Tle(Task))^2)
18459  A_1=(L_vp+Length(Task))/(2.96*Rep_tm)
18462  A_2=(ABS(L_vp-Length(Task)))/(2.96*Rep_tm)
18465  IF A_1>33.532 THEN A_1=33.532    ! trunk if area too large
18468  IF A_2>33.532 THEN A_2=33.532    !   "   "
18471  F_1a=.5*A_1*SQR(1-EXP(-.63*A_1^2))+EXP(-.5*A_1^2)/SQR(2*PI)
18474  F_1b=.5*A_2*SQR(1-EXP(-.63*A_2^2))+EXP(-.5*A_2^2)/SQR(2*PI)
18477  F_1=F_1a-F_1b
18480  Ec_r=2.96*Rep_tm*F_1/Length(Task)  ! expected frac coverage in range
18483  B_1=(W_vp+Width(Task))/(2.96*Dep_tm)
18486  B_2=ABS(W_vp-Width(Task))/(2.96*Dep_tm)
18489  IF B_1>33.532 THEN B_1=33.532    ! trunk if area too large
18492  IF B_2>33.532 THEN B_2=33.532   .!   "
18495  F_wa=.5*B_1*SQR(1-EXP(-.63*B_1^2))+EXP(-.5*B_1^2)/SQR(2*PI)
18498  F_wb=.5*B_2*SQR(1-EXP(-.63*B_2^2))+EXP(-.5*B_2^2)/SQR(2*PI)
18501  F_w=F_wa-F_wb
18504  Ec_d=2.96*Dep_tm*F_w/Width(Task)    ! expected frac coverage in df
18507  F_d(Task,Element)=Ec_r*Ec_d*P_nv    !    "          "   damage
18510  !  accumulate the prob of surv(Ps) since it is joint prob for all killer
18513  Ps(Element)=Ps(Element)*(1-F_d(Task,Element))
18516  RETURN
18519!
18522!-----------------------------------------------------------------------
18525!
18528 Re_read:!
18531  L_dp=Del_par(Type,1)
18534  L_dp_f=Del_par(Type,2)
18537  P_dp_f=Del_par(Type,3)
18540  Krep_p=Del_par(Type,4)
18543  Kdep_p=Del_par(Type,5)
18546  L_v_fac=Del_par(Type,6)
18549  W_v_fac=Del_par(Type,7)
```

Table 6-14. Ground combat code (continued).

```
18552  R_r=Del_par(Type,8)
18555  Dep_m=Del_par(Type,9)
18558  Rep_m=Del_par(Type,10)
18561  RETURN
18564!
18567!----------------------------------------------------------------
18570!
18573 Sub_end:!
18576  ASSIGN @Readred TO *
18579  ASSIGN @Readblue TO *
18582  ASSIGN @Areafile TO *
18585 SUBEND
18588 !
18591 !*********************************************************************
18594 !
18597 SUB Ck_var(Var_name$,T$,Variable,Min_value,Max_value)
18600  PRINTER IS 1
18603  SELECT T$
18606  CASE "THRU"
18609    WHILE Variable<Min_value OR Variable>Max_value
18612      GOSUB Print_error
18615    END WHILE
18618  CASE "OR"
18621    GOTO Case_to
18624  CASE "TO"
18627 Case_to:FOR M=Min_value TO Max_value
18630      IF Variable=M THEN GOTO End_select
18633    NEXT M
18636    GOSUB Print_error
18639    GOTO Case_to
18642 End_select:!
18645  END SELECT
18648  GOTO Rtrn
18651 Print_error:   !
18654  PRINT
18657  PRINT "** ERROR: ";Variable;" IS INVALID FOR ";Var_name$
18660  PRINT "INPUT: ";Min_value;" ";T$;" ";Max_value;" ONLY"
18663  INPUT Variable
18666  RETURN
18669 Rtrn:!
18672 SUBEND
18675 !
18678 !**********************************************************************
18681 !
18684 SUB Smoke(Iunit,Ielem,Amowt(*),Fwidt,Mtime,Ikm,Range,Sleng(*),Tot_sleng.
n(*),Pseeopt(*),Pseecs(*),Pseethm(*),Irh)
18687  OPTION BASE 1
18690  DIM Smkdat(3,2,4,7),Snrd(7),Disk3$[50]
18693  DIM Elem$[1],Type$[4],Side$[1]
18696  COM /Smoke/ Amwtpp(3,11),Irof(3,11)
18699 !
18702  Disk3$=":9134,704,0"
```

Table 6-14. Ground combat code (continued).

```
18705  IF Ikm=1 THEN Ikm=4
18708  IF Ikm=2 THEN Ikm=3
18711  IF Ikm=3 THEN Ikm=2
18714  IF Ikm=4 THEN Ikm=1
18717  IF Range<=500 THEN Irg=1
18720  IF Range<=1000 AND Range>500 THEN Irg=2
18723  IF Range<=1500 AND Range>1000 THEN Irg=3
18726  IF Range<=2000 AND Range>1500 THEN Irg=4
18729  IF Range<=2500 AND Range>2000 THEN Irg=5
18732  IF Range<=3000 AND Range>2500 THEN Irg=6
18735  IF Range>3000 THEN Irg=7
18738 !
18741  Istart=1
18744  IF Ielem=1 THEN       !READ ARTILLERY DATA
18747    Iend=7
18750    Type$="ARTY"
18753  ELSE                      !READ MORTAR DATA
18756    Iend=4
18759    Type$="MORT"
18762  END IF
18765  IF Iunit=1 OR Iunit=2 THEN Side$="B"
18768  IF Iunit=3 THEN Side$="R"
18771  FOR I=Istart TO Iend
18774    J=I+Ielem-1
18777    Elem$=VAL$(I)
18780    ASSIGN @P TO "SMK_"&Type$&Side$&Elem$&Disk3$
18783    ENTER @P,1;Smkdat(*)
18786    ASSIGN @P TO *
18789    LET Pseeopt(I)=Smkdat(1,Irh,Ikm,Irg)
18792    LET Pseecs(I)=Smkdat(2,Irh,Ikm,Irg)
18795    LET Pseethm(I)=Smkdat(3,Irh,Ikm,Irg)
18798 !
18801    IF Amowt(J)>0 THEN
18804      LET Amwtp=Amowt(J)*2000
18807      LET Nrd=INT(Amwtp/Amwtpp(Iunit,J))
18810      LET Snrd(I)=Nrd      !SAVE NUMBER OF ROUNDS FIRED
18813 !
18816 ! THIS PORTION OF THE PROGRAM CALCULATES THE LENGTH(LENG) OF THE
18819 !SCREEN, WHERE LENGTH IS EQUAL TO (NRD/(MTIME*IROF))*200
18822      LET Sleng(I)=(Nrd/(Mtime*Irof(Iunit,J)))*200
18825      Tot_sleng=Tot_sleng+Sleng(I)    !  TOTAL LENGTH OF SCREEN
18828    ELSE
18831      Snrd(I)=0
18834      Sleng(I)=0
18837    END IF
18840  NEXT I
18843 !
18846 ! THIS PORTION OF THE PROGRAM DETERMINES IF THE SCREEN LENGTH(SLENG)
18849 ! IS GREATER THAN FRONTAGE WIDTH(FWIDT)...IF SO THEN SLENG BECOMES
18852 ! FWIDT AND EXTRA AMMO IS RETURNED TO DRIVER ROUTINE
18855  IF Tot_sleng>Fwidt THEN
18858    Perc_used=1-(Tot_sleng-Fwidt)/Tot_sleng
```

Table 6-14. Ground combat code (continued).

```
18861     Tot_sleng=Fwidt
18864   ELSE
18867     Perc_used=1.0
18870   END IF
18873 ! DETERMINE ANY UNUSED AMMO AND RETURN IN VARIABLE--TON
18876   FOR I=Istart TO Iend
18879     J=I+Ielem-1
18882     Sleng(I)=Sleng(I)*Perc_used
18885     IF Sleng(I)>0 THEN Nrd=INT(Sleng(I)/200*(Mtime*Irof(Iunit,J))+.5)
18888     IF Sleng(I)<=0 THEN Nrd=0
18891     IF Snrd(I)>Nrd THEN
18894       Nrd1=Snrd(I)-Nrd
18897       LET Amwtp1=Amwtpp(Iunit,J)*Nrd1
18900       LET Ton(I)=Amwtp1/2000
18903     END IF
18906   NEXT I
18909    !DETERMINE THE % OF FRONTAGE COVERED BY THE SCREEN, WHERE THE % OF
18912    !FRONTAGE SMOKED(PSMOKED) = LENGTH OF SCREEN DIVIDED BY THE TOTAL
18915    !FRONTAGE BETWEEN UNITS(FWIDT)
18918   Psmoked=Tot_sleng/Fwidt
18921    !
18924    !% FRONTAGE NOT COVERED BY SCREEN
18927   Pnotsmk=1-Psmoked
18930    !
18933 SUBEND
18936 !
18939 !************************************************************************
18942 !
18945 SUB Smkemp(Iunt,Ielem,Amowt(*),Fwidt,Mxtime,Ik,Range,Mton(*),Aton(*),Pvc
,Pvcs,Pvth,Irh)
18948 ! THIS ROUTINE CALLS FOR SMOKE. 1) MORTARS EMPLACE SMOKE...IF MORTARS
18951 ! ARE UNABLE TO COVER ENTIRE FRONTAGE,THEN 2)ARTILLERY EMPLACE SMOKE,
18954 ! BASED ON THE UNSMOKED FRONTAGE.
18957 !
18960   OPTION BASE 1
18963   DIM Mpseeopt(4),Mpseecs(4),Mpseeth(4),Apseeopt(7),Apseecs(7),Apseeth(7)
18966   DIM Mleng(4),Aleng(7)
18969 !
18972   Iunit=Iunt
18975   Ikm=Ik
18978   Amwt=0
18981   FOR I=1 TO 4        !  TOTAL WT. OF MORT AMMO AVAILABLE TO FIRE FOR SMOKE
18984     Amwt=Amwt+Amowt(I+7)
18987   NEXT I
18990   IF Amwt>0 THEN
18993     Ielem=8
18996     CALL Smoke(Iunit,Ielem,Amowt(*),Fwidt,Mxtime,Ikm,Range,Mleng(*),Tot_m
ng,Mton(*),Mpseeopt(*),Mpseecs(*),Mpseeth(*),Irh)
18999   END IF
19002 ! DETERMINE IF FRONTAGE IS COMPLETELY COVERED...IF NOT CALL EMPLACE ARTI
ERY SMOKE
19005   Amwt=0
```

Table 6-14. Ground combat code (continued).

```
19008  FOR I=1 TO 7        !  TOTAL WT. OF ARTY AMMO AVAILABLE TO FIRE FOR SMO
19011    Amwt=Amwt+Amowt(I)
19014  NEXT I
19017  IF Amwt>0 THEN
19020    IF Fwidt>Tot_mleng THEN
19023      F1widt=Fwidt-Tot_mleng
19026      Ielem=1
19029      CALL Smoke(Iunit,Ielem,Amowt(*),F1w.dt,Mxtime,Ikm,Range,Aleng(*),-
aleng,Aton(*),Apseeopt(*),Apseecs(*),Apseeth(*),Irh)
19032    ELSE
19035      FOR I=1 TO 7
19038        Aton(I)=Amowt(I)
19041      NEXT I
19044    END IF
19047  END IF
19050 ! CALCULATE THE PERCENT VISIBLE ACROSS ENTIRE FRONTAGE FOR MAX TIME,AS
19053 ! A FUNCTION OF OPTICS SENSORS
19056 !
19059  LET Pnotsmok=((Fwidt-(Tot_mleng+Tot_aleng))/Fwidt)*1
19062  Pmorto=0
19065  FOR I=1 TO 4
19068    LET Pmorto=Pmorto+((Mleng(I)/Fwidt)*Mpseeopt(I))
19071  NEXT I
19074  Partyo=0
19077  FOR I=1 TO 7
19080    LET Partyo=Partyo+((Aleng(I)/Fwidt)*Apseeopt(I))
19083  NEXT I
19086  LET Pvopt=Pmorto+Partyo+Pnotsmok
19089 !CALCULATE THE PERCENT VISIBLE ACROSS ENTIRE FRONTAGE FOR MAX TIME
19092 ! AS A FUNCTION OF CREW SERVED SENSORS
19095 !
19098  Pmortc=0
19101  FOR I=1 TO 4
19104    LET Pmortc=Pmortc+((Mleng(I)/Fwidt)*Mpseecs(I))
19107  NEXT I
19110  Partyc=0
19113  FOR I=1 TO 7
19116    LET Partyc=Partyc+((Aleng(I)/Fwidt)*Apseecs(I))
19119  NEXT I
19122  LET Pvcs=Pmortc+Partyc+Pnotsmok
19125 !CALCULATE THE PERCENT VISIBLE ACROSS ENTIRE FRONTAGE FOR MAX TIME
19128 ! AS A FUNCTION OF THERMAL SENSORS
19131 !
19134  Pmortt=0
19137  FOR I=1 TO 4
19140    LET Pmortt=Pmortt+((Mleng(I)/Fwidt)*Mpseeth(I))
19143  NEXT I
19146  Partyt=0
19149  FOR I=1 TO 7
19152    LET Partyt=Partyt+((Aleng(I)/Fwidt)*Apseeth(I))
19155  NEXT I
19158  LET Pvth=Pmortt+Partyt+Pnotsmok
```

Table 6-14. Ground combat code (continued).

```
19161 SUBEND
19164 !
19167 !*********************************************************************
19170 !
19173 SUB Df_attrition(B_vua(*),R_vua(*),Terrain,Day_nite,Vis,Num_bands,Rng_ba
,Atk_def,Sys(*),B_tons(*),R_tons(*),B_vis(*),R_vis(*),T,S,Bu,Ru,St,Dc)
19176 !
19179  OPTION BASE 1
19182 !
19185  DIM Init_b_targets(70),Init_r_targets(70)
19188  DIM B_ammo(20),R_ammo(20)
19191  DIM B_ecf(20),R_ecf(20),B_ecf_vis(20,4),R_ecf_vis(20,4)
19194  DIM B_fire_d(20,20),R_fire_d(20,20)
19197  DIM B_pk_fe(20,20),R_pk_fe(20,20)
19200  DIM B_pk_hd(20,20),R_pk_hd(20,20)
19203  DIM B_targ(20),R_targ(20)
19206  DIM B_rnds_wep(20),R_rnds_wep(20)
19209  DIM B_sum(20),R_sum(20)
19212  DIM B_fdf(20,20),R_fdf(20,20)
19215  DIM Tot_b_rnds(20),Tot_r_rnds(20)
19218  DIM B_rnds_cat(20,17),R_rnds_cat(20,17)
19221  DIM Loss_b_cat(17),Loss_r_cat(17)
19224  DIM Loss_blue(70),Loss_red(70)
19227  DIM B_psuv(17),R_psuv(17)
19230  DIM B_targ_vis(17),R_targ_vis(17)
19233  DIM B_ex_r(20,17),R_ex_r(20,17)
19236  DIM B_ps(20,17),R_ps(20,17)
19239  DIM Lo_b_cats(17),Lo_r_cats(17)
19242  DIM Lo_b_cat(20,17),Lo_r_cat(20,17)
19245  DIM B_vul(17),R_vul(17),B_vul_t(17),R_vul_t(17)
19248  DIM B_sen(70),R_sen(70),B_look(17),R_look(17)
19251  DIM Df_pk_fe(20,20),Df_pk_hd(20,20)
19254  INTEGER I,J
19257 !
19260  COM /Direct_fire/ B_cat(70),R_cat(70),B_sen_d(70),B_sen_n(70),R_sen_d(7
,R_sen_n(70),B_ammo_wt(20),R_ammo_wt(20)
19263 !
19266 COM /Helo_info/ Btl_rg,Rg_avg(2,3,20),Rg_avg_pd(2,3,5),Df_ammo(2),Df_fir
dist(2,20,3),Df_pk_helo(2,20,3,2),INTEGER Df_sen_ptr(2,20),Df_muni_ptr(2,20)
19269 !
19272 COM /No_helos/ Cell(2,2,3)
19275 !
19278 Main:!
19281  GOSUB Set_call
19284  GOSUB Init_reads
19287 !BEGIN RANGE BAND LOOP
19290  FOR Band_loop=1 TO Loops
19293    GOSUB Read_files
19296    GOSUB Categorize
19299    GOSUB Smokes
19302    GOSUB Calc_fdf
19305    GOSUB Ammo_available
```

Table 6-14. Ground combat code (continued).

```
19308     GOSUB Calc_rnd
19311     GOSUB Update_ammo
19314     GOSUB Calc_loss
19317     GOSUB Decat_losses
19320     GOSUB Set_next_loop
19323     IF Abort$="YES" THEN GOTO Out_loop
19326   NEXT Band_loop
19329 Out_loop:!
19332   GOTO Set_return
19335 !
19338 !-----------------------------------------------------------------
19341 !
19344 Set_cal::!
19347   Abort$="NO"
19350   Rng_band_save=Rng_band
19353 !INITIALIZE AMMO USED TO ZERO AND ELEMENTS KILLED TO ZERO
19356   FOR I=1 TO 70
19359       !SAVE INITIAL TARGETS
19362     Init_b_targets(I)=Sys(1,I)
19365     Init_r_targets(I)=Sys(3,I)
19368   NEXT I
19371 !INITIALIZE FOR LOOPS
19374   Loops=Num_bands
19377   Partial=1
19380   IF Num_bands<1 THEN
19383     Loops=1
19386     Partial=Num_bands
19389   END IF
19392   RETURN
19395!
19398!-----------------------------------------------------------------
19401!
19404 Init_reads: !
19407   IF Day_nite=0 THEN       !DAY
19410     FOR I=1 TO 70
19413       B_sen(I)=B_sen_d(I)
19416       R_sen(I)=R_sen_d(I)
19419     NEXT I
19422   ELSE                     !NIGHT
19425     FOR I=1 TO 70
19428       B_sen(I)=B_sen_n(I)
19431       R_sen(I)=R_sen_n(I)
19434     NEXT I
19437   END IF
19440    !
19443   Disk3$=":9134,704,0"
19446   R$="RD"
19449   B$="BH"
19452   Rr$="RH"
19455 !SELECT ECF FILES
19458   SELECT Atk_def
19461   CASE 0
```

Table 6-14. Ground combat code (continued).

```
19464     Rad$="A"
19467     Bad$="D"
19470   CASE 1
19473     Rad$="D"
19476     Bad$="A"
19479   END SELECT
19482   !
19485   SELECT Day_nite
19488   CASE 0
19491     Dn$="D"
19494   CASE 1
19497     Dn$="N"
19500   END SELECT
19503   !
19506   SELECT Terrain
19509   CASE 1
19512     T$="O"
19515   CASE 2
19518     T$="R"
19521   CASE 3
19524     T$="H"
19527   CASE 4
19530     T$="M"
19533   END SELECT
19536   !
19539 !ENTER FIRE DISTRIBUTIONS
19542   ASSIGN @Path1 TO B$&"_FIRE_"&Bad$&":9134,704,0"   !Disk3$
19545   ASSIGN @Path2 TO R$&"_FIRE_"&Rad$&":9134,704,0"   !Disk3$
19548   ENTER @Path1,1;B_fire_d(*)
19551   ENTER @Path2,1;R_fire_d(*)
19554   ASSIGN @Path1 TO *
19557   ASSIGN @Path2 TO *
19560 !
19563 !  OPEN FILES THAT ARE READ WITHIN RANGE BAND LOOP
19566   ASSIGN @Path1 TO B$&Dn$&Bad$&"_ECF_"&T$&Disk3$
19569   ASSIGN @Path2 TO R$&Dn$&Rad$&"_ECF_"&T$&Disk3$
19572   ASSIGN @Path_b_fe TO B$&"_PKFE"&":9134,704,0"    !Disk3$
19575   ASSIGN @Path_b_hd TO B$&"_PKHD"&":9134,704,0"    !Disk3$
19578   ASSIGN @Path_r_fe TO Rr$&"_PKFE"&":9134,704,0"   !Disk3$
19581   ASSIGN @Path_r_hd TO Rr$&"_PKHD"&":9134,704,0"   !Disk3$
19584   FOR Side=1 TO 2   !READ IN & SAVE THE PK'S FOR HELOS BY DF
19587     Side_def=(Side MOD 2)+1
19590     FOR I=1 TO 3     !LOOP ON HELOS
19593       Helo_bnd=INT(Rg_avg(Side_def,I,1)/500+.5)  !CALC HELO PK RANGE BAN
19596       IF Helo_bnd<=0 THEN Helo_bnd=1
19599       IF Helo_bnd<=6 THEN
19602         IF Side=1 THEN    !BLUE
19605           ENTER @Path_b_fe,Helo_bnd;Df_pk_fe(*)
19608           ENTER @Path_b_hd,Helo_bnd;Df_pk_hd(*)
19611         ELSE              !RED
19614           ENTER @Path_r_fe,Helo_bnd;Df_pk_fe(*)
19617           ENTER @Path_r_hd,Helo_bnd;Df_pk_hd(*)
```

Table 6-14. Ground combat code (continued).

```
19620          END IF
19623          FOR J=1 TO 20   !LOOP ON DF
19626            Df_pk_helo(Side,J,I,1)=Df_pk_fe(J,I+17)  !NON MAST MOUNTED
19629            Df_pk_helo(Side,J,I,2)=Df_pk_hd(J,I+17   !MAST MOUNTED
19632          NEXT J
19635        ELSE     !OUTSIDE OF DF RANGE
19638          FOR J=1 TO 20
19641            Df_pk_helo(Side,J,I,1)=0
19644            Df_pk_helo(Side,J,I,2)=0
19647          NEXT J
19650        END IF
19653      NEXT I
19656    NEXT Side
19659  !
19662    RETURN
19665  !
19668  !----------------------------------------------------------------------
19671  !
19674 Read_files:!
19677   !
19680   ENTER @Path1,Rng_band;B_ecf_vis(*)
19683   ENTER @Path2,Rng_band;R_ecf_vis(*)
19686  !
19689   FOR I=1 TO 20
19692     B_ecf(I)=B_ecf_vis(I,Vis)
19695     R_ecf(I)=R_ecf_vis(I,Vis)
19698   NEXT I
19701   !
19704  !DIVIDE ECF'S FOR PART OF RANGE BAND
19707   FOR I=1 TO 20
19710     B_ecf(I)=B_ecf(I)*Partial
19713     R_ecf(I)=R_ecf(I)*Partial
19716   NEXT I
19719   !
19722  !ENTER PK FILES
19725   ENTER @Path_b_fe,Rng_band;B_pk_fe(*)
19728   ENTER @Path_b_hd,Rng_band;B_pk_hd(*)
19731   ENTER @Path_r_fe,Rng_band;R_pk_fe(*)
19734   ENTER @Path_r_hd,Rng_band;R_pk_hd(*)
19737   FOR I=1 TO 20
19740     FOR J=1 TO 3
19743       B_pk_fe(I,J+17)=Df_pk_helo(1,I,J,1)   !SET HELO'S PK IN ARRAY
19746       R_pk_fe(I,J+17)=Df_pk_helo(2,I,J,1)
19749     NEXT J
19752   NEXT I
19755   RETURN
19758!
19761!------------------------------------------------------------------------
19764!
19767 Categorize:!   ROB
19770   Dc=Dc+1
19773   !         Roy_file$="R_"&VAL$(T)&"_"&VAL$(S)&"_"&VAL$(Dc)
```

Table 6-14. Ground combat code (continued).

```
19776  !        CREATE BDAT Roy_file$&":HP9121,700,1",10,1550
19779  !        ASSIGN @Proy TO Roy_file$&":HP9121,700,1"
19782  !        OUTPUT @Proy,1;T,S,Bu,Ru,St
19785  !        OUTPUT @Proy,2;Sys(*)
19788!INITIALIZE CATEGORIES
19791  FOR I=1 TO 17
19794    B_vul(I)=0
19797    R_vul(I)=0
19800    B_vul_t(I)=0
19803    R_vul_t(I)=0
19806  NEXT I
19809  FOR I=1 TO 20
19812    B_targ(I)=0
19815    R_targ(I)=0
19818  NEXT I
19821!ADD ELEMENTS TO CATEGORIES
19824  FOR I=1 TO 70
19827    IF B_cat(I)>0 AND B_cat(I)<=17 THEN
19830      B_targ(B_cat(I))=B_targ(B_cat(I))+Sys(1,I)
19833      B_vul_t(B_cat(I))=B_vul_t(B_cat(I))+Sys(1,I)*B_vua(I)
19836    END IF
19839    IF R_cat(I)>0 AND R_cat(I)<=17 THEN
19842      R_targ(R_cat(I))=R_targ(R_cat(I))+Sys(3,I)
19845      R_vul_t(R_cat(I))=R_vul_t(R_cat(I))+Sys(3,I)*R_vua(I)
19848    END IF
19851  NEXT I
19854  FOR I=18 TO 20   !ADD HELO ELEMENTS TO TGT CATEGORIES
19857    B_targ(I)=Cell(1,1,I-17)
19860    R_targ(I)=Cell(2,1,I-17)
19863  NEXT I
19866  FOR I=1 TO 17
19869    IF B_targ(I)>0 THEN B_vul(I)=B_vul_t(I)/B_targ(I)
19872    IF R_targ(I)>0 THEN R_vul(I)=R_vul_t(I)/R_targ(I)
19875  NEXT I
19878         !            OUTPUT @Proy,3;B_targ(*),R_targ(*)
19881  RETURN
19884 !
19887 !----------------------------------------------------------------------
19890 !
19893 Smokes: !
19896   ! CALCULATE TARGETS VISIBLE THROUGH SMOKE
19899  FOR I=1 TO 17
19902    B_targ_vis(I)=0
19905    R_targ_vis(I)=0
19908    B_look(I)=0
19911    R_look(I)=0
19914  NEXT I
19917   ! ADD VISIBLE ELEMENTS TO EACH CATEGORY
19920  FOR I=1 TO 17     !GROUND TGT CATEG
19923    FOR J=1 TO 20  !DIRECT FIRERS
19926      B_targ_vis(I)=B_targ_vis(I)+Sys(6,J)*R_vis(R_sen(J))
19929      R_targ_vis(I)=R_targ_vis(I)+Sys(5,J)*B_vis(B_sen(J))
```

Table 6-14. Ground combat code (continued).

```
19932      R_look(I)=R_look(I)+Sys(6,J)
19935      B_look(I)=B_look(I)+Sys(5,J)
19938    NEXT J
19941    IF R_look(I)>0 THEN
19944      B_targ_vis(I)=B_targ_vis(I)/R_look(I)
19947    ELSE
19950      B_targ_vis(I)=0
19953    END IF
19956    IF B_look(I)>0 THEN
19959      R_targ_vis(I)=R_targ_vis(I)/B_look(I)
19962    ELSE
19965      R_targ_vis(I)=0
19968    END IF
19971  NEXT I
19974  RETURN
19977!
19980!------------------------------------------------------------------
19983!
19986 Calc_fdf:! FIRE DISTRIBUTION FACTOR
19989  FOR I=1 TO 20
19992    B_sum(I)=0
19995    R_sum(I)=0
19998  NEXT I
20001   !SUM
20004  FOR I=1 TO 20
20007    FOR J=1 TO 20
20010      B_sum(I)=B_sum(I)+R_targ(J)*B_fire_d(I,J)*B_pk_fe(I,J)
20013      R_sum(I)=R_sum(I)+B_targ(J)*R_fire_d(I,J)*R_pk_fe(I,J)
20016    NEXT J
20019  NEXT I
20022   !
20025   !CALC FDF
20028  FOR I=1 TO 20
20031    FOR J=1 TO 20
20034      R_fdf(I,J)=0
20037      B_fdf(I,J)=0
20040      IF B_sum(I)<>0 THEN
20043        B_fdf(I,J)=R_targ(J)*B_fire_d(I,J)*B_pk_fe(I,J)/B_sum(I)
20046      END IF
20049      IF R_sum(I)<>0 THEN
20052        R_fdf(I,J)=B_targ(J)*R_fire_d(I,J)*R_pk_fe(I,J)/R_sum(I)
20055      END IF
20058    NEXT J
20061    FOR J=1 TO 3      !STORE FIRE DISTRIBUTION OF HELO TGTS FOR USE IN
                          !HELO_KILLS SUBROUTINE
20064      Df_fire_dist(1,I,J)=B_fdf(I,J+17)
20067      Df_fire_dist(2,I,J)=R_fdf(I,J+17)
20070    NEXT J
20073  NEXT I
20076  RETURN
20079!
20082!--------------------------------------------------------------------
```

Table 6-14. Ground combat code (continued).

```
20085!
20088 Ammo_available:!
20091  !CHANGE TONS PER SINGLE WEAPON TO RNDS PER TYPE IN POUNDS
20094  FOR I=1 TO 20
20097    B_ammo(I)=0
20100    IF B_ammo_wt(I)<>0 THEN
20103      B_ammo(I)=B_tons(1,I)/B_ammo_wt(I)*2000
20106    END IF
20109    R_ammo(I)=0
20112    IF R_ammo_wt(I)<>0 THEN
20115      R_ammo(I)=R_tons(1,I)/R_ammo_wt(I)*2000
20118    END IF
20121  NEXT I
20124  RETURN
20127  !
20130  !--------------------------------------------------------------------
20133  !
20136 Calc_rnd:! TOTAL ROUNDS FIRED
20139  FOR I=1 TO 20
20142     !ROUNDS FIRED
20145    IF B_ammo(I)>B_ecf(I) THEN
20148      B_rnds_wep(I)=B_ecf(I)
20151    ELSE
20154      B_rnds_wep(I)=B_ammo(I)
20157    END IF
20160      !
20163    IF R_ammo(I)>R_ecf(I) THEN
20166      R_rnds_wep(I)=R_ecf(I)
20169    ELSE
20172      R_rnds_wep(I)=R_ammo(I)
20175    END IF
20178     !
20181  NEXT I
20184  !
20187  !CALC. ROUNDS FIRED BY ALL CATEGORIES
20190  FOR I=1 TO 20
20193    Tot_b_rnds(I)=Sys(5,I)*B_rnds_wep(I)
20196    Tot_r_rnds(I)=Sys(6,I)*R_rnds_wep(I)
20199  NEXT I
20202   !
20205   !TEMPORARY CODE FOR HTLD/ADEA ONLY
20208      ! DET MAX # GLLD DESIGNATORS
20211  Glds=Sys(5,4)/5       !CALCS GLLDS AVAIL FOR GLH
20214  Glh=MIN(Glds,Sys(5,5))
20217  Tot_b_rnds(5)=Glh*B_rnds_wep(5)
20220       ! ENDS TEMP CODE
20223   !
20226   !ROUNDS PER CATEGORY
20229  FOR I=1 TO 20
20232    FOR J=1 TO 17
20235      B_rnds_cat(I,J)=B_fdf(I,J)*Tot_b_rnds(I)
20238      R_rnds_cat(I,J)=R_fdf(I,J)*Tot_r_rnds(I)
```

Table 6-14. Ground combat code (continued).

```
20241     NEXT J
20244   NEXT I
20247       !     OUTPUT @Proy,4:Tot_b_rnds(*),Tot_r_rnds(*),B_rnds_cat(*),R_rnd
cat(*)
20250   RETURN
20253!
20256!-------------------------------------------------------------
20259!
20262 Update_ammo:!
20265    !CALC. TONS USED; TAKE RNDS USED AND CHANGE TO TONS PER SINGLE WEAPON
20268   FOR I=1 TO 20
20271     IF Sys(5,I)<>0 THEN B_tons(2,I)=Tot_b_rnds(I)*B_ammo_wt(I)/2000/Sys(5
)+B_tons(2,I)
20274     IF Sys(6,I)<>0 THEN R_tons(2,I)=Tot_r_rnds(I)*R_ammo_wt(I)/2000/Sys(6
)+R_tons(2,I)
20277   NEXT I
20280   RETURN
20283 !
20286 !-------------------------------------------------------------
20289 !
20292 Calc_loss:!
20295   ! INITIALIZE
20298   Nslices=15
20301   FOR J=1 TO 17
20304     B_psuv(J)=1
20307     R_psuv(J)=1
20310     Loss_r_cat(J)=0
20313     Loss_b_cat(J)=0
20316     Lo_r_cats(J)=0
20319     Lo_b_cats(J)=0
20322     FOR I=1 TO 20
20325       Lo_r_cat(I,J)=0
20328       Lo_b_cat(I,J)=0
20331       B_ex_r(I,J)=0
20334       R_ex_r(I,J)=0
20337     NEXT I
20340   NEXT J
20343   !
20346   ! CALCULATE THE AVERAGE NUMBER OF ROUNDS FIRED PER SLICE
20349   FOR I=1 TO 20     !FIRERS
20352     FOR J=1 TO 17   !TARGET CATEGORIES
20355       IF Sys(5,I)=0 THEN Rnd_red
20358       B_ex_r(I,J)=B_rnds_cat(I,J)/Sys(5,I)/Nslices
20361 Rnd_red:   !
20364       IF Sys(6,I)=0 THEN Rnd_end
20367       R_ex_r(I,J)=R_rnds_cat(I,J)/Sys(6,I)/Nslices
20370 Rnd_end: !
20373     NEXT J
20376   NEXT I
20379   !
20382   !
20385 ! CALCULATE LOSSES
```

Table 6-14. Ground combat code (continued).

```
20388 FOR Slice=1 TO Nslices
20391   FOR I=1 TO 20   !FIRERS
20394     IF B_cat(I)>0 AND B_cat(I)<=17 THEN
20397       B_firers=B_psuv(B_cat(I))*Sys(5,I)
20400        ! TEMP CODE FOR HTLD/ADEA
20403       IF I=5 THEN
20406         Glds=B_psuv(B_cat(4))*Sys(5,4)/5
20409         IF Sys(1,5)>0 THEN          !STAY VULNERABLE
20412           IF Glds>B_firers THEN Glds=B_firers
20415         ELSE
20418           IF Glds>Sys(5,5) THEN Glds=Sys(5,5)
20421         END IF
20424         B_firers=Glds
20427       END IF
20430        !ENDS TEMP CODE
20433     END IF
20436     IF R_cat(I)>0 AND R_cat(I)<=17 THEN
20439       R_firers=R_psuv(R_cat(I))*Sys(6,I)
20442     END IF
20445     FOR J=1 TO 17
20448           !
20451       R_ps(I,J)=1
20454       B_ps(I,J)=1
20457           !
20460       Red_target=1
20463       IF R_targ(J)*R_targ_vis(J)>=1 THEN Red_target=R_targ(J)*R_targ_v:
J)
20466       Blue_target=1
20469       IF B_targ(J)*B_targ_vis(J)>=1 THEN Blue_target=B_targ(J)*B_targ_
(J)
20472           !
20475       IF R_targ(J)>0 THEN
20478         Ps_r_ps=R_vul(J)*B_pk_fe(I,J)+(1-R_vul(J))*B_pk_hd(I,J)
20481         R_ps(I,J)=(1-Ps_r_ps/Red_target)^(B_firers*B_ex_r(I,J))
20484         Lossr=R_targ(J)*R_targ_vis(J)*R_psuv(J)*(1-R_ps(I,J))
20487         Lo_r_cat(I,J)=Lo_r_cat(I,J)+Lossr
20490         Lo_r_cats(J)=Lo_r_cats(J)+Lossr
20493                 !
20496       END IF
20499           !
20502       IF B_targ(J)>0 THEN
20505         Ps_b_ps=B_vul(J)*R_pk_fe(I,J)+(1-B_vul(J))*R_pk_hd(I,J)
20508         B_ps(I,J)=(1-Ps_b_ps/Blue_target)^(R_firers*R_ex_r(I,J))
20511         Lossb=B_targ(J)*B_targ_vis(J)*B_psuv(J)*(1-B_ps(I,J))
20514         Lo_b_cat(I,J)=Lo_b_cat(I,J)+Lossb
20517         Lo_b_cats(J)=Lo_b_cats(J)+Lossb
20520       END IF
20523           !
20526     NEXT J
20529   NEXT I
20532    !
20535   FOR I=1 TO 20
```

Table 6-14. Ground combat code (continued).

```
20538       FOR J=1 TO 17
20541         B_psuv(J)=B_psuv(J)*B_ps(I,J)
20544         R_psuv(J)=R_psuv(J)*R_ps(I,J)
20547              !
20550       NEXT J
20553     NEXT I
20556   NEXT Slice
20559     !
20562     !
20565   FOR J=1 TO 17
20568     Loss_b_cat(J)=B_targ(J)*B_targ_vis(J)*(1-B_psuv(J))
20571     Loss_r_cat(J)=R_targ(J)*R_targ_vis(J)*(1-R_psuv(J))
20574       !
20577   NEXT J
20580   GOSUB Roy_dumps
20583   RETURN
20586     !
20589     !--------------------------------------------------------------------
20592     !
20595 Roy_dumps:    !
20598   ! OUTPUT @Proy,5;Loss_b_cat(*)
20601   ! OUTPUT @Proy,6;Lo_b_cats(*)
20604   ! OUTPUT @Proy,7;Loss_r_cat(*)
20607   ! OUTPUT @Proy,8;Lo_r_cats(*)
20610     !NORMALIZE EACH GROUND TGT CATEGORY
20613   FOR J=1 TO 17
20616     IF Lo_b_cats(J)=0 THEN
20619       Loss_b_cat(J)=0
20622       Blossfac=0
20625     ELSE
20628       Blossfac=Loss_b_cat(J)/Lo_b_cats(J)
20631     END IF
20634     IF Lo_r_cats(J)=0 THEN
20637       Loss_r_cat(J)=0
20640       Rlossfac=0
20643     ELSE
20646       Rlossfac=Loss_r_cat(J)/Lo_r_cats(J)
20649     END IF
20652     FOR I=1 TO 20
20655       Lo_r_cat(I,J)=Lo_r_cat(I,J)*Rlossfac
20658       Lo_b_cat(I,J)=Lo_b_cat(I,J)*Blossfac
20661     NEXT I
20664   NEXT J
20667   ! OUTPUT @Proy,9;Lo_b_cat(*)
20670   ! OUTPUT @Proy,10;Lo_r_cat(*)
20673   RETURN
20676 !
20679 !----------------------------------------------------------------------
20682 !
20685 Decat_losses:!
20688   !APPORTION LOSSES TO EACH ELEMENT
20691   FOR I=1 TO 70
```

Table 6-14. Ground combat code (continued).

```
20694     IF B_cat(I)>0 AND B_cat(I)<=17 THEN
20697       IF B_targ(B_cat(I))<>0 THEN
20700         Loss_blue(I)=Sys(1,I)*Loss_b_cat(B_cat(I))/B_targ(B_cat(I))
20703       ELSE
20706         Loss_blue(I)=0
20709       END IF
20712     END IF
20715       !
20718     IF R_cat(I)>0 AND R_cat(I)<=17 THEN
20721       IF R_targ(R_cat(I))<>0 THEN
20724         Loss_red(I)=Sys(3,I)*Loss_r_cat(R_cat(I))/R_targ(R_cat(I))
20727       ELSE
20730         Loss_red(I)=0
20733       END IF
20736     END IF
20739     !
20742   NEXT I
20745   RETURN
20748    !
20751    !-----------------------------------------------------------------
20754    !
20757 Set_next_loop:!  SET DATA BEFORE CONTINUING BAND_LOOP
20760   !UPDATE SYS
20763   FOR I=1 TO 70
20766     IF Sys(1,I)<>0 THEN Sys(5,I)=Sys(5,I)-(Sys(5,I)/Sys(1,I))*Loss_blue
20769     IF Sys(3,I)<>0 THEN Sys(6,I)=Sys(6,I)-(Sys(6,I)/Sys(3,I))*Loss_red(
20772      !SYS(2,I)&SYS(4,I) ARE FILLED WITH LOSSES
20775     Sys(2,I)=Sys(2,I)+Loss_blue(I)
20778     Sys(4,I)=Sys(4,I)+Loss_red(I)
20781      !
20784     Sys(1,I)=Sys(1,I)-Loss_blue(I)
20787     Sys(3,I)=Sys(3,I)-Loss_red(I)
20790   NEXT I
20793   !UPDATE RANGE_BAND
20796   Rng_band=Rng_band-1
20799   IF Rng_band=0 AND Band_loop<Num_bands THEN
20802     PRINT
20805     PRINT "** ERROR: # OF RANGE BANDS DO NOT CORRESPOND WITH PRESENT PO
ON"
20808     Abort$="YES"
20811   END IF
20814   RETURN
20817   !
20820   !------------------------------------------------------------------
20823   !
20826 Set_return:! RESET DATA BEFORE EXITING
20829   Rng_band=Rng_band_save
20832   !RESTORE SYS(1,I)&SYS(3,I)
20835   FOR I=1 TO 70
20838     Sys(1,I)=Init_b_targets(I)
20841     Sys(3,I)=Init_r_targets(I)
20844   NEXT I
```

Table 6-14. Ground combat code (continued).

```
20847    !
20850  ASSIGN @Path1 TO *      !CLOSE ALL FILES
20853  ASSIGN @Path2 TO *
20856  ASSIGN @Path_b_fe TO *
20859  ASSIGN @Path_b_hd TO *
20862  ASSIGN @Path_r_fe TO *
20865  ASSIGN @Path_r_hd TO *
20868 SUBEND
20871 !
20874 !*********************************************************************
20877 !
20880 SUB Pgm_atrit(Fir_typ(*),N_rnds(*),Vis_rng,Terr_typ,Sens_typ(*),Cloud_ht
ust_index,Targets(*))
20883  OPTION BASE 1
20886  DIM Prob_desg(6,4),Sskp_clgp(70),Clgp_mask(70),Sskp(2,70)
20889  DIM P_surv(70),N_rnds_fired(2),Psurv_tf(2,70)
20892  DIM Tgts_avail(70),Tgt_mask(2,70),W_sum(2)
20895 !
20898  COM /Pgm/ Tgt_value(2,70),Tgt_mask1(2,70),Terr_factor(4),Prob_dustabort
,4),Clgp_msk_ns(70),Clgp_msk_gl(70),Clgp_msk_rp(70)
20901  COM /Pgm/ Prob_dsg_ns(6,4),Prob_dsg_gl(6,4),Prob_dsg_rp(6,4),Sskp_ns(70
Sskp_gl(70),Sskp_rp(70)
20904 !
20907 ! NOTE -- CLGP IS NOT LOADED BUT WILL LATER BE REPLACED WITH ALL 1'S.
20910 !
20913  Dcdisk$=":9134,704,0"
20916  ASSIGN @Psskp TO "SSKP"&Dcdisk$
20919  ENTER @Psskp,1;Sskp(*)
20922  ASSIGN @Psskp TO *
20925 !
20928 ! SET THE CLOUD INDEX
20931 !
20934  Cloud=Cloud_ht*3.28
20937  SELECT Cloud
20940  CASE <1500
20943    Cld_ndx=1
20946  CASE 1500 TO 2000
20949    Cld_ndx=2
20952  CASE 2000 TO 2500
20955    Cld_ndx=3
20958  CASE 2500 TO 3000
20961    Cld_ndx=4
20964  CASE 3000 TO 4500
20967    Cld_ndx=5
20970  CASE >4500
20973    Cld_ndx=6
20976  END SELECT
20979 !
20982 !    SET VISIBILITY INDEX
20985 !
20988  Vis_ndx=Vis_rng
20991  Nslices=15
```

Table 6-14. Ground combat code (continued).

```
20994  Nfirers=2 !  USING CLGP AND GAMP
20997  Ntargets=70
21000 !
21003 !Set probability of survival for all targets at 1 to start
21006  FOR Target=1 TO Ntargets
21009    P_surv(Target)=1
21012    FOR Firer=1 TO Nfirers
21015      Psurv_tf(Firer,Target)=1
21018    NEXT Firer
21021  NEXT Target
21024    !
21027  N_tot=0
21030  FOR Init=1 TO 2
21033    N_rnds_fired(Init)=0
21036  NEXT Init
21039 !
21042 !Attrition calculations.
21045 !SET ARRAYS FOR PGM'S
21048 !
21051  IF Fir_typ(1)=0 THEN Set_new_pgm
21054 !
21057 ! IF THIS IS CLGP THEN LOAD APPROPRIATE PK'S
21060  IF Sens_typ(1)<0 OR Sens_typ(1)>2 THEN Set_new_pgm
21063 !
21066  SELECT Sens_typ(1)
21069  CASE 0  !NO SENSOR AVAILABLE
21072    FOR I=1 TO 70
21075      Clgp_mask(I)=Clgp_msk_ns(I)
21078      Sskp_clgp(I)=Sskp_ns(I)
21081    NEXT I
21084    FOR I=1 TO 6
21087      FOR J=1 TO 4
21090        Prob_desg(I,J)=Prob_dsg_ns(I,J)
21093      NEXT J
21096    NEXT I
21099  CASE 1  !GLLD
21102    FOR I=1 TO 70
21105      Clgp_mask(I)=Clgp_msk_gl(I)
21108      Sskp_clgp(I)=Sskp_gl(I)
21111    NEXT I
21114    FOR I=1 TO 6
21117      FOR J=1 TO 4
21120        Prob_desg(I,J)=Prob_dsg_gl(I,J)
21123      NEXT J
21126    NEXT I
21129  CASE 2,3   !RPV
21132    FOR I=1 TO 70
21135      Clgp_mask(I)=Clgp_msk_rp(I)
21138      Sskp_clgp(I)=Sskp_rp(I)
21141    NEXT I
21144    FOR I=1 TO 6
21147      FOR J=1 TO 4
```

Table 6-14. Ground combat code (continued).

```
21150         Prob_desg(I,J)=Prob_dsg_rp(I,J)
21153       NEXT J
21156     NEXT I
21159   END SELECT
21162 !
21165 Set_new_pgm: !
21168   FOR Pgm=1 TO 2
21171     IF Fir_typ(Pgm)=0 THEN N_pgm
21174     FOR Target=1 TO Ntargets
21177          !SET SENSOR MASK
21180       IF Pgm=1 THEN
21183         S_mask=Clgp_mask(Target)
21186       ELSE
21189         S_mask=1        !ALL SENSOR MASKS=1 FOR AMSAA PGM'S
21192       END IF
21195          !SET TARGET MASK WITH MISSION VALUE
21198       Tgt_mask(Pgm,Target)=Tgt_mask1(Pgm,Target)*S_mask
21201          !SET SSKP
21204       IF Pgm=1 THEN Sskp(1,Target)=Sskp_clgp(Target)
21207     NEXT Target
21210 N_pgm:NEXT Pgm
21213 !SET PK'S
21216   FOR Target=1 TO Ntargets
21219     P_surv(Target)=1
21222   NEXT Target
21225 !
21228 !CALCULATE THE WEIGHT SUM FOR EACH FIRER
21231   FOR Slice=1 TO Nslices
21234     FOR Target=1 TO Ntargets
21237       FOR Firer=1 TO Nfirers
21240         Psurv_tf(Firer,Target)=1
21243       NEXT Firer
21246     NEXT Target
21249     FOR Pgm=1 TO 2
21252       W_sum(Pgm)=0
21255       IF Fir_typ(Pgm)=0 THEN End_wt
21258       FOR Target=1 TO Ntargets
21261         Des=1
21264         IF Pgm=1 THEN Des=Sskp(1,Target)
21267         W_sum(Pgm)=W_sum(Pgm)+Tgt_value(Pgm,Target)*P_surv(Target)*Target
1,Target)*Des*Tgt_mask(Pgm,Target)
21270           !
21273 N_targ:NEXT Target
21276 End_wt:NEXT Pgm
21279 !
21282 !
21285 !NEW BEGIN THE FIRER LOOP FOR THIS SLICE
21288     FOR Firer=1 TO Nfirers
21291       IF Fir_typ(Firer)=0 THEN N_firer
21294         ! SET LOCAL PARAMETERS FOR THIS FIRER
21297       IF Firer>1 THEN New_pgm
21300       Lase_reliab=.8
```

Table 6-14. Ground combat code (continued).

```
21303      P_desg=Prob_desg(Cld_ndx,Vis_ndx)
21306      P_dustabort=Prob_dustabort(Dust_index,Vis_ndx)
21309      Terr_degrd=Terr_factor(Terr_typ)
21312      GOTO Ct_kills
21315        ! SET PARAMETERS FOR GAMP. ALL SET TO NO EFFECT BECAUSE
21318        ! OF AMSAA DATA.
21321 New_pgm:Lase_reliab=1
21324      P_desg=1
21327      P_dustabort=0
21330      Terr_degrd=1
21333 Ct_kills:FOR Target=1 TO Ntargets
21336        N_tgts=Targets(1,Target)*P_surv(Target)
21339        IF Tgt_mask(Firer,Target)*N_tgts>0 THEN
21342          Pk=Sskp(Firer,Target)
21345              ! SET SWITCH FOR SMART DESIGNATOR DISCRIMINATING
21348              ! BETWEEN HIGH PK AND LOW PK TARGETS
21351          Des=1
21354          IF Firer=1 THEN Des=Sskp(Firer,Target)
21357          IF W_sum(Firer)<=0 THEN N_firer
21360          Nrnds=((N_tgts*Tgt_value(Firer,Target)*Des)/W_sum(Firer))*(N_rn
(Firer)/Nslices)
21363          N_rnds_fired(Firer)=N_rnds_fired(Firer)+Nrnds
21366          Nrnds=Nrnds*Terr_degrd*(1-P_dustabort)*P_desg
21369          Smk=1
21372          IF (N_tgts*Smk)<1 THEN
21375            Div=1
21378          ELSE
21381            Div=N_tgts*Smk
21384          END IF
21387          Psurv_tf(Firer,Target)=(1-Pk*Lase_reliab/Div)^Nrnds
21390        END IF
21393      NEXT Target
21396 N_firer:NEXT Firer
21399        ! UPDATE OVERALL PROBABILITY OF SURVIVAL FOR EACH TARGET
21402    FOR Target=1 TO Ntargets
21405      FOR Firer=1 TO Nfirers
21408        P_surv(Target)=P_surv(Target)*Psurv_tf(Firer,Target)
21411      NEXT Firer
21414    NEXT Target
21417  NEXT Slice
21420    !
21423    !Talley the kills
21426    !
21429  FOR Target=1 TO Ntargets
21432    Targets(2,Target)=(1-P_surv(Target))*Targets(1,Target)
21435  NEXT Target
21438 End_pgm:SUBEND
21441 !
21444 !************************************************************************
21447 SUB Missn_tmpls(Side,I,T,K,Unit_no(*),Hfile(*),Listop)
21450  OPTION BASE 1
21453  DIM Unitname$[16],Cdesc$[28]
```

Table 6-14. Ground combat code (continued).

```
21456  INTEGER J,X
21459   ! ADDED TO PREVENT 2ND 3 HRS FROM APPEARING
21462   ! REMOVE FOLLOWING  "IF" STATEMENT WHEN NO LONGER REQUIRED
21465  IF I=2 THEN
21468    FOR L=1 TO 8
21471      Hfile(2,K,L)=Hfile(1,K,L)
21474    NEXT L
21477    GOTO End_missn
21480  END IF
21483  FOR X=1 TO 2
21486    FOR J=1 TO 12
21489      IF Hfile(X,J,9)=Side AND Hfile(X,J,10)=T THEN
21492        FOR L=1 TO 8
21495          Hfile(I,K,L)=Hfile(X,J,L)
21498        NEXT L
21501        PRINTER IS 1
21504        GOSUB Prt_missn_tmpls
21507        IF Listop=2 THEN
21510          PRINTER IS 702
21513          GOSUB Prt_missn_tmpls
21516        ELSE
21519          PRINTER IS 702
21522        END IF
21525        GOTO End_missn
21528      END IF
21531    NEXT J
21534  NEXT X
21537 Start_missn: !
21540  PRINTER IS 1
21543  GOSUB Prt_missn_tmpls
21546 Chg_values:!
21549  INPUT "FIELD #,NEW VALUE  TO CHANGE (9,9 TO END)",Fieldn,Nvalue
21552 Sel_field: !
21555  SELECT Fieldn
21558  CASE 1 TO 8
21561    Hfile(I,K,Fieldn)=Nvalue
21564  CASE 9
21567    IF Listop=2 THEN
21570      PRINTER IS 702
21573      GOSUB Prt_missn_tmpls
21576    ELSE
21579      PRINTER IS 702
21582    END IF
21585    DISP "CHANGE ALL ";
21588    IF Side=1 THEN
21591      Cdesc$="BLUE "&Cdesc$
21594    ELSE
21597      Cdesc$="RED "&Cdesc$
21600    END IF
21603    DISP Cdesc$;
21606    DISP "S TO THE SAME VALUES (Y/N)";
21609    INPUT Ans$
```

Table 6-14. Ground combat code (continued).

```
21612     IF Ans$="Y" THEN
21615       Hfile(I,K,9)=Side
21618       Hfile(I,K,10)=T
21621     END IF
21624     GOTO End_missn
21627   CASE ELSE
21630     GOTO Chg_values
21633   END SELECT
21636   GOTO Start_missn
21639 Prt_missn_tmpls: !
21642   ASSIGN @Name TO "NAMEFILE:9134,704,0"
21645   ENTER @Name,Unit_no(K);Unitname$
21648   ASSIGN @Name TO *
21651   PRINT USING "/,6A,3D,3A,16A,#";"UNIT #",Unit_no(K)," - ",Unitname$
21654   SELECT T
21657   CASE 1,11
21660     Cdesc$="COMBAT "
21663   CASE 2,12
21666     Cdesc$="ARTILLERY "
21669   CASE 3,13
21672     Cdesc$="AIR DEFENSE "
21675   CASE 4,14
21678     Cdesc$="FARRP "
21681   CASE 5,15
21684     Cdesc$="COMMAND POST "
21687   CASE 6,16
21690     Cdesc$="ENGINEER "
21693   CASE 7,17
21696     Cdesc$="SUPPLY "
21699   CASE 8,18
21702     Cdesc$="MAINTENANCE "
21705   CASE 9,19
21708     Cdesc$="BRIDGE "
21711   CASE 10,20
21714     Cdesc$="COMMO/EW SITE "
21717   END SELECT
21720   SELECT Side
21723   CASE 1
21726     SELECT T
21729     CASE 1 TO 10
21732       Cdesc$=Cdesc$&"BATTALION"
21735     CASE 11 TO 20
21738       Cdesc$=Cdesc$&"COMPANY"
21741     END SELECT
21744   CASE 2
21747     SELECT T
21750     CASE 1 TO 10
21753       Cdesc$=Cdesc$&"REGIMENT"
21756     CASE 11 TO 20
21759       Cdesc$=Cdesc$&"BATTALION"
21762     END SELECT
21765   END SELECT
```

Table 6-14. Ground combat code (continued).

```
21768  PRINT Cdesc$&", ";
21771  SELECT I
21774  CASE 1
21777    PRINT "1ST 3 HOURS"
21780  CASE 2
21783    PRINT "2ND 3 HOURS"
21786  END SELECT
21789  PRINT USING "/,8(2X,7A)";"  (1)  ","  (2)  ","  (3)  ","  (4)  ","  (5)
","  (6)  ","  (7)  ","  (8)  "
21792  PRINT USING "8(2X,7A)";"  MIFT ","  MDFT "," MIFDT "," MDFDT "," MFIRE
"MDFIRE ","  MVUL "," MDVUL "
21795  PRINT USING "8(2X,7A)";"-------","-------","-------","-------","-------
"-------","-------","-------"
21798  PRINT USING "8(3X,1D.4D)";Hfile(I,K,1),Hfile(I,K,2),Hfile(I,K,3),Hfile(
K,4),Hfile(I,K,5),Hfile(I,K,6),Hfile(I,K,7),Hfile(I,K,8)
21801  RETURN
21804 End_missn: !
21807 SUBEND
21810 !**********************************************************************
21813 SUB Helo_range(Cell(*),Helo_mis(*),Stnd_off_rg(*))
21816 OPTION BASE 1
21819!--------------------------------------------------------------
21822!  CALCULATE THE RANGES FROM HELO TO GROUND & HELO TO HELO
21825!--------------------------------------------------------------
21828 INTEGER Side,Side_def
21831 !
21834 COM /Helo_info/ Btl_rg,Rg_avg(2,3,20),Rg_avg_pd(2,3,5),Df_ammo(2),Df_fir
dist(2,20,3),Df_pk_helo(2,20,3,2),INTEGER Df_sen_ptr(2,20),Df_muni_ptr(2,20)
21837 !
21840  FOR Side=1 TO 2
21843    Side_def=(Side MOD 2)+1        !CALCULATE DEFENDING SIDE
21846    FOR I=1 TO 3               ! COMPUTE HELICOPTER TO HELICOPTER RANGES
21849      IF Cell(Side,1,I)<=0 THEN GOTO No_rg
21852      FOR J=1 TO 3
21855        SELECT Helo_mis(Side,I)
21858        CASE =1
21861          SELECT Helo_mis(Side_def,J)
21864          CASE =0    ! NO ENEMY HELOS FLYING FOR THIS BATTLE
21867            Rg_avg(Side,I,J+17)=0
21870          CASE =1    ! DIRECT SUPPORT<>DIRECT SUPPORT
21873            Rg_avg(Side,I,J+17)=ABS(Stnd_off_rg(Side,I)+Stnd_off_rg(Side_
f,J)-Btl_rg)
21876          CASE =2    ! DIRECT SUPPORT<>HELO TO HELO
21879            Rg_avg(Side,I,J+17)=Stnd_off_rg(Side_def,J)
21882          CASE =3    ! DIRECT SUPPORT<>SEAD
21885            Rg_avg(Side,I,J+17)=SQR((Stnd_off_rg(Side,I)-Btl_rg)^2+(Stnd_
f_rg(Side_def,J))^2)
21888          END SELECT
21891        CASE =2
21894          SELECT Helo_mis(Side_def,J)
21897          CASE =0    ! NO ENEMY HELOS
21900            Rg_avg(Side,I,J+17)=0
```

Table 6-14. Ground combat code (continued).

```
            CASE =1    ! HELO TO HELO<>DIRECT SUPPORT
              Rg_avg(Side,I,J+17)=Stnd_off_rg(Side,I)
            CASE =2    ! HELO TO HELO<>HELO TO HELO
              Rg_avg(Side,I,J+17)=(Stnd_off_rg(Side,I)+Stnd_off_rg(Side_def
))/2
            CASE =3    ! HELO TO HELO<>SEAD
              Rg_avg(Side,I,J+17)=Stnd_off_rg(Side,I)
            END SELECT
          CASE =3
            SELECT Helo_mis(Side_def,J)
            CASE =0    ! NO ENEMY HELOS FLYING
              Rg_avg(Side,I,J+17)=0
            CASE =1    ! SEAD<>DIRECT SUPPORT
              Rg_avg(Side,I,J+17)=SQR((Stnd_off_rg(Side_def,J)-Btl_rg)^2+(S
d_off_rg(Side,I))^2)
            CASE =2    ! SEAD<>HELO TO HELO
              Rg_avg(Side,I,J+17)=Stnd_off_rg(Side_def,J)
            CASE =3    ! SEAD<>SEAD
              Rg_avg(Side,I,J+17)=(Btl_rg/2)+(SQR(Btl_rg*Btl_rg+(Stnd_off_r
Side,I)+Stnd_off_rg(Side_def,J))^2))/2
            END SELECT
          CASE ELSE    ! NO ATTACKING HELOS OF THIS TYPE
            Rg_avg(Side,I,J+17)=0
          END SELECT
        NEXT J
 No_rg:NEXT I
! COMPUTE RANGE BETWEEN GROUND AND HELICOPTER
      Tot_en_helos=Cell(Side_def,1,1)+Cell(Side_def,1,2)+Cell(Side_def,1,3)
        !TOTAL NO. OF ENEMY HELOS
      FOR I=1 TO 3
        IF Helo_mis(Side,I)<=0 OR Cell(Side,1,I)<=0 THEN !NO ATTACKING
          FOR K=1 TO 17                                  !HELOS OF TYPE I
            Rg_avg(Side,I,K)=0
          NEXT K
          GOTO Grd_hlo
        END IF
        SELECT Helo_mis(Side,I)
        CASE =1        ! DIRECT SUPPORT
          FOR K=1 TO 17
            Rg_avg(Side,I,K)=Stnd_off_rg(Side,I)
          NEXT K
        CASE =2        ! HELO TO HELO
          Rg_avg(Side,I,1)=0
          IF Tot_en_helos<=0 THEN GOTO H_h_rg
          FOR J=1 TO 3          ! COMPUTE WEIGHTED AVERAGE DEPENDING UPON
                                ! PERCENTAGE OF ENEMY HELICOPTERS ON A
                                ! PARTICULAR MISSION
            SELECT Helo_mis(Side_def,J)
            CASE 1
              Rg_avg(Side,I,1)=Rg_avg(Side,I,1)+(SQR((Stnd_off_rg(Side_def,
-Btl_rg)^2+(Stnd_off_rg(Side,I)^2)))*(Cell(Side_def,1,J)/Tot_en_helos)
            CASE 2
```

Table 6-14. Ground combat code (continued).

```
22044              Rg_avg(Side,I,1)=Rg_avg(Side,I,1)+(Btl_rg/2+((Stnd_off_rg(Sid
I)+Stnd_off_rg(Side_def,J))/4))*(Cell(Side_def,1,J)/Tot_en_helos)
22047            CASE 3
22050              Rg_avg(Side,I,1)=Rg_avg(Side,I,1)+(Btl_rg+SQR(ABS(Stnd_off_rg
ide,I)^2-Stnd_off_rg(Side_def,J)^2)))*(Cell(Side_def,1,J)/Tot_en_helos)
22053            END SELECT
22056          NEXT J
22059 H_h_rg:FOR K=2 TO 17
22062            Rg_avg(Side,I,K)=Rg_avg(Side,I,1)
22065          NEXT K
22068        CASE =3       ! SEAD
22071          FOR K=1 TO 17
22074            Rg_avg(Side,I,K)=Stnd_off_rg(Side,I)
22077          NEXT K
22080        END SELECT
22083 Grd_hlo:NEXT I
22086      FOR I=1 TO 3
22089        FOR J=1 TO 4!STORE THE RANGES FOR THE PROB OF DETECTION CATEGORIES
22092          Rg_avg_pd(Side,I,J)=Rg_avg(Side,I,1)
22095        NEXT J
22098        Rg_avg_pd(Side,I,5)=0
22101        IF Tot_en_helos>0 THEN
22104          FOR J=18 TO 20   !LOOP ON HELO TGT CATEGORIES
22107            Rg_avg_pd(Side,I,5)=Rg_avg_pd(Side,I,5)+Rg_avg(Side,I,J)*(Cell(
de_def,1,J-17)/Tot_en_helos)
22110          NEXT J
22113        END IF
22116      NEXT I
22119   NEXT Side
22122   SUBEND
22125!**********************************************************************
22128 SUB Helo_kills(Cell(*),Target(*),Ad_ammo(*),Terr,Atk_prof(*),Helo_mis(*)
ay_nite,Time_step,P_def_ray(*),Arty(*),Veh_ada(*),Hnd_ada(*),Stnd_off_rg(*),Vi
22131!*********************************************************************
22134!SUBROUTINE HELO_KILLS CALCULATES KILLS BY HELICOPTERS AGAINST GROUND *
22137!ELEMENTS AND ENEMY HELICOPTERS, AND HELICOPTER LOSSES BY AD WEAPONS   *
22140!*********************************************************************
22143!
22146   OPTION BASE 1
22149   DIM Red_blue$[4],No_targets(2,20),Cell_save(2,3),Helo_loss(2,3)
22152   DIM Target_loss(2,20),Helo_pd_targ(3,20),Helo_pd_tgt(5),Ut(3),Plos(2,3)
22155   DIM Ad_pd_helo(7,3),Ad_avg_fired(7,3),Ad_pk_helo(2,7,3)
22158   DIM Helo_avg_fired(3,20),Helo_fire_dist(3,20),Helo_act_fired(3,20)
22161   DIM Helo_pk_targ(3,20),Ad_fire_dist(7,3),Ad_act_fired(7,3)
22164   DIM Pop_tstep(3),No_pd_tgt(5),Adust(2),Ad_ele(7),Te_firer(3)
22167   DIM Air_rnds_avl(2,3),Helo_rnds_avl(2,3),Target_psrv(2,20),Helo_psrv(2,
22170   DIM Pct_force_fe(2),Pct_force_hd(2),Pct_mm(2),Pct_non_mm(2)
22173   DIM Df_pd_helo(20,3),Df_avg_fired(20,3),Df_act_fired(20,3),Dfinf(2,3)
22176   DIM Dftbar(2,3)
22179   INTEGER I,J,K,L,M,Atmos,Side_def,H_lase(2),Lase,Tim_step,Ctg,Muni(3),En
uni(3)
22182  !
```

Table 6-14. Ground combat code (continued).

```
22185  COM /Helo_attrite/ Helo_load(2,3,3),Pd_fe_inf_a(2,3,5),Pd_fe_inf_b(2,3.
,Pd_fe_inf_c(2,3,5),Pd_hd_inf_a(2,3,5),Pd_hd_inf_b(2,3,5)
22188  COM /Helo_attrite/ Pd_hd_inf_c(2,3,5),Pd_fe_tbar_a(2,3,5),Pd_fe_tbar_b
3,5),Pd_fe_tbar_c(2,3,5),Pd_hd_tbar_a(2,3,5),Pd_hd_tbar_b(2,3,5)
22191  COM /Helo_attrite/ Pd_hd_tbar_c(2,3,5),Pd_rmin(2,3,8),Pd_rmax(2,3,8),Pk
e_a(2,3,3,20),Pk_fe_b(2,3,3,20),Pk_fe_c(2,3,3,20),Pk_hd_a(2,3,3,20)
22194  COM /Helo_attrite/ Pk_hd_b(2,3,3,20),Pk_hd_c(2,3,3,20),Pk_rmin(2,3,3),P
rmax(2,3,3),Np(2,3,3),Fm(2,3,3),Tm(2,3,3),Te(2,3,3)
22197  COM /Helo_attrite/ Plos_alpha(2,3),Plos_beta(2,3),Pd_inf_ad_a(2,7,2),Pd
nf_ad_b(2,7,2),Pd_inf_ad_c(2,7,2),Pd_tbar_ad_a(2,7,2),Pd_tbar_ad_b(2,7,2)
22200  COM /Helo_attrite/ Pd_tbar_ad_c(2,7,2),Pd_ad_rmin(2,7,8),Pd_ad_rmax(2,7
),Pk_ad_a(2,7,2),Pk_ad_b(2,7,2),Pk_ad_c(2,7,2),Pk_ad_rmin(2,7),Pk_ad_rmax(2,7)
22203  COM /Helo_attrite/ Rnd_wt(2,7),Rnds(2,7),Fad(2,7),Pd_inf_df_a(2,2,2),Pd
nf_df_b(2,2,2),Pd_inf_df_c(2,2,2),Pd_tbar_df_a(2,2,2),Pd_tbar_df_b(2,2,2)
22206  COM /Helo_attrite/ Pd_tbar_df_c(2,2,2),Pd_df_rmin(2,2,8),Pd_df_rmax(2,2
),Df_rnds_eng(2,2),F_df(2,2)
22209  COM /Helo_attrite/ INTEGER Mast_mnt(2,3),Tgt_pref(2,3,20),Ad_pref(2,7,3
Pd_cat(2,20)
22212!
22215  COM /Direct_fire/ B_cat(70),R_cat(70),B_sen_d(70),B_sen_n(70),R_sen_d(7
,R_sen_n(70),B_ammo_wt(20),R_ammo_wt(20)
22218!
22221  COM /Helo_info/ Btl_r .! g_avg(2,3,20),Rg_avg_pd(2,3,5),Df_ammo(2),Df_fi
_dist(2,20,3),Df_pk_helo(2,20,3,2),INTEGER Df_sen_ptr(2,20),Df_muni_ptr(2,20)
22224!
22227  Atmos=Day_nite*4+Vis            !COMPUTE ATMOSPHERIC CONDITIONS
22230  Adust(1)=.5
22233  Adust(2)=1.0
22236  Time_step=Time_step*60     !CHANGE TO SECONDS
22239  PRINT
22242  PRINT USING "22X,11A,19X,4A";"    BLUE    ";" RED"
22245  PRINT USING "17X,52A";"HELO 1  HELO 2  HELO 3     HELO 1  HELO 2  HELO
22248  PRINT USING "17X,52A";"------  ------  ------     ------  ------  -----
22251  PRINT USING Fmt1;"STANDOFF RANGE:";Stnd_off_rg(1,1);Stnd_off_rg(1,2);St
_off_rg(1,3);Stnd_off_rg(2,1);Stnd_off_rg(2,2);Stnd_off_rg(2,3)
22254  PRINT USING Fmt1;"HELO MSN:       ";Helo_mis(1,1);Helo_mis(1,2);Helo_mis
,3);Helo_mis(2,1);Helo_mis(2,2);Helo_mis(2,3)
22257  PRINT USING Fmt1;"ATK_PROF:       ";Atk_prof(1,1);Atk_prof(1,2);Atk_prof
,3);Atk_prof(2,1);Atk_prof(2,2);Atk_prof(2,3)
22260  PRINT USING Fmt2;"#  HELOS:       ";Cell(1,1,1),Cell(1,1,2),Cell(1,1,3),
ll(2,1,1),Cell(2,1,2),Cell(2,1,3)
22263 Fmt1:IMAGE  15A,3(2X,6D),3X,3(2X,6D)
22266 Fmt2:IMAGE  15A,3(2X,3D.2D),3X,3(2X,3D.2D)
22269  PRINT
22272!----------------------------------------------------------------------
22275  FOR Side=1 TO 2          !1=BLUE   2=RED
22278    Ad_wt_avl(Side)=Ad_ammo(Side)*2000  !CHANGE AD AMMO AVAIL-TONS TO LBS
22281    Df_wt_avl(Side)=Df_ammo(Side)*2000  !CHANGE DF AMMO AVAIL-TONS TO LBS
22284    FOR I=1 TO 70          !PLACE 70 TARGETS INTO 20 CATEGORIES
22287      IF Side=1 THEN Ctg=B_cat(I)
22290      IF Side=2 THEN Ctg=R_cat(I)
22293      Target(Side,2,I)=Target(Side,1,I)
```

Table 6-14. Ground combat code (continued).

```
22296       IF Ctg>0 AND Ctg<=17 THEN No_targets(Side,Ctg)=No_targets(Side,Ctg)
arget(Side,1,I)
22299     NEXT I
22302!------------------------------------------------------------------------
22305     FOR I=1 TO 3        !PLACE HELO TARGETS INTO LAST 3 CATEG (18,19,20)
22308       No_targets(Side,I+17)=Cell(Side,1,I)  !MAP IN ONE-TO-ONE
22311       IF No_targets(Side,I+17)<.1 THEN No_targets(Side,I+17)=0
22314       Cell(Side,2,I)=No_targets(Side,I+17)    !INIT REMAINING # OF TGT HEL
22317     NEXT I
22320   NEXT Side
22323!------------------------------------------------------------------------
22326   FOR Side=1 TO 2
22329     Side_def=(Side MOD 2)+1      !DEFENDING SIDE
22332 !     PRINT       !COMMENT THESE PRINT STATEMENTS OUT, BUT LEAVE IN CODE!
22335 !     IF Side=1 THEN PRINT "     *** BLUE HELO RANGES ***"
22338 !     IF Side=2 THEN PRINT "     *** RED  HELO RANGES ***"
22341 !     FOR I=1 TO 3
22344 !       PRINT "Helo ";I;"HELO TO GRND RG:";Rg_avg(Side,I,1)
22347 !       PRINT TAB(9);"HELO TO HELO RG:";Rg_avg(Side,I,18);Rg_avg(Side,I,1
;Rg_avg(Side,I,20)
22350 !       PRINT TAB(9);"AVG RG TO HELOS FOR PROB OF DET:";Rg_avg_pd(Side,I.
22353 !       PRINT
22356 !     NEXT I
22359!------------------------------------------------------------------------
22362 Ful_exp:Totnum=0    !CALCULATE % OF FORCE FULLY EXPOSED & HULL DEFILADE
22365     Totden=0            !FOR BOTH SIDES
22368     FOR I=1 TO 70
22371       Totnum=Totnum+Target(Side,2,I)*P_def_ray(Side,I)
22374       Totden=Totden+Target(Side,2,I)
22377     NEXT I
22380     Pct_force_fe(Side)=Totnum/Totden .
22383     Pct_force_hd(Side)=1.0-Pct_force_fe(Side)
22386 !                    CALCULATE % OF TGT HELOS MAST MOUNTED & NON-MAST-MNT
22389     Totden=Cell(Side,1,1)+Cell(Side,1,2)+Cell(Side,1,3)
22392     IF Totden>0 THEN
22395       Totnum=0
22398       FOR I=1 TO 3   !CALCULATE NO. OF HELOS NON MAST MOUNTED
22401         Totnum=Totnum+Cell(Side,1,I)*Mast_mnt(Side,I)
22404       NEXT I
22407       Pct_mm(Side)=Totnum/Totden             !PERCENT MAST MOUNTED
22410       Pct_non_mm(Side)=1.0-Pct_mm(Side)     !PERCENT NON MAST MOUNTED
22413     ELSE      !NO TARGET HELOS
22416       Pct_mm(Side)=0
22419       Pct_non_mm(Side)=0
22422     END IF
22425!------------------------------------------------------------------------
22428     FOR I=1 TO 3            !LOOP ON HELO
22431       Air_rnds_avl(Side,I)=0
22434       Helo_rnds_avl(Side,I)=0
22437       SELECT Atk_prof(Side,I)  !CALCULATE NO. OF ROUNDS AVAILABLE
22440       CASE 1    !MSL LOAD
22443         Helo_rnds_avl(Side,I)=Helo_load(Side,I,1)*Cell(Side,1,I)
```

Table 6-14. Ground combat code (continued).

```
22446       CASE 2    !MSL/GUN   (MSL LOAD FIRST)
22449         Helo_rnds_avl(Side,I)=Helo_load(Side,I,1)*.5*Cell(Side,1,I)
22452       CASE 3    !GUN
22455         Helo_rnds_avl(Side,I)=Helo_load(Side,I,2)*Cell(Side,1,I)
22458       CASE 4    !AIR-TO-AIR
22461         Air_rnds_avl(Side,I)=Helo_load(Side,I,3)*Cell(Side,1,I)
22464       CASE 5    !AIR/MSL
22467         Air_rnds_avl(Side,I)=Helo_load(Side,I,3)*Cell(Side,1,I)
22470         Helo_rnds_avl(Side,I)=Helo_load(Side,I,1)*Cell(Side,1,I)
22473       CASE 6    !AIR/MSL/GUN
22476         Air_rnds_avl(Side,I)=Helo_load(Side,I,3)*Cell(Side,1,I)
22479         Helo_rnds_avl(Side,I)=Helo_load(Side,I,1)*.5*Cell(Side,1,I)
22482       CASE 7    !AIR/GUN
22485         Air_rnds_avl(Side,I)=Helo_load(Side,I,3)*Cell(Side,1,I)
22488         Helo_rnds_avl(Side,I)=Helo_load(Side,I,2)*Cell(Side,1,I)
22491       END SELECT
22494     NEXT I
22497     GOSUB Line_of_sight   !CALC PROB OF LINE OF SIGHT BETWEEN GRND & HELO
22500     GOSUB Ad_pk_helo      !CALCULATE PROB. OF KILL BY AD WEAPONS
22503   NEXT Side
22506!
22509!------------------------------------------------------------------------
22512   FOR Tim_step=1 TO 2      !LOOP ON 15 MINUTE TIME STEP
22515     FOR Side=1 TO 2
22518       FOR J=1 TO 20      !INITIALIZE TARGETS' PROB OF SURVIVAL ARRAY
22521         Target_psrv(Side,J)=1.0
22524       NEXT J
22527       FOR J=1 TO 3       !INITIALIZE HELOS' PROB OF SURVIVAL ARRAY
22530         Helo_psrv(Side,J)=1.0
22533       NEXT J
22536       IF Side=1 THEN Red_blue$="BLUE"
22539       IF Side=2 THEN Red_blue$="RED"
22542       IF Tim_step=1 THEN GOTO Skp_chk   !REST IS FOR 2ND TIMESTEP
22545       FOR I=1 TO 3
22548         IF Cell(Side,1,I)<=0 THEN Nxt_lup
22551         Cell_save(Side,I)=Cell(Side,2,I)    !SAVE REMAINING NO. OF HELOS
22554         IF Cell(Side,2,I)<.8*Cell(Side,1,I) THEN
22557           IF I=3 AND Cell(Side,1,I)>0 THEN GOTO Nxt_lup 'NO MSG IF SCTS LE
T
22560           Cell(Side,2,I)=0   !TEMPORARILY ZERO OUT HELOS SO AD & ENEMY HEL
S WON'T SHOOT AT IT
22563           No_targets(Side,I+17)=0
22566           PRINT
22569           PRINT "   ";Red_blue$;" HELICOPTER MISSION ":I:"ABORTED DUE TO E
CESSIVE LOSSES."
22572         ELSE     !CHECK FOR ENOUGH MUNITIONS FOR THIS TIMESTEP
22575           IF Cell(Side,1,I)=0 THEN GOTO Nxt_lup
22578           IF Helo_mis(Side,I)=1 OR Helo_mis(Side,I)=3 THEN
22581             IF Helo_rnds_avl(Side,I)<=0 THEN
22584               IF Atk_prof(Side,I)<>2 AND Atk_prof(Side,I)<>6 THEN
22587                 PRINT
22590                 PRINT "   NO MORE ":
```

Table 6-14. Ground combat code (continued).

```
22593                   IF Atk_prof(Side,I)=1 OR Atk_prof(Side,I)=5 THEN PRINT "(
UND MISSILES AVAILABLE FOR ";Red_blue$;" HELO ";I;".  WILL RETURN TO BASE."
22596                   IF Atk_prof(Side,I)=3 OR Atk_prof(Side,I)=7 THEN PRINT "(
 ROUNDS AVAILABLE FOR ";Red_blue$;" HELO ";I;".  WILL RETURN TO BASE."
22599                   Cell(Side,2,I)=0
22602                   No_targets(Side,I+17)=0
22605                 END IF
22608               END IF
22611             ELSE   !CHECK FOR ENOUGH AIR MISSILES FOR AIR MISSION
22614               IF Air_rnds_avl(Side,I)<=0 THEN
22617                 PRINT
22620                 PRINT "   NO MORE AIR MISSILES AVAILABLE FOR ";Red_blue$;"
LO ";I;".  WILL RETURN TO BASE."
22623                 Cell(Side,2,I)=0
22626                 No_targets(Side,I+17)=0
22629               END IF
22632             END IF
22635           END IF
22638 Nxt_lup:NEXT I
22641! IF PRIMARY MSN IS D.S. OR SEAD, MUNITIONS IS MISSILES AND SCT AND AH1 E
ST, THEN SCTS LASING
22644 Skp_chk:H_lase(Side)=1
22647       IF (Helo_mis(Side,1)<>2) AND (Atk_prof(Side,1)=1 OR Atk_prof(Side,1
5) AND Cell(Side,1,3)>0 AND Cell(Side,2,1)>0 THEN
22650         IF Cell(Side,2,3)<.8*Cell(Side,1,3) THEN    !CHECK FOR ENUF SCTS
22653           PRINT
22656           PRINT "   INSUFFICIENT ";Red_blue$;" SCOUTS TO CONTINUE LASING.
REMAINING AH1 WILL OPERATE          AUTONOMOUS."
22659           Cell(Side,2,3)=0        !ZERO OUT SCOUTS
22662           No_targets(Side,20)=0
22665         ELSE
22668           H_lase(Side)=3      !SCOUTS WILL LASE FOR HELO 1
22671         END IF
22674       END IF
22677     NEXT Side
22680!
22683     FOR Side=1 TO 2
22686       IF Cell(Side,2,1)+Cell(Side,2,2)+Cell(Side,2,3)<=0 THEN GOTO Nxt_si
22689       Side_def=(Side MOD 2)+1
22692       IF Side=1 THEN Red_blue$="BLUE"
22695       IF Side=2 THEN Red_blue$="RED"
22698       FOR J=1 TO 20      !WHEN # OF TGTS ARE CLOSE TO 0, SET IT TO 0
22701         IF No_targets(Side_def,J)<.1 THEN No_targets(Side_def,J)=0
22704       NEXT J
22707       Tot_en_helos=No_targets(Side_def,18)+No_targets(Side_def,19)+No_tar
ts(Side_def,20)        !TOTAL NO. OF ENEMY HELOS
22710       FOR J=1 TO 3          !LOOP ON TARGET HELOS
22713         SELECT Atk_prof(Side_def,J)!SET ENEMY MUNI TYPE USED FOR PRIMARY
N
22716         CASE 1,5 !MSL
22719           En_muni(J)=1
22722         CASE 2,6 !MSL/GUN
```

Table 6-14. Ground combat code (continued).

```
22725             IF Tim_step=1 THEN En_muni(J)=1 !SET TO MISSILES FIRST
22728             IF Tim_step=2 THEN En_muni(J)=2 !SET TO GUNS SECOND
22731           CASE 3,7
22734             En_muni(J)=2
22737           CASE 4    !AIR-TO-AIR
22740             En_muni(J)=3
22743           CASE ELSE!NO ATTACK PROFILE
22746             En_muni(J)=0
22749           END SELECT
22752           IF Helo_mis(Side_def,J)=2 AND Atk_prof(Side_def,J)>4 THEN En_muni
)=3  !PRIMARY MSN IS ENEMY HELO KILLS, THEN SET MUNITION USED TO AIR MSLS
22755         NEXT J
22758!
22761         FOR I=1 TO 3
22764           Muni(I)=0
22767           IF Atk_prof(Side,I)>=4 THEN Muni(I)=3!AIR MSLS ON BOARD
22770         NEXT I
22773         IF Muni(1)+Muni(2)+Muni(3)>0 THEN !AT LEAST 1 HELO HAS AIR MSLS
22776           Beg_cat=18              !LOOP THRU AIR TGT PK W/ AIR MUNI
22779           End_cat=20
22782           GOSUB Helo_pk_targ      !CALCULATE PROB. OF KILL OF TGT HELOS
22785         END IF
22788!
22791         FOR I=1 TO 3          !LOOP ON HELO
22794           SELECT Atk_prof(Side,I)!SET ARRAY TO MUNITION TYPE USED THIS TSTE
22797           CASE 1,5 !MSL
22800             Muni(I)=1
22803           CASE 2,6 !MSL/GUN
22806             IF Tim_step=1 THEN Muni(I)=1 ! MISSILE
22809             IF Tim_step=2 THEN Muni(I)=2 !GUN
22812           CASE 3,7 !GUN
22815             Muni(I)=2
22818           CASE 4    !AIR-TO-AIR
22821             Muni(I)=3
22824           CASE ELSE!NO ATTACK PROFILE
22827             Muni(I)=0
22830           END SELECT
22833           Lase=I                  !SET Lase TO WHICH HELO IS LASING-ITSELF OR SCT
22836           IF I=1 THEN Lase=H_lase(Side)
22839           Pop_tstep(I)=0
22842           SELECT Helo_mis(Side,I)   !CALCULATE POPUPS PER TIMESTEP
22845           CASE 1,3   !PRIMARY MSN IS GROUND KILLS, USE MSL OR GUN TE & TM
22848             IF Tm(Side,Lase,Muni(I))+Te(Side,Lase,Muni(I))>0 THEN Pop_tstep
)=Time_step/(Tm(Side,Lase,Muni(I))+Te(Side,Lase,Muni(I)))
22851             Te_firer(I)=Te(Side,Lase,Muni(I)) !STORE TIME EXPOSED OF FIRER
22854           CASE 2     !PRIMARY MSN IS HELO KILLS, USE AIR MSL TE & TM
22857             IF Tm(Side,Lase,3)+Te(Side,Lase,3)>0 THEN Pop_tstep(I)=Time_ste
(Tm(Side,Lase,3)+Te(Side,Lase,3))
22860             Te_firer(I)=Te(Side,Lase,3)
22863           END SELECT
22866         NEXT I
22869         IF (Muni(1)>0 AND Muni(1)<3) OR (Muni(2)>0 AND Muni(2)<3) OR (Muni
```

Table 6-14. Ground combat code (continued).

```
>0 AND Muni(3)<3) THEN !AT LEAST 1 HELO HAS GRND MUNIT
22872        Beg_cat=1             !LOOP THRU GROUND TGT PK W/ GROUND MUNI
22875        End_cat=17
22878        GOSUB Helo_pk_targ   !CALCULATE PROB. OF KILL BY HELOS
22881      END IF
22884      GOSUB Helo_pd_targ  !CALCULATE PROB. OF DETECTION BY HELOS
22887      GOSUB Ad_pd_helo    !CALCULATE PROB. OF DETECTION BY AD WEAPONS
22890      GOSUB Ad_avg_fired  !CALCULATE AVERAGE ROUNDS FIRED BY AD
22893      GOSUB Df_pd_helo    !CALCULATE PROB. OF DETECTION BY DIRECT FIRE
22896      GOSUB Df_avg_fired  !CALCULATE AVERAGE ROUNDS FIRED BY DIRECT FIRE
22899!
22902      FOR I=1 TO 3         !LOOP ON HELO TYPE
22905        IF I=3 AND H_lase(Side)=3 THEN    !SET SCOUT'S POPUPS SAME AS AH1
22908          Pop_tstep(3)=Pop_tstep(1)
22911          GOTO Next_loop
22914        END IF
22917        IF Cell(Side,2,I)<=0 THEN GOTO Next_loop  !NO HELOS OF TYPE I
22920        IF Helo_mis(Side,I)=2 AND Tot_en_helos<=0 THEN
22923          Cell(Side,2,I)=0  !HELO ON AIR MSN BUT NO ENEMY HELOS THIS 30 M
 BTL
22926          PRINT "    ";Red_blue$;" HELICOPTER ";I;" ON AIR-TO-AIR MISSION I
T NO ENEMY HELOS FLYING."
22929          PRINT "    WILL RETURN TO BASE."
22932          GOTO Next_loop
22935        END IF
22938        Lase=I                !SET Lase TO WHICH HELO IS LASING-ITSELF OR SCT
22941        IF I=1 THEN Lase=H_lase(Side)
22944!
22947        ON Atk_prof(Side,I) GOSUB Missile_attrite,Msl_gun,Gun_attrite,Next
air,Missile_attrite,Msl_gun,Gun_attrite
22950        IF Cell(Side,2,I)<=0 THEN GOTO Next_loop  !MEANS NO MUNITIONS LEFT
22953 Next_air:IF Atk_prof(Side,I)>=4 THEN GOSUB Air_attrite    !CALC AIR KILLS
22956 Next_loop:NEXT I
22959!
22962      IF Cell(Side,2,1)+Cell(Side,2,2)+Cell(Side,2,3)>0 THEN
22965        GOSUB Ad_fire_dist !CALCULATE AD FIRE DISTRIBUTION AGAINST HELI'S
22968        GOSUB Ad_act_fired !CALCULATE ACTUAL ROUNDS FIRED BY AD
22971        GOSUB Df_act_fired !CALCULATE ACTUAL ROUNDS FIRED BY DIRECT FIRE
22974        GOSUB Helo_psrv    !CALCULATE HELICOPTER PROB. OF SURVIVAL
22977      END IF
22980!
22983 Nxt_side:NEXT Side
22986!
22989    FOR Side=1 TO 2     !DO ALL THE LOSS CALCULATIONS
22992      FOR I=1 TO 3      !RESET CELLS IF NEEDED
22995        IF Cell(Side,2,I)=0 AND Cell_save(Side,I)>0 THEN
22998          Cell(Side,2,I)=Cell_save(Side,I)
23001          No_targets(Side,I+17)=Cell_save(Side,I)
23004        END IF
23007      NEXT I
23010!
23013      FOR J=1 TO 17     !CALCULATE GROUND TARGET LOSSES BY ATK HELO
```

Table 6-14. Ground combat code (continued).

```
23016       Target_loss(Side,J)=(1-Target_psrv(Side,J))*No_targets(Side,J)
23019     NEXT J
23022     FOR I=1 TO 3     !HELO PROB OF SURV AND LOSS BY AD, DF & ENEMY HELO
23025       Helo_psrv(Side,I)=Target_psrv(Side,I+17)*Helo_psrv(Side,I)
23028       Helo_loss(Side,I)=(1.0-Helo_psrv(Side,I))*Cell(Side,2,I)
23031       Target_loss(Side,I+17)=Helo_loss(Side,I)
23034     NEXT I
23037!
23040     FOR I=1 TO 70    !DECATEGORIZE LOSSES TO GROUND ELEMENTS
23043       IF Side=1 THEN Ctg=B_cat(I)
23046       IF Side=2 THEN Ctg=R_cat(I)
23049       IF Ctg>0 AND Ctg<=17 THEN
23052         IF Target(Side,1,I)>0 AND No_targets(Side,Ctg)>0 THEN
23055           Target(Side,2,I)=Target(Side,2,I)-Target(Side,2,I)*(Target_lo
(Side,Ctg)/No_targets(Side,Ctg))
23058         END IF
23061       END IF
23064     NEXT I
23067!
23070     FOR I=1 TO 20    !CALCULATE NO. OF TARGETS REMAINING
23073       No_targets(Side,I)=No_targets(Side,I)-Target_loss(Side,I)
23076       Target_loss(Side,I)=0
23079     NEXT I
23082!
23085     FOR I=1 TO 3     !SUBTRACT HELO LOSSES
23088       Cell(Side,2,I)=Cell(Side,2,I)-Helo_loss(Side,I)
23091       Helo_loss(Side,I)=0     !ZERO OUT AGAIN
23094     NEXT I
23097   NEXT Side
23100 NEXT Tim_step
23103!
23106 FOR Side=1 TO 2
23109   Ad_ammo(Side)=Ad_wt_avl(Side)/2000    !CHANGE BACK TO TONS
23112   Df_ammo(Side)=Df_wt_avl(Side)/2000
23115 NEXT Side
23118 GOTO Subrout_end
23121!-----------------------------------------------------------------
23124 Missile_attrite: !
23127 IF Helo_rnds_avl(Side,I)<=0 THEN
23130   PRINT "   NO MORE GROUND MISSILES AVAILABLE FOR ";Red_blue$;" HELO ";I
23133   FOR J=1 TO 17
23136     Helo_avg_fired(I,J)=0 !ZERO OUT AVG ROUNDS FIRED PER POPUP
23139   NEXT J
23142   RETURN
23145 END IF
23148 GOSUB Helo_avg_fired      !CALCULATE AVERAGE ROUNDS FIRED BY HELOS
23151 IF Atk_prof(Side,I)<=2 THEN     !NO AIR TO AIR KILLS - DO LOSSES
23154   GOSUB Helo_fire_dist    !CALCULATE FIRE DISTRIBUTION BY HELOS
23157   GOSUB Helo_act_fired    !CALCULATE ACTUAL ROUNDS FIRED BY HELOS
23160   GOSUB Target_psrv       !CALCULATE TGTS' PROB OF SURVIVAL
23163 END IF
23166 RETURN
```

Table 6-14. Ground combat code (continued).

```
23169!-----------------------------------------------------------------------
23172 Gun_attrite:    !
23175  IF Helo_rnds_avl(Side,I)<=0 THEN
23178    PRINT "    NO MORE GUN ROUNDS AVAILABLE FOR ";Red_blue$;" HELO ";I
23181    FOR J=1 TO 17
23184      Helo_avg_fired(I,J)=0 !ZERO OUT AVG. ROUNDS FIRED PER POPUP
23187    NEXT J
23190    RETURN
23193  END IF
23196  GOSUB Helo_avg_fired       !CALCULATE AVERAGE ROUNDS FIRED BY HELOS
23199  IF Atk_prof(Side,I)=2 OR Atk_prof(Side,I)=3 THEN   !NO AIR KILLS-DO LOS
23202    GOSUB Helo_fire_dist     !CALCULATE FIRE DISTRIBUTION BY HELOS
23205    GOSUB Helo_act_fired     !CALCULATE ACTUAL ROUNDS FIRED BY HELOS
23208    GOSUB Target_psrv        !CALCULATE TGTS' PROB OF SURVIVAL
23211  END IF
23214  RETURN
23217!-----------------------------------------------------------------------
23220 Air_attrite:    !
23223  IF Air_rnds_avl(Side,I)<=0 THEN
23226    PRINT "    NO MORE AIR MISSILES AVAILABLE FOR ";Red_blue$;" HELO ";I
23229    FOR J=18 TO 20
23232      Helo_avg_fired(I,J)=0 !ZERO OUT AVG AIR MSLS FIRED PER POPUP
23235    NEXT J
23238  ELSE
23241    Muni(I)=3                !SET POINTER TO AIR MUNITIONS
23244    GOSUB Air_avg_fired      !CALCULATE AVERAGE ROUNDS FIRED BY HELOS
23247  END IF
23250  GOSUB Helo_fire_dist       !CALCULATE FIRE DISTRIBUTION BY HELOS
23253  GOSUB Air_act_fired        !CALCULATE ACTUAL ROUNDS FIRED BY HELOS
23256  GOSUB Target_psrv          !CALCULATE TGTS' PROB OF SURVIVAL
23259  RETURN
23262!-----------------------------------------------------------------------
23265 Msl_gun:   !
23268  IF Tim_step=1 THEN
23271    GOSUB Missile_attrite
23274  ELSE
23277    Helo_rnds_avl(Side,I)=Helo_load(Side,I,2)*.5*Cell(Side,2,I)
                                                    !GUN BASIC LOAD
23280    GOSUB Gun_attrite
23283  END IF
23286  RETURN
23289!-----------------------------------------------------------------------
23292 Line_of_sight:                    ! Prob of Line of Sight bet. helo & grnd
23295  FOR I=1 TO 3
23298    IF Helo_mis(Side,I)>0 THEN
23301      Los_msn=Helo_mis(Side,I)
23304      IF Los_msn=2 THEN Los_msn=1!Use D.S. msn data to calc. plos if helo
s                                      on an air to air msn
23307      Plos(Side,I)=Plos_alpha(Side,Los_msn)*(EXP(-Plos_beta(Side,Los_msn)*
g_avg(Side,I,1)))
23310    END IF
23313  NEXT I
```

Table 6-14. Ground combat code (continued).

```
23316  RETURN
23319 '
23322 !-------------------------------------------------------------------
23325 Helo_pk_targ:                     ' Calculate the Prob. of Killing Target
23328  FOR I=1 TO 3                     ! Looping on Helicopter Type
23331    IF Cell(Side,1,I)<=0 OR Muni(I)<=0 THEN GOTO Next_i_2
23334    FOR J=Beg_cat TO End_cat        ' Looping on Target Category
23337      IF Muni(I)=0 THEN
23340        Helo_pk_targ(I,J)=0
23343        GOTO Next_helo_pk_tg
23346      END IF
23349 ! Is this Target within Range?
23352      IF Rg_avg(Side,I,J)<Pk_rmin(Side,I,Muni(I)) OR Rg_avg(Side,I,J)>Pk_
ax(Side,I,Muni(I)) THEN
23355        Helo_pk_targ(I,J)=0     ' No-then Prob. of Killing is Zero
23358        GOTO Next_helo_pk_tg
23361      END IF
23364 ! Calculate Fully Exposed & Hull Defiladed Prob. of Killing
23367 ! Target using a Specified Munitions at a given Range
23370      Pkhd=(Pk_hd_a(Side,I,Muni(I),J)*(Rg_avg(Side,I,J)^2))+(Pk_hd_b(Side
,Muni(I),J)*Rg_avg(Side,I,J))+Pk_hd_c(Side,I,Muni(I),J)
23373      Pkfe=Pk_fe_a(Side,I,Muni(I),J)*Rg_avg(Side,I,J)^2+Pk_fe_b(Side,I,Mur
(I),J)*Rg_avg(Side,I,J)+Pk_fe_c(Side,I,Muni(I),J)
23376 ! Calculate the Prob. of Killing this Target Category
23379      IF J<=17 THEN Helo_pk_targ(I,J)=Pkhd*Pct_force_hd(Side)+Pkfe*Pct_for
e_fe(Side)
23382      IF J>=18 THEN Helo_pk_targ(I,J)=Pkhd*Pct_mm(Side)+Pkfe*Pct_non_mm(S
e)
23385      IF Helo_pk_targ(I,J)<0 THEN    ' If Prob. of Kill is Negative (bad
23388        Helo_pk_targ(I,J)=0          ' points) then Set it to Zero
23391      END IF
23394 Next_helo_pk_tg:      '
23397    NEXT J
23400 Next_i_2:NEXT I
23403  RETURN
23406!--------------------------------------------------------------------
23409 Helo_pd_targ:                    ! Helicopter Prob. of Detecting Target
23412  FOR I=1 TO 3                     ! Looping on Helicopter Type
23415    IF Cell(Side,1,I)<=0 OR Muni(I)<=0 THEN GOTO Next_i_1
23418    FOR J5=1 TO 5                  ! Looping on Collapsed Target Category
23421 ! Is this Target within Range?
23424      IF Rg_avg_pd(Side,I,J5)<Pd_rmin(Side,I,Atmos) OR Rg_avg_pd(Side,I,J5
>Pd_rmax(Side,I,Atmos) THEN
23427        Helo_pd_tgt(J5)=0   ! No-then Prob. of Detecting is Zero
23430        GOTO Next_coltg_pd
23433      END IF
23436 ! Calculate Fully Exposed & Hull Defiladed Prob. of Detecting; Infinite
23439 ! Time & Average Time to Detect a Target at a Specified Range
23442      Hdinf=Pd_hd_inf_a(Side,I,J5)*Rg_avg_pd(Side,I,J5)^2+Pd_hd_inf_b(Side
I,J5)*Rg_avg_pd(Side,I,J5)+Pd_hd_inf_c(Side,I,J5)
23445      Feinf=Pd_fe_inf_a(Side,I,J5)*Rg_avg_pd(Side,I,J5)^2+Pd_fe_inf_b(Side
I,J5)*Rg_avg_pd(Side,I,J5)+Pd_fe_inf_c(Side,I,J5)
```

Table 6-14. Ground combat code (continued).

```
23448      Hdtbar=Pd_hd_tbar_a(Side,I,J5)*Rg_avg_pd(Side,I,J5)^2+Pd_hd_tbar_b
de,I,J5)*Rg_avg_pd(Side,I,J5)+Pd_hd_tbar_c(Side,I,J5)
23451      Fetbar=Pd_fe_tbar_a(Side,I,J5)*Rg_avg_pd(Side,I,J5)^2+Pd_fe_tbar_b
de,I,J5)*Rg_avg_pd(Side,I,J5)+Pd_fe_tbar_c(Side,I,J5)
23454 ! Calculate Search Time, then Prob. of Detection for
23457 ! Fully Exposed & Hull Defiladed Collapsed Target Category
23460      Ut(I)=Te_firer(I)-(Rg_avg_pd(Side,I,J5)/Fm(Side,I,Muni(I)))
23463      IF Ut(I)<0 THEN Ut(I)=0
23466 ! Calculate Prob. of Helicopter Detecting this Collapsed Target
23469 ! Category
23472      Pdt_fe=Feinf*(1-EXP(-MIN(Ut(I)/Fetbar,708)))
23475      Pdt_hd=Hdinf*(1-EXP(-MIN(Ut(I)/Hdtbar,708)))
23478      Helo_pd_tgt(J5)=Pdt_fe*Pct_force_fe(Side)+Pdt_hd*Pct_force_hd(Side)
23481      IF J5=5 THEN                      ! AIR TARGETS
23484        Pdt_rate(I)=(Pdt_fe*Pct_non_mm(Side))+(Pdt_hd*Pct_mm(Side))
23487      END IF
23490      IF Helo_pd_tgt(J5)<0 THEN   ! If Prob. of Detecting is Negative
23493        Helo_pd_tgt(J5)=0         ! (bad points) then Set it to Zero
23496      END IF
23499 Next_coltg_pd:   !
23502    NEXT J5
23505 Next_helo_pd:   !
23508 ! Expand the 5 Collapsed Target Categories to 20 Target Categories
23511    FOR J=1 TO 20
23514      IF Pd_cat(Side_def,J)>0 AND Pd_cat(Side_def,J)<=5 THEN Helo_pd_targ
,J)=Helo_pd_tgt(Pd_cat(Side_def,J))
23517    NEXT J
23520    IF H_lase(Side_def)=3 THEN Helo_pd_targ(I,18)=0   !NO PROB OF DET OF H
0 1 IF SCTS LASING
23523 Next_i_1:NEXT I
23526  RETURN
23529 !
23532 !-----------------------------------------------------------------
23535 Helo_avg_fired:                ! Calculate Average Rounds Fired at Target
23538  FOR J=1 TO 17                  ! Looping on Target Category
23541    Helo_avg_fired(I,J)=0
23544    IF No_targets(Side_def,J)>0 THEN
23547 ! Calculate Average Rounds Fired by Helicopter using Specified
23550 ! Munition Type
23553      Helo_avg_fired(I,J)=Helo_pd_targ(Lase,J)*Plos(Side,I)*Np(Side,I,Muni
I))
23556    END IF
23559  NEXT J
23562  RETURN
23565 !
23568 !-----------------------------------------------------------------
23571 Helo_fire_dist:                ! Calculate Fire Distribution against Targe
23574  Helo_tot_dist=0
23577  Helo_grd_dist=0
23580  FOR J=1 TO 20                  ! Looping on Target Category
23583 ! Calculate the Total Distribution of Fire across all Targets
23586    Helo_tot_dist=Helo_tot_dist+(Helo_pd_targ(Lase,J)*(No_targets(Side_def
```

Table 6-14. Ground combat code (continued).

```
J)*Helo_pk_targ(I,J))*Tgt_pref(Side,Helo_mis(Side,I),J))
23589  NEXT J
23592 ! Calculate the Distribution of Fire for a Single Target
23595  FOR J=1 TO 20
23598    IF Helo_tot_dist<>0 THEN
23601      Helo_fire_dist(I,J)=(Helo_pd_targ(Lase,J)*(No_targets(Side_def,J)*H
o_pk_targ(I,J))*Tgt_pref(Side,Helo_mis(Side,I),J))/Helo_tot_dist
23604      IF J<=17 THEN Helo_grd_dist=Helo_grd_dist+Helo_fire_dist(I,J)
23607    ELSE
23610      Helo_fire_dist(I,J)=0
23613    END IF
23616  NEXT J
23619  RETURN
23622 !
23625 !-----------------------------------------------------------------------
23628 Helo_act_fired:                 ! Calculate Actual Rounds Fired at Target
23631  Tot_rnds_fired=0
23634  Grd_act_pp(I)=0
23637  FOR J=1 TO 20                  ! Looping on Target Category
23640 ! Calculate Actual Rounds Fired at Target per Timestep
23643    Helo_act_fired(I,J)=Helo_avg_fired(I,J)*Pop_tstep(I)*Helo_fire_dist(I
)*Cell(Side,2,I)
23646    IF Muni(I)=1 THEN Helo_act_fired(I,J)=Helo_act_fired(I,J)*.8*Adust(Ar
(Side))     !MUNITIONS IS MISSILE
23649 ! Calculate Total Rounds Fired per Timestep
23652    Tot_rnds_fired=Tot_rnds_fired+Helo_act_fired(I,J)
23655  NEXT J
23658 ! Are There Enough Rounds Available?
23661  IF Tot_rnds_fired>Helo_rnds_avl(Side,I) THEN
23664    Adj_tot_rnds=0
23667    FOR J=1 TO 20
23670 ! NO-then Adjust Each Category By an Equivalent Percentage of the Rounds
23673 ! Available for Firing
23676      Helo_act_fired(I,J)=(Helo_rnds_avl(Side,I)/Tot_rnds_fired)*Helo_act
ired(I,J)
23679    NEXT J
23682 ! Also Adjust the Number of Popups in a Timestep Downward by an
23685 ! Equivalent Percentage
23688    IF Tot_rnds_fired>0 THEN
23691      Pop_tstep(I)=(Helo_rnds_avl(Side,I)/Tot_rnds_fired)*Pop_tstep(I)
23694    END IF
23697 ! Adjust Total Rounds Fired (you've fired all you have)
23700    Tot_rnds_fired=Helo_rnds_avl(Side,I)
23703  END IF
23706 ! Reset the Rounds Available
23709  Helo_rnds_avl(Side,I)=Helo_rnds_avl(Side,I)-Tot_rnds_fired
23712  RETURN
23715 !
23718 !-----------------------------------------------------------------------
23721 Air_avg_fired: ! CALCULATE AVERAGE ROUNDS FIRED AT OPPOSING HELICOPTERS
23724  FOR J=18 TO 20
23727    IF En_muni(J-17)=0 THEN
```

Table 6-14. Ground combat code (continued).

```
23730         Helo_avg_fired(I,J)=0
23733         GOTO E_air_avg_fired
23736       END IF
23739       Te_targ_recip=1/Te(Side_def,J-17,En_muni(J-17))
23742       Tm_targ_recip=1/Tm(Side_def,J-17,En_muni(J-17))
23745 ! CALCULATE RATE AT WHICH FIRING HELICOPTER "I" DETECTS TARGET
23748 ! HELICOPTER "J" WITH BOTH OF THEM "POPPING" IN & OUT OF THE LINE
23751 ! OF SIGHT
23754       Pdt_rate_los=(Pdt_rate(Lase)*Tm_targ_recip)/(Tm_targ_recip+Te_targ_rec
p)
23757       IF No_targets(Side_def,J)<=0 THEN
23760         Helo_avg_fired(I,J)=0
23763         GOTO E_air_avg_fired
23766       END IF
23769       Time_to_impact=Rg_avg(Side,I,J)/Fm(Side,I,Muni(I))
23772       Te_recip=1/Te_firer(I)
23775       Den_90=LOG(.1)/(-Te_recip)
23778       P_engage=0
23781 ! CALCULATE THE PROB. OF ENGAGEMENT
23784       FOR T=Time_to_impact TO Den_90
23787         Pdt_t1=(Pdt_rate_los/(Te_targ_recip-Pdt_rate_los))*(EXP(-MIN(Pdt_rat
_los*(T-Time_to_impact),708))-EXP(-MIN(Te_targ_recip*(T-Time_to_impact),708)))
23790         Pdt_time=1-(1-Pdt_t1)^No_targets(Side_def,J)
23793         P_engage=P_engage+(Te_recip*EXP(-MIN(Te_recip*T,708))*Pdt_time)
23796       NEXT T
23799       Helo_avg_fired(I,J)=P_engage*Np(Side,I,Muni(I))
23802 E_air_avg_fired:   !
23805   NEXT J
23808   RETURN
23811 !
23814 !-----------------------------------------------------------------------
23817 Air_act_fired:     ! CALCULATE ACTUAL ROUNDS FIRED AT OPPOSING HELICOPTERS
23820                    ! AND GROUND TARGETS
23823   Tot_rnds_fired=0
23826   Grd_pp=0
23829   Air_pp=0
23832   Helo_actual=0
23835   Air_actual=0
23838   Helo_air_dist=1.0-Helo_grd_dist     !TOTAL AIR DISTRIBUTION
23841   FOR J=1 TO 20                     ! Looping on Target Category
23844     Helo_act_fired(I,J)=0
23847     IF J<=17 THEN
23850! CALCULATE ACTUAL ROUNDS FIRED PER POPUP AT ALL GROUND TARGETS
23853       IF Helo_grd_dist>0 THEN
23856         Grd_pp=Grd_pp+Helo_avg_fired(I,J)*Helo_fire_dist(I,J)/Helo_grd_dis
23859! CALCULATE ACTUAL ROUNDS FIRED PER TIMESTEP AT ALL GROUND TARGETS
23862         Helo_act_fired(I,J)=Helo_avg_fired(I,J)*Pop_tstep(I)*Helo_fire_dis
(I,J)*Cell(Side,2,I)
23865         IF Muni(I)=1 THEN Helo_act_fired(I,J)=Helo_act_fired(I,J)*.8*Adust
rty(Side)) !MUNITIONS IS MISSILE
23868         Helo_actual=Helo_actual+Helo_act_fired(I,J)
23871       END IF
```

Table 6-14. Ground combat code (continued).

```
23874    ELSE
23877! CALCULATE ACTUAL ROUNDS FIRED PER POPUP AT ALL OPPOSING HELICOPTERS
23880       IF Helo_air_dist>0 THEN
23883         Air_pp=Air_pp+Helo_avg_fired(I,J)*Helo_fire_dist(I,J)/Helo_air_di
23886! CALCULATE ACTUAL ROUNDS FIRED PER TIMESTEP AT ALL OPPOSING HELICOPTERS
23889         Helo_act_fired(I,J)=Helo_avg_fired(I,J)*Pop_tstep(I)*Helo_fire_di
(I,J)*Cell(Side,2,I)
23892         Air_actual=Air_actual+Helo_act_fired(I,J)
23895       END IF
23898    END IF
23901  NEXT J
23904! ARE THERE ENOUGH AIR & GROUND ROUNDS AVAILABLE TO SUPPORT CALCULATED
23907! FIRING.  IF NOT CONVERT EXCESS AIR ROUNDS TO GROUND FIRING AND VICE VER
23910  Old_air_actual=Air_actual
23913  Old_helo_actual=Helo_actual
23916  IF Air_actual>Air_rnds_avl(Side,I) THEN
23919    Helo_actual=Helo_actual+(Air_actual-Air_rnds_avl(Side,I))*(Grd_pp/Air
p)
23922    Air_actual=Air_rnds_avl(Side,I)
23925    IF Helo_actual>Helo_rnds_avl(Side,I) OR Helo_actual=0 THEN
23928      IF Helo_actual>0 THEN Helo_actual=Helo_rnds_avl(Side,I)
23931     ! Also Adjust the Number of Popups in a Timestep Downward by an
23934     ! Equivalent Percentage
23937      IF Grd_pp>0 THEN A_g_ratio=Air_pp/Grd_pp !AIR/GROUND RATIO
23940      IF Grd_pp<=0 THEN A_g_ratio=0
23943      Pop_tstep(I)=(Helo_rnds_avl(Side,I)*A_g_ratio+Air_rnds_avl(Side,I))
Old_helo_actual*A_g_ratio+Old_air_actual)*Pop_tstep(I)
23946    END IF
23949    GOTO E_air_act_fired
23952  END IF
23955  IF Helo_actual>Helo_rnds_avl(Side,I) THEN
23958    Air_actual=Air_actual+((Helo_actual-Helo_rnds_avl(Side,I))*(Air_pp/Gr
pp))
23961    Helo_actual=Helo_rnds_avl(Side,I)
23964    IF Air_actual>Air_rnds_avl(Side,I) OR Air_actual<=0 THEN
23967      IF Air_actual>0 THEN Air_actual=Air_rnds_avl(Side,I)
23970     ! Also Adjust the Number of Popups in a Timestep Downward by an
23973     ! Equivalent Percentage
23976      IF Air_pp>0 THEN G_a_ratio=Grd_pp/Air_pp !GROUND/AIR TRADEOFF RATE
23979      IF Air_pp<=0 THEN G_a_ratio=0
23982      Pop_tstep(I)=(Helo_rnds_avl(Side,I)+Air_rnds_avl(Side,I)*G_a_ratio)
Old_helo_actual+Old_air_actual*G_a_ratio)*Pop_tstep(I)
23985    END IF
23988  END IF
23991 E_air_act_fired:      !
23994 ! ARE THERE ENOUGH GROUND ROUNDS AVAILABLE?
23997  IF Old_helo_actual<>Helo_actual THEN
24000    FOR J=1 TO 17
24003   ! NO-THEN ADJUST EACH GROUND CATEGORY BY AN EQUIVALENT PERCENTAGE OF T
24006   ! ROUNDS AVAILABLE FOR FIRING
24009      Helo_act_fired(I,J)=(Helo_actual/Old_helo_actual)*Helo_act_fired(I,
24012    NEXT J
```

Table 6-14. Ground combat code (continued).

```
24015  END IF
24018! DECREMENT GROUND ROUNDS AVAILABLE
24021  Helo_rnds_avl(Side,I)=Helo_rnds_avl(Side,I)-Helo_actual
24024 ! ARE THERE ENOUGH AIR ROUNDS AVAILABLE?
24027  IF Old_air_actual<>Air_actual THEN
24030    FOR J=18 TO 20
24033   ! NO-THEN ADJUST EACH AIR CATEGORY BY AN EQUIVALENT PERCENTAGE OF THE
24036   ! ROUNDS AVAILABLE FOR FIRING
24039      Helo_act_fired(I,J)=(Air_actual/Old_air_actual)*Helo_act_fired(I,J)
24042    NEXT J
24045  END IF
24048! DECREMENT AIR ROUNDS AVAILABLE
24051  Air_rnds_avl(Side,I)=Air_rnds_avl(Side,I)-Air_actual
24054  RETURN
24057 !
24060 !-----------------------------------------------------------------------
24063 Target_psrv:                      ! Calculate Prob. of Target Survival
24066  FOR J=1 TO 20                    ! Looping on Target Category
24069    Tgt=MAX(No_targets(Side_def,J),1.0)        ! HAS TO BE 1.0 OR MORE
24072    Target_psrv(Side_def,J)=Target_psrv(Side_def,J)*((1-(Helo_pk_targ(I,J)
Tgt))^Helo_act_fired(I,J))
24075  NEXT J
24078  RETURN
24081 !
24084 !-----------------------------------------------------------------------
24087 Ad_pk_helo:                       ! AD Prob. of Killing Helicopter
24090  FOR I=1 TO 7                     ! Looping on AD Type
24093    FOR J=1 TO 3                   ! Looping on Helicopter Type
24096      IF Rg_avg(Side,J,1)<Pk_ad_rmin(Side_def,I) OR Rg_avg(Side,J,1)>Pk_ac
rmax(Side_def,I) THEN
24099        Ad_pk_helo(Side_def,I,J)=0  ! No-then Prob. of Killing is Zero
24102        GOTO Next_ad_pk_helo
24105      END IF
24108   ! Use Probabilities for the Appropriate Mast Type
24111      IF Mast_mnt(Side,J)=0 THEN
24114        Mt=2     !non mast-mounted
24117      ELSE
24120        Mt=1     !mast-mounted
24123      END IF
24126 ! Calculate the Prob. of Killing at a Specified Range
24129      Ad_pk_helo(Side_def,I,J)=Pk_ad_a(Side_def,I,Mt)*Rg_avg(Side,J,1)^2+F
_ad_b(Side_def,I,Mt)*Rg_avg(Side,J,1)+Pk_ad_c(Side_def,I,Mt)
24132 Next_ad_pk_helo:!
24135    NEXT J
24138  NEXT I
24141  RETURN
24144 !
24147 !-----------------------------------------------------------------------
24150 Ad_pd_helo:                       ! AD Prob. of Detecting Helicopter
24153  FOR I=1 TO 7                     ! Looping on AD Type
24156    IF Fad(Side_def,I)<=0 THEN  !NO AD TYPE I DATA AVAIL
24159      FOR J=1 TO 3
```

Table 6-14. Ground combat code (continued).

```
24162          Ad_pd_helo(I,J)=0
24165        NEXT J
24168        GOTO Next_ad_i
24171      END IF
24174      FOR J=1 TO 3                    ! Looping on Helicopter Type
24177        Ad_pd_helo(I,J)=0
24180        IF J=1 AND H_lase(Side)=3 THEN GOTO Next_ad_pd_helo
24183        IF Cell(Side,1,J)<=0 OR Muni(J)<=0 THEN GOTO Next_ad_pd_helo
24186        IF Rg_avg(Side,J,1)<Pd_ad_rmin(Side_def,I,Atmos) OR Rg_avg(Side,J,1
Pd_ad_rmax(Side_def,I,Atmos) THEN                 !Is it within range?
24189          GOTO Next_ad_pd_helo    !no-Prob of detection is zero
24192        END IF
24195    ! Use Probabilities for the Appropriate Mast Type
24198        IF Mast_mnt(Side,J)=0 THEN
24201          Mt=2        !non mast-mounted
24204        ELSE
24207          Mt=1        !mast-mounted
24210        END IF
24213  ! Calculate the Prob. of Detecting; Infinite Time & Average Time
24216  ! to Detect at a Specified Range
24219        Adinf=Pd_inf_ad_a(Side_def,I,Mt)*Rg_avg(Side,J,1)^2+Pd_inf_ad_b(Sid
def,I,Mt)*Rg_avg(Side,J,1)+Pd_inf_ad_c(Side_def,I,Mt)
24222        Adtbar=Pd_tbar_ad_a(Side_def,I,Mt)*Rg_avg(Side,J,1)^2+Pd_tbar_ad_b(
de_def,I,Mt)*Rg_avg(Side,J,1)+Pd_tbar_ad_c(Side_def,I,Mt)
24225  ! Calculate the Time to Detect this Helicopter Assuming an
24228  ! Engagement will follow
24231        Adteng=Te_firer(J)-(Rg_avg(Side,J,1)/Fad(Side_def,I))
24234        IF Adteng<0 THEN Adteng=0
24237  ! Calculate Prob. of the AD Detecting this Helicopter
24240        IF Adtbar>0 THEN Ad_pd_helo(I,J)=Adinf*(1-EXP(-Adteng/Adtbar))
24243 Next_ad_pd_helo:NEXT J
24246 Next_ad_i:NEXT I
24249  RETURN
24252 !
24255 !-----------------------------------------------------------------
24258 Ad_avg_fired:                      ! AD Average Rounds Fired at Helicopter
24261  FOR I=1 TO 7                      ! Looping on AD Type
24264    FOR J=1 TO 3                    ! Looping on Helicopter Type
24267      IF Cell(Side,1,J)<=0 OR Helo_mis(Side,J)<=0 THEN GOTO Next_j_1
24270 ! Calculate the Average Rounds Fired at the Helicopter
24273      Ad_avg_fired(I,J)=Ad_pd_helo(I,J)*Plos(Side,J)
24276 Next_j_1:NEXT J
24279  NEXT I
24282  RETURN
24285 !
24288 !-----------------------------------------------------------------
24291 Ad_fire_dist:                      ! Calculate AD Distribution against Helo
24294  FOR I=1 TO 7                      ! Looping on AD Type
24297    Ad_tot_dist=0
24300    FOR J=1 TO 3                    ! Looping on Helicopter Type
24303 ! Calculate the Total Distribution of Fire across all Helicopters
24306      Ad_tot_dist=Ad_tot_dist+(Ad_pd_helo(I,J)*(Cell(Side,2,J)*Ad_pk_helo(
```

Table 6-14. Ground combat code (continued).

```
ide_def,I,J))*Ad_pref(Side_def,I,J))
24309     NEXT J
24312 ! Calculate the Distribution of Fire for a Single Target
24315     FOR J=1 TO 3                  ! Looping on Helicopter Type
24318       IF Ad_tot_dist<>0 THEN
24321         Ad_fire_dist(I,J)=(Ad_pd_helo(I,J)*(Cell(Side,2,J)*Ad_pk_helo(Sid
def,I,J))*Ad_pref(Side_def,I,J))/Ad_tot_dist
24324       ELSE
24327         Ad_fire_dist(I,J)=0
24330       END IF
24333     NEXT J
24336   NEXT I
24339   RETURN
24342 !
24345 !-------------------------------------------------------------------------
24348 Ad_act_fired:                     ! Calculate Actual Rounds Fired at Helo
24351   FOR I=1 TO 5          !FIGURE OUT NO. OF AD ELEMENTS ABLE TO FIRE
24354     Ad_ele(I)=Target(Side_def,2,I+47)*(1-Veh_ada(Side_def))
24357   NEXT I
24360   Ad_ele(6)=Target(Side_def,2,53)*(1-Hnd_ada(Side_def))
24363   Ad_ele(7)=Target(Side_def,2,54)*(1-Hnd_ada(Side_def))
24366   Tot_rnds_fired=0
24369   Tot_wt_fired=0
24372   FOR I=1 TO 7                    ! Looping on AD Type
24375     FOR J=1 TO 3                  ! Looping on Helicopter Type
24378 ! If Scouts are Lasing for AH1's then they can't be Detected
24381       IF H_lase(Side)=3 AND J=1 THEN
24384         Ad_act_fired(I,J)=0
24387       ELSE
24390         Ad_act_fired(I,J)=Ad_avg_fired(I,J)*Pop_tstep(J)*Ad_fire_dist(I,J)
Ad_ele(I)
24393       END IF
24396 ! Calculate Total Rounds Fired per Timestep
24399       Tot_rnds_fired=Tot_rnds_fired+Ad_act_fired(I,J)
24402     NEXT J
24405     Tot_wt_fired=Tot_wt_fired+Tot_rnds_fired*Rnd_wt(Side_def,I)    !IN LBS
24408     Tot_rnds_fired=0
24411   NEXT I
24414 ! Are There Enough Rounds Available?
24417   IF Tot_wt_fired>Ad_wt_avl(Side_def) THEN
24420     Adj_tot_rnds=0
24423     FOR I=1 TO 7                  ! Looping on AD Type
24426       FOR J=1 TO 3                ! Looping on Helicopter Type
24429   ! NO-then Adjust Each Category By an Equivalent Percentage of the Round
24432   ! Available for Firing
24435         Ad_act_fired(I,J)=(Ad_wt_avl(Side_def)/Tot_wt_fired)*Ad_act_fired(
,J)
24438   ! Calculate NEW Total Rounds Fired
24441       NEXT J
24444     NEXT I
24447 ! Adjust Total weight Fired (you've fired all you have)
24450     Tot_wt_fired=Ad_wt_avl(Side_def)
```

Table 6-14. Ground combat code (continued).

```
24453  END IF
24456 ! Reset the ammo weight Available
24459  Ad_wt_avl(Side_def)=Ad_wt_avl(Side_def)-Tot_wt_fired
24462  RETURN
24465 !
24468 !----------------------------------------------------------------------
24471 Df_pd_helo:                         ! DF Prob. of Detecting Helicopter
24474 FOR I=1 TO 2                        ! Looping on Sensor Type
24477   FOR J=1 TO 3                      ! Looping on Helicopter Type
24480     Dfinf(I,J)=0
24483     Dftbar(I,J)=0
24486     IF J=1 AND H_lase(Side)=3 THEN GOTO Next_df_pd_helo
24489     IF Cell(Side,1,J)<=0 OR Muni(J)<=0 THEN GOTO Next_df_pd_helo
24492     IF Rg_avg(Side,J,1)<Pd_df_rmin(Side_def,I,Atmos) OR Rg_avg(Side.J.1)
d_df_rmax(Side_def,I,Atmos) THEN                   !Is it within range?
24495       GOTO Next_df_pd_helo      ! No-Prob of detection is zero
24498     END IF
24501 ! Use Probabilities for the Appropriate Mast Type
24504     IF Mast_mnt(Side,J)=0 THEN
24507       Mt=2                          ! Non mast-mounted
24510     ELSE
24513       Mt=1                          ! Mast-mounted
24516     END IF
24519 ! Calculate the Prob. of Detecting; Infinite Time & Average Time
24522 ! to Detect at a Specified Range
24525     Dfinf(I,J)=Pd_inf_df_a(Side_def,I,Mt)*Rg_avg(Side,J,1)^2+Pd_inf_df_b
ide_def,I,Mt)*Rg_avg(Side,J,1)+Pd_inf_df_c(Side_def,I,Mt)
24528     Dftbar(I,J)=Pd_tbar_df_a(Side_def,I,Mt)*Rg_avg(Side,J,1)^2+Pd_tbar_d
b(Side_def,I,Mt)*Rg_avg(Side,J,1)+Pd_tbar_df_c(Side_def,I,Mt)
24531 Next_df_pd_helo:    !
24534   NEXT J
24537 NEXT I
24540 FOR I=1 TO 20                             ! LOOPING ON DF TYPE
24543   IF Df_muni_ptr(Side_def,I)<=0 OR Df_sen_ptr(Side_def,I)<=0 THEN
                                                       ! NO DF TYPE I DATA AVAIL
24546     FOR J=1 TO 3
24549       Df_pd_helo(I,J)=0
24552     NEXT J
24555     GOTO Next_df_i
24558   END IF
24561   FOR J=1 TO 3                            ! LOOPING ON HELICOPTER TYPE
24564 ! Calculate the Time to Detect this Helicopter Assuming an
24567 ! Engagement will follow
24570     Dfteng=Te_firer(J)-(Rg_avg(Side,J,1)/F_df(Side_def,Df_muni_ptr(Side_c
f,I)))
24573     IF Dfteng<0 THEN Dfteng=0
24576 ! Calculate Prob. of the DF Detecting this Helicopter
24579     IF Dftbar(Df_sen_ptr(Side_def,I),J)>0 THEN
24582       Df_pd_helo(I,J)=Dfinf(Df_sen_ptr(Side_def,I),J)*(1-EXP(-Dfteng/Dftt
r(Df_sen_ptr(Side_def,I),J)))
24585     END IF
24588   NEXT J
```

Table 6-14. Ground combat code (continued).

```
24591 Next_df_i:
24594 NEXT I
24597 RETURN
24600 !
24603 !------------------------------------------------------------------------
24606 Df_avg_fired:                    ! DF Average Rounds Fired at Helicopter
24609 FOR I=1 TO 20                    ! Looping on DF Type
24612   IF Df_muni_ptr(Side_def,I)<=0 THEN
24615      FOR J=1 TO 3     !ZERO OUT ROUNDS FIRED BY THIS DIRECT FIRER
24618         Df_avg_fired(I,J)=0
24621      NEXT J
24624      GOTO Nxt_i_1
24627   END IF
24630   FOR J=1 TO 3                     ! Looping on Helicopter Type
24633     Df_avg_fired(I,J)=0
24636     IF Cell(Side,1,J)<=0 OR Helo_mis(Side,J)<=0 THEN GOTO Nxt_j_1
24639 ! Calculate the Average Rounds Fired at the Helicopter
24642     Df_avg_fired(I,J)=Df_pd_helo(I,J)*Plos(Side,J)*Df_rnds_eng(Side_def,I
_muni_ptr(Side_def,I))
24645 Nxt_j_1:NEXT J
24648 Nxt_i_1:NEXT I
24651 RETURN
24654 !
24657 !------------------------------------------------------------------------
24660 Df_act_fired:                    ! Calculate Actual Rounds Fired at Helo
24663 Tot_rnds_fired=0
24666 Tot_wt_fired=0
24669 FOR I=1 TO 20                    ! Looping on DF Type
24672   FOR J=1 TO 3                   ! Looping on Helicopter Type
24675 ! If Scouts are Lasing for AH1's then they can't be Detected
24678     IF H_lase(Side)=3 AND J=1 THEN
24681       Df_act_fired(I,J)=0
24684     ELSE
24687       Df_act_fired(I,J)=Df_avg_fired(I,J)*Pop_tstep(J)*Df_fire_dist(Side_
ef,I,J)*Target(Side_def,2,I)
24690     END IF
24693 ! Calculate Total Rounds Fired per Popup
24696     Tot_rnds_fired=Tot_rnds_fired+Df_act_fired(I,J)
24699   NEXT J
24702   IF Side_def=1 THEN Df_rnd_wt=B_ammo_wt(I)
24705   IF Side_def=2 THEN Df_rnd_wt=R_ammo_wt(I)
24708   Tot_wt_fired=Tot_wt_fired+Tot_rnds_fired*Df_rnd_wt        !IN LBS
24711   Tot_rnds_fired=0
24714 NEXT I
24717 ! Are There Enough Rounds Available?
24720 IF Tot_wt_fired>Df_wt_avl(Side_def) THEN
24723   Dfj_tot_rnds=0
24726   FOR I=1 TO 20                  ! Looping on DF Type
24729     FOR J=1 TO 3                 ! Looping on Helicopter Type
24732 ! NO-then adjust Each Category By an Equivalent Percentage of the Rounds
24735 ' Available for Firing
24738       Df_act_fired(I,J)=(Df_wt_avl(Side_def)/Tot_wt_fired)*Df_act_fired(I
```

Table 6-14. Ground combat code (concluded).

```
J)
24741      NEXT J
24744    NEXT I
24747 ! Adjust Total weight Fired (you've fired all you have)
24750    Tot_wt_fired=Df_wt_avl(Side_def)
24753 END IF
24756 ! Reset the ammo weight Available
24759 Df_wt_avl(Side_def)=Df_wt_avl(Side_def)-Tot_wt_fired
24762 RETURN
24765 !
24768 !---------------------------------------------------------------------
24771 Helo_psrv:                        ! Calculate Prob. of Helicopter Survival
24774 FOR J=1 TO 3                      ! Looping on Helicopter Type
24777!   Helo_psrv(Side,J)=1            ! Set initial Prob. of Survival to 1.0
24780    IF Cell(Side,2,J)<=0 THEN GOTO No_psrv
24783    H_alv=MAX(Cell(Side,2,J),1.0)           ! HAS TO BE >=1.0
24786    FOR I=1 TO 7                   ! Looping on AD Type
24789      Helo_psrv(Side,J)=Helo_psrv(Side,J)*((1-(Ad_pk_helo(Side_def,I,J)/H_
v))^Ad_act_fired(I,J))
24792    NEXT I
24795    IF Mast_mnt(Side,J)=0 THEN Mt=1   !NON MAST MT (FE)
24798    IF Mast_mnt(Side,J)=1 THEN Mt=2   !MAST MNT   (HD)
24801    FOR I=1 TO 20                  ! Looping on DF Type
24804      Helo_psrv(Side,J)=Helo_psrv(Side,J)*((1-(Df_pk_helo(Side_def,I,J,Mt)
_alv))^Df_act_fired(I,J))
24807    NEXT I
24810 No_psrv:NEXT J
24813 RETURN
24816 !
24819 Subrout_end: !
24822 SUBEND
24825 !
24828 !**********************************************************************
       *********************************************************************
```

CHAPTER 7

CHEMICAL ATTRITION

1. PURPOSE.

The purpose of the chemical attrition program is to calculate attrition from chemical munitions based on agent type, firing unit, unit type, composition, and mission-oriented protective posture (MOPP) status.

2. GENERAL.

A. The chemical attrition program (P5) is a slightly modified version of the DAME chemical module discussed in CAORA/TR-5/83, Deep Attack Map Exercise (DAME) Game Rules and Operation Procedures. The attrition process remains the same.

B. Attrition of Red and Blue forces due to chemical warfare is played in DIME using the chemical program. However, it is important to realize that chemical attrition is not a separate process but must be integrated into the overall loss assessment process used in the game. The chemical munitions used in DIME are artillery or rocket delivered and are measured in battery (48-round) or battalion (144-round) missions. The program allows Red and Blue forces to deliver either persistent or non-persistent agents against enemy forces. The use of chemical munitions must be integrated into the artillery fire planning process, and use of chemical munitions must not exceed prescribed firing rates, range requirements, or basic loads. Chemical missions replace conventional missions in the overall firing profile of an artillery unit. An artillery unit may not exceed its conventional rate of fire or basic load.

3. DATA FLOW.

A. Input data. As with other programs, an input sheet has been developed to simplify the development of chemical attrition assessments. Figure 7-1 shows an example of the form. The following paragraphs describe the required inputs.

(1) Type of mission. Circle the type of mission desired. A separate input sheet is required for each individual mission type. For example, if the gamer desires to shoot Red battalion missions of persistent agent and Blue battery missions of non-persistent agent, two separate input sheets are required. Note that Red battalion mission denotes the Red attacker.

GAMER INPUT SHEET

A. Circle one of the following:

(1) Red Battery of Persistent

(2) Red Battery of Non-persistent

(3) Red Battalion of Persistent

(4) Red Battalion of Non-persistent

(5) Blue Battery of Persistent

(6) Blue Battery of Non-persistent

(7) Blue Battalion of Persistent

(8) Blue Battalion of Non-persistent

B. Enter number of units to assess as targets for the above mission (max 10). ________

C. Fill in the table with the following information for each target selected:

UNIT - A legitimate unit number from the unit file.

FRACTION - Fraction of unit affected by mission.

MISSIONS - Number of mission assessments against unit.

MOPP - MOPP status (1 = Not in MOPP, 2 = In MOPP)

TARGET	UNIT	FRACTION	MISSIONS	MOPP
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Figure 7-1. Chemical attrition input sheet

(2) Number of targets. Enter the number of units which will be targeted by a particular type of mission. A target may only have one unit in it. A maximum of 10 targets may be designated for a mission type. If the gamer desires more than 10 targets, an additional input sheet must be used.

(3) Target data matrix. Information on each target is specified by entering:

(a) The unit number of the target (1-400).

(b) The fraction of the unit which is affected by the mission. For example, if only half a unit is in the target area, then .50 is entered.

(c) The number of missions which will be fired against the target.

(d) The MOPP status of the target unit (1 = not in MOPP, 2 = in MOPP). Units which are in MOPP do not sustain casualties from chemical attack.

B. Output data. Output consists of the total chemical kills for each side per mission and total chemical kills for each side per critical incident (CI).

C. Data flows are depicted in Figure 7-2.

4. FILE STRUCTURE.

Data files supporting the chemical program are held external to the program. These files consist of target radius files and target profile files.

A. A target radius file exists for both defenders, Blue and Red (BLTEMP and RDTEMP). This data is read into the Trgt_rds(I,J) array, where:

I = unit type (1-10)
J = major mission
1 = attack
2 = defend
3 = reserve
4 = movement.

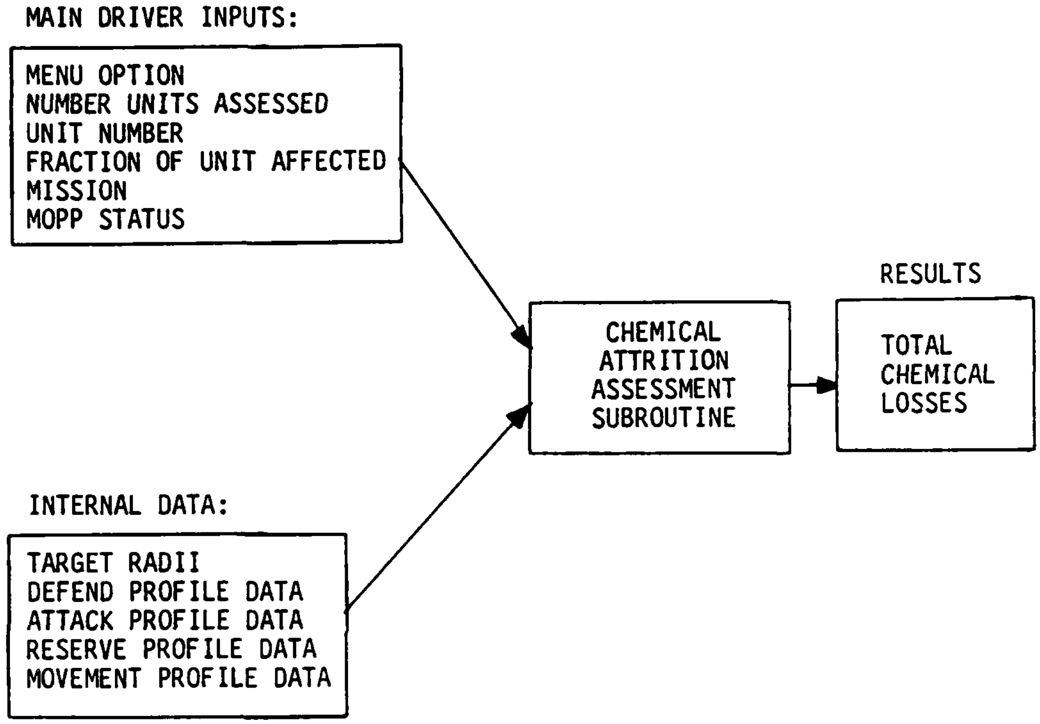


Figure 7-2. Chemical data flow.

The appropriate value from this array, for the unit being attacked, is used to access the correct target profile array.

B. A target profile file containing casualty fractions exists for the combinations of the following:

(1) Blue or Red defender.

(2) Battery or Battalion.

(3) Persistent or non-persistent chemical.

(4) Major mission (defend, attack, reserve, move).

The appropriate one-record file is read into the Defend_file(*), Attack_file(*), Reserve_file(*), and Move_file(*) arrays. The casualty fraction for each element type within a unit is accessed by choosing the appropriate major mission file for the defending unit (Defend_file (I,J), Attack_file (I,J), Reserve_file (I,J), and Move_file (I,J)), where:

I = 1-70 element types.
J = 1-5, target radius value divided by 100.

C. It should be noted that no data currently exists for the chemical program. Therefore, volume III of this report does not contain information and data concerning the chemical program.

5. ALGORITHMS.

The primary algorithm/equation used to assess chemical attrition is:

Kills = Cslty_frct * N_elements (Eq. 7-1)

This calculation is repeated for the number of missions using the new number of elements (N_elements = N_elements - Kills) for each repetition where:

Kills = number of victims.
Cslty_frct = casualty fraction to be assessed, depends on target radius and profile.
N_elements = number of elements to assess.

The logic flow of the chemical program is depicted in Figure 7-3.

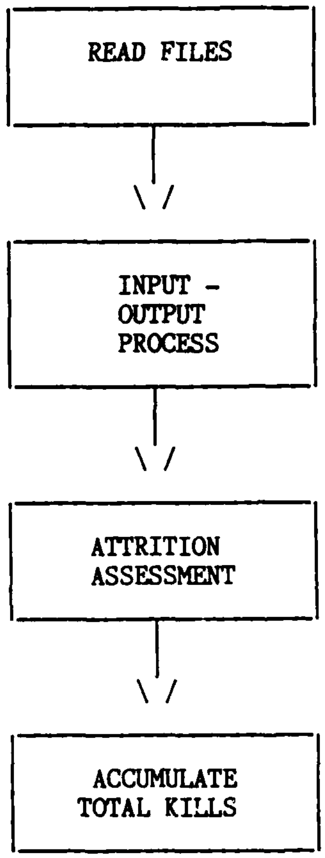


Figure 7-3. Chemical Logic Flow

6. "UNITFILE" IMPACT.

This program changes several elements in the unit status file ("UNITFILE"). Elements 1-70 are suffer attrition if chemical losses are subtracted. Element 77 (MOPP status) is changed to a 2 which indicates that the unit is in MOPP. There is no interaction with any other programs. Control is returned to the DIME driver program.

7. CODE.

A. The chemical program code is discussed in the following paragraphs. The functional areas discussed are represented in Figure 7-4.

(1) Initialization of variables and selection from the menu shown in Figure 7-1 (gamer input sheet), part A, are the first occurrences within the program. Appropriate data files are then read.

(2) Following other inputs (see Figure 7-1), the MOPP value entered is assessed. If the MOPP status is 2, the unit is in MOPP and cannot be assessed. If the value 1 is entered for a unit, the assessment is continued. It should be noted that the MOPP value input for each unit from the menu should correspond accordingly with the unit's MOPP value held within the "UNITFILE".

(3) If assessment is to continue, another check must be made. An attacker may not inflict attrition on a friendly unit (on the same side). If the unit suffering attrition is unfriendly, the appropriate casualty fraction is multiplied by the number of elements to get the number of elements killed.

(4) MOPP status is changed to "in MOPP" for those units assessed. A summary of the inputs and total chemical kills for the current critical incident (CI) is printed if the choice was to update the kills to the units.

B. A subroutine variable listing appears in Table 7-1. Table 7-2 contains a listing of the chemical program code.

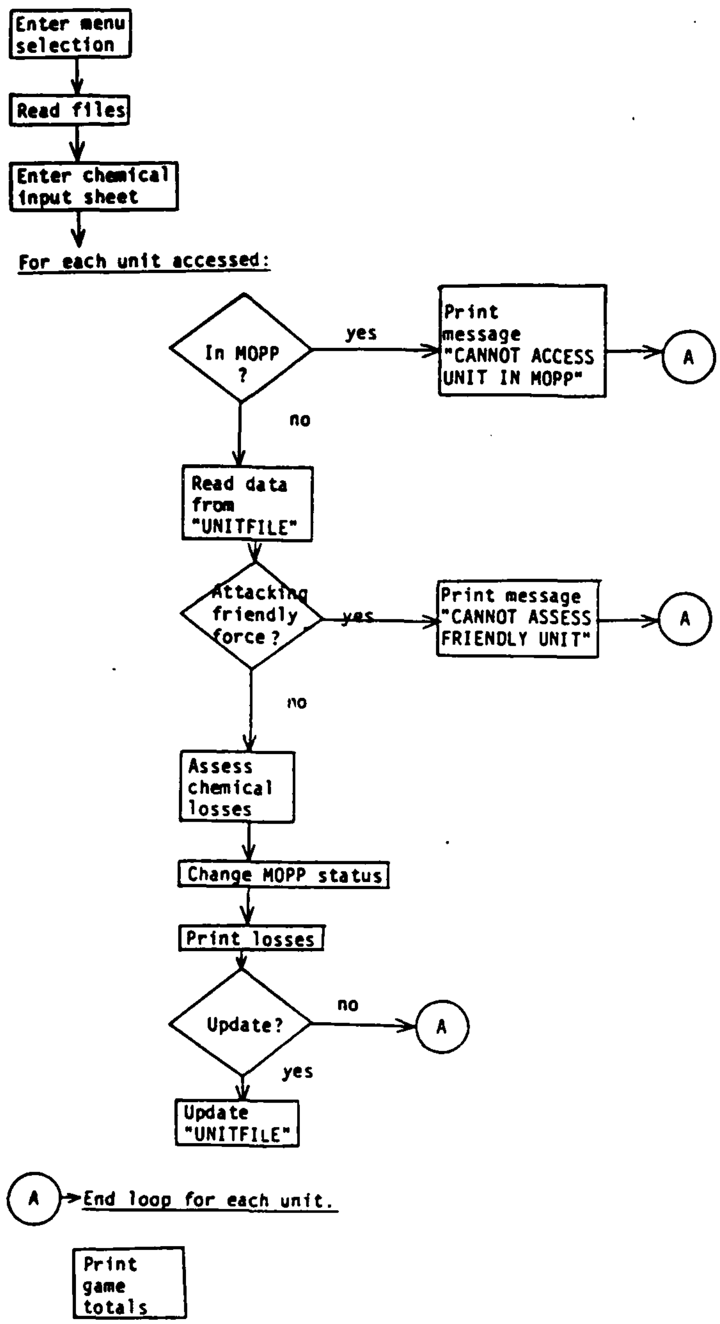


Figure 7-4. Chemical functional flow.

Table 7-1. Chemical subroutine table.

Functional area(s): Chemical program

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Menu_selection	Select appropriate menu options and set variables.	A. Menu_optn	Option 1-9 on menu 1 = Red battery of persistent 2 = Red battery of non-persistent 3 = Red battal'n of persistent 4 = Red battal'n of non-persistent 5 = Blue battery of persistent 6 = Blue battery of non-persistent 7 = Blue battal'n of persistent 8 = Blue battal'n of non-persistent 9 = Exit chemical module
		B. Name_p1$	Mission for firer abbreviation
		C. Np1$	Mission firer
		D. Name_p2$	Chemical agent fired abbreviation
		E. Np2$	Chemical agent fired
		F. Name_p3$	Victim force abbreviation
		G. Attacker$	Attacking force
		H. Victim$	Victim force

Table 7-1. Chemical subroutine table.

Functional area(s): Chemical program

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Read_files	Opens needed files	A. Trgt_rds (*)	Target radii for Red and Blue forces.
		B. Defend_file(*)	Defend target profile data
		C. Attack_file(*)	Attack target profile data
		D. Reserve_file(*)	Reserve target profile data
		E. Move_file(*)	Movement target profile data
Input_output	Processes input and output data from keyboard.	A. Num_units	Number of units to assess.
		B. Unit_num(*)	Unit being assessed.
		C. Trgt_frct(*)	Fraction of area targeted
		D. Nm_missions(*)	Number of missions required/ unit.
		E. Mopp_status(*)	1 = not in MOPP 2 = in MOPP.
		F. Targ_id	Target number.
Attrition	Assesses attrition losses.	A. Chem_losses(*)	Assessed chemical attrition losses.
Assess	Called by Attrition	A. Force_type	1 = Blue; 2 = Red
		B. Force$	Force type from N(26) of unit file.

Table 7-1. Chemical subroutine table.

Functional area(s): Chemical program

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Chem_kill	Called by Assess; assesses chemical losses.	A. Unit_type	Unit member to be chemically assessed.
		B. Trgt_radius	Target radius based on unit type and mission profile.
		C. Calty_frct	Casualty fraction from a selected table.
		D. Sum_kill	Element victim tabulator.
		E. N_elements	Number of elements times fraction targeted.
Print_losses	Called by Assess; prints loss assessment.	A. Mission_tot(I)	Total chemical kills for an entire mission. (I = 1 - 70)
Accumulate	Totals kills	A. Bl_game_tot(I)	Total Blue chemical kills for an entire CI.
		B. Rd_game_tot(I)	Total Red chemical kills for an entire CI.
Sub_end	Returns to DIME menu.		

Table 7-1. Chemical subroutine table.

Functional area(s): Check variables

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Ck_var	Called by all routines with inputs. Checks inputs for correct values.	A. Min_value	Minimum value of variable.
		B. Max_value	Maximum value of variable.
		C. Variable	Variable to be checked.
Print_error	Prints error messages.		

Table 7-2. Chemical code.

```
10    !!! P5 - CHEMICAL ATTRITION
20    !         REWRITTEN FROM DAME CHEMICAL MODULE INTO HP BASIC
30    ! EXPANDED VERSION -- JUNE 9, 1986 -- BY OAO CORP.
40    OPTION BASE 1
50    DIM N(150),Trgt_rds(10,4)
60    DIM Chem_losses(70),Bl_game_tot(70),Rd_game_tot(70),Mission_tot(70)
70    DIM Defend_file(70,5),Attack_file(70,5),Reserve_file(70,5),Move_file(70,5
80    DIM Unit_num(10),Trgt_frct(10),Nm_missions(10),Mopp_status(10)
90   !
100  !
110 Main_program:  !
120   PRINT USING "@,#"
130   Disk$=":HP9134,701"
140   Menu$="REPEAT"
150   WHILE Menu$="REPEAT"
160     GOSUB Menu_selection
170     SELECT Option$
180     CASE "RUN"
190       GOSUB Read_files
200       GOSUB Input_output
210       GOSUB Attrition
220       GOSUB Accumulate
230     CASE "EXIT"
240       PRINT
250       PRINT
260       PRINT
270       PRINT "EXIT CHEMICAL MODULE"
280       Menu$="STOP"
290     END SELECT
300   END WHILE
310  !
320   GOTO Sub_end
330  !
340 Menu_selection:  !
350   REPEAT
360     PRINT
370     PRINT "CHEMICAL MODULE MENU--INPUT FOLLOWING OPTION:"
380     PRINT "     (1)RED  BATTERY   OF PERSISTENT"
390     PRINT "     (2)RED  BATTERY   OF NON-PERSISTENT"
400     PRINT "     (3)RED  BATTALION OF PERSISTENT"
410     PRINT "     (4)RED  BATTALION OF NON-PERSISTENT"
420     PRINT "     (5)BLUE BATTERY   OF PERSISTENT"
430     PRINT "     (6)BLUE BATTERY   OF NON-PERSISTENT"
440     PRINT "     (7)BLUE BATTALION OF PERSISTENT"
450     PRINT "     (8)BLUE BATTALION OF NON-PERSISTENT"
460     PRINT "     (9)EXIT CHEMICAL MODULE"
470     INPUT Menu_optn
480     Option$="RUN"
490     SELECT Menu_optn
500     CASE 1
510       Name_p1$="BTY"
520       Np1$="BATTERY"
```

Table 7-2. Chemical code (continued).

```
530        Name_p2$="PS"
540        Np2$="PERSISTENT"
550        Name_p3$="BL"
560        Attacker$="RED"
570        Victim$="BLUE"
580      CASE 2
590        Name_p1$="BTY"
600        Np1$="BATTERY"
610        Name_p2$="NP"
620        Np2$="NON-PERSISTENT"
630        Name_p3$="BL"
640        Attacker$="RED"
650        Victim$="BLUE"
660      CASE 3
670        Name_p1$="BN"
680        Np1$="BATTALION"
690        Name_p2$="PS"
700        Np2$="PERSISTENT"
710        Name_p3$="BL"
720        Attacker$="RED"
730        Victim$="BLUE"
740      CASE 4
750        Name_p1$="BN"
760        Np1$="BATTALION"
770        Name_p2$="NP"
780        Np2$="NON-PERSISTENT"
790        Name_p3$="BL"
800        Attacker$="RED"
810        Victim$="BLUE"
820      CASE 5
830        Name_p1$="BTY"
840        Np1$="BATTERY"
850        Name_p2$="PS"
860        Np2$="PERSISTENT"
870        Name_p3$="RD"
880        Attacker$="BLUE"
890        Victim$="RED"
900      CASE 6
910        Name_p1$="BTY"
920        Np1$="BATTERY"
930        Name_p2$="NP"
940        Np2$="NON-PERSISTENT"
950        Name_p3$="RD"
960        Attacker$="BLUE"
970        Victim$="RED"
980      CASE 7
990        Name_p1$="BN"
1000       Np1$="BATTALION"
1010       Name_p2$="PS"
1020       Np2$="PERSISTENT"
1030       Name_p3$="RD"
1040       Attacker$="BLUE"
```

Table 7-2. Chemical code (continued).

```
1050        Victim$="RED"
1060      CASE 8
1070        Name_p1$="BN"
1080        Np1$="BATTALION"
1090        Name_p2$="NP"
1100        Np2$="NON-PERSISTENT"
1110        Name_p3$="RD"
1120        Attacker$="BLUE"
1130        Victim$="RED"
1140      CASE 9
1150        Option$="EXIT"
1160      CASE ELSE
1170        PRINT
1180        PRINT "** ERROR:  INVALID MENU SELECTION"
1190        WAIT 1
1200      END SELECT
1210    UNTIL Menu_optn>=1 AND Menu_optn<=9
1220    RETURN
1230!
1240 Read_files:!
1250    File$=Name_p3$&"TEMP"
1260    ASSIGN @Path TO File$&Disk$
1270    ENTER @Path,1;Trgt_rds(*)
1280    Name$=Name_p1$&Name_p2$&Name_p3$
1290    File$=Name$&"DFN"
1300    ASSIGN @Path TO File$&Disk$
1310    ENTER @Path,1;Defend_file(*)
1320    File$=Name$&"ATK"
1330    ASSIGN @Path TO File$&Disk$
1340    ENTER @Path,1;Attack_file(*)
1350    File$=Name$&"RS"
1360    ASSIGN @Path TO File$&Disk$
1370    ENTER @Path,1;Reserve_file(*)
1380    File$=Name$&"MV"
1390    ASSIGN @Path TO File$&Disk$
1400    ENTER @Path,1;Move_file(*)
1410    ASSIGN @Path TO *
1420    RETURN
1430!
1440 Input_output:!
1450      !
1460    INPUT "ENTER: # OF UNITS TO ASSESS (MAX 10)",Num_units
1470    CALL Ck_var("# OF UNITS","TO",Num_units,1,10)
1480    PRINT
1490    PRINT "FOR EACH UNIT, ENTER:  UNIT #, TGT FRACTION, # MISSIONS, MOPP #"
1500    FOR I=1 TO Num_units
1510      PRINT USING Fiol;" TGT # ",I
1520 Fiol:IMAGE 5A,2D,2X
1530      INPUT Unit_num(I),Trgt_frct(I),Nm_missions(I),Mopp_status(I)
1540      CALL Ck_var("UNIT #","TO",Unit_num(I),1,400)
1550      CALL Ck_var("TGT FRACTION","THROUGH",Trgt_frct(I),0,1)
1560      CALL Ck_var("# MISSIONS","TO",Nm_missions(I),0,10)
```

Table 7-2. Chemical code (continued).

```
1570     CALL Ck_var("MOPP #","OR",Mopp_status(I),1,2)
1580   NEXT I
1590     '
1600 Pt:Re_enter$="CONT"
1610   PRINT
1620   PRINT "THE FOLLOWING WERE CHOSEN: "
1630   GOSUB Prnt
1640     !
1650   REPEAT
1660     INPUT "DO YOU WISH TO CHANGE INPUTS?  (Y/N)",Answ$
1670   UNTIL Answ$="Y" OR Answ$="N"
1680     !
1690   WHILE Re_enter$="CONT"
1700     SELECT Answ$
1710     CASE "Y"
1720       INPUT "ENTER TARGET #",Targ_id
1730       CALL Ck_var("TARGET #","TO",Targ_id,1,Num_units)
1740       INPUT "ENTER: UNIT #,  TGT FRACTION, # MISSIONS, MOPP #",Unit_num(Targ
_id),Trgt_frct(Targ_id),Nm_missions(Targ_id),Mopp_status(Targ_id)
1750       CALL Ck_var("UNIT #","TO",Unit_num(Targ_id),1,400)
1760       CALL Ck_var("TGT FRACTION","THROUGH",Trgt_frct(Targ_id),0,1)
1770       CALL Ck_var("# MISSIONS","TO",Nm_missions(Targ_id),0,10)
1780       CALL Ck_var("MOPP #","OR",Mopp_status(Targ_id),1,2)
1790           !
1800       REPEAT
1810         INPUT "ARE CHANGES COMPLETE? (Y/N)",An$
1820       UNTIL An$="Y" OR An$="N"
1830           !
1840       SELECT An$
1850       CASE "Y"
1860         Re_enter$="STOP"
1870       CASE "N"
1880         Re_enter$="CONT"
1890       END SELECT
1900           !
1910     CASE "N"
1920       GOTO Ret
1930     END SELECT
1940   END WHILE
1950   GOTO Pt
1960 Ret: !
1970   RETURN
1980  !
1990 Attrition: !
2000   FOR I=1 TO Num_units
2010         !INITIALIZE LOSSES
2020     FOR L=1 TO 70
2030       Chem_losses(L)=0
2040     NEXT L
2050         !CHECK MOPP STATUS
2060     IF Mopp_status(I)=2 THEN
2070       PRINT
```

Table 7-2. Chemical code (continued).

```
2080      PRINT "UNIT ";Unit_num(I);" IN MOPP, CANNOT BE ASSESSED"
2090    ELSE
2100      GOSUB Assess
2110    END IF
2120  NEXT I
2130  RETURN
2140 !
2150 Assess: !
2160  ASSIGN @Path TO "UNITFILE:HP9134,701"
2170  ENTER @Path,Unit_num(I);N(*)
2180   !
2190   !CHECK COLOR OF UNIT
2200  Force_type=INT(N(78))
2210  SELECT Force_type
2220  CASE 1
2230    Force$="BLUE"
2240  CASE 2
2250    Force$="RED"
2260  END SELECT
2270  IF Force$=Attacker$ THEN
2280    PRINT
2290    PRINT "CANNOT ASSESS FRIENDLY UNIT ";Unit_num(I)
2300  ELSE
2310    GOSUB Chem_kill
2320        ! CHANGE MOPP STATUS IN UNITFILE
2330    N(77)=2
2340    GOSUB Prnt_losses
2350  END IF
2360    !
2370    !
2380  OUTPUT @Path,Unit_num(I);N(*)
2390  ASSIGN @Path TO *
2400    !
2410    !
2420  RETURN
2430 !
2440 Chem_kill: !
2450   !UNPACK TO GET UNIT TYPE
2460  Unit_type=(N(78)-Force_type)*10+1
2470  Trgt_radius=Trgt_rds(Unit_type,N(75))/100
2480    !
2490  FOR J=1 TO 70
2500    SELECT N(75)
2510    CASE 1
2520      Cslty_frct=Attack_file(J,Trgt_radius)
2530    CASE 2
2540      Cslty_frct=Defend_file(J,Trgt_radius)
2550    CASE 3
2560      Cslty_frct=Reserve_file(J,Trgt_radius)
2570    CASE 4
2580      Cslty_frct=Move_file(J,Trgt_radius)
2590    END SELECT
```

Table 7-2. Chemical code (continued).

```
2600         !
2610     Sum_kill=0
2620     N_elements=N(J)*Trgt_frct(I)
2630     FOR K=1 TO Nm_missions(I)
2640       Sum_kill=Sum_kill+Cslty_frct*N_elements
2650       N_elements=N_elements-Sum_kill
2660       IF N_elements<=0 THEN
2670         N_elements=0
2680       END IF
2690     NEXT K
2700         !
2710     Chem_losses(J)=Sum_kill
2720   NEXT J
2730   RETURN
2740 !
2750 Prnt: !
2760   PRINT
2770   PRINT "TGT #          UNIT #          TGT FRACTION          # MISSIONS          MOP
P #"
2780   FOR I=1 TO Num_units
2790     PRINT USING Fio2;I,Unit_num(I),Trgt_frct(I),Nm_missions(I),Mopp_status(I
)
2800 Fio2:IMAGE 3X,2D,12X,3D,11X,D.2D,16X,2D,16X,D
2810   NEXT I
2820   RETURN
2830 !
2840 Prnt_losses: !
2850   PRINT
2860   PRINT "CHEMICAL LOSSES TO UNIT ";Unit_num(I)
2870   PRINT
2880   PRINT "          ELEMENT #          LOSSES"
2890   FOR M=1 TO 70
2900     PRINT USING Fpl1;M,Chem_losses(M)
2910   NEXT M
2920 Fpl1:IMAGE 15X,3D, 7X,4D.2D
2930   REPEAT
2940     INPUT "DO YOU WISH TO SUBTRACT LOSSES?  (Y/N)",An$
2950   UNTIL An$="Y" OR An$="N"
2960   IF An$="Y" THEN
2970     FOR M=1 TO 70
2980       Mission_tot(M)=Mission_tot(M)+Chem_losses(M)
2990       N(M)=N(M)-Chem_losses(M)
3000     NEXT M
3010   END IF
3020   RETURN
3030 !
3040 Accumulate: !
3050   PRINTER IS 702
3060   PRINT
3070   PRINT
3080   PRINT "THE FOLLOWING PARAMETERS WERE CHOSEN FOR MISSION ":Attacker$:" ":Np
1$:" OF ":Np2$
```

Table 7-2. Chemical code (continued).

```
3090  GOSUB Prnt
3100  PRINT
3110  PRINT
3120  PRINT "              ";Victim$;" CHEMICAL VICTIMS FOR THIS MISSION"
3130  PRINT "                   ELEMENT #           VICTIMS"
3140  FOR I=1 TO 70
3150    PRINT USING Fa1;I,Mission_tot(I)
3160  NEXT I
3170 Fa1:IMAGE 25X,3D,11X,4D.2D
3180   !
3190  ASSIGN @Blvctm TO "BLCHMVCTM"
3200  ASSIGN @Rdvctm TO "RDCHMVCTM"
3210  ENTER @Blvctm,1;Bl_game_tot(*)
3220  ENTER @Rdvctm,1;Rd_game_tot(*)
3230   !
3240  IF Name_p3$="BL" THEN
3250    FOR I=1 TO 70
3260      Bl_game_tot(I)=Bl_game_tot(I)+Mission_tot(I)
3270    NEXT I
3280  END IF
3290  IF Name_p3$="RD" THEN
3300    FOR I=1 TO 70
3310      Rd_game_tot(I)=Rd_game_tot(I)+Mission_tot(I)
3320    NEXT I
3330  END IF
3340  OUTPUT @Blvctm,1;Bl_game_tot(*)
3350  OUTPUT @Rdvctm,1;Rd_game_tot(*)
3360   !
3370  PRINT
3380  PRINT
3390  PRINT "TOTAL BLUE CHEMICAL VICTIMS                  TOTAL RED CHEMICAL VI
TIMS"
3400  PRINT "  ELEMENT #      VICTIMS                      ELEMENT #     VICTI
S"
3410  FOR I=1 TO 70
3420    PRINT USING Fa2;I,Bl_game_tot(I),I,Rd_game_tot(I)
3430  NEXT I
3440 Fa2:IMAGE   9X,2D, 5X,4D.2D,31X,2D, 4X,4D.2D
3450  PRINTER IS 1
3460  RETURN
3470 !
3480 Sub_end: !
3490  LOAD "DIME:HP9134,701"
3500  END
3510   !
3520   !*********************************************************************
3530   !
3540  SUB Ck_var(Var_name$,T$,Variable,Min_value,Max_value)
3550    SELECT T$
3560    CASE "THROUGH"
3570      WHILE Variable<Min_value OR Variable>Max_value
3580        GOSUB Print_error
```

Table 7-2. Chemical code (concluded).

```
3590       END WHILE
3600     CASE "OR"
3610       GOTO Case_to
3620     CASE "TO"
3630 Case_to:FOR M=Min_value TO Max_value
3640           IF Variable=M THEN GOTO End_select
3650       NEXT M
3660       GOSUB Print_error
3670       GOTO Case_to
3680 End_select:!
3690     END SELECT
3700     GOTO Rtrn
3710 Print_error:     !
3720     PRINT
3730     PRINT "** ERROR: ";Variable;" IS INVALID FOR ";Var_name$
3740     PRINT "INPUT: ";Min_value;" ";T$;" ";Max_value;" ONLY"
3750     INPUT Variable
3760     RETURN
3770 Rtrn:!
3780   SUBEND
```

CHAPTER 8

COMMAND AND CONTROL

1. PURPOSE.

The purpose of the DIME command and control program (P10) is to calculate the reaction time for both the commander and staff to begin a new mission.

2. GENERAL.

The command and control program calculates the total reaction time needed to react to a change in mission.

A. The command and control program uses an interactive menu/response format to access the appropriate delay times stored in auxiliary data files.

B. Combining the responses from the gamer with the appropriate delay times, the program calculates the total reaction time for the change in mission.

C. This program develops the command and control table look-up procedure from the Deep Attack Map Exercise (DAME) model into a computerized process.

3. DATA FLOW.

A. The data flow consists of a menu/response format in which the user answers questions concerning:

(1) Side: 1 = Blue; 2 = Red.

(2) Mission: 1 = Defend
2 = Move
3 = Reserve
4 = Attack.

(3) Transmission of commands:

(a) Issuing echelon:
0 = Battalion/regiment
1 = Brigade/division
2 = Division (Blue only)
3 = Corps/army
4 = Army/front.

(b) Receiving echelon:
1 = Brigade/division
2 = Division (Blue only)
3 = Corps/army.

(4) Weather conditions: 1 = Moderate
2 = Severe
3 = Good.

(5) Combat condition: 1 = Conventional
2 = Integrated (chemical/nuclear)
3 = Conventional and integrated.

(6) Day/night condition: 1 = Night (1800 to 0600 hours)
2 = Day.

B. Using the responses input by the user, the program accesses the appropriate delay time array.

C. Figure 8-1 indicates the data flow with the appropriate inputs and outputs.

4. FILE STRUCTURE.

The command and control program data consists solely of eight auxiliary data files which contain the delay time in minutes under various combat and environmental conditions.

A. Combat related delay and effect arrays include: Conv_delay(*), Integ_delay(*), Attrite_eff(*), Inter_eff(*), and Deep_atk_eff(*).

(1) Conv_delay(M). A 40-dimensioned array containing the conventional delay time in minutes. This delay time is dependent upon the side, mission and issuing echelon. For means of simplicity, the array Conv_delay(M) is equivalent in structure to C_d (I,J,K) where:

I = 1 to 2 sides
J = 1 to 4 missions
K = 1 to 5 issuing echelon.
Note: K = 3 is vacant for Red.

(2) Integ_delay(M). A 40-dimensioned array containing the integrated delay time in minutes. Indexes are same as above.

(3) Attrite_eff(N). An eight-dimensioned array containing the attrition delay time in minutes. This delay time is dependent upon the side and the mission for a Blue battalion or a Red regiment. For means of simplicity, the array Attrite_eff(N) is equivalent in structure to A_e (I,J) where:

I = 1 to 2 sides
J = 1 to 4 missions for a Blue battalion or a regiment or a Red regiment.

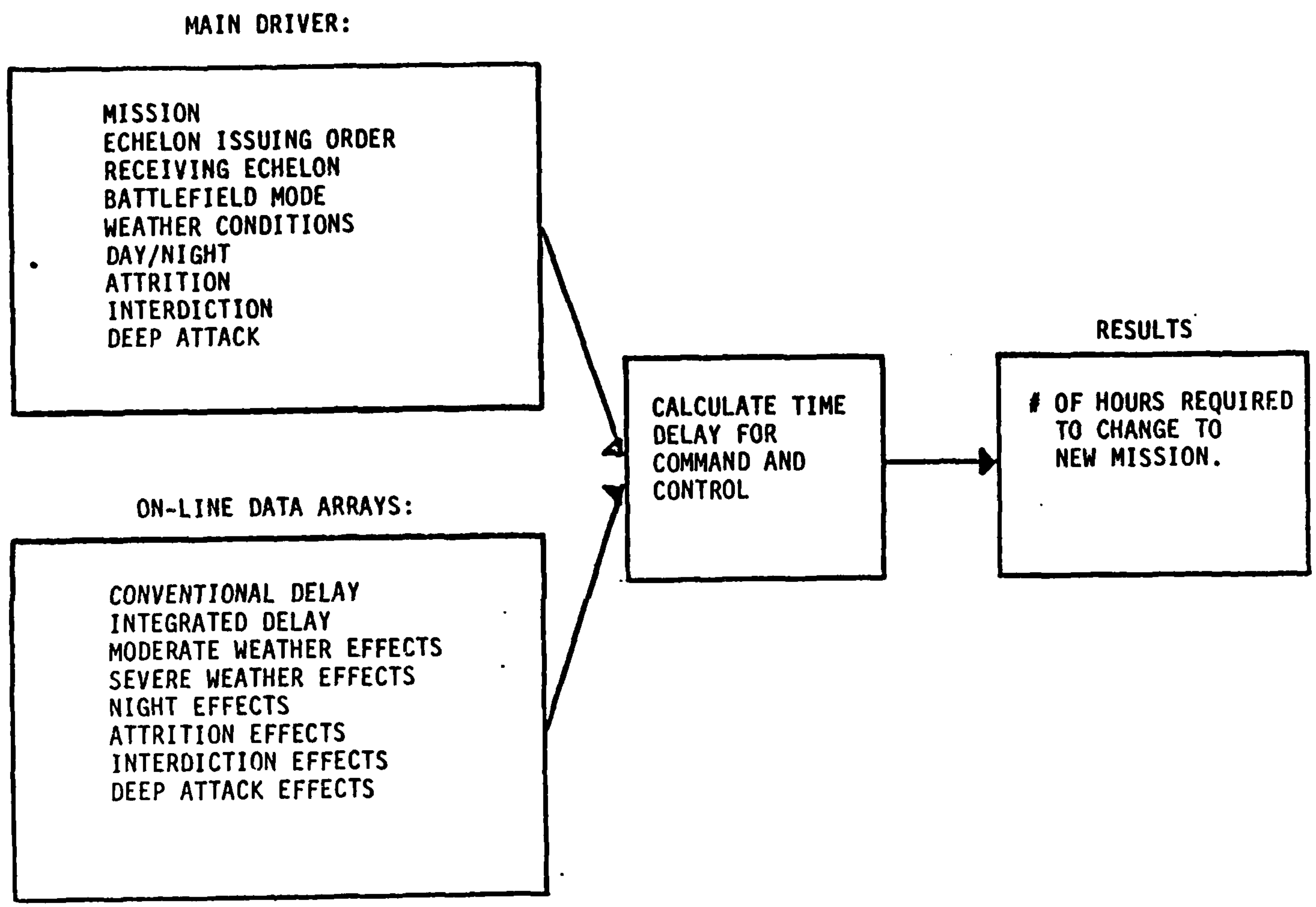


Figure 8-1. Command and control data flow.

(4) Inter_eff(M). A 40-dimensional array containing the air interdiction delay effects in minutes. This delay is dependent upon the side, mission, and issuing echelon. Refer to (1) above for indices.

(5) Deep_atk_eff(P). A 10-dimensioned array containing the deep attack effects (in minutes) for a unit with an attacking mission. For means of simplicity, the array Deep_atk_eff(P) is equivalent in structure to D_e(I,K) where:

I = 1 to 2 sides
K = 1 to 5 issuing echelon.
Note: K = 3 is vacant for Red.

B. Environmental effect arrays include: Mod_weather_eff(*), Sev_weather_eff(*) and Night_eff(*).

(1) Mod_weather_eff(M). A 40-dimensioned array containing the delay time, in minutes, due to moderate weather conditions. Refer to A(1) for index descriptions.

(2) Sev_weather_eff(M). A 40-dimensioned array containing the delay time, in minutes, due to severe weather conditions. Refer to A(1) for index descriptions.

(3) Night_eff(M). A 40-dimensioned array containing the delay time, in minutes, due to nighttime decreases in visibility. Refer to A(1) for index descriptions.

5. **ALGORITHMS.**

A. The command and control program uses an interactive menu/response format to access the delay times associated with battle mode, weather effects, visibility, attrition, interdiction by air, and deep attack.

B. Once the appropriate delay times have been read into the data arrays, the program uses a simple formula to calculate the total reaction time needed to change from an existing mission to a new mission. Equation 8-1 shows this formula.

$$T = \left[\sum_{i=le}^{he} (B_i + W_i + V_i + D_i) + \sum_{j=1}^{ie} (I_j + A) \right] / 60 \qquad \text{(Eq. 8-1)}$$

where:

T = total time to react to mission change (hours).
he = number corresponding to the highest echelon involved.

le = number corresponding to the lowest echelon involved.
B_i = delay (minutes) due to battle mode for each echelon involved (i).
W_i = delay (minutes) due to weather conditions for each echelon involved (i).
V_i = delay (minutes) due to day/night for each echelon involved (i).
D_i = delay (minutes) if unit mission is deep attack for each echelon involved (i).
ie = number of interdicted echelons.
I_j = delay (minutes) due to air interdiction for the j^{th} echelon.
A = delay (minutes) due to attrition suffered by the lowest echelon (Blue battalion/Red regiment).

C. Figure 8-2 contains a generalized flow of the command and control program.

6. "UNITFILE" IMPACT.

The command and control program is a stand-alone program, which does not impact directly with the DIME unit status file ("UNITFILE").

7. CODE.

A. The command and control program consists of 15 subroutines performing three major functions: game initialization, data entry, and calculation of total time required to react to a mission change.

(1) Game initialization. The game initialization routines (Start, Defend, Relocate, Reserve, and Attack) establish the initial pointers and data parameters needed to access the appropriate delay time arrays.

(2) Data entry. The data entry routines (Conven, Integ, Both, Moderate, Severe, Night, Attrite, Interdict, Deep_atk) use the pointers and parameters established by the game initialization routes to access the appropriate delay time arrays.

(3) Total time calculation. The total time is a cumulative value resulting from totaling all the delay times read into the program by the data entry routines. The calculation does not occur in any specific routine but is calculated throughout all the data entry routines.

B. Table 8-1 contains a description of the primary variables associated with the subroutines in the command and control program. Table 8-2 contains a listing of the command and control code.

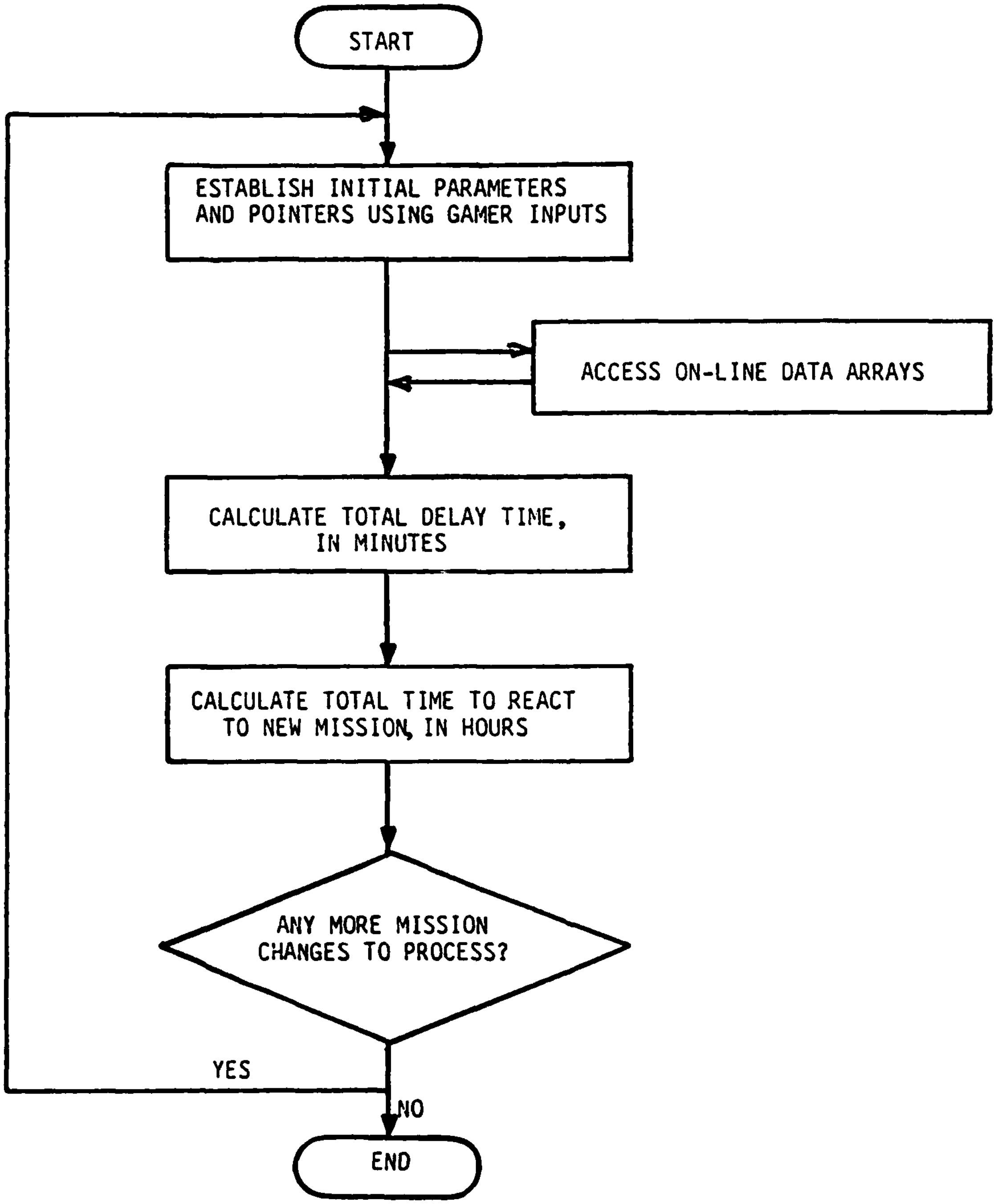


Figure 8-2. Command and control logic flow

Table B-1. Command and control subroutine table.

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
A. Game initialization.	Start	Establishes initial pointer and data parameters by using gamer inputs. Calls the appropriate data routines. Outputs the total time required to change mission.	a. Side	An integer value of 1 or 2, which identifies the force: 1 = Blue 2 = Red
			b. Mission	An integer value of 1 to 4 which identifies the mission: 1 = Defend 2 = Move 3 = Reserve 4 = Attack
			c. Hi_echelon	An integer value of 0 to 4, which represents the echelon issuing the order: 0 = Battalion/Regiment 1 = Brigade/Division 2 = Division (Blue only) 3 = Corps/Army 4 = Army/Front
			d. Lo_echelon	An integer value of 1 to 3 which represents the echelon receiving the order: 1 = Brigade/Division 2 = Division (Blue only) 3 = Corps/Army
			e. Mode	An integer value of 1 to 3 which represents the battlefield mode: 1 = Conventional 2 = Integrated 3 = Both

Table B-1. Command and control subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
A. Game initialization (concluded)	Start (concluded)		f. Weather	An integer value of 1 to 3 which represents the weather condition: 1 = Moderate 2 = Severe 3 = Good
			g. Phase	An integer value of 1 or 2 which represents the visibility due to time of day: 1 = Night (1800 - 0600 hrs) 2 = Day
			h. Response$	A character value of "Y" or "N" which indicates if the unit has suffered attrition
			i. Answer$	A character value of "Y" or "N" which indicates if the unit has suffered air interdiction
			j. Anser$	A character value of "Y" or "N" which indicates if the unit's mission is a deep attack
			k. Hours	A real value which contains the total reaction time in hours
			l. Respond$	An integer value of "Y" or "N" which indicates if any other mission change times need to be calculated "Y" = Repeat program "N" = Return to DIME Menu

Table 8-1. Command and control subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
B. Access data files; calculate total time required to change mission.	Conven: Integ; Both; Moderate; Severe; Night; Attrite; Interdict; Deep_atk	Reads in appropriate delay times from data arrays. Calculates delay times based on data and gamer inputs.	a. Conv_delay(*)	A 1 x 40 array containing the delay time, in minutes, due to a conventional battle
			b. Integ_delay(*)	A 1 x 40 array containing the delay time, in minutes, due to an integrated battle
			c. Mod_weather_eff	A 1 x 40 array containing the delay time, in minutes, due to moderate weather effects
			d. Sev_weather_eff	A 1 x 40 array containing the delay time, in minutes, due to severe weather effects
			e. Night_eff(*)	A 1 x 40 array containing the delay time, in minutes, due to night conditions
			f. Attrite_eff(*)	A 1 x 8 array containing the delay time, in minutes, due to attrition suffered by a Battalion or Regiment
			g. Percent	A 2-digit, integer number which represents the percent of the unit suffering attrition
			h. Amount	A real value indicating the percent of a unit suffering attrition
			i. Inter_eff(*)	A 1 x 40 array containing the delay time, in minutes, due to air interdition

Table 8-1. Command and control subroutine table (concluded).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
B. Access data files; calculate total time required to change mission (concluded).	Conven; Integ; Both; Moderate; Severe; Night; Attrite; Interdict; Deep_atk (concluded)		j. Inter_ech	An integer value of 0 to 4 which indicates the echelon suffering air interdiction
			k. Deep_atk_eff(*)	A 1 x 10 array containing the delay time, in minutes, due to a deep attack

Table 8-2. Command and control code.

```
110    REM-"P10" IS THE COMMAND & CONTROL DELAY TIME CALCULATOR PROGRAM FOR DIM
;
20     !CODED BY MAJ A. RESNICK, FDAD, SAD, CAORA, AVN 552-5481/3595.
30     !THE PROGRAM WAS LAST CHANGED ON 23 MAY 1983.
40     !
50     !
60     !
70     OPTION BASE 1
80     DIM Conv_delay(40),Integ_delay(40),Mod_weather_eff(40),Sev_weather_eff(40
,Night_eff(40),Inter_eff(40),Deep_atk_eff(10),Dataline(18),Attrite_eff(8)
90     !
100    Datdrive$=":HP9121,700,0"
110 Start:PRINT USING "@,#"
120    PRINT "THIS PROGRAM CALCULATES THE STAFF & COMMANDER REACTION TIME"
130    INPUT "WHICH SIDE? (1=BLUE   2=RED)",Side
140    IF Side<>1 AND Side<>2 THEN Start
150    IF Side=1 THEN Start_point=1
160    IF Side=2 THEN Start_point=21
170    Delay=0
180    PRINT USING "@"
190    PRINT "WHAT IS THE ASSIGNED MISSION?"
200    PRINT " 1 = DEFEND"
210    PRINT " 2 = MOVE"
220    PRINT " 3 = RESERVE"
230    PRINT " 4 = ATTACK"
240    INPUT "ENTER ASSIGNED MISSION:",Mission
250    IF Mission<>1 AND Mission<>2 AND Mission<>3 AND Mission<>4 THEN 190
260    ON Mission GOTO Defend,Relocate,Reserve,Attack
270 Defend:Pointer=Start_point
280    GOTO 340
290 Relocate:Pointer=Start_point+5
300    GOTO 340
310 Reserve:Pointer=Start_point+10
320    GOTO 340
330 Attack:Pointer=Start_point+15
340    PRINT USING "@"
350    PRINT "WHAT IS HIGHEST ECHELON ISSUING ORDER?"
360    PRINT "0 = BATTALION/REGIMENT (BLUE/RED)"
370    PRINT "1 = BRIGADE/DIVISION"
380    PRINT "2 = DIVISION (BLUE ONLY)"
390    PRINT "3 = CORPS/ARMY"
400    PRINT "4 = ARMY/FRONT"
410    INPUT "ENTER HIGHEST ECHELON:",Hi_echelon
420    IF Hi_echelon<>0 AND Hi_echelon<>1 AND Hi_echelon<>2 AND Hi_echelon<>3 AN
 Hi_echelon<>4 THEN 350
430    PRINT USING "@"
440    PRINT "IS THE RECEIVING ECHELON A BATTALION/REGIMENT?"
450    PRINT "Y = YES,       N = NO"
460    INPUT "ENTER RESPONSE:",Anwer$
470    IF Anwer$<>"Y" AND Anwer$<>"N" THEN 440
480    IF Anwer$="N" THEN
490      PRINT USING "@"
```

Table 8-2. Command and control code (continued).

```
500     PRINT "WHAT LEVEL IS RECEIVING THE ORDER?"
510     PRINT "1 = BRIGADE/DIVISION"
520     PRINT "2 = DIVISION (BLUE ONLY)"
530     PRINT "3 = CORPS/ARMY"
540     INPUT "ENTER RECEIVING ECHELON:",Lo_echelon
550     IF Lo_echelon<>1 AND Lo_echelon<>2 AND Lo_echelon<>3 THEN 500
560   ELSE
570     Lo_echelon=0
580   END IF
590 !
600   ASSIGN @Cc_file TO "CMD_CNTRL"&Datdrive$
610   ASSIGN @Cc_file2 TO "CC_EFF"&Datdrive$
620   ENTER @Cc_file2,1;Dataline(*)
630   ASSIGN @Cc_file2 TO *
640   FOR I=1 TO 8
650      Attrite_eff(I)=Dataline(I)
660   NEXT I
670   FOR I=1 TO 10
680      Deep_atk_eff(I)=Dataline(I+8)
690   NEXT I
700 !
710   PRINT USING "@"
720   PRINT "WHAT IS BATTLEFIELD MODE?"
730   PRINT "1=CONVENTIONAL   2=INTEGRATED   3=BOTH"
740   INPUT "ENTER MODE:",Mode
750   IF Mode<>1 AND Mode<>2 AND Mode<>3 THEN 720
760   ON Mode GOSUB Conven,Integ,Both
770   PRINT USING "@"
780   PRINT "WHAT ARE WEATHER CONDITIONS?"
790   PRINT "1=MODERATE   2=SEVERE   3=GOOD"
800   INPUT "ENTER WEATHER CONDITIONS:",Weather
810   IF Weather<>1 AND Weather<>2 AND Weather<>3 THEN 780
820   ON Weather GOSUB Moderate,Severe,Good
830   PRINT USING "@"
840   PRINT "IS IT DAY OR NIGHT?"
850   PRINT "1=NIGHT (2100 TO 0600 HRS)     2=DAY"
860   INPUT "PHASE OF DAY:",Phase
870   IF Phase<>1 AND Phase<>2 THEN 840
880   IF Phase=1 THEN GOSUB Night
890   PRINT USING "@"
900   PRINT "HAS THIS BATTALION/REGIMENT BEEN ATTRITED?"
910   PRINT "Y = YES        N = NO"
920   INPUT "ENTER RESPONSE:",Response$
930   IF Response$<>"Y" AND Response$<>"N" THEN 900
940   IF Response$="Y" THEN GOSUB Attrite
950   PRINT USING "@"
960   PRINT "HAS ANY ECHELON BEEN INTERDICTED BY AIR?"
970   PRINT "Y = YES        N = NO"
980   INPUT "ENTER ANSWER:",Answer$
990   IF Answer$<>"Y" AND Answer$<>"N" THEN 960
1000  IF Answer$="Y" THEN GOSUB Interdict
1010  PRINT USING "@"
```

Table 8-2. Command and control code (continued).

```
1020  PRINT "IS THIS A DEEP ATTACK?"
1030  PRINT "Y = YES        N = NO"
1040  INPUT "ENTER RESPONSE:",Anser$
1050  IF Anser$<>"Y" AND Anser$<>"N" THEN 1020
1060  IF Anser$="Y" THEN GOSUB Deep_atk
1070  Hours=Delay/60.
1080  PRINT USING "@"
1090  PRINT "TIME DELAY IMPOSED FOR COMMAND AND"
1100  PRINT "CONTROL PROCESSING IS ",Hours,"HOURS"
1110  PRINT USING "/////"
1120  PRINT "PRESS CONT TO PROCEED"
1130  PAUSE
1140  PRINT USING "@"
1150  PRINT "IS THERE ANY FURTHER PROCESSING REQUIRED?"
1160  PRINT "Y = YES           N = NO"
1170  INPUT "ENTER RESPONSE:",Respond$
1180  IF Respond$<>"Y" AND Respond$<>"N" THEN 1150
1190  IF Respond$="Y" THEN Start
1200          ! LOAD "DIME"
1210  GOTO Halt
1220          !
1230          !
1240          !
1250 Conven: !CONVENTIONAL MODE
1260  ENTER @Cc_file,1;Conv_delay(*)
1270  FOR I=Pointer+Lo_echelon TO Pointer+Hi_echelon
1280    Delay=Delay+Conv_delay(I)
1290  NEXT I
1300  RETURN
1310          !
1320 Integ:   !INTEGRATED MODE
1330  ENTER @Cc_file,2;Integ_delay(*)
1340  FOR I=Pointer+Lo_echelon TO Pointer+Hi_echelon
1350    Delay=Delay+Integ_delay(I)
1360  NEXT I
1370  RETURN
1380          !
1390 Both:    !COMBINED CONVENTIONAL & INTEGRATED
1400  PRINT USING "@"
1410  PRINT "FOR COMBINED CONVENTIONAL & INTEGRATED MODE:"
1420  PRINT "WHAT IS LOWEST INTEGRATED ECHELON?"
1430  PRINT "0 = BATTALION/REGIMENT"
1440  PRINT "1 = BRIGADE/DIVISION"
1450  PRINT "2 = DIVISION (BLUE ONLY)"
1460  PRINT "3 = CORPS/ARMY"
1470  PRINT "4 = ARMY/FRONT"
1480  INPUT "ENTER LOWEST INTEGRATED ECHELON:",Lo_integ_ech
1490  IF Lo_integ_ech<>0 AND Lo_integ_ech<>1 AND Lo_integ_ech<>2 AND Lo_integ_e
h<>3 AND Lo_integ_ech<>4 THEN 1420
1500          !COMPUTES DELAY TIME FOR CONVEN & INTEG SEPARETELY
1510  Hi_echelon_tem=Hi_echelon
1520  Hi_echelon=Lo_integ_ech-1
```

Table 8-2. Command and control code (continued).

```
1530  GOSUB Conven
1540  Hi_echelon=Hi_echelon_tem
1550  Lo_echelon_tem=Lo_echelon
1560  Lo_echelon=Lo_integ_ech
1570  GOSUB Integ
1580  Lo_echelon=Lo_echelon_tem
1590  RETURN
1600          !
1610 Moderate:!MODERATE WEATHER EFFECTS
1620  ENTER @Cc_file,3;Mod_weather_eff(*)
1630  FOR J=Pointer+Lo_echelon TO Pointer+Hi_echelon
1640    Delay=Delay+Mod_weather_eff(J)
1650  NEXT J
1660  RETURN
1670          !
1680 Severe: !SEVERE WEATHER EFFECTS
1690  ENTER @Cc_file,4;Sev_weather_eff(*)
1700  FOR J=Pointer+Lo_echelon TO Pointer+Hi_echelon
1710    Delay=Delay+Sev_weather_eff(J)
1720  NEXT J
1730  RETURN
1740          !
1750 Good:    !THERE ARE NO EFFECTS FOR GOOD WEATHER
1760  RETURN
1770          !
1780 Night:   !
1790  ENTER @Cc_file,5;Night_eff(*)
1800  FOR K=Pointer+Lo_echelon TO Pointer+Hi_echelon
1810    Delay=Delay+Night_eff(K)
1820  NEXT K
1830  RETURN
1840          !
1850 Attrite:!
1860  PRINT USING "@"
1870  INPUT "ENTER PERCENT OF UNIT ATTRITED (AS 2-DIGIT NO.):",Percent
1880  Amount=Percent/10
1890  IF Side=2 THEN
1900    Factor=Mission+4
1910  ELSE
1920    Factor=Mission
1930  END IF
1940  IF Lo_echelon=0 THEN
1950    Attrition=Attrite_eff(Factor)*Amount
1960  ELSE
1970    Attrition=0
1980  END IF
1990  Delay=Delay+Attrition
2000  RETURN
2010          !
2020 Interdict:!ADDS TIME FOR INTERDICTED MODES
2030  ENTER @Cc_file,6;Inter_eff(*)
2040  PRINT USING "@"
```

Table 8-2. Command and control code (concluded).

```
2050 Again:PRINT "WHICH ECHELON IS INTERDICTED?"
2060  PRINT "0 = BATTALION/REGIMENT"
2070  PRINT "1 = BRIGADE/DIVISION"
2080  PRINT "2 = DIVISION (BLUE ONLY)"
2090  PRINT "3 = CORPS/ARMY"
2100  PRINT "4 = ARMY/FRONT"
2110  INPUT "ENTER INTERDICTED ECHELON:",Inter_ech
2120  IF Inter_ech<>0 AND Inter_ech<>1 AND Inter_ech<>2 AND Inter_ech<>3 AND Int
er_ech<>4 THEN 2050
2130  Delay=Delay+Inter_eff(Pointer+Inter_ech)
2140  PRINT USING "@"
2150  INPUT "ARE THERE ANY OTHER ECHELONS INTERDICTED? (Y OR N)",Q$
2160  IF Q$<>"Y" AND Q$<>"N" THEN 2140
2170  IF Q$="Y" THEN GOTO Again
2180  RETURN
2190         !
2200 Deep_atk:!
2210  FOR K=Lo_echelon+1 TO Hi_echelon+1
2220    Delay=Delay+Deep_atk_eff(K)
2230  NEXT K
2240  RETURN
2250         !
2260 Halt:ASSIGN @Cc_file TO *
2270  LOAD "DIME:HP9134,701"
2280  END
2290               !FINIS CORONAT OPUM
```

CHAPTER 9

MOVEMENT GENERATOR[1]

1. PURPOSE.

The purpose of the DIME movement generator (P9) is to provide a deterministic method for calculating troop and cargo movement capabilities and timelines.

2. GENERAL.

The DIME movement program portrays two movement phases: troop movement and cargo transport.

A. Troops move as either dismounted or mounted during the ground movement phase.

(1) The user decides whether troop movement is dismounted or mounted and the number of resting periods the unit shall receive.

(2) A movement rate value is used by the troop movement phase to determine the total time needed to complete a march along a predetermined route of specified length.

B. Cargo transport is modeled in phase two of the movement program. Cargo may be moved by two helicopters, the CH47D and the UH60A.

(1) The cargo transport phase loads the helicopters by a simplistic loading routine which gives first priority to transporting troops, second priority to loading helicopter slings, and third priority to loading the interior of the helicopter.

(2) The cargo transport module uses gamer inputs to access data on the transportability, rate of movement, and fuel use needed to transport cargo by the helicopters.

(3) Once the appropriate data has been accessed, the cargo transport phase calculates the amount of cargo and the time necessary to transport that cargo along the route(s) indicated by the gamer.

---

[1] This chapter describes the original DIME program before it was changed by ADEA in Ft. Lewis, WA. It has been retained to demonstrate the methodology and structure. The original code is listed in Table 9-3. The code developed by ADEA is listed in Table 9-3a.

C. In either phase, a summary report is generated following the reaching of a final destination by all movement types.

3. DATA FLOW.

The DIME movement program requires a separate data flow for the troop movement phase and the cargo transport phase.

A. The troop movement phase uses two auxiliary data files: mounted movement rates and dismounted movement rates.

(1) The movement rates are for ground forces moving under noncombat conditions. The rates are expressed in km/hr as a function of five geographic area and eight traveling conditions. For a detailed discussion of the troop movement data files, refer to paragraph 4 of this chapter.

(2) The user accesses the appropriate movement rates through a question/answer format. The following inputs must be entered to establish the parameters necessary to access the correct file.

(a) Force: 1 = Blue; 2 = Red.

(b) Pace: 1 = Normal day
2 = Forced day
3 = Normal night
4 = Forced night.

(c) March: MH = Hours to march
RH = Hours to rest.

(d) Start time: The time for start of movement.

(e) Column length: NP = Number of persons in longest column
NV = Number of vehicles in longest column.

(f) Tankers: The number of tankers available.

(g) Distance of leg: 1 = Flat road/trail
2 = Hilly road/trail
3 = Cross country flat
4 = Cross country hilly
5 = Cross country mountainous.

(3) The troop movement phase combines the movement rates with the methodology discussed in paragraph 5 to determine the total time needed to move a march column over a specified route. See Figure 9-1 for a generalized data flow of the troop movement phase.

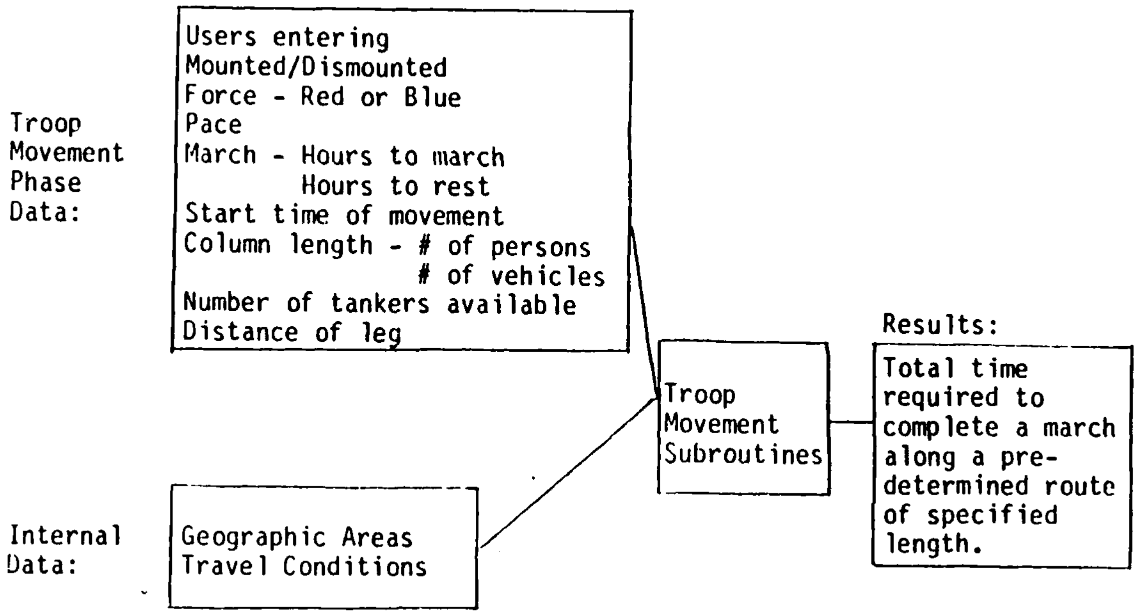


Figure 9-1. Troop movement data flow.

B. The cargo transport phase accesses three sets of data files which provide information on the transportability, rate of movement, and fuel usage needed to transport cargo from one destination to another by the UH60A and the CH47D helicopters. The file structure for these three files will be discussed in paragraph 4 of this chapter.

(1) The cargo transport files, similar to the troop movement files, are accessed through gamer inputs. The following inputs must be supplied by the gamer.

(a) Force: 1 = Blue; 2 = Red.

(b) Start time: The time in hours and minutes for start of mission.

(c) Air temperature: 1 = -10° C
2 = 0° C
3 = 10° C
4 = 20° C
5 = 30° C
6 = 40° C.

(d) Pressure density altitude: 1 = sea level
2 = 2,000 ft
3 = 4,000 ft
4 = 6,000 ft
5 = 8,000 ft
6 = 10,000 ft.

(e) Number of troops to move, including hand-carried weapons and crew members.

(f) Helicopter data:

1. Number of helicopters available for both the CH47D and the UH60A.

2. Loading profile; whether interior and sling or interior only.

3. Refueling status; whether full tanks or one-hour fuel and reserves.

4. CH47D usage only (yes/no).

(g) Routes of travel. Distance in miles along designated routes. Figure 9-2 indicates the seven legs of the route the helicopters may travel.

Leg 1 = Distance from CH47D origin point to load zone.
Leg 2 = Distance from UH60A origin point to load zone.
Leg 3 = Distance from load zone to drop zone.
Leg 4 = Distance from load zone to refuel point.
Leg 5 = Distance from drop zone to refuel point.
Leg 6 = Distance from drop zone to CH47D destination.
Leg 7 = Distance from drop zone to UH60A destination.

(h) Cargo data includes a cargo number identifier (element number) and the number of cargo elements to move.

(2) The cargo transport phase combines the transportability, rate of movement, and fuel use data files with the methodology discussed in paragraph 5 to determine the amount of cargo and the time needed to transport that cargo along the predetermined route(s). See Figure 9-3 for a generalized data flow of the cargo transport phase.

4. FILE STRUCTURE.

The movement program portrays two movement phases (ground and cargo) with each phase having unique data files.

A. Ground movement of troops uses mounted and dismounted movement rates.

(1) The mounted movement rates file structure consists of an 8x5 file with the following records and indexes:

(a) Record 1 contains the movement rates for the five geographic areas when a Blue force is traveling at a normal march pace during the day. This record is composed of the following five indexes:

1. Index 1 contains the movement rate for ground troops traveling on an open road or trail.

2. Index 2 contains the movement rate for ground troops traveling on a hilly road or trail.

3. Index 3 contains the movement rate for ground troops traveling in open cross country.

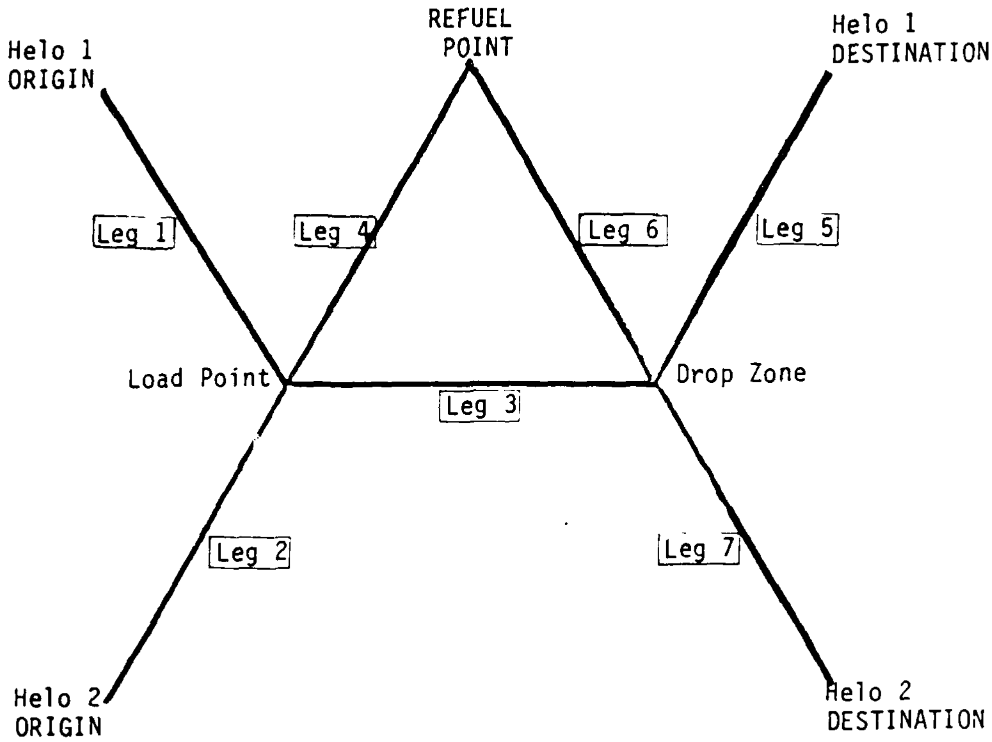


Figure 9-2. Conceptual map of routes to fly.

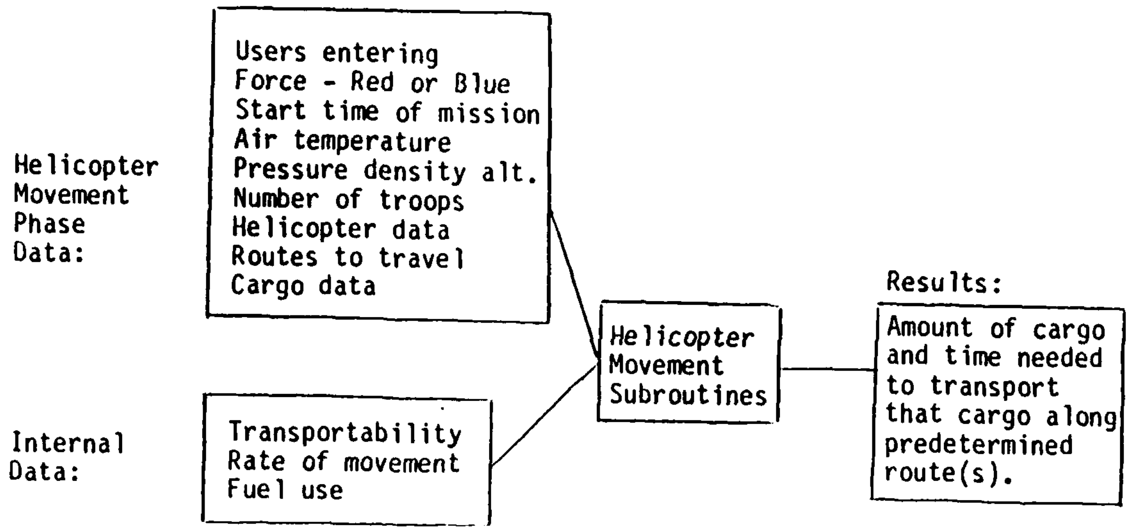


Figure 9-3. Cargo transport data flow.

4. Index 4 contains the movement rate for ground troops traveling in hilly cross country.

5. Index 5 contains the movement rate for ground troops traveling in mountainous cross country.

(b) Record 2 contains the movement rates for the five geographic areas when a Blue force is traveling at forced march pace during the day.

(c) Record 3 contains the movement rates for the five geographic areas when a Blue force is traveling at a normal march pace during the night.

(d) Record 4 contains the movement rates for the five geographic areas when a Blue force is traveling at a forced march pace during the night.

(e) Record 5 contains the movement rates for the five geographic areas when a Red force is traveling at normal march pace during the day.

(f) Record 6 contains the movement rates for the five geographic areas when a Red force is traveling at forced march pace during the day.

(g) Record 7 contains the movement rates for the five geographic areas when a Red force is traveling at normal march pace during the night.

(h) Record 8 contains the movement rates for the five geographic areas when a Red force is traveling at forced march pace during the night.

(2) The dismounted movement rate file is in the same format as the mounted movement rate file previously discussed.

B. The cargo transport module uses three auxiliary files: Copter(*), Cargo(*) and Fuelarray(*).

(1) Copter(*). The Copter(I,J,K) array contains information about the two helicopters used within the cargo transport phase. The array is created interactively by the running of the cargo transport phase and is based on the helicopter's maximum lift weight, maximum fuel capacity, cruising and hovering speed, and time needed to refuel and load cargo. The copter array is dimensioned (2,10,10) and has indexes as follows:

(a) Index I identifies the helicopter being used this mission (1 = CH47D, 2 = UH60A). The cargo movement module allows the user to fly 10 helicopters of each type.

(b) Index J identifies which of the 10 helicopters of the type indicated by index I are being used during this mission. For example, if the third helicopter in the CH47D group were being employed, then the first and second subscripts of Copter(*) would have the values Copter(1,3,K).

(c) Index K contains the following:

K	Description
1	Vacant
2	Weight allowable for transporting of cargo.
3	Weight of crew + fuel reserve + fuel load.
4	Total allowable weight for the helicopter.
5	Vacant
6	Total cargo space, in inches, available on the helicopter: CH47D = 366 inches; UH60A = 151 inches.
7	Total cargo space, in inches, taken up within the helicopter.
8	Current amount of fuel on board the helicopter.
9	Vacant
10	Current weight of helicopter with crew + fuel reserve + fuel load + cargo + weight of the helicopter.

(2) Cargo(*). The Cargo (I,J) array contains information about the cargo being transported by the helicopters within the cargo transport phase and is dimensioned (71,9).

(a) Index I contains the type of cargo, identifiable by its location on the weapons list, which is to be moved by the helicopters. Items 1-70 are equivalent to those listed on the unit status file as the weapons list. Item 71 was added to provide the transporting of strictly personnel.

(b) Index J contains the description of indexes as follows:

J	Description
1	The total number of each item that is being carried by the helicopter.
2	An integer value of 1, 2, or 3 that indicates which helicopter is being used to transport cargo. 1 = CH47D; 2 = UH60A; 3 = Either.
3	Weight of cargo item.

4 Vacant

5 Either a 2 digit or 4 digit number that describes how an item may be carried by a given helicopter.

If item is only transportable by one helicopter (xy).
x = Helicopter used
y = How item may be transported.
1 = Sling only
2 = Sling or interior.
3 = Interior only.

If an item may be transported by both helicopters (wxyz).
w = helicopter one.
x = how item may be transported.
y = helicopter two.
z = how item may be transported

6–8 Vacant

9 Cargo space required for transporting of item by helicopter.

(3) Fuelarray(*). The fuel use file contains the fuel needed to transport cargo of a specified weight over a predetermined route. The fuel use file is a function of helicopter type I, altitude J, temperature K, flying speed L, and weight class M:

I = 1 = CH47D
2 = UH60A.

J = 1 = sea level
2 = 2,000 ft
3 = 4,000 ft
4 = 6,000 ft
5 = 8,000 ft
6 = 10,000 ft.

K = 1 = -10° C
2 = 0° C
3 = 10° C
4 = 20° C.
5 = 30° C.
6 = 40° C.

L = 1 = 40 knots
2 = 60 knots
3 = 80 knots
4 = 100 knots
5 = 120 knots

6 = 140 knots
7 = 160 knots.

M = Weight class for CH47D	Weight class for UH60A
1 = 22,000 to 26,000 lbs	1 = 10,000 to 12,000 lbs
2 = 26,000 to 30,000 lbs	2 = 12,000 to 14,000 lbs
3 = 30,000 to 34,000 lbs	3 = 14,000 to 16,000 lbs
4 = 34,000 to 38,000 lbs	4 = 16,000 to 18,000 lbs
5 = 38,000 to 42,000 lbs	5 = 18,000 to 20,500 lbs
6 = 42,000 to 46,000 lbs	
7 = 46,000 to 50,000 lbs	

5. ALGORITHMS.

The DIME movement program portrays two movement phases with each having unique algorithms. The two sections consist of ground troop movement and cargo transport by helicopters.

A. The troop movement section depicts dismounted and mounted troop movement.

(1) Mounted. The mounted troop movement routine requires the execution of six formulas in order to calculate the total time required to move mounted troops.

(a) Time required to refuel a column of tankers.

$$Trf = (Pr / Tkrs / 4) * 8 \qquad \text{(Eq. 9-1)}$$

where:

Pr = length of a march column.
Tkrs = number of tankers available.
Trf = time required to refuel a column of tankers in hours.
Constants = time required for one tanker to discharge fuel.

(b) Time spent resting (Tr).

$$Tr = \sum_{1=1}^{nl} \left(\left[\frac{Dd_i}{Vdtr} / Mp \right]_I + \left[\frac{Dr_i}{Vntr} / Mp \right]_I \right) * Rp \qquad \text{(Eq. 9-2)}$$

Tr = time spent resting, in hours.
nl = number of legs.
Dd_i = distance of leg i that may be moved under daylight conditions.
Dn_i = distance of leg i that may be moved under night conditions.

Vdtr = velocity possible under day conditions through the current terrain type.
Vntr = Velocity possible under night conditions through the current terrain type.
Mp = march period; march cycle time in hours.
Rp = rest period, in hours.

(c) Time for refueling the force.

$$Trff = \left[\frac{\sum_{i=1}^{nl} (Dd_i + Dn_i)}{Drf} \right] * Trf \qquad \text{(Eq. 9-3)}$$

where:

Trff = total time for refueling the force, in hours.
Drf = distance between refueling points.
Trf = total time required to refuel the column, in hours.

(d) Closure time for mounted troops (Mclose), in hours.

$$Mclose = Pr/180 \qquad \text{(Eq. 9-4)}$$

where:

Pr = length of a march column.
180 = factor necessary for closure of mounted troops.

(e) Total time required for mounted troops to complete a march along a predetermined route (Tmtime).

$$Tmtime = \left[\sum_{i=1}^{nl} \left(\frac{Dd_i}{Vdtr} + \frac{Dn_i}{Vntr} \right) \right] + Max\ (Tr, Trff) + Mclose \qquad \text{(Eq. 9-5)}$$

(2) Dismounted. The dismounted troop movement routine uses equations 9-2 and 9-3 without any modifications. In addition, the dismounted routine requires that two other formulas be executed in order to

calculate the total time required to move dismounted troops over a predetermined route.

(a) Closure time for dismounted troops (Dclose), in hours.

$$Dclose = Pr/360 \qquad \text{(Eq. 9-6)}$$

where:

Pr = length of a column.
360 = factor necessary for closure of dismounted troops.

(b) Total time required for movement of dismounted troops along a predetermined route (Tdtime).

$$Tdtime = \left[\sum_{i=1}^{nl} \left(\frac{Dd_i}{Vdtr} + \frac{Dn_i}{Vntr} \right) \right] + Tr + Dclose \qquad \text{(Eq. 9-7)}$$

B. The cargo movement section consists of formulas which schedule, screen, load, move, and unload cargo.

(1) Fuel usage for leg 1 of the route. This route is the distance between the load zone and the refueling point.

$$Fuel1 = Route1 * Fuelarray/Flightsped \qquad \text{(Eq. 9-8)}$$

where:

Fuel1 = fuel needed for leg 1 of the route.
Route1 = distance in kilometers for the first leg of the route; the distance between the load zone and the refuel point.
Fuelarray = fuel required to transport the cargo given the helicopter type, altitude, temperature, speed and weight class.
Flightsped = flight speed.

(2) Fuel usage for leg 2 of the route. This route is the distance between the refueling point and the drop zone.

$$Fuel2 = Route2 * Fuelarray/Flightsped \qquad \text{(Eq. 9-9)}$$

where:

Fuel2 = fuel needed for leg 2 of the route.
Route2 = distance in kilometers for the second leg of the route; the distance between the refueling point and the drop zone.

(3) Fuel needed is the total fuel for both legs (Fuelneed).

$$Fuelneed = Fuel1 + Fuel2 \qquad \text{(Eq. 9-10)}$$

(4) The maximum weight of cargo which is transportable by each helicopter is calculated as follows:

$$Mwc_i = Wta_i - Wcfrfl_{ci} \qquad \text{(Eq. 9-11)}$$

where:

Mwc_i = maximum weight of cargo transportable by the helicopter where: i = 1 is the CH47D and i = 2 is the UH60A.
Wta_i = the total allowable weight for helicopter i.
$Wcfrfl_{ci}$ = weight of helicopter i + crew + fuel reserve + fuel load + cargo.

(5) Before any item is loaded, the earliest time the event may occur is calculated as:

$$Facilitime = MAX(Load\_area, Fac\_freed) \qquad \text{(Eq. 9-12)}$$

where:

Facilitime = the earliest time the loading of items may occur.
Load_area = the earliest the helicopter can arrive at the loading area.
Fac_freed = the time the load area was last freed for use.

6. "UNITFILE" IMPACT.

The movement program does not impact directly with the "UNITFILE". It is a stand-alone program.

7. CODE.

The DIME movement program consists of a driver and two major phases: troop movement and cargo transport. The driver allows selection of one of the two phases.

A. The troop movement phase depicts dismounted and mounted troop movement.

(1) Both the mounted and dismounted routines follow the same general flow through the code. Figure 9-4 shows this flow.

(2) The program begins by asking the user to indicate whether the ground movement is mounted or dismounted.

(3) Based on the response to the user inputs, the program reads in the appropriate movement rates from an auxiliary data file.

(4) After the proper data is accessed, the program solicits additional information from the gamer concerning the force, march pace, rest periods, column length, and number of tankers available for this march.

(5) At this point, the program combines the movement rates, the gamer responses, and the appropriate formulas to produce the total time required for the movement of troops along the predetermined route.

(6) The mounted and dismounted movement routines produce a printed summary of the total time and the time for closure.

B. The cargo transport phase algorithms consist of routines and formulas which: set up the initial data, establish an event file, schedule, screen, load, move, and unload cargo. See Figure 9-5 for a generalized flow diagram of the cargo transport phase.

(1) Initial setup. The initial setup of the cargo transport phase requires the calling of six subroutines: Datain, Map, Catset, Speedscren, Xfuelcheck, Wtallowed.

(a) Datain is the main data input subroutine. It utilizes the Map, Catset, Speedscren, and Wtallowed routines to check input and set initial parameters. Datain uses a menu procedure for data entry. The operator responses are used to access, build, and load the appropriate data into the Cargo(*), Maxflow(*), and Copter(*) arrays.

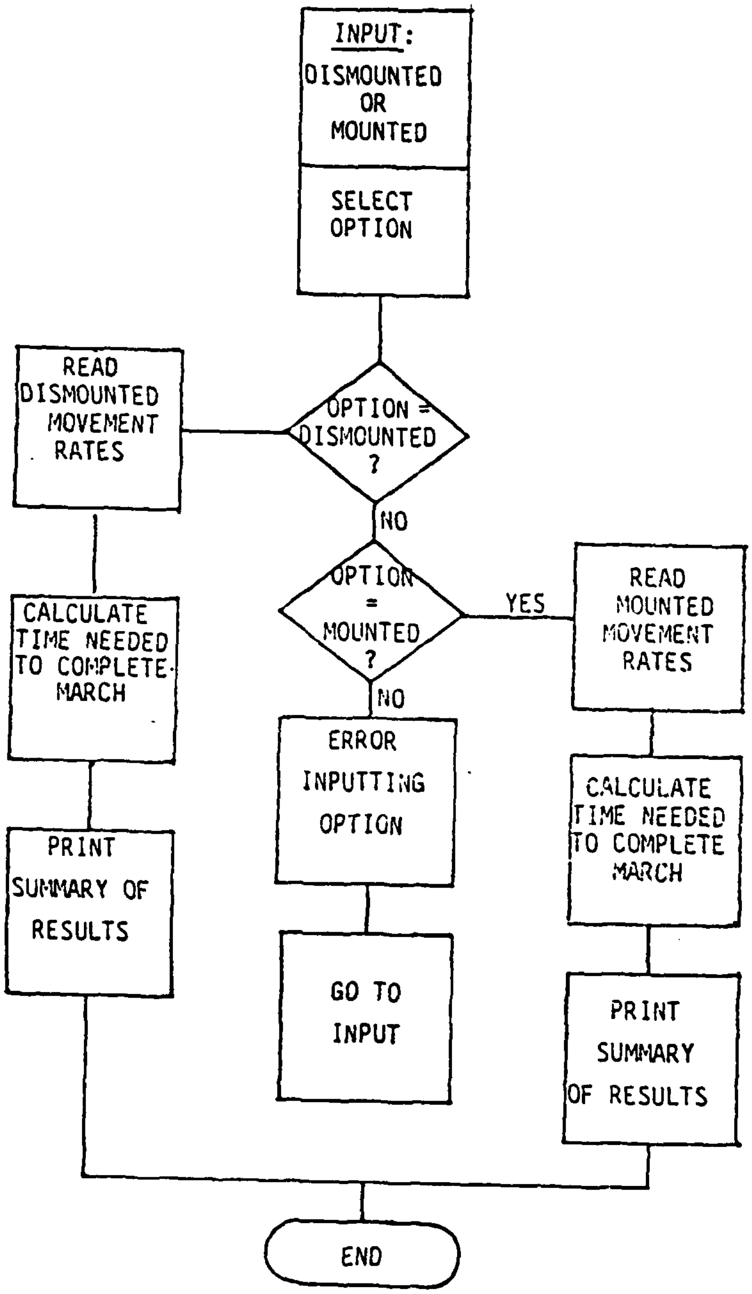


Figure 9-4 General flow of troop movement phase.

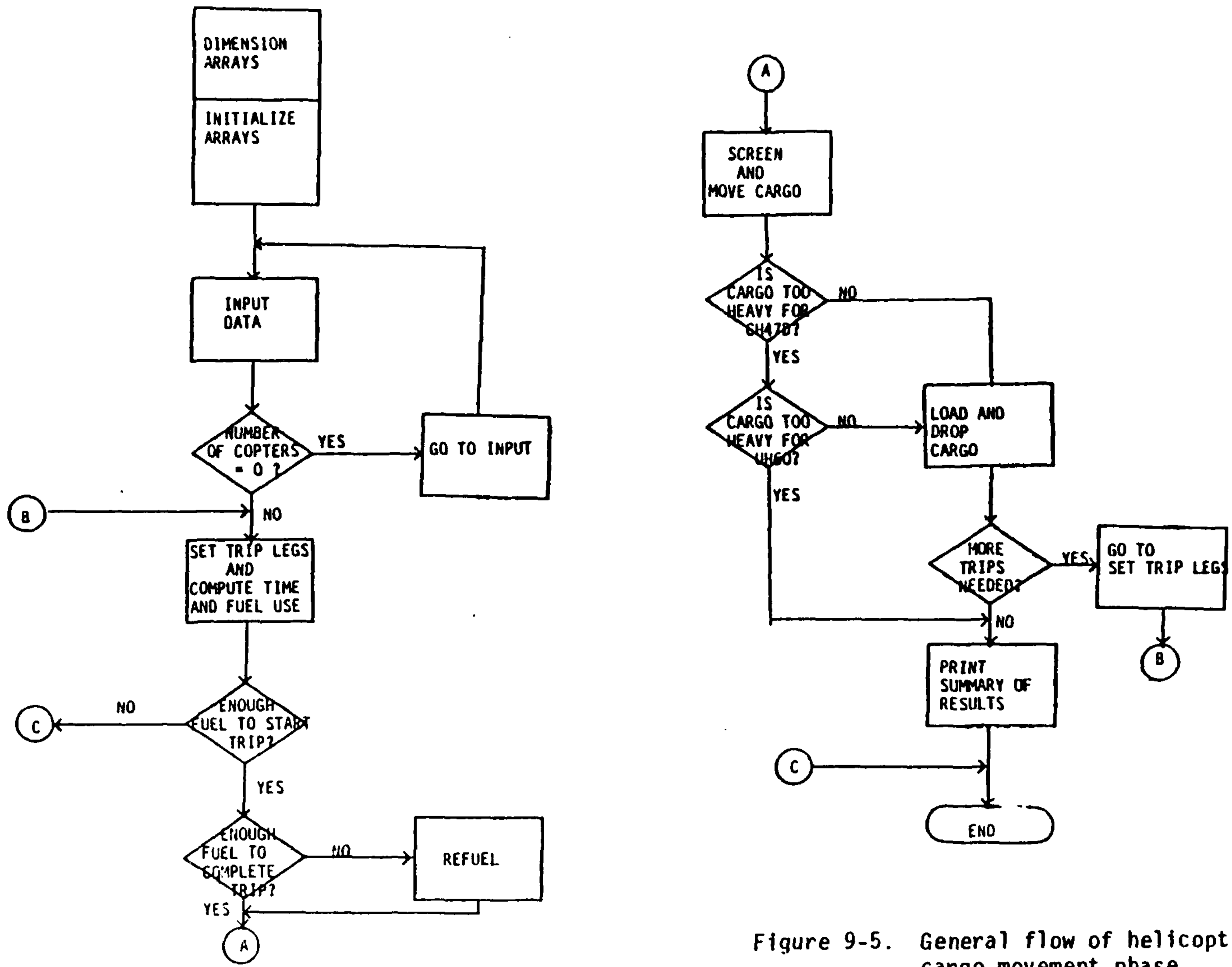


Figure 9-5. General flow of helicopter cargo movement phase.

1. Map. The Map routine provides the operator with a conceptual map of the seven legs of the route. Figure 9-2 shows this map.

2. Catset. The Catset routine converts the temperature and altitude, input by the operator, into the appropriate temperature and altitude categories.

3. Speedscren. The Speedscren routine checks flight speed fuel usage for two legs within the route and speed/weight restrictions to determine if a specified helicopter can transport the indicated cargo.

a. Speed/weight restrictions. The Speedscren routine checks to see if the flight speed, indicated by the operator, will allow for sufficient fuel to complete the movement of the cargo. The helicopter must be able to go to the load zone from the refuel point, then to the drop zone loaded, and finally to the refuel point empty, or the helicopter must be able to go from the refuel point to the load point empty, then to the refuel point loaded, refuel, then go to the drop zone loaded, and finally back to the refuel point unloaded.

b. If either of the speed/weight restrictions are true, then the Speedscren routine calculates the total allowable weight for each helicopter. The weight in pounds allowable for the CH47D is 4000* weight class + 22000. The weight allowable for the UH60A is 2000* weight class + 10,000.

4. Wtallowed. The Wtallowed routine determines the maximum weight of cargo which is transportable by each helicopter.

(b) Once the three data arrays (Maxflow(*), Cargo(*), Copter(*)) have been established, Datain returns control to the driver.

(c) The driver, in turn, calls the Initl routine to place an event on the schedule array which indicates the helicopters have moved from their place of origin to the loading area.

1. Prior to placing this event onto the schedule array, the Initl routine calls the Dfuelcheck routine to ensure the helicopters have the needed fuel. In addition, the Dfuelcheck routine determines the total time which will be required for the helicopters to move between the point of origin and the loading area. This time will serve as a lease time for the next scheduled event, since one event may not start until the first event has been completed.

a. The Dfuelcheck routine uses the same formula as the Xfuelcheck routine to determine the fuel needed. The Dfuelcheck routine returns the total fuel needed to travel over two legs of the route, as does the Xfuelcheck routine. In addition, the Dfuelcheck routine sets an internal flag (Xstatus) if the fuel needed is greater than the fuel available.

b. Once the fuel needed is determined, the Dfuelcheck routine calculates the total time required to travel over two legs of a route.

2. The Initl routine uses the total time calculated by the Dfuelcheck routine to build the event array, Schedule(*). The Schedule (I,J) array has dimensions of (100,3) where each I represents a different event and each J is as follows:

J	Description
1	The helicopter type (1 - UH60A, 2 - CH47D) for event I.
2	The mission (1 - load, 3 - drop) assigned to the helicopter for event I.
3	The earliest time the event I may occur.

3. The fuel used to move from the point of origin to the load area is subtracted from the helicopter's available fuel. This updated value is replaced in the Copter(*) array.

(d) The arrival of the helicopters to the loading area indicates the end of the initial setup phase.

(2) Cargo transport. The cargo transport phase includes methodology and formulas which screen, schedule, load, move, and unload cargo between two points.

(a) Screen. The Cargoscren routine is called to determine if the cargo contained within the Cargo array is transportable by the designated helicopter. The Cargoscren routine accesses the Cargo array and searches for any cargo item which may exceed the maximum weight lift capacity of the helicopter. If the cargo exceeds the weight lift capacity, it is removed from the Cargo array. The Cargoscren routine performs a second search of the Cargo array to determine if a cargo item must be moved by an attached sling. If so, it checks to see if a sling has been placed on the copter. If a sling has not been attached to the helicopter, then the Cargoscren routine removes the sling-only cargo from the Cargo array.

(b) Schedule. The Scheduler routine is called to determine the next event on the Schedule(*) array. The Schedule(*) array, established by the initial setup phase, is searched to determine the event with the earliest time. The main driver uses the mission to call the Loader or the Drop routine for this event.

(c) Load. Provided the mission indicated by the Scheduler routine for the next scheduled event is to load cargo, the main driver calls the Loader routine.

1. The Loader routine will access the Cargo(*) array and assign the items to the helicopter to be transported to the drop zone.

2. However, before any item is loaded, the Loader routine sets the earliest time the event may occur and reinitializes the Schedule(*) array to zero for this event.

3. In addition, the Cargo(*) array is searched for any restrictions which would require the cargo be transported by the larger CH47D helicopter. The Loader routine calls the Chneed routine to determine if the cargo is restricted. The Chneed routine checks the cargo items for excessive weight and then checks the Copter(*) array for environmental conditions which might limit the weight allowable. If the restriction is valid, the Chneed routine sets a flag and returns control to the Loader routine.

4. Using the flag set by the Chneed routine, the Loader routine passes control in one of two directions.

a. If the flag indicates the cargo must be transported by the CH47D, then the helicopter type, for this event, is checked. Provided the helicopter type is for the CH47D, the cargo is loaded. Otherwise, the Coptdone routine is called. The cargo remaining to be loaded must be transported by a CH47D helicopter.

b. If the flag indicates the cargo has no helicopter- type restriction, then the cargo is loaded.

5. The Loader routine loads the cargo in a simplistic method of troops first, slingable items next, and items which must be loaded inside the helicopter last.

a. Thirty-three infantry personnel may be loaded on a CH47D helicopter, while only 11 will fit inside the UH60A. However, the actual number of personnel loaded during any trip may vary. The module assigns each person a weight of 250 pounds. This weight is then multiplied by the number awaiting transport to produce a total weight for all the troops awaiting transport. If this total weight is less than the maximum allowable weight for the helicopter, then all the troops are loaded. Otherwise, the number of personnel which can be loaded is the integer value of the maximum allowable weight divided by 250 lbs. If no troops can be moved, then the module outputs a message to the terminal and stops. Otherwise, each helicopter, of the type indicated, is loaded with troops until no helicopters remain or until no infantry personnel remain. When the troop loading is completed, the sling loads are considered next.

b. Slingable items are loaded in two passes: items which must be transported by sling and items which can be transported by sling or inside the cargo bay. Sling loading is strictly by weight allowance remaining on each helicopter. While there are limitations on the number of slings that each helicopter can employ, the data base as it currently exists ensures the cargo weights preclude violation of these limitations.

c. Following the loading of troops and slings, the interior of the helicopter is loaded. The cargo list is searched for items whose weight and space requirements allow it to be placed inside the helicopter type requested. If there is sufficient weight allowance remaining and the cargo bay has space, then the cargo list is decremented by one for that item. This procedure is repeated until all helicopters, of the type specified, have been used or until all cargo has been loaded.

6. If any item was loaded, a fixed time of 0.25 hours is added to the elapsed time counter to reflect loading time. This time is added to the previously set Faciltime value to reflect the earliest time the facility will be free for the next loading event to occur.

(d) Move. The Uhaul routine is called by the Loader and the Drop routines to calculate the actual movement along the specified routes of travel.

1. Irrespective of which routine calls it, Uhaul checks to see that sufficient fuel exists to travel to the drop area (loaded) and then to the refuel area (unloaded) by calling the Xfuelcheck routine.

a. If there is enough fuel for this direct route, then the fuel needed and the total time for completion of the trip is calculated.

(1) The refuel point computes the time to refuel a helicopter by dividing the amount of fuel needed by 13,320 which is the number of pounds of fuel which may be dispensed in an hour.

(2) This refuel time is added to the travel time to determine the total time for completion of the trip.

b. If insufficient fuel is available for this direct route, then the Xfuelcheck routine returns a flag indicating insufficient fuel.

2. If the direct route is infeasible, Uhaul calls Xfuelcheck a second time to check that sufficient fuel exist to travel to the fuel point (loaded), refuel, travel to the drop area (loaded) and then return to the refuel area.

a. If there is enough fuel for this second route, then the fuel needed and the total time for completion of the trip is calculated. This total time is increased for the refuel time. Uhaul calls the Refuel routine to compute the fueling time and the fuel required.

(1) The Refuel routine computes the time to refuel a helicopter by dividing the amount of fuel needed by 13,320, which is the number of pounds of fuel which may be dispensed in an hour.

(2) This refuel time is added to the travel time to determine the total time for completion of the trip.

(3) The refueling area is not treated as a facility, as are the loading areas and the drop areas. However, a count is kept of the number of times the refuel area is used. Since the refuel area is not treated as a facility, the process is modeled as if both helicopter types are refueled simultaneously.

b. If there is insufficient fuel to complete the trip, a message is generated and the program stops.

3. Finally, the Uhaul routine places the move event onto the Schedule(*) array by assigning the current helicopter type a new mission of load or drop and setting the earliest time for the event to begin. This earliest time is the total time computed to complete either the direct or indirect route.

(e) Unload. Provided the mission indicated by the Scheduler routine for the next scheduled event is to drop cargo, the main driver calls the Drop routine.

1. The Drop routine will unload the cargo passed from the loading area and return the helicopter to either the loading area or to its place of origin.

a. The Drop routine begins the unloading process by resetting the Schedule(*) array for this event to zero.

b. In addition, the Drop routine sets the earliest time the event may occur and begins unloading the cargo.

(1) The unloading process involves resetting the helicopter status to an unloaded state, resetting the cargo space used to zero, and incrementing Faciltime by 0.25 hours.

(2) In addition, the Drop routine determines if a helicopter should be returned to the loading area or to its place of origin by screening the Cargo(*) array for the remaining items.

(a) If cargo remains to be loaded, then the mission for the Schedule(*) array is returning the helicopter to the loading area.

(b) If no cargo remains to be loaded, then the Coptdone routine is called to indicate the helicopter has finished its mission and is returning to its point of origin.

2. If more cargo remains to be transported, the Drop routine returns the helicopters to the loading area. This is accomplished by returning control to the main driver routine.

(3) Summary. When all cargo and troops have been transported to the drop area and all helicopters have arrived at their final destinations, the Schedule(*) array will have been reset to contain only zero entries. The call to Scheduler will find that no events remain to occur. At this point, subroutine Summary is called to provide a screen or hardcopy output of the results of the movement.

(a) Subroutine Summary generates a formatted report giving information about the time required to move the cargo and get the helicopters to their final destinations.

1. Faciltime(3) and Faciltime(4) contain the times the CH47D and the UH60A arrived at their termination points. The greater of these times is chosen as the time for movement termination. Faciltime(2) contains the time the drop area was last free; this is reported as the time until all cargo and troops are delivered to their target area.

2. Summary calculates the number of days needed for the move and creates a string variable to report the 24-hour clock time of the day that the move terminates.

3. Information summarized includes:

a. Number of helicopters of each type used.

b. Number of sorties by each helicopter type.

c. Speed maintained by each helicopter type.

d. Miles flown by each helicopter type.

e. Total fuel dispensed.

f. Number of hours from origin to destination for each helicopter type.

g. A listing of all cargo left behind as too heavy for conditions.

(b) At this point, the program may be rerun with alterations to the gamer inputs used. The operator will be asked if a rerun is desired. A negative response causes the program to terminate and the DIME menu to be loaded. A positive reply causes a message to be printed giving instruction for rerunning the program using some of the previous inputs, and the program stops.

(c) If the rerun option is chosen, the user is directed to type "RUN RERUNIT". This causes all variables to be re-initialized and execution to begin in subroutine Rerunit. Rerunit causes flag Repcount to be set to 1 to indicate that a rerun is taking place. Subroutine Fromfile is called to reenter inputs stored in the previous run. The start time, temperature, altitude, CH47D use option, number of each type of helicopter to use, sling options for each helicopter type, refueling option chosen for each type, and finally the cargo list from the last run are reloaded. The user is asked if the cargo series has changed, and if so, to what. Portions of the main program are utilized to reset name strings and header strings, as well as to initialize capacities, etc. Program control resumes in subroutine Datain, with the display of the data entry menu. The gamer may alter any or all of the input parameters; any not chosen for alteration will retain previous values. The only major pieces of information that must be reentered are the speeds to be flown by each helicopter. The program then proceeds as described.

C. The movement program consists of a driver routine and two phases: ground movement and cargo transport. Each phase has its own algorithms and subroutines consisting of local variables which are not common to both phases. These phases, with their accompanying subroutines and primary variables, are contained in Tables 9-1 and 9-2. Table 9-3 contains a listing of the original movement program code and Table 9-3a contains a listing of ADEA's movement program code.

Table 9-1. Troop movement subroutine table.

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable Description
A. Controls flow of troop movement module.	Header	Determines if gamer wishes to move troops which are dismounted or mounted.	Choice	An integer value which indicates the flow of the troop movement module: 1 = Dismounted movement 2 = Mounted movement 3 = Return to main movement menu
B. Movement of dismounted troops.	Dismount	Reads in dismounted movement rates.	Move_rate(I,J)	An 8x5 real array which contains the dismounted movement rates in km/hr, for eight traveling conditions within five geographic/terrain types I = 1 to 8, where: 1 = Blue/normal/day 2 = Blue/forced/day 3 = Blue/normal/night 4 = Blue/forced/night 5 = Red/normal/day 6 = Red/forced/day 7 = Red/normal/night 8 = Red/forced/night J = 1 to 5, where: 1 = Open road/trail 2 = Hilly road/trail 3 = Open cross country 4 = Hilly cross country 5 = Mountainous cross country

Table 9-1. Troop movement subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable Description
B. Movement of dismounted troops (continued).	Start_dis_mov; Zero_gnd_data; Start_up	Establishes dimensions for new dismounted run: resets values input by previous run to zero; reads in new values input by gamer for present run.	a. Describe$	A character string containing header information for run
			b. Force; Side	An integer value of 1 or 2 which indicates the force being moved where: 1 = Blue 2 = Red
			c. Mar_pace; Pace	An integer value of 1 or 2 which indicates the march pace, where: 1 = Normal 2 = Forced
			d. Mar_time; March	An integer value which indicates the time, in hours, for march time
			e. Rest_time	A real value which indicates the rest time between moves
			f. St_time	An integer value which indicates the starting time for the move, to the nearest hour...based on a 24 hour clock
			g. Persons	An integer value which indicates the number of persons in a marching column

Table 9-1. Troop movement subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable Description
B. Movement of dismounted troops (continued).	Start_dis_mov; Zero_gnd_data; Start_up (continued)		h. Tkrs	An integer value which indicates the number of tankers per march column
			i. Legs(*)	A real array which contains the distance in kilometers, and the road condition for 5 legs: Legs(1) = Distance on first leg of march Legs(2) = Road condition on first leg of march Legs(3) = Distance on second leg of march . . . Legs(10) = Road condition on fifth leg of march
			j. Day_nite	An integer value of 0 or 1 which indicates the light conditions based on day (0) or night (1)

Table 9-1. Troop movement subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable Description
8. Movement of dismounted troops (continued).	Walk_start; Walk_over	Calculate the total time needed to move along entire march, where march is equal to total distance + total rest periods during entire march.	a. Leg_now	A real value which contains the distance of the leg to be moved in current leg of march
			b. Leg_frac	A real value which contains the distance remaining on the current leg to use in computing the time required for a move which is less than 1 hour
			c. Rate	A real value which contains the calculated movement rate as a function of side, pace, and day-night status
			d. Time	A real value which contains the elapsed time for march, where time is equal to total movement time + rest time + refueling time for each leg of march
	Err_check_1; Ch_var	Checks for errors entered by gamer through input routine.	Err	A flag value of 0 or 1 which indicates if an error has occurred during the input routine where: 0 = No error 1 = Error

Table 9-1. Troop movement subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable Description
B. Movement of dismounted troops (concluded).	Prt_dis_out	Provides a printed summary of dismounted movement results.	a. Up_time	A real value which contains the start time adjusted to appear as standard 24-hour figure
			b. Min_time	A real value which contains the calculated number of minutes past the hour at which column closure occurs
			c. Close_time	A real value which contains the time the final element in the column reaches the final destination on the march
C. Movement of mounted troops.	Mount	Reads in mounted movement rates.	Move_rate(*)	An 8x5 real array which contains the mounted movement rates in Km/hr (see detailed discussed in (B) above)
	Start_gnd_mov; Zero_gnd_data; Start_up	Establish dimensions for new mounted run; resets values to zero; reads in new values input by gamer for present run. (Note: Variables discussed under dismounted troops (B) are common to both dismounted and mounted troops.)	a. Refuel_dist	A real value which contains the distance remaining until a refueling stop must occur. This value is initially set to 200Km and reset to 200Km each time a refueling occurs.
			b. Refuel_time	A real value which contains the calculated time required for refueling

Table 9-1. Troop movement subroutine table (concluded).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable Description
C. Movement of mounted troops (concluded).	Gnd_mov_over	Calculates the total time needed to move along entire march.	Time	A real value which contains the total time needed for a column to reach the final destination, where time includes movement, rest, and refueling time enroute.

Table 9-2. Cargo transport subroutine table.

Functional area(s): A. Controls flow of movement module

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Master menu	Determines if gamer wishes to move cargo or troops.	A. Opt$	A number indicating the flow of the movement module: 1 = Cargo transport 2 = Troop movement 3 = Return to DIME master menu

Functional area(s): B. Establishes data arrays

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Datain	Initializes cargo, helicopter and fuel use arrays for new runs. Fill new cargo, fuel and helicopter arrays with new data.	A. Fuelcap(I)	Maximum fuel capacity for helicopter type I. 1 = UH60A 2 = CH47D
		B. Lowwt (I)	Weight of empty helicopter I and crew. 1 = 22499 lbs (CH47D) 2 = 12165 lbs (UH60A)
		C. Typecargo	An integer indicating which force's cargo list is to be loaded. 1 = A force cargo loaded 0 = B force cargo loaded.
		D. Cargo (I,J)	A 71x9 array containing the cargo which is to be moved to the support unit. I = 1-70 - the 70 system elements of the unitfile. = 71 - personnel. J = 1 - integer containing one of the 71 elements carried. = 2 - type of helicopter used = 3 - weight of item = 4 - vacant = 5 - described how item is carried = 6-8 - vacant = 9 - cargo space in helicopter required by item.

Functional area(s): B. Establishes data arrays

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Datain (continued)		E. Maxflow(H,I,J)	A 2x6x6 array containing helicopter fuel use (pounds/hour). H = 1 - UH60A = 2 - CH47D I is altitude above sea level = 1 - 0 ft. = 2 - 2000 ft. = 3 - 4000 ft. = 4 - 6000 ft. = 5 - 8000 ft. = 6 - 10000 ft. J is temperature (degrees celsius) = 1 - -10 = 2 - 0 = 3 - +10 = 4 - +20 = 5 - +30 = 6 - +40
		F. Fuel_array (A,B,C,D,E)	A 2x6x6x7x7 array containing fuel use data. A = 1 - UH60A = 2 - CH47D B = 1-6 (altitude above sea level) C = 1-6 (temperature (degrees celsius)) D is speed (knots) of helicopter = 1 - 40 = 2 - 60 = 3 - 80 = 4 - 100 = 5 - 120 = 6 - 140 = 7 - 160 E is 1-7 weight classes for helicopters weight to max. lift weight for each of two helicopters: CH47D = 1 - 7 UH60A = 1 - 5.

Functional area(s): B. Establishes data arrays

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Datain (continued)		G. Wtcap(I)	An array containing the maximum weight of helicopter I. I = 1 - 50,000 lbs (CH47D) = 2 - 20,500 lbs (UH60A).
		H. Copter(H,I,J)	A 2x10x10 array containing helicopter characteristics. H = 1 - UH60A = 2 - CH47D I = 1 - 10 (number of helicopters available) J = 1-10 and describes dimensions = 1 - Vacant = 2 - Weight allowable for cargo transport = 3 - Weight of crew + fuel reserve + fuel load = 4 - Total permitted weight of helicopter = 5 - Vacant = 6 - Total cargo space (inches) available = 7 - Total cargo space (inches) taken up in helicopter = 8 - Current weight of fuel on helicopter = 9 - Vacant =10 - Weight of crew + fuel reserve + fuel load + cargo + helicopter weight.
		I. Fuelr(I)	Fuel expenditure in 1/2 hour increments as a function of maximum fuel flow for a specified altitude and temperature for helicopter I. I = 1 - CH47D = 2 - UH60A

Table 9-2. Cargo transport subroutine table.

Functional area(s): B. Establishes data arrays

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Datain (continued)		J. Wta(I)	An array containing the maximum weight of helicopter I as a function of speed, temperature, altitude and fuel level input by the gamer.
		K. Schedule (*)	A 100x3 array used to schedule cargo movement events. Up to K = 100 events may be queued for execution, where (K,1) = Helicopter type for event (K,2) = Indicates if cargo is to be dropped of loaded (K,3) = Indicates earliest time helicopter will be available to take part in movement.
		L. Hname$	A character string containing the name of the helicopter moving cargo.

Menu	Provides gamer with input questions and reads responses into cargo, fuel and helicopter arrays.	A. Choice	An integer (1-8) used to choose whether to input all new responses for each run or to change only selected responses for a new run.
			1 = change start time only
			2 = Change temperature only
			3 = Change max. altitude only
			4 = Conagel only # troops to move
			5 = Change only helicopter profiles
			6 = Change only flight routes
			7 = Change only cargo profiles
			8 = Change all entries

		B. Starttime	A real number indicating the start time in hours and minutes.
		C. Temp	A real value indicating the ambient air temperature (celsius) from -10 to +40.

Table 9-2. Cargo transport subroutine table.

Functional area(s): B. Establishes data arrays

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Menu (continued)		D. Altitude	A real value indicating the average altitude flown by each helicopter (0-10000 ft.).
		E. Ncopts(I)	Number of helicopters of type I available for this game turn.
		F. Sling(I)	Flag indicating how cargo may be carried by the helicopter type I. 1 = Load only interior 2 = Load interior and sling as needed.
		G. Fuelusstat(I)	Fuel use status for helicopter I. 1 = 1 hr. fuel load 2 = full tank.
		H. Chstat	Integer indicating restrictions on CH47D helicopter use: 1 = use only for cargo that can be lifted by CH47D 2 = no restrictions on CH47D usage.
Map	Provides a conceptual map of the routes for helicopters. Maximum of 7 routes designated.	A. Route(*)	An array containing the distance nautical miles) for 7 routes. 1 = Distance between original location of helicopter 1 and point where cargo is loaded. 2 = Distance between original location of helicopter 2 and point where cargo is loaded. 3 = Distance between load point and drop zone. 4 = Distance between load point and refuel point. 5 = Distance between refuel point and drop zone. 6 = Distance between drop zone and final destination, helicopter 1. 7 = Distance between drop zone and final destination, helicopter 2.

Table 9-2. Cargo transport subroutine table.

Functional area(s): B. Establishes data arrays

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Catset	Sets the altitude and temperature categories for the fuel array.	A. Tcat	An integer (1-6) indicating the temperature categories for the fuel array. Categories 1 - 6 correspond to temperatures -10C to +40C in 10 degree increments.
		B. Altcat	An integer (1-6) indicating the altitude categories for the fuel array. Categories 1 - 6 correspond to altitudes 0 - 10,000 ft. in 2000 foot increments.
Speedscren	Establishes the flying speed of the helicopters As a function of fuel use and weight lift capacity.	A. Flightsped(I)	The flight speed (knots) of helicopter I. Speed is expressed in increments of of 20 knots from 40 - 160 knots.
		B. Fteller1	Fuel needed to complete the distance from refuel point to load zone to drop zone and to refuel point.
		C. Fteller2	Fuel needed to complete the distance from refuel point to load zone to refuel point.
		D. Fteller3	Fuel needed to complete the distance from refuel point to drop point to refuel point.
		E. Spokay	Flag indicating if speed input by games is feasible based on fuel needed to complete the distances indicated by Fteller1, Fteller2 or Fteller3. 0 = not feasible 1 = feasible

Table 9-2. Cargo transport subroutine table.

Functional area(s): B. Establishes data arrays

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Xfuelcheck	Determines how much fuel is required to complete up to 2 legs of the route.	A. Fuelneed	The amount of fuel needed to travel the distance between 2 legs of the route.

Functional area(s): C. Moves the helicopters to the load zone

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Initl	Calculates the time and fuel needed by each helicopter to move from original point to the load point. Establishes initial load/unload queues on the schedule array.	A. Milesflown	Total miles flown by each helicopter for each of the 7 routes.
		B. Uhtime Chtime	Time required to get from the original point to the load point. Uhtime for UH60A; Chtime for CH47D.
		C. Uhfuel Chfuel	Fuel needed to get from the original point to the load point
Dfuelcheck	Computes fuel and time usage for up to two legs of travel.	A. Xstatus	A flag indication that a helicopter has sufficient fuel to complete two legs of a route. 1 = sufficient fuel 0 = insufficient fuel

Functional area(s): D. Screens cargo for loading

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Cargpscreen	Screens cargo to determine if it can be transported by a helicopter of specific type.	A. Kill	A flag indicating special cargo transportation requirements. 0 = cargo can be transported by either helicopter type 1 = only transportable by UH60A 2 = only transportable by CH47D 3 = requires a sling for transport.

Functional area(s): E. Loads and transports cargo

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Loader	Assigns cargo to helicopter and moves to drop area.	A. Troops	Total number of troops which can be transported by helicopter type specified.
		B. Timeused	A constant of 15 minutes added to each helicopter's total flight time to represent load time.
Refuel	Calculates the time required to refuel helicopter.	A. Timef	Refuel time (hours).
		B. Kallit	A counter indicating the total number of times a helicopter used the refuel point.
Uhaul	Calculates the time required to move from load zone to drop zone.	A. X	The total time needed to complete a trip, including refuel time (if any).

Functional area(s): F. Unloads cargo and returns helicopter to origin or load zone.

Subroutine called	Subroutine function(s)	Primary variables	Variable descriptions
Drop	Sets counter for number of sorties used to transport cargo. Returns helicopters to either load zone or origin point.	A. Dropcount(I)	A counter holding the total number of type I helicopters used to transport cargo.
		B. Qui	A flag indicating if helicopter I is needed for loading additional cargo. 0 = not needed, return to origin point 1 = required to transport more cargo, return to load zone.

Table 9-3. Movement code.

```
10 ! "P9" IS THE MOVEMENT PROGRAM FOR THE DIVISION MAP EXERCISE (
DIME)
20     !
60     !MAIN MENU OF MODULE
80     OPTION BASE 1
90 Mastermenu:     !
100    PRINT USING "@,#"
110    PRINT TABXY(1,7),TAB(25),"DIME MOVEMENT CALCULATOR"
120    PRINT "  "
130    PRINT "THIS PROGRAM CALCULATES MOVEMENT TIME FOR UNITS USIN
G VARIOUS"
140    PRINT "  MODES OF TRANSPORT.  THESE ARE:"
150    PRINT
160    !
170    PRINT "MOVEMENT MENU:  1-HELICOPTER MOVEMENT OF CARGO"
180    PRINT "                2-GROUND MOVEMENT"
190    PRINT "                3-RETURN TO MASTER MENU"
200    INPUT "SELECT OPTION",Opt$
210    IF Opt$="3" THEN LOAD "DIME:HP9134,701,0"
220    IF Opt$="2" THEN
230      CALL P9
240      GOTO Mastermenu
250    END IF
260    PRINT "THIS PROGRAM REQUIRES A FILE NAMED <LOGDATA> ON YOUR
 DISC"
270    PRINT " IT SHOULD BE 1 RECORD OF LENGTH 2400"
280    Repcount=0
290     !DIMENSION ARRAYS NEEDED IN PROGRAM
300 Dimit:DIM Hname$(3)[5],Milesflown(2)
310    DIM Cargo(22,9),Sling(2),Leg(2),Faciltime(4),Schedule(100,3
),Templ(2)
320    DIM Route(7),Ncopts(2),Copter(2,10,10),Fuelarray(2,6,6,7,7)
330    DIM Maxflow(2,6,6),Wtcap(2),Fuelr(2)
340    DIM Fuel(2,2),Headr$(1)[80]
350    DIM Fuelusstat(2),Legwt(2),Dropcount(2)
360    DIM Flightsped(2),Lowwt(2),Wta(2),Fuelcap(2),Totfuelusd(2),
Fhelo(2)
370    DIM Fuelend(2),Dropfuel(2,10),Firer$(2,22)[15],Labels$(7)[4
5]
380    REAL Fuel1,Fuel2
390    Qr$=""
400    IF Repcount=1 THEN GOTO Datget
410     !HELICOPTER NAMES
420 Progtop:Hname$(1)="CH47D"
430    Hname$(2)="UH60"
440    Hname$(3)="BOTH "
450     !NAMES OF THE ELEMENTS OF THE CARGO FILES
460     !FORCE 'A' CARGO
470    Firer$(1,1)="descriptive name for system element 1 restrict
```

Table 9-3. Movement code (continued).

```
ed to 14 char."
480   Firer$(1,2)="descriptive name for system element 2 restrict
ed to 14 char."
490   Firer$(1,3)="                                        3
      "
500   Firer$(1,4)="                                        4
      "
510   Firer$(1,5)="                                        5
      "
520   Firer$(1,6)="                                        6
      "
530   Firer$(1,7)="                                        7
      "
540   Firer$(1,8)="                                        8
      "
550   Firer$(1,9)="                                        9
      "
560   Firer$(1,10)="                                      10
      "
570   Firer$(1,11)="                                      11
      "
580   Firer$(1,12)="                                      12
      "
590   Firer$(1,13)="                                      13
      "
600   Firer$(1,14)="                                      14
      "
610   Firer$(1,15)="                                      15
      "
620   Firer$(1,16)="                                      16
      "
630   Firer$(1,17)="                                      17
      "
640   Firer$(1,18)="                                      18
      "
650   Firer$(1,19)="                                      19
      "
660   Firer$(1,20)="                                      20
      "
670   Firer$(1,21)="                                      21
      "
680   Firer$(1,22)="INFANTRY"  ! always reserved for infantry
690    !FORCE 'B' CARGO
700   Firer$(2,1)="descriptive name for system element 1 restrict
ed to 14 char."
710   Firer$(2,2)="descriptive name for system element 2 restrict
ed to 14 char."
720   Firer$(2,3)="                                        3
      "
730   Firer$(2,4)="                                        4
      "
740   Firer$(2,5)="                                        5
```

Table 9-3. Movement code (continued).

```
                 "
750   Firer$(2,6)="                                      6
                 "
760   Firer$(2,7)="                                      7
                 "
770   Firer$(2,8)="                                      8
                 "
780   Firer$(2,9)="                                      9
                 "
790   Firer$(2,10)="                                    10
                 "
800   Firer$(2,11)="                                    11
                 "
810   Firer$(2,12)="                                    12
                 "
820   Firer$(2,13)="                                    13
                 "
830   Firer$(2,14)="                                    14
                 "
840   Firer$(2,15)="                                    15
                 "
850   Firer$(2,16)="                                    16
                 "
860   Firer$(2,17)="                                    17
                 "
870   Firer$(2,18)="                                    18
                 "
880   Firer$(2,19)="                                    19
                 "
890   Firer$(2,20)="                                    20
                 "
900   Firer$(2,21)="                                    21
                 "
910   Firer$(2,22)="INFANTRY" ! always reserved for infantry
920    !LABELS OF ROUTES 1 - 7
930   Labels$(1)="DISTANCE FROM CH ORIGIN TO LOAD ZONE"
940   Labels$(2)="DISTANCE FROM UH ORIGIN TO LOAD ZONE"
950   Labels$(3)="DISTANCE FROM LOAD ZONE TO DROP ZONE"
960   Labels$(4)="DISTANCE FROM LOAD ZONE TO REFUEL POINT"
970   Labels$(5)="DISTANCE FROM DROP ZONE TO REFUEL POINT"
980   Labels$(6)="DISTANCE FROM DROP ZONE TO CH DESTINATION"
990   Labels$(7)="DISTANCE FROM DROP ZONE TO UH DESTINATION"
1000  Kallit=0
1010   !NUMBER OF SORTIES MADE BY <1=CH, 2=UH>
1020  Dropcount(1)=0
1030  Dropcount(2)=0
1040   !FUEL USAGE COUNTERS
1050  Totfuelusd(1)=0
1060  Totfuelusd(2)=0
1070  Fuelend(1)=0
1080  Fuelend(2)=0
1090   !ARRAY TO RECORD FUEL AVAILABLE AT DROP POINT
```

Table 9-3. Movement code (continued).

```
1100  FOR I=1 TO 2
1110    FOR J=1 TO 10
1120      Dropfuel(I,J)=0
1130    NEXT J
1140  NEXT I
1150  Flaga=0
1160  GOSUB Datain        !ENTER DATA
1170  GOSUB Initl         !SEND COPTERS TO LOAD ZONE
1180  GOSUB Cargoscren    !SCREEN OUT UNMOVABLE CARGO
1190  PRINT "PERFORMING CARGO MOVEMENT---WAIT..."
1200 More:GOSUB Scheduler    !LOOP AS APT FOR EVENTS
1210  SELECT What  !1-READY TO LOAD CARGO   3-READY TO DROP OFF C
ARGO
1220  CASE 1
1230    GOSUB Loader
1240  CASE 3
1250    GOSUB Drop
1260  CASE ELSE
1270    PRINT "SHOULD NOT HAVE GOTTEN HERE!"
1280    LOAD "DIME:HP9134,701,0"
1290  END SELECT
1300  GOTO More
1310  !***********************************************************
***********
1320 Xfuelcheck: !
1330                !DETERMINE HOW MUCH FUEL WOULD BE REQUIRED FOR U
P TO 2 LEGS
1340  Fuel1=0.
1350  Fuel2=0.
1360  Tstvlu=INT((Flightsped(Helo)-20)/20)
1370  IF Leg(1)<>0 THEN
1380    Fuel1=Route(Leg(1))*Fuelarray(Helo,Altcat,Tcat,Tstvlu,Leg
wt(1))/Flightsped(Helo)
1390  END IF
1400  IF Leg(2)<>0 THEN
1410    Fuel2=Route(Leg(2))*Fuelarray(Helo,Altcat,Tcat,Tstvlu,Leg
wt(2))/Flightsped(Helo)
1420  END IF
1430  Fuelneed=Fuel1+Fuel2
1440  RETURN
1450  !***********************************************************
***********
1460 Speedscren: ! SET WT OF HELO TO 0 INITIALLY
1470  Wta(1)=0
1480  Wta(2)=0
1490  FOR Helo=1 TO 2
1500    IF Ncopts(Helo)=0 THEN GOTO Twohelo
1510    PRINT "ENTER FLIGHT SPEED TO USE FOR HELO ";Hname$(Helo);
" BETWEEN 40 AND 160 KNOTS:  ENTER 0 IF NO SPEED HAS WORKED."
1520 Newsp:INPUT Flightsped(Helo)
1530    IF Flightsped(Helo)=0 THEN
1540      Ncopts(Helo)=0
```

Table 9-3. Movement code (continued).

```
1550        PRINT Hname$(Helo);" COULD NOT BE USED UNDER THESE COND
ITIONS"
1560        GOTO Twohelo
1570      END IF
1580      Atstvlu=INT((Flightsped(Helo)-20)/20)
1590      IF Flightsped(Helo)<40 OR Flightsped(Helo)>160 THEN
1600        PRINT "SPEED MUST BE BETWEEN 40 AND 160 KNOTS"
1610        PRINT "RE-ENTER SPEED FOR ";Hname$(Helo)
1620        GOTO Newsp
1630      END IF
1640      Spokay=0
1650      Top=7
1660      IF Helo=2 THEN Top=5
1670      FOR Wcat=Top TO 2 STEP -1
1680        IF Fuelarray(Helo,Altcat,Tcat,Atstvlu,Wcat)>Maxflow(Hel
o,Altcat,Tcat) THEN GOTO Newcat   !SCREEN OUT NON-PERMISSABLE WT/
SPEED COMBINATIONS
1690        Leg(1)=4                          !COPTER MUST BE ABLE TO GO
TO LOAD
1700        Leg(2)=3                          !ZONE FROM REFUEL POINT, TH
EN TO DROP
1710        Legwt(1)=1                        !ZONE LOADED, THEN TO REFUE
L
1720        Legwt(2)=Wcat                                 !OR
1730        GOSUB Xfuelcheck                  !COPTER MUST BE ABLE TO GO
FROM REFUEL
1740        Fteller1=Fuelneed                 !POINT TO LOAD POINT EMPTY,
 THEN TO REFUEL
1750        Leg(1)=5                          !POINT LOADED, REFUEL, THEN
 GO TO DROP
1760        Leg(2)=0                          !ZONE LOADED, THEN BACK TO
REFUEL UNLOADED
1770        GOSUB Xfuelcheck
1780        Fteller1=Fteller1+Fuelneed        !IF EITHER CHECK IS PASSED,
 THE SPEED/WT
1790        Leg(1)=4                          !COMBINATION IS OKAY.
1800        Leg(2)=4
1810        GOSUB Xfuelcheck
1820        Fteller2=Fuelneed
1830        Leg(1)=5
1840        Leg(2)=5
1850        GOSUB Xfuelcheck
1860        Fteller3=Fuelneed
1870        IF Fteller1<=Copter(Helo,1,8) OR (Fteller2<=Copter(Helo
,1,8) AND Fteller3<=Copter(Helo,1,8)) THEN Spokay=1
1880        IF Spokay=1 THEN
1890          Wta(Helo)=Wcat
1900          GOTO Newhelo
1910        END IF
1920 Newcat:NEXT Wcat
1930      PRINT "THAT SPEED NOT FEASIBLE FOR ";Hname$(Helo)
1940      PRINT "ENTER NEW SPEED TO TRY FOR ";Hname$(Helo)
```

Table 9-3. Movement code (continued).

```
1950     GOTO Newsp
1960 Newhelo:!
1970     IF Helo=1 AND Ncopts(Helo)<>0 THEN Wta(1)=Wta(1)*4000+220
00
1980     IF Helo=2 AND Ncopts(Helo)<>0 THEN Wta(2)=Wta(2)*2000+100
00
1990     IF Wta(2)=20000 AND Helo=2 AND Ncopts(Helo)<>0 THEN Wta(2
)=20500
2000 Twohelo:NEXT Helo
2010  RETURN
2020  !***********************************************************
***********
2030 Cargoscren: !
2040  FOR I=1 TO 22
2050     Kill=0
2060     IF Cargo(I,1)=0 THEN GOTO Nexone
2070  !MUST BE SLUNG AND CANNOT
2080     IF Sling(1)=0 AND Cargo(I,5)=11 THEN Kill=1
2090     IF Sling(2)=0 AND Cargo(I,5)=21 THEN Kill=2
2100     IF Sling(1)=0 AND Sling(2)=0 AND Cargo(I,5)=1121 THEN Kil
l=3
2110  !TOO HEAVY EVEN FOR CH47D
2120     IF Ncopts(1)<>0 THEN
2130       IF Cargo(I,3)>Copter(1,1,2) THEN Kill=1
2140     END IF
2150     IF Kill>0 THEN
2160       PRINTER IS 702
2170       PRINT "CARGO ITEM # ";I;" CANNOT BE MOVED: REQUIRES SLI
NG ON OR IS TOO HEAVY FOR ";Hname$(Kill)
2180       IF I=16 THEN
2190         PRINT "MISSILES FOR ITEM 16 ALSO CANNOT BE MOVED"
2200         PRINT "     AND WILL NOT SHOW IN THE FINAL SUMMARY"
2210         Cargo(1,1)=0
2220         PRINTER IS 1
2230       END IF
2240     END IF
2250 Nexone:NEXT I
2260  Kill=0
2270  FOR I=1 TO 22
2280     IF Cargo(I,1)<>0 THEN Kill=1
2290  NEXT I
2300  IF Kill<>1 THEN
2310     PRINT "NO CARGO LEFT TO MOVE; DROPPED OUT IN CARGOSCREN"
2320     PRINT "DONE SCREENING CARGO"
2330     PRINT " "
2340     LOAD "DIME:HP9134,701,0"
2350  END IF
2360  RETURN
2370     !*********************************************************
*************
2380 Coptdone:    !
2390  Legwt(1)=1
```

Table 9-3. Movement code (continued).

```
2400   Legwt(2)=1
2410   Originl=Point          !RECORD WHERE COMING FROM IN CASE IMP
ROPERLY SENT
2420                          !BACK TO LOAD POINT WHEN NO CARGO WAS
 LOADABLE
2430   Leavetime=Faciltime(Point)
2440   IF Dropcount(Helo)>0 THEN
2450     Originl=Point
2460     Point=2
2470   END IF
2480   SELECT Point           !COMING EITHER FROM (1)LOAD ZONE OR (2
)DROP ZONE
2490   CASE 1
2500     Leg(1)=3
2510     Leg(2)=6
2520     IF Helo=1 THEN Leg(2)=7
2530 !        COMPUTE TIME AND FUEL NEEDED TO GET TO DESTINATION
2540     GOSUB Dfuelcheck
2550     X=Tempus
2560     IF Xstatus=0 THEN
2570       Leg(1)=4
2580       Leg(2)=0
2590       GOSUB Dfuelcheck
2600       X=Tempus
2610       Leg(1)=5
2620       Leg(2)=6
2630       IF Helo=2 THEN Leg(2)=7
2640       GOSUB Refuel
2650       GOSUB Dfuelcheck
2660       Tempus=Tempus+X
2670            !ADD IN DISTANCE FROM ENTRY POINT TO DESTINATION
2680       Milesflown(Helo)=Milesflown(Helo)+Route(4)+Route(5)+Rou
te(Leg(2))
2690     ELSE
2700       Milesflown(Helo)=Milesflown(Helo)+Route(3)+Route(Leg(2)
)
2710       GOTO Timer
2720     END IF
2730   CASE 2
2740     SELECT Helo
2750     CASE 1
2760       Leg(1)=6
2770     CASE 2
2780       Leg(1)=7
2790     END SELECT
2800     Leg(2)=0
2810       !IF POINT COMING FROM IS 1, INADVERTANTLY SENT BACK TO
LOAD ZONE
2820       !SO RESET FUEL ON HAND TO WHAT IT WAS WHEN IT WAS AS TH
E DROP ZONE
2830       !ALSO RESET THE TIME IT LEFT FOR IT'S DESTINATION, AND
THE FUEL
```

Table 9-3. Movement code (continued).

```
2840       !USED AS OF THAT TIME
2850     IF Point<>Originl THEN
2860       FOR I=1 TO Ncopts(Helo)
2870         Copter(Helo,I,8)=Dropfuel(Helo,I)
2880       NEXT I
2890       Leavetime=Fl elo(Helo)
2900       Totfuelusd(Helo)=Fuelend(Helo)
2910     END IF
2920     GOSUB Dfuelcheck
2930     Milesflown(Helo)=Milesflown(Helo)+Route(Leg(1))
2940   END SELECT
2950     !ADD TRIP TIME TO TIME STARTED FOR DESTINATION;RECORD IN
2960     !FACILITY 3 FOR CH, FACILITY 4 FOR UH
2970 Timer:Faciltime(Helo+2)=Leavetime+Tempus
2980   FOR I=1 TO Ncopts(Helo)
2990     Copter(Helo,I,8)=Copter(Helo,I,8)-Fuelneed
3000   NEXT I
3010    !HELICOPTER TYPE HELO IS DONE, SO ENSURE THAT NO EVENTS FO
R IT
3020    !REMAIN IN THE SCHEDULE ARRAY
3030   FOR I=1 TO 100
3040     IF Schedule(I,1)=Helo THEN
3050       FOR J=1 TO 3
3060         Schedule(I,J)=0
3070       NEXT J
3080     END IF
3090   NEXT I
3100   IF Helo=1 THEN Flaga=1
3110   RETURN
3120!*************************************************************
***********
3130 Datain:!
3140   PRINT "ENTER DESCRIPTIVE TITLE FOR THIS RUN"
3150   INPUT Headr$(1)
3160   PRINTER IS 702
3170   PRINT " "
3180   PRINT Headr$(1)
3190   PRINT " "
3200   PRINT " "
3210   PRINTER IS 1
3220   INPUT "ENTER <1> FOR FORCE 'A', <2> FOR FORCE 'B'",Typecarg
o
3230   IF Repcount<>0 THEN
3240     PRINT "YOU WILL NEED TO RE-ENTER CARGO DATA IF YOUR LAST"
3250     PRINT "   RUN WAS NOT WITH THE SAME CARGO SERIES AS THIS
ONE"
3260     INPUT "HAS THE CARGO SERIES CHANGED?",Qr$
3270   END IF
3280   PRINT "LOADING ARRAYS"
3290 !INITIALIZE FACILITY TIMES
3300   FOR I=1 TO 4
3310     Faciltime(I)=0
```

Table 9-3. Movement code (continued).

```
3320  NEXT I
3330 !SET FUEL CAPACITIES OF EACH HELO
3340  Fuelcap(1)=6180
3350  Fuelcap(2)=2172
3360 !SET LOWWEST WEIGHT OF HELOS
3370  Lowwt(1)=22499
3380  Lowwt(2)=12165
3390 !****************************************************************
*************
3400 !ENTER CARGO DATA
3410  IF Repcount=1 AND Qr$="N" THEN GOTO Aleph
3420  IF Typecargo=1 THEN
3430    RESTORE 3530      !LOAD FORCE 'A' CARGO
3440  ELSE
3450    RESTORE 3750      !LOAD FORCE 'B' CARGO
3460  END IF
3470  FOR I=1 TO 22
3480    FOR J=1 TO 9
3490      READ Cargo(I,J)
3500    NEXT J
3510  NEXT I
3520  ! FORCE 'A' DATA
3530  DATA                  ! THESE DATA STATEMENTS SHOULD CONTAIN
3540  DATA                  ! DATA FOR THE ARRAY 'CARGO(I,J)'.
3550  DATA                  ! I=1 TO 22; 1-21 ARE THE 21 SYSTEM ELEMEN
TS
3560  DATA                  !   OF THE UNITFILE AND 22 IS ADDED TO PRO
VIDE
3570  DATA                  !   THE TRANSPORTING OF PERSONNEL.
3580  DATA                  ! J=1 TO 9; (1)INTEGER CONTAINING # OF EAC
H OF THE
3590  DATA                  !   22 ELEMENTS BEING CARRIED BY HELO,
3600  DATA                  !   (2)REPRESENTS TYPE OF HELO BEING USED-
-1 for
3610  DATA                  !   CH47D   2 for UH60   3 for either,
3620  DATA                  !   (3)WEIGHT OF ITEM,          (4)VACANT
3630  DATA                  !   (5)DESCRIBES HOW ITEM IS TO BE CARRIED
:
3640  DATA                  !      If item only transportable by one h
elo (xy)
3650  DATA                  !         x = helo used    y = how item is
transported
3660  DATA                  !      If item may be transported by both
helos (wxyz)
3670  DATA                  !         w = CH47D        x = how item is
transported
3680  DATA                  !         y = UH60         z = how item is t
ransported
3690  DATA                  !      Items may be transported in the fol
lowing manners:
3700  DATA                  !         1=sling    2=sling or interior
 3=interior
```

Table 9-3. Movement code (continued).

```
3710  DATA                    !    (6 THRU 8) VACANT
3720  DATA                    !    (9)CARGO SPACE REQUIRED FOR TRANSPORTI
NG
3730  DATA                    !        OF ITEM BY HELO.
3740  !FORCE 'B' DATA
3750  DATA                    ! THESE DATA STATEMENTS SHOULD CONTAIN
3760  DATA                    ! DATA FOR THE ARRAY 'CARGO(I,J)'.
3770  DATA                    ! USE SAME FORM DESCRIBED ABOVE FOR FORCE
'A'.
3780 !***********************************************************
**************
3790 !ENTER MAXFLOW DATA  (MAXIMUM FUEL FLOW ALLOWABLE)
3800 Aleph:RESTORE 3900
3810  FOR H=1 TO 2       !HELO TYPE
3820    FOR I=1 TO 6     !ALTITUDE
3830      FOR J=1 TO 6 !TEMPERATURE
3840        READ Maxflow(H,I,J)
3850        Maxflow(H,I,J)=Maxflow(H,I,J)*100
3860      NEXT J
3870    NEXT I
3880  NEXT H
3890  !  THE FOLLOWING DATA CONTAINS MAXIMUM HELO FUEL USAGE IN P
OUNDS PER HOUR
3900  DATA              !MAXFLOW(H,I,J) CONTAINS:
3910  DATA              ! H=1 TO 2--(1)CH47D   (2)UH60
3920  DATA              ! I=1 TO 6--ALTITUDE above sea level: (1)0 f
t.  (2)2000 ft.
3930  DATA              !      (3)4000 ft.  (4)6000 ft.   (5)8000 ft.
   (6)10000 ft.
3940  DATA              ! J=1 TO 6--TEMP in Centi.: (1)-10  (2)0  (3
)+10
3950  DATA              !      (4)+20   (5)+30   (6)+40
3960 !***********************************************************
**************
3970 !ENTER FUEL ARRAY DATA
3980  RESTORE 4520
3990  FOR Helo=1 TO 2
4000 !   IF Helo=1 THEN PRINT "DATA FOR CH47D";TAB(10);" "
4010 !   IF Helo=2 THEN PRINT "DATA FOR UH60";TAB(10);" "
4020    FOR Alt=1 TO 6   !ALTITUDE
4030      FOR T=1 TO 6   !TEMPERATURE
4040        FOR S=1 TO 7 !SPEED
4050          FOR W=1 TO 7!WEIGHT
4060            READ Fuelarray(Helo,Alt,T,S,W)
4070            Fuelarray(Helo,Alt,T,S,W)=Fuelarray(Helo,Alt,T,S,
W)*100
4080          NEXT W
4090        NEXT S
4100      NEXT T
4110    ! PRINT " "
4120    ! PRINT " "
4130    ! PRINT "ALTITUDE: ";Alt*2000-2000
```

Table 9-3. Movement code (continued).

```
4140    ! PRINT " "
4150    !  FOR T=1 TO 6
4160    !  PRINT "TEMPERATURE: ";(T-2)*10;" DEGREES CENTIGRADE"
4170    !   FOR S=1 TO 7
4180    !    FOR W=1 TO 7
4190    !    PRINT TAB(W*8);Fuelarray(Helo,Alt,T,S,W);
4200    !    NEXT W
4210    !   NEXT S
4220    !   PRINT " "
4230    !  PRINT "***********************************************
***********"
4240    !  NEXT T
4250      BEEP
4260     NEXT Alt
4270 ! PRINT " "
4280 ! PRINT " "
4290  NEXT Helo
4300  PRINTER IS 1
4310 !************************************************************
*************
4320 DATA    ! THE FOLLOWING DATA STATEMENTS ARE TO REPRESENT
4330 DATA    ! THE ARRAY Fuelarray(I,J,K,L,M) WHICH CONTAINS
4340 DATA    ! THE FUEL USAGE UNDER THE FOLLOWING CONDITIONS:
4350 DATA    ! NOTE: ** for conditions unable to fly use 1.E+11
4360 DATA    !   I=HELICOPTER TYPE
4370 DATA    !     (1)CH47
4380 DATA    !     (2)UH60
4390 DATA    !   J=ALTITUDE(FEET PA)
4400 DATA    !     (1)SEA LEVEL     (4)6000
4410 DATA    !     (2)2000          (5)8000
4420 DATA    !     (3)4000          (6)10000
4430 DATA    !   K=TEMPERATURE(C)
4440 DATA    !     (1)-10           (4)+20
4450 DATA    !     (2) 0            (5)+30
4460 DATA    !     (3)+10           (6)+40
4470 DATA    !   L=SPEED IN KNOTS
4480 DATA    !     (1)40            (4)100
4490 DATA    !     (2)60            (5)120           (7)160
4500 DATA    !     (3)80            (6)140
4510 DATA    !   M=WEIGHT CLASS FOR CH47D
4520 DATA    !     (1)22000to26000 (4)34000to38000
4530 DATA    !     (2)26000to30000 (5)38000to42000 (7)46000to50000
4540 DATA    !     (3)30000to34000 (6)42000to46000
4550 DATA    !   M=WEIGHT CLASS FOR UH60
4560 DATA    !     (1)10000to12000 (3)14000to16000
4570 DATA    !     (2)12000to14000 (4)16000to18000 (5)18000to20500
4580 !************************************************************
**************
4590 !ENTER COPTER DATA: FIRST SET MAXIMUM WEIGHT EVER ALLOWABLE
FOR HELO
4600  Wtcap(1)=50000
4610  Wtcap(2)=20500
```

Table 9-3. Movement code (continued).

```
4620  FOR H=1 TO 2  !FOR EACH HELO TYPE
4630    FOR I=1 TO 10!FOR MAX OF 10 HELOS OF THAT TYPE
4640      FOR J=1 TO 10!FOR 10 ELEMENTS FOR EACH HELO
4650        Copter(H,I,J)=0!INITIALIZE ARRAY WITH APT DATA
4660        IF J=4 THEN Copter(H,I,J)=Wtcap(H)
4670        IF J=6 THEN
4680          IF H=1 THEN Copter(H,I,J)=366
4690          IF H=2 THEN Copter(H,I,J)=151
4700        END IF
4710      NEXT J
4720    NEXT I
4730  NEXT H
4740 Menu:PRINT "MENU:";TAB(1);" 1-START TIME";TAB(1);" 2-TEMP";T
AB(1);" 3-MAXIMUM PRESSURE ALTITUDE";TAB(1);" 4-# TROOPS TO MOVE"
;TAB(1);" 5-HELO DATA"
4750  PRINT " 6-FLIGHT ROUTES";TAB(1);" 7-CARGO DATA";TAB(1);" 8-
RUN PROGRAM"
4760  BEEP
4770  INPUT "SELECT MENU ITEM",Choice
4780 Which:ON Choice GOTO L1,L2,L3,L4,L5,L6,L7,L8
4790 La:INPUT "CORRECT? ",A$
4800  IF A$="N" THEN GOTO Which
4810  GOTO Menu
4820 L1:INPUT "ENTER START TIME <HOUR.MINUTES>",Starttime
4830  PRINT "START TIME IS ";Starttime;
4840  Cstarttime=Starttime
4850  Starttime=INT(Starttime)+(Starttime MOD 1)*5/4
4860  GOTO La
4870 L2:INPUT "ENTER TEMPERATURE IN CENTIGRADE (-10 TO 40) ",Temp
4880  PRINT "TEMPERATURE IS: ";Temp;
4890  GOTO La
4900 L3:INPUT "ENTER MAXIMUM PRESSURE ALTITUDE TO BE FLOWN (<=10,
000 FEET)",Altitude
4910  PRINT "ALTITUDE IS: ";Altitude;
4920  GOTO La
4930 L4:INPUT "ENTER NUMBER OF INFANTRY TO MOVE: INCLUDE WEAPON C
REWMEMBERS",Cargo(22,1)
4940  PRINT Cargo(22,1);" TROOPS TO MOVE,CORRECT? <WILL BE STORED
 AS CARGO ITEM 22>";
4950  GOTO La
4960 L5:FOR Helo=1 TO 2
4970 L51:PRINT "ENTER # OF ";Hname$(Helo);" AVAILABLE "
4980    INPUT Ncopts(Helo)
4990    PRINT Ncopts(Helo);
5000    INPUT "CORRECT?",A$
5010    IF A$="N" THEN GOTO L51
5020    IF Ncopts(Helo)=0 THEN
5030      Sling(Helo)=0
5040      GOTO Anhelo
5050    END IF
5060 L52:PRINT "ENTER LOAD PROFILE FOR ";Hname$(Helo);TAB(5);"HOW
 SHOULD HELO BE LOADED?";TAB(5);"0-INTERIOR ONLY";TAB(5);"1-INTER
```

Table 9-3. Movement code (continued).

```
IOR OR EXTERIOR"
5070     INPUT Sling(Helo)
5080     PRINT "LOAD PROFILE FOR ";Hname$(Helo);" IS ";Sling(Helo)
5090     INPUT "CORRECT?",A$
5100     IF A$="N" THEN GOTO L52
5110 L53:INPUT "ENTER FUEL LOAD TO USE: 1-ONE HOUR RANGE   2-FULL
 TANKS",Fuelusstat(Helo)
5120     PRINT "FUEL USE FOR ";Hname$(Helo);" IS ",Fuelusstat(Helo
)
5130     INPUT "CORRECT?",A$
5140     IF A$="N" THEN GOTO L53
5150 L54:IF Helo=1 THEN
5160       PRINT "ENTER CH47D USAGE CODE HERE. 'AS NEEDED' MEANS U
SE IT"
5170       PRINT "ONLY AS LONG AS THERE ARE ITEMS THAT ONLY IT CAN
 MOVE. "
5180       PRINT "'FULLY' MEANS USE IT AS LONG AS THERE IS CARGO T
O MOVE."
5190       INPUT "ENTER CH47D UTILIZATION: 1-AS NEEDED ONLY    2-F
ULLY",Chstat
5200       PRINT "CH47D UTILIZATION IS: ";Chstat
5210       INPUT "CORRECT?",A$
5220       IF A$="N" THEN GOTO L54
5230     END IF
5240 Anhelo:NEXT Helo
5250   GOTO Menu
5260 L6:GOSUB Map     !PRINT CONCEPTUAL MAP OF ROUTES
5270   PRINT "ENTER DATA FOR THE 7 MAP ROUTES: ENTER A <0> IF THAT
 ROUTE WON'T BE USED"
5280   PRINT "ENTER THE DATA AS FOLLOWS:"
5290   PRINT "ENTER THE DISTANCE IN STATUTE MILES FOR EACH ROUTE:"
5300   FOR I=1 TO 7
5310     PRINT TABXY(0,23);"FOR ROUTE: ";I;" ,ENTER THE ";Labels$(
I)
5320     INPUT Route(I)
5330   NEXT I
5340 L61:FOR I=1 TO 7
5350     PRINT "ROUTE ";I;"   DISTANCE: ";Route(I)
5360   NEXT I
5370   INPUT "ANY CHANGES",A$
5380   IF A$="Y" THEN
5390 L62:INPUT "ENTER <ROUTE,DISTANCE>: 0,0 TO CEASE CHANGE",I1,I
2
5400     IF I1=0 THEN GOTO L61
5410     Route(I1)=I2
5420     GOTO L62
5430   END IF
5440             !CONVERT MILES TO NAUTICAL MILES
5450   FOR I=1 TO 7
5460     Route(I)=Route(I)/1.1
5470   NEXT I
5480   GOTO Menu
```

Table 9-3. Movement code (continued).

```
5490 L7:PRINT "ENTER CARGO DATA AS FOLLOWS:"
5500 L71:INPUT "ENTER ELEMENT # OF ITEM, HOW MANY TO MOVE: ENTER
0,0 TO CEASE DATA INPUT",I1,I2
5510   IF I1=0 THEN GOTO L72
5520   IF Typecargo=1 THEN
5530      !FORCE 'A'
5540     IF I1=1 OR I1=6 OR I1=8 OR I1=9 OR I1=9 OR I1=13 THEN
5550       PRINT "ITEM ";I1;" CANNOT BE MOVED BY HELICOPTER"
5560       GOTO L71
5570     END IF
5580   ELSE
5590      !FORCE 'B'
5600     IF I1=1 OR I1=2 OR I1=3 OR I1=10 OR I1=12 OR I1=13 THEN
5610       PRINT "ITEM ";I1;" CANNOT BE MOVED BY HELICOPTER"
5620       GOTO L71
5630     END IF
5640   END IF
5650   PRINT I2;" OF ITEM ";I1;" ARE TO BE MOVED"
5660   INPUT "CORRECT",A$
5670   IF A$="N" THEN GOTO L71
5680   Cargo(I1,1)=I2
5690   IF I1=16 THEN                                 !SEPERATE OUT MIS
SILES
5700     Cargo(1,1)=6*I2
5710     PRINT "MISSILES FOR ITEM 16 WILL BE STORED IN CARGO ITEM
1"
5720   END IF
5730   GOTO L71
5740 L72:PRINT "ELEMENT   NUMBER OF ITEMS TO MOVE"
5750   FOR I=1 TO 22
5760     PRINT TAB(4);I;TAB(22);Cargo(I,1)
5770   NEXT I
5780   INPUT "ANY CHANGES?",A$
5790   IF A$="Y" THEN GOTO L71
5800   GOTO Menu
5810 L8: ASSIGN @P TO "LOGDATA:HP9134,701,0"!RECORD SELECTED DATA
 FOR RERUNS
5820   OUTPUT @P,1;Starttime,Temp,Altitude,Chstat,Ncopts(*),Sling(
*),Fuelusstat(*),Route(*),Cargo(*),Cstarttime
5830   ASSIGN @P TO *
5840   PRINTER IS 702
5850   PRINT " "
5860   PRINT " "
5870   IF Typecargo=1 THEN
5880     PRINT "FORCE 'A' CARGO TO MOVE:"
5890   ELSE
5900     PRINT "FORCE 'B' CARGO TO MOVE:"
5910   END IF
5920   PRINT "SUMMARY OF CARGO TO BE MOVED:"
5930   PRINT "         NOTES: 1- ITEM 1 IS MISSILES FOR ITEM 16"
5940   PRINT "                2- ITEM 22 IS TROOPS INCLUDING WEAPON C
REWS"
```

Table 9-3. Movement code (continued).

```
5950  PRINT " "
5960  PRINT "ITEM:";TAB(20);"# TO MOVE";TAB(35);"ITEM:";TAB(55);"
# TO MOVE"
5970  FOR I=1 TO 11
5980    PRINT I;"-";Firer$(Typecargo,I);TAB(20);Cargo(I,1);TAB(35
);I+11;"-";Firer$(Typecargo,I+11);TAB(55);Cargo(I+11,1)
5990  NEXT I
6000  PRINT " "
6010  PRINT "ROUTE";TAB(13);"DISTANCE"
6020  FOR I=1 TO 7
6030    PRINT TAB(3);I;TAB(13);INT((Route(I)*1.1)*100.)/100.;TAB(
25);Labels$(I)
6040  NEXT I
6050  PRINT " "
6060  PRINT " "
6070  PRINTER IS 1
6080  GOSUB Catset     !SET ALTITUDE,TEMP CATEGORIES FOR ARRAY USE
6090  !
6100  !DETERMINE HOW MUCH FUEL CAN BE CARRIED ON HELO'S
6110  FOR H=1 TO 2
6120    IF Ncopts(H)=0 THEN GOTO Newh1
6130    Fuelr(H)=.5*(Maxflow(H,Altcat,Tcat))
6140    FOR I=1 TO Ncopts(H)
6150      Copter(H,I,3)=Fuelr(H)+Lowwt(H)
6160    NEXT I
6170    IF Fuelusstat(H)=1 THEN      !1 HOUR FUEL
6180      FOR I=1 TO Ncopts(H)
6190        IF Copter(H,I,3)+2*Fuelr(H)<Copter(H,I,3)+Fuelcap(H)
THEN
6200          Copter(H,I,3)=Copter(H,I,3)+2*Fuelr(H)
6210        ELSE
6220          Copter(H,I,3)=Copter(H,I,3)+Fuelcap(H)
6230        END IF
6240        IF Copter(H,I,3)+2*Fuelr(H)>Copter(H,I,3)+Fuelcap(H)
THEN
6250          Copter(H,I,8)=Fuelcap(H)-Fuelr(H)
6260        ELSE
6270          Copter(H,I,8)=2*Fuelr(H)
6280          Fuel(H,1)=Copter(H,I,8)
6290        END IF
6300      NEXT I
6310    ELSE        !FULL FUEL
6320      FOR I=1 TO Ncopts(H)
6330        Copter(H,I,3)=Copter(H,I,3)+Fuelcap(H)-Fuelr(H)
6340        Copter(H,I,8)=Fuelcap(H)-Fuelr(H)
6350        Fuel(H,2)=Copter(H,I,8)
6360      NEXT I
6370    END IF
6380 Newh1:NEXT H
6390  GOSUB Speedscren          !SET SPEED TO FLY AT
6400  FOR I=1 TO 2
6410    IF Wta(I)>Wtcap(I) THEN Wta(I)=Wtcap(I)
```

Table 9-3. Movement code (continued).

```
6420    IF Copter(I,1,3)>Wta(I) THEN
6430      PRINT "FUEL LOAD ON ";Hname$(I);"TOO HEAVY"
6440      PRINT "RERUN WITH NEW ALT/TEMP/SPEED OR FUEL USE"
6450      STOP
6460    END IF
6470  NEXT I
6480  GOSUB Wtallowed   !SET HELO WT ELEMENT IN ARRAY, AND INITIA
LIZE OTHERS
6490  !INITIALIZE SCHEDULE ARRAY
6500  FOR I=1 TO 100    !INITIALIZE SCHEDULE ARRAY TO 0
6510    Schedule(I,1)=0
6520    Schedule(I,2)=0
6530    Schedule(I,3)=0
6540  NEXT I
6550  RETURN
6560!************************************************************
***********
6570 Dfuelcheck:    ! SAME AS XFUELCHECK EXCEPT TIME COMPUTED AND
PUT IN TEMPUS
6580                ! AND A STATUS VARIABLE IS SET IF ANY HELO HAS
 INSUFFICIENT
6590                ! FUEL TO MAKE THE TRIP INDICATED
6600  Xstatus=1
6610  Fuel1=0
6620  Fuel2=0
6630  Templ(1)=0
6640  Templ(2)=0
6650  FOR I=1 TO 2
6660    IF Leg(I)<>0 THEN Templ(I)=Route(Leg(I))/Flightsped(Helo)
6670  NEXT I
6680  Oh1=(Flightsped(Helo)-20)/20
6690  IF Leg(2)<>0 THEN Fuel2=Templ(2)*Fuelarray(Helo,Altcat,Tcat
,Oh1,Legwt(2))
6700  IF Leg(1)<>0 THEN Fuel1=Templ(1)*Fuelarray(Helo,Altcat,Tcat
,Oh1,Legwt(1))
6710  Tempus=Templ(1)+Templ(2)
6720  Fuelneed=Fuel1+Fuel2
6730  FOR I=1 TO Ncopts(Helo)
6740    IF Copter(Helo,I,8)<Fuelneed THEN Xstatus=0
6750  NEXT I
6760  RETURN
6770 !************************************************************
*********
6780 Findwt:   !  DETERMINE THE MAX WEIGHT OF HELO'S  IN THE GROU
P
6790  Maxwt(Helo)=0
6800  FOR I=1 TO Ncopts(Helo)
6810    IF Maxwt(Helo)<Copter(Helo,I,10) THEN Maxwt(Helo)=Copter(
Helo,I,10)
6820  NEXT I
6830  RETURN
6840 !************************************************************
```

Table 9-3. Movement code (continued).

```
*********
6850 Catset:   !   SET TEMP AND ALT CATEGORIES TO USE IN ARRAYS
6860  Tempuse=INT((Temp+.5)/10)*10
6870  IF Tempuse>40 THEN Tempuse=40
6880  IF Tempuse<-10 THEN Tempuse=-10
6890  Tcat=INT((Tempuse+20)/10)
6900  Altcat=INT((Altitude)/2000)+1
6910  IF Altcat<1 THEN Altcat=1
6920  IF Altcat>6 THEN Altcat=6
6930  RETURN
6940    !*************************************************************
*************
6950 Drop:   !
6960  Dropcount(Helo)=Dropcount(Helo)+1          !COUNT # OF SORT
IES
6970      !SET BEGINNING TIME OF EVENT TO GREATER OF WHEN FACILIT
Y LAST
6980      !FREED OR WHEN HELOS CAN ARRIVE, AND ZERO THE EVENT FRO
M THE SCHEDULE
6990  IF Faciltime(2)<Schedule(Which,3) THEN Faciltime(2)=Schedul
e(Which,3)
7000  Schedule(Which,1)=0
7010  Schedule(Which,2)=0
7020  Schedule(Which,3)=0
7030  GOSUB Wtallowed
7040  FOR I=1 TO Ncopts(Helo)
7050    Copter(Helo,I,7)=0
7060 !CARGO SPACE USED RESET TO ZERO
7070  NEXT I
7080  Faciltime(2)=Faciltime(2)+.25     !FLAT 15 MIN. TO UNLOAD C
ARGO
7090  Fhelo(Helo)=Faciltime(2)
7100  IF Helo=1 AND Flaga=0 AND Chstat=1 THEN     !CH47D USED AS
NEEDED AND
7110    GOSUB Chneed                              !NOT ALREADY SE
NT HOME
7120    IF Flag=0 THEN
7130      Point=2
7140      GOSUB Coptdone    !   NOT NEEDED, SEND HOME
7150      Flaga=1
7160      GOTO Outit
7170    END IF
7180  END IF
7190  Qui=0
7200  FOR I=1 TO 22
7210    IF Cargo(I,1)=0 THEN GOTO Nexi   !SCREEN OUT BY DON'T NEE
D REASONS
7220    IF Cargo(I,2)<>Helo AND Cargo(I,2)<>3 THEN GOTO Nexi
7230    IF Cargo(I,2)=Helo AND Sling(Helo)=0 AND (Cargo(I,5) MOD
10=1) THEN GOTO Nexi
7240    IF Cargo(I,3)>Copter(Helo,1,2) THEN GOTO Nexi
7250    IF Cargo(I,2)=3 THEN
```

Table 9-3. Movement code (continued).

```
7260        IF Helo=1 AND Sling(1)=0 AND INT(Cargo(I,5)/100)=11 THE
N GOTO Nexi
7270        IF Helo=2 AND Sling(2)=0 AND (Cargo(I,5) MOD 100)=21 TH
EN GOTO Nexi
7280      END IF
7290      Qui=1      !GETS HERE IF THIS HELO NEEDED TO CARRY IT
7300 Nexi:NEXT I
7310    IF Qui=0 OR Cargoflag=0 THEN    !NOT NEEDED, SEND HOME
7320      Point=2
7330      GOSUB Coptdone
7340      GOTO Outit
7350    END IF
7360    What=4           !SET WHAT=4 FOR UHAUL TO COMPUTE TRIP BACK FR
OM DROP ZONE
7370    Fuelend(Helo)=Totfuelusd(Helo)
7380    FOR I=1 TO Ncopts(Helo)
7390      Dropfuel(Helo,I)=Copter(Helo,I,8)
7400    NEXT I
7410    GOSUB Uhaul
7420 Outit:RETURN
7430    !*********************************************************
*********
7440 Loader:   !
7450    Check=1
7460    Check1=0
7470     !SET TIME FACILITY CAN BE ENGAGED TO BE MAX(TIME FACILITY
LAST FREED,
7480     !TIME HELO'S ARRIVE), THEN ZERO EVENT FROM SCHEDULE ARRAY
7490    IF Faciltime(1)<Schedule(Which,3) THEN Faciltime(1)=Schedul
e(Which,3)
7500    Schedule(Which,1)=0
7510    Schedule(Which,2)=0
7520    Schedule(Which,3)=0
7530     !SCREEN FOR CH NEED
7540     !FLAGA=1 IF CH'S FINISHED ALREADY
7550    IF Flaga=0 AND Helo=1 AND Chstat=1 THEN
7560      GOSUB Chneed
7570      IF Flag=0 THEN
7580 Outit5:Point=1
7590        GOSUB Coptdone
7600        IF Helo=1 THEN Flaga=1
7610        GOTO Quit10
7620      END IF
7630    END IF
7640    Check=0
7650    FOR I=1 TO 22                                 !SEE IF CARGO LEFT
 TO MOVE
7660      IF Cargo(I,1)<>0 THEN Check=Check+1
7670    NEXT I
7680    IF Check=0 THEN GOTO Outit5
7690                                                  ! LOAD INFANTRY
7700    IF Cargo(22,1)<>0 THEN
```

Table 9-3. Movement code (continued).

```
7710     Check1=1
7720     FOR I=1 TO Ncopts(Helo)
7730       IF Helo=1 THEN
7740         Troops=33
7750         IF Troops>Cargo(22,1) THEN Troops=Cargo(22,1)
7760       ELSE
7770         Troops=11
7780         IF Troops>Cargo(22,1) THEN Troops=Cargo(22,1)
7790       END IF
7800       IF (Copter(Helo,I,10)+Troops*250)>Wta(Helo) THEN
7810         Troops=INT((Wta(Helo)-Copter(Helo,I,10))/250)
7820       END IF
7830       IF (Troops<=0) THEN
7840         PRINT "NO WEIGHT ALLOWANCE FOR TROOPS---STOPPING"
7850         STOP
7860       END IF
7870       Cargo(22,1)=Cargo(22,1)-Troops
7880       Copter(Helo,I,10)=Copter(Helo,I,10)+Troops*250
7890       IF (Troops=33 AND Helo=1) OR (Troops=11 AND Helo=2) THE
N
7900         Copter(Helo,I,7)=Copter(Helo,I,6)
7910       ELSE
7920         IF Helo=1 THEN
7930           Copter(Helo,I,7)=Copter(Helo,I,6)-Troops*(366/33)
7940         ELSE
7950           Copter(Helo,I,7)=Copter(Helo,I,6)-Troops*(151/11)
7960         END IF
7970       END IF
7980       PRINTER IS 1
7990       IF Cargo(22,1)<=0 THEN GOTO Loadsling
8000 Anotheri:NEXT I
8010   END IF
8020                                                              !LOAD S
LINGS
8030 Loadsling:!
8040   FOR Copt=1 TO Ncopts(Helo)
8050     FOR Q=1 TO 2
8060       FOR I=1 TO 21
8070         IF Cargo(I,1)<=0 THEN GOTO Newi1
8080         IF Cargo(I,2)<>Helo AND Cargo(I,2)<>3 THEN GOTO Newi1
8090         IF Q=1 THEN                           !SLING ONLY ITEMS
ON PASS 1
8100           IF Cargo(I,2)=3 THEN
8110             IF INT(Cargo(I,5)/100) MOD 10<>1 AND Helo=1 THEN
GOTO Newi1
8120             IF Cargo(I,5) MOD 10<>1 AND Helo=2 THEN GOTO Newi
1
8130           ELSE
8140             IF Cargo(I,5) MOD 10<>1 THEN GOTO Newi1
8150           END IF
8160         END IF
8170         IF Cargo(I,2)=3 THEN    !SCREEN NON-SLINGABLES; LOAD I
```

Table 9-3. Movement code (continued).

```
NTERIOR LAST
8180            IF (INT(Cargo(I,5)/100) MOD 10)=3 AND Helo=1 THEN G
OTO Newil
8190            IF (Cargo(I,5) MOD 10)=3 AND Helo=2 THEN GOTO Newil
8200          ELSE
8210            IF (Cargo(I,5) MOD 10)=3 THEN GOTO Newil
8220          END IF
8230 Reload:          !
8240          IF Copter(Helo,Copt,10)+Cargo(I,3)>Wta(Helo) THEN GOT
O Newil
8250          Copter(Helo,Copt,10)=Copter(Helo,Copt,10)+Cargo(I,3)
8260          Cargo(I,1)=Cargo(I,1)-1
8270          Check1=1
8280          IF Cargo(I,1)<=0 THEN GOTO Newil
8290          GOTO Reload
8300 Newil:NEXT I
8310      NEXT Q
8320   NEXT Copt
8330                                                              !LO
AD INTERIOR
8340 Loadinside: !
8350   FOR Copt=1 TO Ncopts(Helo)
8360     IF Copter(Helo,Copt,7)>=Copter(Helo,Copt,6) THEN GOTO Get
acoptr
8370     FOR Q=1 TO 2
8380       FOR I=1 TO 21
8390          IF Cargo(I,1)<=0 THEN GOTO Nexil
8400          IF Cargo(I,2)<>Helo AND Cargo(I,2)<>3 THEN GOTO Nexil
8410          IF Q=1 THEN                 ! LOAD INTERIOR ONLY ITEMS
ON PASS 1
8420            IF Cargo(I,2)=3 THEN
8430              IF INT(Cargo(I,5)/100) MOD 10<>3 AND Helo=1 THEN
GOTO Nexil
8440              IF Cargo(I,5) MOD 10<>3 AND Helo=2 THEN GOTO Nexi
1
8450            ELSE
8460              IF Cargo(I,5) MOD 10<>3 THEN GOTO Nexil
8470            END IF
8480          END IF
8490          IF Cargo(I,2)=3 THEN
8500            IF INT(Cargo(I,5)/100)=11 AND Helo=1 THEN GOTO Nexi
1
8510            IF Cargo(I,5) MOD 10=1 AND Helo=2 THEN GOTO Nexil
8520          ELSE
8530            IF Cargo(I,5) MOD 10=1 THEN GOTO Nexil
8540          END IF
8550 Morein:          ! IF IT GETS HERE, ITEM CAN GO INSIDE
8560          IF Wta(Helo)<Copter(Helo,Copt,10)+Cargo(I,3) THEN GOT
O Nexil
8570          Sizex=Cargo(I,9)
8580          IF Copter(Helo,Copt,6)<Copter(Helo,Copt,7)+Sizex THEN
 GOTO Nexil
```

Table 9-3. Movement code (continued).

```
8590        Copter(Helo,Copt,7)=Copter(Helo,Copt,7)+Sizex
8600        Copter(Helo,Copt,10)=Copter(Helo,Copt,10)+Cargo(I,3)
8610        Cargo(I,1)=Cargo(I,1)-1
8620        Check1=1
8630        IF Cargo(I,1)<=0 THEN GOTO Nexi1
8640        GOTO Morein
8650 Nexi1:NEXT I
8660    NEXT Q
8670 Getacoptr:NEXT Copt
8680  IF Check1=0 THEN          !NO CARGO COULD BE LOADED THIS TIM
E
8690    Point=1
8700    GOSUB Coptdone
8710    PRINTER IS 1
8720    PRINT " "
8730    PRINT " "
8740    PRINT "NO CARGO COULD BE LOADED ON: ";Hname$(Helo);" THIS
 TIME . HELO SENT TO DESTINATION."
8750    PRINT " "
8760    PRINT " "
8770    GOTO Quit10
8780  END IF
8790       !ALL COPTERS OF TYPE HELO ARE LOADED
8800  Timeused=.25
8810  Faciltime(1)=Faciltime(1)+Timeused      !ADD IN FLAT 15 MIN.
 TO LOAD CARGO
8820  What=2       !SET WHAT=2 FOR UHAUL TO COMPUTE TRIP FROM LOAD
 POINT
8830  GOSUB Uhaul
8840    !TRIP COMPLETED, READY TO GOTO DROP:  SEE WHAT WAS CARRIE
D"
8850  Cargoflag=0
8860  FOR I=1 TO 22
8870    IF Cargo(I,1)<>0 THEN
8880      Cargoflag=1
8890!       PRINT Cargo(I,1);" OF ITEM ";I;" LEFT"
8900    END IF
8910  NEXT I
8920 Quit10:RETURN
8930    !*****************************************************
********
8940 Scheduler:    !
8950  Minx=1.E+6
8960  What=0          !INITIALLY, NOTHING TO BE DONE
8970  Which=0
8980  Helo=0
8990  FOR I=1 TO 100
9000    IF Schedule(I,1)=0 THEN GOTO Loopit
9010    IF Schedule(I,3)<Minx THEN
9020      Minx=Schedule(I,3)
9030      What=Schedule(I,2)      !SET WHAT TO NEXT EVENT TO OCCUR
  <1 OR 3>
```

Table 9-3. Movement code (continued).

```
9040       Which=I
9050     END IF
9060 Loopit:NEXT I
9070   IF What=0 THEN            !NOTHING LEFT IN SCHEDULE ARRAY, PRI
NT SUMMARY
9080     GOSUB Summary
9090   END IF
9100   When=Minx
9110   Helo=Schedule(Which,1)
9120   RETURN
9130 !**************************************************************
*********
9140 Refuel:!
9150   Lowf=500000
9160   FOR I=1 TO Ncopts(Helo)
9170     IF Copter(Helo,I,8)<Lowf THEN Lowf=Copter(Helo,I,8)!FIND
GREATEST
9171                                                          !FUEL
 NEEDED
9180   NEXT I
9190   Fuelin=Fuel(Helo,Fuelusstat(Helo))-Lowf
9200   Timef=Fuelin/(37*6*60) !REFUEL TIME IN HOURS
9210   X=X+Timef
9220   FOR I=1 TO Ncopts(Helo)
9230     Copter(Helo,I,8)=Fuel(Helo,Fuelusstat(Helo))
9240     Totfuelusd(Helo)=Totfuelusd(Helo)+Fuelin
9250   NEXT I
9260   Kallit=Kallit+1  !COUNT NUMBER OF TIMES REFUEL POINT USED
9270   RETURN
9280 !**************************************************************
*****
9290 Uhaul:!
9300   X=0
9310   IF What=2 THEN      !WORKING ON TRIP TO DROP ZONE
9320     GOSUB Findwt
9330     IF Helo=2 THEN   !SET MAXIMUM WEIGHT CLASS OF HELO
9340       Legwt(1)=INT(((Maxwt(Helo)-Lowwt(Helo))/2000)+1)
9350     ELSE
9360       Legwt(1)=INT(((Maxwt(Helo)-Lowwt(Helo))/4000)+1)
9370     END IF
9380     IF Legwt(1)<1 THEN
9390       PRINT "ERROR IN LEG WT IN UHAUL ROUTINE"
9400       LOAD "DIME:HP9134,701,0"
9410     END IF
9420     IF Legwt(1)<1 THEN Legwt(1)=1
9430     Tleg1=Legwt(1)  !STORE LEG 1 WEIGHT
9440     Legwt(2)=1       !LEG WEIGHT OF SECOND LEG=LOWWEST CLASS
9450     Leg(1)=3         !AT LOAD, SEE IF CAN GO LEG3 LOADED AND L
EG5 UNLOADED
9460     Leg(2)=5
9470     GOSUB Xfuelcheck
9480     Till1=Fuelneed
```

Table 9-3. Movement code (continued).

```
9490     IF Till1<=Copter(Helo,1,8) THEN      !CAN DROP CARGO AND R
EFUEL
9500       Legwt(1)=Tleg1
9510       GOTO Trip
9520     END IF
9530     Leg(1)=4
9540     Leg(2)=0
9550     GOSUB Xfuelcheck
9560     Till2=Fuelneed
9570     Leg(1)=5
9580     Leg(2)=5
9590     GOSUB Xfuelcheck
9600     Till3=Fuelneed
9610     Spokay=0
9620     IF Till2<=Copter(Helo,1,8) AND Till3<=Fuel(Helo,Fuelussta
t(Helo)) THEN
9630       Leg(1)=4
9640       Leg(2)=0
9650       GOSUB Dfuelcheck
9660       IF Xstatus=0 THEN LOAD "DIME:HP9134,701,0"
9670       X=Tempus
9680       GOSUB Refuel
9690       Leg(1)=5
9700       Leg(2)=0
9710       GOSUB Dfuelcheck
9720       IF Xstatus=0 THEN LOAD "DIME:HP9134,701,0"
9730       Milesflown(Helo)=Milesflown(Helo)+Route(4)+Route(5)
9740       GOTO Checker
9750     END IF
9760     PRINT "OHBOY SOMETHINGWRONGHASGONE"
9770     LOAD "DIME:HP9134,701,0"
9780   END IF
9790   IF What=4 THEN      !WORKING ON TRIP FROM DROP ZONE TO LOAD
 POINT
9800     IF Helo=2 THEN
9810       Legwt(1)=INT(((Wta(Helo)-Lowwt(Helo))/2000)+1)
9820     ELSE
9830       Legwt(1)=INT(((Wta(Helo)-Lowwt(Helo))/4000)+1)
9840     END IF
9850     Legwt(2)=1
9860     Leg(1)=4
9870     Leg(2)=3
9880     GOSUB Xfuelcheck
9890     IF Fuelneed<=Copter(Helo,1,8) THEN
9900       Legwt(1)=1
9910       GOTO Trip
9920     END IF
9930     Leg(1)=0
9940     Leg(2)=5
9950     GOSUB Xfuelcheck
9960     Till1=Fuelneed
9970     Leg(1)=0
```

Table 9-3. Movement code (continued).

```
9980     Leg(2)=4
9990     GOSUB Xfuelcheck
10000    Till2=Fuelneed
10010    Leg(1)=4
10020    Leg(2)=0
10030    GOSUB Xfuelcheck
10040    Till3=Fuelneed+Till2
10050    IF Till1<=Copter(Helo,1,8) AND Till3<=Fuel(Helo,Fuelussta
t(Helo)) THEN
10060      Leg(1)=0
10070      Leg(2)=5
10080      GOSUB Dfuelcheck
10090      X=Tempus
10100      GOSUB Refuel
10110      Leg(2)=4
10120      GOSUB Dfuelcheck
10130      Milesflown(Helo)=Milesflown(Helo)+Route(4)+Route(5)
10140      GOTO Checker
10150    END IF
10160    PRINT "MYOHMY YOUBLEWITAGAIN"
10170    LOAD "DIME:HP9134,701,0"
10180 END IF
10190 Trip:!
10200 Leg(1)=3
10210 Leg(2)=0
10220 Milesflown(Helo)=Milesflown(Helo)+Route(3)
10230 GOSUB Dfuelcheck
10240 Checker:X=X+Tempus
10250 FOR I=1 TO Ncopts(Helo)
10260    Copter(Helo,I,8)=Copter(Helo,I,8)-Fuelneed
10270 NEXT I
10280 IF What=2 THEN That=3      !TRIP TO DROP COMPUTED, SCHEDULE
CARGO DROP
10290 IF What=4 THEN That=1      !TRIP TO LOAD COMPUTED, SCHEDULE
LOADING
10300 FOR I=1 TO 100
10310    IF Schedule(I,1)=0 THEN
10320      Schedule(I,1)=Helo
10330      Schedule(I,2)=That
10340      Schedule(I,3)=Faciltime(What/2)+X !RECORD EARLIEST HELO
'S CAN ARRIVE
10341                !AT <WHAT=2,LOAD> <WHAT=4,DROP>  FACILITY=TIME
LEFT+TRIP TIME
10350      GOTO Outit1
10360    END IF
10370 NEXT I
10380 Outit1:RETURN
10390!***********************************************************
**********
10400 Chneed:!   CHECK ALL REASONS WHY CH47D MAY STILL BE NEEDED
10410 Flag=0
10420 FOR I=1 TO 22
```

Table 9-3. Movement code (continued).

```
10430    IF Cargo(I,1)=0 THEN GOTO Ani10
10440    IF Cargo(I,2)=1 AND Copter(1,1,2)>=Cargo(I,3) THEN Flag=1
10450    IF Copter(2,1,2)<Cargo(I,3) AND Copter(1,1,2)>=Cargo(I,3)
 AND NOT (Sling(1)=0 AND INT(Cargo(I,5)/100)=11) THEN Flag=1
10460    IF Sling(2)=0 AND Sling(1)<>0 AND Cargo(I,5)=1121. AND Co
pter(1,1,2)>=Cargo(I,3) THEN Flag=1
10470    IF Cargo(I,2)=3 THEN
10480      IF Sling(1)=0 AND INT(Cargo(I,5)/100)=11 THEN GOTO Ani1
0
10490      IF Sling(2)=0 AND (Cargo(I,5) MOD 100)=21 AND Copter(1,
1,2)>=Cargo(I,3) THEN Flag=1
10500    END IF
10510 Ani10:NEXT I
10520 RETURN
10530!***************************************************************
************
10540 Wtallowed:!  SET WT ELEMENTS OF COPTER ARRAY
10550 FOR Xhelo=1 TO 2
10560   FOR I=1 TO Ncopts(Xhelo)   !WTA=MAX WT OF HELO (1) AND (2)
10570     IF Ncopts(Xhelo)=0 THEN GOTO Getanewi
10580     Copter(Xhelo,I,4)=Wta(Xhelo)     !MAX WT ALLOWED AT THIS
 TEMP,ALT,SPD
10590     Copter(Xhelo,I,2)=Wta(Xhelo)-Copter(Xhelo,I,3)   !MAX CA
RGO WT ALLOWED
10600     Copter(Xhelo,I,10)=Copter(Xhelo,I,3)   !CURRENT HELO WT
COUNTER
10610 Getanewi:NEXT I
10620 NEXT Xhelo
10630 RETURN
10640!***************************************************************
**********
10650 Summary:!
10660 INPUT "SCREEN (S) OR PRINTER (P) SUMMARY?",C$
10670 PRINT "PRINTING SUMMARY"
10680 IF C$="P" THEN PRINTER IS 702
10690 Timedone=Faciltime(3)
10700 IF Timedone<Faciltime(4) THEN Timedone=Faciltime(4)
10710 FOR I=1 TO 8
10720   PRINT " "
10730 NEXT I
10740 Daystaken=INT(Timedone/24)
10750 Clockdone=(Timedone+Starttime) MOD 24
10760 Clockhr=INT(Clockdone)
10770 IF Clockhr<10 THEN
10780   Clockhr$="0"&VAL$(Clockhr)
10790 ELSE
10800   Clockhr$=VAL$(Clockhr)
10810 END IF
10820 Clockmin=INT(DROUND((Clockdone MOD 1)*60,2))
10830 PRINT "START TIME WAS: ";Cstarttime
10840 IF Chstat=1 THEN
10850   PRINT "CH47D  USED ONLY AS NEEDED"
```

Table 9-3. Movement code (continued).

```
10860 ELSE
10870   PRINT "CH47D  USED FULLY"
10880 END IF
10890 PRINT "TIME REQUIRED (IN HOURS) UNTIL ALL CARGO HAS BEEN DE
LIVERED: ";
10900 PRINT INT(Faciltime(2)*100)/100
10910 PRINT " "
10920 PRINT "LAST COPTER AT DESTINATION AT ";Clockhr$;":";
10930 IF Clockmin<10 THEN
10940   PRINT "0"&VAL$(Clockmin)
10950 ELSE
10960   PRINT VAL$(Clockmin)
10970 END IF
10980 PRINT " "
10990 PRINT " "
11000 PRINT TAB(20);Hname$(1);TAB(30);Hname$(2)
11010 PRINT " "
11020 PRINT "# OF HELOS USED";TAB(20);Ncopts(1);TAB(30);Ncopts(2)
11030 PRINT "NUMBER OF SORTIES";TAB(20);Dropcount(1);TAB(30);Drop
count(2)
11040 PRINT "SPEED FLOWN";TAB(20);Flightsped(1);TAB(30);Flightspe
d(2)
11050 PRINT "MILES FLOWN";TAB(20);1.1*Milesflown(1);TAB(30);1.1*M
ilesflown(2)
11060 PRINT "FUEL DISPENSED";TAB(20);INT(Totfuelusd(1)*100/6)/100
;TAB(30);INT(Totfuelusd(2)*100/6)/100
11070 FOR I=3 TO 4
11080   Xqr=Faciltime(I)*10.
11090   IF Xqr MOD 1>=.5 THEN Xqr=Xqr+1.
11100   Faciltime(I)=Xqr/10.
11110 NEXT I
11120 PRINT "HOURS TILL DONE";TAB(20);INT(Faciltime(3)*10)/10;TAB
(30);INT(Faciltime(4)*10)/10
11130 PRINT " "
11140 PRINT TAB(15);"TOTAL SORTIES FLOWN: ";Dropcount(1)+Dropcoun
t(2)
11150 PRINT TAB(15);"TOTAL FUEL DISPENSED: ";INT((Totfuelusd(1)+T
otfuelusd(2))*100/6)/100;" GALLONS"
11160 PRINT TAB(15);"TOTAL MILES FLOWN: ";1.1*(Milesflown(1)+Mile
sflown(2))
11170 PRINT TAB(15);"REFUELING POINT USED ";Kallit;" TIMES"
11180 PRINT " "
11190 PRINT " "
11200 PRINT " "
11210 PRINT " "
11220 Checklate=0
11230 FOR I=1 TO 22
11240   IF Cargo(I,1)<>0 THEN Checklate=1
11250 NEXT I
11260 IF Checklate=1 THEN
11270   PRINT "CARGO ITEMS LEFT BEHIND: TOO HEAVY FOR CONDITIONS
OR CANNOT BE SLUNG ON APPROPRIATE HELOCOPTER"
```

Table 9-3. Movement code (continued).

```
11280    PRINT " "
11290    FOR I=1 TO 22
11300       IF Cargo(I,1)<>0 THEN PRINT Cargo(I,1);" OF THE ";Firer
$(Typecargo,I)
11310    NEXT I
11320 END IF
11330 PRINT " "
11340 PRINT " "
11350 PRINTER IS 1
11360 PRINT "IF THESE RESULTS ARE UNSATISFACTORY, YOU MAY MAKE CH
ANGES TO"
11370 PRINT "YOUR INITIAL SET-UP"
11380 INPUT "RERUN WITH NEW SET-UP?",A$
11390 IF A$="N" THEN GOTO Whoopee
11400 PRINT "ENTER <RUN RERUNIT> AND FOLLOW DIRECTIONS"
11410 Whoopee:PRINT "PROGRAM DONE"
11420 STOP
11430 LOAD "DIME:HP9134,701,0"
11440 RETURN
11450!****************************************************************
**************
11460 Initl:!
11470 IF Ncopts(1)=0 AND Ncopts(2)=0 THEN
11480    PRINT "YOU DIDN'T GIVE ME ANY HELICOPTERS--STOPPING!"
11490    LOAD "DIME:HP9134,701,0"
11500 END IF
11510 FOR Helo=1 TO 2
11520    Milesflown(Helo)=0
11530    IF Ncopts(Helo)<>0 THEN Milesflown(Helo)=Milesflown(Helo)
+Route(Helo)
11540 NEXT Helo
11550 Helo=2
11560 Legwt(1)=1     !SET INITIAL TRIP LEGS TO BE AT MINIMUM WEIGH
T
11570 Legwt(2)=1
11580 Leg(1)=2       !UH TRIP ON LEG 2
11590 Leg(2)=0
11600 IF Ncopts(2)=0 THEN GOTO Aleph1
11610 GOSUB Dfuelcheck        !COMPUTE TRIP TIME, FUEL USED TO LOA
D POINT
11620 IF Xstatus=0 THEN GOTO Oops
11630 Uhtime=Tempus           !TIME TO GET TO LOAD POINT
11640 Uhfuel=Fuelneed
11650 FOR I=1 TO Ncopts(2)
11660    Copter(2,I,8)=Copter(2,I,8)-Uhfuel  !DECRIMENT FUEL ON HA
ND BY TRIP USE
11670 NEXT I
11680 Schedule(2,1)=2       !SCHEDULE THE UH
11690 Schedule(2,2)=1       !TO LOAD CARGO
11700 Schedule(2,3)=Uhtime !NO EARLIER THAN IT'S ARRIVAL AT LOAD
POINT
11710 Aleph1:IF Ncopts(1)=0 THEN GOTO Outer
```

Table 9-3. Movement code (continued).

```
11720 Helo=1
11730 Leg(1)=1              !CH TRIP LEG 1 TO LOAD POINT
11740 GOSUB Dfuelcheck
11750 IF Xstatus=0 THEN GOTO Oops
11760 Chtime=Tempus         !TIME FOR UH TO GET TO LOAD POINT
11770 Chfuel=Fuelneed
11780 Schedule(1,1)=1       !SCHEDULE THE CH
11790 Schedule(1,2)=1       !TO LOAD CARGO
11800 Schedule(1,3)=Chtime !NO EARLIER THAN IT'S ARRIVAL AT LOAD
POINT
11810 FOR I=1 TO Ncopts(1)
11820   Copter(1,I,8)=Copter(1,I,8)-Chfuel!DECRIMENT FUEL BY TRIP
 USE
11830 NEXT I
11840 GOTO Outer
11850 Oops:PRINT "BOMBED OFF AT START; CAN'T GET TO LOAD POINT."
11860 LOAD "DIME:HP9134,701,0"
11870 Outer:RETURN
11880!*************************************************************
*********
11890 Fromfile:!  RE-ENTER DATA FROM LOGFILE FOR RERUN
11900 PRINT "ENTERING DATA FROM DISC"
11910 ASSIGN @P TO "LOGDATA:HP9134,701,0"
11920 ENTER @P,1;Starttime,Temp,Altitude,Chstat,Ncopts(*),Sling(*
),Fuelusstat(*),Route(*),Cargo(*),Cstarttime
11930 ASSIGN @P TO *
11940 RETURN
11950!*************************************************************
*************
11960 !RERUN PROGRAM WITH SOME CHANGES
11970 Rerunit:!
11980 Repcount=1     !SET FLAG INDICATING IT'S A RERUN
11990 GOTO Dimit     !REDIMENSION ARRAYS AND REINITIALIZE WHERE AP
T
12000 Datget:GOSUB Fromfile     !RELOAD DATA FROM LAST RUN
12010 GOTO Progtop  !RUN PROGRAM
12020!*************************************************************
***************
12030 Map:!   CONCEPTUAL MAP OF ROUTES TO FLY
12040 PRINT USING "@"
12050 PRINT "       CH47D      REFUEL       CH47D"
12060 PRINT "      ORIGIN      POINT     DESTINATION"
12070 PRINT "       X            X            X"
12080 PRINT "         -         - -          -"
12090 PRINT "          -       -   -        -"
12100 PRINT "           1     4     5      6"
12110 PRINT "            -   -       -    -"
12120 PRINT "             - -         - -"
12130 PRINT "LOAD POINT  X-----3-----X   DROP ZONE"
12140 PRINT "             -             -"
12150 PRINT "            -               -"
12160 PRINT "           2                 7"
```

Table 9-3. Movement code (continued).

```
12170 PRINT "          -                         -"
12180 PRINT "         -                           -"
12190 PRINT "        X                             X"
12200 PRINT "       UH60                          UH60"
12210 PRINT "      ORIGIN                    DESTINATION"
12220 RETURN
12230 END
12240 !*********************************************************
****************************************************************
*****************************
12250 SUB P9
12260   REM "P9" IS THE MOVEMENT CALCULATOR PROGRAM FOR DIME.  CO
DED BY
12270 !   MAJ T. REISCHL, OAB, CAORA, A/V 552-4613/5122. THIS PRO
GRAM
12280 !   LAST CHANGED ON 12 JUNE 1983. HELO PORTION CODED BY
12290 !   TERRY D. BRASHLEY <SAME ADDRESS>, LAST CHANGED ON 30DEC
83.
12300   OPTION BASE 1
12310   DIM Move_rate(8,5),Leg(10),P1$[4],P2$[6],Describe$[40]
12320   INTEGER I,J,K
12330 !
12340 ! PRINT OPTION PAGE
12350 Header:PRINT USING "@"
12360   PRINTER IS 702
12370   PRINT USING "@,#"
12380   PRINTER IS 1
12390   PRINT TABXY(1,7),TAB(25),"GROUND MOVEMENT CALCULATOR"
12400   PRINT "  "
12410   PRINT "THIS PROGRAM CALCULATES MOVEMENT TIME FOR UNITS US
ING VARIOUS"
12420   PRINT "  MODES OF TRANSPORT.  THESE ARE:"
12430   PRINT
12440   PRINT "          1.  DISMOUNTED"
12450   PRINT "          2.  GROUND (WHEEL/TRACK VEHICLE)"
12460   PRINT "          3.  RETURN TO MOVEMENT MENU"
12470   INPUT "ENTER DESIRED OPTION:",Choice
12480   IF Choice<>1 AND Choice<>2 AND Choice<>3 THEN Header
12490   IF Choice=3 THEN
12500     GOTO Backuptop
12510   END IF
12520   ON Choice GOSUB Dismount,Ground
12530   GOTO Header
12540 !
12550 ! ******************* END OF MAIN PROGRAM *****************
***************
12560 !
12570 Dismount: !SBR DISMOUNT CALCULATES DISMOUNTED MOVEMENT TIME
S
12580 !
12590   RESTORE 12610
12600   READ Move_rate(*)
```

Table 9-3. Movement code (continued).

```
12610    DATA      ! THE FOLLOWING IS THE DISMOUNTED MOVEMENT RATE
VELOCITIES
12620    DATA      ! (Km/Hr) IN NONCOMBAT CONDITIONS FOR 1 BLUE AND
 1 RED FORCE.
12630    DATA      ! Move_rate(I,J) CONTAINS:
12640    DATA      !   I=GEOGRAPHIC AREA              J=TRAVEL CONDI
TIONS
12650    DATA      !    (1)open road/trail             (1)blue norma
l day
12660    DATA      !    (2)hilly road/trail            (2)blue force
d day
12670    DATA      !    (3)open cross-country          (3)blue norma
l night
12680    DATA      !    (4)hilly cross-country         (4)blue force
d night
12690    DATA      !    (5)mountainous x-country       (5-8)same as
above for red
12700 !
12710 !
12720 Start_dis_mov:PRINT USING "@"
12730    PRINT TABXY(26,17),"DISMOUNTED MOVEMENT MODULE"
12740    Move_type=1
12750    GOSUB Start_up
12760    GOTO Walk_start
12770 Start_up:GOSUB Zero_gnd_data
12780    INPUT "ENTER DESCRIPTION: ",Describe$
12790    PRINT "MARCH FACTORS ARE:"
12800    PRINT "FORCE (1,2), MARCH PACE(1,2), MARCH TIME, REST TIM
E,START TIME"
12810    PRINT "NUMBER OF PERSONS, NUMBER OF TANKERS"
12820    INPUT "ENTER MARCH FACTORS: ",Force,Mar_pace,Mar_time,Res
t_time,St_time,Persons,Tkrs
12830    INPUT "ENTER TERRAIN PROFILE FOR 5 LEGS:  DIST(km) AND LE
G TYPE ",Leg(*)
12840    Type_bound=5
12850    IF Choice=2 THEN Type_bound=4
12860    FOR Ck_loop=1 TO 9 STEP 2
12870      CALL Ck_var("TERRAIN PROFILE DIST(km)","THRU",Leg(Ck_lo
op),0,40)
12880    NEXT Ck_loop
12890    FOR Ck_loop=2 TO 10 STEP 2
12900      CALL Ck_var("TERRAIN PROFILE LEG TYPE","TO",Leg(Ck_loop
),1,Type_bound)
12910    NEXT Ck_loop
12920 !
12930    GOSUB Err_check_1
12940    IF Err=1 THEN 12820
12950 !    SET LIGHT CONDITIONS
12960    IF St_time>=6 AND St_time<=18 THEN
12970      Day_nite=0                                          !D
AY=0
12980      Vis_left=18-St_time
```

Table 9-3. Movement code (continued).

```
12990    ELSE
13000      Day_nite=1                                                    !N
IGHT=1
13010      IF St_time>18 THEN
13020        Vis_left=St_time-12
13030      ELSE
13040        Vis_left=St_time+6
13050      END IF
13060    END IF
13070    Pace=Mar_pace
13080    Side=4*(Force-1)
13090    March=Mar_time
13100    RETURN
13110 !
13120 ! **************************************************************
***************
13130 !
13140 Walk_start:FOR I=1 TO 9 STEP 2                          !   BEGIN MOVEME
NT CALCULATION
13150      IF Leg(I)=0 OR Leg(I+1)=0 THEN Walk_over
13160 !      DETERMINE MOVEMENT TIME
13170      Leg_now=Leg(I)
13180      LOOP
13190        Rate=Side+Pace+Day_nite*2
13200        Leg_frac=Leg_now
13210        Leg_now=Leg_now-Move_rate(Rate,Leg(I+1))
13220        IF Leg_now<0 THEN Time=Time+Leg_frac/Move_rate(Rate,L
eg(I+1))
13230      EXIT IF Leg_now<=0
13240        Time=Time+1
13250        Vis_left=Vis_left-1
13260        March=March-1
13270        IF March=0 THEN
13280          Time=Time+Rest_time
13290          March=Mar_time
13300        END IF
13310        IF Vis_left<=0 THEN
13320          Vis_left=12
13330          IF Day_nite=0 THEN
13340            Day_nite=1
13350          ELSE
13360            Day_nite=0
13370          END IF
13380        END IF
13390      END LOOP
13400    NEXT I
13410 Walk_over: !   ADD IN COLUMN CLOSURE TIME
13420    Time=Time+(Persons/360)
13430 !PRINT RESULTS OF MOVEMENT
13440    Prt=1
13450    GOSUB Prt_dis_out
13460    INPUT "DO YOU WANT HARD COPY OUTPUT?  (Y OR N)",Q$
```

Table 9-3. Movement code (continued).

```
13470   IF Q$<>"Y" AND Q$<>"N" THEN 13460
13480   IF Q$="Y" THEN
13490     Prt=702
13500     GOSUB Prt_dis_out
13510   END IF
13520   PRINTER IS 1
13530   INPUT "MORE DISMOUNTED MOVEMENT TO CALCULATE?  (Y OR N)",
Q$
13540   IF Q$<>"Y" AND Q$<>"N" THEN 13530
13550   IF Q$="Y" THEN Start_dis_mov
13560   RETURN
13570 !
13580 ! ************************************************************
***************
13590 !
13600 Zero_gnd_data:  ! THIS SBR ZEROES VARIABLES FOR THE GND/DSM
TD MODULES
13610   FOR I=1 TO 10
13620     Leg(I)=0
13630   NEXT I
13640   Pace=1
13650   Force=1
13660   Mar_pace=0
13670   Mar_time=8
13680   Rest_time=0
13690   St_time=0
13700   Persons=0
13710   Time=0
13720   Tkrs=1
13730   Day_nite=0
13740   Close_time=0
13750   Describe$="                                            "
13760   Refuel_dist=200
13770   RETURN
13780 !
13790 ! ************************************************************
***************
13800 !
13810 Err_check_1:!   THIS SBR CHECKS FOR ERRORS IN DISMOUNTED/GR
OUND INPUT
13820   Err=0
13830   IF Force<>1 AND Force<>2 THEN
13840     PRINT "FORCE INPUT ERROR"
13850     Err=1
13860   END IF
13870   IF Mar_pace<>1 AND Mar_pace<>2 THEN
13880     Err=1
13890     PRINT "MARCH CYCLE INPUT ERROR"
13900   END IF
13910   IF St_time>24 OR St_time<0 THEN
13920     Err=1
13930     PRINT "START TIME INPUT ERROR"
```

Table 9-3. Movement code (continued).

```
13940   END IF
13950   FOR I=2 TO 10 STEP 2
13960     IF Leg(I)<0 OR Leg(I)>5 THEN
13970       Err=1
13980       PRINT "TERRAIN PROFILE INPUT ERROR"
13990     END IF
14000   NEXT I
14010   RETURN
14020 !
14030 ! ******************************************************************
****************
14040 !
14050 Prt_dis_out:!    THIS SBR PRINTS OUT DISMOUNTED MOVEMENT RE
SULTS
14060 !
14070   PRINTER IS Prt
14080   IF Prt=1 THEN PRINT USING "@"
14090   PRINT "  "
14100   PRINT " "
14110   PRINT "  "
14120   IF Move_type=1 THEN
14130     PRINT TAB(10),"DISMOUNTED MOVEMENT RESULTS"
14140   ELSE
14150     PRINT TAB(10),"GROUND MOVEMENT RESULTS"
14160   END IF
14170   PRINT "  "
14180   PRINT USING "40A";Describe$
14190   PRINT "  "
14200   IF Force=1 THEN
14210     P1$="BLUE"
14220   ELSE
14230     P1$="RED "
14240   END IF
14250   IF Mar_pace=1 THEN
14260     P2$="NORMAL"
14270   ELSE
14280     P2$="FORCED"
14290   END IF
14300   PRINT USING "7A,4A,13X,6A,6A";"FORCE: ",P1$,"PACE: ",P2$
14310   Up_time=St_time*100
14320   PRINT USING "13A,2D,3A,2D,4X,12A,4Z,4A";"MARCH CYCLE: ",M
ar_time," - ",Rest_time,"START TIME: ",Up_time," HRS"
14330   PRINT USING "15A,4D,5X,13A,3D";"COLUMN LENGTH: ",Persons,
"NO. TANKERS: ",Tkrs
14340   PRINT USING "9A, 5(3D,1X,1D,3X)";"PROFILE: ",Leg(*)
14350   PRINT "  "
14360   PRINT "  "
14370   PRINT USING "18A,3D.2D,1X,3A";"TOTAL MARCH TIME: ",Time,"
HRS"
14380 !   CALCULATE CLOSURE TIME
14390   I_time=St_time+INT(Time)
14400   Min_time=(Time-INT(Time))*60
```

Table 9-3. Movement code (continued).

```
14410   Min_time=INT(Min_time)
14420   IF I_time>23 THEN
14430     I_time=I_time MOD 24
14440     Close_time=(I_time*100)+Min_time
14450   ELSE
14460     Close_time=(I_time*100)+Min_time
14470   END IF
14480   PRINT "  "
14490   PRINT USING "14A,4Z,1X,3A";"CLOSURE TIME: ",Close_time,"H
RS"
14500   RETURN
14510 !
14520 ! ************************************************************
***************
14530 !
14540 Ground:! SBR GROUND CALCULATES GROUND MOVEMENT TIMES
14550 !
14560   RESTORE 14580
14570   READ Move_rate(*)
14580   DATA      ! CONTAINS SAME DATA FORMAT FOR THESE MOUNTED MO
VEMENT RATES
14590   DATA      ! AS THE DISMOUNTED MOVEMENT RATES PREVIOUSLY DE
SCRIBED.
14600 Start_gnd_mov:PRINT USING "@"
14610   PRINT TABXY(26,17),"GROUND MOVEMENT MODULE"
14620   Move_type=2
14630   GOSUB Start_up
14640 !
14650 !     BEGIN MOVEMENT TIME CALCULATION
14660   FOR I=1 TO 9 STEP 2
14670     IF Leg(I)=0 OR Leg(I+1)=0 THEN Gnd_mov_over
14680 !
14690 !     DETERMINE MOVEMENT TIME
14700     Leg_now=Leg(I)
14710     LOOP
14720       Rate=Side+Pace+Day_nite*2
14730       Leg_frac=Leg_now
14740       Leg_now=Leg_now-Move_rate(Rate,Leg(I+1))
14750       IF Leg_now<0 THEN
14760         Time=Time+Leg_frac/Move_rate(Rate,Leg(I+1))
14770         Refuel_dist=Refuel_dist-Move_rate(Rate,Leg(I+1))
14780       END IF
14790     EXIT IF Leg_now<=0
14800       Refuel_dist=Refuel_dist-Move_rate(Rate,Leg(I+1))
14810       Time=Time+1
14820       Vis_left=Vis_left-1
14830       March=March-1
14840       IF March=0 THEN
14850         March=Mar_time
14860         Refuel_time=0
14870         IF Refuel_dist<=30 THEN
14880           Refuel_dist=200
```

Table 9-3. Movement code (continued).

```
14890             Refuel_time=Persons/Tkrs/4*8
14900           END IF
14910           IF Refuel_time>Rest_time THEN
14920             Time=Time+Refuel_time
14930           ELSE
14940             Time=Time+Rest_time
14950           END IF
14960         END IF
14970         IF Refuel_dist<=0 AND March>0 THEN
14980           Refuel_dist=200
14990           Refuel_time=Persons/Tkrs/4*8
15000           Time=Time+Refuel_time
15010           IF Refuel_time>=Rest_time THEN March=Mar_time
15020         END IF
15030         IF Vis_left<=0 THEN
15040           Vis_left=12
15050           IF Day_nite=0 THEN
15060             Day_nite=1
15070           ELSE
15080             Day_nite=0
15090           END IF
15100         END IF
15110       END LOOP
15120     NEXT I                              !  GOTO NEXT LEG
15130 Gnd_mov_over:!    ADD IN CLOSURE TIME
15140     Time=Time+Persons/180
15150 !
15160 !      PRINT RESULTS OF MOVEMENT
15170     Prt=1
15180     GOSUB Prt_dis_out
15190     INPUT "DO YOU WANT HARDCOPY OUTPUT?    (Y OR N)",Q$
15200     IF Q$<>"Y" AND Q$<>"N" THEN 15190
15210     IF Q$="Y" THEN
15220       Prt=702
15230       GOSUB Prt_dis_out
15240     END IF
15250     PRINTER IS 1
15260     INPUT "MORE GROUND MOVEMENT TO CALCULATE?  (Y OR N)",Q$
15270     IF Q$<>"Y" AND Q$<>"N" THEN 15260
15280     IF Q$="Y" THEN Start_gnd_mov
15290 Backuptop:PRINT USING "@"
15300 SUBEND
15310 SUB Ck_var(Var_name$,T$,Variable,Min_value,Max_value)
15320     SELECT T$
15330     CASE "THRU"
15340       WHILE Variable<Min_value OR Variable>Max_value
15350         GOSUB Print_error
15360       END WHILE
15370     CASE "OR"
15380       GOTO Case_to
15390     CASE "TO"
15400 Case_to:FOR M=Min_value TO Max_value
```

Table 9-3. Movement code (concluded).

```
15410         IF Variable=M THEN GOTO End_select
15420       NEXT M
15430       GOSUB Print_error
15440       GOTO Case_to
15450 End_select:!
15460   END SELECT
15470   GOTO Rtrn
15480 Print_error:    !
15490   PRINT
15500   PRINT "** ERROR: ";Variable;" IS INVALID FOR ";Var_name$
15510   PRINT "INPUT: ";Min_value;" ";T$;" ";Max_value;" ONLY"
15520   INPUT Variable
15530   RETURN
15540 Rtrn:!
15550 SUBEND
```

Table 9-3a. ADEA movement code.

```
10     ! PROGRAM MOVEPLAN -- a unit movement planning aid
20     Prtr=702  !Prtr is the printer
30     PRINTER IS 1
40     OPTION BASE 1
50     DIM Spd(22),Klm(22),Dut(21,100),Dun(21,100),Lc(100),Hlt(21)
60     DIM Back(100)  ! used in control move
70     DIM Bran(10),Vehs_in_mu(100),Mus_in_serial(10),Avg_spd(20,100)
80     INTEGER Pf,Id,A,An,I,J
90     ! title routine
100    Home  ! clear the screen
110    PRINT TABXY(26,6);"UNIT PLANNING AID FOR MOVEMENT"
120    PRINT TAB(24);"COMMAND CONTROL ANALYSIS DIVISION"
130    PRINT TAB(37);"CAORA"
140    PRINT TAB(27);"FT LEAVENWORTH, KANSAS 66027"
150    PRINT TAB(33);"AUTOVON 552-3595"
160    WAIT 3
161    PRINT
162    PRINT
164    PRINT "THIS PROGRAM HAS BEEN SUBSTITUED FOR THE ORIGINAL P9"
165    PRINT "MOVEMENT CALCULATOR BY ROB BELFLOWER, BDM"
166    PRINT
167    PRINT "THIS PROGRAM REQUIRES A DATA FILE ON THE DEFAULT MASS"
168    PRINT "STORAGE DEVICE.  IF THAT IS NOT YOUR HARD DISK, INSERT"
169    PRINT "A FORMATTED MICROFLOPPY DISK INTO YOUR DEFAULT DRIVE"
170    PRINT
171    PRINT "PRESS ENTER TO CONTINUE"
172    INPUT A$
173 Menu:Home
180    Flag=0   ! flag for creating files
190    Flag1=0  ! flag for changing file names
200    Flag2=0  ! flag for changing files
210    PRINT
220    PRINT TAB(1);"THE FOLLOWING CHOICES CAN BE MADE FROM THE MAIN MENU:"
230    PRINT
240    PRINT TAB(1);"1.  RUN PROBLEM "
250    PRINT TAB(1);"2.  GENERAL PROGRAM INFORMATION"
260    PRINT TAB(1);"3.  QUIT"
270    PRINT TAB(1);"4.  CREATE A DATA FILE"
280    PRINT TAB(1);"5.  CHANGE A DATA FILE"
290    PRINT TAB(1);"6.  GET INPUT SHEET"
300    PRINT TAB(1);"7.  DELETE A DATA FILE"
310    PRINT TAB(1);"8.  LISTING OF FILES"
320    PRINT
330    DISP "( ENTER 1,2,3,4,5,6,7,OR 8)"
340    ENTER 2;A
350    IF A<1 OR A>8 THEN GOTO 330
360    ON A GOTO Run_problem,Help,Finish,Run_problem,Change_file,Sheet,Delete_+
e,File_list
370 Help:CALL Information
380    GOTO Menu
390 Sheet:CALL Input_sheet
```

Table 9-3a. ADEA movement code (continued).

```
400    GOTO Menu
410 Finish:Home
420    PRINT TABXY(1,15);"THE END"
421    LOAD "DIME:HP913X,701"
430    STOP
440 Run_problem:  ! get inputs and calculate results
450    Home
460    IF A=1 THEN  ! if menu choice is #1
470      DISP "IS DATA ON A DISK(1) OR IS DATA KEYBOARD INPUT(2) (ENTER 1 OR 2)
480      ENTER 2;An
490      IF An<1 OR An>2 THEN GOTO 470
500      IF An=1 THEN
510        GOSUB Read_file  ! read data from disk
520        GOTO 790
530      END IF
540      Home
550    END IF
560    DISP "NEED A DATA INPUT SHEET ? ENTER Y TO GET INPUT SHEET OTHERWISE PRE
 ENTER"
570    GOSUB Answer
580    IF Yes THEN CALL Input_sheet
590    Home
600    GOSUB Data_input
610    IF A=4 THEN  ! if creating a data file
620      GOTO Create_file
630      ASSIGN @Path TO File$
640      OUTPUT @Path,1;Spd(*),Klm(*),Hlt(*),Bran(*),Vehs_in_mu(*),Lc(*),Pace,V
_int,Veh_length,Unint,Nunit,Slint,Nseg,Lenng,Colng,Nserl,Mus_in_serial(*),Vehs
650      ASSIGN @Path TO *
660      GOTO Menu
670    END IF
680    IF An=2 THEN  ! if keyboard input
690      DISP "WANT TO SAVE INPUT DATA TO A FILE ? ENTER Y TO SAVE OTHERWISE PR
S ENTER"
700      GOSUB Answer
710      IF Yes THEN
720        Flag=1
730        GOTO Create_file
740        ASSIGN @Path TO File$
750        OUTPUT @Path,1;Spd(*),Klm(*),Hlt(*),Bran(*),Vehs_in_mu(*),Lc(*),Pace
eh_int,Veh_length,Unint,Nunit,Slint,Nseg,Lenng,Colng,Nserl,Mus_in_serial(*),Ve
760        ASSIGN @Path TO *
770      END IF
780    END IF
790    Home
800    PRINT "RESULTS FROM WHICH MARCH DISCIPLINE?"
810    PRINT "1.  HASTY WITH SPEED CHANGES"
820    PRINT "2.  HASTY WITH ROLLBACK"
830    PRINT "3.  CONTROLLED MOVE"
840    PRINT
850    DISP " ENTER 1 OR 2 OR 3 OR 4(TO RETURN TO MAIN MENU)"
860    ENTER 2;Pf
```

Table 9-3a. ADEA movement code (continued).

```
870   IF Pf<1 OR Pf>4 THEN GOTO 850
880   SELECT Pf
890   CASE <=2  ! if choice is 1 or 2
900     CALL Hasty(Dun(*),Dut(*),Lc(*),Klm(*),Nseg,Spd(*),Pace,Unint,Slint,Bra
*),Nunit,Pf)
910   CASE =3   ! if choice is 3
920     CALL Control_move(Unint,Slint,Spd(*),Klm(*),Colng,Dun(*),Dut(*),Pace,N
g,Nserl,Bran(*),Lc(*),Back(*),Nunit)
930   CASE =4   ! if choice is 4
940     GOTO Menu
950   END SELECT
960   GOSUB Choose_time
970   GOSUB Output
980   Home
990   GOTO Menu
1000!**************************** SUBROUTINES *********************************
1010!
1020!***** this routine prints the results *****
1030 Output:!
1040  PRINTER IS Prtr
1050  IF Pf=1 THEN PRINT "HASTYMOVE "
1060  IF Pf=2 THEN PRINT "HASTYMOVE WITH TIMES ROLLBACKED"
1070  IF Pf=3 THEN PRINT "CONTROL MOVE"
1080  PRINT
1090  PRINT
1100  FOR I=1 TO Nunit
1110    FOR J=1 TO Nseg
1120      Tim=Dun(J+1,I)-Dun(J,I)
1130      Avg_spd(J,I)=(60*Klm(J))/Tim  ' calculate avg speed
1140    NEXT J
1150  NEXT I
1160  Ttime=Dut(Nseg+1,1)-Dun(1,1)
1170  GOSUB Resting
1180  PRINT TAB(10);"ROAD MOVEMENT TABLE"
1190  PRINT
1200  PRINT "PACE(KMPH)=";FNRound(Pace);TAB(39);"VEHICLE INTERVAL(METERS)=";Ve
int
1210  PRINT "AVG SPEED FOR 1ST UNIT(KMPH)=";FNRound((Lenng*60)/Ttime):TAB(39):
ARCH UNIT INTERVAL(KILOMETERS)=";Unint
1220 Img1:IMAGE "LENGTH OF COLUMN(KM)= ",DDDD.DD,#  ' print format
1230  PRINT USING Img1;Colng
1240  PRINT TAB(39);"SERIAL INTERVAL(KILOMETERS)=";Slint
1250  PRINT "AVG VEHICLE LENGTH(METERS)=";FNRound(Veh_length)
1260 Img2:IMAGE "VEHICLE DENSITY(VPKM)=",DDDD.D,#  ' print format
1270  PRINT USING Img2;Vehs/Colng
1280  PRINT
1290  PRINT
1300  PRINT TAB(8);"MARCH";TAB(14);"NUMBER"
1310  PRINT TAB(1);"SERIAL";TAB(8);"UNIT";TAB(14);"    OF";TAB(23);"CHECK";TAB
);"DUE IN";TAB(38);"RELEASE"
1320  PRINT TAB(1);"  NO";TAB(8);" NO";TAB(14);"VEHICLES";TAB(23);"POINT";TAB
);"TIME";TAB(37);" TIME      REMARKS";TAB(71);"CAST"
```

Table 9-3a. ADEA movement code (continued).

```
1330  PRINT TAB(28);"(HHMM DD)";TAB(37);"(HHMM DD)";TAB(67);"(KMPH)  (MPH)"
1340  PRINT
1350  Id=1
1360  FOR I=1 TO Nunit
1370    IF I=Bran(Id) THEN
1380      PRINT TAB(1);"  ";Id;   ! print serial number
1390      IF Id<Nserl THEN Id=Id+1
1400    END IF
1410    PRINT TAB(8);I;TAB(14);Vehs_in_mu(I);   ! print march unit # and number
1420                                            ! of vehicles in that unit
1430    FOR J=1 TO Nseg+1
1440      PRINT TAB(23);
1450      IF J=1 THEN
1460        A$="SP"
1470      ELSE
1480        IF J=Nseg+1 THEN
1490          A$="RP"
1500        ELSE
1510          A$=VAL$(J-1)
1520        END IF
1530      END IF
1540      Hrsmindays(Dun(J,I),Hours_in,Mins_in,Days_in)   ! convert to military
ime
1550      Hrsmindays(Dut(J,I),Hours_out,Mins_out,Days_out)
1560      PRINT USING Img3;A$,Hours_in,Mins_in,Days_in,Hours_out,Mins_out,Days
ut  ! print CP,due in and due out time
1570 Img3:IMAGE AA,4X,ZZ,ZZ,"+",ZZ,2X,ZZ,ZZ,"+",ZZ,X,#   ! print format
1580      IF Hlt(J)>0 THEN
1590        PRINT Hlt(J);"-MINUTE REST";TAB(68);
1600      ELSE
1610        PRINT TAB(68);
1620      END IF
1630      IF J<Nseg+1 THEN
1640        PRINT USING Img4;Avg_spd(J,I),Avg_spd(J,I)*.62   ! print speed in k
h and mph
1650 Img4:   IMAGE DD.D,2X,DD.D   ! print format
1660      ELSE
1670        PRINT
1680      END IF
1690    NEXT J
1700    PRINT
1710  NEXT I
1720  Ff
1730  PRINT TAB(7);"TIME ANALYSIS--FULL COLUMN"
1740  PRINT
1750  PRINT
1760  PRINT "START TIME = ";
1770  Hrsmindays(Dun(1,1),Hrs,Mins,Days)
1780  PRINT USING Img5;Hrs,Mins,Days
1790  PRINT "COMPLETION TIME = ";
1800  Hrsmindays(Dut(Nseg+1,Nunit),Hrs,Mins,Days)
1810  PRINT USING Img5;Hrs,Mins,Days
```

Table 9-3a. ADEA movement code (continued).

```
1820 Img5:IMAGE ZZ,ZZ,"+",ZZ   ! print format
1830 Img6:IMAGE DDDD,":",ZZ     ! print format
1840  PRINT "MARCH TIME (INCLUDING HALTS) = ";
1850  Hrsmindays(Dut(Nseg+1,Nunit)-Dun(1,1),Hrs,Mins,Days)
1860  PRINT USING Img6;Hrs+Days*24,Mins
1870  PRINT
1880  PRINT
1890  PRINT TAB(8);"TIME";TAB(16);" TIME";TAB(24);" TIME";TAB(32);" TIME";TAB(
);" PASSTIME"
1900  PRINT TAB(1);"SERIAL";TAB(8);"LEFT";TAB(16);"CLEARED";TAB(24);"ARRIVED";
B(32);"CLEARED";TAB(39);"     AT"
1910  PRINT TAB(1);"  NO";TAB(8);" SP";TAB(16);"  SP";TAB(24);"  RP";TAB(32);'
RP";TAB(39);"    RP"
1920  PRINT
1930  PRINT
1940  FOR J=1 TO Nserl
1950    PRINT TAB(1);"  ";J;
1960    Id=Bran(J)
1970    IF J<Nserl THEN
1980      Mark=Bran(J+1)-1
1990    ELSE
2000      Mark=Nunit
2010    END IF
2020    Hrsmindays(Dun(1,Id),Hours_in,Mins_in,Days_in) ! time arrive SP
2030    Hrsmindays(Dut(1,Mark),Hours_out,Mins_out,Days_out) ! time cleared SP
2040    Hrsmindays(Dun(Nseg+1,Id),Hrs_inrp,Mins_inrp,Days_inrp) ! arrive RP
2050    Hrsmindays(Dut(Nseg+1,Mark),Hrs_outrp,Mins_outrp,Days_outrp) ! clear F
2060    Hrsmindays(Dut(Nseg+1,Mark)-Dun(Nseg+1,Id),Hrs,Mins,Days)'passtime at
2070    PRINT USING Img7;Hours_in,Mins_in,Days_in,Hours_out,Mins_out,Days_out,
s_inrp,Mins_inrp,Days_inrp,Hrs_outrp,Mins_outrp,Days_outrp,Hrs+Days*24,Mins
2080 Img7:IMAGE ZZ,ZZ,"+",ZZ,2X,ZZ,ZZ,"+",ZZ,2X,ZZ,ZZ,"+",ZZ,2X,ZZ,ZZ,"+",ZZ,.
D,":",ZZ   ! print format
2090  NEXT J
2100  Ff
2110  PRINT TAB(17);"ROUTE"
2120  PRINT TAB(14);"DESCRIPTION"
2130  PRINT
2140  PRINT
2150  PRINT "ROAD";TAB(20);"MAX RATE"
2160  PRINT "SEGMENT";TAB(9);"DISTANCE";TAB(20);"OF TRAVEL"
2170  PRINT TAB(9);"(KM)";TAB(14);"(MI)";TAB(21);"(KMPH)"
2180  PRINT
2190  FOR I=1 TO Nseg
2200    PRINT TAB(4);I;TAB(9);
2210    IF Klm(I)<1 THEN PRINT USING Img8;Klm(I),Klm(I)*.62
2220    IF Klm(I)>1 THEN PRINT FNRound(Klm(I));TAB(14);FNRound(Klm(I)*.62);
2230    PRINT TAB(22);FNRound(Spd(I))
2240  NEXT I
2250 Img8:IMAGE .DD,2X,.DD,#    !print format
2260  Ff
2270  Ff
2280  PRINTER IS 1
```

Table 9-3a. ADEA movement code (continued).

```
2290  RETURN
2300 !***** this routine adds on halt times *****
2310 Resting:!
2320  FOR I=1 TO Nseg+1
2330    IF Hlt(I)>0 THEN
2340      FOR J=I TO Nseg+1
2350        FOR K=1 TO Nunit
2360          IF J<>I THEN Dun(J,K)=Dun(J,K)+Hlt(I)
2370          Dut(J,K)=Dut(J,K)+Hlt(I)
2380        NEXT K
2390      NEXT J
2400    END IF
2410  NEXT I
2420  RETURN
2430 !***** this routine allows user to define start or arrival time *****
2440 Choose_time:!
2450  MAT Dun= (60)*Dun   ! change from hours to minutes
2460  MAT Dut= (60)*Dut
2470  Halt_total=0
2480  FOR I=1 TO Nseg+1
2490    Halt_total=Halt_total+Hlt(I)   ! total time of all halts
2500  NEXT I
2510  Totaltime=Dut(Nseg+1,Nunit)+Halt_total
2520  Hours=INT(Totaltime/60)
2530  Mins=FNRound(Totaltime MOD 60)
2540  Home
2550  PRINT TABXY(1,12);"THE MOVE TAKES";
2560  PRINT " ";Hours;" ";"HOURS AND ";Mins;" MINUTES"
2570  PRINT
2580  Id=99
2590  PRINT TABXY(1,19);"[1]DO YOU WANT TO COMPUTE A START TIME BASED ON AN AR
VAL DEFINED BY YOU ?"
2600  PRINT TAB(1);"[2]DO YOU WANT TO COMPUTE AN ARRIVAL TIME BASED ON A START
IME DEFINED BY YOU ?"
2610  INPUT "ENTER   1  OR  2  OR  0(TO RETURN TO MAIN MENU)",Id
2620  IF Id<0 OR Id>2 THEN GOTO 2610
2630  SELECT Id
2640  CASE =0
2650    GOTO Menu
2660  CASE =1
2670    Time_input("DUE TIME","TO THE DUE DATE",Alfa)
2680    Home
2690    PRINT TABXY(1,23);"ENTER THE CURRENT TIME IN MILITARY CLOCK"
2700    DISP "HOUR(S)   (0-24) : "
2710    ENTER 2;A
2720    IF A<0 OR A>24 THEN GOTO 2700
2730    DISP "MINUTE(S) (0-59) : "
2740    ENTER 2;B
2750    IF B<0 OR B>59 THEN GOTO 2730
2760    IF A=0 AND B=0 THEN GOTO 2700
2770    IF A=24 AND B<>0 THEN GOTO 2700
2780    Now=B+(60*A)
```

Table 9-3a. ADEA movement code (continued).

```
2790     IF Alfa>=Totaltime+Now THEN
2800       Alfa=Alfa-Totaltime
2810       MAT Dun= Dun+(Alfa)
2820       MAT Dut= Dut+(Alfa)
2830     ELSE
2840       Home
2850       PRINT TABXY(1,12);"THERE IS NOT ENOUGH TIME TO EXECUTE THE MOVE"
2860       PRINT
2870       PRINT
2880       PRINT "ENTER C TO CHANGE EITHER THE DUE TIME OR THE CURRENT TIME OR'
2890       PRINT "ENTER R TO RETURN TO MAIN MENU "
2900       ENTER 2;An$
2910       IF An$="R" THEN GOTO Menu
2920       IF An$="C" THEN GOTO 2670
2930       Home
2940       GOTO 2880
2950     END IF
2960     Id=0
2970   CASE =2
2980     Time_input("STARTING TIME","FROM TODAY
TO THE DEPARTURE DATE",Alfa)
2990     MAT Dun= Dun+(Alfa)
3000     MAT Dut= Dut+(Alfa)
3010   END SELECT
3020   RETURN
3030 !*************************** file routines ******************************>
3040 !
3050 !***** this routine prints a listing of files *****
3060 File_list:Home
3070   ON ERROR GOTO Err3
3080   CAT
3090   DISP "PRESS ENTER TO RETURN TO MENU"
3100   ENTER 2;C$
3110   OFF ERROR
3120   GOTO Menu
3130 !***** this routine deletes files *****
3140 Delete_file:Home
3150   DISP "ENTER FILE NAME TO BE DELETED"
3160   ENTER 2;File$
3170   IF File$[1,8]<>"MOVEPLAN" THEN   ! don't delete main program file
3180     ON ERROR GOTO Err3
3190     PURGE File$
3200   ELSE
3210     Home
3220     PRINT TABXY(1,15);"ERROR--YOU TRIED TO DELETE MAIN PROGRAM"
3230     WAIT 3
3240     DISP "ENTER Y TO TRY AGAIN OTHERWISE PRESS RETURN"
3250     GOSUB Answer
3260     IF Yes THEN GOTO Delete_file
3270     GOTO Menu
3280   END IF
3290   OFF ERROR
```

Table 9-3a. ADEA movement code (continued).

```
3300  GOTO Menu
3310 !***** this routine allows changes to made to data files *****
3320 Change_file:Home
3330  Flag2=1
3340  GOSUB Open_file
3350  ENTER @Path,1;Spd(*),Klm(*),Hlt(*),Bran(*),Vehs_in_mu(*),Lc(*),Pace,Veh_
t,Veh_length,Unint,Nunit,Slint,Nseg,Lenng,Colng,Nserl,Mus_in_serial(*),Vehs
3360  ASSIGN @Path TO *
3370  GOSUB Data_input
3380  Home
3390  WAIT 2
3400  DISP "TO SAVE THIS DATA ON A DIFFERENT FILE THAN '";File$;"' ENTER Y OTH
WISE PRESS ENTER"
3410  GOSUB Answer
3420  IF Yes THEN
3430    Flag1=1
3440    GOSUB Open_file
3450    ON ERROR GOTO Err2
3460    CREATE BDAT File$,1,4500
3470  END IF
3480  ASSIGN @Path TO File$
3490  OUTPUT @Path,1;Spd(*),Klm(*),Hlt(*),Bran(*),Vehs_in_mu(*),Lc(*),Pace,Veh
nt,Veh_length,Unint,Nunit,Slint,Nseg,Lenng,Colng,Nserl,Mus_in_serial(*),Vehs
3500  ASSIGN @Path TO *
3510  OFF ERROR
3520  GOTO Menu
3530 !***** this routine opens files *****
3540 Open_file:ON ERROR GOTO Err1
3550  Home
3560  DISP "ENTER FILE NAME (10 ALPHANUMERIC CHARACTERS IS MAX)"
3570  ENTER 2;File$
3580  IF LEN(File$)>10 THEN GOTO 3560
3590  ASSIGN @Path TO File$
3600  OFF ERROR
3610  RETURN
3620 !***** this routine creates data files *****
3630 Create_file:Home
3640  GOSUB Open_file
3650  ON ERROR GOTO Err2
3660  CREATE BDAT File$,1,4500
3670  OFF ERROR
3680  IF A=4 THEN
3690    GOTO 630
3700  ELSE
3710    GOTO 740
3720  END IF
3730 !***** this routine reads data from files *****
3740 Read_file:!
3750  GOSUB Open_file
3760  OFF ERROR
3770  ENTER @Path,1;Spd(*),Klm(*),Hlt(*),Bran(*),Vehs_in_mu(*),Lc(*),Pace,Veh_
t,Veh_length,Unint,Nunit,Slint,Nseg,Lenng,Colng,Nserl,Mus_in_serial(*),Vehs
```

Table 9-3a. ADEA movement code (continued).

```
3780   ASSIGN @Path TO *
3790   RETURN
3800 '************************** error routines ******************************
3810 Err1:OFF ERROR
3820   IF ERRN=80 THEN
3830     Home
3840     PRINT TABXY(1,15);"ERROR--DISK IS NOT IN CORRECT DRIVE"
3850     WAIT 3
3860     GOTO 3540
3870   END IF
3880   IF ERRN=53 THEN
3890     Home
3900     PRINT TABXY(1,15);"ERROR--INVALID FILE NAME"
3910     WAIT 3
3920     GOTO 3540
3930   END IF
3940   IF ERRN=56 THEN
3950     IF NOT Flag AND A<>4 AND NOT Flag1 THEN
3960       Home
3970       PRINT TABXY(1,15);"ERROR--FILE DOESN'T EXIST"
3980       WAIT 3
3990       DISP "ENTER Y TO TRY AGAIN OTHERWISE PRESS ENTER"
4000       GOSUB Answer
4010       IF Yes THEN
4020         GOTO 3540
4030       ELSE
4040         GOTO Menu
4050       END IF
4060     END IF
4070   END IF
4080   GOTO 3610
4090 Err2:OFF ERROR
4100   Home
4110   PRINT TABXY(1,15);"ERROR--FILE ALREADY EXISTS"
4120   WAIT 3
4130   IF Flag1 THEN
4140     GOTO 3440
4150   ELSE
4160     GOTO 3630
4170   END IF
4180 Err3:OFF ERROR
4190   Home
4200   IF ERRN=80 THEN
4210     PRINT TABXY(1,15);"ERROR--DISK IS NOT IN DRIVE"
4220     WAIT 2
4230     IF A=7 THEN GOTO Delete_file
4240     GOTO Menu
4250   END IF
4260   PRINT TABXY(1,15);"ERROR--FILE DOESN'T EXIST"
4270   WAIT 3
4280   DISP "ENTER Y TO TRY AGAIN OTHERWISE PRESS RETURN"
4290   GOSUB Answer
```

Table 9-3a. ADEA movement code (continued).

```
4300  IF Yes THEN GOTO Delete_file
4310  GOTO Menu
4320 !*************************** MORE SUBROUTINES ****************************
4330 !
4340 !***** this routine inputs and changes data *****
4350 Data_input:!
4360  IF Flag2 THEN
4370    DISP "PACE =";Pace;"  ENTER Y TO CHANGE OTHERWISE PRESS ENTER"
4380    GOSUB Answer
4390  END IF
4400  IF Yes OR NOT Flag2 THEN
4410    DISP "ENTER THE PACE (THE MAX SPEED OF THE SLOWEST VEHICLE)"
4420    ENTER 2;Pace
4430    IF Pace<=0 OR Pace>76 THEN GOTO 4410
4440  END IF
4450  IF Flag2 THEN
4460    DISP "VEHICLE INTERVAL =";Veh_int;"  ENTER Y TO CHANGE OTHERWISE PRESS
NTER"
4470    GOSUB Answer
4480  END IF
4490  IF Yes OR NOT Flag2 THEN
4500    DISP "ENTER VEHICLE INTERVAL IN METERS"
4510    ENTER 2;Veh_int
4520    IF Veh_int<=0 OR Veh_int>999 THEN GOTO 4500
4530  END IF
4540  IF Flag2 THEN
4550    DISP "AVG VEHICLE LENGTH =";Veh_length;"  ENTER Y TO CHANGE OTHERWISE F
ESS ENTER"
4560    GOSUB Answer
4570  END IF
4580  IF Yes OR NOT Flag2 THEN
4590    DISP "ENTER THE AVG VEHICLE LENGTH IN METERS"
4600    ENTER 2;Veh_length
4610    IF Veh_length<=0 OR Veh_length>99 THEN GOTO 4590
4620  END IF
4630  IF Flag2 THEN
4640    DISP "MARCH UNIT INTERVAL=";Unint;"  ENTER Y TO CHANGE OTHERWISE PRESS
NTER"
4650    GOSUB Answer
4660  END IF
4670  IF Yes OR NOT Flag2 THEN
4680    DISP "ENTER MARCH UNIT INTERVAL IN KM"
4690    ENTER 2;Unint
4700    IF Unint<=0 OR Unint>9 THEN GOTO 4680
4710  END IF
4720  IF Flag2 THEN
4730    DISP "SERIAL INTERVAL IN KM =";Slint;"  ENTER Y TO CHANGE OTHERWISE PRE
S ENTER"
4740    GOSUB Answer
4750  END IF
4760  IF Yes OR NOT Flag2 THEN
4770    DISP "ENTER SERIAL INTERVAL IN KILOMETERS"
```

Table 9-3a. ADEA movement code (continued).

```
4780    ENTER 2;Slint
4790    IF Slint<=0 OR Slint>25 THEN GOTO 4770
4800  END IF
4810  IF Flag2 THEN
4820    DISP "NUMBER OF ROAD SEGMENTS =";Nseg;"  ENTER Y TO CHANGE OTHERWISE F
SS ENTER"
4830    GOSUB Answer
4840  END IF
4850  IF Yes OR NOT Flag2 THEN
4860    DISP "A MAX OF 20 ROAD SEGMENTS CAN BE USED ";
4870    DISP "FOR A GIVEN ROUTE. ENTER # OF SEGMENTS"
4880    ENTER 2;Nseg
4890    Nseg=INT(Nseg)
4900    IF Nseg<=0 OR Nseg>20 THEN GOTO 820
4910    Home
4920  END IF
4930  FOR I=1 TO Nseg
4940    IF Flag2 THEN
4950      IF Spd(I)=0 THEN GOTO 5000
4960      DISP "FOR SEGMENT";I;" SPEED =";Spd(I);"  ENTER Y TO CHANGE OTHERWIS
PRESS ENTER"
4970      GOSUB Answer
4980    END IF
4990    IF Yes OR NOT Flag2 THEN
5000      DISP "ENTER MAX SPEED IN KMPH FOR ROAD SEGMENT";I
5010      ENTER 2;Spd(I)
5020      IF Spd(I)<=0 OR Spd(I)>100 THEN GOTO 5000
5030    END IF
5040    IF Flag2 THEN
5050      IF Klm(I)=0 THEN GOTO 5100
5060      DISP "FOR SEGMENT";I;" DISTANCE=";Klm(I);"  ENTER Y TO CHANGE OTHERW
E PRESS ENTER"
5070      GOSUB Answer
5080    END IF
5090    IF Yes OR NOT Flag2 THEN
5100      DISP "ENTER DISTANCE IN KM FOR ROAD SEGMENT";I
5110      ENTER 2;Klm(I)
5120      IF Klm(I)<=0 THEN GOTO 5100
5130    END IF
5140  NEXT I
5150  Lenng=0
5160  FOR I=1 TO Nseg
5170    Lenng=Lenng+Klm(I)   ! total length of route
5180  NEXT I
5190  IF Flag2 THEN
5200    DISP "NUMBER OF SERIALS =";Nserl;"  ENTER Y TO CHANGE OTHERWISE PRESS
TER"
5210    GOSUB Answer
5220  END IF
5230  IF Yes OR NOT Flag2 THEN
5240    DISP "ENTER THE NUMBER OF SERIALS ";
5250    DISP "(MAX IS 10)"
```

Table 9-3a. ADEA movement code (continued).

```
5260     ENTER 2;Nserl
5270     Nserl=INT(Nserl)
5280     IF Nserl<=0 OR Nserl>10 THEN GOTO 5240
5290   END IF
5300   Vehs=0
5310   Nunit=0
5320   FOR I=1 TO Nserl
5330     Home
5340     IF Flag2 THEN
5350       IF Mus_in_serial(I)=0 THEN GOTO 5400
5360       DISP "# OF MARCH UNITS IN SERIAL";I;"=";Mus_in_serial(I);" ENTER Y T
CHANGE OTHERWISE PRESS ENTER"
5370       GOSUB Answer
5380     END IF
5390     IF Yes OR NOT Flag2 THEN
5400       DISP "ENTER THE # OF MARCH UNITS IN SERIAL";I;
5410       DISP "   (MAX IS 10)"
5420       ENTER 2;Mus_in_serial(I)
5430       Mus_in_serial(I)=INT(Mus_in_serial(I))
5440       IF Mus_in_serial(I)<=0 OR Mus_in_serial(I)>10 THEN GOTO 5400
5450     END IF
5460     Munit=Mus_in_serial(I)
5470     Nunit=Nunit+Munit   ! total number of march units
5480     Bran(I)=Nunit-Munit+1   ! marks head of a serial
5490     FOR J=Bran(I) TO Bran(I)+Munit-1
5500       Count=J-Bran(I)+1
5510       IF Flag2 THEN
5520         IF Vehs_in_mu(J)=0 THEN GOTO 5570
5530         DISP "# OF VEHICLES IN MU";Count;"=";Vehs_in_mu(J);" ENTER Y TO CH
GE OTHERWISE PRESS ENTER"
5540         GOSUB Answer
5550       END IF
5560       IF Yes OR NOT Flag2 THEN
5570         DISP "ENTER THE # OF VEHICLES IN MU";Count
5580         ENTER 2;Vehs_in_mu(J)
5590         Vehs_in_mu(J)=INT(Vehs_in_mu(J))
5600         IF Vehs_in_mu(J)<=0 OR Vehs_in_mu(J)>999 THEN GOTO 5570
5610       END IF
5620       Mveh=Vehs_in_mu(J)
5630       Lc(J)=(Veh_length*Mveh)/1000
5640       Lc(J)=Lc(J)+((Veh_int*(Mveh-1))/1000)   ! length of march unit
5650       Vehs=Vehs+Mveh   ! total number of vehicles
5660     NEXT J
5670   NEXT I
5680   Klm(Nseg+1)=2999
5690   Spd(Nseg+1)=Spd(Nseg)
5700   FOR I=1 TO Nseg+1
5710     Home
5720     IF I=1 THEN I$="SP"
5730     IF I=Nseg+1 THEN I$="RP"
5740     IF I<>Nseg+1 AND I<>1 THEN I$=VAL$(I-1)
5750     IF Flag2 THEN
```

Table 9-3a. ADEA movement code (continued).

```
5760      DISP "HALT AT CP-";I$;" =";Hlt(I);"  ENTER Y TO CHANGE OTHERWISE PF
 ENTER"
5770      GOSUB Answer
5780    END IF
5790    IF Yes OR NOT Flag2 THEN
5800      DISP "ENTER HALT TIME (IN MINUTES) FOR CP-";I$
5810      ENTER 2;Hlt(I)
5820      IF Hlt(I)<0 OR Hlt(I)>1440 THEN GOTO 5800
5830    END IF
5840  NEXT I
5850  Home
5860  Colng=0
5870  FOR I=1 TO Nunit
5880    Colng=Colng+Lc(I)
5890  NEXT I
5900  Colng=Colng+((Nserl-1)*Slint)
5910  Colng=Colng+((Nunit-Nserl)*Unint)   ! total column length
5920  RETURN
5930 !***** this routine prompts for yes/no anwser *****
5940 Answer:!
5950  ENTER 2;B$
5960  IF B$="Y" THEN
5970    Yes=1
5980  ELSE
5990    Yes=0
6000  END IF
6010  RETURN
6020  END
6030  !****************************** SUB PROGRAMS ***************************
6040  !
6050  !***** this routine clears the screen *****
6060  SUB Home
6070    DISP "                                                            "
6080    PRINT CHR$(12)
6090  SUBEND
6100  !***** this routine sets up the nodes for calculations *****
6110  SUB Calculate_nodes(Node(*),Length_segment(*),Key_node(*),Length_serial
eg)
6120    INTEGER I
6130    Node(0)=0
6140    FOR I=0 TO Nseg
6150      Node(I+1)=Node(I)+Length_segment(I)
6160      Node(Nseg+I+1)=Node(I)+Length_serial
6170    NEXT I
6180    MAT SORT Node TO Key_node
6190    MAT SORT Node
6200  SUBEND
6210  !***** this routine calulates speeds over road segments *****
6220  SUB Control_speeds(Speed(*),Maxnode,Segment_speed(*),Length_serial,Pace
y_node(*),Nseg)
6230    INTEGER I
6240    DIM Speedlist(40)
```

Table 9-3a. ADEA movement code (continued).

```
6250    MAT Speedlist= (1000.1)
6260    Segment_speed(Nseg)=Segment_speed(Nseg-1)
6270    Speedlist(0)=Pace
6280    Speed_del=1
6290    Speed_add=1
6300    FOR I=0 TO Maxnode
6310      IF Key_node(I)>Nseg THEN
6320        Speedlist(Speed_del)=1000.1   ! node is artificial
6330        Speed_del=Speed_del+1
6340      ELSE                            ! node is check point
6350        Speedlist(Speed_add)=Segment_speed(Speed_add-1)
6360        Speed_add=Speed_add+1
6370      END IF
6380      Speed(I)=MIN(Speedlist(*))
6390    NEXT I
6400  SUBEND
6410  !***** this routine calculates distance between nodes *****
6420  SUB Calculate_lengt(Length(*),Node(*),Maxnode)
6430    INTEGER I
6440    FOR I=1 TO Maxnode
6450      Length(I-1)=Node(I)-Node(I-1)
6460    NEXT I
6470  SUBEND
6480  !***** this routine calculates time to travel to node *****
6490  SUB Calculate_times(Maxnode,Speed(*),Length(*),Time(*))
6500    INTEGER I
6510    Time(0)=0
6520    FOR I=1 TO Maxnode
6530      Time(I)=Time(I-1)+Length(I-1)/Speed(I-1)
6540    NEXT I
6550  SUBEND
6560  !***** this function returns the time that the head of the ser- *****
6570  !***** ial crosses the point a distance x from the start point *****
6580  DEF FNCross(X,Time(*),Nodes(*),Speed(*))
6590    N1=1
6600    LOOP
6610    EXIT IF X<Nodes(N1)
6620      N1=N1+1
6630    END LOOP
6640    N1=N1-1
6650    T=Time(N1)+(X-Nodes(N1))/Speed(N1)
6660    RETURN T
6670  FNEND
6680  !***** this routine calculates a controlled move *****
6690  SUB Control_move(Mu_int,Ser_int,Speeds(*),Length_seg(*),Length_serial,Du
*),Dut(*),Pace,Nseg,Nser,Bran(*),Lc(*),Back(*),Nunit)
6700    DIM Nodes(41),Key_nodes(41),Speed(21),Length_segment(21),Length(41),Ti
(41),Speed_art(41)
6710    REDIM Nodes(2*Nseg+1),Key_nodes(2*Nseg+1)
6720    INTEGER I,J
6730    MAT Speed= Speeds
6740    MAT Length_segment= Length_seg
```

Table 9-3a. ADEA movement code (continued).

```
6750    Maxnode=2*Nseg+1
6760    CALL Calculate_nodes(Nodes(*),Length_segment(*),Key_nodes(*),Length_se
al,Nseg)
6770    CALL Control_speeds(Speed_art(*),Maxnode,Speed(*),Length_serial,Pace,K
_nodes(*),Nseg)
6780    CALL Calculate_lengt(Length(*),Nodes(*),Maxnode)
6790    CALL Calculate_times(Maxnode,Speed_art(*),Length(*),Time(*))
6800    FOR I=2 TO Nunit
6810      Back(I)=Back(I-1)+Mu_int+Lc(I-1)
6820    NEXT I
6830    FOR I=2 TO Nser
6840      FOR L=Bran(I) TO Nunit
6850        Back(L)=Back(L)-Mu_int+Ser_int
6860      NEXT L
6870    NEXT I
6880    MAT SORT Key_nodes TO Key_nodes
6890    FOR I=1 TO Nunit
6900      FOR J=1 TO Nseg+1
6910        X1=Nodes(Key_nodes(J-1))+Back(I)
6920        X2=X1+Lc(I)
6930        Dun(J,I)=FNCross(X1,Time(*),Nodes(*),Speed_art(*))
6940        Dut(J,I)=FNCross(X2,Time(*),Nodes(*),Speed_art(*))
6950      NEXT J
6960    NEXT I
6970  SUBEND
6980  '***** this function rounds to nearest integer
6990  DEF FNRound(X)
7000    RETURN INT(X+.5)
7010  FNEND
7020  '***** this routine tab feeds the page in the printer *****
7030  SUB Ff
7040    PRINT CHR$(12)
7050  SUBEND
7060  '***** this routine prints a data input sheet *****
7070  SUB Input_sheet
7080    INTEGER C,D
7090    PRINTER IS 702
7100    PRINT TAB(32);"DATA INPUT SHEET"
7110    PRINT
7120    PRINT "PACE SPEED ______ KMPH  "
7130    PRINT "VEHICLE INTERVAL ______ METERS    AVG VEHICLE LENGTH ______ MET
S"
7140    PRINT "MARCH UNIT INTERVAL ______ KM     SERIAL INTERVAL ______ KM"
7150    PRINT
7160    PRINT "# OF ROAD SEGMENTS IN ROUTE  ___"
7170    PRINT
7180    PRINT "SEGMENT#  MAX RATE(KMPH)  DISTANCE(KM)  SEGMENT#  MAX RATE(KMPH
 DISTANCE(KM)"
7190    FOR D=1 TO 10
7200      PRINT TAB(3);D;TAB(13);"____                ____";TAB(43);D+10;TAB(53);
__                ____"
7210    NEXT D
```

Table 9-3a. ADEA movement code (continued).

```
7220    PRINT
7230    PRINT "# OF SERIALS ____"
7240    PRINT
7250    FOR D=1 TO 5
7260      PRINT "SERIAL NUMBER";D;TAB(40);"SERIAL NUMBER";D+5
7270      PRINT "# OF MARCH UNITS IN SERIAL";D;"____";TAB(40);"# OF MARCH UNIT
IN SERIAL";D+5;"____"
7280      PRINT "MU#    # OF VEHICLES";TAB(40);"MU#    # OF VEHICLES"
7290      FOR C=1 TO 10
7300        PRINT TAB(1);C;TAB(11);"_____";TAB(40);C;TAB(50);"___ _"
7310      NEXT C
7320      PRINT
7330    NEXT D
7340    PRINT
7350    PRINT " CP     HALT TIME(MINS)     CP     HALT TIME(MINS)     CP     HALT T
E(MINS)"
7360    FOR D=1 TO 7
7370      IF D=1 THEN
7380        PRINT TAB(2);"SP";
7390      ELSE
7400        PRINT TAB(1);D-1;
7410      END IF
7420      PRINT TAB(10);"____";TAB(26);D+6;TAB(35);"____";
7430      IF D=7 THEN
7440        PRINT TAB(52);"RP";TAB(60);"____"
7450      ELSE
7460        PRINT TAB(51);D+13;TAB(60);"____"
7470      END IF
7480    NEXT D
7490    PRINT
7500    PRINT "RESULTS FROM WHICH MARCH DISCIPLINE (CIRCLE ONE)"
7510    PRINT
7520    PRINT "1.  HASTY WITH SPEED CHANGES"
7530    PRINT "2.  HASTY WITH ROLLBACK"
7540    PRINT "3.  CONTROLLED MOVE"
7550    PRINT
7560    PRINT
7570    PRINT "DESIRED START TIME ______  OR DESIRED ARRIVAL TIME ______"
7580    PRINT "NUMBER OF DAYS TO DEPARTURE ______"
7590    PRINT CHR$(12)
7600    PRINT CHR$(12)
7610    PRINTER IS 1
7620  SUBEND
7630  !***** this routine converts time to military clock *****
7640  SUB Hrsmindays(Minutes,Hrs,Mins,Days)
7650    M=FNRound(Minutes)
7660    Days=M DIV 1440
7670    Hrs=(M MOD 1440) DIV 60
7680    Mins=FNRound((M MOD 1440) MOD 60)
7690    IF (Hrs=0 AND Mins=0 AND M<>0) THEN
7700      Hrs=24
7710      Days=Days-1
```

Table 9-3a. ADEA movement code (continued).

```
7720    END IF
7730  SUBEND
7740  !***** this routine prompts for a start time or due time *****
7750  SUB Time_input(A1$,A2$,Alfa)
7760    INTEGER A,B
7770    Home
7780    PRINT TABXY(1,23);"ENTER THE ";A1$;" IN MILITARY CLOCK"!A1$ IS STARTIN
TIME OR DUE TIME
7790    DISP "HOUR(S)   (0-24) : "
7800    ENTER 2;A
7810    IF A<0 OR A>24 THEN GOTO 7790
7820    DISP "MINUTE(S) (0-59) : "
7830    ENTER 2;B
7840    IF B<0 OR B>59 THEN GOTO 7820
7850    IF A=0 AND B=0 THEN GOTO 7790
7860    IF A=24 AND B<>0 THEN GOTO 7790
7870    Zorro=B+(60*A)
7880    Home
7890    PRINT TABXY(1,23);"ENTER THE NUMBER OF DAYS ";A2$
7900    PRINT "(e.g. ENTER 2 FOR THE DAY AFTER TOMMOROW)"
7910    ENTER 2;Mark
7920    IF Mark<0 OR Mark>365 THEN GOTO 7890
7930    Alfa=Zorro+(Mark*1440)
7940  SUBEND
7950  !***** this routine provides general information to the user *****
7960  SUB Information
7970    Home
7980    PRINT TABXY(1,6);"STEP ONE-ORGANIZE ROUTE INFORMATION"
7990    PRINT
8000    PRINT TAB(4);"LIST ON THE INPUT SHEET, BY ROAD"
8010    PRINT TAB(4);"SEGMENT, THE LENGTH AND MAXIMUM"
8020    PRINT TAB(4);"TRAVEL RATE FOR EACH SEGMENT."
8030    PRINT TAB(4);"LIST ON THE INPUT SHEET, ANY"
8040    PRINT TAB(4);"HALT TIMES FOR ANY CHECK POINT."
8050    PRINT
8060    PRINT
8070    INPUT "PRESS ENTER TO CONTINUE",An$
8080    Home
8090    PRINT TABXY(1,6);"STEP TWO-ORGANIZE SERIAL INFORMATION"
8100    PRINT
8110    PRINT
8120    PRINT TAB(1);"LIST ON THE INPUT SHEET, BY SERIAL, THE"
8130    PRINT TAB(1);"NUMBER OF MARCH UNITS IN EACH SERIAL AND"
8140    PRINT TAB(1);"THEN NUMBER OF VEHICLES IN EACH MARCH"
8150    PRINT TAB(1);"UNIT.       ENTER IN SERIAL ORDER"
8160    PRINT TAB(1);"IE. (SERIAL1,MU1), (SERIAL1,MU2),...."
8170    PRINT TAB(1);"(SERIALN,MU1),.... (SERIALN,MUQ)."
8180    PRINT
8190    PRINT
8200    INPUT "PRESS ENTER TO CONTINUE",An$
8210    Home
8220    PRINT TABXY(1,6);"STEP THREE-DETERMINE SERIAL CONSTRAINTS"
```

Table 9-3a. ADEA movement code (continued).

```
8230    PRINT
8240    PRINT
8250    PRINT TAB(1):"THE FOLLOWING CONSTANTS ARE REQUIRED:"
8260    PRINT TAB(1):"MAXIMUM SPEED FOR LEAD VEHICLE (PACE)"
8270    PRINT TAB(1):"VEHICLE INTERVAL"
8280    PRINT TAB(1);"AVERAGE VEHICLE LENGTH"
8290    PRINT TAB(1);"MARCH UNIT INTERVAL"
8300    PRINT TAB(1);"SERIAL INTERVAL"
8310    PRINT TAB(1);"MARCH DISCIPLINE"
8320    PRINT TAB(1);"START TIME OR ARRIVAL TIME"
8330    PRINT TAB(1);"NUMBER OF DAYS TO DEPARTURE"
8340    PRINT
8350    PRINT
8360    INPUT "PRESS ENTER TO CONTINUE",An$
8370    PRINT
8380    PRINT
8390    PRINT TAB(25);"MOVEMENT RATE GUIDE(UNOPPOSED)"
8400    PRINT TAB(37);"(KMPH)"
8410    PRINT
8420    PRINT TAB(44);"WEATHER CONDITIONS"
8430    PRINT TAB(40);"(DAY/N-LIGHTS/N-BLACKOUT)"
8440    PRINT TAB(8);"                                              LIGHT          F
/HVY"
8450    PRINT TAB(8);"TERRAIN     UNIT TYPE      GOOD        PRECIPITATION  PREC
ITATION"
8460    PRINT TAB(8);"ROAD        FOOT TRPS    4/3.2/3.2      3.2/2.5/2.5    2.4
.9/1.9"
8470    PRINT TAB(8);"            TRK.GENRL     40/40/16       32/32/12.8       2
24/9.6"
8480    PRINT TAB(8);"            TRCKD VEH     24/24/16     16.8/16.8/11.2     1
12/8"
8490    PRINT TAB(8);"            ARTY.TRCK     40/40/16        32/32/16        2
24/9.6"
8500    PRINT TAB(8);"            ARTY.TRCTR    32/32/16     22.4/22.4/11.2     1
16/8"
8510    PRINT
8520    PRINT TAB(8);"X-COUNTRY   FOOT TRPS   2.4/1.6/1.6     1.9/1.2/1.2     1.4
.9/0.9"
8530    PRINT TAB(8);"            TRK.GENRL      12/12/8      9.6/6.4/6.4      7.2
.8/4.8"
8540    PRINT TAB(8):"            TRCKD VEH      14/8/8      11.2/5.6/5.6
/8/4"
8550    PRINT TAB(8);"            ARTY.TRCK      12/8/8       9.6/6.4/6.4      7.2
.8/4.8"
8560    PRINT TAB(8);"            ARTY.TRCTR     16/8/8      11.2/5.5/5.6
8/4"
8570    DISP "PRESS ENTER TO CONTINUE"
8580    ENTER 2:An$
8590  SUBEND
8600  !***** this routine calculates a hasty move with speed  *****
8610  !***** changes or a hasty move with times rollbacked *****
8620  SUB Hasty(Dun(*),Dut(*),Lc(*),Klm(*),Nseq,Spd(*),Pace,Mu_int,Ser_int,Br
```

Table 9-3a. ADEA movement code (concluded).

```
*),Nunit,INTEGER Pf)
8630    DIM Nodes(41),Key_nodes(41),Speed(21),Length_segment(21),Length(41),Ti
(41),Speed_art(41),Due_diff(20)
8640    REDIM Nodes(2*Nseg+1),Key_nodes(2*Nseg+1)
8650    INTEGER I,J
8660    MAT Speed= Spd
8670    MAT Length_segment= Klm
8680    Maxnode=2*Nseg+1
8690    FOR I=1 TO Nunit
8700      CALL Calculate_nodes(Nodes(*),Length_segment(*),Key_nodes(*),Lc(I),N
g)
8710      Control_speeds(Speed_art(*),Maxnode,Speed(*),Lc(I),Pace,Key_nodes(*)
seg)
8720      Calculate_lengt(Length(*),Nodes(*),Maxnode)
8730      Calculate_times(Maxnode,Speed_art(*),Length(*),Time(*))
8740      MAT SORT Key_nodes TO Key_nodes
8750      FOR J=1 TO Nseg+1
8760        X1=Nodes(Key_nodes(J-1))
8770        X2=X1+Lc(I)
8780        Dun(J,I)=FNCross(X1,Time(*),Nodes(*),Speed_art(*))
8790        Dut(J,I)=FNCross(X2,Time(*),Nodes(*),Speed_art(*))
8800      NEXT J
8810    NEXT I
8820    FOR I=2 TO Nunit
8830      FOR J=1 TO Nseg+1
8840        Gap=Mu_int
8850        FOR K=2 TO SIZE(Bran,1)
8860          IF I=Bran(K) THEN Gap=Ser_int
8870        NEXT K
8880        Gap=Gap/Pace
8890        IF Pf=1 THEN  ! if hasty with speed changes
8900          IF J<Nseg+1 THEN Due_diff(J)=Dun(J+1,I)-Dun(J,I)
8910          Diff=Dut(J,I)-Dun(J,I)
8920          IF J>1 THEN
8930            Dun(J,I)=Dun(J-1,I)+Due_diff(J-1)
8940            IF Dun(J,I)<Dut(J,I-1)+Gap THEN Dun(J,I)=Dut(J,I-1)+Gap
8950          ELSE
8960            Dun(J,I)=Dut(J,I-1)+Gap
8970          END IF
8980          Dut(J,I)=Dun(J,I)+Diff
8990        END IF
9000        IF Pf=2 THEN  ! if hasty with rollback
9010          IF Dun(J,I)<Dut(J,I-1)+Gap THEN
9020            Delay=Dut(J,I-1)+Gap-Dun(J,I)
9030            FOR K=1 TO Nseg+1
9040              Dun(K,I)=Dun(K,I)+Delay
9050              Dut(K,I)=Dut(K,I)+Delay
9060            NEXT K
9070          END IF
9080        END IF
9090      NEXT J
9100    NEXT I
9110  SUBEND
```

CHAPTER 10

UNIT STATUS REPORT

1. PURPOSE.

The purpose of the DIME unit status report (**P8**) is to generate reports summarizing divisional activities for a six-hour period.

2. GENERAL.

A. Unit history. The unit history is a cumulative display of ammunition levels remaining after the six-hour critical incident (CI). These values, along with current mission, combat effectiveness, and ammunition/fuel (AMMO/POL) logistics data, are given for each active unit in the "UNITFILE." (See Figure 10-1.)

B. Killer/victim summary. The killer/victim (K/V) segment of the DIME unit status report (**P8**) generates two reports: a killer/victim scoreboard and attack helicopter results. The K/V scoreboard shows both Blue vs. Red and Red vs. Blue. (See Figure 10-2.)

C. Gamer/staff worksheets. Gamer/staff worksheets are printed for each active unit in the CI. Information from the unit history is used to calculate resupply values which are input from the worksheets in the game initialization program (**P1**). (See Figure 10-3.)

3. DATA FLOW.

Data used in **P8** consist of interactive question/answer information and external data files.

A. Operator input.

(1) It is possible to enter **P8** anytime during a game turn in order to obtain a current K/V scoreboard. By answering "Y" to the question "Killer/victim table listing only?", calculations or updates are made to the "UNITFILE." The questions concerning attrition and game turn identification which follow must then be answered. The K/V summary is the only portion printed before control returns to the DIME menu.

UNIT HISTORY FOR GAME 1 TURN 2

RED UNITS:

UNIT: 66 111THTB TYPE: 2.0 ACTIVE CBT-EFF: .80 MISSION: 2.0

SYSTEMS REMAINING:

1:	17.4	2:	11.5	3:	.8	4:	0.0	5:	4.0	6:	0.0	7:	9.1
8:	.6	9:	8.8	10:	100.9	11:	0.0	12:	.4	13:	0.0	14:	3.8
15:	0.0	16:	0.0	17:	1.9	18:	.1	19:	5.0	20:	43.0	21:	9.0

DETECTION: NOT DET SENS STATUS (Z/G): 4.4 FRA CVG: .50 KM MOVED: 0

AIR DEFENSE: SUPPRESSION: 0.00 CORPS SUPPORT ADA: 71

LOGISTICS:

TYPE	STATUS	PROFILE	CONSUMED	RESUPPLY	ON-TRK	ON-GND	DISPND	ON-VEH	CURRNT	USD/DT
DF	1.00	2	9	0	181	749	0	82	1011	9
IF	1.00	2	10	0	16	67	0	3	86	11
AD	1.00	2	1	0	18	76	0	1	95	2
FU	.26	2	3826	0	0	0	0	3193	3193	12641

EQUIVALENT EMPTY AMMO TRUCKS 0

UNIT: 67 112THTB TYPE: 2.0 ACTIVE CBT-EFF: .80 MISSION: 2.0

SYSTEMS REMAINING:

1:	25.7	2:	11.5	3:	.8	4:	0.0	5:	4.0	6:	0.0	7:	9.1
8:	.6	9:	0.0	10:	100.9	11:	0.0	12:	.4	13:	0.0	14:	3.8
15:	0.0	16:	0.0	17:	1.9	18:	.1	19:	5.0	20:	43.0	21:	9.0

DETECTION: NOT DET SENS STATUS (Z/G): 3.4 FRA CVG: .50 KM MOVED: 0

AIR DEFENSE: SUPPRESSION: 0.00 CORPS SUPPORT ADA: 71

LOGISTICS:

TYPE	STATUS	PROFILE	CONSUMED	RESUPPLY	ON-TRK	ON-GND	DISPND	ON-VEH	CURRNT	USD/DT
DF	1.00	2	9	0	182	800	0	80	1063	10
IF	1.00	2	10	0	15	68	0	3	86	11
AD	1.00	2	1	0	18	77	0	1	95	2
FU	.24	2	4866	0	0	0	0	2885	2885	14721

EQUIVALENT EMPTY AMMO TRUCKS 0

UNIT: 68 113THTB TYPE: 2.0 ACTIVE CBT-EFF: .80 MISSION: 2.0

SYSTEMS REMAINING:

1:	25.7	2:	11.5	3:	.8	4:	0.0	5:	4.0	6:	0.0	7:	9.1
8:	.6	9:	0.0	10:	100.9	11:	0.0	12:	.4	13:	0.0	14:	3.8
15:	0.0	16:	0.0	17:	1.9	18:	.1	19:	5.0	20:	43.0	21:	9.0

Figure 10-1. Unit history output.

RED KILLER--BLUE VICTIM FILE FOR GAME 1 TURN 2

VICTIM	D/F	I/F	PGM	A/H	INF	MIN	CHM	AIR	T/KILL
	<----------			KILLER				---------->	
LAV25	7.5	3.1	0.0	2.1	0.0	.1	0.0	8.6	21.4
FAV-T	10.6	2.1	0.0	1.8	0.0	0.0	0.0	3.9	18.4
HMM-T	4.2	.9	0.0	.8	0.0	.1	0.0	3.5	9.5
FAV40	2.5	1.0	0.0	1.8	0.0	0.0	0.0	2.0	7.3
HMM40	19.3	4.4	0.0	10.0	0.0	.1	0.0	23.8	57.6
FGATM	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.7	1.7
DRAGN	39.7	32.3	0.0	.1	0.0	.6	0.0	5.6	78.3
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
INFTY	0.0	294.4	0.0	.1	56.9	1.2	0.0	25.2	377.8
ARTY	0.0	3.4	0.0	21.9	0.0	0.0	0.0	4.4	29.7
VULCN	0.0	1.1	0.0	1.4	0.0	0.0	0.0	.3	2.8
MLRS	0.0	.7	0.0	0.0	0.0	0.0	0.0	0.0	.7
MORTR	0.0	10.9	0.0	0.0	0.0	0.0	0.0	1.2	12.1
ICHAP	0.0	.5	0.0	.8	0.0	0.0	0.0	.1	1.4
IHAWK	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STING	2.0	7.0	0.0	0.0	0.0	0.0	0.0	.3	9.3
CMD-V	14.5	5.8	0.0	15.7	0.0	.2	0.0	.3	36.5
F-TRK	0.0	.1	0.0	0.0	0.0	0.0	0.0	1.6	1.7
A-TRK	0.0	.6	0.0	.1	0.0	0.0	0.0	14.9	15.6
SP-VE	12.3	.2	0.0	0.0	0.0	.2	0.0	9.3	22.0

BLUE KILLER--RED VICTIM FILE FOR GAME 1 TURN 2

VICTIM	D/F	I/F	PGM	A/H	INF	MIN	CHM	AIR	T/KILL
	<----------			KILLER				---------->	
T80	13.3	3.6	0.0	4.9	0.0	.4	0.0	12.5	34.7
BMP	7.5	11.4	0.0	2.9	0.0	.4	0.0	2.5	24.7
BMP81	1.5	.5	0.0	.5	0.0	0.0	0.0	2.7	5.2
BTR	7.1	5.6	0.0	13.2	0.0	1.1	0.0	17.5	44.5
BRDM2	1.9	2.4	0.0	2.7	0.0	.2	0.0	2.7	9.9
SPAT	.4	.2	0.0	.8	0.0	.1	0.0	.5	2.0
AT-4	1.2	3.0	0.0	0.0	0.0	.2	0.0	1.3	5.7
ASU85	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
BMD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.5	9.5
INFTY	104.3	119.4	0.0	118.8	98.0	11.1	0.0	233.8	665.4
ARTY	0.0	3.4	0.0	9.5	0.0	0.0	0.0	60.8	73.7
ZSU-X	.4	1.0	0.0	.4	0.0	0.0	0.0	1.8	3.6
MRL	0.0	2.7	0.0	4.8	0.0	0.0	0.0	19.7	27.2
MORTR	2.2	26.5	0.0	0.0	0.0	.3	0.0	6.4	35.4
SA-13	0.0	1.7	0.0	.2	0.0	0.0	0.0	2.3	4.2
SA-8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	.2	.2
SA-14	2.5	15.4	0.0	0.0	0.0	0.0	0.0	8.5	26.4
CMD-V	5.7	7.5	0.0	13.6	0.0	1.3	0.0	32.5	60.6
F-TRK	0.0	0.0	0.0	.1	0.0	0.0	0.0	5.3	5.4
A-TRK	0.0	0.0	0.0	.2	0.0	0.0	0.0	40.1	40.3
SP-VE	15.6	0.0	0.0	.3	0.0	4.0	0.0	32.5	52.4

ATTACK HELICOPTER RESULTS

TYPE	#KILLED	#SORTIES
AH64	1.9	52.4
LCH	0.0	0.0
AHIP	1.2	4.5
HIND	17.4	49.5

Figure 10-2. Killer/victim scoreboard.

```
GAMER STAFF WORK  SHEET FOR TURN 3

BLUE UNITS:


UNIT    1      LIDTEST      TYPE: 1.6

LINE 1:  ACTIVITY: 1         MOPP LEVEL: 1       MISSION: 8.0    KM MOVED:    0

LINE 2:  SENSOR GP: 4          ZONE: 4           PCT COVERED:  50

LINE 3:  PCT ADA SUPPRESSION:  VEH:    0         HAND:   0         CORPS ADA:    6

LINE 4:  RESUPPLY:  %DF: __   %IF: __   %AD: __   AMMO(tons): __  FUEL(gal): __

LINE 5:  DISPENSED: %DF: __   %IF: __   %AD: __   AMMO(tons): __  FUEL:(gal) __


UNIT    2      LID1-3RD     TYPE: 1.6

LINE 1:  ACTIVITY: 1         MOPP LEVEL: 1       MISSION: 8.0    KM MOVED:    0

LINE 2:  SENSOR GP: 4          ZONE: 4           PCT COVERED:  50

LINE 3:  PCT ADA SUPPRESSION:  VEH:    0         HAND:   0         CORPS ADA:    6

LINE 4:  RESUPPLY:  %DF: __   %IF: __   %AD: __   AMMO(tons): __  FUEL(gal): __

LINE 5:  DISPENSED: %DF: __   %IF: __   %AD: __   AMMO(tons): __  FUEL:(gal) __


UNIT    3      3RDEHQTR     TYPE: 1.6

LINE 1:  ACTIVITY: 1         MOPP LEVEL: 1       MISSION: 8.0    KM MOVED:    0

LINE 2:  SENSOR GP: 4          ZONE: 4           PCT COVERED:  50

LINE 3:  PCT ADA SUPPRESSION:  VEH:    0         HAND:   0         CORPS ADA:    6

LINE 4:  RESUPPLY:  %DF: __   %IF: __   %AD: __   AMMO(tons): __  FUEL(gal): __

LINE 5:  DISPENSED: %DF: __   %IF: __   %AD: __   AMMO(tons): __  FUEL:(gal) __


UNIT    4      CBAAHQTR     TYPE: 1.6

LINE 1:  ACTIVITY: 1         MOPP LEVEL: 1       MISSION: 8.0    KM MOVED:    0

LINE 2:  SENSOR GP: 4          ZONE: 4           PCT COVERED:  50

LINE 3:  PCT ADA SUPPRESSION:  VEH:    0         HAND:   0         CORPS ADA:    5

LINE 4:  RESUPPLY:  %DF: __   %IF: __   %AD: __   AMMO(tons): __  FUEL(gal): __

LINE 5:  DISPENSED: %DF: __   %IF: __   %AD: __   AMMO(tons): __  FUEL:(gal) __
```

Figure 10-3. Gamer/staff worksheet.

(2) If game turn updates are wanted (answer the first question with "N"), more information is needed before the process may continue. It is important to know what attrition programs (P3, P4, P5 or P6) have been run during the CI.

(3) After answering correctly about the attrition programs, enter characters to identify the game and the turn (CI). A final abort of P8 is then offered before irreversible changes are made during the running of this program.

B. Outputs consist of the unit history, K/V summary, and gamer/staff worksheets discussed in paragraph 2. For necessary file outputs of P8, refer to paragraph 6 of this chapter.

C. A data flow chart is shown in Figure 10-4.

D. Information about the external data files may be found in the following paragraph.

4. FILE STRUCTURE.

A. "UNITFILE". The unit status file is a 400-record file which holds 150 elements of information for each of the 400 units. A complete description may be found in Chapter 1, Table 1-1.

B. "TOEFILE". The "TOEFILE" holds the beginning 70 weapon systems from the "UNITFILE", an effectiveness number, side, and mission data.

C. "NAMEFILE". The "NAMEFILE" holds the name of each of the 400 units in the "UNITFILE".

D. "KVFILE". Holds killer/victim data for Blue and Red.

E. "HELOFILE". Holds attack helicopter data.

F. "BL/RDCHMVCTM". Holds Blue and Red chemical victim data.

G. "B/RAIR_IN". Holds Red and Blue air losses. The "BAIR_IN" file holds the number of kills by the Blue aircraft (number of Red system losses) and the "RAIR_IN" files contains the number of kills by Red aircraft (Blue system losses).

H. All P1 Files. See Chapter 2 for greater detail.

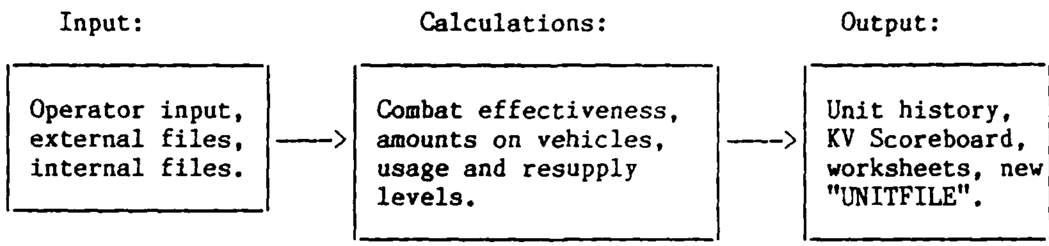


Figure 10-4. Unit status report data flow.

5. ALGORITHMS.

The logic flow of the unit status report is shown in Figure 10-5. Most of the calculations in P8 are done through loss assessment files that have been accumulated over the preceding six-hour game turn. The main algorithms are as follows:

A. Current ammunition on cargo vehicles.

$$New\_N123 = N123 * N58 / N84 \qquad \text{(Eq. 10-1)}$$

where:

New_N123 = current ammunition on cargo vehicles in short tons.
N123 = preceding ammunition on cargo vehicles in short tons.
N58 = current number of ammo trucks.
N84 = cargo trucks alive at start of preceding game turn.

B. Current fuel on fuel vehicles.

$$New\_N103 = N103 * N55 / N85 \qquad \text{(Eq. 10-2)}$$

where:

New_N103 = current fuel on fuel vehicles in gallons.
N103 = preceding fuel on fuel trucks in gallons.
N55 = current number of fuel trucks.
N85 = fuel trucks alive at start of preceding game turn.

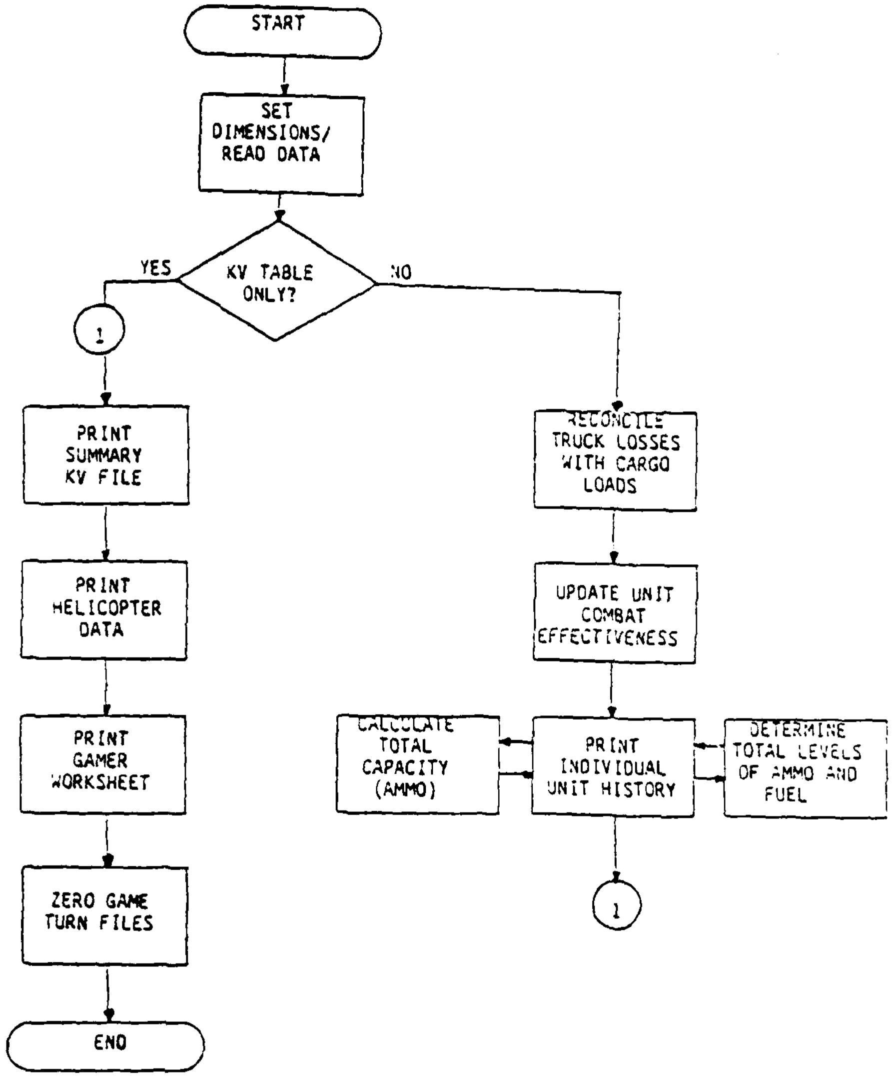


Figure 10-5. Unit status report logic flow.

C. Calculation of the total amount of ammunition on vehicles for each of the three weapon types (DF, IF, AD) is done as follows:

$$Tra_i = \sum_{j=1}^{70} (N_{ij} * Ac_{ij}) \qquad \text{(Eq. 10-3)}$$

where:

Tra_i = total amount of ammunition on vehicles of type i (DF, IF, AD) in short tons.
N_{ij} = number of "UNITFILE" system elements (j) which are of type i.
Ac_{ij} = ammunition capacity for system element (j) which is of type i in short tons.

D. The total amount of ammunition per weapon type (DF, IF, AD) is:

$$Ta_i = \sum_{j=1}^{70} (Ac_{ij} * N_{ij} * As_i) \qquad \text{(Eq. 10-4)}$$

where:

Ta_i = total amount of ammunition per weapon type i (DF, IF, AD) (short tons).
As_i = ammunition status for weapon type i; the percentage of total ammunition capacity the vehicles are actually holding.

E. Total fuel for all system elements is calculated as follows:

$$Tf = \sum_{j=1}^{70} (N_j * Fc_j * N101) \qquad \text{(Eq. 10-5)}$$

where:
Tf = total fuel (gallons) for all system elements.
N_j = number of "UNITFILE" system elements (j).
Fc_j = fuel capacity (gallons) for system element j.

N101 = fuel status of unit vehicles; the percentage of total fuel capacity the vehicles are actually holding.

6. FILE EFFECTS.

P8 affects various files used in DIME, some in such a way that the only way they can be recovered is by rerunning a CI.

A. "UNITFILE" impact. Updates the fuel status, ammunition status, and combat effectiveness of each unit.

B. "KVFILE". Zeros out all entries in the K/V file after it has printed the worksheets.

C. "HELOFILE". Zeroes the helicopter file at the end of processing.

D. Chem files. Zeroes chemical files at the end of processing.

E. Air files. Zeroes air loss files at the end of processing.

7. CODE.

A. This section contains information on the unit status report program code. For a generalized flow of this program, refer to Figure 10-5.

B. The program begins by entering data necessary to open the appropriate files. Once the files have been accessed, the following areas are done for each of the 400 units:

(1) The appropriate data is read for each unit.

(2) If the unit is active, the current fuel and ammunition on the cargo vehicles is calculated. The number of ammunition and fuel trucks that will begin in the following critical incident (CI) are saved within the "UNITFILE". Finally, for each active unit, the combat effectiveness is recalculated.

(3) All units, active or inactive, go through the following process:

(a) The total amount of ammunition on vehicles for each of the three weapon types (DF, IF, AD) is calculated.

(b) Ammunition status for each weapon type (DF, IF, AD) is recalculated. Ammunition available to be consumed is set to zero and the ammunition which remained after the CI is placed in entirety on the ground.

(c) The new distribution of ammunition on the ground for DF, IF, and AD is figured.

(d) Calculations for the total fuel over all systems and the total ammunition for each weapon type (DF, IF, AD) then occur.

(e) Resupply and dispensation information is zeroed out to prepare for new resupply information to be entered in P1 (game initialization).

(f) The status report for each unit is printed and information is updated to the "UNITFILE".

(4) The killer/victim scoreboard and helicopter information are printed. Following these are the gamer/staff worksheets.

(5) All cumulative kill files are zeroed to prepare for the following CI.

C. A subroutine and variable listing is shown in Table 10-1. A code listing appears in Table 10-2.

Table 10-1. Unit status report subroutine table.

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
A. Read data and determines which reports will be produced.	Main program	Opens and enters required files and calls subroutines.	a. N(*)	75 dimensional array holding data provided by the "UNITFILE"
			b. U(*)	23 dimensional array holding data provided by the "FOEFILE"
	Blue_data	Reads Blue unit arrays.	a. Wpn_type(*)	Array containing the weapon type of each element. Weapon types are: 1 = Indirect fire 2 = Direct fire 3 = Air defense.
			b. Ammo_cap(*)	Array containing amount of ammo capacity for a particular element from 1 to 21
			c. Fuel_cap(*)	Array containing amount of fuel capacity for a particular element from 1 to 21
	Red_data	Reads Red unit arrays.	See Blue_data subroutine variable list.	
B. Killer/victim output.	Zero	Zeroes chemical and air loss variables.	a. R_air_loss(*)	Array containing losses of Red targets due to Blue air fire
			b. B_air_loss(*)	Array containing losses of Blue targets due to Red air fire
			c. R_chem_loss(*)	Array containing losses of Red targets due to Blue chemicals
			d. B_chem_loss(*)	Array containing losses of Blue targets due to Red chemicals

Table 10-1. Unit status report subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
B. Killer/victim output (concluded).	Print_kvfile; Zero_lost; List_kv	Calls the subroutines. Zero_list and list_kv; Zeroes variables for Kv print; prints kv table.	a. Kv(I,J)	Array containing losses to target element J due to weapon type I where: I = 1 Direct fire = 2 Indirect fire = 3 PGM = 4 Attack helicopter = 5 Infantry = 6 Mines = 7 Chemicals = 8 Air defense J = 1 to 21 elements
			b. Kv_tot(*)	Array containing total number of losses to target elements due to all weapon types
	Print_helo_file	Prints helicopter data.	Ci_helo_b(I,J)	Array containing Blue type I helicopter data where: I = 1 HIND J = 2 # killed J = 6 # sorties
C. Unit history and gamer input worksheet portion.	Update_fuel	Reconciles truck losses with cargo loads.	a. New_ammo	Current ammo truck loads in tons
			b. New_fuel	Current fuel truck loads in gallons

Table 10-1. Unit status report subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
C. Unit history and gamer input worksheet portion (continued).	Update_cbt_eff	Updates unit combat effectiveness.	a. Eff	Systems effectiveness
			b. Cbt_eff	Unit combat effectiveness
	Print_unit_stat	Prints individual unit history.	a. Df_stat(I)	Array containing direct fire supply status where: I = 1 ammo resupply profile I = 2 distribution of ammo in cargo vehicles I = 3 distribution of ground ammo I = 4 ammo dispensed to other vehicles
			b. If_stat(I)	Array containing indirect fire supply status I = same as above
			c. Ad_stat(I)	Array containing air defense supply status I = same as above
			d. Tdf	Current level of direct fire ammo in tons
			e. Tif	Current level of indirect fire ammo in tons
			f. Tad	Current level of air defense ammo in tons
			g. Df_used	Direct fire ammo consumed in tons

Table 10-1. Unit status report subroutine table (continued).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
C. Unit history and gamer input worksheet portion (continued).	Print_unit_stat (concluded)		h. If_used	Indirect fire ammo consumed in tons
			i. Ad_used	Air defense ammo consumed in tons
			j. C_fuel	Unused fuel in gallons
			k. Gnd_ammo	Total direct fire, indirect fire, and air defense ammo on the ground
			l. Trk_cap	Combined fuel capacity of trucks
	Print_sup_out	Prints required systems lists.	N(*)	75 dimensional array for holding data provided by the "UNITFILE"
	Update_ammo	"Loads trucks" after calculating DF, IF and AD ammo capacities.	a. Tot_veh_ammo(*)	Array containing total amount of ammo capacity by vehicles for each weapon type 1 to 3
			b. Ammo_left(*)	Array containing ammo available to be consumed for each weapon type 1 to 3
			c. Pdf	% direct fire ammo on ground in tons
			d. Pif	% indirect fire ammo on ground in tons
			e. Atrk_cap	Combined ammo capacity of trucks in tons

Table 10-1. Unit status report subroutine table (concluded).

Functional area(s)	Subroutine called	Subroutine function(s)	Primary variables	Variable description
C. Unit history and gamer input worksheet portion (concluded).	Update_ammo (concluded)		f. Tk_emp	Number of empty trucks
			g. Tk_ammo(*)	Array containing capacity of trucks for resupplying weapon types 1 to 3
			h. Tk_perc(*)	Array containing weapon type 1 percent of ground ammo for weapon types 1 to 3
	Veh_ammo	Calculates total levels of ammo and fuel as related to weapon systems.	a. Tot_ammo(*)	Array containing total ammo capacity of weapon types from 1 to 3
			b. T_fuel	Fuel level of weapon systems combined

Table 10-2. Unit status report code.

```
10    REM   "P8" DOES THE END OF GAMER TURN UPDATES, PRINTS GAME RECORDS
20    !   AND THE GAMER COMBAT SUPPORT INPUT SHEET.
30    ! DATA CHANGED FEBRUARY 3, 1986, ROB BELFLOWER, BDM
40    !*******************************************************************
50    ! EXPANDED VERSION -- JUNE 9, 1986 -- BY OAO CORP.
51    !      DECLASSIFIED -- AUG 7, 1986 -- BY OAO CORP.   ** DC **
60    OPTION BASE 1
70    DIM N(150),S(70),U(72),A(2,70),F(2,70),W(2,70),Sys_eff(2,70),Helo(3,6)
80    DIM Ci_kv_b(8,70),Ci_kv_r(8,70),Fil1$[10],Fil2$[10],Fil3$[10],Fil4$[10]
90    DIM Turn$[3],Game$[2],Msus$[13],Id$[4],M$[16],M1$[9],T$[7]
100   DIM B_veh$[350],R_veh$[350],Veh$[350]
110   DIM R_air_loss(72),B_air_loss(72),R_chem_loss(70),B_chem_loss(70)
120   DIM Df_stat(4),If_stat(4),Ad_stat(4),Kv(8,70),Kv_tot(70)
130   DIM Ci_helo_b(3,6),Ci_helo_r(3,6),Tot_ammo(3),Fuel_cap(70)
140   DIM Ammo_cap(70),Wpn_type(70),Tot_veh_ammo(3),Ammo_left(3)
150   DIM T1(28),Tk_ammo(3),Tk_perc(3),Truck_cap(7)
151   DIM Dummyrec(70)   ! ** DC **
160   INTEGER I,J,K,T
170   !
180   ! READ REQUIRED DATA ARRAYS
190   Disk$=":9134,704,0"
191   Dcdisk$=":9134,704,0"   !   ** DC **
200   GOSUB Read_data
210   !
220   ! PRINT HEADERS
230   PRINTER IS 1
240   PRINT USING "@,#"
250   PRINT TABXY(28,8),"DIME GAME TURN PROCESSOR"
260   PRINT USING "///"
270   PRINT "          BE SURE THAT ALL PROCESSING PROGRAMS HAVE BEEN COMPLETED
280   PRINT "                          BEFORE RUNNING THIS PROGRAM"
290   PRINT USING "////"
300   PRINT "PLEASE ANSWER THE FOLLOWING:"
310   REPEAT
320     INPUT "KILLER VICTIM TABLE LISTING ONLY?(Y/N)",Kvans$
330   UNTIL Kvans$="Y" OR Kvans$="N"
340   GOTO Open
350   REPEAT
360     INPUT "HAVE YOU RUN P3 AIRSTRIKE (Y/N)",P3$
370   UNTIL P3$="Y" OR P3$="N"
380   REPEAT
390     INPUT "HAVE YOU RUN P4 COMBAT (Y/N)",P4$
400   UNTIL P4$="Y" OR P4$="N"
410   REPEAT
420     INPUT "HAVE YOU RUN P5 CHEMICAL (Y/N)",P5$
430   UNTIL P5$="Y" OR P5$="N"
440   REPEAT
450     INPUT "HAVE YOU RUN P6 LOSS ASSESSMENT (Y/N)",P6$
460   UNTIL P6$="Y" OR P6$="N"
470   INPUT "ENTER GAME NUMBER:",Game$,"ENTER TURN NUMBER:",Turn$
480   T=VAL(Turn$)
490   !    OPEN REQUIRED FILES
```

Table 10-2. Unit status report code (continued).

```
500    ASSIGN @Punit TO "UNITFILE"&Disk$
510    ASSIGN @Ptoe TO "TOEFILE"&Disk$
520    ASSIGN @Pname TO "NAMEFILE"&Disk$
530    IF P4$="Y" OR P6$="Y" THEN
540      ASSIGN @Pkv TO "KVFILE"&Disk$
550      ASSIGN @Phelo TO "HELOFILE"&Disk$
560    END IF
570    IF P5$="Y" OR P6$="Y" THEN
580      ASSIGN @Pbchem TO "BLCHMVCTM"&Disk$
590      ASSIGN @Prchem TO "RDCHMVCTM"&Disk$
600    END IF
610    IF P3$="Y" OR P6$="Y" THEN
620      ASSIGN @Bair TO "BL_AIR_INF"&Disk$
630      ASSIGN @Rair TO "RD_AIR_INF"&Disk$
640    END IF
650 Open:    ! ASSIGNS ANSWERS TO EXTRANEOUS QUESTIONS ABOVE.
660    P3$="Y"
670    P4$="Y"
680    P5$="N"
690    P6$="Y"
700    INPUT "ENTER GAME NUMBER:",Game$,"ENTER TURN NUMBER:".Turn$
710    INPUT "ENTER THE GAME TIME",T$
720    T=VAL(Turn$)
730    !     OPEN REQUIRED FILES
740    ASSIGN @Punit TO "UNITFILE"&Disk$
750    ASSIGN @Ptoe TO "TOEFILE"&Disk$
760    ASSIGN @Pname TO "NAMEFILE"&Disk$
770    ASSIGN @Pkv TO "KVFILE"&Disk$
780    ASSIGN @Phelo TO "HELOFILE"&Disk$
790    ASSIGN @Bair TO "BL_AIR_INF"&Disk$
800    ASSIGN @Rair TO "RD_AIR_INF"&Disk$
810    ASSIGN @Pbchem TO "BLCHMVCTM"&Disk$
820    ASSIGN @Prchem TO "RDCHMVCTM"&Disk$
830    IF Kvans$="Y" THEN Kv_only
840    !
850    REPEAT
860      INPUT "DO YOU WISH TO RUN P8? (Y/N)",Y_n_$
870    UNTIL Y_n_$="Y" OR Y_n_$="N"
880    IF Y_n_$="N" THEN GOTO Halt
890    !
900    !
910    !  PRINT HEADERS FOR UNIT HISTORY FILE
920    PRINT USING "@,#"
930    PRINTER IS 702
940    PRINT USING "@.#,10X,22A,2A,7A,3A,2X,4A,1X,7A,//";"UNIT HISTORY FOR GAME
,Game$," TURN ",Turn$,"TIME",T$
950    PRINT USING "11A./";"BLUE UNITS:"
960    PRINTER IS 1
970    !
980    ! BEGIN INDIVIDUAL UNIT PROCESSING
990    !
1000   Tdf_used=0
```

Table 10-2. Unit status report code (continued).

```
1010  Tif_used=0
1020  Tad_used=0
1030  Totaldf=0
1040  Totalif=0
1050  Totalad=0
1060  Totalfuel=0
1070  Currentfuel=0
1080  GOSUB Blue_data
1090  !
1100  FOR I=1 TO 400
1110    IF I=192 THEN
1120      PRINTER IS 702
1130      PRINT USING "@,#,10X,22A,1A,7A,3A,2X,4A,1X,7A,//";"UNIT HISTORY FOR G
ME ",Game$," TURN ",Turn$,"TIME",T$
1140      PRINT USING "11A,/";"RED UNITS:"
1150      PRINTER IS 1
1160      Tdf_used=0
1170      Tif_used=0
1180      Tad_used=0
1190      Totaldf=0
1200      Totalif=0
1210      Totalad=0
1220      Totalfuel=0
1230      Currentfuel=0
1240      GOSUB Red_data
1250    END IF
1260    ENTER @Punit,I;N(*)
1270    ENTER @Ptoe,I;U(*)
1280    ENTER @Pname,I;M$
1290    PRINTER IS 1
1300    IF M$="UNUSED              " THEN GOTO Print_out
1310    PRINT USING "/,16A,3D,/";"PROCESSING UNIT ";I
1320    IF I<192 THEN
1330      Side=1
1340    ELSE
1350      Side=2
1360    END IF
1370    !
1380    ! DO UNIT UPDATES
1390    GOSUB Update_fuel
1400    GOSUB Update_cbt_eff
1410    !
1420    !
1430  Print_out:GOSUB Print_unit_stat
1440    GOSUB Print_work_sht
1450    OUTPUT @Punit,I;N(*)
1460    IF I=191 OR I=400 THEN
1470      PRINT
1480      PRINT
1490      PRINT    ! ROB
1500      PRINT USING "40A,2X,8D,1X,4A";"TOTAL DF USED DURING THE LAST SIX HOUR
:";Tdf_used;"TONS"
```

Table 10-2. Unit status report code (continued).

```
1510        PRINT USING "40A,2X,8D,1X,4A";"TOTAL IF USED DURING THE LAST SIX HOUR
:";Tif_used;"TONS"
1520        PRINT USING "40A,2X,8D,1X,4A";"TOTAL AD USED DURING THE LAST SIX HOUR
:";Tad_used;"TONS"
1530        PRINT USING "46A,2X,14D,1X,7A";"TOTAL FUEL CONSUMED DURING THE LAST S
X HOURS:";Totalfuel;"GALLONS"
1540        PRINT
1550        PRINT USING "18A,8D,1X,4A";"TOTAL DF REMAINING";Totaldf;"TONS"
1560        PRINT USING "18A,8D,1X,4A";"TOTAL IF REMAINING";Totalif;"TONS"
1570        PRINT USING "18A,8D,1X,4A";"TOTAL AD REMAINING";Totalad;"TONS"
1580        PRINT USING "20A,10D,1X,7A";"TOTAL FUEL REMAINING";Currentfuel;"GALLON
S"
1590      END IF
1600    NEXT I
1610    !
1620 Kv_only:  !
1630    PRINTER IS 1
1640    PRINT USING "@,#"
1650    PRINT TABXY(26,12),"PRINTING KILLER-VICTIM TABLE"
1660    PRINTER IS 702
1670    !PRINT USING "@,#"
1680    GOSUB Zero
1690    !
1700    IF P4$="Y" OR P6$="Y" THEN
1710      ENTER @Pkv,1;Ci_kv_b(*)
1720      ENTER @Pkv,2;Ci_kv_r(*)
1730    END IF
1740    !
1750    !  READ IN AND ENTER IN CHEMICAL/CLOSE AIR KILLS
1760    IF P5$="Y" OR P6$="Y" THEN
1770      ENTER @Pbchem,1;B_chem_loss(*)
1780      ENTER @Prchem,1;R_chem_loss(*)
1790    END IF
1800    IF P3$="Y" OR P6$="Y" THEN
1810      ENTER @Bair,1;T1(*),R_air_loss(*)
1820      ENTER @Rair,1;T1(*),B_air_loss(*)
1830    END IF
1840    FOR I=1 TO 70
1850      IF P3$="Y" OR P6$="Y" THEN
1860        Ci_kv_r(8,I)=R_air_loss(I)+Ci_kv_r(8,I)
1870        Ci_kv_b(8,I)=B_air_loss(I)+Ci_kv_b(8,I)
1880      END IF
1890      IF P5$="Y" OR P6$="Y" THEN
1900        Ci_kv_b(7,I)=B_chem_loss(I)+Ci_kv_b(7,I)
1910        Ci_kv_r(7,I)=R_chem_loss(I)+Ci_kv_r(7,I)
1920      END IF
1930    NEXT I
1940    !
1950    ! PRINT OUT KILLER-VICTIM SUMMARY FILE
1960    GOSUB Print_kvfile
1970    !
1980    ! PRINT OUT HELICOPTER FILE
```

Table 10-2. Unit status report code (continued).

```
1990  IF P4$="Y" THEN GOSUB Print_helo_file
2000  IF Kvans$="Y" THEN Halt
2010  IF Kvans$="N" THEN Zero_files
2020  !
2030  !  PRINT OUT GAMER INPUT WORKSHEET HEADER
2040  T1=T+1
2050  PRINTER IS 1
2060  PRINT USING "@,#"
2070  PRINT TABXY(24,12),"PRINTING GAMER STAFF WORK SHEETS"
2080  PRINTER IS 702
2090  PRINT USING "@,#,32A,2D";"GAMER STAFF WORK  SHEET FOR TURN ",T1
2100  PRINT USING "//,13A";"BLUE UNITS:"
2110  !
2120  ! PRINT INDIVIDUAL UNITS
2130  FOR I=1 TO 400
2140    ENTER @Punit,I;N(*)
2150    ENTER @Pname,I;M$
2160    GOSUB Print_work_sht
2170  NEXT I
2180  !
2190  !
2200 Zero_files:  !STAFF WORK SHEETS ARE PRINTED WITH UNIT STATUS
2210  !   ZERO GAME TURN FILES FOR USE IN NEXT GAME TURN
2220  IF P4$="Y" OR P6$="Y" THEN
2230    FOR I=1 TO 8
2240      FOR J=1 TO 70
2250        Ci_kv_b(I,J)=0
2260      NEXT J
2270    NEXT I
2280    FOR I=1 TO 2
2290      OUTPUT @Pkv,I;Ci_kv_b(*)
2300    NEXT I
2310  !ZERO HELICOPTER FILES
2320    FOR I=1 TO 3
2330      FOR J=1 TO 6
2340        Helo(I,J)=0
2350      NEXT J
2360    NEXT I
2370    OUTPUT @Phelo,1;Helo(*)
2380    OUTPUT @Phelo,2;Helo(*)
2390  END IF
2400  !
2410  IF P5$="Y" OR P6$="Y" THEN
2420  !ZERO CHEMICAL FILE
2430    FOR I=1 TO 70
2440      R_chem_loss(I)=0
2450    NEXT I
2460    OUTPUT @Pbchem,1;R_chem_loss(*)
2470    OUTPUT @Prchem,1;R_chem_loss(*)
2480  END IF
2490  !
2500  IF P3$="Y" OR P6$="Y" THEN
```

Tabke 10-2. Unit status report code (continued).

```
2510  !ZERO AIR LOSS FILES
2520    FOR I=1 TO 72
2530      B_air_loss(I)=0
2540    NEXT I
2550    FOR I=1 TO 28
2560      T1(I)=0
2570    NEXT I
2580    OUTPUT @Rair,1;T1(*),B_air_loss(*)
2590    OUTPUT @Bair,1;T1(*),B_air_loss(*)
2600  END IF
2610  !
2620  !   CLOSE ALL FILES AND RETURN TO MENU PROGRAM
2630  ASSIGN @Punit TO *
2640  ASSIGN @Ptoe TO *
2650  ASSIGN @Pname TO *
2660  IF P4$="Y" OR P6$="Y" THEN ASSIGN @Pkv TO *
2670  IF P4$="Y" OR P6$="Y" THEN ASSIGN @Phelo TO *
2680  IF P3$="Y" OR P6$="Y" THEN ASSIGN @Bair TO *
2690  IF P3$="Y" OR P6$="Y" THEN ASSIGN @Rair TO *
2700  IF P5$="Y" OR P6$="Y" THEN ASSIGN @Pbchem TO *
2710  IF P5$="Y" OR P6$="Y" THEN ASSIGN @Prchem TO *
2720  GOTO Halt
2730  !
2740  !
2750  !     *********   END OF MAIN PROGRAM   *********
2760  !
2770  !
2780 Read_data:  !  THIS SBR READS REQUIRED DATA
2790  !
2800  !   ** DC **
2810  !
2820  ASSIGN @Psyseff TO "SYS_EFF"&Dcdisk$
2830  ENTER @Psyseff,1;Sys_eff(*)
2840  ASSIGN @Psyseff TO *
2850  !   ** END DC **
2960  B_veh$[1,125]="DF    FAV-TM551 FAV40HMV-GDF   DRAGNLAW  DF   CMD-VDF   DF
 DF   DF   DF   HMV40DF-ICDF-ICDF-ICDF-ICARTY ARTY ARTY ARTY ARTY "
2970  B_veh$[126,250]="ARTY ARTY MORTRMORTRMORTRMORTRMLRSTMLRSTMLRSTMLRSTINF  IN
F  INF  INF  INF  SARMSSARMSSARMSSARMSSARMSSARMSSARMSVULCNAVNGRIHAWK"
2980  B_veh$[251,350]="ADA  ADA  STINGADAHHF-TRKJ4TRKWATERCGO-TNATRKEWTRKEWTRKEN
GR OBSCEAVLB PONBRENGEQENGEQMATHEMATHEAATHE"
2990  R_veh$[1,125]="T55  DF   BMP73DF   BRDM3BRDM5AT-75AGS17T12  CMD-VDF   DF
 DF   DF   DF   BMPATBTR  DF-ICDF-ICDF-ICARTY ARTY ARTY ARTY ARTY "
3000  R_veh$[126,250]="ARTY ARTY MORTRMORTRMORTRMORTRMRL  MRL  MRL  MRL  INF  IN
F  INF  INF  INF  SARMSSARMSSARMSSARMSSARMSSARMSSARMSZSU-XSA-13SA-6 "
3010  R_veh$[251,350]="ADA  ADA  SA-14ADAHHF-TRKJ4TRKWATERCGO-TNATRKEWTRKEWTRKEN
GR OBSCEAVLB PONBRENGEQENGEQMATHEMATHEAATHE"
3020  RETURN
3030 Print_sys_out:  !  THIS SBR PRINTS OUT THE REQUIRED SYSTEMS LISTS
3040  FOR X=7 TO 70 STEP 7
3050    PRINT USING Fmt1:X-6,":",N(X-6),X-5,":",N(X-5),X-4,":",N(X-4),X-3,":",N(
X-3),X-2,":",N(X-2),X-1,":",N(X-1),X,":",N(X)
```

Table 10-2. Unit status report code (continued)

```
3060  NEXT X
3070 Fmt1:IMAGE 7(2D,1A,4D.1D,2X)
3080  !
3090  RETURN
3100  '
3110  !************************************************************************
3120  '
3130 Zero:  !
3140  FOR I=1 TO 8
3150    FOR J=1 TO 70
3160      Ci_kv_b(I,J)=0
3170      Ci_kv_r(I,J)=0
3180    NEXT J
3190  NEXT I
3200  FOR I=1 TO 70
3210    B_chem_loss(I)=0
3220    R_chem_loss(I)=0
3230    B_air_loss(I)=0
3240    R_air_loss(I)=0
3250  NEXT I
3260  B_air_loss(71)=0
3270  B_air_loss(72)=0
3280  R_air_loss(71)=0
3290  R_air_loss(72)=0
3300  RETURN
3310  !
3320  ! ***********************************************************************
3330  !
3340 Update_fuel:!  THIS SBR RECONCILES TRUCK LOSSES WITH CARGE LOADS
3350  '
3360  Ammo=N(123)
3370  Ammog=N(125)
3380  IF N(84)<=0 THEN
3390    N(123)=0
3400    N(125)=0
3410    GOTO 3480
3420  END IF
3430  New_ammo=Ammo*N(58)/N(84)
3440  N(123)=New_ammo
3450  New_ammog=Ammog*N(58)/N(84)
3460  N(125)=New_ammog
3470  '
3480  Fuel=N(103)
3490  Fuelg=N(105)
3500  IF N(85)<=0 THEN
3510    N(103)=0
3520   'N(105)=0
3530    GOTO 3590
3540  END IF
3550  New_fuel=Fuel*N(55)/N(85)
3560  N(103)=New_fuel
3570  N(105)=Fuelg*N(55)/N(85)
```

Table 10-2. Unit status report code (continued).

```
3580  !
3590  N(85)=N(55)
3600  N(84)=N(58)
3610  !
3620  RETURN
3630  !
3640  ! ***********************************************************************
3650  !
3660 Update_cbt_eff:!  THIS SBR UPDATES UNIT COMBAT EFFECTIVENESS
3670  !
3680  Eff=0
3690  FOR J=1 TO 70
3700    Eff=Eff+Sys_eff(Side,J)*N(J)
3710  NEXT J
3720  IF U(71)=0 THEN 3760
3730  Cbt_eff=Eff/U(71)
3740  N(79)=Cbt_eff
3750  !
3760  RETURN
3770  !
3780  ! ***********************************************************************
3790  !
3800 Print_unit_stat:  ! THIS SBR PRINTS OUT THE INDIVIDUAL UNIT HISTORY
3810  !
3820  PRINTER IS 702
3830  IF N(82)=0 THEN M1$="INACTIVE"
3840  IF N(82)=1 THEN M1$="ACTIVE  "
3850  IF N(82)=2 THEN M1$="DESTROYED"
3860 !  PRINT USING Fmt2:"UNIT: ",I,M$,"TYPE: ",N(78),M1$,"CBT-EFF: ",N(79),"MI
SION: ",N(83)
3870 ! Fmt2:IMAGE /,6A,3D,3X,8A,3X,6A,1D.1D,3X,8A,3X,9A,3D.2D,3X,9A,1D.1D
3880  PRINT USING Fmt2;"UNIT: ",I,M$,"TYPE: ",N(78),M1$,"CBT-EFF: ",N(79)
3890 Fmt2:IMAGE /,6A,3D,3X,16A,3X,6A,1D.1D,3X,9A,3X,9A,3D.2D,3X
3900  IF N(82)=0 OR N(82)=2 THEN End_stat_print
3910  FOR J=1 TO 70
3920    S(J)=N(J)
3930  NEXT J
3940  PRINT USING "/,26A";"SYSTEMS REMAINING:"
3950  GOSUB Print_sys_out
3960  GOTO Log
3970  SELECT N(91)
3980  CASE =0
3990    M1$="NOT DET "
4000  CASE =1
4010    M1$="DETECTED"
4020  CASE =2
4030    M1$="VERIFIED"
4040  CASE =3
4050    M1$="LOST    "
4060  END SELECT
4070  PRINT USING Fmt3;"DETECTION: ",M1$,"SENS STATUS (Z/G): ",N(89),"FRA CVG:
,N(90),"KM MOVED: ",N(146)
```

Table 10-2. Unit status report code (continued).

```
4080 Fmt3:IMAGE /,11A,8A,3X,19A,1D.1D,4X,9A,1D.2D,6X,10A,3D
4090  PRINT USING Fmt4;"AIR DEFENSE:","SUPPRESSION: ".N(80)."CORPS SUPPORT ADA:
".N(81)
4100 Fmt4:IMAGE /,12A,4X,13A,2D.2D,4X,19A,3D
4110 Log:  !LOGISTICS
4120  PRINT USING "/,10A";"LOGISTICS:"
4130  PRINT USING Fmt5;"TYPE","STATUS","PROFILE","CONSUMED","RESUPPLY","ON-TRK".
"ON-GND","DISPND","ON-VEH","CURRNT","USD/DT"
4140 Fmt5:IMAGE 4A,1X,6A,1X,7A,1X,8A,1X,8A,6(1X,6A)
4150  !
4160  Df_used=N(116)-N(131)
4170  If_used=N(117)-N(132)
4180  Ad_used=N(118)-N(133)
4190  Tdf_used=Tdf_used+Df_used
4200  Tif_used=Tif_used+If_used
4210  Tad_used=Tad_used+Ad_used
4220  !
4230  N135_sv=N(135)
4240  GOSUB Update_ammo
4250  !
4260  !  FILL IN SUPPLY STATUS MATRICES
4270  Df_stat(1)=INT(N(136))/1000*N135_sv
4280  Df_stat(2)=INT(N(124))/1000*N(123)
4290  Df_stat(3)=INT(N(126))/1000*N(125)
4300  Df_stat(4)=INT(N(138))/1000*N(137)
4310  !
4320  If_stat(1)=(N(136)-INT(N(136)))*N135_sv
4330  If_stat(2)=(N(124)-INT(N(124)))*N(123)
4340  If_stat(3)=(N(126)-INT(N(126)))*N(125)
4350  If_stat(4)=(N(138)-INT(N(138)))*N(137)
4360  !
4370  Ad_stat(1)=N135_sv-(Df_stat(1)+If_stat(1))
4380  Ad_stat(2)=N(123)-(Df_stat(2)+If_stat(2))
4390  Ad_stat(3)=N(125)-(Df_stat(3)+If_stat(3))
4400  Ad_stat(4)=N(137)-(Df_stat(4)+If_stat(4))
4410  !
4420  GOSUB Veh_ammo
4430  C_fuel=T_fuel+N(105)+N(103)
4440  Totalfuel=Totalfuel+N(108)
4450  Currentfuel=Currentfuel+C_fuel
4460  !
4470  Tdf=Df_stat(2)+Df_stat(3)+Tot_ammo(1)
4480  Tif=If_stat(2)+If_stat(3)+Tot_ammo(2)
4490  Tad=Ad_stat(2)+Ad_stat(3)+Tot_ammo(3)
4500  Totaldf=Totaldf+Tdf
4510  Totalif=Totalif+Tif
4520  Totalad=Totalad+Tad
4530  Tdf_used=Tdf_used-Df_stat(4)
4540  Tif_used=Tif_used-If_stat(4)    !TEST
4550  Tad_used=Tad_used-Ad_stat(4)
4560  !
4570  Gnd_ammo=Df_stat(3)+If_stat(3)+Ad_stat(3)
```

Table 10-2. Unit status report code (continued).

```
4580  Ftrk_cap=N(55)*Truck_cap(1)
4590  Ft4=0
4600  IF Tk_emp<0 THEN Tk_emp=0
4610  IF N(103)<N(55)*Truck_cap(1) THEN
4620    Ft1=N(103)/(N(55)*Truck_cap(1))
4630    Ft2=N(55)*(1-Ft1)
4640    Ft4=INT(Ft2)
4650  END IF
4660  PRINT USING Fmt6;"DF",N(119),N(127),Df_used,Df_stat(*),Tot_ammo(1).Tdf.N(
39)
4670  PRINT USING Fmt6;"IF",N(120),N(128),If_used,If_stat(*),Tot_ammo(2),Tif,N(
40)
4680  PRINT USING Fmt6;"AD",N(121).N(129),Ad_used,Ad_stat(*).Tot_ammo(3),Tad.N(
41)
4690  PRINT USING Fmt6;"FU",N(101),N(107),N(108),N(110),N(103),N(105),N(112).T_
uel,C_fuel,N(143)
4700 Fmt6:IMAGE 1X,2A,1X,4D.2D,1X,7D,1X,8D,1X,8D,6(1X,6D)
4710  !
4720  PRINT
4730  Tk_emp=INT(Tk_emp)
4740  PRINT "EMPTY AMMO TRUCKS ";Tk_emp;"      EMPTY FUEL TRUCKS ";Ft4
4750 !PRINT " EQUIVALENT EMPTY FUEL TRUCKS ";Ft4
4760 !PRINT USING "///"
4770  N(110)=0
4780  N(135)=0
4790  N(136)=0
4800  N(112)=0
4810  N(137)=0
4820  N(138)=0
4830 End_stat_print:RETURN
4840  !
4850  ! ***********************************************************************
4860  !
4870 Print_kvfile:  !  THIS SBR PRINTS OUT THE SUMMARY KV FILE
4880  !
4890  GOSUB Zero_list
4900  FOR I=1 TO 8
4910    FOR J=1 TO 70
4920      Kv(I,J)=Ci_kv_b(I.J)
4930      Kv_tot(J)=Kv_tot(J)+Ci_kv_b(I,J)
4940    NEXT J
4950  NEXT I
4960  Veh$=B_veh$
4970  PRINT USING "@,38A.1A.7A.3A.2X.4A.1X,7A./";"RED KILLER--BLUE VICTIM FILE
OR GAME ",Game$,"  TURN ",Turn$,"TIME",T$
4980  GOSUB List_kv
4990  !
5000  ! PRINT OUT RED KV LIST
5010  GOSUB Zero_list
5020  FOR I=1 TO 8
5030    FOR J=1 TO 70
5040      Kv(I,J)=Ci_kv_r(I.J)
```

Table 10-2. Unit status report code (continued).

```
5050      Kv_tot(J)=Kv_tot(J)+Ci_kv_r(I,J)
5060    NEXT J
5070  NEXT I
5080  Veh$=R_veh$
5090  PRINT USING "@,38A,1A,7A,3A,2X,4A,1X,7A,/";"BLUE KILLER--RED VICTIM FILE F
OR GAME ",Game$,"  TURN ",Turn$,"TIME",T$
5100  GOSUB List_kv
5110  !
5120  RETURN
5130  !
5140  ! ***********************************************************************
5150  !
5160 Zero_list:  ! ZERO VARIABLES FOR KV PRINT
5170  !
5180  FOR I=1 TO 8
5190    FOR J=1 TO 70
5200      Kv(I,J)=0
5210      Kv_tot(J)=0
5220    NEXT J
5230  NEXT I
5240  !
5250  RETURN
5260  !
5270  ! ***********************************************************************
5280  !
5290 List_kv:  !  THIS SBR PRINTS THE KV TABLE
5300  PRINT USING "//,6A,13X,49A";"VICTIM","<------------------ KILLER ---------
------------>"
5310  PRINT USING Fmt10;"          ","D/F","I/F","FGM","A/H","INF","MIN","CHM","AI
R","T/KILL"
5320 Fmt10:IMAGE /,8A,3X,7(3A,5X),3A,4X,6A,/
5330  FOR I=1 TO 70
5340    PRINT USING Fmt11;Veh$[(I-1)*5+1,I*5],Kv(1,I),Kv(2,I),Kv(3,I),Kv(4,I),Kv
(5,I),Kv(6,I),Kv(7,I),Kv(8,I),Kv_tot(I)
5350    IF I=55 THEN
5360      PRINT USING "@"
5370    END IF
5380  NEXT I
5390 Fmt11:IMAGE 5A,4X,8(4D.1D,2X),1X,4D.1D
5400  !
5410  RETURN
5420  !
5430  ! ***********************************************************************
5440  !
5450 Print_helo_file:  !  THIS SBR PRINTS OUT BLUE HELICOPTER DATA
5460  !
5470  ENTER @Phelo,1;Ci_helo_b(*)
5480  ENTER @Phelo,2;Ci_helo_r(*)
5490  PRINT USING "@,#,27A";"ATTACK HELICOPTER RESULTS"
5500  PRINT USING Fmt32;"TYPE","#KILLED","#SORTIES"
5510  PRINT USING Fmt33;"LCH ",Ci_helo_b(1,2),Ci_helo_b(1,6)
5520  PRINT USING Fmt33;"AH-1",Ci_helo_b(2,2),Ci_helo_b(2,6)
```

Table 10-2. Unit status report code (continued).

```
5530  PRINT USING Fmt33;"OH58",Ci_helo_b(3,2),Ci_helo_b(3,6)
5540  PRINT USING "//"
5550  PRINT USING Fmt33;"HIP",Ci_helo_r(2,2),Ci_helo_r(2,6)
5560  PRINT USING Fmt33;"HIND",Ci_helo_r(1,2),Ci_helo_r(1,6)
5570 Fmt32:IMAGE ///,4A,4X,7A,3(4X,8A),//
5580 Fmt33:IMAGE 4A, 5X,3(3D.1D,7X),5D
5590  !
5600  RETURN
5610  !
5620  ! ***********************************************************************
5630  !
5640 Print_work_sht:  ! THIS SBR PRINTS GAMER WORK SHEETS
5650  !
5660 !IF I=192 THEN
5670 !  PRINT USING "@,#,32A,2D";"GAMER STAFF WORK  SHEET FOR TURN ",T1
5680 !  PRINT USING "/,13A";"RED UNITS:"
5690 !END IF
5700  IF N(82)=0 OR N(82)=2 THEN End_print
5710 !PRINT USING Fmt20;"UNIT ",I,M$,"TYPE: ",N(78)
5720 !Fmt20:IMAGE ///,5A,3D,5X,8A,4X,6A,1D.1D
5730  PRINT USING Fmt21;"LINE 1:  ACTIVITY: ",N(82),"MOPP LEVEL: ",N(77),"MISSI
N: ",N(83),"KM MOVED: ",N(146)
5740 Fmt21:IMAGE /,19A,1D,7X,12A,1D,6X,9A,1D.1D,4X,10A,3D
5750  N2=INT(N(89))
5760  N1=INT((N(89)-N2)*10+.5)
5770  N3=INT(N(90)*100)
5780  PRINT USING Fmt22;"LINE 2:  SENSOR GP: ",N1,"ZONE: ",N2,"PCT COVERED: ",N
5790 Fmt22:IMAGE /,20A,1D,8X,6A,1D,10X,13A,3D
5800  N1=INT(N(80))
5810  N2=INT(((N(80)-N1)*100))
5820  PRINT USING Fmt23;"LINE 3:  PCT ADA SUPPRESSION:  VEH: ",N1,"HAND: ",N2,"
ORPS ADA: ",N(81)
5830 Fmt23:IMAGE /,36A,3D,7X,6A,3D,8X,11A,3D
5840 !
5850  PRINT
5860  PRINT "LINE 4:  RESUPPLY: DF(Tons)____IF(Tons)____AD(Tons)____ FUEL(gal):
__"
5870  PRINT
5880  PRINT "LINE 5:  DISPENSED: DF(Tons)____IF(Tons)____AD(Tons)____ FUEL:(ga
__"
5890  PRINT
5900  IF I<192 THEN
5910    PRINT "LINE 6: GDRADAR____";N(95);"ARTRADAR____";N(96);"LRRP____";N(97
"RPV____";N(98);"SLAR____";N(99);"FO____";N(100)
5920  ELSE
5930    PRINT "LINE 6: GDRADAR____";N(95);"ARTRADAR____";N(96);"LRRP____";N(97
"SLAR____";N(98);"RPV____";N(99);"FO____";N(100)
5940  END IF
5950  PRINT
5960  PRINT "CURRENT LOCATION _______________     PROPOSED LOCATION __________
___"
5970  PRINT
```

Table 10-2. Unit status report code (continued).

```
5980  PRINT "ACTUAL LOCATION ARRIVED AT _______________"
5990  PRINT
6000  PRINT "ACTION TO BE TAKEN AT THE OBJECTIVE:"
6010  PRINT
6020  PRINT "_______________________________________________________________________"
6030 End_print:  !
6040  RETURN
6050  !
6060  ! ************************************************************************
6070  !
6080 Blue_data:  !
6090  !
6091  !   ** DC **
6092  !
6100  ASSIGN @Pwpntyp TO "WPN_TYP"&Dcdisk$
6110  ENTER @Pwpntyp,1;Wpn_type(*)
6120  ASSIGN @Pwpntyp TO *
6130  !
6180  ASSIGN @Pammocap TO "AMMO_CAP"&Dcdisk$
6190  ENTER @Pammocap,1;Ammo_cap(*)
6200  ASSIGN @Pammocap TO *
6210  !
6270  ASSIGN @Pfuelcap TO "FUEL_CAP"&Dcdisk$
6280  ENTER @Pfuelcap,1;Fuel_cap(*)
6290  ASSIGN @Pfuelcap TO *
6300  !
6310  ASSIGN @Ptrkcap TO "BL_TRK_CAP"&Dcdisk$
6320  ENTER @Ptrkcap,1;Truck_cap(*)
6330  ASSIGN @Ptrkcap TO *
6360  !Truck_cap(4)=6.5
6370  !Truck_cap(1)=1800
6380  RETURN
6390 !
6400 !***************************************************************************
6410 !
6420 Red_data: !
6430  ASSIGN @Pwpntyp TO "WPN_TYP"&Dcdisk$
6440  ENTER @Pwpntyp,1;Dummyrec(*),Wpn_type(*)
6441  ASSIGN @Pwpntyp TO *
6450  !
6520  ASSIGN @Pammocap TO "AMMO_CAP"&Dcdisk$
6530  ENTER @Pammocap,1;Dummyrec(*),Ammo_cap(*)
6540  ASSIGN @Pammocap TO *
6550  !
6610  ASSIGN @Pfuelcap TO "FUEL_CAP"&Dcdisk$
6620  ENTER @Pfuelcap,1;Dummyrec(*),Fuel_cap(*)
6630  ASSIGN @Pfuelcap TO *
6640  !
6650  ASSIGN @Ptrkcap TO "RD_TRK_CAP"&Dcdisk$
6660  ENTER @Ptrkcap,1;Truck_cap(*)
6670  ASSIGN @Ptrkcap TO *
6700  !Truck_cap(4)=4.5
```

Table 10-2. Unit status report code (continued).

```
6710  !Truck_cap(1)=1288
6711  !   ** END DC **
6720  RETURN
6730 !
6740 !*************************************************************************
6750 !
6760 Update_ammo: !CALCULATE TOTAL CAPACITY
6770  FOR Ikp=1 TO 3
6780    Tot_veh_ammo(Ikp)=0
6790  NEXT Ikp
6800  FOR J=1 TO 70
6810    Tot_veh_ammo(Wpn_type(J))=Tot_veh_ammo(Wpn_type(J))+N(J)*Ammo_cap(J)
6820  NEXT J
6830  N(125)=0
6840  FOR Ikp=1 TO 3
6850    IF N(130+Ikp)<0 THEN N(130+Ikp)=0
6860    IF Tot_veh_ammo(Ikp)<=0 THEN
6870      Ammo_left(Ikp)=N(130+Ikp)
6880      N(118+Ikp)=0
6890      N(130+Ikp)=0
6900      GOTO Ecl
6910    END IF
6920    IF N(130+Ikp)>Tot_veh_ammo(Ikp) THEN
6930      N(118+Ikp)=1
6940      Ammo_left(Ikp)=N(130+Ikp)-Tot_veh_ammo(Ikp)
6950      N(130+Ikp)=0
6960    ELSE
6970      N(118+Ikp)=N(130+Ikp)/Tot_veh_ammo(Ikp)
6980      N(130+Ikp)=0
6990      Ammo_left(Ikp)=0
7000    END IF
7010 Ecl:N(125)=N(125)+Ammo_left(Ikp)
7020  NEXT Ikp
7030  !
7040  IF N(125)=0 THEN
7050    N(126)=0
7060  ELSE
7070    Pdf=Ammo_left(1)/N(125)
7080    Pif=Ammo_left(2)/N(125)
7090    N(126)=INT(Pdf*1000)+Pif
7100  END IF
7110  ! CALC PROPER TRUCK CAPACITY
7120  Atrk_cap=N(58)*Truck_cap(4)
7130  ! TAKE AMMO FROM PILES IN SAME RATIO AS AVAILABLE TRUCKS
7140  Tk_emp=0
7150  ! ZERO AMOUNT IN TRUCKS
7160  FOR Ikp=1 TO 3
7170    Tk_ammo(Ikp)=0
7180  NEXT Ikp
7190  N(123)=0
7200  IF N(125)<=0 OR Atrk_cap<=0 THEN No_trks
7210  ! AMMO IS ON GROUND AND TRUCKS HAVE A CAPACITY LOAD ON EMPTY TRUCKS
```

Table 10-2. Unit status report code (concluded).

```
7220  Tk_perc(1)=INT(N(126))/1000
7230  Tk_perc(2)=N(126)-INT(N(126))
7240  Tk_perc(3)=1-(Tk_perc(1)+Tk_perc(2))
7250  IF Atrk_cap<N(125) THEN
7260    Tk_ammo(1)=Atrk_cap*Tk_perc(1)
7270    Tk_ammo(2)=Atrk_cap*Tk_perc(2)
7280    Tk_ammo(3)=Atrk_cap*Tk_perc(3)
7290    N(123)=Atrk_cap
7300    N(124)=N(126)
7310    N(125)=N(125)-N(123)
7320    Tk_emp=0
7330  ELSE
7340       !MORE TRUCKS THAN ON GROUND AVAILABLE
7350    Tk_ammo(1)=N(125)*Tk_perc(1)
7360    Tk_ammo(2)=N(125)*Tk_perc(2)
7370    Tk_ammo(3)=N(125)*Tk_perc(3)
7380    N(123)=Tk_ammo(1)+Tk_ammo(2)+Tk_ammo(3)
7390    N(124)=N(126)
7400    Tk_emp=(Atrk_cap-N(125))/Truck_cap(4)
7410    N(125)=0
7420  END IF
7430  N(131)=0
7440  N(132)=0
7450  N(133)=0
7460  N(135)=0
7470 No_trks: !
7480 !IF Ammog=0 THEN N(125)=0
7490  IF N(146)>0 THEN N(125)=0
7500  IF N(146)>0 THEN N(105)=0
7510 !IF Fuelg=0 THEN N(105)=0
7520  RETURN
7530 !
7540 !*************************************************************************
7550 !
7560 Veh_ammo: !
7570  FOR J=1 TO 3
7580    Tot_ammo(J)=0
7590  NEXT J
7600  T_fuel=0
7610  FOR J=1 TO 70
7620    Tot_ammo(Wpn_type(J))=Tot_ammo(Wpn_type(J))+Ammo_cap(J)*N(J)*N(118+Wpn_t
ype(J))
7630    T_fuel=T_fuel+N(J)*Fuel_cap(J)*N(101)
7640  NEXT J
7650  RETURN
7660 !
7670 !*************************************************************************
7680 !
7690 Halt: !
7700  LOAD "DIME"&Disk$
7710  END
```

CHAPTER 11

UTILITY ROUTINES

1. PURPOSE.

DIME utility routines are available to assist in the creation of needed files, in listing files and in some cases, changing data values.

2. CREATE ROUTINES.

The following routines are used preceding the use of game initialization (P1). These are necessary before any DIME game operations may be done.

A. "CR_NAME". Initially creates and blanks the unit name file, "NAMEFILE".

B. "CR_TOE". Initially creates and zeros the table of organization/ equipment (TOE) file, "TOEFILE".

C. "CR_UNIT". Initially creates and zeros the unit status file, "UNITFILE".

3. "UNITFILE" UTILITIES.

The following routines may be used to list and check the "UNITFILE", "NAMEFILE" and "TOEFILE" after they have been built in P1.

A. "NAMEDUMP". This routine lists the names ("NAMEFILE") of the 400 possible units created.

B. "STRENGTH". This routine lists the total of each 21 systems in the 191 Blue units and 209 Red units. These totals are listed from both the "UNITFILE" and "TOEFILE".

C. "EFFECTIVE". This routine recalculates effectiveness totals and percentages. The internal system effectiveness data may be changed and recalculated in both the "TOEFILE" and "UNITFILE".

4. CREATE/ZEROING ROUTINES.

These routines may be used when initially creating the game's cumulative kill files, when zeroing is necessary before the beginning of a game or when the unit status report (P8) is not run to completion. The following routines create and/or zero files which contain accumulated kills for one complete game turn.

A. "CR_KV". Creates and/or zeros the killer/victim file, "KVFILE".

B. "CR_HELO". Creates and/or zeros the helicopter file, "HELOFILE".

C. "AIRPLANE". Creates and/or zeros the air defense victim files. This is the same routine which is used to create the entire air defense (P3) data base. The two files consist of:

(1) The "RAIR_INF" contains the number of Blue victims.

(2) The "BAIR_INF" contains the number of Red victims.

D. "CR_CHEM". Creates and/or zeros both chemical files which consist of:

(1) "BLCHMVCTM" which contains the number of Blue victims.

(2) "RDCHMVCTM" which contains the number of Red victims.

5. "OMNI" DATA FILE UTILITIES

The following routines are primarily used to alter the data values in files used by DIME. In addition, they may be used to create new data files and/or list the values in these files. All "OMNI" routines may be executed from the "OMNI" menu. The routines are described below.

A. <u>Ground Combat Files.</u>

(1) "OMNI_ECF" processes the expected number of completed firing files.

(2) "OMNI_SSKP" processes the probability of kill files.

(3) "OMNI_CAT" processes the pointer to the P(K) files.

(4) "OMNI_AMMO" processes the ammunition weight files.

(5) "OMNI_FIRE" processes the fire distribution files.

B. <u>Helicopter Files</u>

(1) "HELOCREATE" creates helicopter files.

(2) "HELOTGTPRG" creates helicopter target preference files.

(3) "HELOFORM" lists helicopter files.

C. Main Driver Files.

(1) "OPERATION" processes operational mission template files.

(2) "AMMO"

D. Artillery Files

(1) "LETHALAREA" processes lethal area files.

(2) "FIREDELVRY" processes fire delivery files.

E. Infantry Files.

(1) "FIREPOWER" processes firepower scores files.

F. Projectile Guided Missile (PGM) Files.

(1) "PGMDATAPRG" processes PGM files.

G. Air Attack/Air Defense Files.

(1) "AIRPLANE" processes the following:

(a) Air ingress/egress profile files.

(b) Air strike profile files.

(c) Unit area template files.

(d) Weapon load template files.

(e) Air mission/target priority files.

(f) Initializes air and target loss files.

H. Declassification Files. In previous versions of the DIME model, data was contained in data statements. These data are now contained in data files. The following utility programs are used to process these files.

(1) "AMMOCAPdc" processes ammunition capacity files.

(2) "FUELCAPdc" processes fuel capacity files.

(3) "SYSEFFdc" processes system effectiveness files.

(4) "WPNTYPdc" processes weapon type files.

(5) "AMMOUSEdc" processes ammunition use files.

(6) "FUELUSEdc" processes fuel use files.

(7) "TRUCKCAPdc" processes truck capacity files.

(8) "TOTPLOSSdc" processes total probability of loss files.

(9) "FRACDISMdc" processes dismounted fraction files.

(10) "AMMOWTdc" processes ammunition weight files.

(11) "BASICLDdc" processes basic load files.

(12) "ARTYRATEdc" processes artillery rate files.

(13) "ARTYWTdc" processes artillery weight files.

(14) "FMASKdc" processes indirect fire mask files.

(15) "DFMASKdc" processes direct fire mask files.

(16) "DSSTARTdc" processes start range for direct fire files.

(17) "ARTYALOCdc" processes artillery allocation files.

(18) "MINEFRCTdc" processes mine fraction files.

(19) "CONVERTDdc" processes defender loss coefficient files.

(20) "CONVERTAdc" processes attacker loss coefficient files.

(21) "AREABANDdc" processes area band files.

(22) "DISPMASKdc" processes dispersion mask files.

(23) "TGTMASKdc" processes target mask files.

(24) "PSNLPOSTdc" processes personnel pasture files.

(25) "TLEdc" processes target location error files.

(26) "ROUNDWTdc" processes round weight files.

(27) "AMWTPPdc" processes ammunition weight per round files.

(28) "IROFdc" processes integer rate of fire files.

(29) "SENSORdc" processes sensor files.

(30) "TGTVALSdc" processes target values files.

(31) "SSKPdc" processes single shot kill probability files.

(32) "TGTMASK1dc" processes target mask files.

(33) "DUSTABRTdc" processes probability of dust abort files.

(34) "CLGPMASKdc" processes cannon loaded guided projectile files.

(35) "PROBDESGdc" processes probability designator files.

(36) "SSKPCLGPdc" processes single shot kill probabilities for cannon loaded guided projectile files.

6. OFF-LINE LOSS ASSESSMENT (P6).

A. This routine, available through the DIME menu, may be used to create losses to units separate from the attrition calculations of air attack/air defense (P3), ground combat (P4), and chemical (P5).

B. In order to cause attrition through this off-line loss assessment, the following items must be identified:

(1) The unit number being assessed.

(2) The killing category (direct fire, indirect fire, precision guided munitions, attack helicopters, infantry, mines, chemical, and air attack/air defense).

(3) The system number (1-70).

(4) The amount of losses to the particular system in a specific unit.

C. Included in the loss with the systems, ammunition and fuel losses are calculated for the systems.

7. CODE.

Listings for the utility routines (except OMNI) may be found in Tables 11-1 through 11-10. A listing of the P6 code appears in Table 11-11.

Table 11-1. "CR_NAME" program.

```
10      !
20      !  PROGRAM : CR_NAME                                    DIME V5.0 - MAR 1986
30      !
40         OPTION BASE 1
50         DIM M$[16]
60         PRINT CHR$(12)
70         REPEAT
80             INPUT "(CR)CREATE OR (IN)INITIALIZE 'NAMEFILE'?",A$
90         UNTIL A$="CR" OR A$="IN"
100        IF A$="CR" THEN
110            CREATE BDAT "NAMEFILE:HP9134,701,0",400,20
120            PRINT TABXY(2,2);"NAMEFILE CREATED."
130        END IF
140        PRINT TABXY(2,4);"INITIALIZING NAMEFILE (400 RECORDS)"
150        ASSIGN @Pname TO "NAMEFILE:HP9134,701,0"
160        M$="UNUSED         "
170        FOR Recnum=1 TO 400
180            OUTPUT @Pname,Recnum;M$
190            IF Recnum MOD 10=0 THEN PRINT TABXY(2,7);Recnum;" . . . "
200        NEXT Recnum
210        PRINT TABXY(2,10);"NAMEFILE INITIALIZED."
220        ASSIGN @Pname TO *
230        PRINT TABXY(2,17);CHR$(130);"INSERT";CHR$(128);" OMNI MENU DISK. THEN P
RESS ENTER..."
240        INPUT "",A$
250        LOAD "OMNI_MENU"
260        END!
```

Table 11-2. "CR_TOE program.

```
10      '
20      '  PROGRAM : CR_TOE                                  DIME V5.0 - MAR 1986
30      '
40        OPTION BASE 1
50        DIM U(72)
60        PRINT CHR$(12)
70        REPEAT
80            INPUT "(CR)CREATE OR (IN)INITIALIZE 'TOEFILE'?",A$
90        UNTIL A$="CR" OR A$="IN"
100       IF A$="CR" THEN
110           CREATE BDAT "TOEFILE:HP9134,701,0",400,576
120           PRINT TABXY(2,2);"TOEFILE CREATED. "
130       END IF
140       PRINT TABXY(2,4);"INITIALIZING TOEFILE (400 RECORDS)"
150       ASSIGN @Ptoe TO "TOEFILE:HP9134,701,0"
160       FOR I=1 TO 72
170           U(I)=0
180       NEXT I
190       FOR Recnum=1 TO 400
200           OUTPUT @Ptoe,Recnum;U(*)
210           IF Recnum MOD 10=0 THEN PRINT TABXY(2,7);Recnum;" . . . "
220       NEXT Recnum
230       PRINT TABXY(2,10);"TOEFILE INITIALIZED."
240       ASSIGN @Ptoe TO *
250       PRINT TABXY(2,17);CHR$(130);"INSERT";CHR$(128);" OMNI MENU DIS. THEN
RESS ENTER..."
260       INPUT "",A$
270       LOAD "OMNI_MENU"
280       END!
```

Table 11-3. CR_UNIT program.

```
10     !
20     !  PROGRAM : CR_UNIT                                   DIME V5.0 - MAR 1986
30     !
40        OPTION BASE 1
50        DIM N(150)
60        PRINT CHR$(12)
70        REPEAT
80            INPUT "(CR)CREATE OR (IN)INITIALIZE 'UNITFILE'?",A$
90        UNTIL A$="CR" OR A$="IN"
100       IF A$="CR" THEN
110           CREATE BDAT "UNITFILE:HP9134,701,0",400,1200
120           PRINT TABXY(2,2);"UNITFILE CREATED. "
130       END IF
140       PRINT TABXY(2,4);"INITIALIZING UNITFILE (400 RECORDS)"
150       ASSIGN @Punit TO "UNITFILE:HP9134,701,0"
160       FOR I=1 TO 150
170           N(I)=0
180       NEXT I
190       FOR Recnum=1 TO 400
200           OUTPUT @Punit,Recnum;N(*)
210           IF Recnum MOD 10=0 THEN PRINT TABXY(2,7);Recnum;" . . . "
220       NEXT Recnum
230       PRINT TABXY(2,10);"UNITFILE INITIALIZED."
240       ASSIGN @Punit TO *
250       PRINT TABXY(2,17);CHR$(130);"INSERT";CHR$(128);" OMNI MENU DISK. THEN P
RESS ENTER..."
260       INPUT "",A$
270       LOAD "OMNI_MENU"
280       END!
```

Table 11-4. "NAMEDUMP" program.

```
10      !
20      !   PROGRAM : NAMEDUMP                              DIME V5.0 - MAR 1986
30      !
40          OPTION BASE 1
50          DIM A$(400)[16]
60          PRINT CHR$(12)
70          PRINT TABXY(2,2);"READING NAMEFILE"
80          ASSIGN @Pname TO "NAMEFILE:HP9134,701,0"
90          FOR Recnum=1 TO 400
100             ENTER @Pname,Recnum;A$(Recnum)
110         NEXT Recnum
120         ASSIGN @Pname TO *
130         !
140         PRINT TABXY(2,4);"PRINTING NAMEFILE"
150         PRINTER IS 702
160         !
170         PRINT CHR$(10)
180         PRINT TAB(10);"NAMEFILE VALUES (400 RECORDS) :"
190         PRINT CHR$(10)
200         FOR I=1 TO 400
210             IF I MOD 50=1 AND I>1 THEN
220                 PRINT CHR$(12)
230                 PRINT CHR$(10)
240                 PRINT TAB(10);"NAMEFILE VALUES (continued) "
250                 PRINT CHR$(10)
260             END IF
270             PRINT TAB(15);
280             PRINT USING "DDD,4A,#";I;"   - "
290             PRINT A$(I)
300         NEXT I
310         PRINT CHR$(12)
320         PRINTER IS 1
330         PRINT TABXY(2,10);" DONE PRINTING."
340         STOP
350         END'
```

Table 11-5. "STRENGTH" program.

```
10      !
20      !  PROGRAM : STRENGTH                                        DIME 5.0 - 1986
30      !
40      OPTION BASE 1
50      DIM U(150),T(72),W(70),N(70),D(70),Sys$[350],Rsys$[350],T$[7]
60      Sys$[1,125]="DF   FAV-TM551 FAV40HMV-GDF   DRAGNLAW  DF   CMD-VDF   DF
F   DF   DF   HMV40DF-ICDF-ICDF-ICDF-ICARTY ARTY ARTY ARTY ARTY "
70      Sys$[126,250]="ARTY ARTY MORTRMORTRMORTRMORTRMLRSTMLRSTMLRSTMLRSTINF  INF
 INF  INF  INF  SARMSSARMSSARMSSARMSSARMSSARMSSARMSVULCNAVNGRIHAWK"
80      Sys$[251,350]="ADA  ADA  STINGADAHHF-TRKJ4TRKWATERCGO-TNATRKEWTRKEWTRKENG
 OBSCEAVLB PONBRENGEQENGEQMATHEMATHEAATHE"
90      Rsys$[1,125]="T55  DF   BMP73DF   BRDM3BRDMSAT-75AGS17T12  CMD-VDF   DF
DF   DF   DF   BMPATBTR  DF-ICDF-ICDF-ICARTY ARTY ARTY ARTY ARTY "
100     Rsys$[126,250]="ARTY ARTY MORTRMORTRMORTRMORTRMRL  MRL  MRL  MRL  INF  INF
  INF  INF  INF  SARMSSARMSSARMSSARMSSARMSSARMSSARMSZSU-XSA-13SA-6 "
110     Rsys$[251,350]="ADA  ADA  SA-14ADAHHF-TRKJ4TRKWATERCGO-TNATRKEWTRKEWTRKEN
R OBSCEAVLB PONBRENGEQENGEQMATHEMATHEAATHE"
120     ASSIGN @Pu TO "UNITFILE:HP9134,701"
130     ASSIGN @Pt TO "TOEFILE:HP9134,701"
140     INPUT "ENTER GAME TIME",T$
150     PRINTER IS 702
160     RESTORE 180
170     READ W(*)
180     DATA -,-,-,-,-,-,-,-,-,-
190     DATA -,-,-,-,-,-,-,-,-,-
200     DATA -,-,-,-,-,-,-,-,-,-
210     DATA -,-,-,-,-,-,-,-,-,-
220     DATA -,-,-,-,-,-,-,-,-,-
230     DATA -,-,-,-,-,-,-,-,-,-
240     DATA -,-,-,-,-,-,-,-,-,-                                      !BLUE
250     Side$="BLUE"
260     GOSUB Prt_pg_hdrs
270     FOR I=1 TO 191
280       ENTER @Pu,I;U(*)
290       ENTER @Pt,I;T(*)
300       IF U(82)=0 THEN 350
310       FOR J=1 TO 70
320         D(J)=D(J)+T(J)
330         IF U(79)>.3999 THEN N(J)=N(J)+U(J)
340       NEXT J
350     NEXT I
360     FOR J=1 TO 70
370       IF J=55 THEN GOSUB Prt_pg_hdrs
380       Num=Num+N(J)*W(J)
390       Den=Den+D(J)*W(J)
400       PRINT USING "20X,3D,5X,5A,5X,7D,5X,7D";J,Sys$[(J-1)*5+1,J*5],N(J),D(J)
410       N(J)=0
420       D(J)=0
430     NEXT J
440     IF Den<>0 THEN Beff=Num/Den
450     X=Num
460     PRINT USING "18X,15A,3X,9D,3X,9D";"WEIGHTED TOTALS",Num,Den
```

Table 11-5. "STRENGTH" program.

```
470   PRINT USING "18A,3X,D.3D":"BLUE-EFFECTIVENESS".Beff
480   Side$=" RED"
490   GOSUB Prt_pg_hdrs
500   Num=0
510   Den=0
520   RESTORE 540
530   READ W(*)
540   DATA -,-,-,-,-,-,-,-,-,-
550   DATA -,-,-,-,-,-,-,-,-,-
560   DATA -,-,-,-,-,-,-,-,-,-
570   DATA -,-,-,-,-,-,-,-,-,-
580   DATA -,-,-,-,-,-,-,-,-,-
590   DATA -,-,-,-,-,-,-,-,-,-
600   DATA -,-,-,-,-,-,-,-,-,-                                  'RED
610   FOR I=192 TO 400
620     ENTER @Pu,I;U(*)
630     ENTER @Pt,I;T(*)
640     IF U(82)=0 THEN 690
650     FOR J=1 TO 70
660       D(J)=D(J)+T(J)
670       IF U(79)>.3999 THEN N(J)=N(J)+U(J)
680     NEXT J
690   NEXT I
700   FOR J=1 TO 70
710     IF J=55 THEN GOSUB Prt_pg_hdrs
720     Num=Num+N(J)*W(J)
730     Den=Den+D(J)*W(J)
740     PRINT USING "20X,3D,5X,5A,5X,7D,5X,7D";J,Rsys$[(J-1)*5+1,J*5],N(J),[
750   NEXT J
760   PRINT USING "18X,15A,3X,9D,3X,9D";"WEIGHTED TOTALS",Num,Den
770   IF Den<>0 THEN Reff=Num/Den
780   Y=Num
790   PRINT USING "18A,3X,D.3D";" RED-EFFECTIVENESS",Reff
800   PRINT
810   PRINT USING "31A,2D.2D";"CURRENT FORCE RATIO RED:BLUE = ";Y/X;":1"
820   PRINTER IS 1
830   PRINT "DO YOU WANT TO RUN 1) EFFECTIVE?"
840   PRINT "                   2) ARMY?"
850   PRINT "                   3) DIME?"
860   INPUT X
870   IF X=1 THEN LOAD "EFFECTIVE:HP9134,701"
880   IF X=2 THEN LOAD "ARMY:HP9134,701"
890   IF X=3 THEN LOAD "DIME:HP9134,701"
900 Prt_pg_hdrs:   !
910   PRINT USING "@,#"
920   PRINT USING "20X,30A,7A";"STRENGTH REPORT FOR GAME TIME ",T$
930   PRINT USING "//,38X,6A,7X,5A";"ACTIVE","TOTAL"
940   PRINT USING "4A,1X,9A,6X,3A,6X,5A,3X,8A,3X,7A";Side$,"STRENGTH:","SYS"
ME","UNITFILE","TOEFILE"
950   RETURN
960   END
```

Table 11-6. "EFFECTIVE" program.

```
1      !
2      !  PROGRAM : EFFECTIVE                                  DIME 5.0 - MAR 1986
3      !
10     OPTION BASE 1
20     ! THIS PROGRAM WILL LOOK AT THE UNIT'S EFFECTIVENESS AND PRINT
30     ! OUT THE PERCENTAGE.
40     DIM U(150),T(72),M$[16]
50     ASSIGN @Pu TO "UNITFILE:HP9134,701"
60     ASSIGN @Pt TO "TOEFILE:HP9134,701"
70     ASSIGN @Pname TO "NAMEFILE:HP9134,701"
80     INPUT "WHAT IS THE GAME TIME ",X
90     INPUT "PRINT TO SCREEN OR PRINTER(S/P)",S$
100    PRINTER IS 1
110    IF S$="P" THEN
120      PRINTER IS 702
130      PRINT USING "@,#"
140      GOTO 170
150    END IF
160    IF S$<>"S" THEN GOTO 90
170    PRINT "******************** BLUE UNIT EFFECTIVENESS ********** TIME ";X;"*"
180    PRINT "   "
190    PRINT "          ACTIVE                                                 INACTIVE"
200    PRINT "   "
210    FOR I=1 TO 191
220      ENTER @Pu,I;U(*)
230      ENTER @Pt,I;T(*)
240      ENTER @Pname,I;M$
250      IF M$="UNUSED" OR M$="                " THEN 440
260      IF U(82)=1 OR U(82)=2 THEN
270        U_790=DROUND(U(79),2)
280        IF U_790>=.8 THEN
290          PRINT USING "3D,5X,16A,3X,D.2D";I,M$,U_790
300          GOTO 440
310        END IF
320        IF U_790>=.6 THEN
330          PRINT USING "3D,5X,16A,10X,D.2D";I,M$,U_790
340          GOTO 440
350        END IF
360        IF U_790>=.4 THEN
370          PRINT USING "3D,5X,16A,20X,D.2D";I,M$,U_790
380          GOTO 440
390        END IF
400        IF U_790<.4 THEN
410          PRINT USING "3D,5X,16A,30X,D.2D,3A";I,M$,U_790,"***"
420        END IF
430      END IF
440    NEXT I
450    PRINT "   "
460    PRINT CHR$(12)
470    PRINT "   "
480    PRINT "******************** RED UNIT EFFECTIVENESS ********** TIME ";X;"*"
490    PRINT "   "
```

Table 11-6. "EFFECTIVE" program.

```
500   PRINT "          ACTIVE                                                INACTIVE"
510   PRINT "    "
520   FOR I=192 TO 400
530     ENTER @Pu,I;U(*)
540     ENTER @Pt,I;T(*)
550     ENTER @Pname,I;M$
560     IF M$="UNUSED" OR M$="                " THEN 750
570     IF U(82)=1 OR U(82)=2 THEN
580       U_790=DROUND(U(79),2)
590       IF U_790>=.8 THEN
600         PRINT USING "3D,5X,16A,3X,D.2D";I,M$,U_790
610         GOTO 750
620       END IF
630       IF U_790>=.6 THEN
640         PRINT USING "3D,5X,16A,10X,D.2D";I,M$,U_790
650         GOTO 750
660       END IF
670       IF U_790>=.4 THEN
680         PRINT USING "3D,5X,16A,20X,D.2D";I,M$,U_790
690         GOTO 750
700       END IF
710       IF U_790<.4 THEN
720         PRINT USING "3D,5X,16A,30X,D.2D,3A";I,M$,U_790,"***"
730       END IF
740     END IF
750   NEXT I
760   PRINTER IS 1
770   PRINT USING "@"
780   PRINT "DO YOU WANT TO RUN 1)ARMY?"
790   PRINT "                     2)STRENGTH?"
800   PRINT "                     3)DIME?"
810   INPUT X
820   IF X=1 THEN LOAD "ARMY:HP9134.701"
830   IF X=2 THEN LOAD "STRENGTH:HP9134.701"
840   IF X=3 THEN LOAD "DIME:HP9134.701"
850   END
```

Table 11-7. "CR_KV" program.

```
10     !
20     !   PROGRAM : CR_KV                              OMNI V5.0 - MAR 1986
30     !
40         OPTION BASE 1
50         DIM Kv_data(8,70)
60         PRINT CHR$(12)
70         REPEAT
80             INPUT "CREATE OR INIT THE KV FILE(CR/IN)?",A$
90         UNTIL A$="CR" OR A$="IN"
100        IF A$="CR" THEN
110            CREATE BDAT "KVFILE:HP9134,701,0",2,1360
120            PRINT TABXY(2,2);"CREATING KVFILE (2 RECORDS)"
130        END IF
140        PRINT TABXY(2,4);"INITIALIZING KVFILE"
150        ASSIGN @Path TO "KVFILE:HP9134,701,0"
160        FOR I=1 TO 8
170            FOR J=1 TO 70
180                Kv_data(I,J)=0
190            NEXT J
200        NEXT I
210        FOR Recnum=1 TO 2
220            OUTPUT @Path,Recnum;Kv_data(*),END
230            PRINT TABXY(2,7);Recnum;" . . . "
240        NEXT Recnum
250        PRINT TABXY(2,10);"KVFILE INITIALIZED."
260        ASSIGN @Path TO *
270        PRINT TABXY(2,17);CHR$(130);"INSERT";CHR$(128);" OMNI MENU DISK, THEN P
RESS ENTER..."
280        INPUT "",A$
290        LOAD "OMNI_MENU"
300        END
```

Table 11-8. "CR_HELO" program.

```
10      !
20      !   PROGRAM : CR_HELO                                   DIME V5.0 - MAR 1986
30      !
40          OPTION BASE 1
50          DIM Helo_data(3,6)
60          PRINT CHR$(12)
70          REPEAT
80              INPUT "(CR)CREATE OR (IN)INITIALIZE 'HELOFILE'?",A$
90          UNTIL A$="CR" OR A$="IN"
100         IF A$="CR" THEN
110             CREATE BDAT "HELOFILE:HP9134,701,0",2,144
120             PRINT TABXY(2,2);"CREATING HELOFILE (2 RECORDS)"
130         END IF
140         PRINT TABXY(2,4);"INITIALIZING HELOFILE"
150         ASSIGN @Phelo TO "HELOFILE:HP9134,701,0"
160         FOR I=1 TO 3
170             FOR J=1 TO 6
180                 Helo_data(I,J)=0
190             NEXT J
200         NEXT I
210         FOR Recnum=1 TO 2
220             OUTPUT @Phelo,Recnum;Helo_data(*)
230             PRINT TABXY(2,7);Recnum;" . . . "
240         NEXT Recnum
250         PRINT TABXY(2,10);"HELOFILE INITIALIZE."
260         ASSIGN @Phelo TO *
270         PRINT TABXY(2,17);CHR$(130);"INSERT";CHR$(128);" OMNI MENU DISK. THEN
RESS ENTER..."
280         INPUT "",A$
290         LOAD "OMNI_MENU"
300         END!
```

Table 11-9 "AIRPLANE" program.

```
10 !
20 !   PROGRAM : AIRPLANE                                OMNI V5.0 - MAR 1986
30 !
40 ! THIS IS THE UTILITY PROGRAM FOR BUILDING AND MAINTAINING THE AIR ATTACK/
50 ! AIR DEFENSE DATABASE.  THIS PROGRAM WAS CODED BY STEVE ARRINGTON AND CINDY
60 ! JAHNKE AND LAST CHANGED BY JIM LUNN ON 09 AUG 84.
70 !
80  !NOTE:  THIS DOCUMENT IS PROTECTED BY PROVISIONS UNDER AR 600-50
90  !       UNTIL PUBLICATION IN THE PUBLIC DOMAIN - "All DA Personnel will
100 !       refrain from releasing to an individual or business concern or
110 !       its representatives any knowledge such persons may possess or
120 !       have acquired in any way concerning proposed acquisition or
130 !       purchases by any contracting activity of DA . . . .  Such
140 !       information will be released to all potential contractors as
150 !       nearly simultaneous as possible . . . .  Such information will
160 !       be provided in accordance with existing authorized procedures
170 !       and only in connection with the necessary and proper discharge
180 !       of official duties."
190 !
191 !-------------------------------------------------------------------------
192 !       THIS CODE LISTING ONLY CONTAINS THE PART NECESSARY TO INITIALIZE
193 !       THE ACCUMULATIVE KILL FILES.  THE REST OF THE CODE DEALING WITH
194 !       THE ENTIRE DATABASE CAN BE FOUND IN VOLUME III OF THE DIME
195 !       DOCUMENTATION.
196 !-------------------------------------------------------------------------
200      OPTION BASE 1
210      DIM Area(10,4)! CONTAINS UNIT AREAS BASED ON 10 UNIT TYPES AND
220                    ! 4 MISSION POSTURES
230      DIM Flt(7,8)   ! CONTAINS INGRESS/EGRESS PROFILE DATA
240      DIM Flt_info(56)! CONTAINS INGRESS/EGRESS PROFILE DATA
250      DIM Load$(3)[14]! CONTAINS FILE NAME OF STRIKE PROFILES (MAX 3)
260      DIM Stk(28,8)  ! CONTAINS STRIKE PROFILE DATA
270      DIM Stk_info(224)! CONTAINS STRIKE PROFILE DATA
280      DIM Totals(51)! CONTAINS ACCUMULATION LOSSES FOR AIR ATTACK/AIR
290                    ! DEFENSE MODULE
300      DIM Wts(72,5)  ! CONTAINS MISSION/TARGET WEIGHT VALUES BASED ON
310                    ! 23 TARGET ELEMENTS AND 5 MISSION TYPES
320 !
330 !*************************************************************************
340 !
350 Main_program:   ! UTILITY PROGRAM MENU
360 !
370      REPEAT
380          PRINT CHR$(12)
390          PRINT "AIRPLANE - AIR DEFENSE DATA PROGRAM"
400          INPUT "SELECT  (D)DISKETTE OR (W)WINCHESTER ?",An$
410      UNTIL An$="W" OR An$="D"
420      IF An$="W" THEN
430          Disk$=":HP913X,701"
440      ELSE
450          Disk$=""
460      END IF
```

Table 11-9 "AIRPLANE" program.

```
470   !
480      Repeat_main$="Y"
490      WHILE Repeat_main$="Y"
500         PRINT CHR$(12)
510         PRINT "       DIME AIR ATTACK/AIR DEFENSE UTILITY PROGRAM"
520         PRINT
530         PRINT "  MENU :"
540         PRINT
550         PRINT "    1 - PROCESS AIR INGRESS/EGRESS PROFILE"
560         PRINT "    2 - PROCESS AIR STRIKE PROFILE"
570         PRINT "    3 - PROCESS UNIT AREA TEMPLATE"
580         PRINT "    4 - PROCESS WEAPON LOAD TEMPLATE"
590         PRINT "    5 - PROCESS AIR MISSION/TARGET PRIORITIES"
600         PRINT "    6 - INITIALIZE  AIR AND TARGET LOSSES"
610         PRINT "    7 - EXIT"
620         INPUT "SELECT OPTION :",Main_option
630         SELECT Main_option
640         CASE 1
650            GOSUB Prcs_ing_egr
660         CASE 2
670            GOSUB Prcs_air_stk
680         CASE 3
690            GOSUB Prcs_area_tmp
700         CASE 4
710            GOSUB Prcs_wpn_load
720         CASE 5
730            GOSUB Prcs_tgt_wt
740         CASE 6
750            GOSUB Init_losses
760         CASE 7
770            PRINT
780            PRINT " EXIT UTILITY PROGRAM."
790            STOP
800         END SELECT
810      END WHILE
820   !
830 !
840 !**********************************************************************
850  !
860 Init_losses:   ! REINITIALIZES AIR/TARGET LOSS FILES TO ZERO
870  !
880      GOSUB Slct_color
890      File$=Color$&"_AIR_INF"
900      PURGE File$&":HP913X,701"
910      CREATE BDAT File$&":HP913X,701",1,408
920      FOR I=1 TO 51
930         Totals(I)=0
940      NEXT I
950      ASSIGN @Path TO File$&":HP913X,701"
960      OUTPUT @Path,1;Totals(*)
970      ASSIGN @Path TO *
980      PRINT
```

Table 11-9 "AIRPLANE" program.

```
990       PRINT "FILE ";File$&":HP913X,701";" HAS BEEN INITIALIZED TO ZERO"
1000    !
1010      RETURN
1020 !
1030 !********************************************************************************
1040 !
1050 Slct_color:  ! SELECTS BLUE OR RED FORCE
1060 !
1070      REPEAT
1080          PRINT
1090          PRINT "SELECT (BL)BLUE OR (RD)RED FORCE  : ";
1100          INPUT Color$
1110      UNTIL Color$="BL" OR Color$="RD"
1120      PRINT Color$
1130 !
1140      RETURN
1150 !
1160 !********************************************************************************
1170 END
```

Table 11-10. CR_CHEM program.

```
10      !
20      !   PROGRAM : CR_CHEM                                   DIME V5.0 - MAR 1986
30      !
40          OPTION BASE 1
50          DIM Chem(70)
60          PRINT CHR$(12)
70          REPEAT
80              INPUT "(CR)CREATE OR (IN)INITIALIZE CHEMICAL FILES?",A$
90          UNTIL A$="CR" OR A$="IN"
100         IF A$="CR" THEN
110             CREATE BDAT "BLCHMVCTM:HP9134,701,0",1,560
120             CREATE BDAT "RDCHMVCTM:HP9134,701,0",1,560
130             PRINT TABXY(2,2);"CHEMICAL FILES CREATED."
140         END IF
150         PRINT TABXY(2,4);"INITIALIZING CHEMICAL FILES (1 RECORD/FILE)"
160         FOR I=1 TO 70
170             Chem(I)=0
180         NEXT I
190         ASSIGN @Pchem TO "BLCHMVCTM:HP9134,701,0"
200         OUTPUT @Pchem,1;Chem(*)
210         ASSIGN @Pchem TO "RDCHMVCTM:HP9134,701,0"
220         OUTPUT @Pchem,1;Chem(*)
230         ASSIGN @Pchem TO *
240         PRINT TABXY(2,10);"CHEMICAL FILES INITIALIZED."
250         PRINT TABXY(2,17);CHR$(130);"INSERT";CHR$(128);" OMNI MENU DISK. THEN
RESS ENTER..."
260         INPUT "",A$
270         LOAD "OMNI_MENU"
280         END!
```

Table 11-11. Off-line assessment program code.

```
10    !!! "P6" ALLOWS OFF-LINE INPUT OF LOSSES TO UNITS BY SEPARATE KILLER
20    !    CATEGORY.  CODED BY MAJ REISCHL, OPNS ANAL BR, CAORA (A/V 552-4613).
30    !    THIS PROGRAM WAS LAST UPDATED ON 11 JAN  1984 BY CINDY JAHNKE.
40    !********************************************************************
50    ! EXPANDED VERSION -- JUNE 9, 1986 -- BY OAO CORP.
51    !      DECLASSIFIED -- AUG 7, 1986 -- BY OAO CORP.   ** DC **
60    OPTION BASE 1
70    DIM Kv_data(8,70),Loss(70),Sys(70),N(150),Loss_allowed(70),S(150)
80    DIM Sys_eff(2,70),T1(28),Air_loss(72)
90    INTEGER I,J,K,System,Killer
91    Dcdisk$=":HP9134,701,0"  !  ** DC **
100   ASSIGN @Unitpath TO "UNITFILE:HP9134,701"
110   ASSIGN @Kvpath TO "KVFILE:HP9134,701"
120   '
121   '  ** DC **
122   '
130   ASSIGN @Psyseff TO "SYS_EFF"&Dcdisk$
140   ENTER @Psyseff,1;Sys_eff(*)
150   ASSIGN @Psyseff TO *
160   '  ** END DC **
290   PRINTER IS 1
300   '
310   ' START INPUT PORTION
320 Start_assess:PRINT USING "@  ,25X,26A":"OFF-LINE UNIT ASSESSMENT"
330   INPUT "ENTER # OF UNIT TO BE ASSESSED (999=STOP): ",I
340   IF I=999 THEN Units_all_done
350   IF I<0 OR I>400 THEN 320
360   '
370   ' READ IN UNIT TO BE CHANGED
380   ENTER @Unitpath,I;N(*)
390   '
400   ' READ IN RED OR BLUE KV MATRIX
410   IF I<192 THEN
420     ENTER @Kvpath,1;Kv_data(*)
430     Side=1
440   ELSE
450     ENTER @Kvpath,2;Kv_data(*)
460     Side=2
470   END IF
480   FOR J=1 TO 70
490     Sys(J)=N(J)
500     S(J)=N(J)
510   NEXT J
520   '
530   ! PRINT OUT CURRENT LIST AND ASK FOR CHANGES
540 Start_input:PRINT "  "
550   PRINT "UNIT: ",I
560   PRINT "  "
570   PRINT "SYSTEMS AVAILABLE: "
580   GOSUB Print_sys_out
590   INPUT "# KILLER CAT: 1-D/F  2-I/F  3-PGM  4-A/H  5-INF  6-MIN  7-CHM  8-A
F  999-STOP",Killer
```

Table 11-11. Off-line assessment program code.

```
600    IF Killer=999 THEN Unit_done
610    IF Killer<1 OR Killer>8 THEN 590
620    FOR J=1 TO 70
630      Loss(J)=0
640    NEXT J
650 Put_in_loss:INPUT "ENTER LOSS BY SYS#, #LOST  (999.999=STOP): ".System,No_l
st
660    IF System=999 THEN Stop_losses
670    IF System<1 OR System>70 OR No_lost<0 THEN 650
680    Loss(System)=No_lost
690    GOTO Put_in_loss
700    !
710 Stop_losses:  ! SUBTRACT LOSSES AND CHECK FOR ERRORS
720    FOR J=1 TO 70
730      Temp_loss=0
740      Temp_loss=Sys(J)-Loss(J)
750      IF Temp_loss<0 THEN
760        Loss(J)=Sys(J)
770        Loss_allowed(J)=Sys(J)
780        Sys(J)=0
790      ELSE
800        Sys(J)=Temp_loss
810        Loss_allowed(J)=Loss(J)
820      END IF
830    NEXT J
840    FOR J=1 TO 70
850      S(J)=Loss_allowed(J)
860    NEXT J
870      !
880      ! PRINT OUT CURRENT FORCE LEVEL AND UPDATE KV FILE
890    PRINT "  "
900    PRINT "LOSSES ALLOWED:"
910    GOSUB Print_sys_out
920    FOR J=1 TO 70
930      S(J)=Sys(J)
940    NEXT J
950    FOR J=71 TO 150
960      S(J)=N(J)
970    NEXT J
980    PRINT "  "
990    PRINT "SYSTEMS REMAINING:"
1000   GOSUB Print_sys_out
1010   FOR J=1 TO 70
1020     Kv_data(Killer,J)=Kv_data(Killer,J)+Loss_allowed(J)
1030   NEXT J
1040 Unit_done:INPUT "DO YOU WISH UNIT ASSESSMENT POSTED TO UNITFILE?  (Y or N)
",Q$
1050   IF Q$<>"Y" AND Q$<>"N" THEN 1040
1060   IF Q$="N" THEN
1070     GOTO Start_assess
1080   ELSE
1090     OUTPUT @Kvpath,Side;Kv_data(*)
```

Table 11-11. Off-line assessment program code.

```
1100     OUTPUT @Unitpath,I;S(*)
1110     ENTER @Unitpath,I;S(*)
1120     GOSUB Upd_ammo_fuel
1130     GOSUB Upd_cbt_eff
1140     OUTPUT @Unitpath,I;S(*)
1150   END IF
1160   PRINT USING "@,#"
1170   GOTO Start_assess
1180     !
1190     !
1200 Units_all_done:       !  UNIT ENTRY COMPLETED
1210   INPUT "DO YOU WANT AN UPDATED KILLER-VICTIM MATRIX?  (Y or N)   ",Q$
1220   IF Q$<>"Y" AND Q$<>"N" THEN 1210
1230   IF Q$="Y" THEN GOSUB Print_kv_matrix
1240   ASSIGN @Unitpath TO *
1250   ASSIGN @Kvpath TO *
1260   GOTO Halt
1270     !
1280     !
1290     ! ********************* END OF MAIN PROGRAM *****************************
1300     !
1310 !
1320 !****************************************************************************
1330 !
1340 Upd_ammo_fuel: !
1350     ! UPDATE CARGO TRUCKS
1360   IF S(58)<N(58) AND N(58)>0 THEN
1370     Tons_on_truck=N(123)/N(58)
1380     S(123)=S(58)*Tons_on_truck
1390     IF S(123)<0 THEN S(123)=0
1400     S(84)=S(58)
1410   END IF
1420     !
1430     !UPDATE FUEL
1440   IF S(55)<N(55) AND N(55)>0 THEN
1450     Gal_on_truck=N(103)/N(55)
1460     S(105)=Gal_on_truck*S(55)
1470     IF S(105)<0 THEN S(105)=0
1480     S(85)=S(55)
1490   END IF
1500     !
1510   RETURN
1520     !
1530     !*************************************************************************
1540     !
1550 Upd_cbt_eff: !
1560   Neff=0
1570   Seff=0
1580   FOR Ieff=1 TO 70
1590     Seff=Seff+S(Ieff)*Svs_eff(Side,Ieff)
1600     Neff=Neff+N(Ieff)*Svs_eff(Side,Ieff)
1610   NEXT Ieff
```

Table 11-11. Off-line assessment program code.

```
1620  S(79)=(Seff/Neff)*N(79)
1630    !
1640  RETURN
1650    !
1660    !**********************************************************************
1670    !
1680 Print_sys_out:    ! THIS SBR PRINTS OUT THE SYSTEMS LIST
1690    !
1700  FOR X=7 TO 70 STEP 7
1710    PRINT USING Fmt1:X-6,":",S(X-6),X-5,":",S(X-5),X-4,":",S(X-4),X-3,":",S(
X-3),X-2,":",S(X-2),X-1,":",S(X-1),X,":",S(X)
1720  NEXT X
1730 Fmt1:IMAGE 7(2D,1A,4D,1D,2X)
1740  !
1750  RETURN
1760  !
1770  ! **********************************************************************
1780  !
1790 Print_kv_matrix:   !  THIS SBR PRINTS OUT THE CURRENT KV MATRIX
1800  !
1810  PRINT USING "//////////"
1820  PRINT "                             TO OBTAIN AN UPDATED "
1830  PRINT "                  "
1840  PRINT "                             KILLER-VICTIM MATRIX "
1850  PRINT
1860  PRINT "                                   PRINT-OUT     "
1870  PRINT
1880  PRINT "        RUN OPTION 8 IN DIME MENU, THEN USE THE KV-FILE ONLY OPTI(
"
1890  PRINT "                             .                  ------------"
1900  PRINT USING " ///"
1910  PRINT "                    PRESS CONT TO RETURN TO THE DIME MENU"
1920  PAUSE
1930  RETURN
1940  !
1950  ! **********************************************************************
1960  !
1970 Halt:LOAD "DIME:HP9134,701"
1980  END
```

APPENDIX A

BIBLIOGRAPHY

1. Bunch, John H., Jr. and Strater, Felix R., Jr. Information Systems. Santa Barbara: Hamilton Publishing Co., 1974.

2. Couger, J. Daniel and Kapp, Robert W. Systems Analysis Techniques. New York: John Wiley & Sons, 1974.

3. Defense Intelligence Agency. DDB-1100-333-82: Soviet Divisional Organizational Guide. July, 1982.

4. ______. DDB-1110-1-79: The Soviet Motorized Rifle Division. September, 1979.

5. ______. DDB-1120-10-80: Soviet Tabk Battalion Tactics. November, 1980.

6. ______. DST-1120F-015-81: Soviet Tank/Antiarmor Threat 1981-2000 (U). August, 1981. (SECRET/REL FRG, FR, UK, AUS & CAN).

7. ______. DST-1120S-114-890: Soviet Close Combat Technical Capabilities Assessment (U). March, 1980.

8. ______. DST-13405-224-81: Hind Helicopter System (U). DST-13405-224-81. June 1981. (SECRET/NOFORN/WNINTEL)

9. Department of the Army. Field Manual FM 3-10-1: Employment of Chemical Agents. Washington, DC: 15 December 1978. (CONFIDENTIAL)

10. ______. Field Manual FM 5-35: Engineer's Reference and Logistical Data. Washington, DC: 30 September 1975.

11. ______. Field Manual FM 6-20: Fire Support in Combined Arms Operation. Washington, DC: 30 September 1977.

12. ______. Field Manual FM 7-20: The Infantry Battalion. Washington, DC: 3 April 1978.

13. ______. Field Manual FM 11-50: Combat Communications Within the Division. Washington, DC: 31 March 1977.

14. ______. Field Manual FM 17-12: Tank Gunnery. Washington, DC: 21 March 1977.

15. ______. Field Manual FM 17-47: Air Cavalry Combat Brigade. Washington, DC: 29 April 1977.

16. ______________. Field Manual FM 17-50: Attack Helicopter Operation. Washington, DC: 1 July 1977.

17. ______________. Field Manual FM 17-95: US Cavalry. Washington, DC: 1 July 1977.

18. ______________. Field Manual FM 21-30: Military Symbols. Washington, DC: 31 March 1980.

19. ______________. Field Manual FM-21-1: Combat Communications. Washington, DC: 30 September 1976.

20. ______________. Field Manual FM 24-22: Communications - Electronic Management Systems. Washington, DC: 1 July 1978.

21. ______________. Field Manual FM 24-22: Communications - Electronic Management Systems. Washington, DC: 1 July 1978.

22. ______________. Field Manual FM 30-5: Combat Intelligence. Washington, DC: 30 October 1973.

23. ______________. Field Manual FM 32-30: Electronic Warfare. Washington, DC: 31 March 1975.

24. ______________. Field Manual FM 44-1: Air Defense Artillery Employment. Washington, DC: 30 September, 1977.

25. ______________. Field Manual FM 44-3: Air Defense Artillery Employment, Chaparral/Vulcan. Washington, DC: 30 September, 1977.

26. ______________. Field Manual FM 44-23: Air Defense Artillery Employment, Redeye. Washington, DC: 30 September 1977.

27. ______________. Field Manual FM 54-2: Division/ Brigade Support Command. Washington, DC: 1 July 1976.

28. ______________. Field Manual FM 54-7: Theater Army Logistics. Washington, DC: 31 March 1975.

29. ______________. Field Manual FM 54-9: Corps Support Command. Washington, DC: 1 July 1978.

30. ______________. Field Manual FM 71-1: Tank and Mechanized Infantry Company Team. Washington, DC: 30 June 1977.

31. ______________. Field Manual FM 71-2: Tank and Mechanized Infantry Battalion Task Force. Washington, DC: 30 June 1977.

32. ______________. Field Manual FM 71-100: Armored and Mechanized Division Operations. Washington, DC: 30 September 1977.

33. ____________________. Field Manual FM 90-1: Aviation Units in a High Threat Environment. Washington, DC: 20 May 1977.

34. ____________________. Field Manual FM 90-3: Desert Operations. Washington, DC: 19 August 1977.

35. ____________________. Field Manual FM 100-2-1: Soviet Army Operations and Tactics (draft). USACACDA Threats Division, Fort Leavenworth, KS: 1 July 1982.

36. ____________________. Field Manual FM 100-2-2: Soviet Army Specialized Warfare and Rear Area Support (draft). USACACDA Threats Division, Fort Leavenworth, KS: 1 July 1982.

37. ____________________. Field Manual FM 100-5: Operations. Washington, DC: 20 August 1982.

38. ____________________. Field Manual FM 100-15: Corps Operations (draft). Fort Leavenworth, KS: 27 June 1980.

39. ____________________. Field Manual FM 100-16: Support Operations - Echelons Above Corps (draft). Fort Leavenworth, KS: June 1982.

40. ____________________. Field Manual FM 105-5: Maneuver Control. Washington, DC: 10 May 1976.

41. ____________________. Field Manual FM 101-10-1: Staff Officer's Field Manual. Washington, DC: 1 July, 1976.

42. ____________________. Training Circular TC 7-24: Antiarmor Tactics for Mechanized Infantry. Fort Benning, GA: (not dated).

43. ____________________. Training Circular TC 24-18: Communications. Washington, DC: (not dated).

44. ____________________. Training Circular TC 101-5: Control and Coordinating Division Operations. Washington, DC: (not dated).

45. Department of Defense. Joint Munitions Effectiveness Manual FM 101-60-3: Effectiveness Data for 155mm Howitzer. Washington, DC: November 1980. (CONFIDENTIAL)

46. ____________________. Joint Munitions Effectiveness Manual FM 101-60-4: Effectiveness Data for 8-inch Howitzer. Washington, DC: July 1980. (CONFIDENTIAL)

47. ____________________. Joint Munitions Effectiveness Manual FM 101-60-17: Basic Effectiveness Manual. Washington, DC: September, 1976. (CONFIDENTIAL)

48. Department of Defense, Australia. Water Buffalo Land Component. 1 April 1982

49. Hembold, Robert L. "Solution of a General, Nonadaptive, Many-VS-Many Duel Model." Operations Research. May/June 1968.

50. Huber, Reiner K., Jones, Lynn F., and Reine, Egile. Military Strategy and Tactics. New York: Plenum Press, 1975.

51. Isbey, David C. Weapons and Tactics of the Soviet Army. London: Jane's, 1981.

52. Department of the Army. CACDA SCORES European Scenario III, Sequence 2A (U). October 1980. (SECRET/NOFORN/WNINTEL)

53. ________________. Combined Arms COmbat Developments Activity. CAS Army Force Planning Data and Assumptions FY 1982-1991 (U). July 1981. (SECRET/NOFORN/WNINTEL)

54. ________________. Combined Arms Combat Developments Activity Letter, ATCD-AN-R "Mission Area Analysis (MAA)". Fort Leavenworth, KS: 18 October 1978.

55. ________________. Combined Arms Combat Developments Activity Letter, ATZL-CA-DL "Logistic Planning Factors, Division 86", Change 1(U). Fort Leavenworth, KS: 3 May 1979.

56. ________________. Combined Arms Combat Developments Activity Letter, ATZL-CAS-FS, 'Ammunition Expenditure Rates." Fort Leavenworth, KS: 26 April 1982.

57. ________________. Combined Arms Combat Developments Activity Technical Report 6-80: Jiffy III War Game Methodology, Vols. II, III, Fort Leavenworth, KS: September, 1980.

58. ________________. Combined Arms Combat Developments Activity Technical Report 4-78: CACDA Ground Combat Model. Fort Leavenworth, KS: August, 1978.

59. ________________. Combined Arms Combat Developments Activity Technical Report 5-83: Deep Attack Map Exercise (DAME) Game Rules and Operation Procedures. Fort Leavenworth, KS: February, 1983.

60. Department of the Army. USACGSC Reference Book RB 3-1: NBC Operations (draft). Fort Leavenworth, KS: 31 July 1973.

61. Department of the Army. USACGSC Reference Book RB 32-20: Electronic Warfare (draft). Fort Leavenworth, KS: 31 March 1980.

62. Department of the Army. USACGSC Reference Book RB 100-1: Organization Data for the Army in the Field. Fort Leavenworth, KS: 9 January 1980.

63. Department of the Army. USACGSC Reference Book RB 100-5: Command and Control of Combat Operation. Fort Leavenworth, KS: 28 November 1977.

64. Department of the Army. USACGSC Reference Book RB 100-8: Reference Data for Heavy Maneuver Forces. Fort Leavenworth, KS: 1 August 1980.

65. Department of the Army. USACGSC Reference Book RB 100-33: Electronic Leavenworth, KS: 1 June 1979.

66. U.S. Army Concepts Analysis Agency. Target Acquisition Study II. Washington, D.C: August 1979. (SECRET)

67. Department of the Army. Field Artillery School, Director of Combat Developments, Programmable Calculator (TI-58 or TI-59) Manual for Evaluating Effectiveness of Non-nuclear Surface-to-Surface Indirect-Fire Weapons Against Area Targets. Fort Sill, OK: 02 June 1981.

68. Wagner, Harvey M. Principles of Operations Research. New York: Prentice - Hall, 1975.