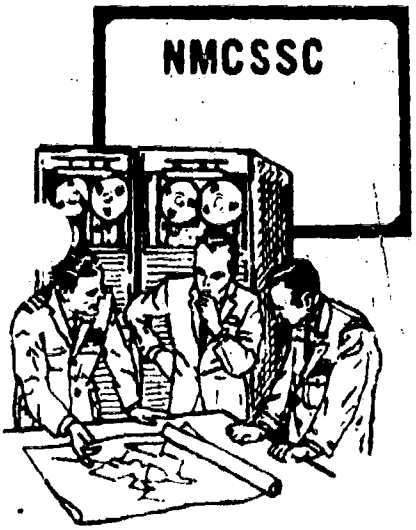
NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER



DEFENSE COMMUNICATIONS AGENCY

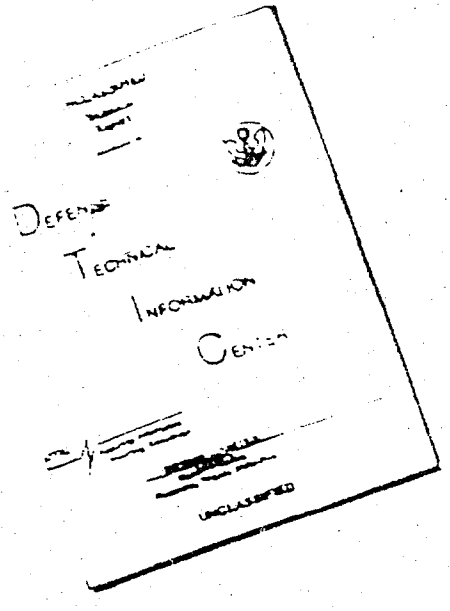
COMPUTER SYSTEM MANUAL
CSM PSM 9A-67
VOLUME II, PART A
29 FEBRUARY 1972

THE NMCSSC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK)

PLAN GENERATION SUBSYSTEM

PROGRAMMING SPECIFICATIONS MANUAL

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER

Computer System Manual Number CSM PSM 9A-67

29 February 1972

THE NMCSSC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK)

Programming Specifications Manual

Volume II - Plan Generation Subsystem

Part A (Chapters 1 through 5)

Submitted by:

Donald F. Webb

DONALD F. WEBB
Major, USAF
Project Officer

REVIEWED BY:

R E Harshbarger

R. E. HARSHBARGER
Technical Director
NMCSSC

APPROVED BY:

Bruce Merritt

BRUCE MERRITT
Colonel, USA
Commander, NMCSSC

ACKNOWLEDGMENT

This document was prepared under the direction of the Chief for Development and Analysis, NMCSSC, in response to a requirement of the Studies, Analysis and Gaming Agency (SAGA), Organization of the Joint Chiefs of Staff. Technical support was provided by Lambda Corporation under Contract Number DCA 100-70-C-0065.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
National Military Command System Support Center (NMCSSC)
Defense Communications Agency (DCA)
The Pentagon
Washington, DC 20301

2a. REPORT SECURITY CLASSIFICATION

2b. GROUP

3. REPORT TITLE
The NMCSSC Quick-Reacting General War Gaming System (QUICK)
Programming Specifications Manual, Volume II, Plan Generation Subsystem

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
N/A

5. AUTHOR(S) *(First name, middle initial, last name)*
NMCSSC: Robert R. Hardiman
Yvonne Maplly
Donald F. Webb
Lambda Corp: Paul D. Flanagan
Patricia M. Parish
Jack A. Sasseen

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
29 February 1972	461	4

8a. CONTRACT OR GRANT NO.
DCA 100-70-C-0065

b. PROJECT NO. NMCSSC Project 631

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
NMCSSC
COMPUTER SYSTEM MANUAL CSM PSM 9A-67

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
None

10. DISTRIBUTION STATEMENT
This document is approved for public release; its distribution is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
National Military Command System Support Center/Defense Communications Agency
The Pentagon, Washington, DC 20301

13. ABSTRACT

This is one of three volumes describing computer programs of the QUICK-Reacting General War Gaming System (QUICK). These volumes complement other NMCSSC Computer System Manuals on QUICK by discussing the programs from a computer programming point of view. This volume, in six parts, concentrates on the Plan Generation Subsystem of QUICK. Other volumes are available for the Input Subsystem and Simulation Subsystem. Collectively, these volumes provide a good basis for maintenance activity on the QUICK System.

Based upon a suitable data base, and user control parameters, QUICK will generate individual bomber and missile plans suitable for war gaming. The generated plans are of a form suitable for independent review and revision. Subsequently, execution of the planned events can be simulated. Various statistical summaries can be produced to reflect the results of the war game. A variety of force postures and strategies can be accommodated.

QUICK is documented extensively in a set of Computer System Manuals (series 9-67) published by the National Military Command System Support Center (NMCSSC), Defense Communications Agency (DCA), The Pentagon, Washington, DC 20301.

DD FORM 1 NOV 68 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

Security Classification

CONTENTS

PART A

Chapter Page

PART B

PART C

PART D

PART E

PART F

ILLUSTRATIONS (PART A)

TABLES (PART A)

ABSTRACT

The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, simulate the planned events, and provide statistical output summaries. QUICK has been programmed in FORTRAN for use on the NMCSSC CDC 3800 computer system.

The QUICK Programming Specifications Manual (PSM) consists of three volumes: Volume I, Data Input Subsystem; Volume II, Plan Generation Subsystem; Volume III, Simulation and Data Output Subsystems. The Programming Specifications Manual complements the other QUICK Computer System Manuals to facilitate maintenance of the war gaming system. This volume, Volume II, provides the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the programs of the Plan Generation subsystem. This volume is in six parts: Parts A and B provide a description of the programs which make up the Subsystem: Parts C through F contain the associated program listings. Companion documents are:

1. GENERAL DESCRIPTION
   Computer System Manual CSM GD 9A-67
   A nontechnical description for senior management personnel

2. ANALYTICAL MANUAL
   Computer System Manual CSM AM 9A-67 (three volumes)
   Provides a description of the system methodology for the non-programmer analysts

3. USER'S MANUAL
   Computer System Manual CSM UM 9-67
   Provides detailed instructions for applications of the system

4. OPERATOR'S MANUAL
   Computer System Manual CSM OM 9A-67
   Provides instructions and procedures for the computer operators

CHAPTER 1
INTRODUCTION

The QUICK system consists of four functional subsystems: the Data Input, Plan Generation, Simulation, and Data Output subsystems*. This volume, Volume II of the Programming Specifications Manual, describes the QUICK Plan Generation subsystem, hereafter referred to as the Plan Generator. The general concept of operation of the QUICK system and the analytical aspects of the design of the Plan Generator are presented in the Analytical Manual, Volume II, Plan Generation subsystem. A detailed description of the user-input parameters required for operating the Plan Generator is contained in Chapter 3, Plan Generation Subsystem, User's Manual, Volume II.

This chapter provides a general overview of the functions performed by the programs of the Plan Generator. Each of these programs is discussed in detail in a separate chapter of this volume. Within each chapter, the initial sections describe the concept of operation and provide a description of the input/output files and common blocks associated with each program. Subsequent sections of the chapter describe the subroutines which constitute the program.

GENERAL DESCRIPTION

The Plan Generator operates using the target system and weapon resources supplied to it from the indexed data base prepared by the Data Input subsystem.

The information included in the data base is categorized by CLASS (e.g., bombers) and by TYPE within class (e.g., B-52). Fifteen classes may be used to describe the targetable-type installations included in the data base. The data categories currently associated with each target class are shown in table 1. These classes are identified within the system by referencing the value of the attribute ICLASS, the class number. The attributes** assigned to the items of ICLASS 1 through 5 and ICLASS 14 define the item not only as a target but include the attributes which establish its offensive/defensive capabilities and characteristics. In

*The QUICK subsystems are also referenced by the names Input subsystem, Plan Generator, Simulator, and Output subsystem.

**For attribute descriptions, see appendix A.

addition to these target classes, nine auxiliary data classes are used to enter weapon-type data such as the specific composition of a bomber payload. The auxiliary classes are identified within the system by the class name shown in table 1.

The Plan Generator is actually a sequence of programs that are executed in series. Each computer program is a separate processor that resides by itself in the memory of the computer. When a processor has completed its assigned operations, it writes on tape or disk all information that will be required by later processors and then returns control to the computer's monitor system. The monitor then calls the next processor into memory.

The information that must be transmitted from one processor to the next consists of two basic types. The first is data required by all processors. These data, which logically could remain in the memory at all times, are assembled by program PREPALOC and read into a file called BASFILE (or base file). This file is then read into memory by each succeeding processor. The second type of data is that which is developed by a processor as input information for later stages. Its transmission via tape or disk is conventional and requires no particular comment here.

Figure 1 illustrates the flow within the Plan Generator. The basic input tape contains a complete and consistent indexed list of all Red and Blue game objects. The output of PLANSET for a single run of the Plan Generator includes either Blue weapons and Red targets for a Blue plan, or alternatively Red weapons and Blue targets for a Red plan. Thus, two complete runs of the Plan Generator from the same indexed data base file are necessary to provide the plans required for the operation of the Simulator. The role of the various programs within the Plan Generator is as follows.

PLANSET: This program processes the indexed data base and selects the offensive system and target data appropriate for the plan. In addition, the program aggregates the offensive weapons into groups and prepares the target list for input to program ALOC.

PREPALOC: This program precomputes much of the information required by later processors. It organizes the input data for efficient use by other components of the Plan Generator. In addition, it provides capabilities for planning factor modification and fixed weapon assignment specification.

The basic data manipulated by this program include the distance and attrition factors for the weapons, the geographic description of the air defense zones and the bomber penetration corridors, the weapon characteristic tables (e.g., warhead and payload tables), and the target characteristics.

Table 1. QUICK Classes
(Sheet 1 of 2)

TARGET CLASSES

ICLASS	CLASS NAME	DATA CATEGORY
1	MISSILE	Offensive missiles
2	BOMBER	Offensive bombers
3	TANKER	Tankers
4	DEFCONTR	Defensive command and control
5	INTCPTOR	Interceptor aircraft
6	C/C	Offensive command and control
7	NUCSTOR	Nuclear storage sites
8	AIRFIELD	Airfields
9	NAVAL	Naval targets
10	TROOPS	Troops
11	COMMUN	Communications
12	MISC	Miscellaneous
13	U/I	Urban/industrial targets
14	ABMDEF	Area antiballistic missile (ABM) defense components
15	-	(Reserved for future use)

Table 1. (cont.)
(Sheet 2 of 2)

AUXILIARY CLASSES

CLASS NAME*	PURPOSE
WARHEAD	Provides warhead characteristics; e.g., yield
ASM	Provides characteristics of air-to-surface missiles; e.g., speed, warhead
PAYLOAD	Identifies weapons and penetration aids carried by a missile or bomber
DBLDATA	Contains time-dependent destruction-before-launch (DBL) probability tables
ZONE	Establishes the area of bomber air defense zones
POINT	Provides latitude and longitude for zone, route, and refuel points
BOUNDARY	Establishes the boundaries of bomber air defense zones
CORRIDOR	Identifies penetration corridors and associated attrition parameters
LEGS	Defines the penetration and depenetration route legs

*The auxiliary classes are not assigned numbers; i.e., the attribute ICLASS is not applicable to these classes.

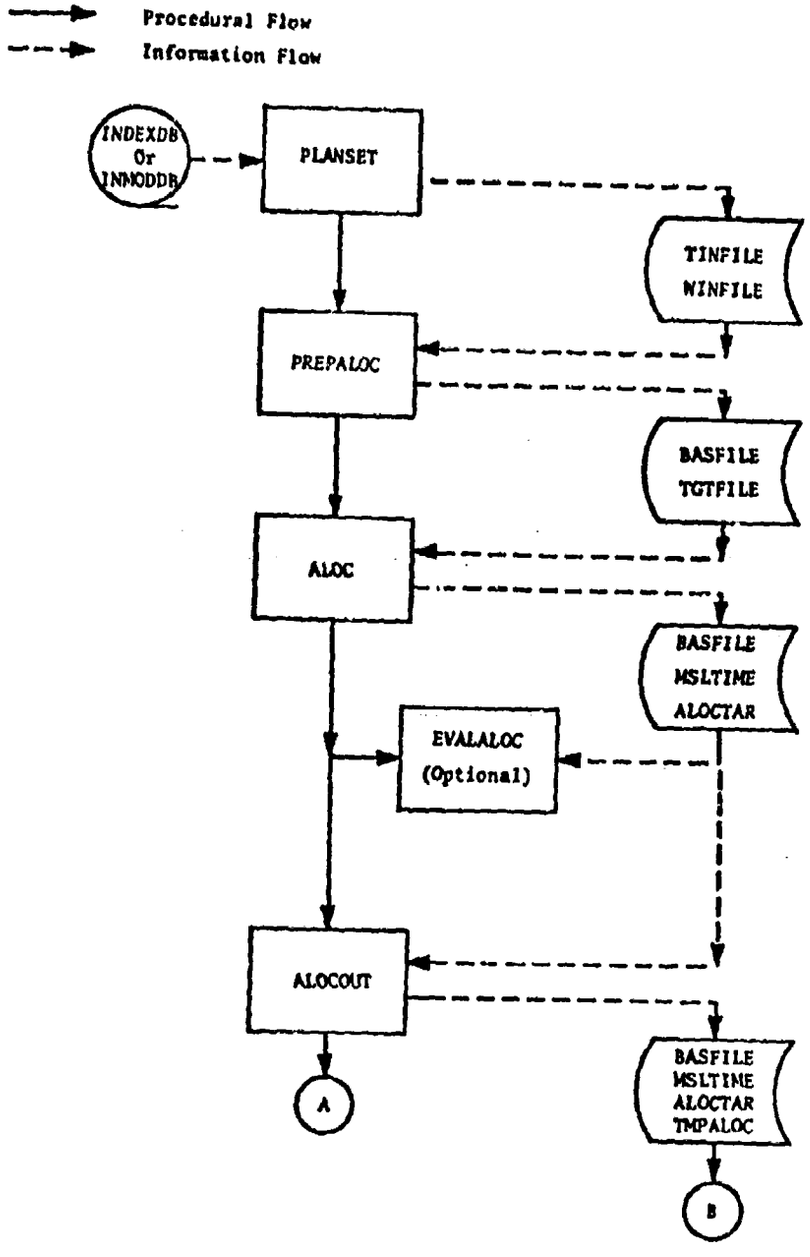


Fig. 1. Flow Within Plan Generator
(Sheet 1 of 2)

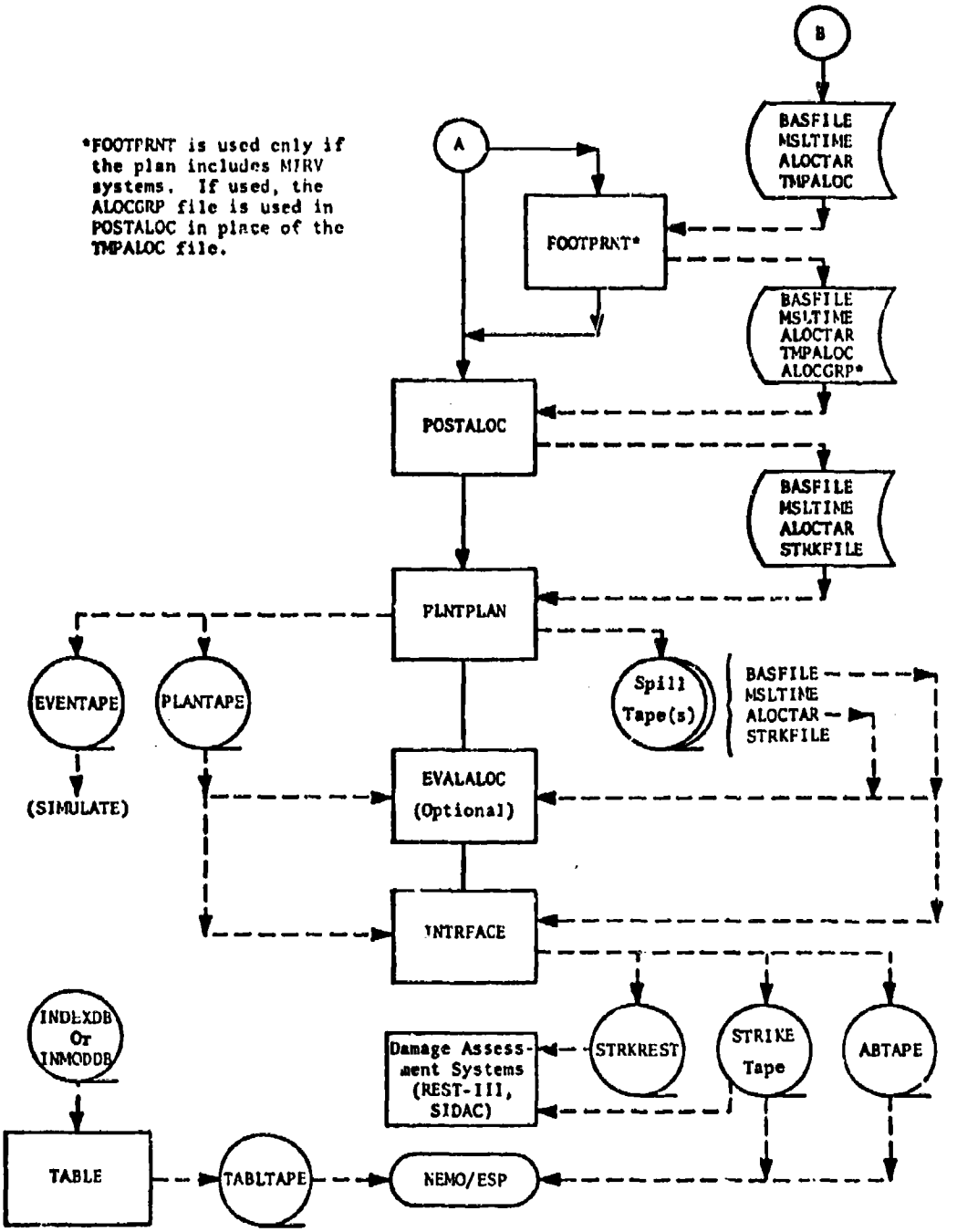


Fig. 1. (cont.)
(Sheet 2 of 2)

ALOC: This program performs the allocation of weapons to targets. Using a generalized Lagrange multiplier method, an optimal allocation is generated subject to several forms of user-input allocation constraints. These constraints include specification of minimum and maximum desired damage levels, restriction of weapons to specified subsets of the target system, and specification of weapons allocated to specific targets by the user. Within these constraints, the program generates the allocation which maximizes the expected value destroyed in the target system. Program ALOC is also referred to as the allocator.

EVALALOC: The main function of program EVALALOC is to provide a summary of the allocation produced in program ALOC and to calculate an expected-value estimate of its results. In addition, the program has the capability of evaluating the effect upon the results of variations in input values for weapon and target parameters. Program EVALALOC may be run either before program ALOCOUT or after program PLNTPLAN.

ALOCOUT: ALOCOUT optimizes the location of aim points for target complexes and collects all the strikes assigned to each weapon group by the allocator, so that detailed plans for each group can be formulated by FOOTPRNT and POSTALOC.

FOOTPRNT: This program processes the target assignments and assigns individual re-entry vehicles to aim points, within a geographic pattern known as a footprint constraint, for weapon groups possessing a multiple independently targetable re-entry vehicle (MIRV) capability.

POSTALOC: This program processes the weapon groups, one group at a time, and associates the individual strikes with specific missiles and specific bomber sorties. (For bombers carrying multiple warheads, several strikes must be combined in a single sortie.)

PLNTPLAN: This program accepts the basic missile and bomber plans output by POSTALOC, adds details required by the Simulator, and finally arranges the plans in the specific format required by the Simulator (EVENTAPE) or in a convenient readable form for manual observation (PLANTAPE) or use by other systems.

INTRFACE
and
TABLE: These programs ar[illegible]cial-purpose processors which provide an interfa[illegible]etween QUICK and other systems (they provide no output used within QUICK). These programs extract and reformat data from QUICK-developed files and output data which can drive the Event Sequence Program (ESP), the Nuclear Exchange MOdel (NEMO), and/or a NMCSSC damage assessment system such as REST-III or SIDAC (REsource STatus Damage Assessment Model III, and Single Integrated Damage Analysis Capability System).

QUICK GENERAL-PURPOSE UTILITY PACKAGE

In addition to the main programs of the four QUICK subsystems, QUICK employs a general-purpose utility package. This utility package consists of programs, subroutines, and functions which perform a variety of support tasks common to two or more system programs. These programs and routines are discussed in chapters 2, 3, and 4 of the Programming Specifications Manual (PSM), Volume I, Data Input Subsystem and, where appropriate, in chapter 1 of the QUICK User's Manual, Volume II (see Special-Purpose Utility Routines). Appendix B of this volume contains a list of the entry points within the utility programs.

The QUICK System Filehandler

The QUICK system filehandler uses a word stream concept of operation. For the calling program, the filehandler retrieves or sends a stream of words from/to the input/output (I/O) devices. Thus, the programmer need never consider the makeup of the logical or physical records on the tape or disk for filehandler files. Only in the filehandler itself need the maintenance programmer be concerned with the physical characteristics of I/O. In the using program, input/output on filehandler files consists of a word stream. The program merely requests transfer of a number of words to/from the device. Thus, a description of the physical record structure of filehandler files is irrelevant to the maintenance programmer except when he is maintaining the filehandler subroutines themselves.

All programs of the Plan Generation subsystem use the filehandler in conjunction with input/output operations. The filehandler subroutines and their functions are summarized below. A detailed description of the QUICK filehandler is contained in chapter 2 of the Programming Specifications Manual, Volume I.

SUBROUTINE	FUNCTION
ALOCDIR	Initializes disk file directory
INITAPE	Initializes filehandler
DEACTIV	Removes file name from active list
SETREAD	Prepares file for reading
RDWORD	Transfers one word from file input buffer to common /TWORD/
RDARRAY	Transfers block of words from file input buffer to user-specified core storage area
SETWRITE	Prepares file for writing
WRWORD	Transfers one word from common /TWORD/ to file output buffer
WRARRAY	Transfers block of words from user-specified core storage area to file output buffer
TERMTAPE	Terminates files after reading or writing and releases buffer area for use by other files

COMPUTER STORAGE REQUIREMENTS

The NMCSSC CDC 3800 computer provides a maximum of 65,534 words of core storage. Excluding the requirements of the operating system, the core storage requirements of the programs of the Plan Generator are as follows.

PLANSET	55,900*
PREPALOC	50,400
ALOC	55,800
ALOCOUT	55,000
FOOTPRNT	48,200
POSTALOC	54,900
PLNTPLAN	45,700
EVALALOC	44,600
INTRFACE	18,[illegible]00
TABLE	13,500

* Decimal words.

CHAPTER 2
PROGRAM PLANSET

PURPOSE

PLANSET prepares the data files required by the Plan Generator to develop a plan for one side. It forms weapon groups, prepares the target list, computes and normalizes the class value factors, calculates the representative attributes for complex targets, and creates the WINFILE (weapon input file) and TINFILE (target input file) required by program PREPALOC. Note that program PLANSET must be processed by program DECLARES before being executed.

INPUT FILES

The principal input to PLANSET consists of the indexed data base, INDEXDB OR INMODDB, generated by program INDEXER or program BASEMOD of the Data Input subsystem. Several user-option data cards are also accepted, which specify:

1. Vulnerability data to be used for blast damage calculations
2. The command and control reliability factor for each region in the plan
3. A request that missile retargeting be used for all missiles with reprogramming capability
4. A range multiplier RANGEMOD to be used when determining whether a weapon is sufficiently within range of a weapon group to be added as a group member
5. The SIDE for which a plan is to be generated
6. Names of each attacking weapon type
7. The maximum absolute difference in DBL probability MAXDBL that is allowed between the first and last weapons of a weapon group
8. The class name and value of an exemplar target for each class in the current plan

9. The values of the attribute TASK to be given priority when assigning the lead target of a complex

10. The alphabetic portion of the values of the attribute DESIG to be given priority when assigning the lead target of a complex

11. Print options allowing a print of the target list weapon group list, and/or complex target list.

OUTPUT FILES

Program PLANSET prepares the TINFILE (target input file) and WINFILE (weapon input file) to be used in program PREPALOC. The TINFILE (see table 2) contains a 29-word block of descriptive information for each target to be considered in the current plan. The targets are placed on the TINFILE in a random order which facilitates evaluation of the allocation process used in program ALOC. The target input file (TINFILE) includes three types of targets:

1. Simple target: one target element

2. Complex target: several target elements either exactly collocated or within the lethal radius of a single weapon (a one-megaton weapon is considered) so that they must be treated as a single target complex

3. Multiple targets: actually several independent identical targets such as separate missile silos in a Minuteman squadron that are close together (relative to the range of the weapon), but far enough apart that each target element must be treated as an independent aim point.

Each complex target is represented on the TINFILE in an aggregated form representing the total value of the complex as required by ALOC. This aggregated representation on TINFILE is paralleled by auxiliary detailed target data on WINFILE which includes a specific representation of each target element as a separate simple target. Similarly each multiple target is represented on TINFILE by a single representative target (of the appropriate multiplicity) as required by ALOC. This representative target is also paralleled by a list of specific coordinates for each target element in the auxiliary target data on WINFILE.

Table 2. TINFILE Format
(Sheet 1 of 2)

BLOCK TYPE	MAXIMUM LENGTH	VARIABLE OR ARRAY	DESCRIPTION
Header	5	TAPETYPE	Run or data base identification number
		DATE	Date of run initiation
		IDENTNO	Run identification number
		ISIDE	Side
		NTGTS	Number of targets in data base
Target Block 2 through (NTGTS+1)	29	TGTNAME	Target name
		INDEXNO	Target index number
		DESIG	Target designator code
		TASK	Target task code
		CNTRLOC	Target country location code
		FLAG	Target flag code
		TGTSTATUS1	Target Status: = 1 for simple target; = multiplicity for multiple target element; = complex number for complex target element
		TGTLAT	Target latitude
		TGTLONG	Target longitude
		TGTRAD	Target radius
		VTQ	Total original value of target
		M	Number of hardness components
		H1	Lethal radius first hardness component (1MT)
		H2	Lethal radius second hardness component (1MT)
		FVALH1	Fractional value of first hardness component

Table 2. (cont.)
(Sheet 2 of 2)

BLOCK TYPE	MAXIMUM LENGTH	VARIABLE OR ARRAY	DESCRIPTION
		NK	Number of time components
		FVALT1	Fractional value of first time component
		FVALT2	Fractional value of second time component
		TAU1	First time component
		TAU2	Second time component
		TAU3	Third time component
		IHCLASS	Target class name
		ICLASS	Target class number
		IHTYPE	Target type name (or number complex target elements if class is complex)
		TARDEF	State of local bomber defense
		MISDEF	State of terminal missile defense
		MINKILL	Minimum kill probability required
		MAXKILL	Maximum kill probability desired
		MAXCOST	Maximum (weapon cost/target value) acceptable to achieve MINKILL

Table 3 shows the format of the WINFILE. It lists 17 separate blocks of information. (The column "BLOCK" includes both a general descriptor which specifies the type data being output and a common block designator (shown parenthetically) which indicates the associated PLANSET common blocks. The variable/array names shown are those associated with this data in subsequent programs of the Plan Generator.) The first block contains header information which includes the run identification together with information on the size of the succeeding blocks. The second block contains the breakpoint tables which reflect the indexed structure of the game data base. The next six blocks contain point information which describes bomber routes, corridors, depenetration corridors, recovery bases, and directed refuel areas. The next block contains the descriptions of the air defense zone boundaries. The tenth block contains information on complex and multiple targets which augments the target data contained on the TINFILE; it contains information on each complex and multiple target element. The next six blocks provide data which define the offensive force. This includes the warhead, ASM (air-to-surface missile), and payload tables and weapon system information by region, type, group, and base. The final block contains information on the tanker units which support the bomber force.

CONCEPT OF OPERATION

Four intermediate files may be used by program PLANSET in processing the required weapon and target data.

1. Intermediate group file (LTGRP): Information pertaining to each weapon in a group and to each tanker is written onto the intermediate group file, LTGRP. For each weapon group element, LTGRP contains a six-word record in the format of array GRPX (IGRPX) in common block /3/. For each tanker squadron (item entry in ICLASS three) a 12-word record is written in the format of array TANK (LTANK) in common block /3/, preceded by a one-word code. The code is -1 for tankers which have been preassigned to refuel areas in the data base, and -4 for those which are to be automatically allocated. The end of file is signalled by a single word, ENDGROUP.

2. Intermediate target file (LTTGT): All target data are stored on the intermediate target file LTTGT. For each individual target (except elements of multiple targets), LTTGT contains a 31-word record corresponding to common block /TD/. For each multiple target, there is a record consisting of the 31 words corresponding to common block /TD/ followed by, for each member target, eight words in the format of array MLTX in common block /MLTX/. The target data end with a dummy target

Table 5. WINFILE Format
(Sheet 1 of 7)

BLOCK TYPE	MAXIMUM LENGTH/ ACTUAL LENGTH	VARIABLE OR ARRAY	DESCRIPTION
Header (WT)	1 (each) 22 total	TAPETYPE	Major run or data base identification
		DATE	Date of run initiation
		IDENTNO	Identification number
		ISIDE	Side
		NRTPT	Number of route points
		NCORR	Number of corridors
		NDPEN	Number of depenetration corridors
		NRECOVER	Number of recovery bases
		NREF	Number of refuel areas
		NBNDRY	Number of points in list describing zone boundary
		NREG	Number of regions
		NTYPE	Number of weapon types used in this plan
		NGROUP	Number of weapon groups
		NTOTBASE	Total number of weapon bases
		NPAYLOAD	Number of entries in payload index table
		NASMTYPE	Number of entries in ASM table
		NWHDTYPE	Number of entries in warhead table
		NTANKBAS	Number of tanker bases
		NCOMPLEX	Number of complex targets
		NCLASS	Number of offensive weapon classes (presently 2)
		NALERT	Number of alert conditions (presently 2)
		NCORTYPE	Number of corrider types

Table 3. (cont.)
(Sheet 2 of 7)

BLOCK TYPE	MAXIMUM LENGTH/ ACTUAL LENGTH	VARIABLE OR ARRAY	DESCRIPTION
Breakpoint Tables (DPOOL)	500	INDBEG(250)	Smallest index number for each type
		TYPENAME(250)	Type names in order of increasing index number
	MTARCLS 45	CUMNO(15)	Cumulative number of types in each class
		BTYPES(15)	Number of BLUE types in each class
		INDCLAS(15)	Smallest index number in each class
Corridor Type Characteristics (DPOOL)	25/(NCORTYPE X 5)	KORSTYLE(5)	Power of y versus x
		HILOATTR(5)	Ratio low to high altitude attrition (less than 1)
		DEFRANGE(5)	Characteristic range of corridor defense
		ATTRSUPE(5)	Suppressed high altitude attrition per nautical mile
		ATTRCORR(5)	Unsuppressed high altitude attrition per nautical mile
Corridor Point (DPOOL)	150/(NCORR X 5)	PCLINK(30)	Precorridor link
		PCLAT(30)	Latitude of corridor origin
		PCLONG(30)	Longitude of corridor origin
		PCZONE(30)	Defense zone in which corridor is located
		PCTYPE(30)	Corridor type
Route Point (DPOOL)	800/(NRTPT X 4)	RPLINK(200)	Route point link
		RPLAT(200)	Route point latitude
		RPLONG(200)	Route point longitude
		ATTRLEG(200)	Attrition in route point segment ending with this point

Table 3. (cont.)
(Sheet 3 of 7)

BLOCK TYPE	MAXIMUM LENGTH/ ACTUAL LENGTH	VARIABLE OR ARRAY	DESCRIPTION
Depenetration Corridor Point /DPOOL/	150/(NDPEN X 3)	DPLINK(50)	Depenetration corridor link
		DPLAT(50)	Depenetration corridor latitude
		DPLONG(50)	Depenetration corridor longitude
Recovery Point /DPOOL/	600/(NRECOVER X 3)	RECLINK(200)	Recovery point link
		RECLAT(200)	Recovery point latitude
		RECLONG(200)	Recovery point longitude
Refuel Point /DPOOL/	40/(NREF X 2)	RELAT(20)	Refuel point latitude
		RELONG(20)	Refuel point longitude
Boundary Point /DPOOL/	1000/(NBNDRY X 5)	BPLINK(200)	Boundary point link
		BPLAT(200)	Boundary point latitude
		BPLONG(200)	Boundary point longitude
		BPZONE(200)	Zone circumscribed by this link list
		NEXTZONE(200)	Zone exterior to line defined by this point and point to which it links
Complex, Multiple Target Data /TD/,(MLTX/	Variable in 29- and 8-word segments	TGTNAME : MAXCOST	29-word record as on TINFILE for each element of a complex target
			8-word record for each multiple target element; variables shown below
		TGTNAME	Target name
		INDEXNO	Target index number
		DESIG	Target designator code
		TASK	Target task code
		CNTRYLOC	Target country location code
		FLAG	Target flag code
		TGTLAT	Target latitude

Table 3. (cont.)
(Sheet 4 of 7)

BLOCK TYPE	MAXIMUM LENGTH/ ACTUAL LENGTH	VARIABLE OR ARRAY	DESCRIPTION
		TGTLONG	Target longitude
		ZZZZZZZZ	= 8HZZZZZZZZ end sentinel
Warhead Table /2/	150/(NWHDTYPE X 3)	YLD(50)	Yield
		PDUD(50)	Dud probability
		FFRAC(50)	Fission fraction
ASM Table /2/	100/(NASMTYPE X 5)	IWHDASM(20)	ASM warhead type
		RANGEASM(20)	ASM range
		RELASM(20)	ASM reliability
		CEPASM(20)	CEP of the ASM
		SPEEDASM(20)	ASM speed
Payload Table /2/	400/(NPAYLOAD X 10)	NOBOMB1(40)	Number of bombs of type 1
		IWHD1(40)	Type index of first bomb
		NOBOMB2(40)	Number of bombs of type 2
		IWHD2(40)	Type index of second bomb
		NASM(40)	Number of ASMs
		IASM(40)	ASM type
		NCM(40)	Number of countermeasures
		NDECOYS(40)	Number of decoys
		NADECOYS(40)	Number of area decoys
		IMIRV(40)	MIRV system identification number
Region /2/	20/NREG	CCREL(20)	Command and control reliability
Weapon Type /2/	1600/(NTYPE X 20)	IHWTYPE(80)	Weapon type name
		RANGE(80)	Weapon range (nautical miles)
		CEP(80)	Weapon CEP (averaged)
		SPEED(80)	Weapon speed (knots)
		ALERTDLY(80)	Weapon delay when on alert status

Table 3. (cont.)
(Sheet 5 of 7)

BLOCK TYPE	MAXIMUM LENGTH/ ACTUAL LENGTH	VARIABLE OR ARRAY	DESCRIPTION
		NALRTDLY(80)	Weapon delay when not on alert status
		RANGEDEC(80)	Weapon range decrement for low altitude flight
		ICLASS(80)	Weapon class index (1 for missiles, 2 for bombers)
		NOPERSQN(80)	Number weapons per squadron
		SPDHI(80)	Speed at high altitude
		SPDLO(80)	Speed at low altitude
		SPDASH(80)	Dash speed
		RANGEREF(80)	Weapon range with refueling (nautical miles)
		REL(80)	Weapon reliability
		NMPSITE(80)	Number of weapons per site
		IREP(80)	Reprogramming index
		IRECMODE(80)	Recovery mode index (1 for normal recovery probability, 0 for no recovery, -1 for low recovery probability)
		IPENMODE(80)	Penetration mode index (1 for normal use of corridors, 0 for corridors not used)
		ISIMTYPE(80)	Weapon type index used by the simulator
		FUNCTION(80)	Weapon function code (Hollerith)
Weapon Group and Base /GROUP/, /GRPX/	(200 blocks, each block = 14 + 5 (150))/ (NGROUP blocks, each block = 14 + 5 (NBASE))	NWPNS(200)	Total number weapons in entire group
		NVEHGRP(200)	Total number vehicles in group
		WLAT(200)	Latitude averaged
		WLONG(200)	Longitude averaged
		IREG	Regional index

Table 3. (cont.)
(Sheet 6 of 7)

BLOCK TYPE	MAXIMUM LENGTH/ ACTUAL LENGTH	VARIABLE OR ARRAY	DESCRIPTION
		ITYPE(200)	Type index used by Plan Generator
		IALERT(200)	Index alert status (1 for alert, 2 for nonalert)
		DBL(200)	Probability of destruction before launch
		IREFUEL(200)	Refueling area index (0 if none assigned, -1 for buddy refueling)
		YIELD(200)	Yield of bomb averaged
		ISTART(200)	Starting index number for group
		NBASE(200)	Number of bases in group
		IDBL(200)	Index to time-dependent DBL data tables
		PKNAV(200)	Single shot kill probability against naval targets
		IBASE(1)	Index of first base
		BASELAT(1)	Latitude of first base
		BASELONG(1)	Longitude of first base
		IPAYLOAD(1)	Payload index of first base
		VONBASE(1)	Number of first vehicle/Number per base
		IBASE(2)	Index of second base
		BASELAT(2)	⋮
		BASELONG(2)	
		IPAYLOAD(2)	
		VONBASE(2)	
		IBASE(3)	
		⋮	
		(NBASE)	

Table 3. (cont.)
(Sheet 7 of 7)

BLOCK TYPE	MAXIMUM LENGTH/ ACTUAL LENGTH	VARIABLE OR ARRAY	DESCRIPTION
Tanker /3/	600/(NTANKBAS X 12)	INDEXTK	Tanker index number
		TKLAT	Tanker base latitude
		TKLONG	Tanker base longitude
		IREFTK	Refuel area assigned to tanker base
		NPSQNTK	Number of tankers per squadron or base
		NALRTK	Number of alert tankers per base
		SPEEDTK	Tanker speed in knots
		DLYALTK	Alert delay
		DLYNTK	Nonalert delay
		TTOS	Total time on station
		ITYPETK	Tanker type index
		RANGE	Tanker range

record containing XXXXXXXX in word 2. Following the target data, the GRP(IGRP) array from common block /GROUP/ is written to enable the core storage it occupied to be reused. It is read back to core after the target data has been processed (in subroutine TGTSORT).

3. Sort files LSRTA and LSRTB: When the complex target print (see User's Manual, Volume II) is requested (optional) two sort files, LSRTA and LSRTB, are used to sort the data pertaining to complex targets. Each record is a 30-word copy of one column (J constant) of the array ICPLX(I,J) contained in common block /12/. The records are sequenced by complex number. The end of the file is signified by a dummy record containing 999999 in word 30.

PLANSET (flowchart shown in figure 2) begins processing the indexed data base by calling subroutine SKIPFILE and reading in the breakpoint tables from the end of INDEXDB (or INMODDB). The input data cards then are read and their information stored. The weapon type names specified on input data cards must be listed in the breakpoint tables (under TYPENAME); if no match is found, an error message is printed and the item is ignored. For all missile and bomber types appearing in this plan, an index (NTYPE) is assigned sequentially and stored in array LTYPE. For types which are not in the plan, the corresponding words of LTYPE are zero.

The first item is then read from INDEXDB (or INMODDB). It is assumed (a data base requirement) that items in classes WARHEAD, ASM, PAYLOAD, and DBLDATA precede potential targets for which INDEXNO has been defined. Type data for WARHEAD, ASM, PAYLOAD, and DBLDATA are retrieved and stored in the corresponding arrays, indexed by type. Geographic type data also is retrieved and stored in memory for items in classes POINT, CORRIDOR, and LEGS for the attacking side, and for items in classes POINT and BOUNDARY on the defending side.

Items in classes (ICLASS) 1 through 15 are separated into defending side (targets) and attacking side (weapons), and processed accordingly.

Target Processing

Targets for which the input class value is zero (i.e. an exemplar target is not defined for the class or the exemplar target is assigned a value of zero) are not to be included in the plan and hence are ignored in the processing. Otherwise, the data base attribute VAL (relative value within class) for each item is accumulated within its class. When the exemplar target specified by the data card for each class is encountered on the data base, a message is printed and the value factor

$$\frac{\text{data card value for exemplar target}}{\text{data base VAL for exemplar target}}$$

is calculated. After the entire data base has been read, the accumulated values (VALs), together with the value factors for each class, are used to compute the final normalized class value factors.

Multiple targets now are made up for missile sites which do not belong to a complex. A multiple target consists of at least two and no more than five consecutively indexed sites from the same squadron. Whenever an eligible missile site is encountered, a check is made to see if a multiple target is currently being processed. If not, one is started by retrieving the target data and storing it in the array MULT. The target counter (NTAR) then is incremented by one, as is the counter for the number of sites in the multiple target (NMULT), and the attributes NAME, INDEXNO, DESIG, TASK, CNTRYLOC, FLAG, LAT and LONG are stored in the array MLTX. If a multiple target was being processed, the new site is added by incrementing the site counter NMULT and storing the appropriate attributes in array MLTX. Subroutine WRMULT is called whenever a multiple target is to be terminated; this occurs if any of the multiple criteria fail.

Targets which are not missile sites are separated into individual targets and targets which belong to a complex. For individual targets, the target data are retrieved and immediately written on an intermediate target file (LTTGT). When a complex target is encountered, the number of targets in the complex is incremented and stored by complex in array NCPX. The maximum complex index (ICOMPLEX) is found and stored under MAXICOMP. Target data then are retrieved and stored in the target array, ITD. The first target in each complex is indicated by placing a "1" under ITD(30); for all other targets in the complex a "2" is placed in this word. After counters for the number of complexes (NCOMPLEX) and number of targets (NTAR) have been incremented, ICOMPLEX is stored under ITD(7) to distinguish the target as belonging to a complex, and the target data array is written onto the target file (LTTGT).

Weapons Processing

If the item being processed is the first site of a missile squadron (ISITE = 1) or a bomber on the attacking side, and if it is the first of its type to be processed (CHK(ITYPE) = 1), weapon type data which are the same for all weapons of the given type are retrieved and stored in the array WTP. In the case of a missile, however, before PLANSET fills the WTP array, it checks the retargeting flag (IRETARG). If on, the user has requested that the data base attribute IREP (reprogramming index) be considered for all missiles. PLANSET then calculates and stores for the current missile type the factors that later will be used to modify the number per squadron, number on alert, alert DBL probability, and reliability for all missiles of that type. The new values will reflect the type of reprogramming capability indicated by IREP. If the IRETARG flag is off, no modifications for retargeting capabilities are made

(these calculations are described in the Analytical Manual, Volume II, Chapter 2, Missile Reprogramming).

After WTP has been filled, missiles and bombers are aggregated to form weapon groups. A weapon group consists of weapons from up to 150 bases. If all the weapons on a given base are nonalert, weapons of the same type are considered as one group. Otherwise, a group is comprised of those weapons on a base which have the same alert status (IALERT), type (ITYPE), region (IREG), and naval data (IDBL, PKNAV). In addition, missiles must be carrying the same payload, and bombers must have the same refueling index (IREFUEL). The maximum number of warheads allowed per group is set at 1000.

Bomber units which do not refuel and missile sites must lie within a geographic region which, for alert weapons, has a radius equal to a certain percentage of the range of the weapon. This percentage is read into the variable RANGEMOD at the beginning of the program; if the percentage is not specified in the data cards, it is assumed to be 15%. For nonalert weapons, this distance criterion is automatically doubled.

In order to form a weapon group, the required radius is expressed in terms of latitude (DLAT) and longitude (DLONG), and the number of bases (NTOTBAS) is counted. If some bombers are to be used as tankers for refueling purposes (i.e., if IREFUEL = -2), the number in commission and the number on alert are cut in half. The number of weapons and total yield of the warheads carried by each vehicle on the base then are computed. Up to 200 groups can be formed for use in the Plan Generator. However, PLANSET processes and prints information for up to 210 weapon groups to enable the planners to adjust their data base should more than 200 groups be formed.

In addition, if the weapons have a time-dependent destruction before launch probability (DBL), then the spread in DBL between the first and last weapons must be less than the input parameter DMAXDBL. If the number of weapons on a base is sufficiently large that this criterion is not met, these weapons are split up. After the weapons are split, the program checks to determine if the weapons can be added to an existing group or if they must begin a new group.

When a new group is started group data are retrieved and stored in array GRP. The corresponding index to GRP and the attributes INDEXNO, LAT, LONG, and PAYLOAD are placed in the first five words of the array GRPX as each new base is added to the group. An index to vehicles on the base (ISTART) and the number of vehicles either on alert or in commission (NX) are packed into the sixth word of GRPX. The array is written immediately onto the intermediate group file (LTGRP). As each new base is added, the group centroid is adjusted accordingly. If there are both alert and

nonalert bombers on a given base, the alert bombers are tested for group assignment first using the distance criterion RANGEMOD; the nonalert bombers then are tested using the criterion 2 x RANGEMOD.

Tanker bases are not included in the group assignment. As each tanker base is encountered, tanker base data are retrieved and stored in the array TANK. If the tankers on the base are to be automatically allocated (IREFUEL = 0 or < -4), ITWORD is set to "-4"; for preassigned tankers (IREFUEL > 0), ITWORD is set to "-1". If IREFUEL = -3, an error message is printed and IREFUEL is reset to automatically allocate the tankers. For bases with IREFUEL = -1 or -2, the same error message is printed and the next data item is read. Otherwise the number of tanker bases (NTANKBAS) is incremented and array TANK, preceded by ITWORD, is written onto the group tape.

After all data base items have been processed, the total numbers of bomber refuels (NBOMB) and tankers (NTANK) are compared. If there are more bomber refuels than tankers, the bombers on the nonalert base with the largest range are changed to refuel once (IREFUEL = -4) if they originally refueled twice, or to nonrefuel (IREFUEL = 0) otherwise; NBOMB is decremented by the number of bombers on the base. This process continues until there are more tankers than bomber refuels. If IREFUEL is changed to zero for all the nonalert bases before the bomber-tanker balance is achieved, the alert bases are then examined and IREFUEL is changed as above. When the bombers and tankers have been balanced, the average yield per warhead for each group is computed, along with the index to the corresponding weapon array (WTP) and both are stored in the appropriate group array.

The intermediate target and group files (LTTGT and LTGRP) now are terminated, and a check is made to insure that all complexes have at least two targets. Then point data (route points, depenetration points, etc.) and tables are printed, and subroutine AROVRFLW is called to print messages indicating any array overflows which occurred during processing. Finally, PLANSET calls subroutine SHUFFLE which will control the reading of LTTGT and LTGRP, and the generation of TINFILE and WINFILE. When SHUFFLE returns, PLANSET terminates.

COMMON BLOCK DEFINITION

The common blocks used by program PLANSET are described in table 4.

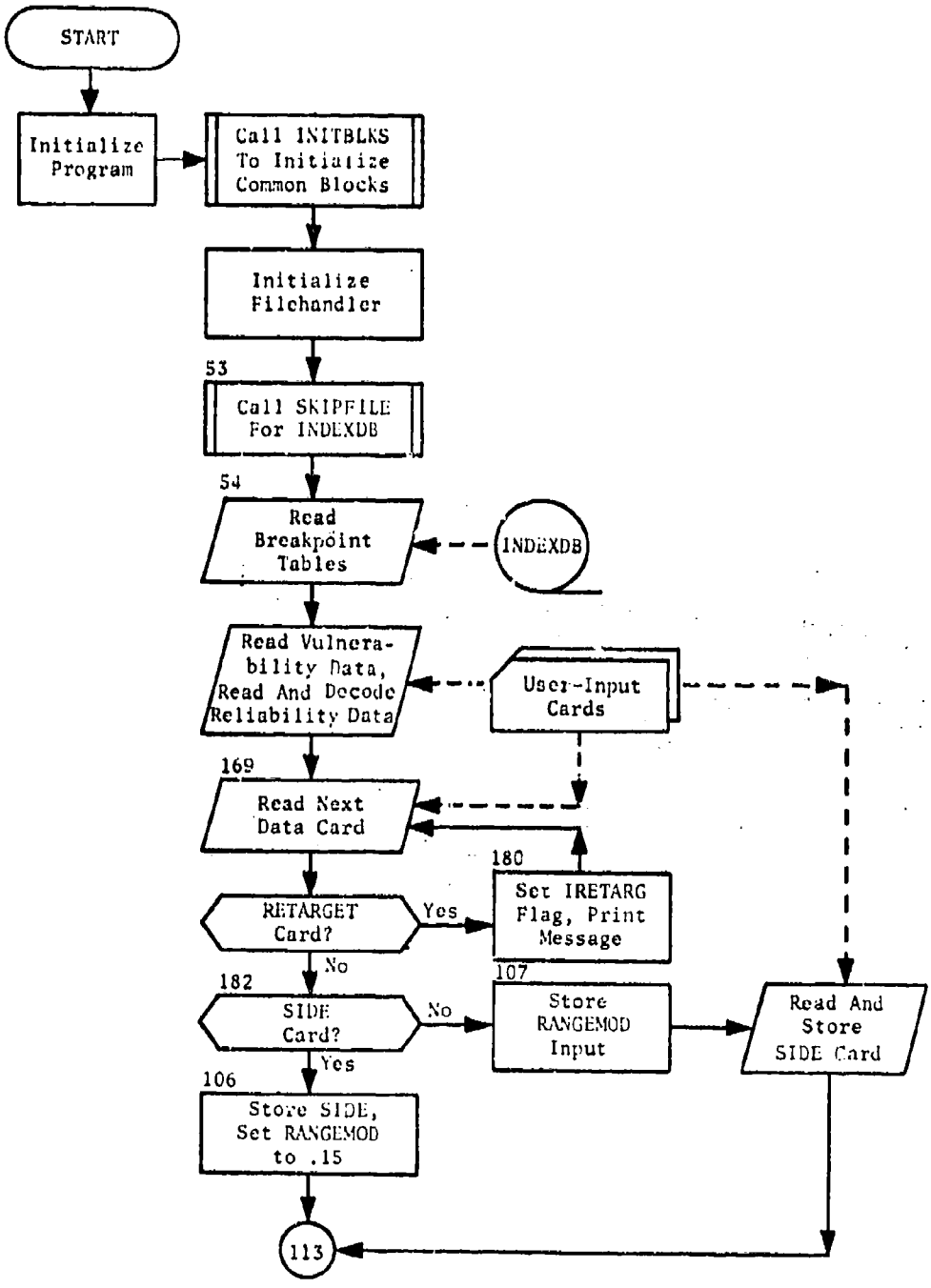


Fig. 2. Program PLANSET
(Sheet 1 of 20)

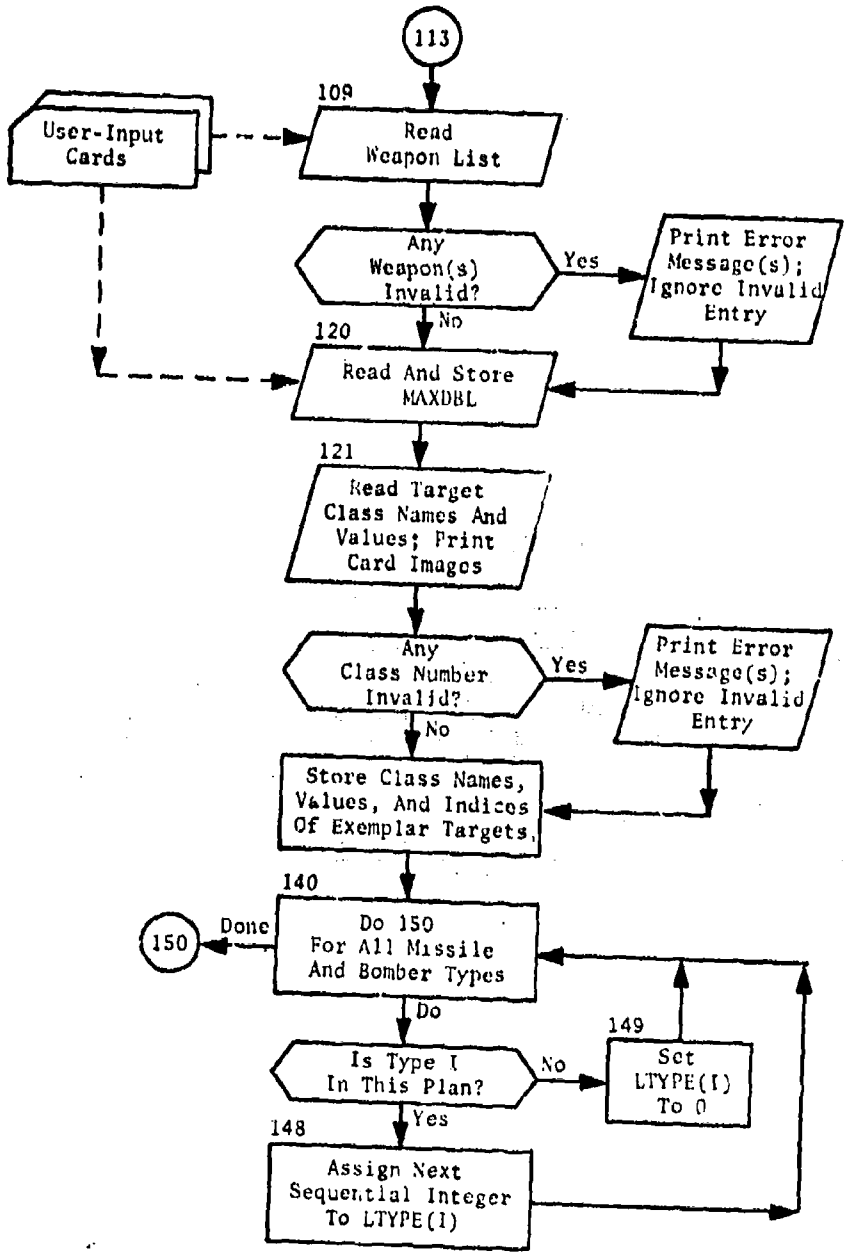


Fig. 2. (cont.)
(Sheet 2 of 20)

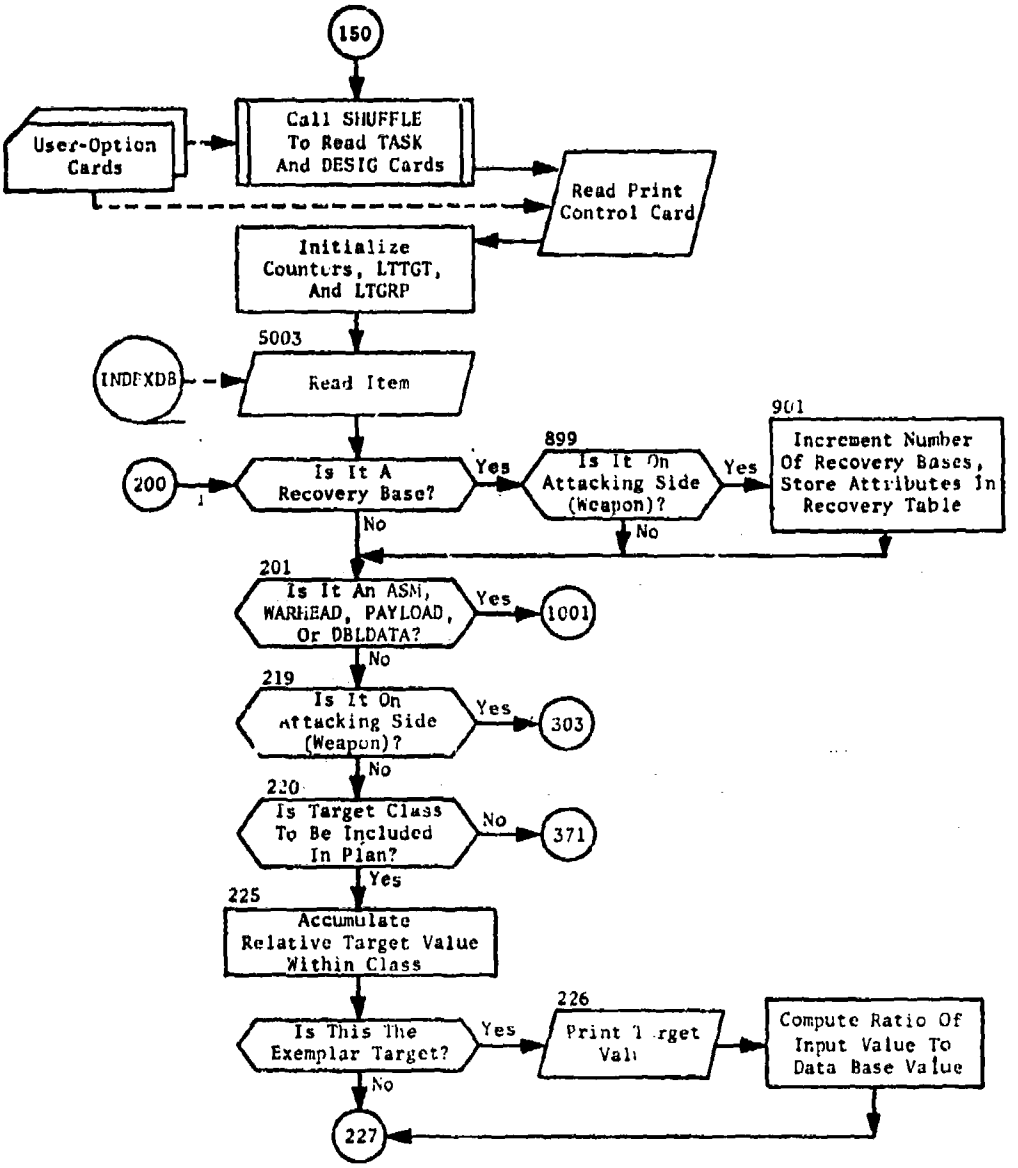


Fig. 2. (cont.)
(Sheet 3 of 20)

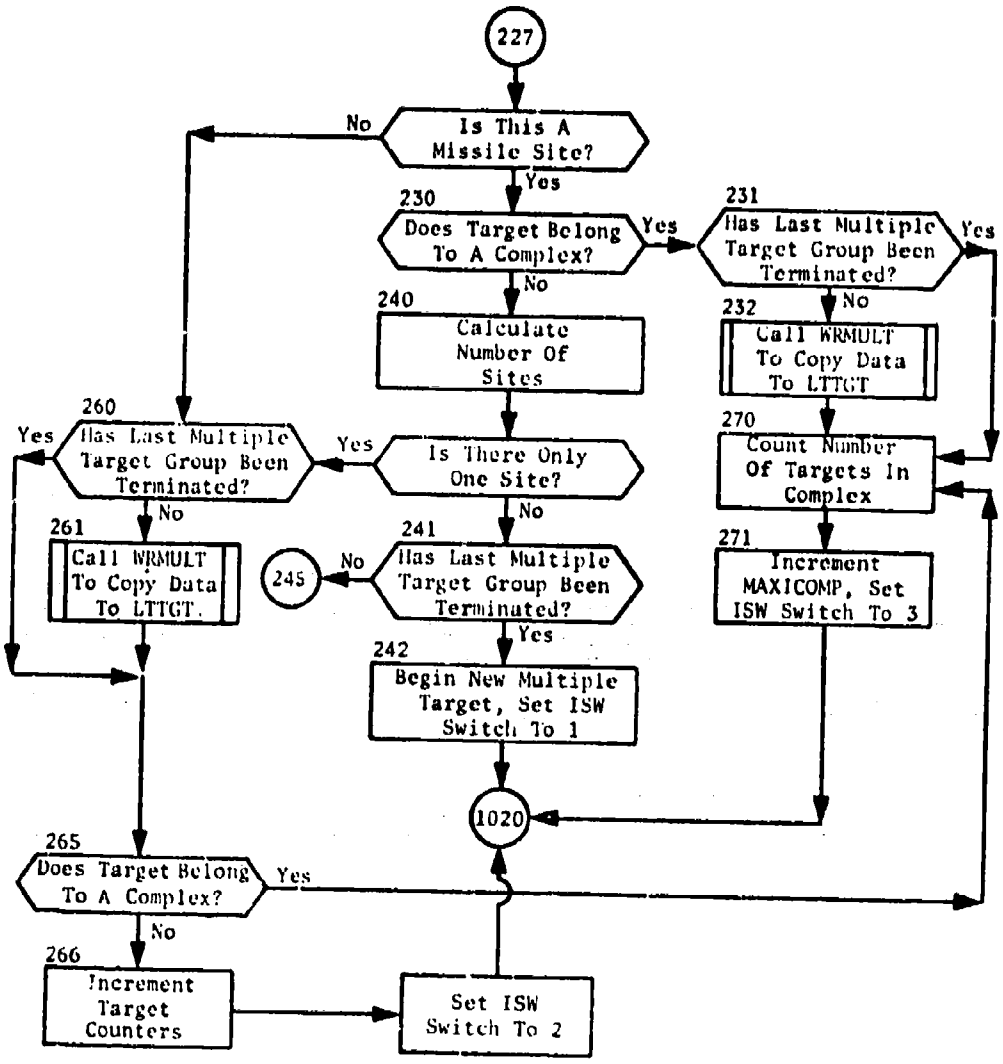


Fig. 2. (cont.)
(Sheet 4 of 20)

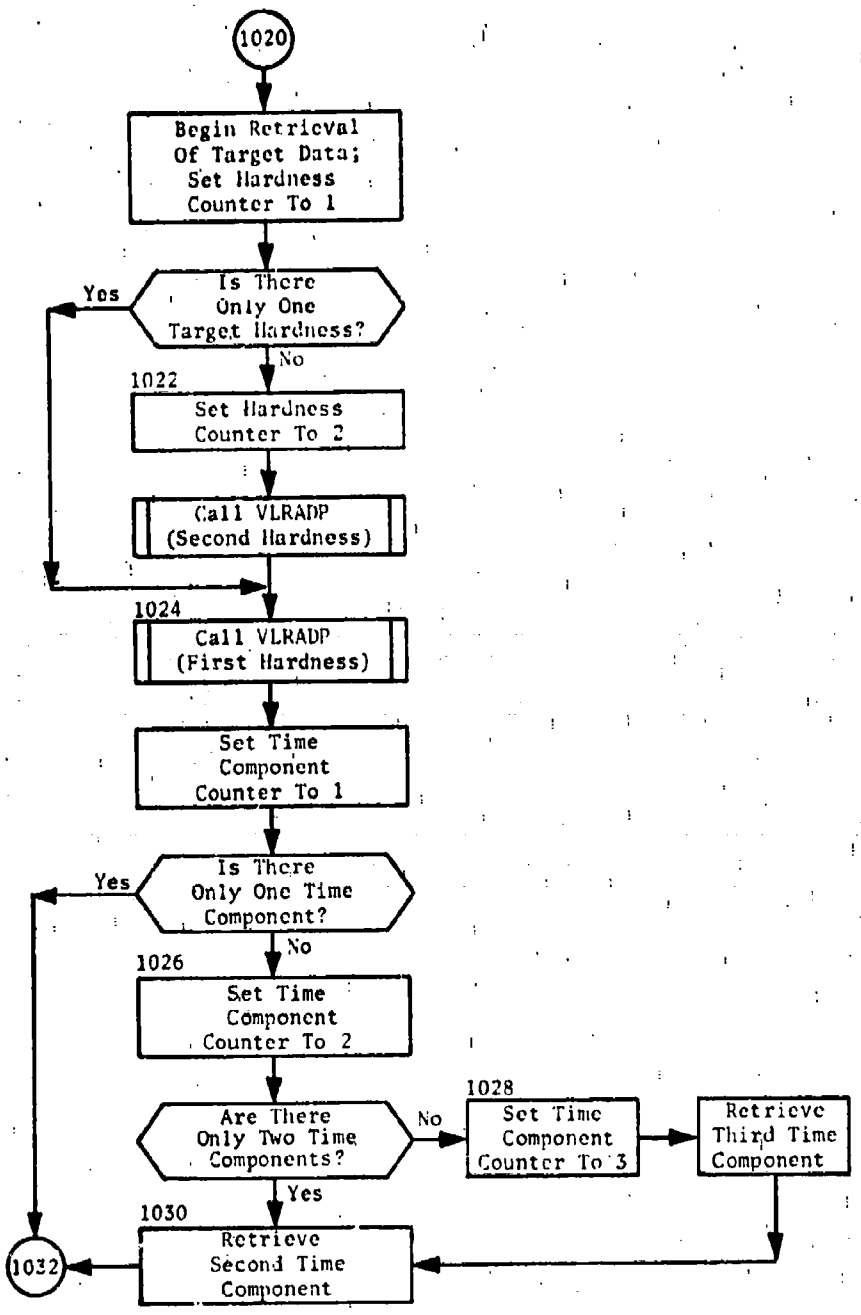


Fig. 2. (cont.)
(Sheet 5 of 20)

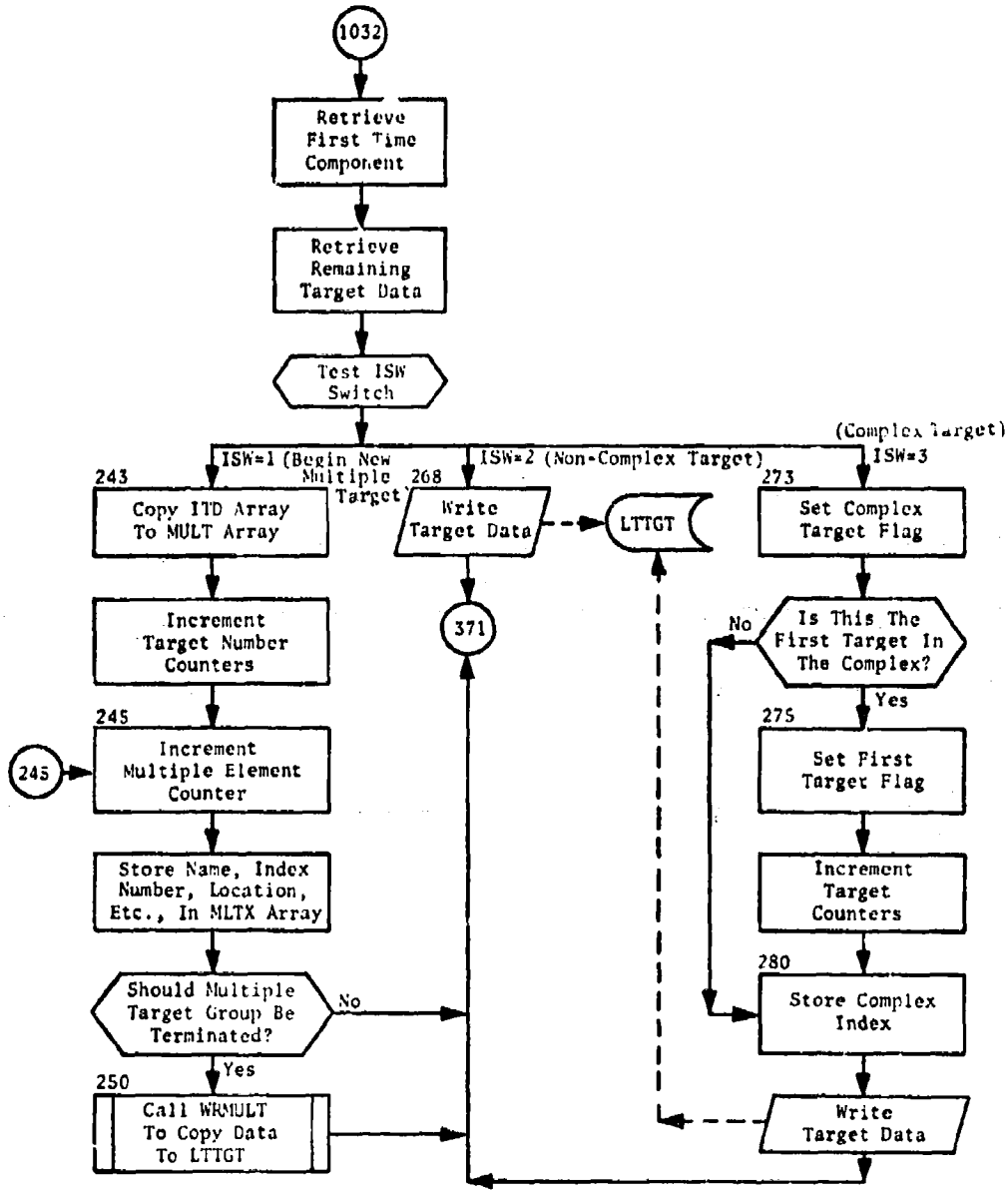


Fig. 2 (cont.)
Sheet 6 of 20)

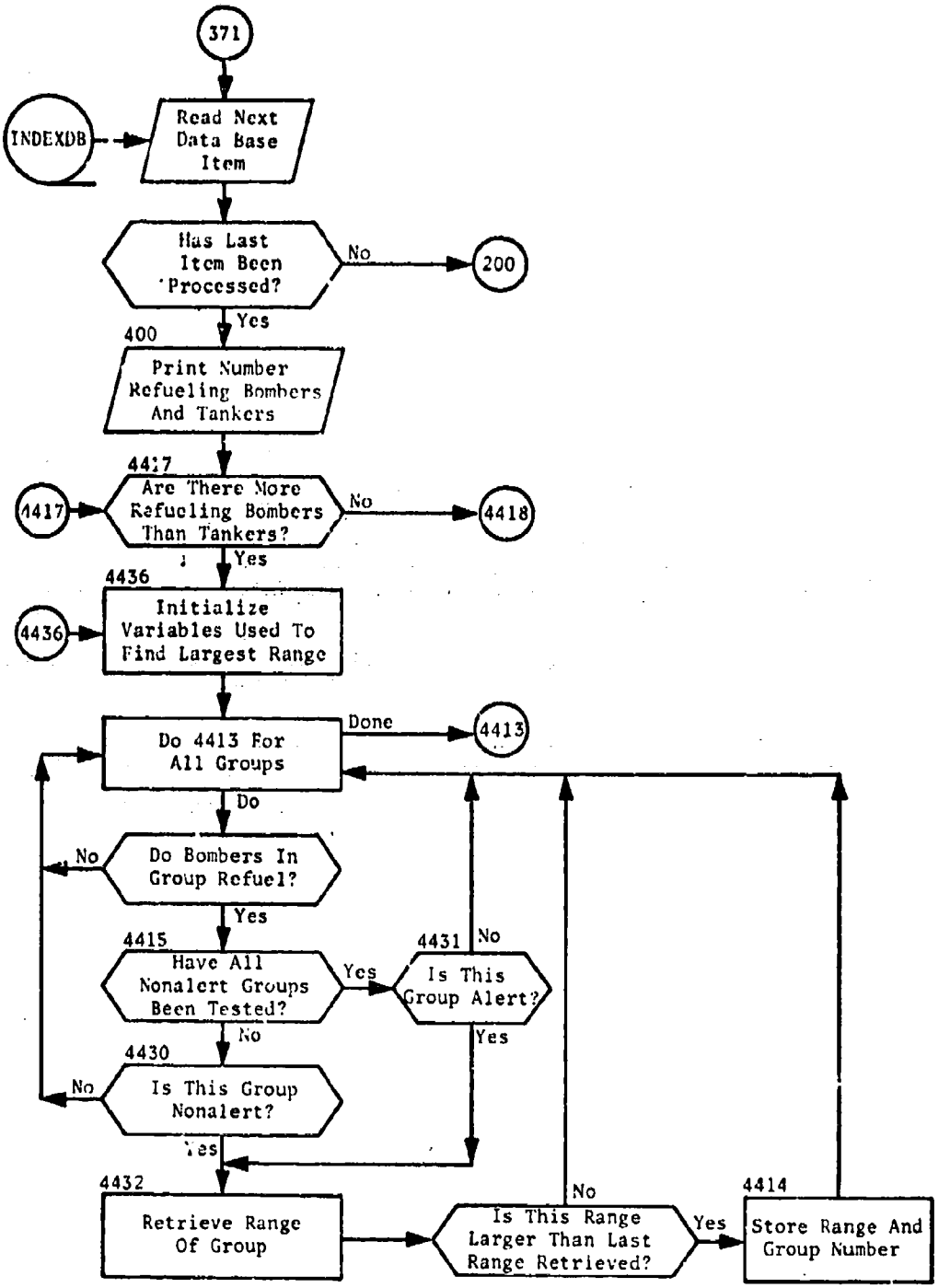


Fig. 2. (cont.)
(Sheet 7 of 20)

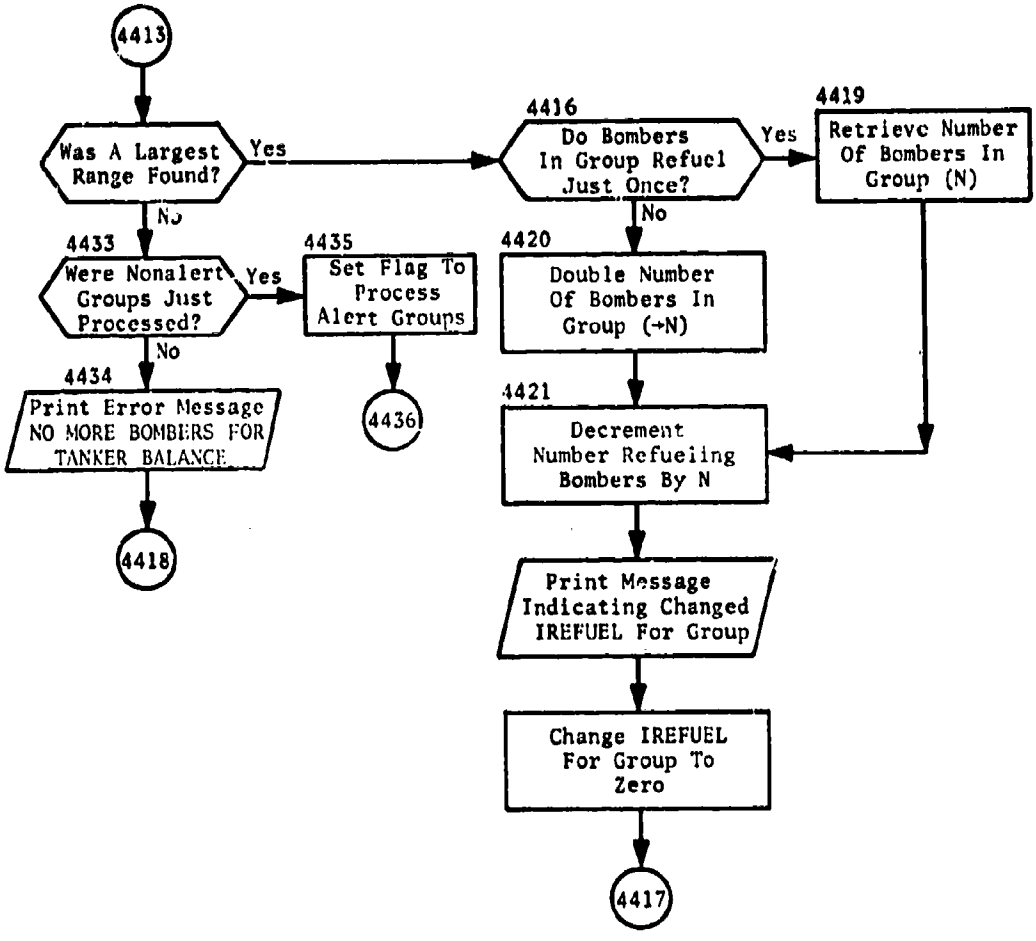


Fig. 2. (cont.)
(Sheet 8 of 20)

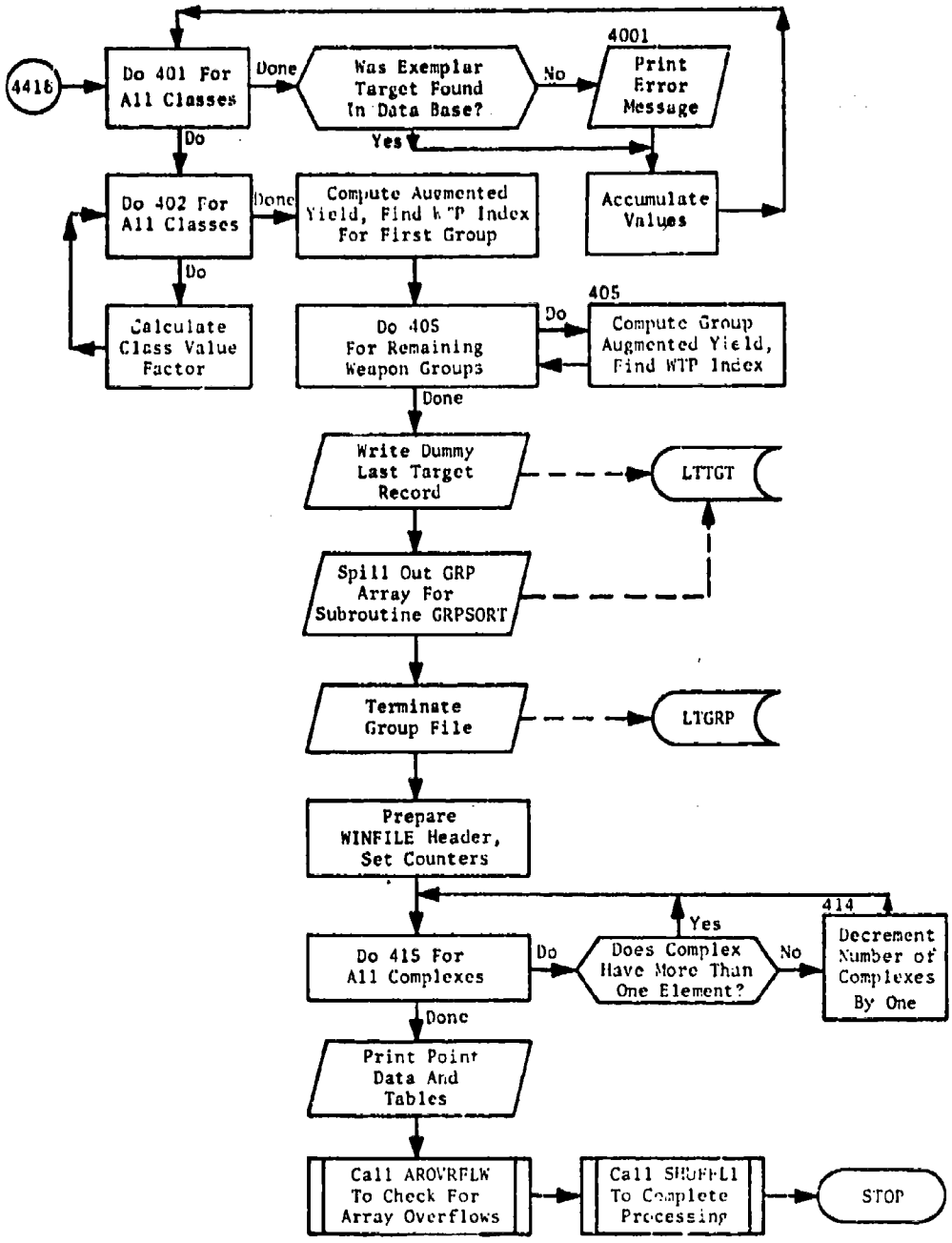


Fig. 2. (cont.)
(Sheet 9 of 20)

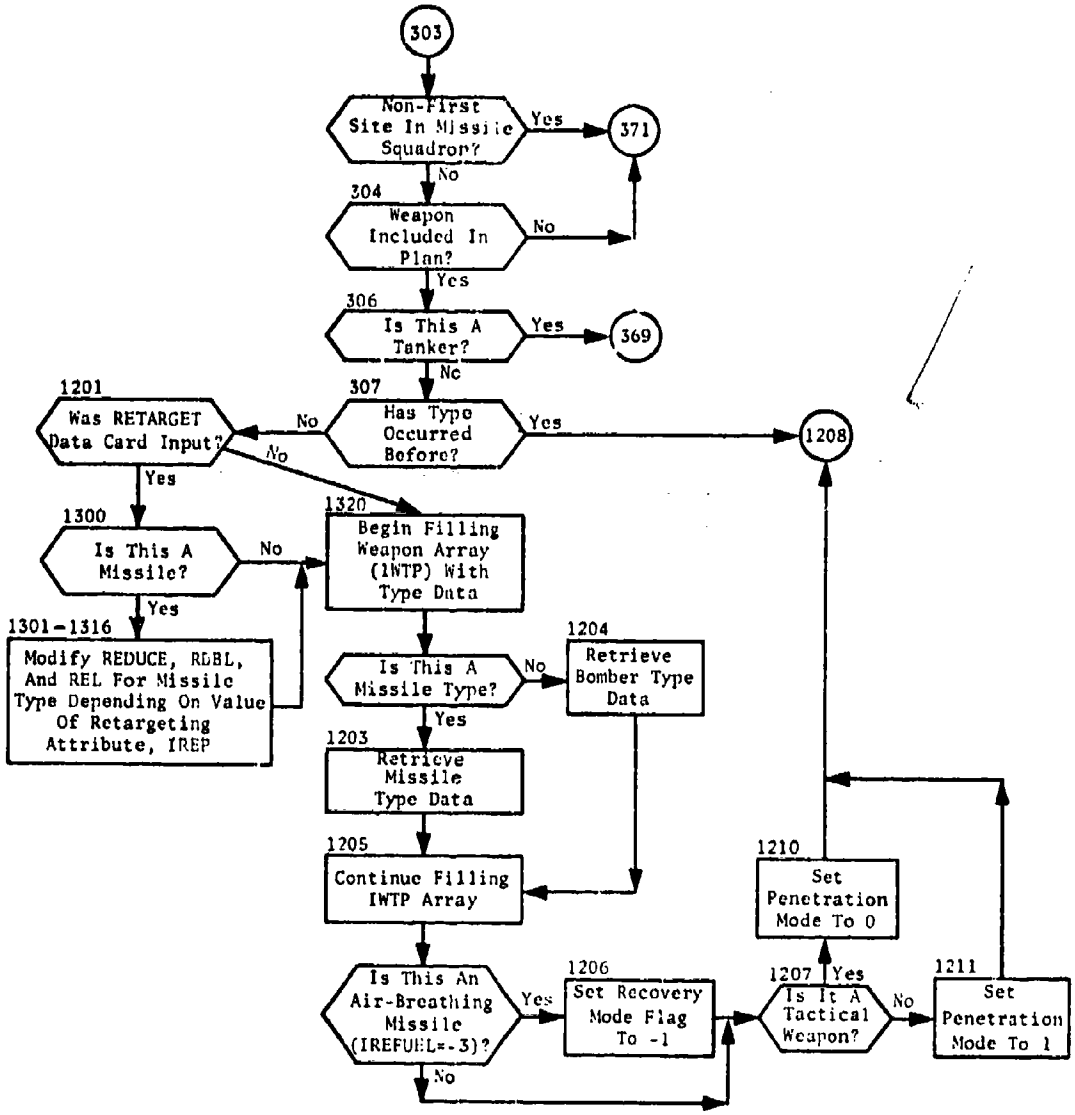


Fig. 2. (cont.)
(Sheet 10 of 20)

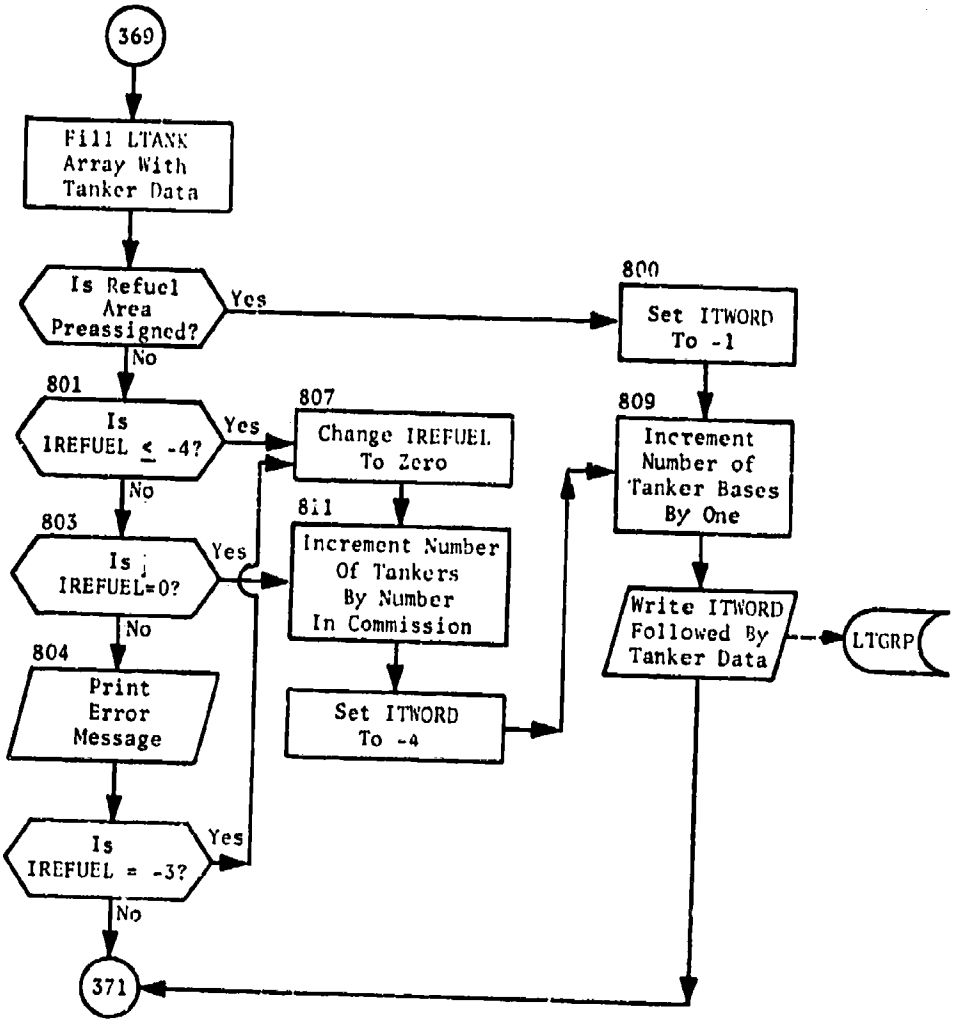


Fig. 2. (cont.)
(Sheet 11 of 20)

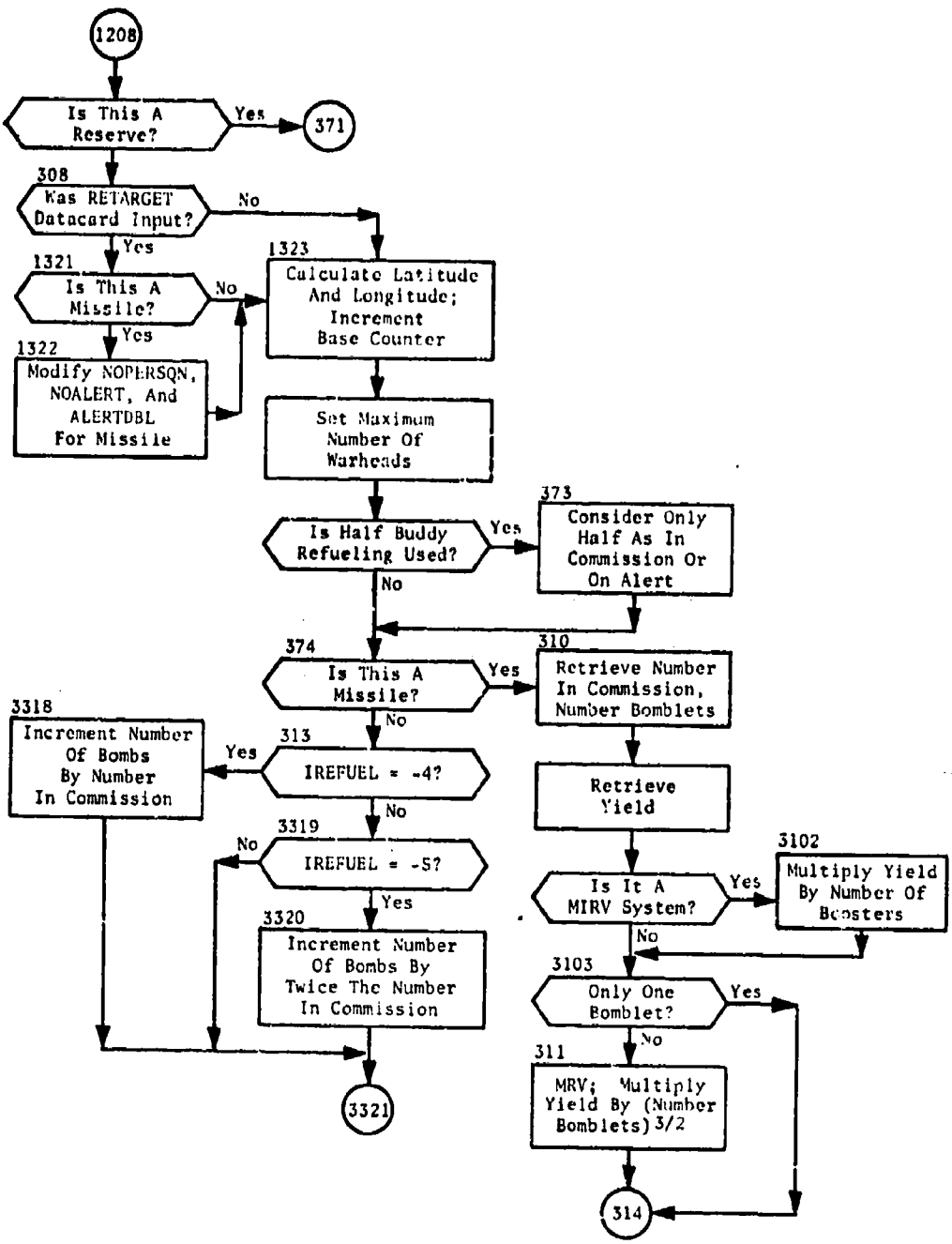


Fig. 2. (cont.)
(Sheet 12 of 20)

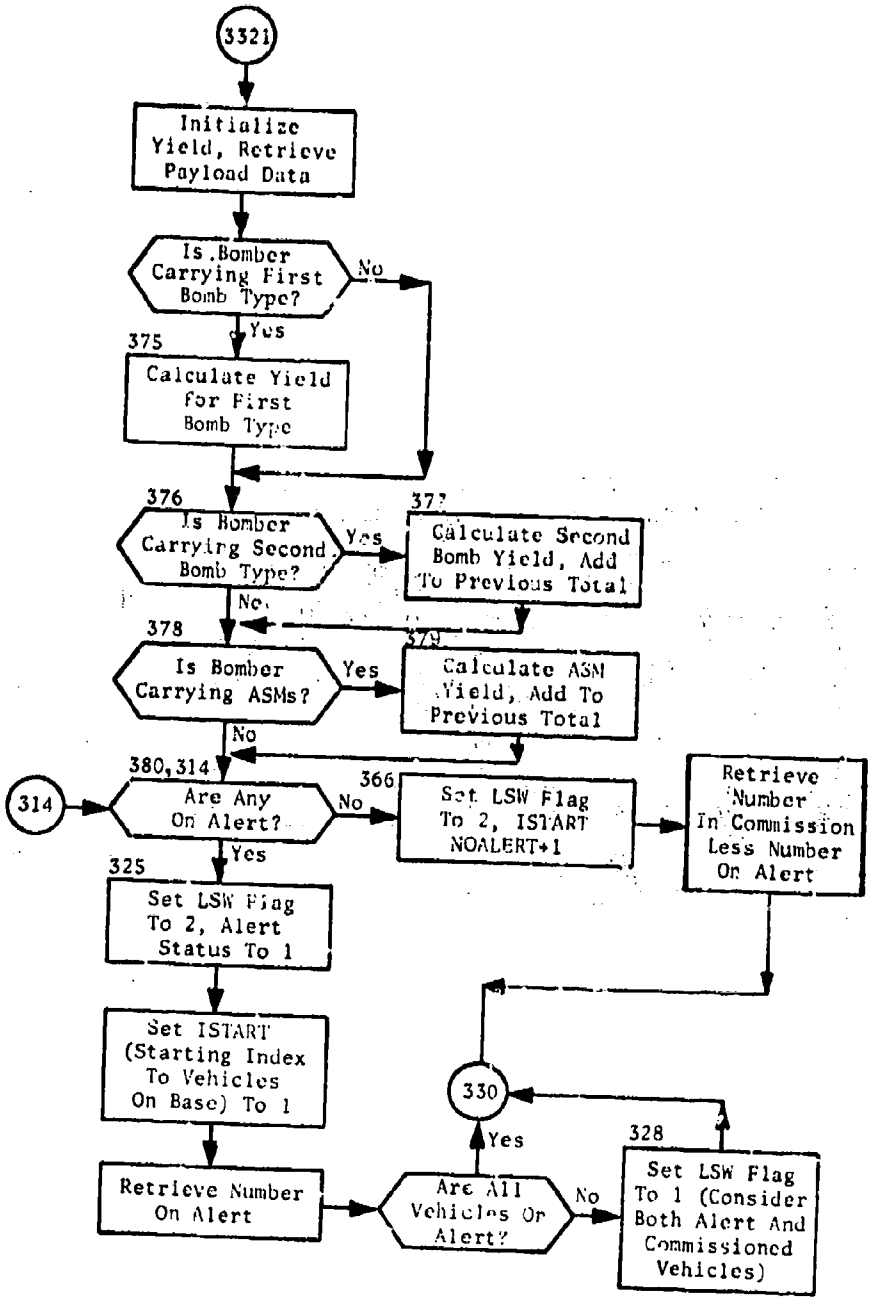


Fig. 2. (cont.)
(Sheet 13 of 20)

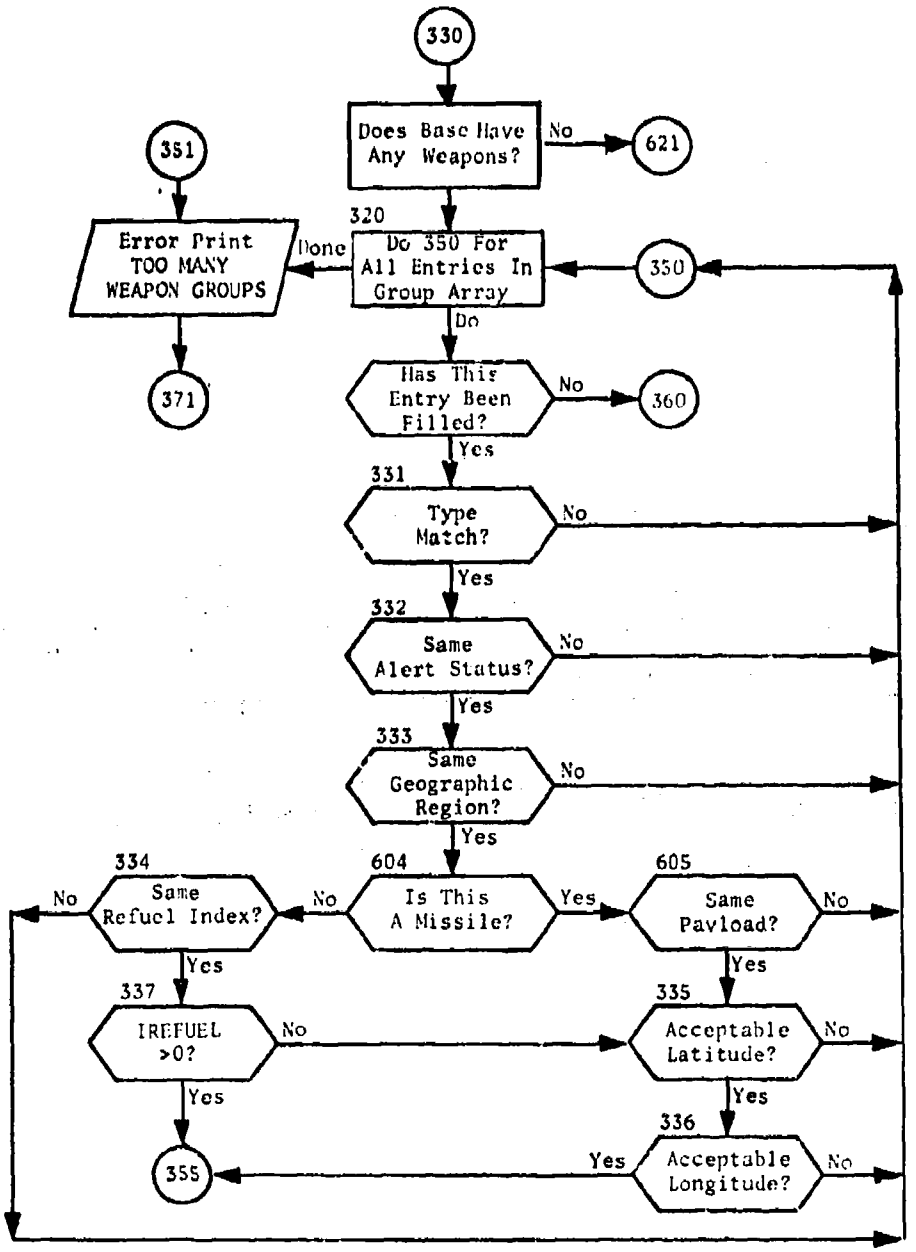


Fig. 2. (cont.)
(Sheet 14 of 20)

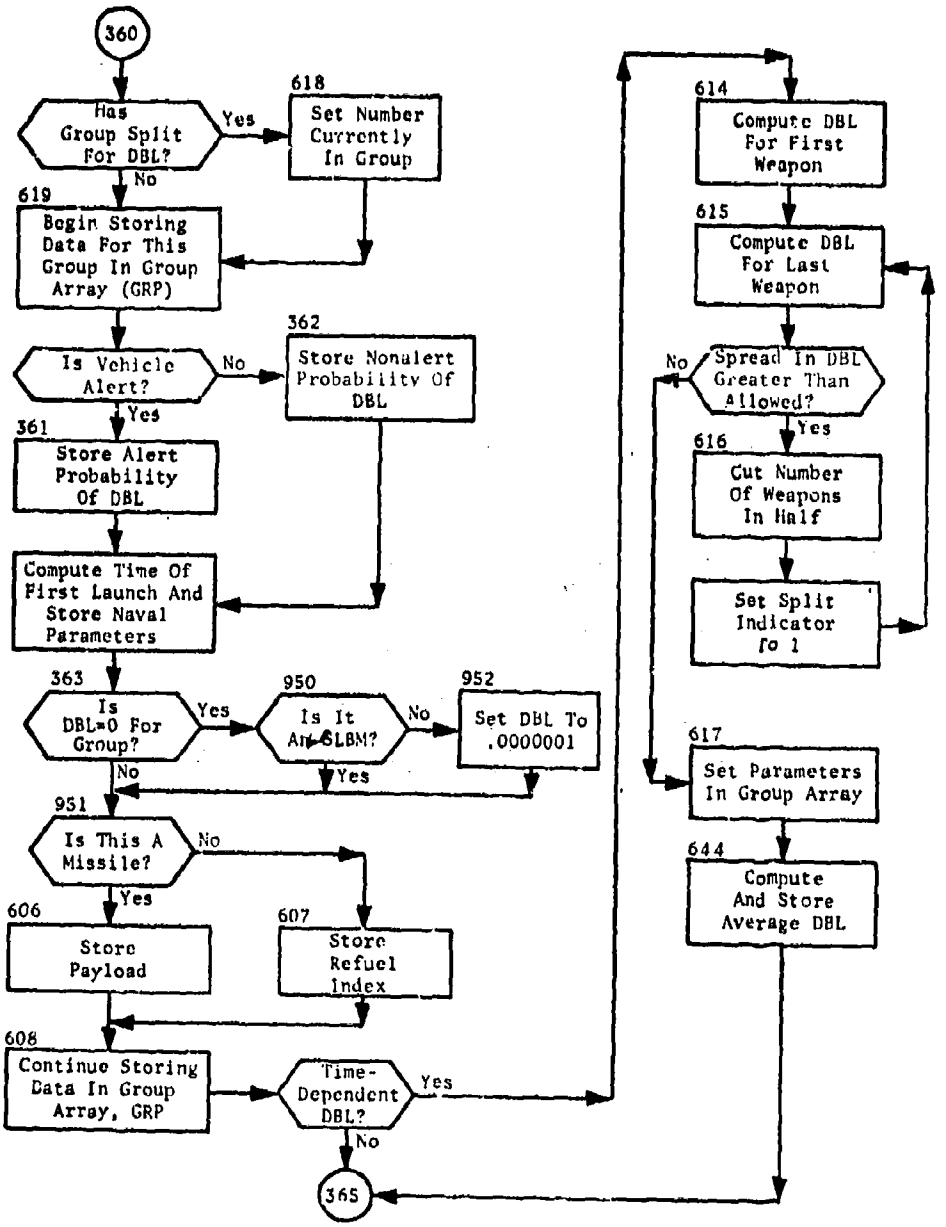


Fig. 2. (cont.)
(Sheet 15 of 20)

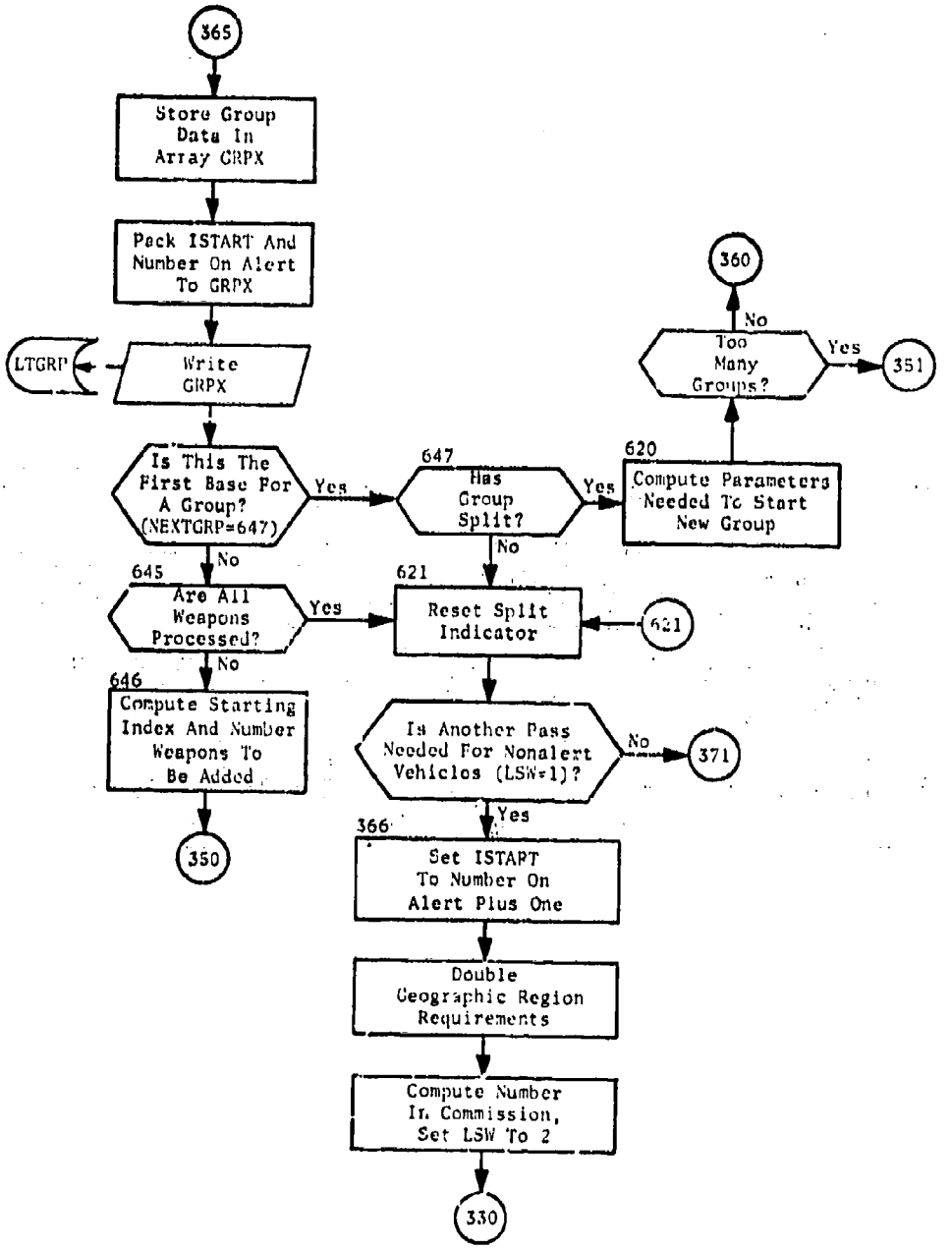


Fig. 2. (cont.)
(Sheet 16 of 20)

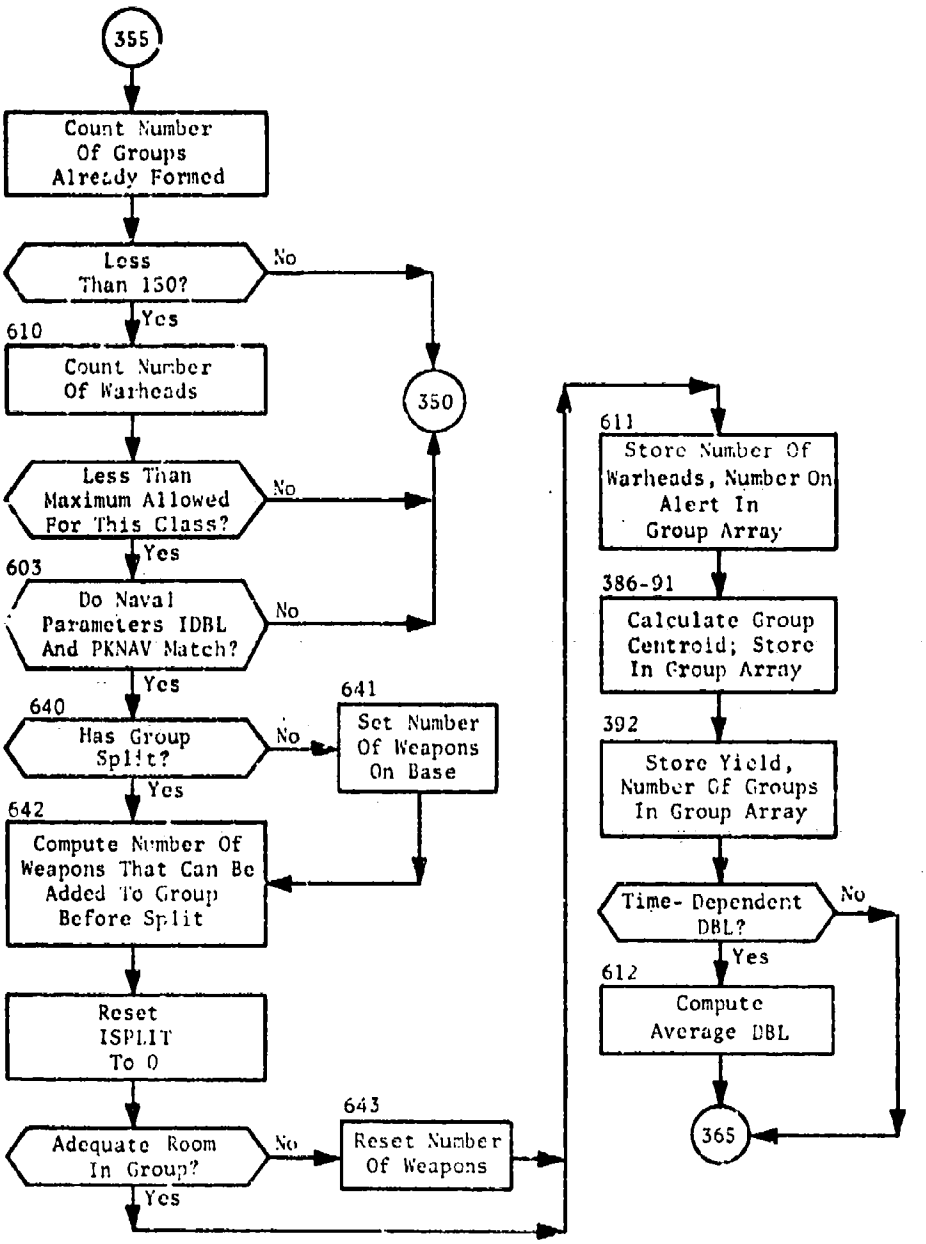


Fig. 2. (cont.)
(Sheet 17 of 20)

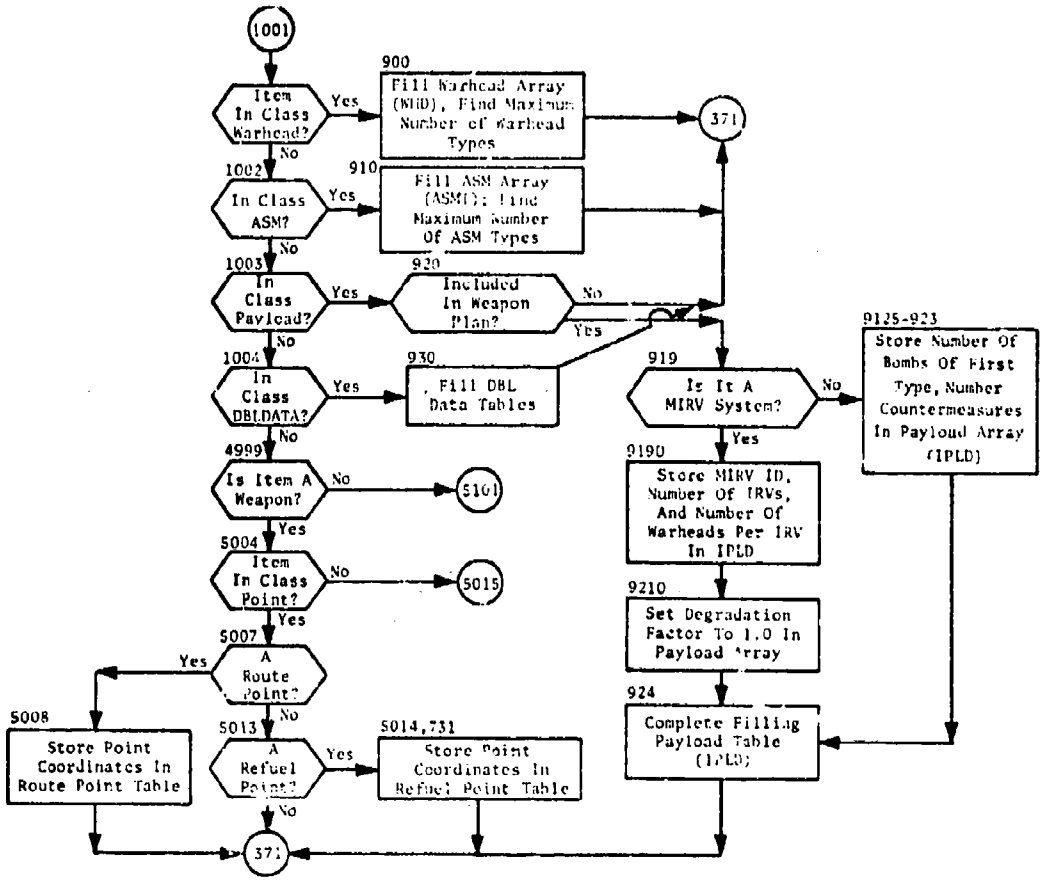


Fig. 2. (cont.)
(Sheet 18 of 20)

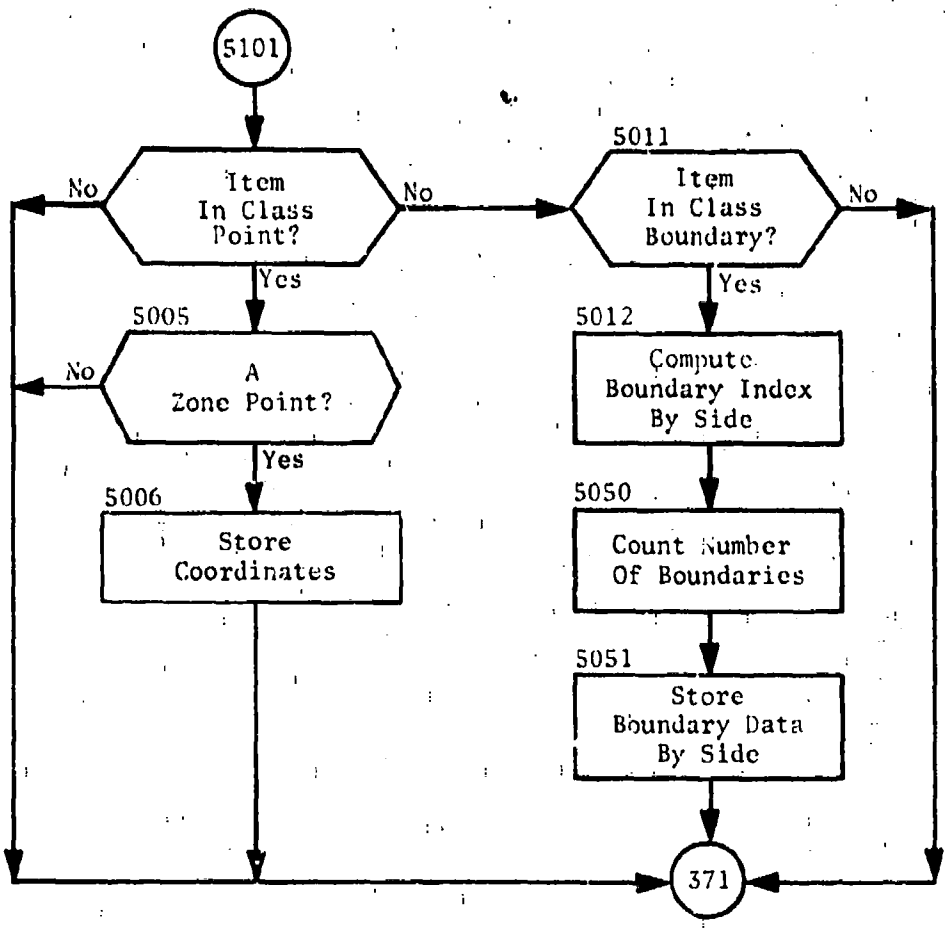


Fig. 2. (cont.)
(Sheet 19 of 20)

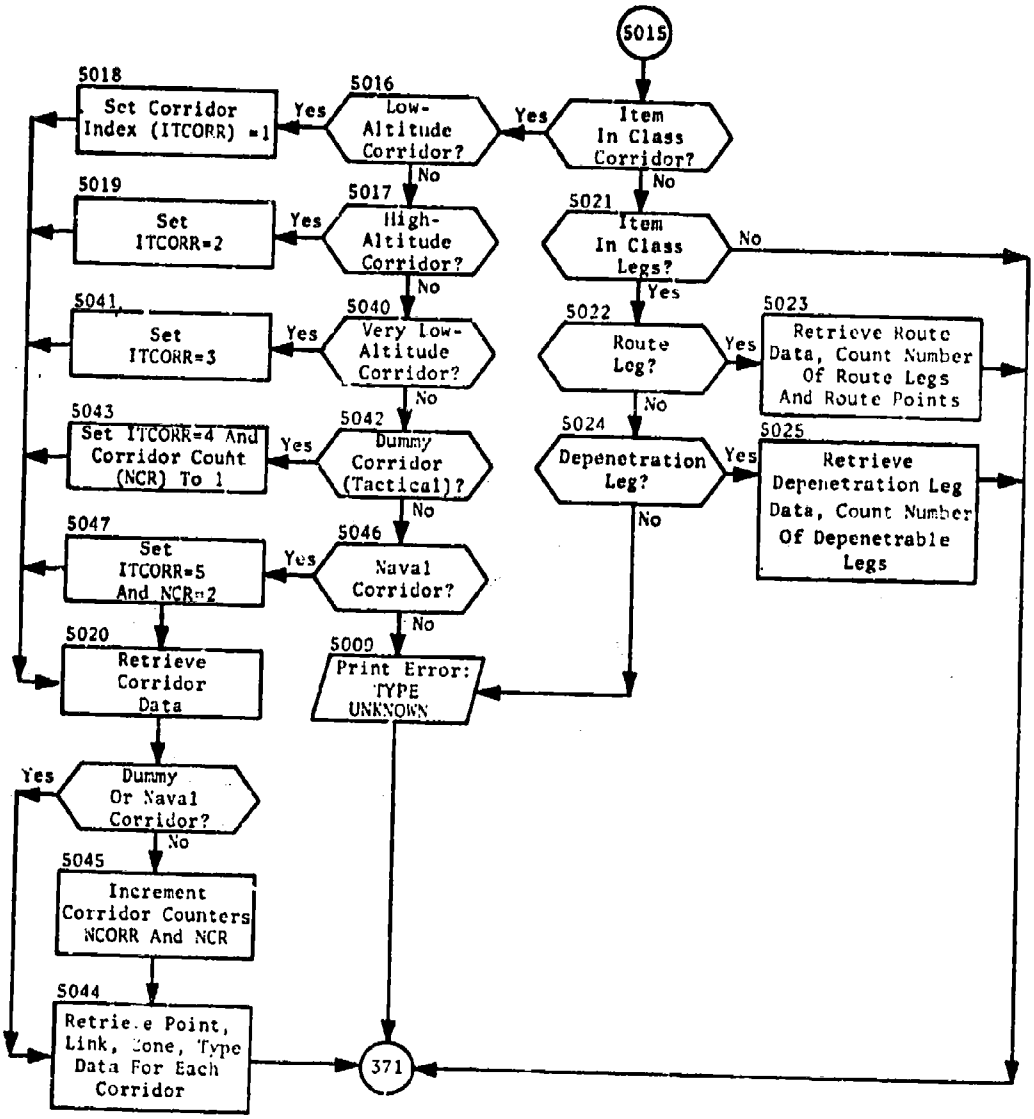


Fig. 2. (cont.)
(Sheet 20 of 20)

Table 4. Program PLANSET Common Blocks
(Sheet 1 of 14)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
CLASNAME	CLASNAME(15)	Hollerith name for Class I
	CLASVAL(15)	Relative value for Class I
	CUMVAL(15)	Total value for Class I
	VALFAC(15)	Fraction of total data base value for Class I to be applied in the current plan
DIRECTRY	MLTCOMP/ FMLTCOMP(I,J) (I = 8, J = 600)	Multiple target data array, where for target J: I = 1: target name 2: target index number 3: target designator code 4: target task code 5: target country location code 6: target flag code 7: target latitude 8: target longitude
	LOOK(30)	Hold area for LSRTA and LSRTB data
DPOOL	IB(200)	Index to point table for boundary point I
	LINKB(200)	Index of a leg linked to the current point (by side)
	ZONEB(200)	Defense zone enclosed by a set of linked boundary points
	NEXTZB(200)	Adjacent zone
	ICHKFLG(20)	Names of arrays which have overflowed
	ICHKNUM(20)	Number of items in overflowed arrays
	IC(30)	Index to point table for corridor point I
	LINKC(30)	Index of a leg linked to the current corridor point

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 4. (cont.)
(Sheet 2 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DPOOL (cont.)	ZONEC(30)	Defense zone in which corridor is located
	ITY(30)	Corridor type
	ID(50)	Index to point table for depenetration point I
	LINKD(50)	Index of a leg linked to the current depenetration point
	KORSTY(5)	Parameter to adjust mode of corridor penetration. Power of y versus x
	HILOAT(5)	Ratio low to high altitude attrition (less than 1)
	DEFR(5)	Characteristic range of corridor defense
	ATTRS(5)	Suppressed high altitude attrition per nautical mile
	ATTRC(5)	Unsuppressed high altitude attrition per nautical mile
	IL(200)	Index to point table for leg I
	LINKL(200)	Index to leg linked to current leg point
	ATRL(200)	Attrition parameter for leg ending with this point
	TMASW(10,10)	Time data for DBL data tables
	DBLASW(10,10)	Probability data for DBL data tables
	NTIMES(10)	Number of entries in Ith table
	TIMESTRT(200)	First launch time for group I
	NALLOW(200)	Maximum number of weapons that can be added to group I
	BLAT(200)	Zone latitude

Table 4. (cont.)
(Sheet 3 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DPOOL (cont.)	BLONG(200)	Zone longitude
	RLAT(200)	Latitude of route point I
	RLONG(200)	Longitude of route point I
	LINKR(200)	Index to the leg linked to recovery point I
	RECLAT(200)	Latitude of recovery base I
	RECLON(200)	Longitude of recovery base I
	IRECPCTY(200)	Capacity of recovery base I
	INDREC(200)	Index of recovery base I
	RFLAT(20)	Latitude of refuel point I
	RFLONG(20)	Longitude of refuel point I
	CUMNO(15)	Total number of types in Class I
	BTYPES(15)	Number of BLUE types in Class I
	INDCLAS(15)	Beginning index number for Class I
	INDBEG(250)	Beginning index number for type I
	TYPENAME(250)	A list of types in the order referred to by INDBEG
	NTYPS	Total number of missile and bomber types
	CHK(250)	Contains flags for types in this plan; corresponds to TYPENAME array
	PG(12)	Vulnerability data for VLRAD
	PA(12)	Vulnerability data for VLRAD
	QG(8)	Vulnerability data for VLRAD
	QA(8)	Vulnerability data for VLRAD
	ITEMP(5000)	Temporary array used to hold target numbers assigned during first pass in subroutine TGTSORT

Table 4. (cont.)
(Sheet 4 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DPOOL (cont)	ITANK(I,J) (I = 12, J = 200)	Each column J contains the 12 words of tanker squadron description as in array TANK of common /3/. The data stored is for tankers which are pre-assigned to refuel areas
	JTANK(I,J) (I = 12, J = 200)	Same as ITANK, except used for tankers to be automatically assigned to a refuel area by the plan generator
GROUP	GRP/IGRP(I,J) (I = 14, J = 210)	Data for weapon group J I = 1: total number weapons in group 2: total number vehicles in group 3: latitude averaged (group centroid) 4: longitude averaged (group centroid) 5: geographical region index 6: type index 7: alert status (1 = alert, 2 = nonalert) 8: DBL probability for alert vehicle 9: payload index for missiles, refuel index for bombers 10: yield of bombs averaged 11: starting index number for group 12: number of bases in group 13: index to time dependent DBL data table 14: single shot kill probability against naval targets
	INDGRP(210)	Group breakpoint table
	NWDSGRP	Number of words per group

Table 4. (cont.)
(Sheet 5 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
GROUP (cont.)	JTGT(2500)	The TGTSORT-assigned target number for elements in complex I
LCPX	LCPX(2500)	Index to ICPLX array for first element of complex I
	ICNUM(125)	List of complex numbers currently stored in ICPLX
	ICPNT(125)	The first column of ICPLX containing the complex number stored in ICNUM(I)
	ICNDX(125)	Indices to ICNUM in ascending complex number order
	LTYPE(250)	Weapon type number assigned by PLANSET. Corresponds to TYPENAME array in common block /DPOOL/
MAX (contains the QUICK maximum limits)	MAIRDEZ	Area ABM defense zones (20)
	MALERT	Alert conditions (2)
	MASMTYP	ASM types (20)
	MBNDRY	Boundary (200)
	MCCREGN	Command/control (20)
	MCLASS	Weapon classes (2)
	MCNTRYS	Country codes (250)
	MCORR	Penetration corridors (30)
	MCORTYP	Corridor types (5)
	MDPEN	Depenetration corridors (points) (50)
	MDEPNLG	Depenetration legs (50)
	MGROUP	Weapon groups (200)
	MPAYLOD	Payload types (per side) (40)
	MRECOVR	Recovery bases (points) (200)
	MRECVLG	Recovery legs (60)
	MREF	Refuel points (directed) (20)

Table 4. (cont.)
(Sheet 6 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
MAX (cont.)	MRTLEG	Route legs (200)
	MRTPT	Route points (200)
	MSPERMT	Sites per multiple target (5)
	MTANKBS	Tanker bases (50)
	MTARCLS	Target classes (15)
	MTARCOL	Targets, collocated (4000)
	MTARCPX	Target complexes (total) (4000)
	MTARERS	Targets per collocation island (100)
	MTARGET	Targets (allocator) (5000)
	MTARIND	Target index numbers (12000)
	MTARSEC	Targets per earth sector (4000)
	MTARTEI	TGTS with terminal interceptors (ABM) (500)
	MTARTYP	Target types (total) (250)
	MTARVAL	Target complex with value > 0 (2500)
	MTELMCM	Target elements per complex (40)
	MTOTBAS	Weapon bases per group (150)
	MTYPE	Weapon types (missiles + bombers per side) (80)
	MVULN	Unique target vulnerabilities within the game base (50)
	MWEAPGP	Weapons per group (missiles + bombers) (1000)
	MWHDTPE	Warhead types (50)
	MZONEPT	Zone points (200)
	MZONES	Zones (63)
	MTARPCL	Target types per class (40 = missiles + bombers/20 = others)
MISC	IN/FIN(5)	Array to which exemplar target input option cards are read

Table 4. (cont.)
(Sheet 7 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
MISC (cont.)	ITTAPE(5)	Array containing data for TINFILE header record
MLTX	MULT(31)	Array containing multiple target data. Entries correspond to those for common block /TD/
	MLTX/FMLTX(I,J) (I = 8, J = 5)	I words of data for the Jth multiple target element I = 1: target name 2: target index number 3: target designator code 4: target task code 5: target country location code 6: target flag code 7: target latitude 8: target longitude
	NMULT	Number of elements in the current multiple target
PRCNTL	JJJTGTS	Contents of target entry on print option card
	JJJGP	Contents of group entry on print option card
	JJJCPX	Contents of complex entry on print option card
PRIOR	IPTASK(48)	Array into which TASK priority option cards are read
	IPDES(96)	Array into which DESIG priority option cards are read
	MPTASK	Number of TASKs in IPTASK array
	MPDES	Number of DESIGs in IPDES array
	ISUBT	Task flag: If 0, only 1 character in TASK
RETARG	REDUCE(40)	The factor to be applied to NOPERSQN for missile type I when retargeting is considered

Table 4. (cont.)
(Sheet 8 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
RETARG (cont.)	RDBL(40)	The new ALERTDBL for missile type I when retargeting is considered
TAPES	LTWIN	Logical unit for WINFILE
	LTTIN	Logical unit for TINFILE
	LTTGT	Logical unit for intermediate target file
	LTGRP	Logical unit for intermediate group file
	LTDB	Logical unit for INDEXDB input
	LSRTA LSRTB	Logical units for sorting complex target information for sequenced printout
TAU	TAU(90)	Time components of all elements of a complex
	HC(60)	Hardness components of all elements of a complex
	V(90)	Values corresponding to TAU or HC
	INDEXT(90)	Array containing ordered indices
	FV(3)	Fractions of total value corresponding to V
	T(3)	Array containing time components for single target
	TBOX(3)	Not used
	VBOX(3)	Not used
	H(2)	Array containing hardness components for single target
	FVH(2)	Array containing fractions of values lost for single target

Table 4. (cont.)
(Sheet 9 of 14)

BLOCK	VARIABLE OR ARRAY		DESCRIPTION
TD	TD/ITD(31)		Target data array, where the information contained for each target is:
		TGTNAME	I = 1: target name
		INDEXNO	2: target index number
		DESIG	3: target designator code
		TASK	4: target task code
		CNTRYLOC	5: target country location code
		FLAG	6: target flag code
		TGTSTATUS1	7: target status =1 for simple target, =multiplicity for a multiple target element, =complex number for a complex target element
		TGTLAT	8: target latitude
		TGTLONG	9: target longitude
		TGTRAD	10: target radius
		VTO	11: total original value of target
		M	12: number of hardness components
		H1	13: lethal radius first hardness component (1MT)
		H2	14: lethal radius second hardness component (1MT)
		FVALH1	15: fractional value of first hardness component
		NK	16: number of time components
		FVALT1	17: fractional value of first time component
		FVALT2	18: fractional value of second time component
		TAU1	19: first time component
		TAU2	20: second time component
		TAU3	21: third time component
		IHCLASS	22: target class name
		ICLASS	23: target class number
		IHTYPE	24: target type name (or number complex target elements if class is complex)

Table 4. (cont.)
(Sheet 10 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
TD (cont.)	TARDEF	25: state of local bomber defense
	MISDEF	26: state of terminal missile defense
	MINKILL	27: minimum kill probability required
	MAXKILL	28: maximum kill probability desired
	MAXCOST	29: maximum (weapon cost/ target value) acceptable to achieve MINKILL
	TGTSTATUS2	30: target status =0 if single or multiple target, =1 if lead element of a complex target, =2 if additional element of a complex target
	TGTNUM	31: target number assigned by subroutine TGTSORT
WT	WT	Dummy - filled as needed
	IDATE	Hollerith date
	IDENTNO	Data input identification number
	LSIDE	Side
	NRTPT	Number of route points
	NCORR	Number of corridors
	NDPEN	Number of depenetration corridors
	NRECOVER	Number of recovery points
	NREF	Number of refuel points
	NBNDRY	Number of boundary points
	NREG	Number of regions
	NTYPE	Number of weapon types
	NGROUP	Number of weapon groups
	NTOTBASE	Total number of bases

Table 4. (cont.)
(Sheet 11 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
WT (cont.)	NPAYLOAD	Number of entries in payload index table
	NASMTYPE	Number of ASM types
	NWHDTYPE	Number of warhead types
	NTANKBAS	Number of tanker bases
	NCOMPLEX	Number of complex targets
	NCLASS	Number of weapon classes
	NALERT	Number of alert conditions
	NCORTYPE	Number of corridor types
1	MAXICOMP	Total number of complex targets
	NCPX(2500)	Number of elements in complex I
2	WHD/IWHD(I,J) (I = 3, J = 50)	Warhead table, where for each warhead type J: I = 1: yield 2: dud probability 3: fission fraction
	ASMT/IASMT(I,J) (I = 5, J = 20)	ASM table, where for each ASM type J: I = 1: ASM warhead type 2: ASM range 3: ASM reliability 4: CEP of the ASM 5: ASM speed
	PLD/IPLD(I,J) (I = 10, J = 40)	Payload table, where for each payload type J: I = 1: number of bombs of type 1 (for MIRVs, the number of IRVs) 2: type index of first bomb 3: number of bombs of type 2 4: type index of second bomb 5: number of ASMs 6: ASM type

Table 4. (cont.)
(Sheet 12 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
2 (cont.)		I = 7: number of countermeasures (bombers). Degradation factor (missiles) 8: number of decoys (for MIRVs, the number of terminal decoys per IRV) 9: number of area decoys 10: MIRV system identification number
	RGN(20)	Reliability factors for each command-and-control region I
	WTP,IWTP(I,J) (I = 20, J = 80)	Weapon table, where for each weapon type J: I = 1: weapon type name 2: weapon range (nautical miles) 3: weapon CEP (averaged) (nautical miles) 4: weapon speed (knots) 5: weapon delay before launch when on alert status (hours) 6: weapon delay before launch when not on alert status (hours) 7: weapon range decrement for low altitude flight 8: weapon class index (1 for missiles, 2 for bombers) 9: number weapons per squadron (missiles), number in commission (bombers) 10: speed at high altitude (knots) 11: speed at low altitude (knots) 12: dash speed (knots) 13: weapon range with refueling (nautical miles)

Table 4. (cont.)
(Sheet 13 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
2 (cont.)		I = 14: weapon reliability 15: number of weapons per site 16: reprogramming index (for missile squadron) 17: recovery mode index (1 for normal recovery probability, 0 for no recovery, -1 for low recovery probability) 18: penetration mode index (1 for normal use of corridors, 0 for corridors not used) 19: weapon type index used by the simulator 20: weapon function code (Hollerith)
3	TANK/LTANK(12)	Tanker data array, where for each tanker base: I = 1: tanker index number 2: tanker base latitude 3: tanker base longitude 4: refuel area assigned to tanker base 5: number of tankers per squadron or base 6: number of alert tankers per base 7: tanker speed in knots 8: alert delay 9: nonalert delay 10: total time on station 11: tanker type index 12: tanker range
	GRPX/IGRPX(6)	Weapon group data for output to LTGRP, where for each group element: I = 1: group number 2: base index number

Table 4. (cont.)
(Sheet 14 of 14)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
3 (cont.)		I = 3: base latitude 4: base longitude 5: base payload index 6: number of first vehicle/ number per base
12	TAR/ITAR(I,J) (I = 29, J = 250)	Array in which target data is stored during subroutine TGTSORT. For each target table entry J, I = 1 to I = 29 correspond to the same variables in common block /TD/
	CPLX/ICPLX(I,J) (I = 30, J = 250)	Array in which complex target data is stored during subroutine TGTSORT. For each table entry J, I = 1 to I = 29 correspond to the same variables in common block /TD/. ICPLX(30, J) is a copy of the complex number
	GRPCOMP/IGRPCOMP(I,J) (I = 5, J = 2500)	Array containing data for each weapon in a group. For each group element J, I = 1: base index 2: base latitude 3: base longitude 4: base payload index 5: number of first vehicle/ number per base

SUBROUTINE AROVRFLW

PURPOSE:	To print an error message in the case of an array overflow during PLANSET execution.
ENTRY POINTS:	AROVRFLW
FORMAL PARAMETERS:	None
COMMON BLOCKS:	DPOOL
SUBROUTINES CALLED:	ABORT
CALLED BY:	PLANSET

Method

The array ICHKFLG contains the name of the type of item which has exceeded the maximum number of that type allowed in the current version of QUICK. (See description of common block MAX.) The corresponding element in the ICHKNUM array contains the number of that item type encountered. AROVRFLW (see figure 3) examines each element of ICHKFLG, and prints the error message ARRAY OVERFLOW * * (ICHKNUM(I)) (ICHKFLG(I)) for each nonzero word it encounters. If any errors are found, the program then aborts.

The flowchart for subroutine AROVRFLW is shown in figure 3.

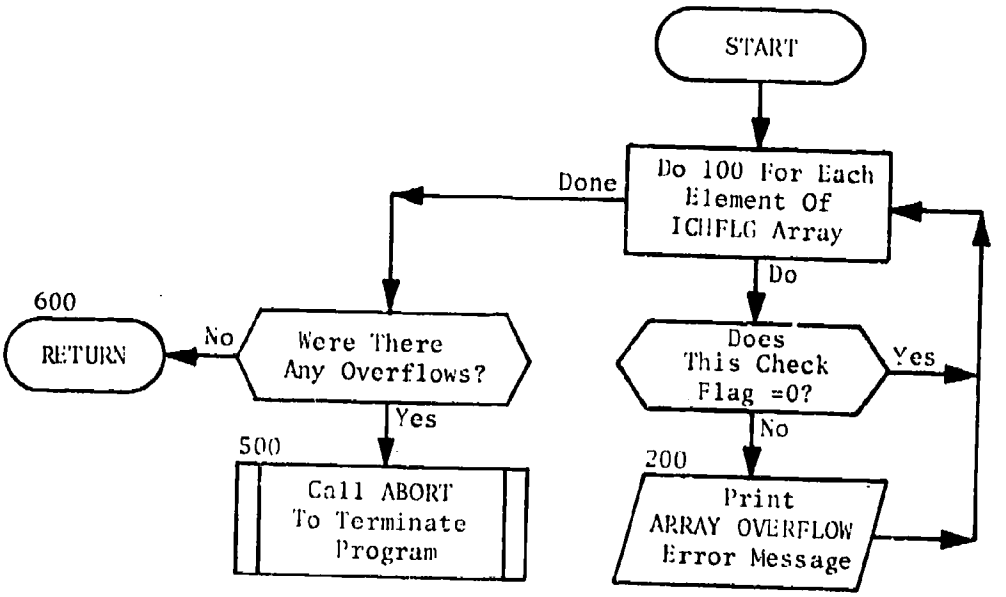


Fig. 3. Subroutine ARGVRFLW

SUBROUTINE CALCOMP

PURPOSE:	To calculate data which represent a complex target from data for the elements of the complex.
ENTRY POINTS:	CALCOMP
FORMAL PARAMETERS:	I - Equals Column of array ITAR K - Equals Column of array ICPLX containing first element of complex
COMMON BLOCKS:	DIRECTRY, PRIOR, TAU, 12
SUBROUTINES CALLED:	ORDER, REORDER
CALLED BY:	TGTSORT

Method

CALCOMP begins by retrieving the number of targets in the complex from array ITAR, and comparing the TASK and DESIG of each component to the input lists of priorities in order to choose the "lead" component for that target. It then rearranges the ICPLX array of components so that the desired component will appear first, and stores the relevant data from that component in the single target array ITAR. It also initializes counters and variables to be used in later calculations. This initialization includes replacing the value of the attributes TGTRAD, TARDEF, MISDEF, MINKILL, MAXKILL, and MAXCOST in array ITAR (see common block /12/ description) with a zero. In addition, the word COMPLEXD is stored in place of ICLASS in ITAR for air defense targets; for all other targets the word COMPLEX replaces ICLASS.

Data required for calculations (TGTRAD, FVALT1, FVALT2, VTO, NK, TAU1, TAU2, TAU3, TARDEF, and MISDEF) now are retrieved from array ICPLX for each target. A search is made for the maximum target radius, which is assigned as the radius of the complex. Similarly, the maximum value of TARDEF is found and assigned to represent the complex. The target value VTO, the number of terminal interceptors (MISDEF), and the weighted (by VTO) attributes MINKILL and MAXKILL are accumulated as each target is encountered. Also, for each target, the time components and the corresponding actual value lost at that time are placed sequentially in the arrays V and TAU.

After all targets in the complex have been processed as above, the total value and the total number of terminal interceptors are stored in the target array ITAR. The average values of MINKILL and MAXKILL also are calculated and stored in ITAR. Subroutines ORDER and REORDER then are called to arrange the elements of TAU in numerical order and to place the elements of V in the corresponding order. Those ordered arrays are used to approximate the time dependence values for the complex (T1, T2, T3) in the following manner.

First the array TAU is checked for equal time components. If any are found, the corresponding values are added together, and all equal components but the last are set to zero. When the entire array has been checked it is collapsed to eliminate any zero components. If the number of remaining entries does not exceed three, the time dependence of the value is approximated by these time components. Otherwise, an elimination procedure to reduce the number of entries to three is begun. To accomplish this, successive values (VTOs) are accumulated under V(1), and the average, weighted by VTO, of the corresponding time components is accumulated under TAU(1). As successive entries are combined with the first entry, they are replaced by zero. When the content of V(1) becomes greater than or equal to 35% of the total complex value (VTOT), TAU again is collapsed. If more than three entries still remain, the above accumulation process is restarted, beginning with the second entry in V. This may be repeated one more time if necessary, to accumulate the remaining data into V(3) and TAU(3). At this point, no more than three time components can remain.

Once the elimination process is complete, the fractional value is computed for the first two components from the sums now stored in V(1) and V(2). These fractions, together with the time components in TAU and the total number of components (KK), are stored in array ITAR.

Calculation now begins to determine the hardness components (H1, H2) and the corresponding fractional value (FVALH1) which represent the complex. VTO, FVALH1, the number of hardness (1 or 2), and the lethal radius corresponding to each hardness (from function VLRAD) are retrieved from array ICPLX for each target in the complex. The complement of FVALH1 is found to represent the second hardness component. If either fractional value is nonzero, it is multiplied by VTO to get the actual value at that hardness. The result is stored in array V, and the corresponding lethal radius is stored in array HC. After all targets have been considered, the lethal radii are separated into radii belonging to hard targets (radii less than 1.5 miles) and radii belonging to soft targets. The average lethal radius, weighted by the actual value at the corresponding hardness, is calculated for both hard and soft targets for those radii and the result (HHARD or HSOFT) is stored in array ITAR. Similarly, the actual value at each hardness (VHARD or VSOFT) is accumulated. If there are no hard targets (i.e., VHARD=0), FVALH1 in array

ITAR is set to 1; otherwise the fraction of actual value for hard targets (VHARD/VTOT) is assigned to FVALH1. The number of hardness components then is placed in ITAR, and control is returned to TGTSORT.

See figure 4 for the logic flow within CALCOMP.

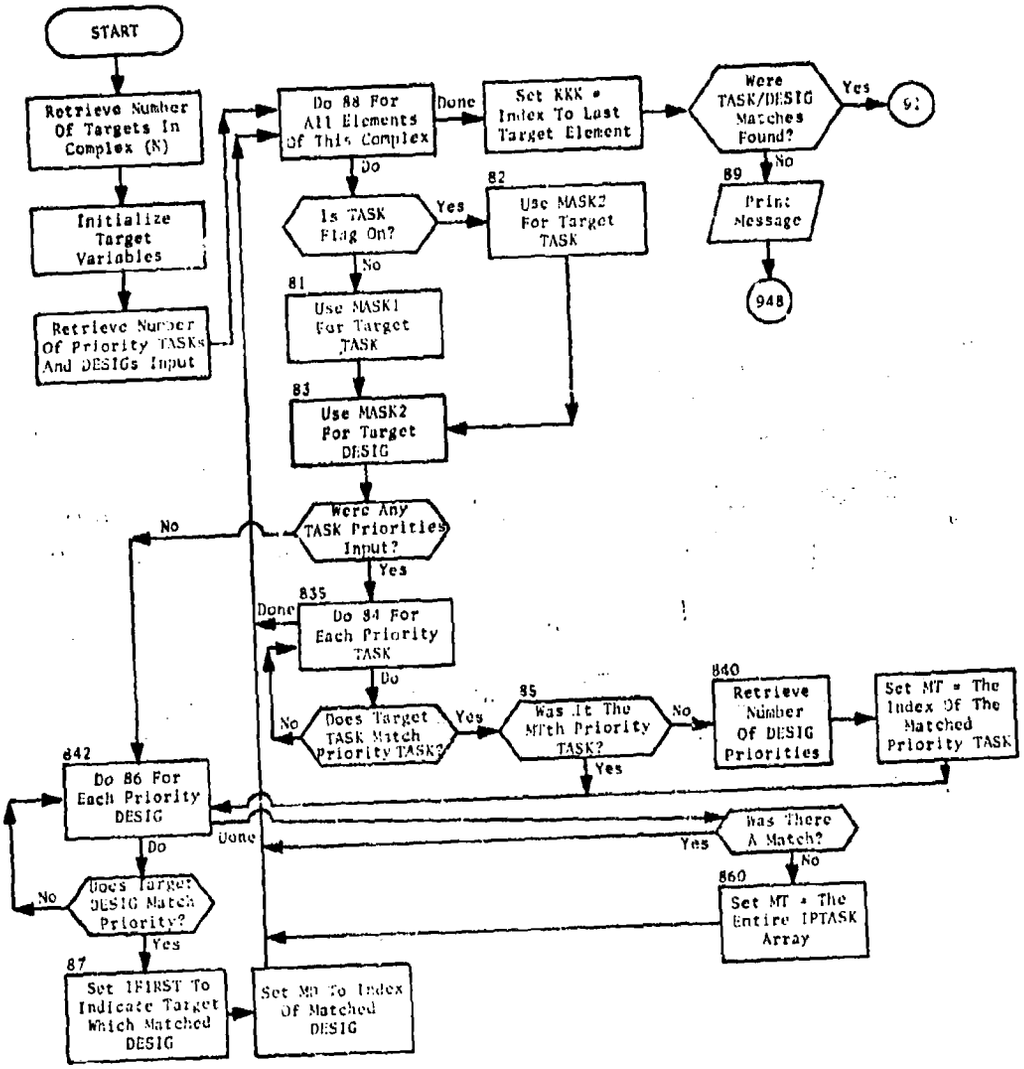


Fig. 4. Subroutine CALCOMP
(Sheet 1 of 6)

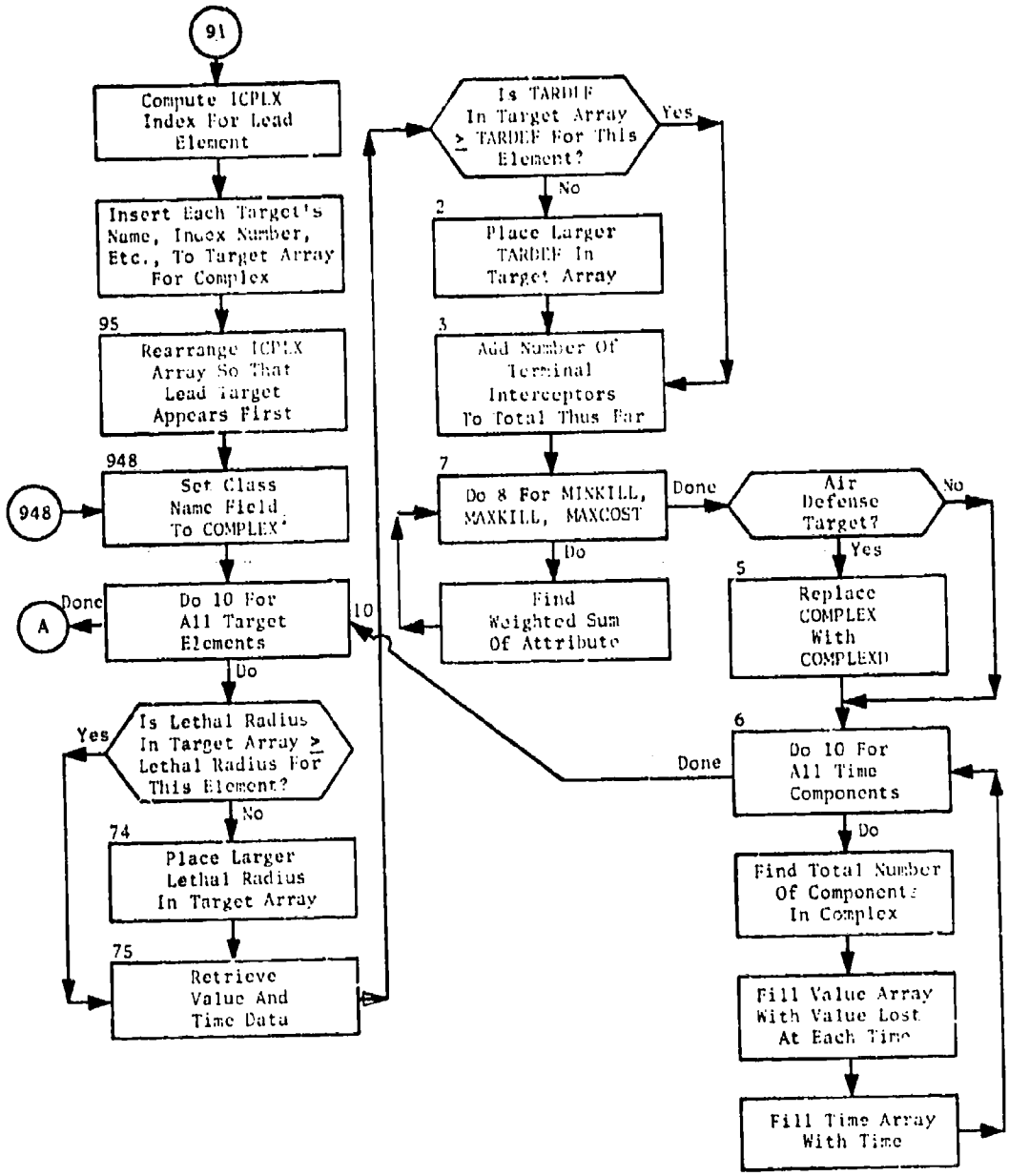


Fig. 4. (cont.)
(Sheet 2 of 6)

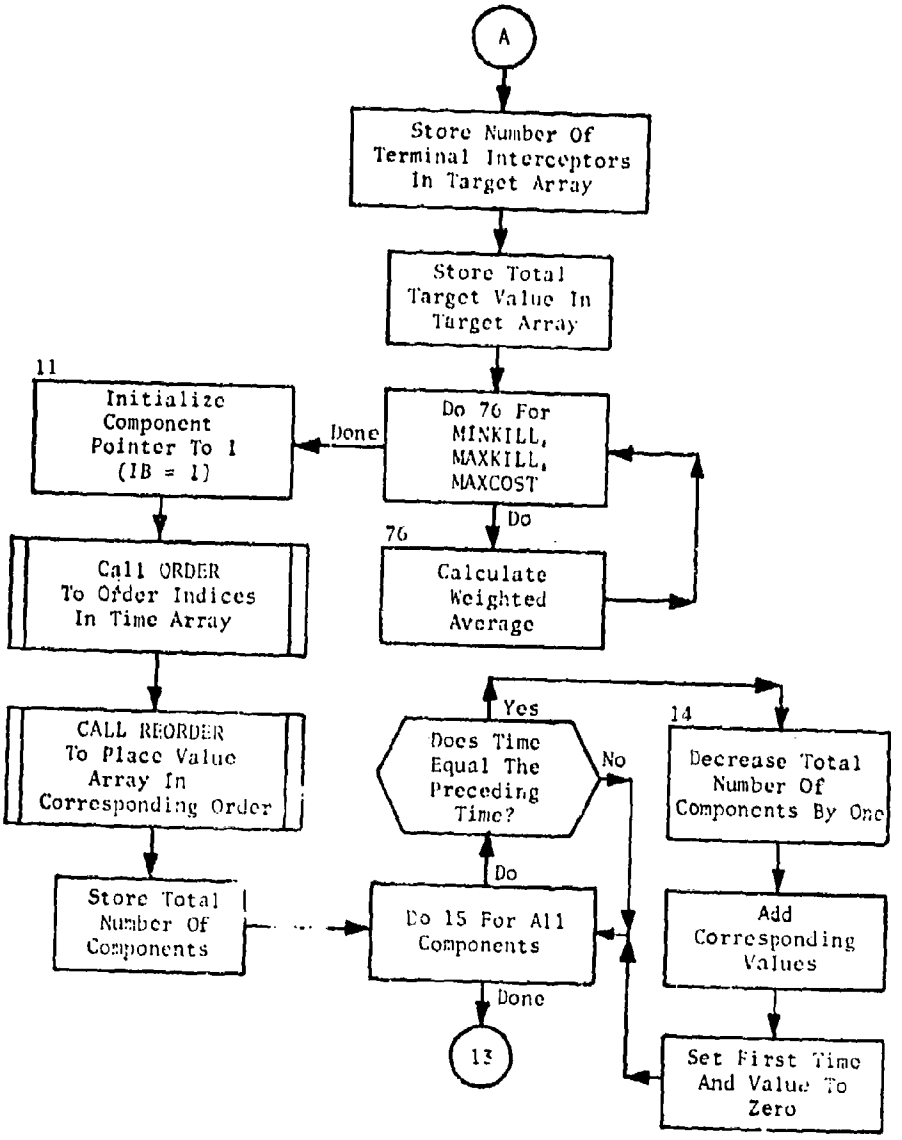


Fig. 4. (cont.)
(Sheet 3 of 6)

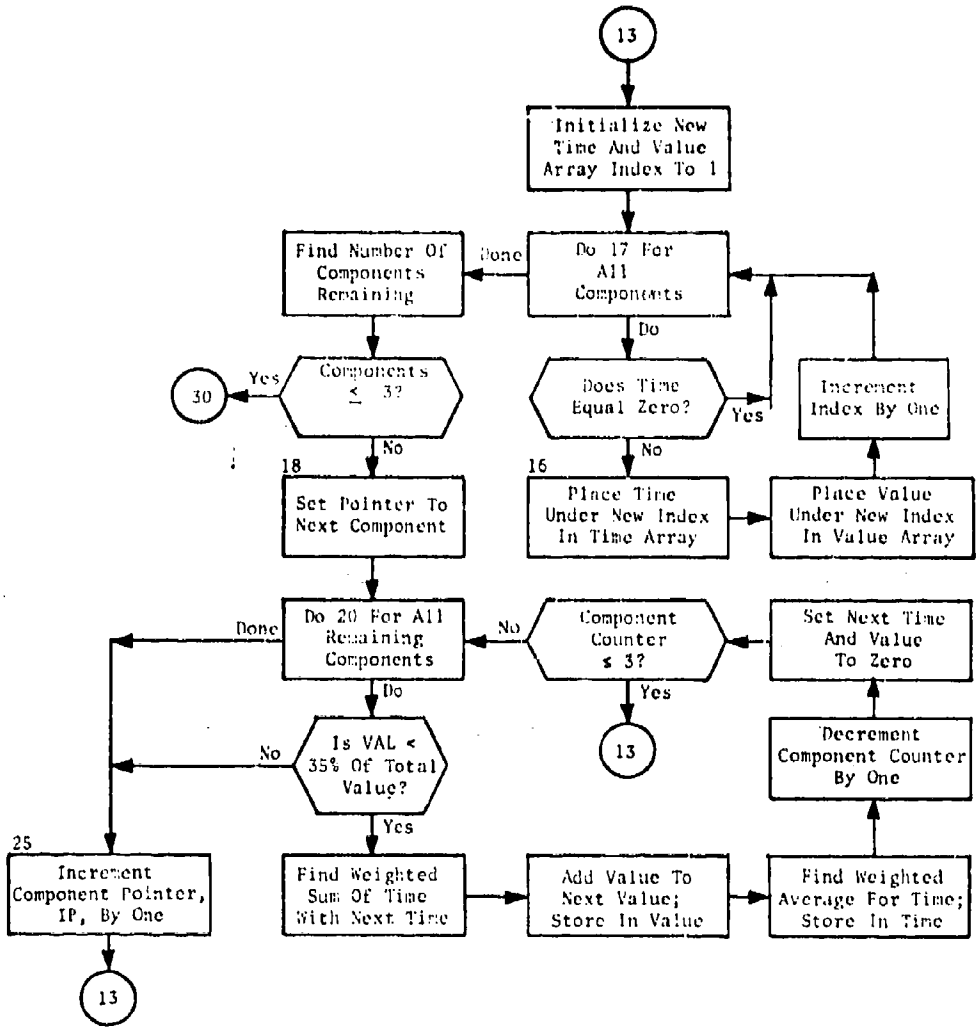


Fig. 4. (cont.)
(Sheet 4 of 6)

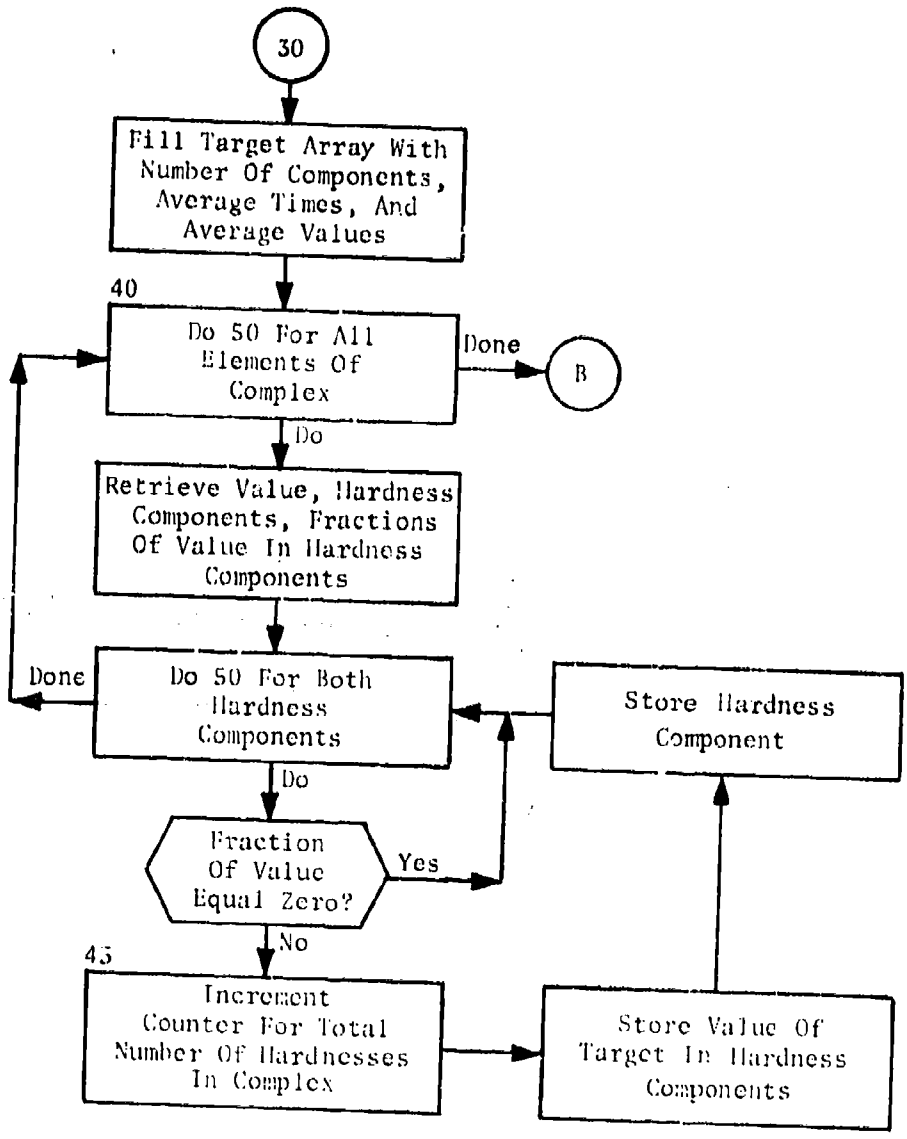


Fig. 4. (cont.)
(Sheet 5 of 6)

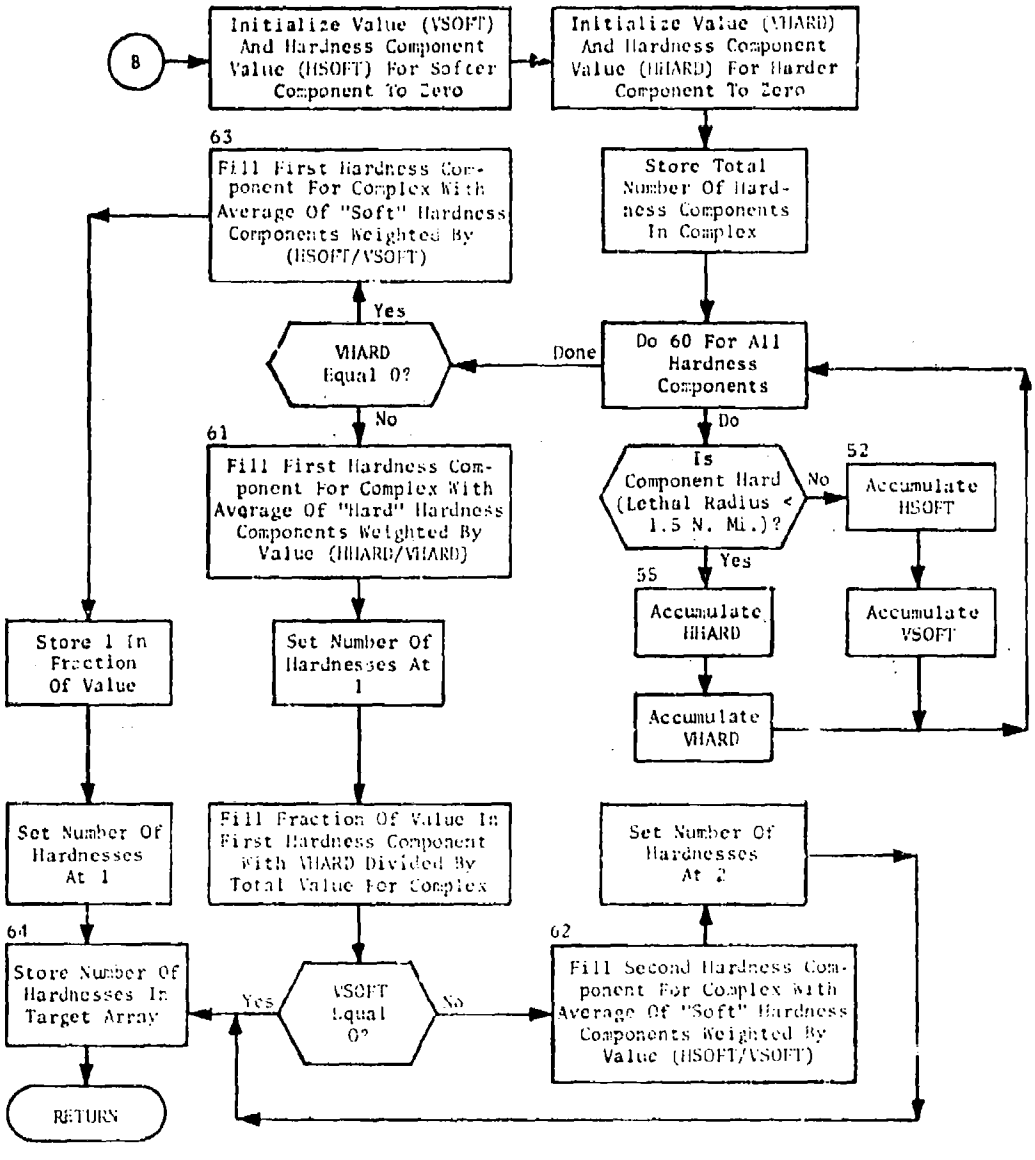


Fig. 4. (cont.)
(Sheet 6 of 6)

FUNCTION DBLCALC

PURPOSE:	To compute destruction before launch (DBL) probabilities for weapon groups with time-dependent DBL.
ENTRY POINTS:	DBLCALC
FORMAL PARAMETERS:	TTIME, IIDBL
COMMON BLOCKS:	DPOOL
SUBROUTINES CALLED:	None
CALLED BY:	PLANSET

Method

This function, shown in figure 5, does a simple table lookup with linear interpolation. TTIME is the time for which the probability is desired. IIDBL is the index of the table from which data is desired. The tables are contained in common block /NAVALX/ in the arrays TMASW and DBLASW.

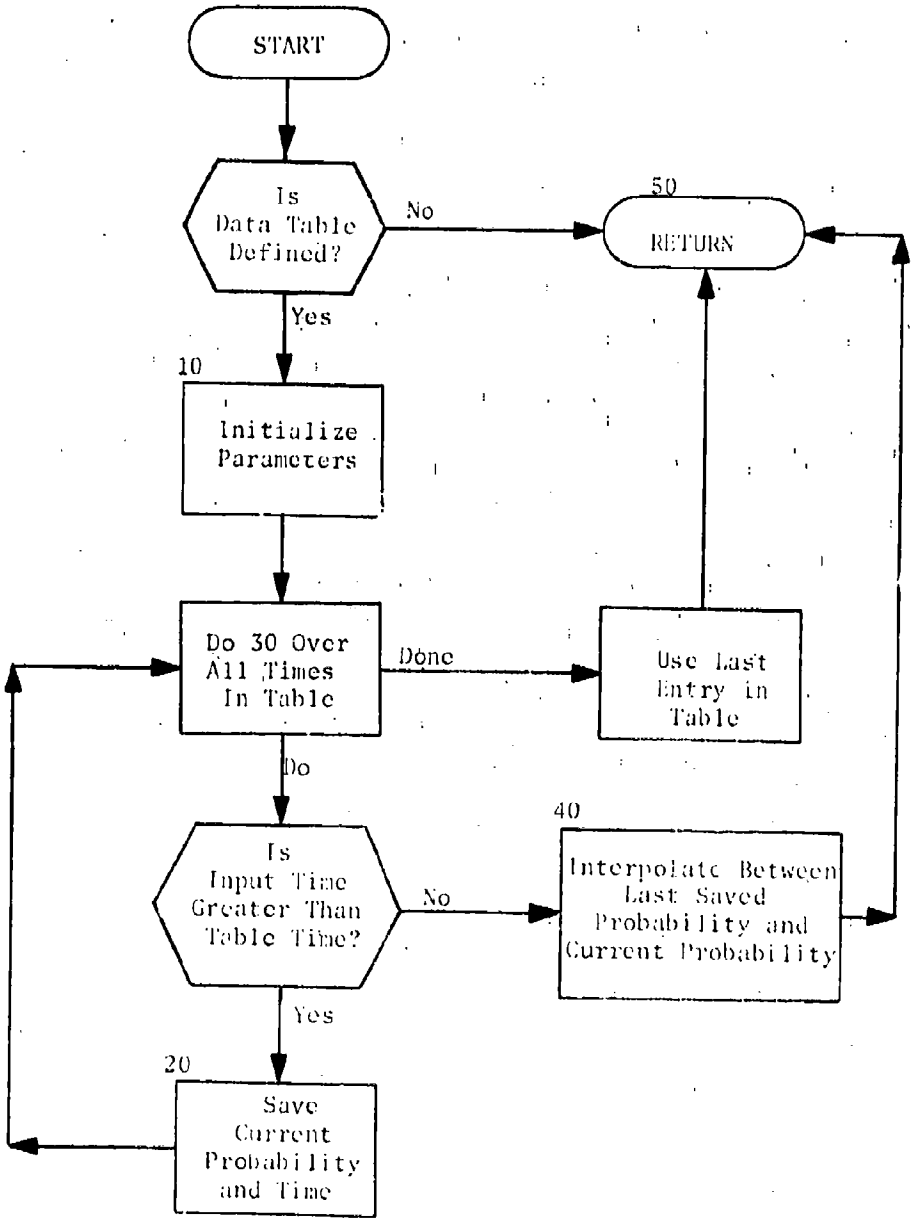


Fig. 5. Function DBLCALC

SUBROUTINE GRPSORT

PURPOSE:	To sort the weapon group, weapon base, and tanker data, and to write the data to WINFILE.
ENTRY POINTS:	GRPSORT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	DIRECTRY, DPOOL, GROUP, ITP, MYIDENT, PRCNTL, TAPES, TWORD, WT, 12
SUBROUTINES CALLED:	ABORT, RDARRAY, RDWORD, SETREAD, TERMTAPE, WRARRAY
CALLED BY:	SHUFFLE

Method

GRPSORT begins by calculating a breakpoint index (INDGRP) for each group formed in PLANSET, thereby reserving an area in array IGRPCOMP to accommodate the data for each base in the group. This group information, together with data for tanker bases, has previously been stored on the intermediate group file (LTGRP) by PLANSET. To distinguish between group and tanker data on LTGRP, the group number (between 1 and 200) was placed in the first word of each group data item, while a -1 or -4 was written as the first word for each tanker data item.

Tanker data which are preceded by a -1 is for tankers which are preassigned refuel areas in the data base, while data preceded by a -4 are for tankers which will be automatically allocated. The incoming information is separated by reading it into array JTANK for automatically allocated tankers (Code = -4), and into array ITANK otherwise. If the first word is positive, the incoming information is read into array IGRPCOMP.

For groups, the group number is also used to find the breakpoint index which corresponds to each group. Tanker data are stored in the order encountered; group data are stored in the next available location in the area designated by INDGRP and the group number.

After all items have been read and sorted, the group data (in array IGRP) and the group composition data (in array IGRPCOMP) are written onto the WINFILE for each weapon group. The group composition data then are printed if the user has requested it. Finally, the tanker information (in arrays ITANK and JTANK) is written onto WINFILE, and control is returned to SHUFFLE.

The flowchart for GRPSORT is shown in figure 6.

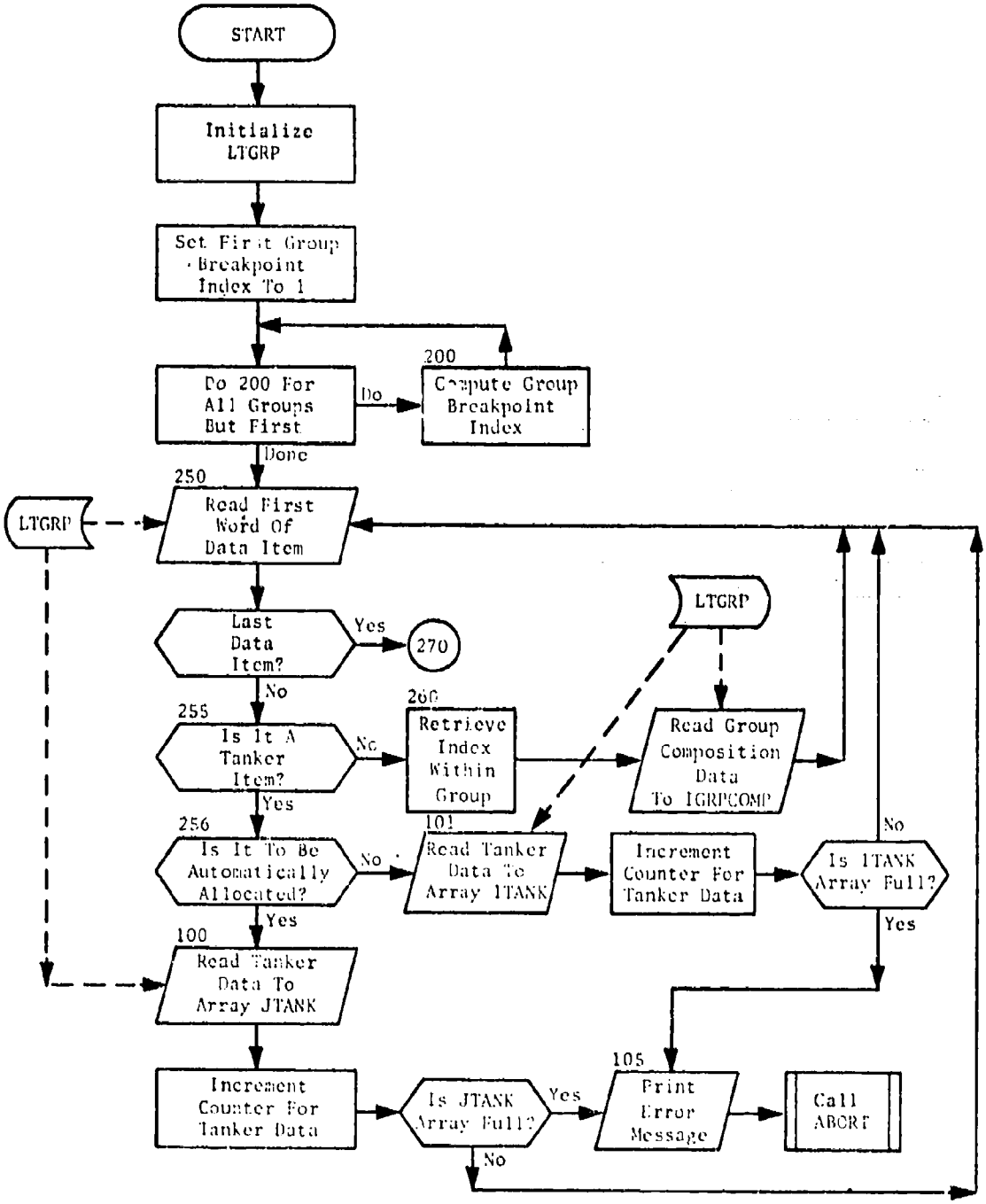


Fig. 6. Subroutine GRPSORT
(Page 1 of 2)

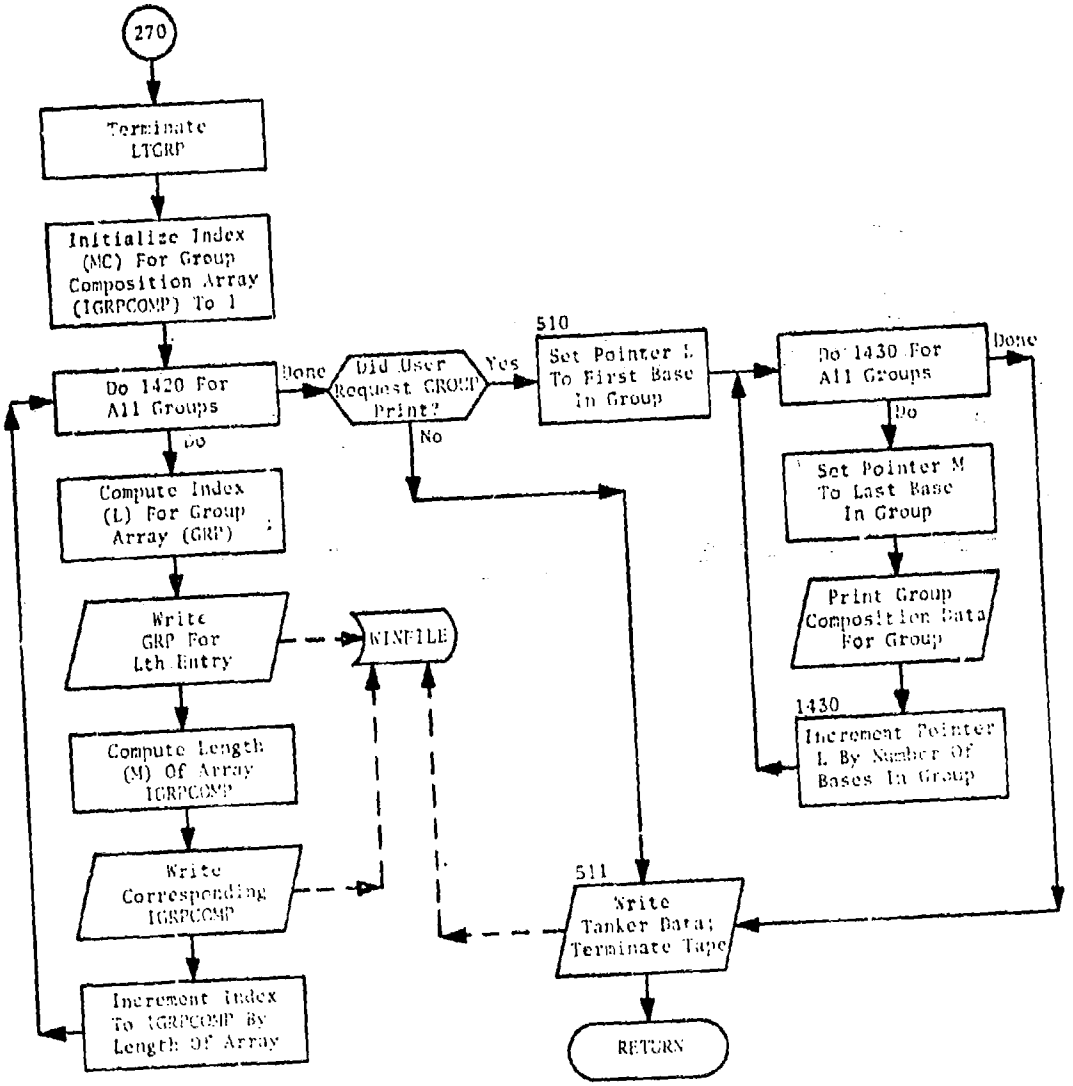


Fig. 6. (cont.)
(Sheet 2 of 3)

SUBROUTINE INITBLKS

PURPOSE:	Initialize all common blocks used by PLANSET.
ENTRY POINTS:	INITBLKS
FORMAL PARAMETERS:	None
COMMON BLOCKS:	DIRECTRY, DPOOL, CLASNAME, GROUP, MAX, MISC, MLTX, PRIOR, TAPES, TAU, TD, WT, 1, 2, 3, 12
SUBROUTINES CALLED:	None
CALLED BY:	PLANSET

Method

INITBLKS (see figure 7) is called at the beginning of PLANSET to initialize all the above named common blocks. All but /MAX/ and /TAPES/ are set to zero. /MAX/ is data set with the maximum number of each type of element allowed in QUICK (see description of common block MAX for values), and /TAPES/ is filled with the logical unit number for each file to be used during the program.

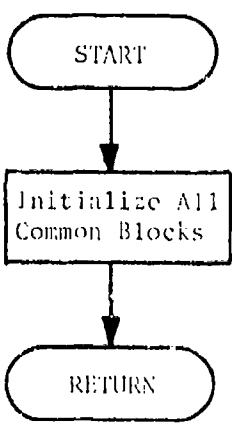


Fig. 7. Subroutine INITBLKS

SUBROUTINE SHUFFLE

PURPOSE:	To read the TASK and DESIG priority input cards and to control the writing of WINFILE and TINFILE.
ENTRY POINTS:	SHUFFLE, SHUFFL1
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CLASNAME, DPOOL, DIRECTRY, GROUP, ITP, MAX, MISC, MYIDENT, NOPRINT, PRIOR, TAPES, TODAY, TWORD, WT, 1, 2, 12
SUBROUTINES CALLED:	GRPSORT, SETWRITE, TGTSORT, WRARRAY, WRITER
CALLED BY:	PLANSET

Method

SHUFFLE is called by PLANSET to read and store the TASK and DESIG priority input cards to be used for selecting the lead element of each complex target. SHUFFL1 is called just before PLANSET terminates. It initializes TINFILE and WINFILE, writes the point data (Breakpoint tables through Boundary Points; see Table 3, WINFILE format) to WINFILE, then calls TGTSORT to add target records to both output files. It is in subroutine TGTSORT that the actual "shuffling" of targets is done, using the algorithm described in the Analytical Manual, Volume II, Chapter 3, "Target Shuffling." After copying the warhead, ASM, payload, etc., tables to WINFILE, it calls GRPSORT to complete the WINFILE. When SHUFFL1 returns, PLANSET terminates.

The flowchart for SHUFFLE is shown in figure 8.

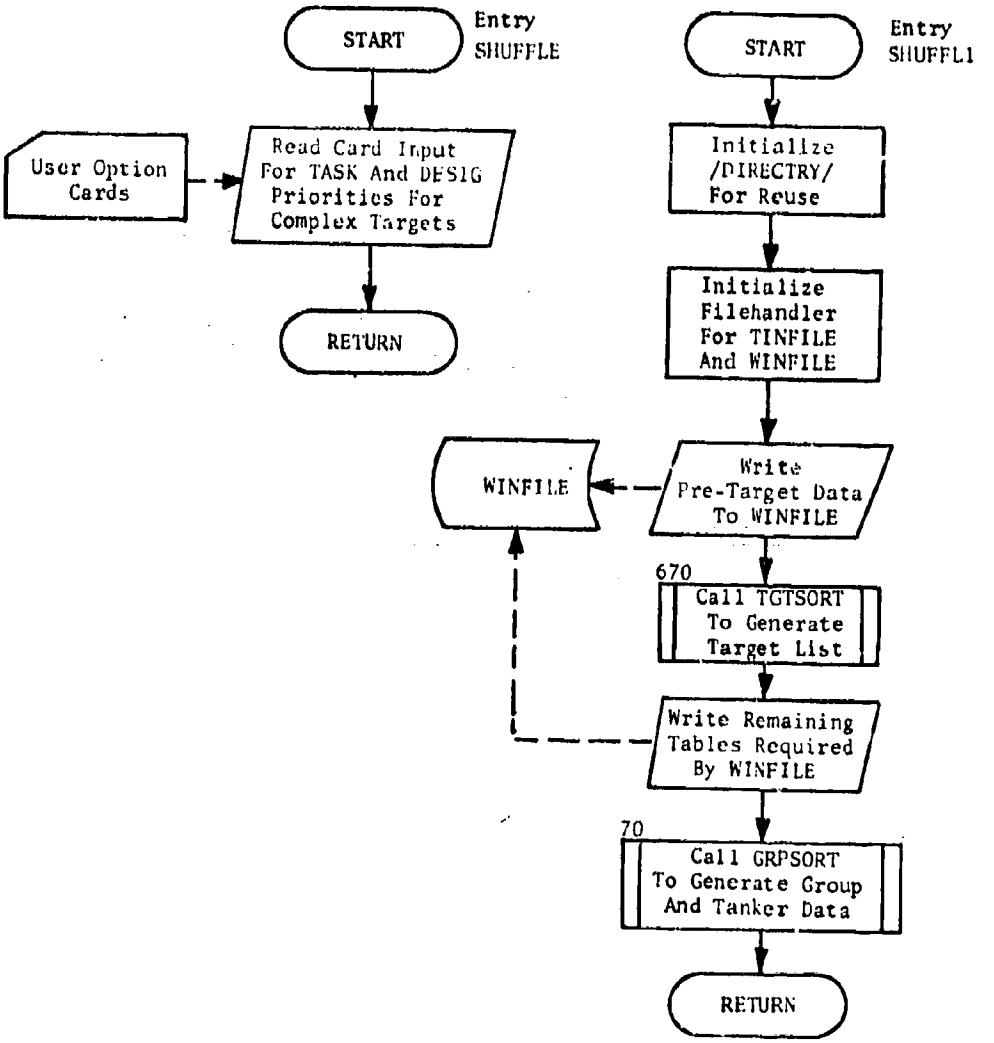


Fig. 8. Subroutine SHUFFLE

SUBROUTINE TGTSORT

PURPOSE:	To rearrange the list of targets so that classes are more evenly distributed throughout.
ENTRY POINTS:	TGTSORT
FORMAL PARAMETERS:	NTAR (number of targets)
COMMON BLOCKS:	CLASNAME, 1, DPOOL, DIRECTRY, GROUP, 12, IFTPRNT, ITP, LCPX, MAX, MLTX, MYIDENT, PRCNTL, TAPES, TWORD, TD
SUBROUTINES CALLED:	ABORT, CALCOMP, ORDER, RDARRAY, SETREAD, SETWRITE, TERMTAPE, WRARRAY, WRWORD
CALLED BY:	SHUFFLE

Method

TGTSORT begins by assigning an index IND to each target as it is read from the target tape. This index is used to distribute the targets uniformly throughout the target list by cycling through them as follows: a sorting index (LEAD), which is a function of the total number of targets NTAR), is calculated by the formula:

$$LEAD = \frac{NTAR\ (3 - \sqrt{5})}{2}$$

To start a cycle, a beginning index (IBEG) is designated and assigned to the target being read. Initially IBEG = LEAD. The index for the next target (IND) then is found by incrementing the previous index (LAST) by LEAD. If the result exceeds NTAR, the cycle is reset by subtracting NTAR from IND. When a cycle is completed (i.e., when IND = IBEG), the next cycle is begun by incrementing IBEG by one and proceeding as above. Thus, a unique nonsequential index is assigned to each target as it is read.

Because of storage limitations on the arrays JTAR, ICPLX, and MLTCOMP several passes through the target list may be necessary before all data can be processed. In order to determine which targets are to be considered during a given pass, the index limits MIN (lowest target number in this pass minus one) and MAX (maximum number of targets which can be processed per pass) are established. These limits, together with the index IND, are

used to find an index (IT) to the column in ITAR in which data corresponding to IND are to be stored. The procedure is as follows.

As each target is encountered, the value of MIN for that pass is subtracted from the index IND corresponding to the target. If the result (IT) lies between one and MAX ($\leq$ 250), the target data is stored in column IT of array ITAR, as required. Otherwise, the target is ignored for the remainder of the pass.

To indicate that a target was not processed during a pass, a zero is placed in the second word of each column of ITAR when the pass begins. If data for a target are stored, the zero is replaced by the attribute INDEXNO. Then, after the pass is completed, a check is made on the content of ITAR(2,I), $1 \leq I \leq 250$. If a positive value is found, target data for the corresponding target are processed as necessary and written onto TINFILE. The first time a zero is encountered, the index MIN is incremented by the number of targets successfully processed (I - 1), and another pass is begun.

When the index check indicates that data for a complex or multiple target are to be saved during a pass, data for all other elements of the target must also be included in the same pass. Therefore, the number of elements is tested against available storage in the complex ICPLX or multiple MLTCOMP target array. If the required data will not fit into the array, the value of MAX is decreased by 10, the variable MAXCHNG is set to two, and the entire pass is restarted. The process continues until the number of targets to be considered during the pass is sufficiently decreased to permit the storage of all necessary data in ICPLX or MLTCOMP. When this occurs, the variable MAXCHNG will still equal one after the last target has been read from the target tape, thereby indicating that a new pass is to begin. If data for the first target in a complex are not stored during a pass, data for additional complex members also are ignored.

In order to avoid repeating index assignments, the value of IND calculated during the first pass is stored in the array ITEMP for all targets. Thus, as each target is encountered during subsequent passes, the appropriate IND is retrieved from ITEMP for use in computing the new value of IT. In addition to calculating IND on the first pass, certain target data (including NAME, INDEXNO, multiplicity, TYPE, complex indicator, VALUE, and NTINT) are printed for each target if the target prints were requested by the user.

When the above testing finds a target to be processed, TGTSORT proceeds in the following manner. The class name and value are stored in the appropriate locations of the target array ITD. For multiple and individual targets, the first 29 items of data then are read into column IT of array ITAR, and the value (VAL) is set at a minimum of 1×10^{-12}.

If the target is a multiple target, additional data (array MLTX in PLANSET) will follow on the target file (LTTGT). These data are read into array MLTCOMP, provided that the check described above has indicated that the array will accept all data. An index (K) to the first location in MLTCOMP in which the data have been stored replaces INDEXNO in array ITAR. The next item then is read from the target file (LTTGT).

Complex targets, if the array ICPLX will accommodate the required data, are processed as follows. For the first member of the complex, NAME, INDEXNO, LAT and LONG, etc., are read into the appropriate locations of the ITAR array. Then, for all members of the complex, target data are read into array ICPLX, the class value is set at a minimum of 1×10^{-12}, and the multiplicity is set at one. An index to the next empty column of ICPLX is stored by ICOMPLEX in array LCPX. If the index IND is being assigned during this pass (i.e., if IPASS = 1), and if the target just processed was not the first member of the complex, the current value of IND is assigned to the next target on the target file. Otherwise, the reading of the target file continues as with individual and multiple targets.

After the pass is completed, targets for which data have been saved are further processed. For complex targets, subroutine CALCOMP is called to calculate certain data to represent the entire complex. These data are placed in ITAR by the subroutine. Data for each member of the complex (array ICPLX) then are written on WINFILE. For multiple targets, the index to MLTCOMP is retrieved from ITAR and the vacated location is filled by INDEXNO. Array MLTCOMP then is written onto WINFILE. Multiplicity is changed to floating point for all targets, and array ITAR is written onto TINFILE. When every target has been considered (i.e., when MIN $\geq$ NTAR) the target lists on TINFILE and WINFILE are complete.

Before TGTSORT returns, the complex target list is printed, if such has been requested by the user print control card. At the end of each pass, the complex target elements in the ICPLX array will have been sequenced by complex number and written to LSRTA (for odd-numbered passes) or LSRTB (for even-numbered passes), the sort workfiles. Except for pass one, they are merged at the same time with the ICPLX string which was saved during the preceding pass. The final string is then printed, and TGTSORT returns control to SHUFFLE.

See figure 9 for TGTSORT logic flow.

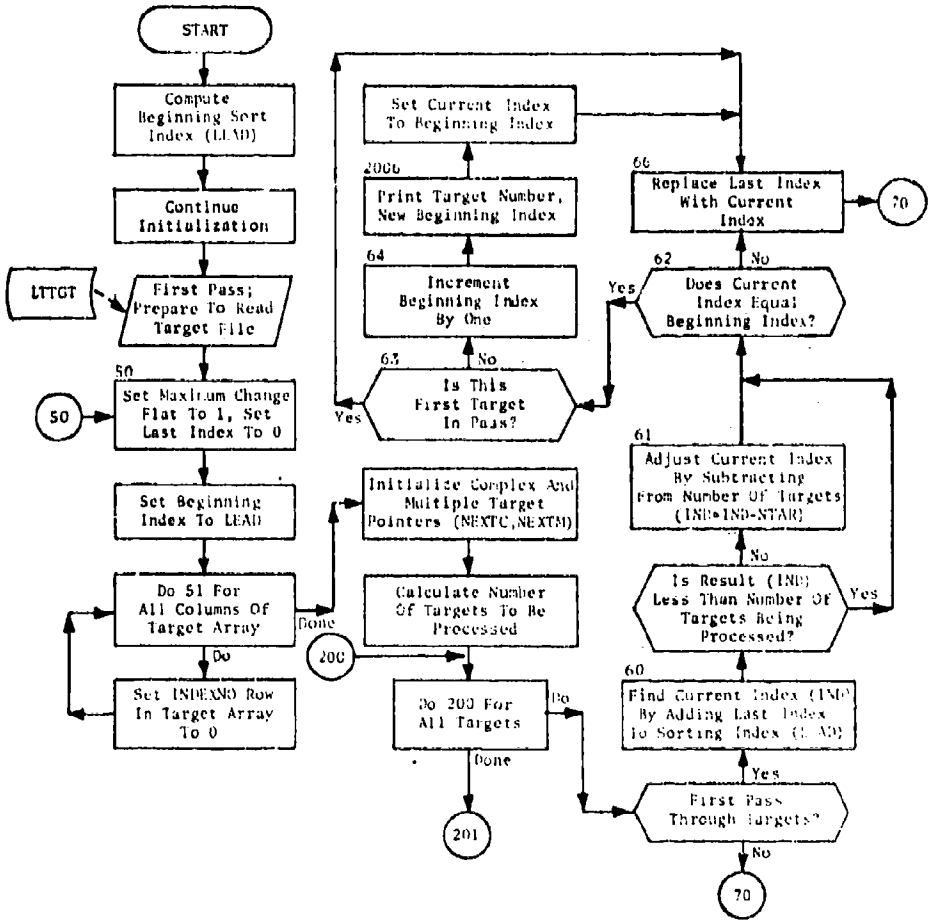


Fig. 9. Subroutine TGTSORT
(Sheet 1 of 6)

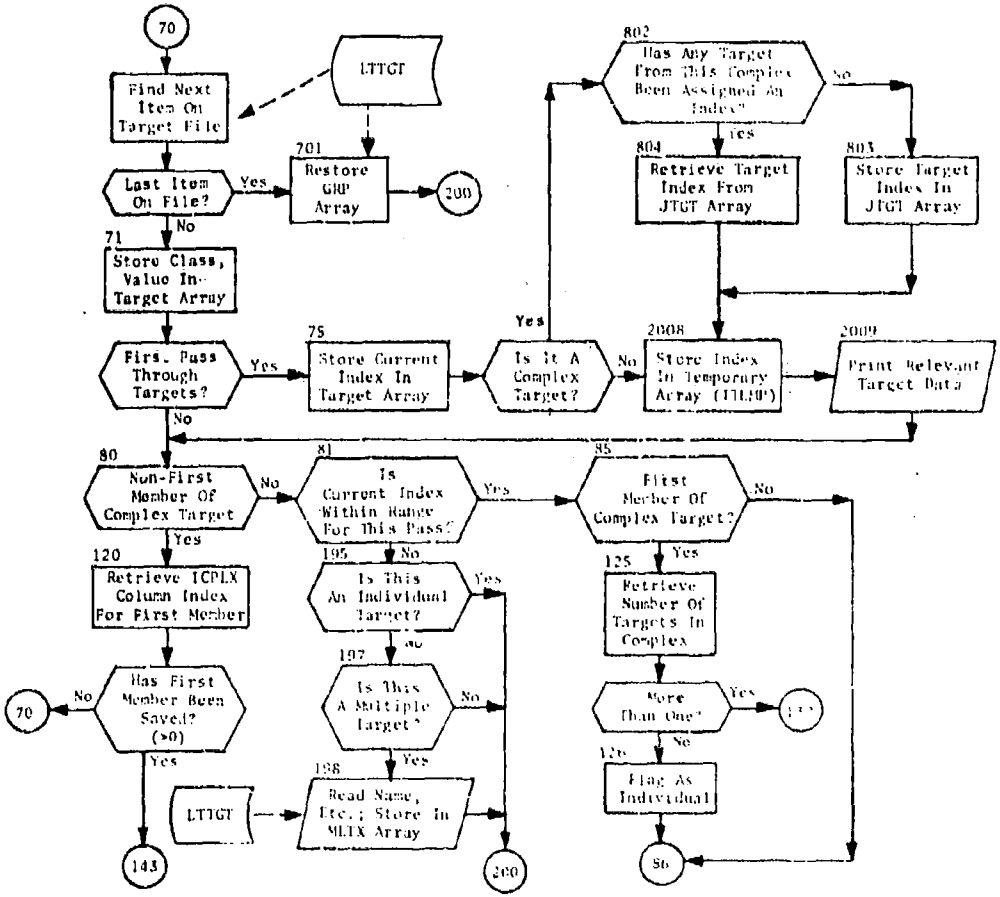


Fig. 9. (cont.)
(Sheet 2 of 6)

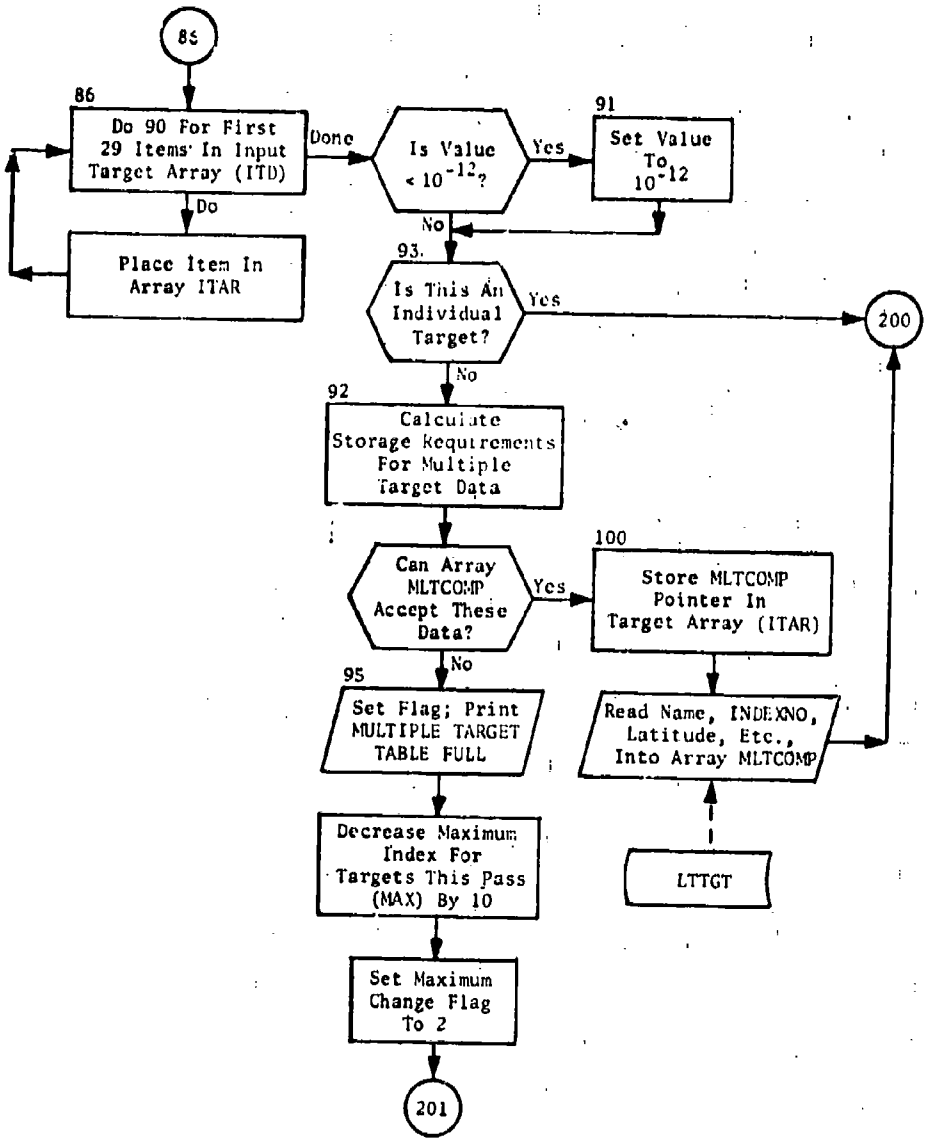


Fig. 9. (cont.)
(Sheet 3 of 6)

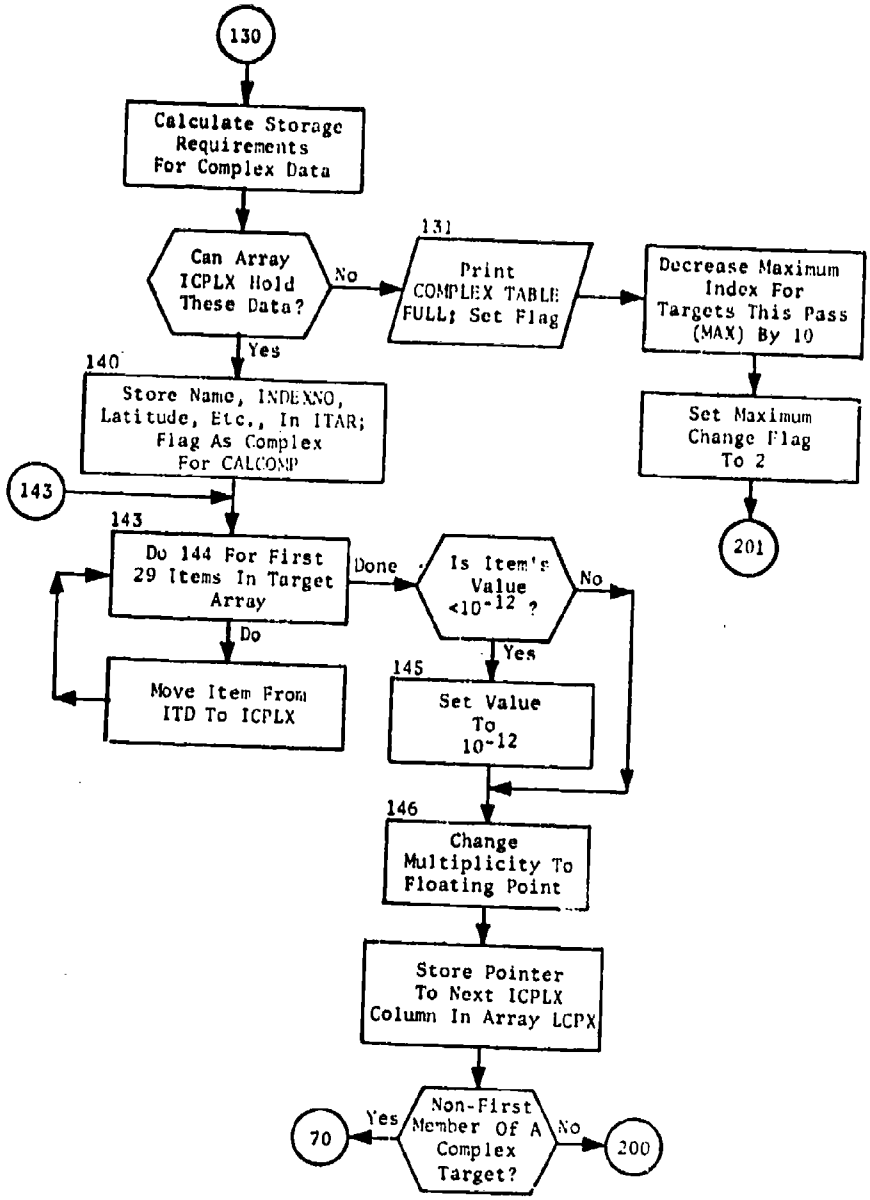


Fig. 9. (cont.)
(Sheet 4 of 6)

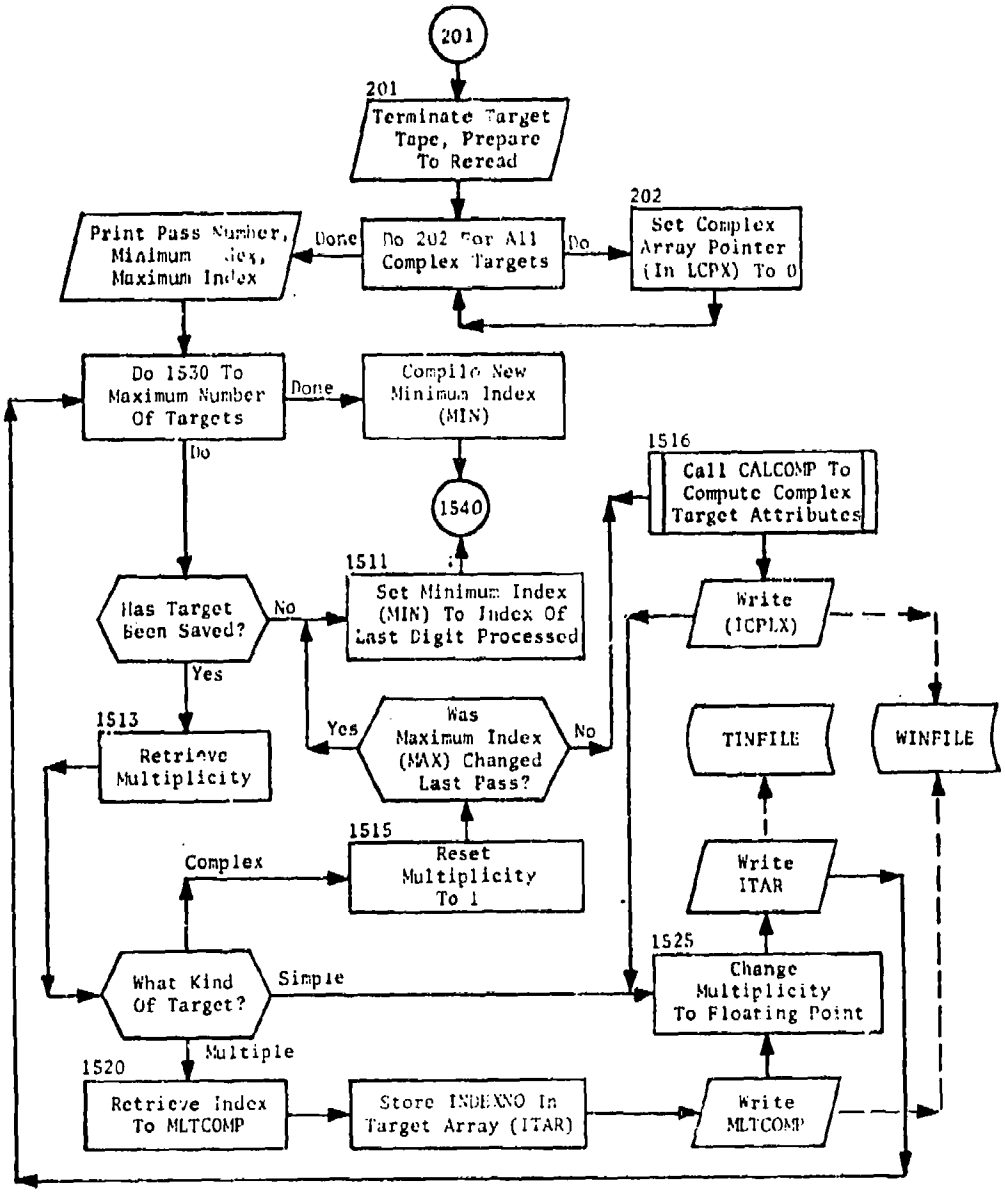


Fig. 9. (cont.)
(Sheet 5 of 6)

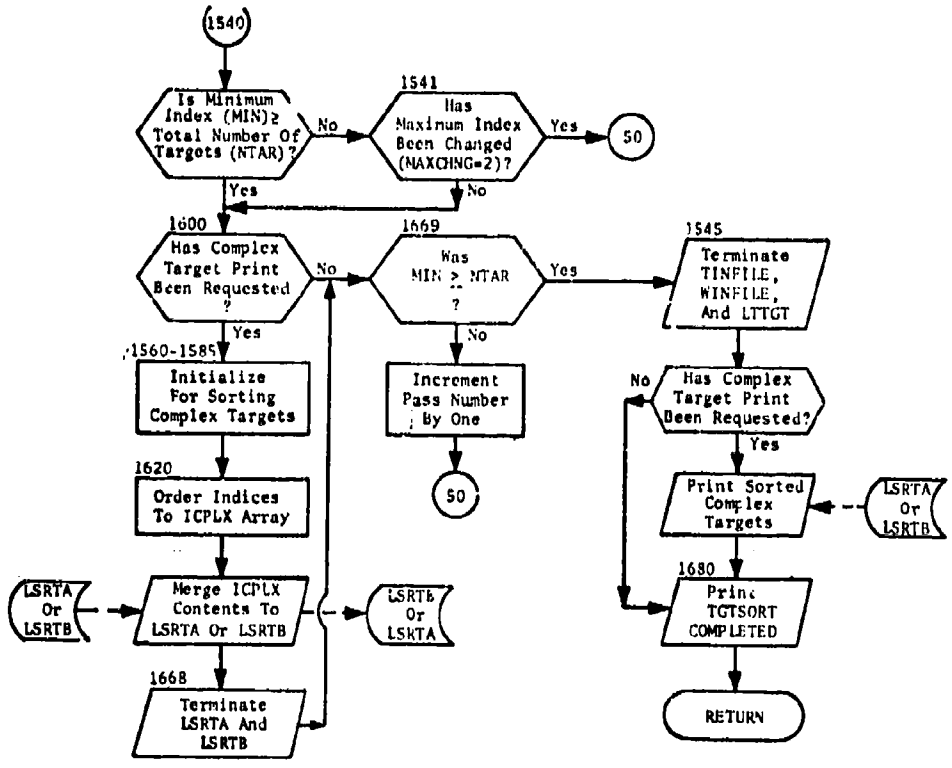


Fig. 9. (cont.)
(Sheet 6 of 6)

SUBROUTINE VLRADP

PURPOSE:

1. Find lethal radius of weapon.
2. Set FN for use by calling subroutine.

ENTRY POINTS: VLRADP

FORMAL PARAMETERS:

YIELD	-	Yield of weapon in megatons
NVN	-	Vulnerability parameter of target
HOB	-	Weapon height of burst
FN	-	Parameter specifying shape of damage function

COMMON BLOCKS: DPOOL

SUBROUTINES CALLED: EXPF

CALLED BY: PLANSET

Method

NVN is decoded into the appropriate vulnerability number VN, the letter (P or Q), and the K-factor XK. The cube root of the yield is extracted. Then the adjusted vulnerability number AVN is determined by methods described in "Computer Computation of Weapon Radius," B-139-61, Air Force Intelligence Center. FN is set to six or three for P and Q type targets, respectively.

Common block /DPOOL/ contains four arrays (for the four combinations of P or Q vulnerability and air-or-surface burst) each of which contains the natural logarithm of the lethal radius (in nautical miles) of a one-megaton burst. The data are at intervals of five vulnerability numbers. Subroutine VLRAD interpolates in the appropriate array to find the logarithm of the one-megaton lethal radius for AVN. The lethal radius of the weapon is then determined by exponentiating and multiplying by the cube root of the yield.

A flowchart for VLRADP is shown in figure 10.

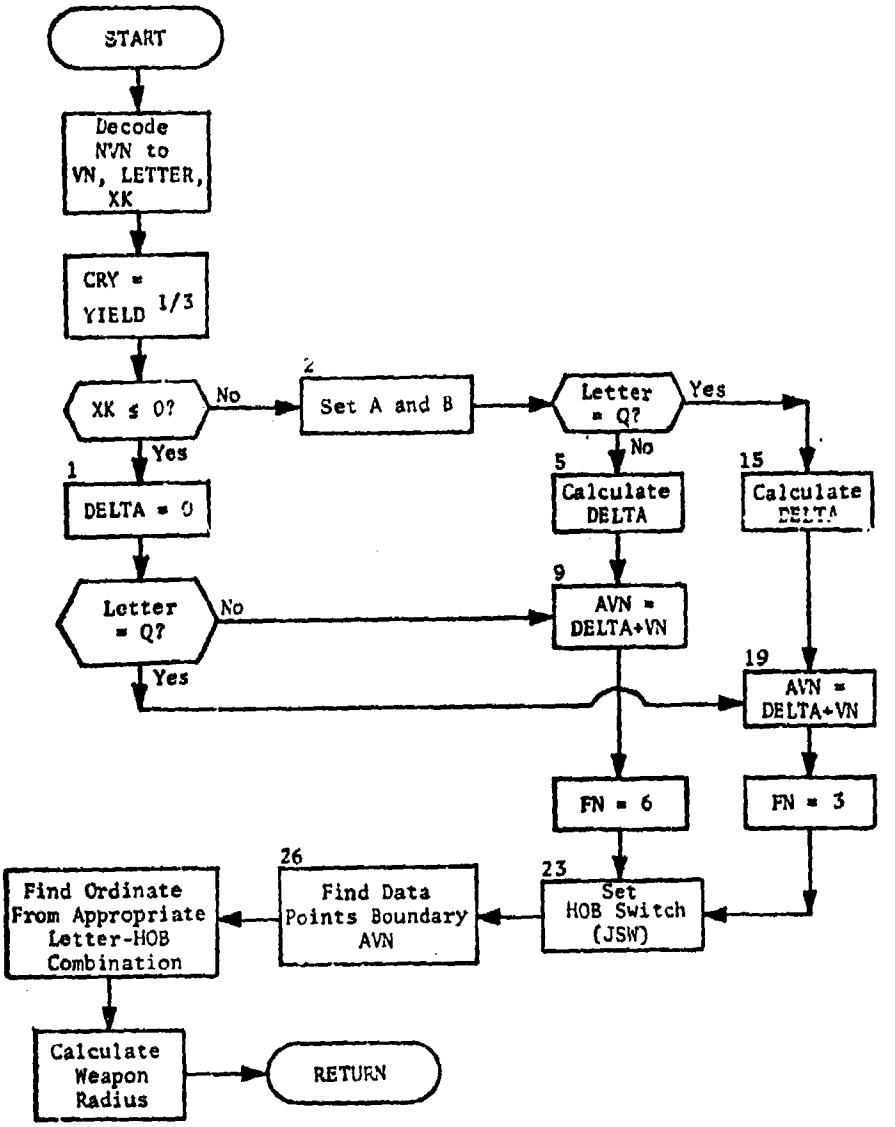


Fig. 10. Subroutine VLRADP

SUBROUTINE WRITER

PURPOSE:	To place up to five items in an array and to write the array on an output file.
ENTRY POINTS:	WRITER
FORMAL PARAMETERS:	N1, N2, N3, N4, N5, NUMB
COMMON BLOCKS:	None
SUBROUTINES CALLED:	WRARRAY
CALLED BY:	SHUFFLE

Method

The items specified by the calling sequence (N1 - N5) are placed in array NAR and immediately written onto the file designated by the calling routine. If less than five items are to be considered, a dummy variable must be provided in the calling sequence. The parameter NUMB contains the number of actual items to be written.

The flowchart for subroutine WRITER is shown in figure 11.

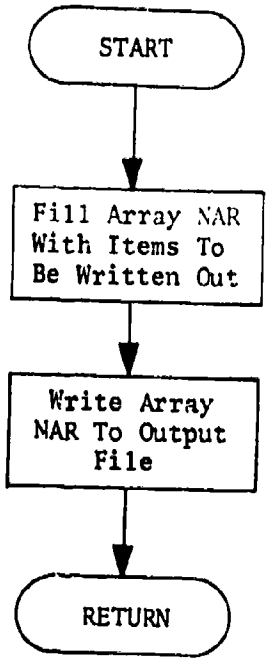


Fig. 11. Subroutine WRITER

SUBROUTINE WRMULT

PURPOSE:	To terminate a multiple target.
ENTRY POINTS:	WRMULT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MLTX
SUBROUTINES CALLED:	WRARRAY
CALLED BY:	PLANSET

Method

WRMULT (see figure 12) begins by placing the number of missile sites belonging to the multiple target (NMULT) in the target data array MULT and writing the array on the logical file unit specified by PLANSET. If two or more sites have been assigned to the multiple target, array MLTX also is written onto the output file. The counter NMULT is reset to zero, and control is returned to PLANSET.

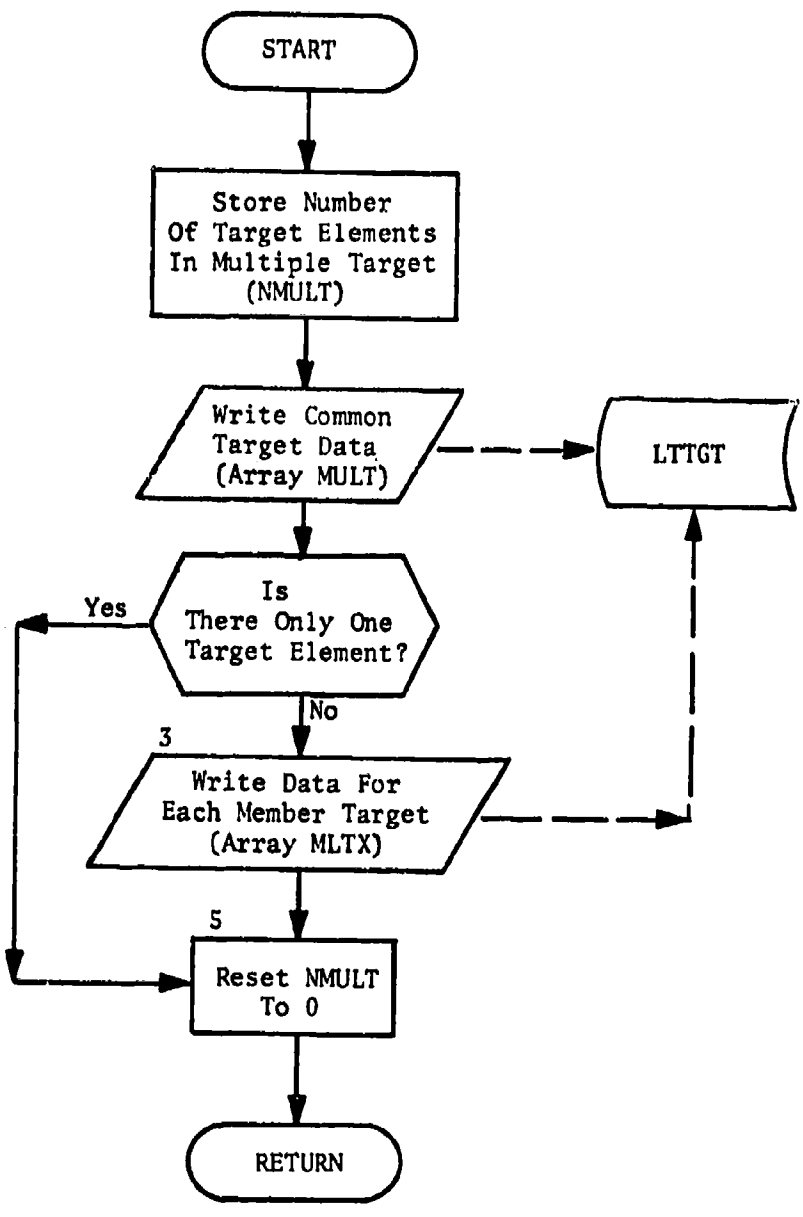


Fig. 12. Subroutine WRMULT

CHAPTER 3
PROGRAM PREPALOC

PURPOSE

The purpose of this program is to perform preliminary calculations on the weapon and target data supplied by program PLANSET. The output data from PREPALOC will be in a form convenient for use by the remaining processors of the Plan Generation subsystem. In addition, the user may select options to modify some of the data at this stage of processing.

Program PREPALOC has three major capabilities: generation of weapon and target data, modification of target values and damage constraints, and preparation of data for the fixed weapon assignment capability of program ALOC.

The data produced by the first capability can be divided into three categories: geographic, weapon, and target. The geographic data are produced from the data on boundaries and routes passed by PLANSET on the WINFILE. Program PREPALOC uses these data to define the legs for each penetration corridor, the location of each refuel area (except for those determined automatically by program PLNTPLAN), the bases available for recovery from each penetration corridor, and the air defense zones to be used in calculating bomber attrition. The weapon data are aggregates of the weapon data input on WINFILE. Such characteristics as speed, range, yield, CEP, and function are aggregated by weapon type. Data on payloads, warheads, and air-to-surface missiles (ASMs) are also calculated for use by later processors. The target data are the characteristics which define the target as a candidate for weapon allocation. The geographic location, vulnerability, value, damage constraints, value sensitivity to time, and other target characteristics are computed for each simple, complex, or multiple target. These data will determine the worth of weapon allocations in program ALOC. The only user-input data required for this capability identify the timing of the strike (e.g., initiative, retaliatory).

The second major capability of this program is the modification of the target characteristics, VTO, MINKILL, and MAXKILL. VTO is the value of the target relative to all the others. MINKILL is the minimum fraction of target value that must be destroyed, and MAXKILL is the maximum desired fraction of target value destroyed. Any of these parameters may be changed for any target. The change requests can change these parameters for a single target or for a set of targets. The set of targets for which a change is requested is identified by target class, type, an individual identifier (either target designator code (DESIG) or index

number (INDEXNO)), or any combination of these. For complex targets, the class, type, designator code, and index number of each component will be checked to determine if a target parameter for the complex is to be changed.

The third major capability is the request for allocation of specific weapons to specific targets. This fixing of weapons to targets enables the user to determine part of the weapon allocation while leaving the allocation program free to determine the remaining allocation. In addition, the time of arrival at target can be fixed for missile weapons. This information will be passed to program PLNTPLAN which will adjust the launch time accordingly. The fixing of weapons remains in effect for the remainder of the plan generation process. Later programs will retain the assignments as best possible. (For example, it is possible to fix a set of weapons from a weapon group with multiple independently targetable reentry vehicles (MIRV) in such a manner that there are no feasible footprints that cover that target set adequately. In that case, some of the fixed assignment requests must be ignored.)

INPUT FILES

The input to program PREPALOC comes from three sources, the WINFILE and TINFILE produced by program PLANSET, and the user-input parameters. The WINFILE contains information on the geographic data base data, auxiliary data class tables, weapon group information and target information for components of complex and multiple targets. The TINFILE identifies the characteristics of each target. The user-input parameters define the changes to be made to the target and weapon group characteristics.

OUTPUT FILES

There are two files output by program PREPALOC, the BASFILE and the TGTFILE.* Since the basic purpose of program PREPALOC is a restructuring of the input data, these files are similar to the input files. The TGTFILE contains the target characteristics input on the TINFILE as modified by the user-input requests. Appended to these characteristics

---

*Both files are output by the QUICK System filehandler on the CDC 814 disk unit.

are geographic data relating each target to each penetration corridor and a preferred depenetration corridor. Table 5 displays the format of the TGTFILE. There is one data block as described in table 5 for each target. The BASFILE is essentially a restructured WINFILE. It contains the basic source of weapon and target component (for complex or multiple targets) information for the remaining processors of the Plan Generation subsystem. Table 6 displays the format of the BASFILE. This file is divided into four major sections: basic data base information, target component information, detailed weapon group information, and a section containing geographic data, recovery base data, and tanker base data.

The first section on the BASFILE contains basic data base information. The first block of data in this section is common /MASTER/ which specifies the number of various entities (targets, groups, etc.) in the data base for the side whose plan is being generated. Next comes common /BRKPNT/, which defines the index number breakpoint tables as computed by program INDEXER of the Data Input subsystem. The third block, common /FILES/, specifies the logical unit number (for use by the filehandler) and the maximum file length for each nonscratch file used by the Plan Generation subsystem from programs ALOC to PLNTPLAN. The two files output on magnetic tape by these programs, PLANTAPE and EVENTAPE, are represented by only their logical tape unit numbers. All other files are placed on the CDC 814 disk unit. The fourth block, common /CORRCHAR/, defines the penetration corridor characteristics. Block five, /ASMTABLE/, describes the characteristics of the air-to-surface missiles (ASMs). The next block, /PAYLOAD/, describes the contents of each weapon vehicle payload. Block /DPENREF/ next describes the locations of the depenetration and refueling points. The eighth block, common /PLANTYPE/, specifies the timing considerations for the plan. Block /WARHEAD/ describes warhead characteristics; block /WPNREG/ specifies the command and control reliability for each command and control region. Block /WPNTYPE/ specifies the weapon type characteristics. Block /WPNGRP/ specifies the weapon group characteristics. Block /EXCESS/ contains information on the number of weapons added by program PREPALOC for allocation purposes. (These excess weapons are removed by either program POSTALOC or program FOOTPRNT, in the case of MIRV weapons.) Block /NAVAL/ describes the nature of the weapon survival probability and the kill probability against targets in the NAVAL class. Finally, block /CTRYCD/ lists all the country location codes (attribute CNTRYLOC) present in the target system for this plan.

The second BASFILE section describes the characteristics of each component of a complex or multiple target. The data blocks in this section are ordered as they appear on the TGTFILE (or TINFILE). That is, the data for complex and multiple target components are shuffled in the same manner as the target complexes or multiples are shuffled. For each complex, there is a set of 29-word data blocks, one block for each component of the complex, including the "lead" element. This block contains the same type of information as is carried on the TINFILE for the target complex.

Table 5. TGTFILE Format (Target Data Block**)
(Written from Common /DYNAMIC/)
(Sheet 1 of 2)

ARRAY OR VARIABLE*	DESCRIPTION
TGTNAMZ	Target name
INDEXNZ	Target index number
DESIGZ	Target designator code
TASKZ	Target task code
CNTRYLCZ	Target country location code
FLAGZ	Target flag code
TGTMULZ	Target multiplicity
TGTLAZ	Target latitude
TGTLONZ	Target longitude
TGTRAZ	Target radius (nautical miles)
VTZ	Original target value
MZ	Number of hardness components
HZ(2)	Lethal radius (1MT) by hardness component (nautical miles)
VOZ(2)	Value by hardness component
NKZ	Number of time components
FVAZ(3)	Time component value by time component
TAZ(3)	Time component time by time component
IHCLASZ	Target class name
ICLASSZ	Target class number
IHTYPZ	Target type name (for complex targets, this is replaced by the number of components in the complex).
TARDEZ	Local bomber defense level

*Parenthetical values indicate array dimensions. All other elements are single word variables.

**There is one block of this format for each target. The blocks are ordered as shuffled by program PLANSET.

Table 5. (cont.)
(Sheet 2 of 2)

ARRAY OR VARIABLE*	DESCRIPTION
MISDEZ	Number of terminal ballistic missile defense interceptors
MINKILZ	Minimum kill probability required
MAXKILZ	Maximum kill probability desired
MAXCOSZ	Maximum (weapon cost/target value) acceptable to get MINKILZ
INDYPEZ	Depenetration corridor index
DISTDFZ	Distance target to end of depenetration corridor (nautical miles)
DISTDGZ	Distance target to recovery base (nautical miles)
DISTCD(30)	Distance corridor origin to target by penetration corridor (nautical miles)
ATTRCD(30)	Attrition corridor origin to target by penetration corridor
NFIXES	Number of fixed assignments for this target
INFIX(NFIXES)	Fixed assignment information for each assignment

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 6. BASFILE Format
(Sheet 1 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
MASTER	23	IHDATE	Date of run initiation
		IDENTNO	Run identification number
		ISIDE	Attacking side
		NRTPT	Number of route points
		NCORR	Number of penetration corridors
		NDPEN	Number of depenetration corridors
		NRECOVER	Number of recovery bases
		NREF	Number of directed refuel areas
		NBNDRY	Number of boundary points
		NREG	Number of command and control regions
		NTYPE	Number of weapon types
		NGROUP	Number of weapon groups
		NTOTBASE	Total number of bases
		NPAYLOAD	Number of payload types
		NASMTYPE	Number of ASM types
		NWHDTYPE	Number of warhead types
		NTANKBAS	Number of tanker bases
		NCOMPLEX	Number of complex targets
		NCLASS	Number of weapon classes (2)
		NALERT	Number of alert conditions (2)

Table 6. (cont.)
(Sheet 2 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
MASTER (cont.)		NTGTS	Number of targets
		NCORTYPE	Number of penetration corridor types
		NCNTRY	Number of distinct country codes
BRKPNT	545	INDBEG(250)	Smallest index number for each type
		TYPENAME(250)	Type names in order of increasing index number
		CUMNO(15)	Cumulative number of types in each class (inclusive)
		BTYPES(15)	Number of BLUE side types in each class
		INDCLAS(15)	Smallest index number in each class
FILES	16	TGTFILE(2)*	Target data file
		BASFILE(2)	Data base information file
		MSLTIME(2)	Fixed missile timing file
		ALOCTAR(2)	Weapon allocation by targets file
		TMPALOC(2)	Temporary allocation file
		ALOCGRP(2)	Allocation by group file
		STRKFIL(2)	Strike file

*First word is logical unit number; second word is maximum file length in words. Single variables are logical unit numbers.

Table 6. (cont.)
(Sheet 3 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
FILES (cont.)		EVENTAPE**	Simulator events tape
		PLANTAPE**	Detailed plans tape
CORRCHAR	601	PCLAT(30)	Latitude of corridor point
		PCLONG(30)	Longitude of corridor point
		PCZONE(30)	Defense zone in which corridor origin is located
		ZPLAT(30)	Latitude of corridor origin
		ZPLONG(30)	Longitude of corridor origin
		ENTLAT(30)	Latitude of corridor entry
		ENTLONG(30)	Longitude of corridor entry
		CRLENGTH(30)	Distance from corridor entry to corridor origin
		KORSTYLE(30)	Power of y versus x***
		ATTRCORR(30)	High altitude attrition per nautical mile unsuppressed
		ATTRSUPF(30)	High altitude attrition per nautical mile suppressed
		HILOATTR(30)	Ratio of low to high altitude attrition (less than 1)
		DEFRANGE(30)	Characteristic range of corridor defense (nautical miles)

**Output on magnetic tape.

***See program POSTALOC.

Table 6. (cont.)
(Sheet 4 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
CORRCHAR (cont.)		NPRCRDEF(30)	Number of attrition sections this corridor
		DEFDIST(30,3)	Distance of precorridor leg
		ATTRPRE(30,3)	Attrition in this precorridor leg
		NDATA	Number of words in common /CORRCHAR/
ASMTABLE	100	IWHDASM(20)	Warhead index } for ASMs
		RANGEASM(20)	Range
		RELASM(20)	Reliability
		CEPASM(20)	CEP
		SPEEDASM(20)	Speed
PAYLOAD	400	NOBOMB1(40)	Number of type 1 bombs
		IWHD1(40)	Type 1 warhead index
		NOBOMB2(40)	Number of type 2 bombs
		IWHD2(40)	Type 2 warhead index
		NASM(40)	Number of ASMs
		IASM(40)	ASM index
		NCM(40)	Number of countermeasures
		NDECOYS(40)	Number of terminal decoys
		NADECOYS(40)	Number of area decoys
		IMIRV(40)	MIRV system identification number

Table 6. (cont.)
(Sheet 5 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
DPENREF	190	DPLINK(50)	Depenetration point link
		DPLAT(50)	Depenetration point latitude
		DPLONG(50)	Depenetration point longitude
		RFLAT(20)	Refuel point latitude
		RFLONG(20)	Refuel point longitude
PLANTYPE	3	INITSTRK	Indicator for first or second strike
		CORMSL	Coordination time parameter for missiles
		CORBOMB	Coordination distance for bombers
WARHEAD	150	YLD(50)	Weapon yield
		PDUD(50)	Weapon dud probability
		FFRAC(50)	Fission fraction
WPNREG	20	CCREL(20)	Command and control reliability
WPNTYPE*	1040	RANGE(80)	Range of vehicle (nautical miles)
		CEP(80)	CEP (nautical miles)
		SPEED(80)	Speed (knots)
		ALERTDLY(80)	Alert delay } (hours)
		NALRTDLY(80)	Nonalert delay } (hours)

*Blocks WPNTYPE and WPNGRP combined in program PREPALOC to common /WPNDATA/.

Table 6. (cont.)
(Sheet 6 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
WPNTYPE* (cont.)		RANGEDEC(80)	Low/high altitude fuel consumption ratio
		ICLASS(80)	Weapon class
		RANGEREF(80)	Refueled range (nautical miles)
		REL(80)	Reliability
		IRECMODE(80)	Recovery mode
		IPENMODE(80)	Penetration mode
		ISIMTYPE(80)	Hollerith type name
		FUNCTION(80)	Function code
WPNGRP*	8200	NWPNS(200)	Number of weapons
		NVEHGRP(200)	Number of vehicles
		WLAT(200)	Centroid latitude
		WLONG(200)	Centroid longitude
		IREG(200)	Command and control region
		ITYPE(200)	Type index (LTYPE)
		IALERT(200)	Alert status
		SBL(200)	Probability of survival before launch
		IREFUEL(200)	Refuel code for bombers payload index for missiles
		YIELD(200)	Weapon yield (megatons)

*Blocks WPNTYPE and WPNGRP combined in program PREPALOC to common /WPNDATA/

Table 6. (cont.)
(Sheet 7 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
WPNGRP* (cont.)		REFTIME(200)	Refuel time
		EXPASM(200)	Fraction of ASMs in each group
		DISTAC(200,29)	Distance to corridor origins for each group (corridors 2-30)
EXCESS	207	NEXCESS	Number of words in this block
		PEXBOMB	Percentage of weapons added to bomber groups in PREPALOC
		EXNBOMB	Number of vehicle loads added to bomber groups in PREPALOC
		PEXMIRV	Same as PEXBOMB for groups with MIRV capability
		EXBMIRV	Same as EXNBOMB for groups with MIRV capability
		PEXMISS	Same as PEXBOMB for non-MIRV missile groups
		EXNMISS	Same as EXNBOMB for non-MIRV missile groups
		SBLREAL(200)	Actual SBL probability
NAVAL	401	NNAVAL	Number of words in this block
		IDBL(200)	Index to time dependent DBL data tables
		PKNAV(200)	Single shot kill probability against NAVAL targets
CTRYCD	150	CTRYCD(150)	List of country location codes
--	1	RRRRRRRR	=8HRRRRRRRR end sentinel for first section

*Blocks WPNTYPE and WPNGRP combined in program PREPALOC to common /WPNDATA/

Table 6. (cont.)
(Sheet 8 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
--	(29 x number of complex target components) + (8 x number of multiple target components)	TGTNAME*	29-word record as TINFILE for each complex target component
		.	
		.	
		.	
		MAXCOST	
		TGTNAME*	8-word record for each multiple target element; variables defined as on TINFILE
		INDEXNO	
		DESIG	
		TASK	
		CNTRYLOC	
		FLAG	
		TGTLAT	
		TGTLONG	
		ZZZZZZZZ	=8HZZZZZZZZ end sentinel for second section
(one block for each weapon group)	(10 x number of groups) + (5 x number of occurrences of bases)	IGROUP	Group number
		NWPNS	Number of weapons
		NVEHGRP	Number of vehicles
		IREG	Command and control region
		ITYPE	Weapon type index (LTYPE)

*In order as complex or multiple target appears on target file, TGTFILE.

Table 6. (cont.)
(Sheet 9 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
		IALERT	Alert status
		IREFUEL	Refuel code
		YIELD	Yield
		ISTART	Starting weapon index
		NBASE	Number of bases
	One five-word Block for Each Base (i.e., NBASE)	IBASE	Base index number
		BLAT	Base latitude
		BLONG	Base longitude
		IPAYLOAD	Payload index
		VONBASE	Number on base and index of starting vehicle
(one block for each weapon type)	(18 x number of weapon types)	ISIMTYPE	Hollerith type name
		RANGE	Range
		CEP	CEP
		SPEED	Speed
		ALERTDLY	Alert delay
		NALRTDLY	Nonalert delay
		RANGEDEC	Low/high altitude fuel consumption ratio
		ICLASS	Weapon class number
		NOPERSQN	Number per squadron
		SPDHI	High altitude speed

Table 6. (cont.)
(Sheet 10 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
		SPDLO	Low altitude speed
		SPDASH	Dash speed
		RANGEREF	Refueled range
		NMPSITE	Number per site
		IREP	Reprogramming index
		IRECMODE	Recovery mode
		IPENMODE	Penetration mode
		FUNCTION	Function code
		YYYYYYYY	= 8HYYYYYYYY end sentinel of third section
BOUNDARY	1000	BPLINK(200)	Boundary point link
		BPLAT(200)	Boundary point latitude
		BPLONG(200)	Boundary point longitude
		BPZONE(200)	Boundary point zone
		NEXTZONE(200)	Boundary point adjacent zone
CHARTER	160	KOUNT(30)	Route point index associated with IHAP
		IHAP(30)	Corridor pointer: /CORRCHAR/ to /HAPPEN/
		MOUNT(50)	Route point index associated with JHAP
		JHAP(50)	Pointer: /OPENREF/ to /HAPPEN/
HAPPEN	1000	JAPTYPE(250)	Go low indicator

Table 6. (cont.)
(Sheet 11 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
HAPPEN (cont.)		HAPLAT(250)	Dogleg latitude
		HAPLONG(250)	Dogleg longitude
		HAPDIST(250)	Length of dogleg
RECOVR	1100	RCBLTX(50)	First recovery base latitude*
		RCBLNX(50)	First recovery base longitude*
		RCBLAT(50,4)	Latitude of actual recovery base**
		RCBLON(50,4)	Longitude of actual recovery base
		INDBAS(50,4)	Recovery base index number
		INDCAP(50,4)	Recovery base capacity
		DIST(50,4)	Distance from depenetration point to recovery base
--	(12 x number of tanker bases)***	INDEXTK	Tanker base index number
		TKLAT	Tanker base latitude
		TKLONG	Tanker base longitude
		IREFTK	Tanker base refuel area
		NPSQNTK	Number of tankers per base or squadron
		NALRTK	Number of alert tankers per squadron

*Indexed by depenetration Corridor.

**Indexed by depenetration corridor and base order.

***There is one of these 12-word blocks for each tanker base.

Table 6. (cont.)
(Sheet 12 of 12)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
--	(12 x number of tanker bases)***	SPEEDTK	Tanker Speed
		DLYALTK	Alert delay
		DLYNLTK	Nonalert delay
		TTOS	Total time on station
		ITYPETK	Tanker type index
		RANGE	Tanker range
--	1	XXXXXXXX	= 81IXXXXXXXX end sentinel for fourth section

***There is one of these 12-word blocks for each tanker base.

For each multiple target, there is a set of eight-word data blocks, one block for each multiple target component. These eight words identify the values of those characteristics that vary from component to component in a multiple target.

The third BASFILE section consists of detailed weapon group data. The first part of this section consists of a set of data blocks (common /GRPDATA/), one block for each weapon group. Each of these blocks describes the weapon group characteristics. Each block is terminated by a set of five-word blocks that describe the location and characteristics of each base that is used by weapons in the group. There is one five-word block for each base used by the group. The latter part of the third BASFILE section describes the weapon type characteristics. There is an 18-word block for each weapon type describing the weapon type characteristics.

The fourth and final BASFILE section describes miscellaneous data required by later processors. The first block in this section, common /BOUNDARY/, identifies the location of all zone boundaries. Common /CHARTER/ comes next with various route point indices and pointers. Common /HAPPEN/ follows with a description of each leg of the penetration and depenetration corridors. Data on location and capacity of all bomber recovery bases follows in block /RECOV/. Finally, the BASFILE is terminated by block /TANKER/, which identifies the characteristics of the tanker vehicles on each tanker base.

CONCEPT OF OPERATION

Program PREPALOC is divided into three computation modules. The first module is the target planning factor change module. These subprograms allow the user to modify the target characteristics value, MINKILL and MAXKILL. The second module, the fixed assignment module, allows the user to direct the assignment of weapons to specific targets. The third module, the data precomputation module, prepares the data files BASFILE and TGTFILE to be used by the remaining processors of the Plan Generation subsystem.

The target planning factor change module is composed of subroutines VALUMOD, MINMOD, and MAXMOD. These three subroutines all perform a similar function. They read the desired target class, type, identification (target designator code or index number), and the new value to be assigned to the appropriate target variable. These data are stored in core memory for use by subroutine MAKECHG. These three subroutines merely read the data, store the data, and store the amount of data read in and stored. Subroutine MAKECHG of the data precomputation module actually effects the

changes. If there are any changes in target value, all target values are renormalized so that the sum of all target values is 1,000. Subroutine SETFILE is the only major subprogram in the fixed weapon assignment module. This subroutine reads the fixed assignment data cards. The data on the cards are packed so that the blank fields on the cards are eliminated. This packing will reduce the requirement for temporary disk storage. The packed data are then output to a temporary disk file called FIXFILE. The format of this file is displayed in table 7. This file is read by subroutine FIXWEAP of the data precomputation module to effect the fixed assignments.

Table 7. Format of FIXFILE File

For each target on which there are fixed assignments there is a data block as follows:

WORD	DESCRIPTION
1	Target identifier (DESIG or INDEXNO or target number)
2	First fixed assignment 3 characters - group number 5 characters - time of arrival
3	Second fixed assignment data
.	
.	
.	
NFIX + 1	Last fixed assignment; there are NFIX assignment requests on this target
NFIX + 2	Block terminator = 8HENDBLOCK

These data blocks appear in the order in which target identifiers are encountered in the user-input data.

The data precomputation module is by far the largest in program PREPALOC. There are five major functions in this module:

1. Preparation of geographic data (subroutine ROUTING),
2. Preparation of weapon group information (subroutine WEAPPREP),
3. Preparation of basic target information (subroutine TGTPREP),
4. Implementation of target factor change requests (subroutine MAKECHG),
5. Implementation of fixed weapon assignment requests (subroutine FIXWEAP).

These five functions are discussed in order as follows.

Preparation of Geographic Data (Subroutine ROUTING)

This subroutine processes bomber routing data from WINFILE and calculates distance and attrition factors associated with bomber penetration and depenetration corridors.

Distances are computed throughout the Plan Generation subsystem by a utility subroutine, DISTF. This subroutine computes distances using a great circle distance formula for a spherical earth. For neighboring points (separated by less than two degrees longitude), an approximation is used to minimize computation time. The maximum error of the approximation is less than 0.1 percent.

This subroutine is called by PREPALOC to process the routing information contained on WINFILE. It processes penetration corridor data (precorridor legs) and stores detailed descriptions of these corridors in common /HAPPEN/ and a summary description of these corridors in common /CORRCHAR/. The pointers which equate these two common blocks are contained in /CHARTER/. Detailed descriptions of the depenetration corridors are also stored in /HAPPEN/. Depenetration and refuel data are stored in the common block /DPENREF/.

As the distances for both precorridor legs and the depenetration corridors are accumulated, they are stored in /PREALOC/ for use by subroutines WEAPPREP and TGTPREP.

Subroutine ROUTING processes the WINFILE data in the order in which it is stored: (1) header data, (2) data pertaining to the precorridor legs, (3) data pertaining to the depenetration corridors, (4) recovery point and refuel point data, and (5) defense zone boundary data.

The header data include information such as the numbers of corridors, depenetration corridors, recovery bases, refuel bases or areas, regions, weapon groups, and tanker bases which are stored in /MASTER/ for use by the Plan Generator as a whole. For each penetration corridor, the latitude

and longitude for each dogleg point are obtained, together with a flag noting whether the point begins or ends an attrition section (JAPTYPE). The distance from the previous point is also noted. This information is stored in /HAPPEN/. Then the length and attrition of each separate attrition section of the corridor are computed and stored in /CORRCHAR/. Finally the overall corridor distance and overall attrition are accumulated and stored in /CORRCHAR/. A similar processing is carried out for the depenetration corridors, which are written into /HAPPEN/. At this point the recovery bases for each depenetration point are processed. The data is stored in common /RECOVR/ for later writing on tape. The defense zone data are copied directly into /BOUNDARY/ from WINFILE.

Preparation of Weapon Group Information (Subroutine WEAPPREP)

The purpose of this subroutine is to read weapon data from WINFILE and to store it in arrays for the BASFILE.

This subroutine is called by PREPALOC; it reads WINFILE and fills arrays in the following common blocks: /WARHEAD/ is filled with information on warhead yield and dud probability; /ASMTABLE/ is filled with information on range, reliability, speed, and CEPs for ASMs; /PAYLOAD/ is filled with information on numbers of bombs and their types, as well as the number of ASMs and decoys carried by each bomber; /WPNREG/ is filled with command and control reliability information which is specified by region; /WPNDATA/ is filled with weapon data by type and also with weapon data specific to individual weapon groups. The third section of the BASFILE contains weapon data organized by weapon type and group respectively, but in an unindexed form.

When called by PREPALOC, this subroutine first copies complex and multiple target data directly from the WINFILE onto the POSTDATA file. This latter file is a temporary (scratch) file which is composed of the second, third, and fourth sections of the BASFILE. This information is stored on the POSTDATA file temporarily while the data for the first section of the BASFILE is being generated. Note that, if there are target factor change requests input by the user, there is a temporary POSTDATA file which contains unnormalized target factors. Otherwise, the file formats are the same.

After this, the warhead, ASM, and payload data are read and copied directly into their respective common blocks. The weapon data for each weapon type are next read and stored in indexed form in the first part of common /WPNDATA/. Similarly, weapon data for individual groups are stored, indexed, in the latter part of common /WPNDATA/. The same data are then written onto the POSTDATA file, unindexed, for use by program POSTALOC.

Distances from each weapon group to corridor origin are also computed at this time. Once this has been done, the common blocks /BOUNDARY/, /CHARTER/, /RECOVR/, and /HAPPEN/ are ready to be written onto the POSTDATA file.

Preparation of Basic Target Information (Subroutine TGTPREP)

The purpose of this subroutine is to prepare target data for program ALOC.

This subroutine is called by PREPALOC; it reads target data from TINFILE and appends distance information to it before writing it out in proper form for the allocator.

As each record of target information is read from TINFILE, the input data are rearranged and the depenetration corridor closest to the target is found and its index is noted. The distance to the target through this depenetration corridor is calculated, and the distance from the target to the recovery base through this depenetration corridor is noted. In addition, the distance to the target from each penetration corridor is computed and stored together with any associated attrition. At this time, the target planning factor changes are effected, and the fixed weapon assignment requests are implemented by the subroutines MAKECHG and FIXWEAP respectively. The TGTFILE must be written twice if either value changes or fixed assignments are requested. Two passes are required for value change so that the target values can be renormalized to a total sum of 1,000. Two passes are required for fixing weapon assignments, because there is insufficient room to store all the data in core memory to perform the necessary calculations in one pass. Thus, subroutine TGTPREP writes the TGTFILE on its normal logical unit only if there are no value change requests or fixed assignments. Otherwise, a temporary file is set up so that the actual TGTFILE can be written later. This temporary file, the TMPTAR file, has a format similar to the TGTFILE (see table 5). Each target is represented by a 94-word block which is the same as the first 94 words in the TGTFILE blocks (i.e., from TGTNAMZ to ATTRCD(30)). In addition, this subroutine prepares a list of the country codes (CNTRYLOC) which are used in the TGTFILE. These data will be passed to program ALOC on the BASFILE so that weapon restrictions by this attribute can be performed.

Implementation of Target Factor Change Requests (Subroutine MAKECHG)

This subroutine is called once during the processing of each target by TGTPREP. Subroutine MAKECHG checks for a change request that matches the target. If a request matches the target data, the new value of MINKILL, MAXKILL, or VTO (value) is inserted on the TGTFILE. If any value changes

were requested, a sum of old and new target values is maintained so that the values can be renormalized. If the target is a complex target, the data on the POSTDATA file are investigated to determine if any of the target components match a change request. If so, the complex factors are modified accordingly. A target may have any combination of the three planning factors changed. In the case of complex targets, any change request for modifying a factor for the complex as a whole overrides any requests for modifying the same factor in one of the components. For a multiple target, changing a factor for one of the components produces the identical change in all of the components of that target.

Implementation of Fixed Weapon Assignment Requests (Subroutine FIXWEAP)

This subroutine adds the fixed assignment data to the TGTFILE. It first loads the fixed assignment data from the temporary file written by SETFILE into core storage no longer needed for weapon or target data processing. Then the routine reads the temporary TGTFILE one target at a time. If there were value change requests, each value is multiplied by the value renormalization variable. Otherwise, the target data are merely transferred to the TGTFILE from the temporary file. The list of fixed assignment identifiers is then searched for a match on the current target. If there is no match, the routine writes out zero for the number of fixes on this target. If there is a match (or matches), the fixed assignment data are added to the end of the regular target data block. This process continues until the entire temporary file has been processed.

Identification of Subroutines Performing Each Function

Program PREPALOC: This is the main program. It acts as a control driver for the subroutines. It reads the function commands (described in the User's Manual, Volume II, Chapter 3, Plan Generation Subsystem, Program PREPALOC, Input) and determines the sequence of calls to the various subroutines that will produce the desired results.

Subroutine VALUMOD: This subroutine reads the value change request cards and stores them in a change request list in common /CHANGES/. The actual investigation and performance of the changes are completed in subroutine MAKECHG.

Subroutine MINMOD: This subroutine reads the MINKILL change requests and stores them in the change request list. Again, the actual change is performed in MAKECHG. (This subroutine is an entry in subroutine VALUMOD.)

Subroutine MAXMOD: This routine is the same as the previous subroutine, except that it reads MAXKILL change requests. (This subroutine is an entry in subroutine VALUMOD.)

Subroutine SETFILE: This subroutine will read the user-input cards which contain the fixed weapon assignment information. The information will be packed and written on a temporary disk file for use by subroutine FIXWEAP.

Subroutine ROUTING: This subroutine performs the preliminary geographic calculations required to define the air defense zones and the bomber penetration corridors as well as other geographic data. This subroutine computes the length and route of each corridor. It calculates the sequence of boundaries that defines each zone. In the course of processing, ROUTING computes:

1. Identification, location, and distances for each refuel area and recovery base (/DPENREF/)

2. Length, location, route identification, and attrition for each penetration corridor (/CORRCHAR/)

3. Identification and linking of all zone boundaries and route points (/HAPPEN/, /CHARTER/, and /BOUNDRY/).

Subroutine WEAPPREP: This subroutine aggregates information on the various weapon types. Such data as speed and reliability are saved and indexed by weapon type. Location, number of weapons and vehicles, and other pertinent data are saved and indexed by weapon group. In addition, data on warheads, payloads, and ASMs are stored for output on the BASFILE.

Subroutine TGTPREP: This subroutine retrieves the raw target data from the TINFILE and places it, in somewhat different format, on the BASFILE and TGTFILE. (If fixed assignments or value changes are to be made in a run, a temporary TGTFILE is written by TGTPREP. Subroutine FIXWEAP or subroutine NORMALZ will then write the actual TGTFILE.) The target data prepared by this subroutine will be used only by program ALOC to determine the weapon-target interactions.

Subroutine MAKECHG: This subroutine performs the actual modifications of target planning factors as requested by the user.

Subroutine FIXWEAP: This routine reads the temporary file written by SETFILE. Since the fixed assignment data require a large amount of storage (up to 20,000 words), these data cannot be maintained in core memory during the other processing functions. After reading the file, FIXWEAP will write a TGTFILE containing the fixed assignment data. The basic target data will be stored in a temporary disk file by TGTPREP so that FIXWEAP can add the fixed assignment information separate from the remainder of target data processing.

Subroutine BASWRIT: This subroutine performs the actual writing of the BASFILE. It merges the information in core (section one of the BASFILE)

and the POSTDATA file (sections two, three, and four of the BASFILE) to complete the BASFILE. If there were target value changes the value data in section two of the BASFILE is normalized by subroutine BASWRIT.

Subroutine NORMALZ: If there were target planning factor change requests but no fixed weapon assignments, this subroutine writes the normalized TGTFILE.

Subroutine CHKCHG: This subroutine checks that all target factor change requests were implemented.

Subroutine RDPRCMP: This subroutine reads the user-input parameter cards for the data precomputation module.

Subroutine PRINTDAT: This subroutine controls the operation of all optional prints available in the data precomputation module. They include the following:

1. Listing of routing data
2. Tape dump of input routing data
3. Listing of weapon groups for each corridor
4. Tape dump of input weapon data
5. Listing of target data
6. Tape dump of input target data
7. Modified target data and fixed weapon assignments.

COMMON BLOCK DEFINITION

External Common Blocks

The common blocks used by program PREPALOC in processing input/output (I/O) files are shown in table 8.

Input From Files: TINFILE and WINFILE: Almost all information stored from these files is read into common block /INPSTOR/ which is used as an input storage buffer. For the value changing options this block is redefined to provide for more convenient input/output.

Output Data For TGTFILE: Block /INPSTOR/ is used for output to the TGTFILE. The fixed assignment data is added by reusing this block after the initial 95 words are output for each target.

Internal Common Blocks

In addition to the common blocks associated with I/O operations, the common blocks described in table 9 are used internally by program PREPALOC.

Table 8. Program PREPALOC External Common Blocks
(Sheet 1 of 8)

INPUT FROM TINFILE AND WINFILE

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
INPSTOR (Subroutines PREPALOC, ROUTING, WEAPPREP, TGTPREP, PRINTDAT)	BLOCK(1600)	Temporary storage area. (also called NLOCK)
INPSTOR (Subroutines RDPRCMP, BASWRIT, FIXWEAP, MAKECHG, NORMALZ)	TGTNAMZ	Target name
	INDEXNZ	Target index number
	DESIGZ	Target designator code
	TASKZ	Target task code
	CNTRYLCZ	Target country location code
	FLAGZ	Target flag code
	TGTMULZ	Target multiplicity
	TGTLAZ	Target latitude
	TGTLONZ	Target longitude
	TGTRAZ	Target radius (nautical miles)
	VTZ	Original target value
	MZ	Number of hardness components
	HZ(2)	Lethal radius (1MT) by hardness component (nautical miles)
	VOZ(2)	Value by hardness component
	NKZ	Number of time components
	FVAZ(3)	Time component value by time component
	TAZ(3)	Time component time by time component
	IHCLASZ	Target class name
	ICLASSZ	Target class number
	IHTYPZ	Target type name [For complex targets, this is replaced by the number of components in the complex.]
	TARDEZ	Local bomber defense level

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 8. (cont.)
(Sheet 2 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
INPSTOR (cont.)	MISDEZ	Number of terminal ballistic missile defense interceptors
	MINKILZ	Minimum kill probability required
	MAXKILZ	Maximum kill probability desired
	MAXCOSZ	Maximum (weapon cost/target value) acceptable to get MINKILZ
	INDYPEZ	Depenetration corridor index
	DISTDFZ	Distance target to end of depenetration corridor (nautical miles)
	DISTDGZ	Distance target to recovery base (nautical miles)
	DISTCD(30)	Distance corridor origin to target by penetration corridor (nautical miles)
	ATTRCD(30)	Attrition corridor origin to target by penetration corridor
	NFIXES	Number of fixed assignments for this target
	MULDATA(8,5)	Temporary storage area for TINFILE data on components of multiple targets (Space for 5 blocks of 8 words each.)
	ICXDATA(29,40)	Temporary storage area for TINFILE data on components of complex targets. (Space for 40 blocks of 29 words each.)

OUTPUT ON FILE BASFILE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
MASTER	IHDATE	Date of run initiation
	IDENTNO	Run identification number
	ISIDE	Attacking side
	NRTPT	Number of route points
	NCORR	Number of penetration corridors
	NDPEN	Number of depenetration corridors
	NRECOVER	Number of recovery bases
	NREF	Number of directed refuel areas
	NBNDRY	Number of boundary points

Table 8. (cont.)
(Sheet 3 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
MASTER (cont.)	NREG	Number of command & control regions
	NTYPE	Number of weapon types
	NGROUP	Number of weapon groups
	NTOTBASE	Total number of bases
	NPAYLOAD	Number of payload types
	NASMTYPE	Number of ASM types
	NMIDTYPE	Number of warhead types
	NTANKBAS	Number of tanker bases
	NCOMPLEX	Number of complex targets
	NCLASS	Number of weapon classes (2)
	NALERT	Number of alert conditions (2)
	NTGTS	Number of targets
	NCORTYPE	Number of penetration corridor types
	NCNTRY	Number of distinct country codes
BRKPNT	INDBEG(250)	Smallest index number for each type
	TYPENAME(250)	Type names in order of increasing index number
	CUMNO(15)	Cumulative number of types in each class (inclusive)
	BTYPES(15)	Smallest index number in each class
	MTARTYP	Maximum number of target types
	MTARCLS	Maximum number of target classes
FILES	TGTFILE(2)*	Target data file
	BASFILE(2)	Data base information file
	MASLTIME(2)	Fixed missile timing file
	ALOCTAR(2)	Weapon allocation by targets file
	TMPALOC(2)	Temporary allocation file
	ALOCGRP(2)	Allocation by group file
	STRKFIL(2)	Strike file
	EVENTAPE ǂ	Simulator events tape
	PLANTAPE ǂ	Detailed plans tape

* In two word arrays, first word is logical unit number; second word is maximum file length in words. Single variables are logical unit numbers.

ǂ These files are output on magnetic tape.

Table 8. (cont.)
(Sheet 4 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
CORRCHAR	PCLAT(30)	Latitude of corridor point
	PCLONG(30)	Longitude of corridor point
	PCZONE(30)	Defense zone in which corridor origin is located
	ZPLAT(30)	Latitude of corridor origin
	ZPLONG(30)	Longitude of corridor origin
	ENTLAT(30)	Latitude of corridor entry
	ENTLONG(30)	Longitude of corridor entry
	CRLENGTH(30)	Distance from corridor entry to corridor origin
	KORSTYLE(30)	Power of y versus x ‡
	ATTCORR(30)	High altitude attrition per nautical mile unsuppressed
	ATTRSUPP(30)	High altitude attrition per nautical mile suppressed
	HILOATTR(30)	Ratio low to high altitude attrition (less than 1)
	DEFRANGE(30)	Characteristic range of corridor defense (nautical miles)
	NPRCRDEF(30)	Number of attrition sections this corridor
	DEFDIST(30,3)	Distance of each precorridor leg
	ATTRPRE(30,3)	Attrition in each precorridor leg
	NDATA	Number of words in common/CORRCHAR/
	LMAX	Maximum number of precorridor legs
ASMTABLE	IWHDASM(20)	Warhead index } for ASMs
	RANGEASM(20)	Range
	RELASM(20)	Reliability
	CEPASM(20)	CEP
	SPEEDASM(20)	Speed
	MASMTYPE	Maximum number of ASM types
PAYLOAD	NOBOMB1(40)	Number of type 1 bombs
	IWHD1(40)	Type 1 warhead index
	NOBOMB2(40)	Number of type 2 bombs
	IWHD2(40)	Type 2 warhead index
	NASM(40)	Number of ASMs
	IASM(40)	ASM index
	NCM(40)	Number of countermeasures
	NDECOYS(40)	Number of terminal decoys

‡See program POSTALOC

Table 8. (cont.)
(Sheet 5 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PAYLOAD (cont.)	NADECOYS(40)	Number of area decoys
	IMIRV(40)	MIRV system identification number
	MPAYLOAD	Maximum number of payload types
DPENREF	DPLINK(50)	Depenetration point link
	DPLAT(50)	Depenetration point latitude
	DPLONG(50)	Depenetration point longitude
	RFLAT(20)	Refuel point latitude
	RFLONG(20)	Refuel point longitude
	MDPEN	Maximum number of depenetration points
	MREF	Maximum number of refueling points
PLANTYPE	INITSTRK	Indicator for first or second strike
	CORMSL	Coordination time parameter for missiles
	CORBOMB	Coordination distance for bombers
WARHEAD	YLD(50)	Weapon yield
	PDUD(50)	Weapon dud probability
	FFRAC(50)	Fission fraction
	MWHDTYPE	Maximum number of warhead types
WPNREG	CCREL(20)	Command and control reliability by command and control region
WPNDATA*	RANGE(80)	Range of vehicle (nautical miles)
	CEP(80)	CEP (nautical miles)
	SPEED(80)	Speed (knots)
	ALERTDLY(80)	Alert delay } (hours)
	NALRTDLY(80)	Nonalert delay } (hours)
	RANGEDEC(80)	Low/high altitude fuel consumption ratio
	ICLASS(80)	Weapon class
	RANGEREF(80)	Refueled range (nautical miles)
	REL(80)	Reliability
	IRECMODE(80)	Recovery mode
	IPENMODE(80)	Penetration mode

*This block is redefined for internal use by program PREPALOC. See definition of internal common blocks for its internal definition. The arrays of length 80 are indexed by weapon type index (LTYPE). The arrays of length 200 are indexed by weapon group number.

Table 8. (cont.)
(Sheet 6 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
WPNDATA (cont.)	ISIMTYPE(80)	Hollerith type name
	FUNCTION(80)	Function code
	NWPNS(200)	Number of weapons
	NVEHGRP(200)	Number of vehicles
	WLAT(200)	Centroid latitude
	WLONG(200)	Centroid longitude
	IREG(200)	Command & control region
	ITYPE(200)	Type index [LTYPE]
	IALERT(200)	Alert status
	SBL(200)	Probability of survival before launch
	IREFUEL(200)	Indicates refuel code for bombers or payload index for missiles
	YIELD(200)	Weapon yield (megatons)
	REFTIME(200)	Refuel time
	DISTAC (200,30)	Distance to corridor origins for each group**
	MTYPE	Maximum number of weapon types
	MGROUP	Maximum number of weapon groups
	MDESREQ	Maximum number of target identifiers for fixed assignments
	NEXCESS	Number of words in this block
EXCESS	PEXBOMB	Percentage of weapons added to bomber groups in PREPALOC
	EXNBOMB	Number of vehicle loads added to bomber groups in PREPALOC
	PEXMIRV	Same as PEXBOMB for groups with MIRV capability
	EXBMIRV	Same as EXNBOMB for groups with MIRV capability
	PEXMISS	Same as PEXBOMB for non-MIRV missile groups
	EXNMISS	Same as EXNBOMB for non-MIRV missile groups
	SBLREAL(200)	Actual SBL probability for each group

** The first 200 words of this array are equivalenced in subroutine WEAPPREP to the array EXPASM which is the fraction of ASM weapons in each group.

Table 8. (cont.)
(Sheet 7 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
NAVAL	NNAVAL	Number of words in this block
	IDBL(200)	Index to time dependent DBL data tables
	PKNAV(200)	Single shot kill probability against NAVAL targets
CTRYCD	CTRYCD(150)	List of country location codes
	MCNTRY	Maximum number of distinct country location codes per side
INPSTOR	------	This block as defined previously is used to output the data on components of complex and multiple targets
BOUNDARY	BPLINK(200)	Boundary point link
	BPLAT(200)	Boundary point latitude
	BPLONG(200)	Boundary point longitude
	BPZONE(200)	Boundary point zone
	NEXTZONE(200)	Boundary point adjacent zone
	MBNDRY	Maximum number of boundary points
CHARTER	KOUNT(30)	Route point index associated with IHAP
	IHAP(30)	Corridor pointer: /CORRCHAR/ to /HAPPEN/
	MOUNT(50)	Route point index associated with JHAP
	JHAP(50)	Pointer: /DPENREF/ to /HAPPEN/
	MCORR	Maximum number of penetration corridors
HAPPEN	JAPTYPE(250)	Go low indicator
	HAPLAT(250)	Dogleg latitude
	HAPLONG(250)	Dogleg longitude
	HAPDIST(250)	Length of dogleg
	MRTPT	Maximum number of route points

Table 8. (cont.)
(Sheet 8 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
RECOVR	RCBLAT(50,4)	Latitude of actual recovery base*
	RCBLON(50,4)	Longitude of actual recovery base*
	INDBAS(50,4)	Recovery base index number *
	INDCAP(50,4)	Recovery base capacity *
	DISTR(50,4)	Distance from depenetration point to recovery base *
	RCBLTX(50)	First recovery base latitude**
	RCBLNX(50)	First recovery base longitude **

* Indexed by depenetration corridor and base order by increasing distance from depenetration point.
** Indexed by depenetration corridor.

Table 9. Program PREPALOC Internal Common Blocks
(Sheet 1 of 5)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
PREALOC	PCLINK(30)	Penetration corridor point link
	RPLINK(150)	Routing point link
	RPLAT(150)	Latitude of routing point
	RPLONG(150)	Longitude of routing point
	ATTRLEG(150)	Attrition associated with routing leg
	DISTEF(45)	Length of depenetration corridor
	DISTEG(45)	Distance from depenetration corridor entry to recovery point
	RECLINK(180)	Recovery point link
	RECLAT(180)	Latitude of recovery point
	RECLONG(180)	Longitude of recovery point
	IRECPCTY(180)	Recovery base capacity
	INDREC(180)	Index number of recovery base
	IBEGIN(30)	Index to beginning of corridor
	PCTYPE(30)	Penetration corridor type index
	ATTRBC(30)	Attrition in corridor
WPNDATA**	MYFIXD(5000)	Target identifiers for fixed assignments
	MYDEXST(5000)	Index to fixed information array in common /CHANGES/ for fixed assignments for target
DUMCORR	NDUMCORR	Number of "dummy" corridors (i.e., those with no associated geography)
NOWFILE	WINFILE	Logical unit number for WINFILE
	TINFILE	Logical unit number for TINFILE
	POSTDATA	Logical unit number for temporary BASFILE (scratch)
	FIXFILE	Logical unit number for scratch file for temporary storage of fixed assignment information
	TMPTAR	Logical unit number of scratch file which is temporary TGTFILE
	TMPOST	Logical unit number for scratch file which is temporary POSTDATA

* Parenthetical values indicate array dimensions. All other elements are single word variables.

** This block is alternately defined for output of weapon descriptors on the BASFILE. The previous section describes this alternate definition.

Table 9. (cont.)
(Sheet 2 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
IODUMMY	INPUT(10)	Temporary storage for user-input parameter card
	NVARS	Number of parameters on user-input parameter card
	NAMES(40)	Names of user-input parameters
	INVALU(2,40)	Input values for user-input parameters
	INDEX1(40)	First array index for user-input parameter
	INDEX2(40)	Second array index for user-input parameter
	INDEX3(40)	Third array index for user-input parameter
	MORE	Input termination indicator
MACHINE	IREAD	Logical unit number for standard input
	IWRIT	Logical unit number for standard output
	ICOMM	Logical unit number for output comment medium
	IPUNCH	Logical unit number for standard punch
DEFAULT	MYNAME(20)	Names of possible user-input parameters
	MYFORM(20)	I/O format for parameter
	MYTYPE(20)	Setting mode of parameter ('DEFAULT' or 'INPUT')
	MYVAL(20)	(Equivalenced to FVAL(20)) Parameter default value
	NDEFLT	Number of parameters available for input
OPTION	ICHANGE	Nonzero if target planning factors changed
	IFIXTGT	Nonzero if fixed assignments requested. (If negative, TGTNUM option exercised in FIXASSGN option.)
	RATIO	Target value multiplier

Table 9. (cont.)
(Sheet 3 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
OPTION (cont.)	INDVAL(2)	Beginning and ending index for target value change requests
	INDMIN(2)	Beginning and ending index for target MINKILL change requests
	INDMAX(2)	Beginning and ending index for target MAXKILL change requests
	NDESREQ	Number of target identifiers in fixed assignment requests
	NFIXREQ	Number of weapons requested for fixed assignment
CHANGES	ICLASWAN(2000)	Desired class name for change request
	ITYPEWAN(2000)	Desired type name for change request
	IDENTWAN(2000)	Desired target identifier for change request
	VALUENEW(2000)	Desired new data for change request
	IFOUND(2000)	Change request fulfillment counter
	INFIX(10000)	Storage area for fixed assignment information (equivalenced to previous five variables)
	MCHANGE	Maximum number of factor change requests
	MFIXREQ	Maximum number of weapons requested for fixed assignment
SUMS	OLDSUM	Sum of target values as input on TINFILE
	SUMNEW	Unnormalized sum of target values as changed by PREPALOC
	NAMCLAS(20)	Target class names
	FRCLAS(20)	Fraction of total target value in each target class
TEMPO	LT(50)	Temporary storage - route point latitude
	LN(50)	Temporary storage - route point longitude
	JT(50)	Temporary storage - route point type indicator
	DT(50)	Temporary storage - route leg length

Table 9. (cont.)
(Sheet 4 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION	
BOUND	X1	Initial point latitude	Temporary Storage For Distance Calculations
	Y1	Initial point longitude	
	X2	Final point latitude	
	Y2	Final point longitude	
	IZN	Index to initial point zone	
	XR	Intermediate point latitude	
	YR	Intermediate point longitude	
	NZN	Intermediate point zone	
	IZIT	Zone determination indicator (not currently used)	
SIZES	MDATMUL	Number of words in WINFILE and BASFILE data blocks for multiple target components	
	MDATCX	Number of words in WINFILE and BASFILE data blocks for complex target components	
	MSPERMT	Maximum number of sites per multiple target	
	MTELMCM	Maximum number of components in a complex target	
	LINSTOR	Length in words of common /INPSTOR/	
	LNG1	Length in words of target data block on TINFILE	
	LNG2	Length in words of target data block on TGTFILE excluding fixed weapon assignment information words	
PRNTCTRL	IPRNTSW(7)	Print activation switch	
	ISTRT(3)	First target to activate print (switches 5-7)	
	ILST(3)	Last target to activate print (switches 5-7)	
	INWDS(3)	Number of words printed (switches 5-7)	

Table 9. (cont.)
(Sheet 5 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PRNTCTRL (cont.)	IPRNTNO	Print code requested
	NTGS	Target number currently being processed
	MPRQ	Number of print switches
	MSET(7)	Setting mode of switch (DEFAULT or INPUT)

PROGRAM PREPALOC

PURPOSE:	This program initializes many variables and controls the flow of processing throughout the program.
ENTRY POINTS:	PREPALOC
FORMAL PARAMETERS:	None
COMMON BLOCKS:	WPNDATA, WPNREG, WARHEAD, PLANTYPE, DPENREP, PAYLOAD, ASMTABLE, CORRCHAR, FILES, NOWFILE, MASTER, BRKPNT, EXCESS, NAVAL, ITP, IFTPRNT, BOUNDARY, HAPPEN, CHARTER, MYIDENT, NOPRINT, TWORD, IODUMMY, MACHINE, DEFAULT, OPTION, CHANGES, SUMS
SUBROUTINES CALLED:	STORAGE, INITAPE, GETVALU, ROUTING, WEAPPREP, TGTPREP, VALUMOD, MINMOD, MAXMOD, SETFILE, ABORT
CALLED BY:	Operating System; this is a main program

Method

This program first initializes the filehandler by calling INITAPE and sets the initial values of many variables used later.

The major part of processing in this program involves the reading and interpretation of the user-input function cards.

This routine will be default negative. That is, if a function is not specifically requested, the subroutine performing that option will not be exercised. Thus, the user need specify only those features that will actually be used. This will simplify run deck setup for the user. As each function command is read, control is passed to the subroutine or subroutines which perform that function.

Program PREPALOC is illustrated in figure 13.

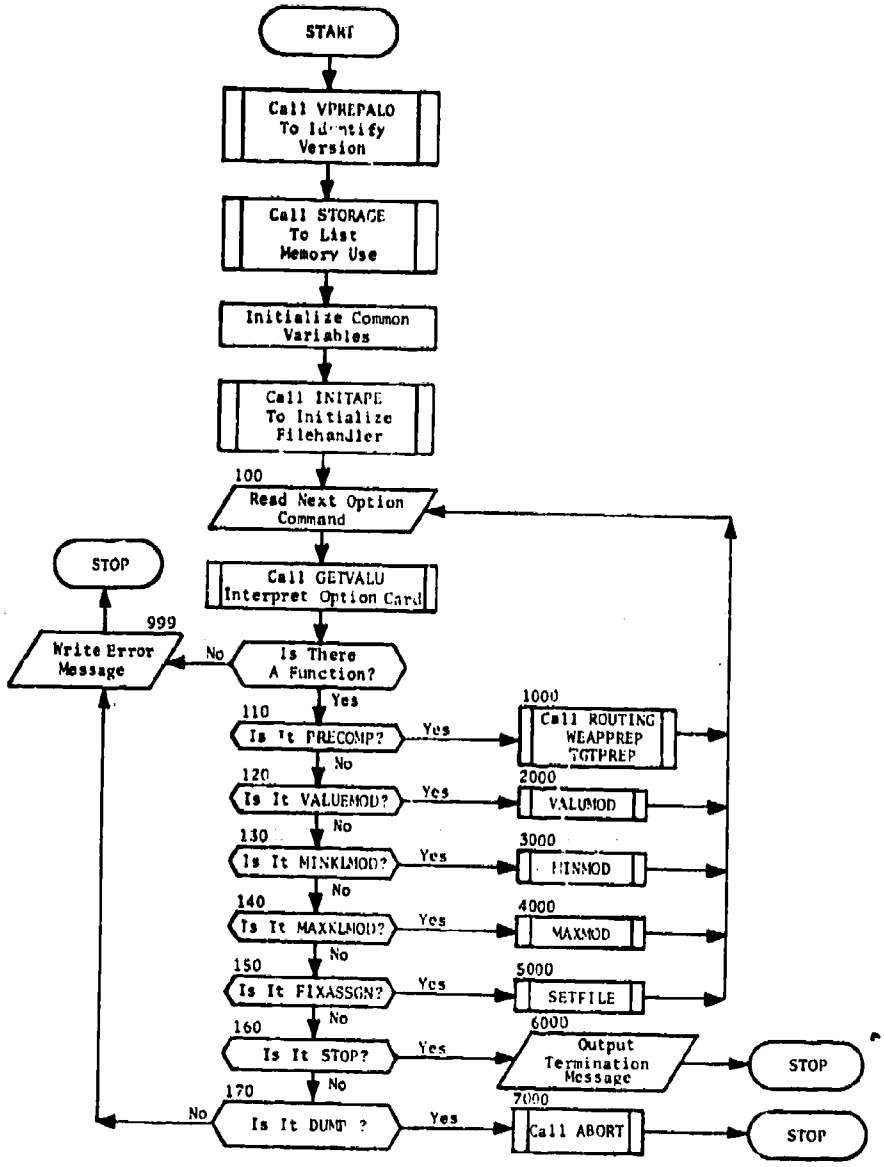


Fig. 13. Program PREPALOC

SUBROUTINE BASWRIT

PURPOSE:	This subroutine writes the BASFILE.
ENTRY POINTS:	BASWRIT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	OPTION, MASTER, BRKPNT, FILES, NOWFILE, CORRCHAR, CHARTER, ASMTABLE, PAYLOAD, DPENREF, PLANTYPE, WARHEAD, WPNREG, WPNDATA, EXCESS, NAVAL, CTRYCD, ITP, MYIDENT, MYLABEL, TWORD, INPSTOR, SIZES, CHANGES, SUMS, FILABEL
SUBROUTINES CALLED:	SETREAD, SETWRITE, WRARRAY, WRWORD, RDARRAY, RDWORD, TERMTAPE
CALLED BY:	TGTPREP

Method

This routine computes the maximum length of each non-scratch file used by the Plan Generation Subsystem and writes the BASFILE. If there were target value change requests, the values for complex target components are renormalized as they are output to the BASFILE. The format of this file is described previously in table 6.

The maximum file lengths are determined by estimating the total number of words required on each file for each missile, bomber, weapon, or target. For the BASFILE, the length of the POSTDATA file is retrieved via variable INLNGTH in common /FILABEL/ of the filehandler. The length of the BASFILE is the length of the POSTDATA file and 12,500 words.

If there were any value changes the POSTDATA file is examined to renormalize the values of the complex target components. The fraction of target value in each target class is also renormalized in this case.

Figure 14 illustrates Subroutine BASWRIT.

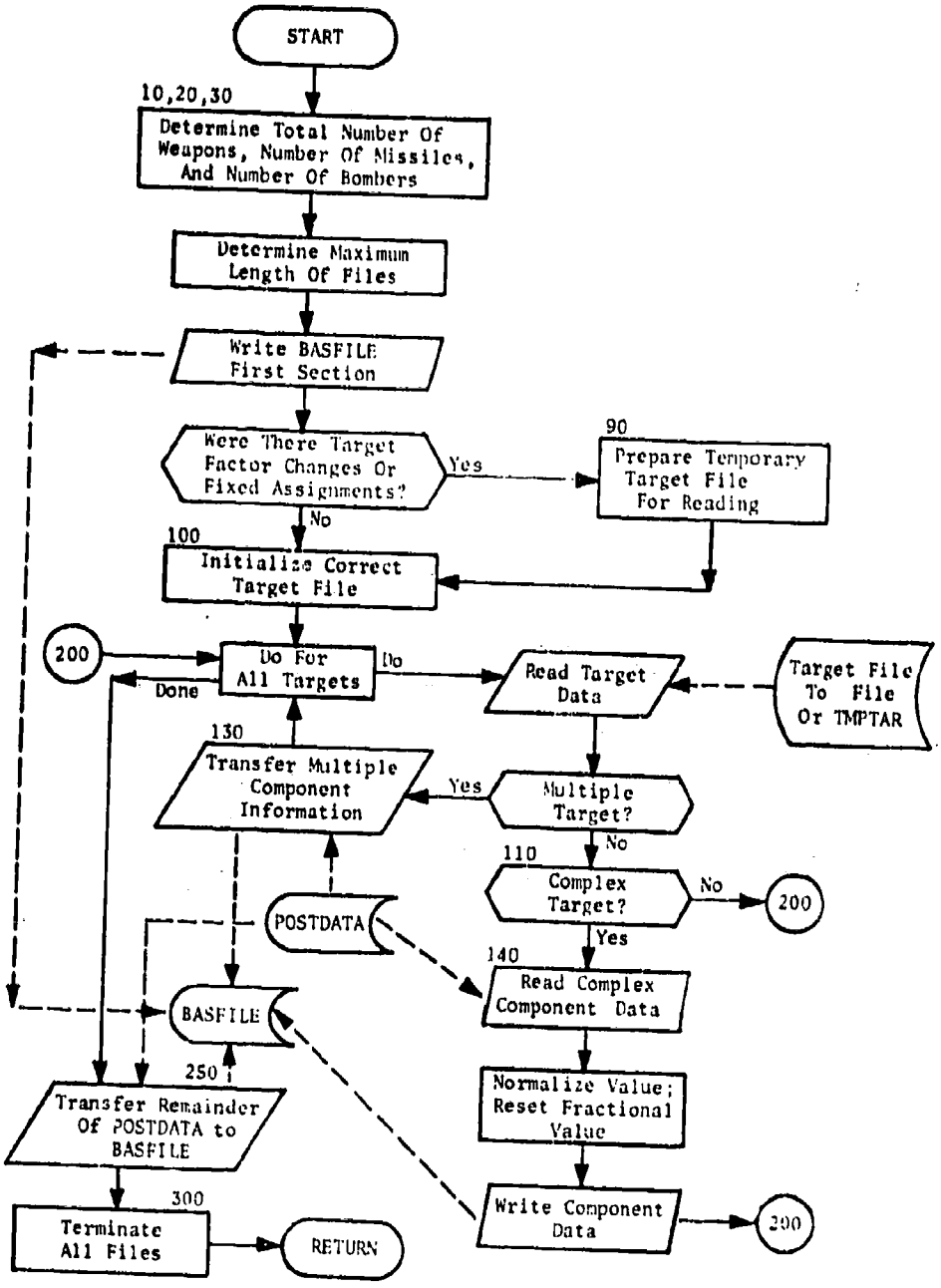


Fig. 14. Subroutine BASWRIT

SUBROUTINE CHKCHG

PURPOSE:	This routine checks the results of target planning factor change requests.
ENTRY POINTS:	CHKCHG
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CHANGES, SUMS, OPTION, FILES, NOWFILE, TWORD, ITP, IODUMMY, MACHINE, DEFAULT
SUBROUTINES CALLED:	RDWORD, WRWORD, TERMTAPE
CALLED BY:	TGTPREP

Method

If there were no target factor change requests, this routine returns control to TGTPREP after setting the target value renormalization factor to 1.0.

If there were target factor change requests, this routine first completes the temporary (unnormalized) POSTADATA file by copying words from POSTDATA to the temporary file until the end sentinel, XXXXXXXX, is encountered. The target value renormalization factor is then set to the ratio of the sum of the target values as input on the TINFILE (i.e., OLDSUM) to the sum of the changed target values (i.e., SUMNEW).

The array IFOUND in common block /CHANGES/ is then investigated. This array lists the number of times each target factor change request was exercised. If all requests were exercised at least once the routine prints the total number of factor changes made and returns control to TGTPREP.

If some requests were never exercised, CHKCHG lists the unused requests with an appropriate heading.

Subroutine CHKCHG is illustrated in figure 15.

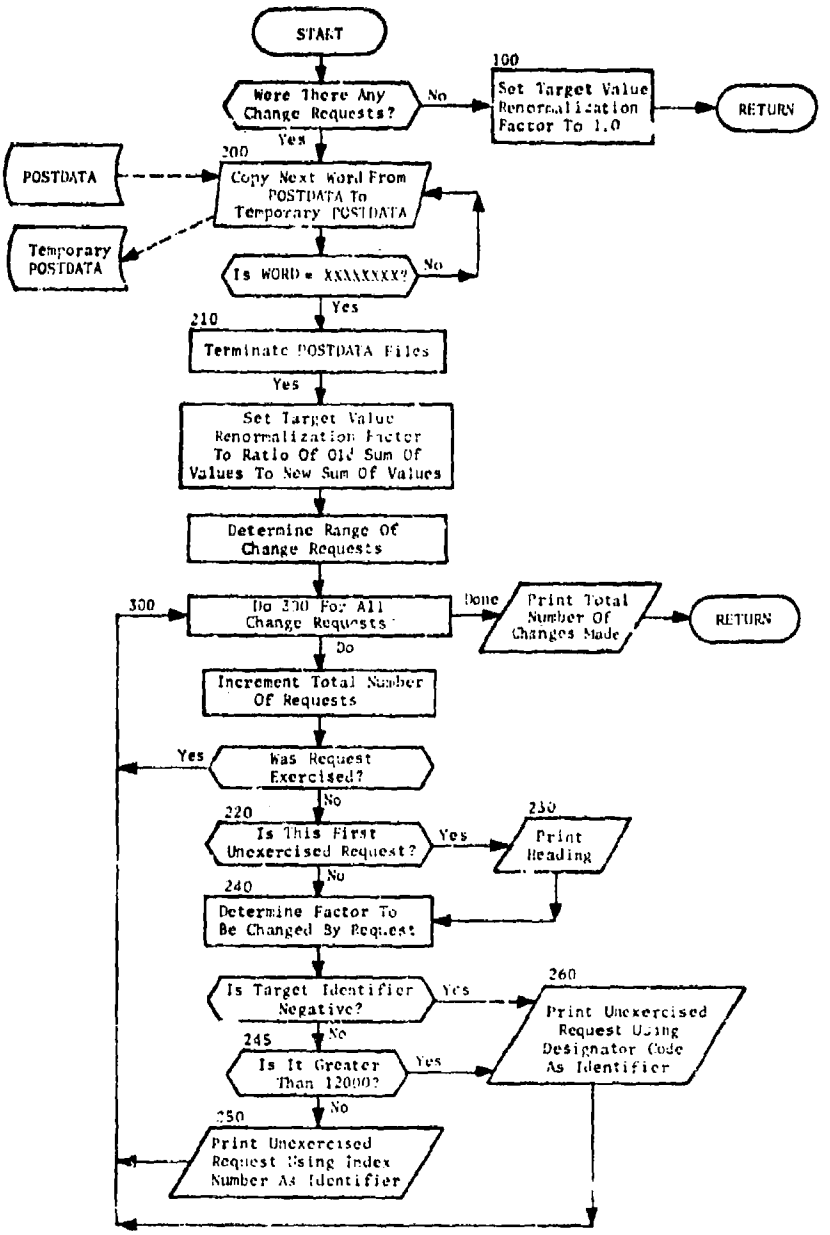


Fig. 15. Subroutine CHKCHG

SUBROUTINE FIXWEAP

PURPOSE:	This routine adds the fixed weapon assignment information to the TGTFILE.
ENTRY POINTS:	FIXWEAP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	OPTION, MASTER, CHANGES, SUMS, WPNDATA, IODUMMY, MACHINE, DEFAULT, NAVAL, EXCESS, FILES, NOWFILE, ITP, MYIDENT, MYLABEL , TWORD, PRNTCTRL, INPSTOR, SIZES
SUBROUTINES CALLED:	SETREAD, SETWRITE, RDWORD, WRWORD, RDARRAY, WRARRAY, TERMTAPE, PRINTDAT, NUMGET
CALLED BY:	TGTPREP

Method

There are three local arrays used in FIXWEAP:

NFIXES(200) - Number of weapons used for fixed assignments in each group
NWPG(200) - Total number of weapons in each group
IBOMB(200) - =0 if a missile group; =1 if a bomber

These arrays are used to perform the following error checks on the fixed assignment requests:

1. If an attempt is made to fix a bomber weapon on a target with more than 30 fix requests, the request will be ignored, since the only targets which can be allocated more than 30 weapons are those targets with terminal ballistic missile defenses which undergo a saturation missile attack. The allocation procedure will not allow a bomber to participate in such an attack.

2. If an attempt is made to fix more weapons than are present in a group, the excess requests are ignored.

Other error checks performed on the data by FIXWEAP are:

1. If an attempt is made to fix a weapon from a nonexistent group, the request is ignored.

2. If more than 1600 weapons are fixed to any one target, the excess requests are ignored.

3. If a fixed assignment request is never exercised because the input target identifier never appears, a warning message is issued.

The fixed assignment data require a large amount of storage (up to 20,000 words). Therefore, these data cannot be maintained in core memory. Since the fixed assignment processing is the last processing of the program, the fixed assignment data can be overlaid in storage areas used previously for other variables. The storage area containing the weapon group and type information as well as the area containing the target data change requests are available for use by the fixed assignment processor. The former data block, common /WPNDATA/, is 10,000 words long. This block is divided in subroutine FIXWEAP into two arrays, each 5,000 elements long. The first array (MYFIXD) contains the target identifier for a fixed assignment. The second array (MYDEXST) contains the indices to the first element in the INFIX array (see below) that contains an assignment for that identifier. Since the information is stored in order of occurrence, this array provides a pointer to the information of the INFIX array. The change request data block, common /CHANGES/, is also 10,000 words long. This block will become the INFIX array of length 10,000 elements. Each element will be the 8-character word containing the group number and timing information for a fixed assignment.* These data are first loaded into the NLOCK array of common / INPSTOR/ and then written on the TGTFILE after the target data block for the appropriate target.

The target identifiers may be target designator code (DESIG) or index number (INDEXNO), intermixed as the user wishes. Of course, if the TGTNUM subcommand is used with the FIXASSGN function, the identifiers for the fixed assignments must <u>all</u> be target numbers as assigned by program PLANSET.

Subroutine FIXWEAP reads the user-input fix requests from the FIXFILE (see table 7) prepared by subroutine SETFILE. Subroutine FIXWEAP then reads the temporary target file, TMPTAR, one target at a time. As each target is input the subroutine checks the designator code and index number (or target number) against the list of identifiers input as targets for fixed assignments (array MYFIXD). If a match is found the

---

* The format of the words in this array is the same as on the FIXFILE (see table 7). The first three characters are the group number; the last five characters are the specified arrival time.

array MYDEXST contains the index to the first location in the INFIX array where the fixed assignments for the target are stored. The fixed weapon information from the INFIX array is loaded into the NLOCK* array in common /INPSTOR/. The remainder of the MYFIXD array is searched to determine if there are any further matches of target identifiers. If so, the necessary data from the INFIX array is added to the NLOCK array. As each target identifier in MYFIXD is used, it is set to zero as a flag that its information has been used to fix weapons.

Subroutine FIXWEAP multiplies the target value parameters by the target value renormalization factor. The first 94-word block for the target is output on the TGTFILE (see table 5). Then the number of weapons fixed to the target is output. If these are fixed assignments the assignment information (one word per fix) is output on the TGTFILE.

Subroutine FIXWEAP is illustrated in figure 16.

* The format of the words in this array is the same as on the FIXFILE (see table 7). The first three characters are the group number; the last five characters are the specified arrival time.

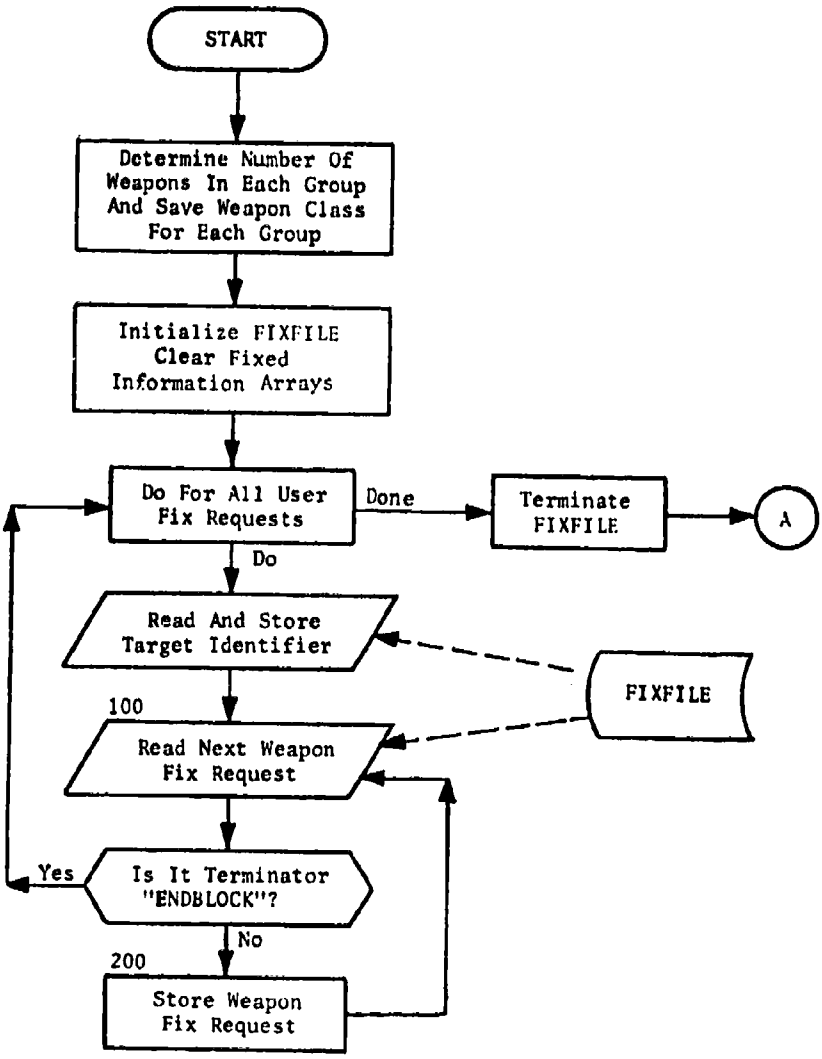


Fig. 16. Subroutine FIXWEAP
(Sheet 1 of 4

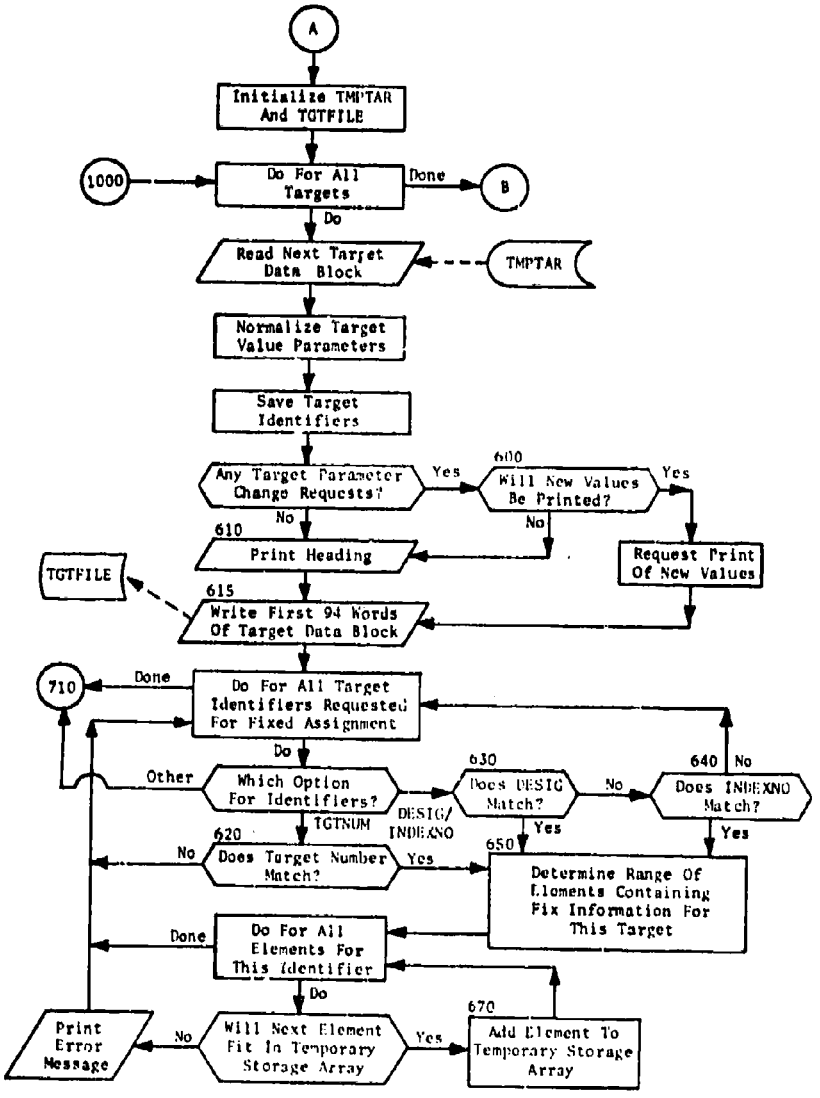


Fig. 16. (cont.)
(Sheet 2 of 4)

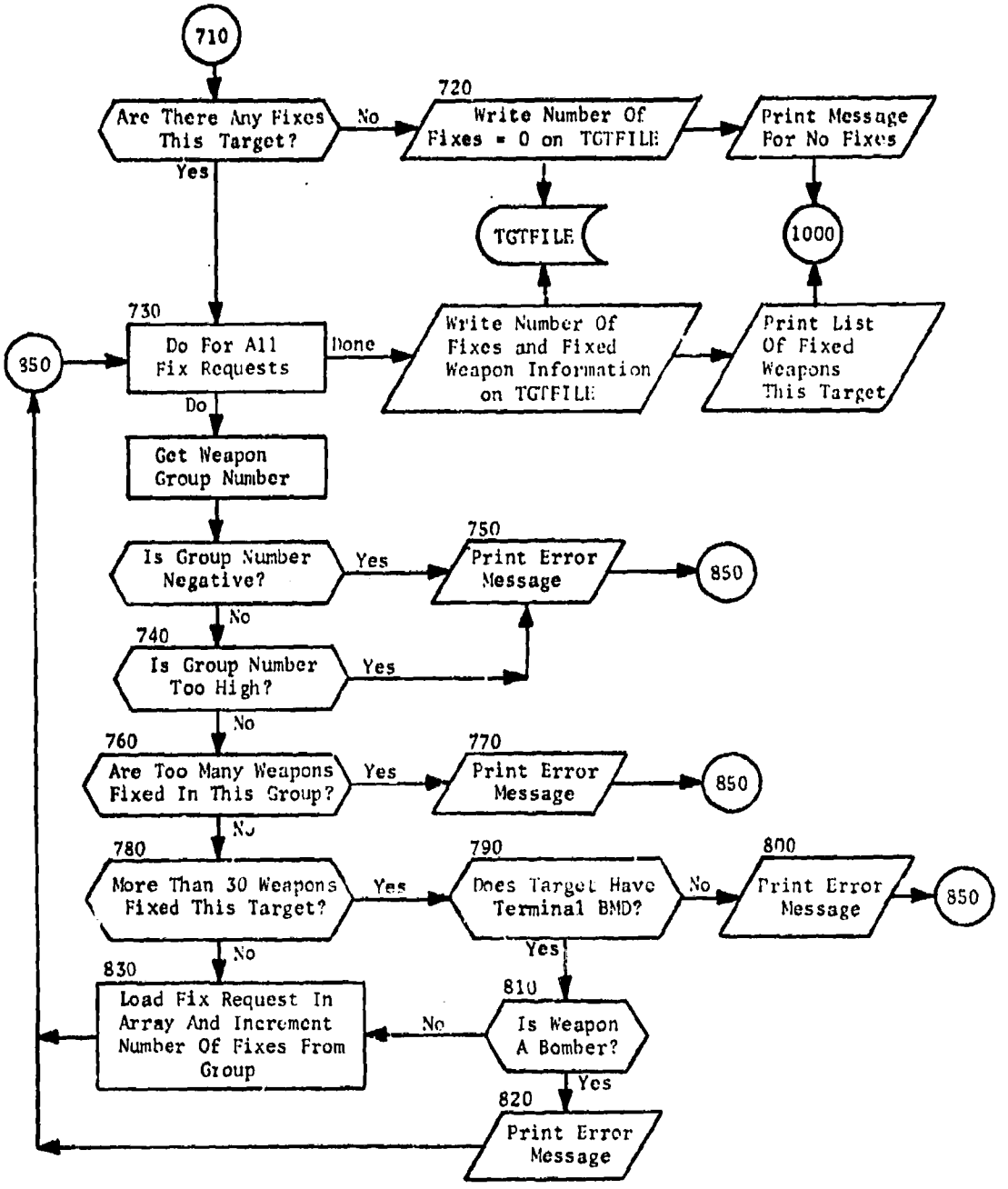


Fig. 16. (cont.)
(Sheet 3 of 4)

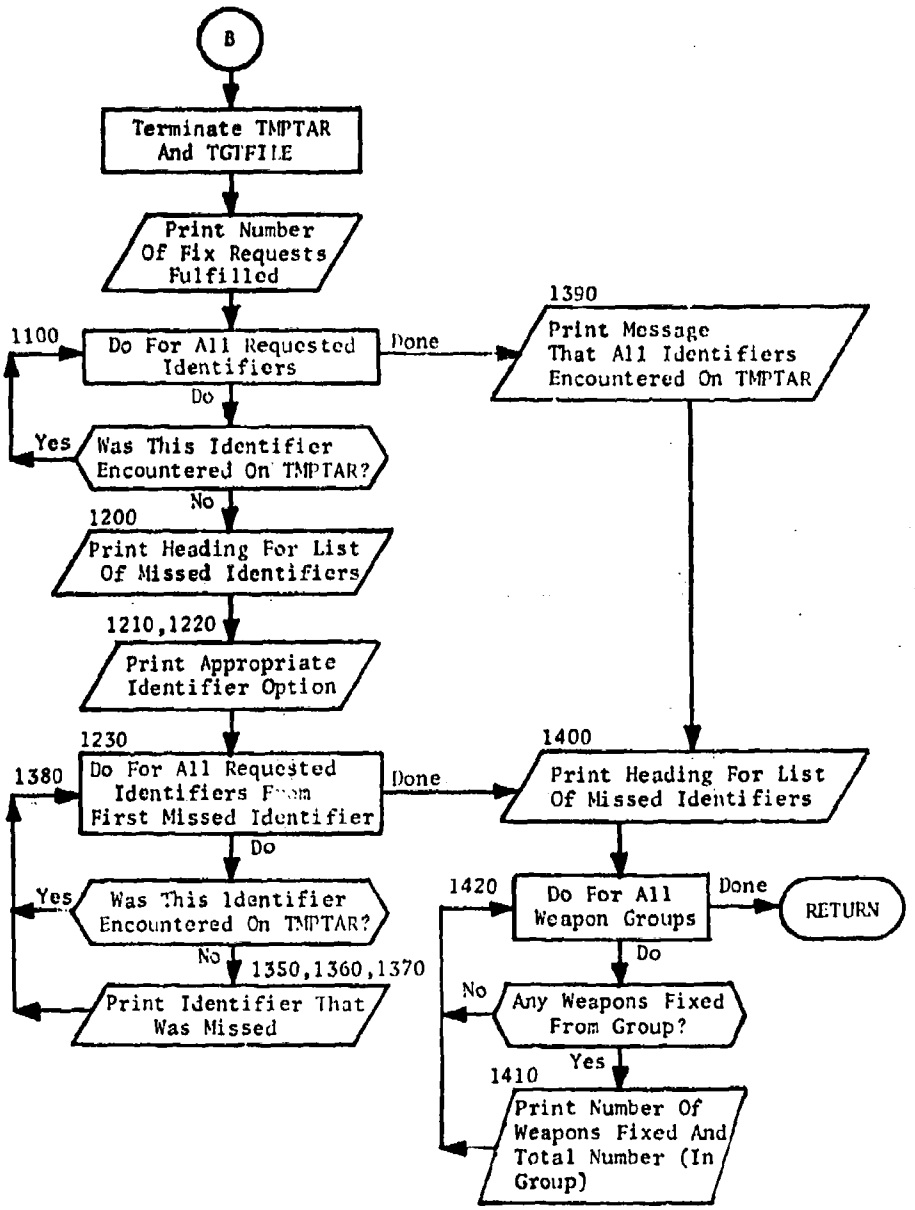


Fig. 16. (cont.)
(Sheet 4 of 4)

SUBROUTINE MAKECHG

PURPOSE:	This routine effects the target planning factor change requests input by the user.
ENTRY POINTS:	MAKECHG
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CHANGES, SUMS, OPTION, FILES, NOWFILE, ITP, INPSTOR, SIZES
SUBROUTINES CALLED:	RDARRAY, WRARRAY
CALLED BY:	TGTPREP

Method

If none of the target planning factor modification options were exercised, this routine merely accumulates the sums of the old and modified target values (which are the same). No further target processing is required.

If there were some planning factor change requests, this routine must determine if the current target, or any of its components, meet the specifications of any request. There are several pointers and counters which are used by this routine to determine the flow of processing. They are as follows:

IVAL: Set to 1 if the main target value is changed
IMIN: Set to 1 if the main target MINKILL is changed
IMAX: Set to 1 if the main target MAXKILL is changed
ICPX: Number of components in complex target (also IIC)
IMUL: Number of components in multiple target (also IIM)
IIC: Set to 1 if any factor is changed on a complex target component
IOPT: Location pointer for local change check "subroutine"

The basic change request checking is done by a local change check "subroutine" which begins at statement 1000. This section of the code determines if the current target or component matches the specifications of the range of change requests delineated by the indexes ISTART and IEND. The variable IOPT is used in statements 1100 and 1200+2 to determine the correct part of the subroutine to return to when the changes have been checked. If the current target or component does not meet the specifications of the requests, the return through

statement 1100 is used. If the target or component is to be changed, the return through statement 1200 is used. In the accompanying flow diagram, the label "CHANGE" on a return from the local subroutine (labelled 1000) denotes the return through 1100. The label "NO CHANGE" on a return branch denotes the return through 1200.

The variables IVAL, IMINK, and IMAXK are used so that there is not more than one change to the main target for each factor. Once a factor has been modified on the main target or any component of a multiple target that factor is never checked again. If a factor is changed on a component of a complex target, that factor is checked on all the remaining components.

If a factor is changed on the main target of a complex, the factors for each component are changed appropriately. For example if the value of the complex is doubled, the value of each component is doubled. If the value of MINKILL for the complex is halved, the value of MINKILL is halved for each component. This procedure is followed to allow the user to change one factor for the entire complex and other factors by component. If a factor is changed for a complex component individually (i.e., not through the main target) the variable IHC is set so that the value of the complex target factors can be recomputed. The method used by MAKECHG in determining value, MINKILL, and MAXKILL for a complex target is identical to that used in subroutine CALCOMP of program PLANSET.

If there are change requests, subroutine MAKECHG transfers data on complex and multiple target components from the POSTDATA file to the temporary POSTDATA file, TMPOST. The factors on the latter file reflect the changes made by MAKECHG in fulfilling the user change requests.

Subroutine MAKECHG is illustrated in figure 17.

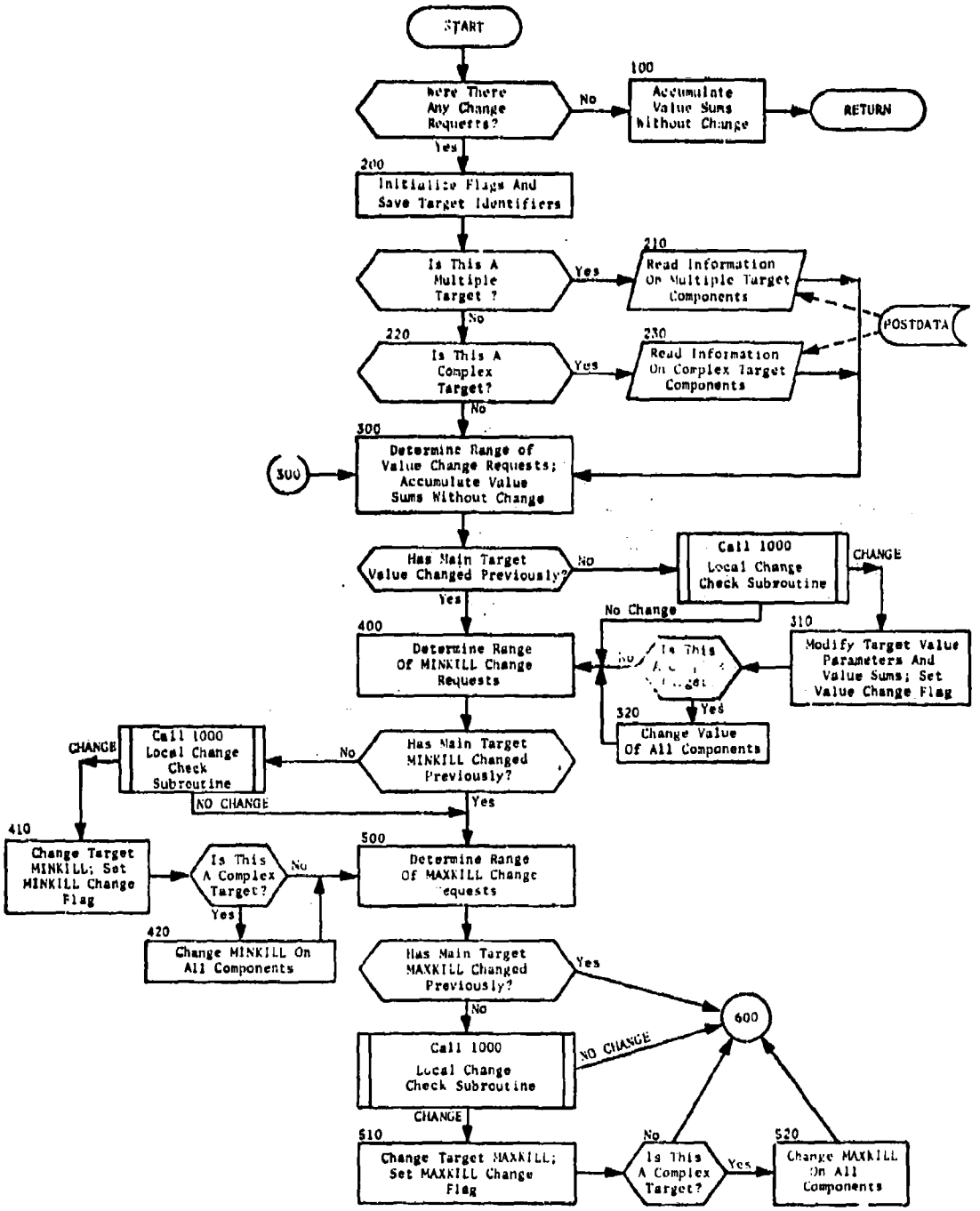


Fig. 17. Subroutine MAKECHG
Part I: Processing on Main Target and Multiple Targets

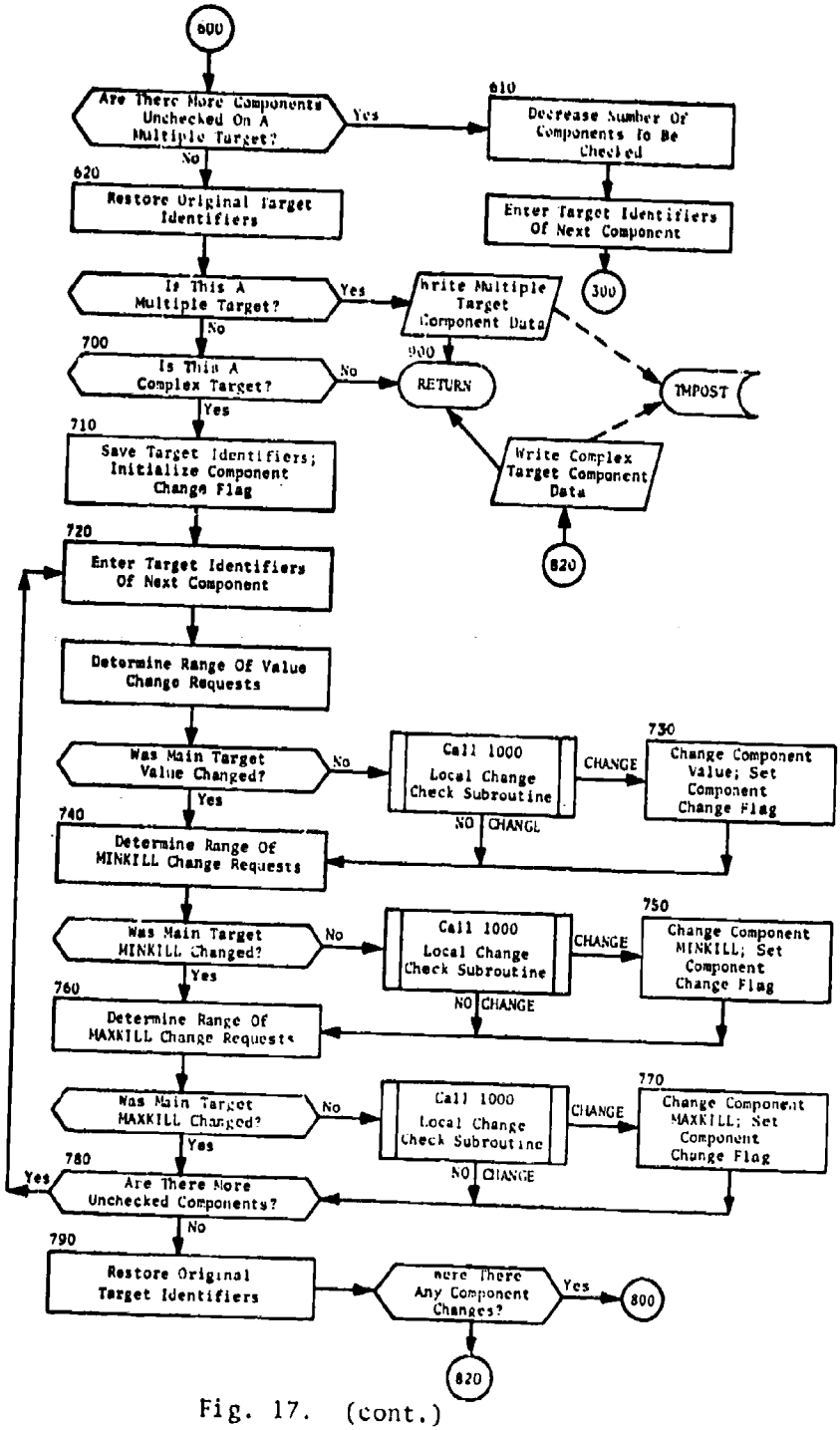


Fig. 17. (cont.)
Part II: Processing of
Complex Target Components

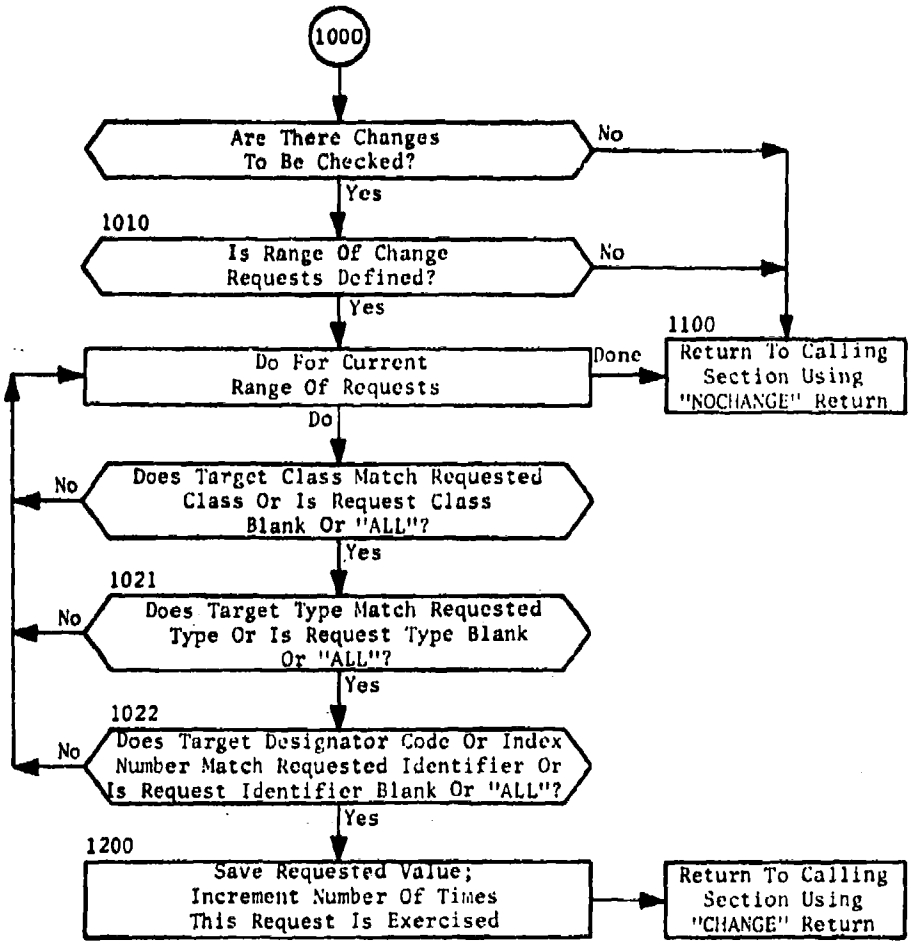


Fig. 17. (cont.)
Part III: Local Change Check Subroutine

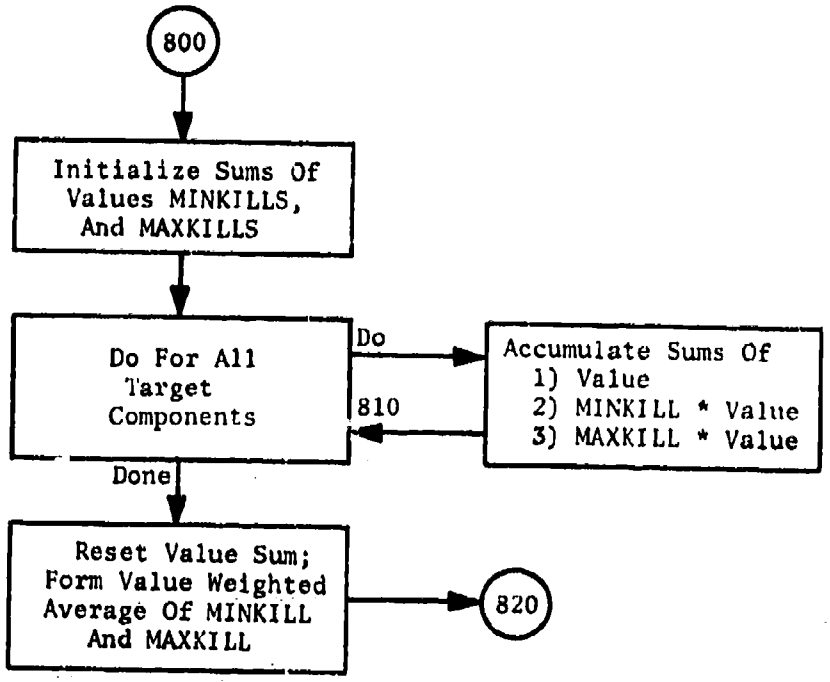


Fig. 17. (cont.)
Part IV: Final Computation of Complex Target Factors

SUBROUTINE NORMALZ

PURPOSE:	This routine renormalizes the target value parameters on the TGTFILE if there were no fixed weapon assignment requests.
ENTRY POINTS:	NORMALZ
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, OPTION, FILES, NOWFILE, ITP, MYIDENT, TWORD, MYLABEL, PRNTCTRL, INPSTOR, SIZES, IODUMMY, MACHINE, DEFAULT
SUBROUTINES CALLED:	SETREAD, SETWRITE, RDARRAY, WRARRAY, TERMTAPE, PRINTDAT
CALLED BY:	TGTPREP

Method

This routine reads the target data blocks for each target from the IMPTAR file. The target value parameters are multiplied by the target value renormalization factor and the modified target data block is written on the TGTFILE. This routine is called only if there were target planning factor change requests but no fixed weapon assignment requests.

Subroutine NORMALZ is illustrated in figure 18.

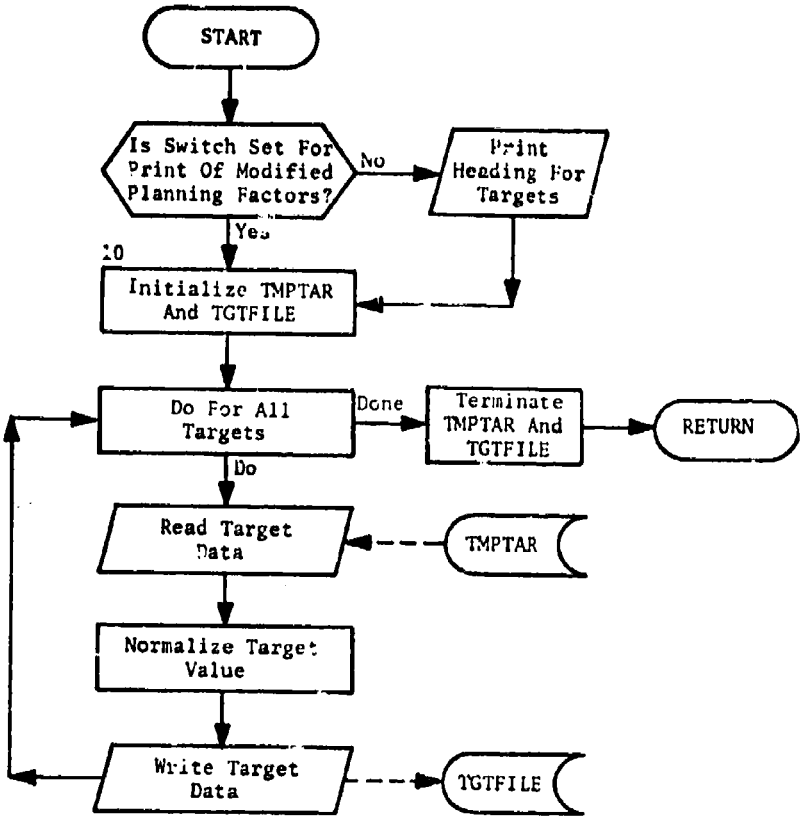


Fig. 18. Subroutine NORMALZ

SUBROUTINE PRINTDAT

PURPOSE:	This routine produces the optional prints for the PRECOMP option of program PREPALOC.
ENTRY POINTS:	PRINTDAT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, BOUNDARY, WPNDATA, FILES, CORRCHAR, PREALOC, PRNTCTRL, DPENREF, WPNREG, PAYLOAD, ASMTABLE, WARHEAD, HAPPEN, CHARTER, TEMPO, BOUND, ITP, IFTPRNT, RECOVR, INPSTOR, NOWFILE, IODUMMY MACHINE, DEFAULT, SIZES
SUBROUTINES CALLED:	None
CALLED BY:	ROUTING, WEAPPREP, TGTPREP, NORMALZ, FIXWEAP

Method

The following print options may be selected:

Option 1 - Listing of routing data
2 - File dump of input routing data (from WINFILE)
3 - Listing of distance from weapon groups to each corridor
4 - File dump of input weapon data (from WINFILE)
5 - Target listing
6 - File dump of input target data (from TINFILE)
7 - Modified target factors

Prior to calling PRINTDAT the print option is stored in common /PRNTCTRL/. If prints have been requested for the stored option, control is transferred to a set of print statements which prints the data from common storage.

Subroutine PRINTDAT is illustrated in figure 19.

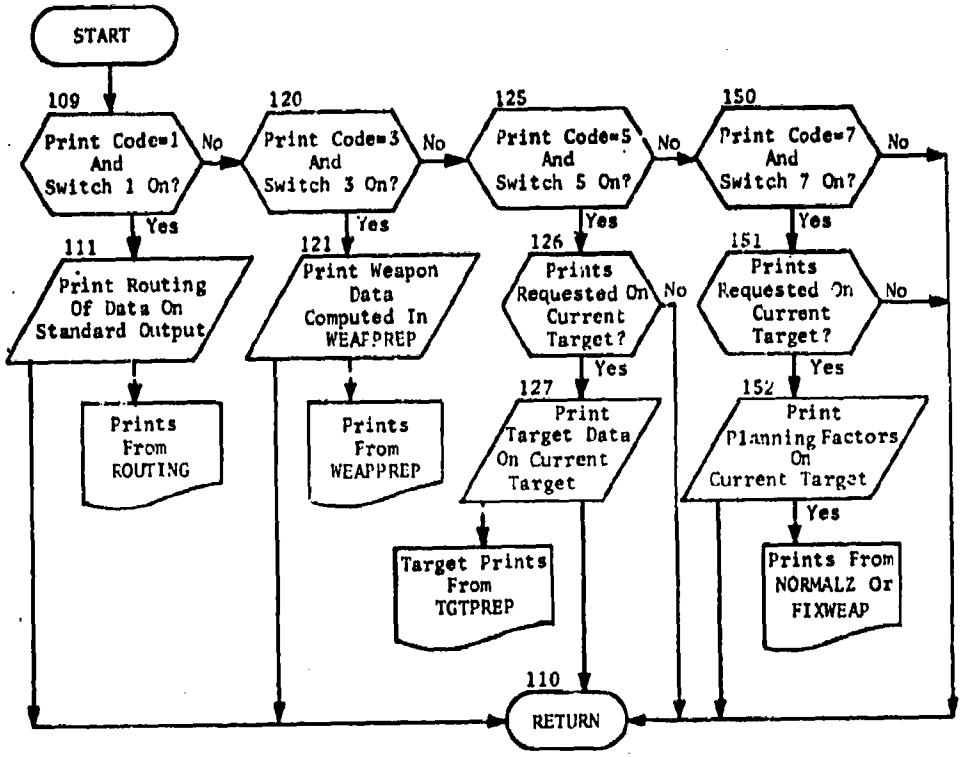


Fig. 19. Subroutine PRINTDAT

SUBROUTINE RDPRCMP

PURPOSE:	This routine reads and interprets the user input parameter cards for the data precomputation function.
ENTRY POINTS:	RDPRCMP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	IODUMMY, MACHINE, DEFAULT, PLANTYPE, EXCESS, PRNTCTRL, INPSTOR, SIZES
SUBROUTINES CALLED:	GETVALU, ITLE, NUMGET
CALLED BY:	ROUTING

Method

This subroutine reads and interprets the input parameters, including the print requests. The input is in free field format which is broken down into attribute (or parameter) value format by utility subroutine GETVALU. This routine produces a list of parameter name-value pairs as they are input on the user input parameter cards.

The arrays in common /IODUMMY/ are used for transferring information between RDPRCMP and GETVALU. The input cards are read into array INPUT. The parameter name-value pairs are returned by GETVALU in arrays NAMES and INVALU. Two words are used for the value of each parameter. (The value is returned in BCD code.) The arrays INDEX1, INDEX2, and INDEX3 are not used by RDPRCMP, but are required for compatibility with GETVALU. The variable NVARS is set by GETVALU to be the number of parameter-value pairs on the input card. The variable MORE is set to 1 unless the termination character $ appears on the card. This character signals the end of the input parameters.

Common block /DEFAULT/ is used to store information on the user input parameters. MYNAME is the parameter name; MYFORM is a hollerith code representing the format in which the parameter will be printed; MYTYPE is the mode of the parameter; MYVAL (equivalent to FVAL) is the value of the parameter. The mode of each parameter is originally set to "DEFAULT". If the parameter is changed on input, the mode is changed to "INPUT". Table 10 displays the default values for the parameters.

There are two types of print requests, modified and unmodified. The modified requests (codes 5, 6, 7) can be activated for specific targets. The array NAMPRNT lists the names of the parameters which will modify the request. The parameter PRINT selects a print request. The parameter NOPRINT removes a print request. The modes of the print requests may be DEFAULT, INPUT, or REMOVED. (The last is generated by a NOPRINT parameter.)

RDPRCMP reads the user input cards until a blank card or a $ sign encountered. Each parameter on each card is checked against the lists of parameter names contained in MYNAME and NAMPRNT. As a parameter is encountered, the value of MYVAL (FVAL) is set appropriately. Only the parameter INITSTRK and the print request parameters are fixed point variables. The value for these is determined by function NUMGET. The other parameter values are determined by a FORTRAN DECODE statement (number 152).

Subroutine RDPRCMP is illustrated in figure 20.

Table 10. Default Parameter Settings

I	MYNAME(I)	MYFORM(I)	MYVAL(I) (FVAL(I))
1	INITSTRK	I8	1
2	CORMSL	F8.4	0.0
3	CORBOMB	F8.4	0.0
4	PEXBOMB	F8.4	0.0
5	EXNBOMB	F8.4	3.0
6	PEXMIRV	F8.4	0.1
7	EXBMIRV	F8.4	2.0
8	PEXMISS	F8.4	0.0
9	EXNMISS	F8.4	0.0

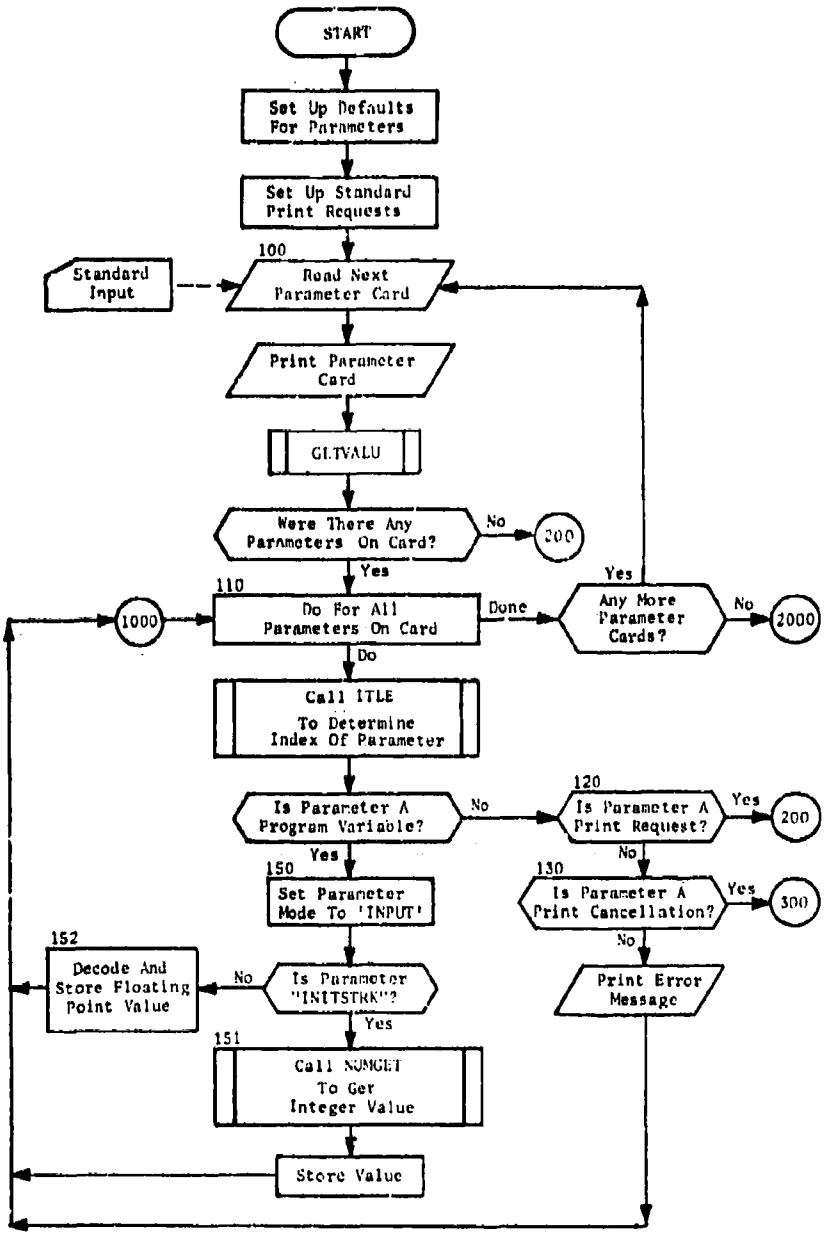


Fig. 20. Subroutine RDPRCMP
Part I: Main Parameter Processing

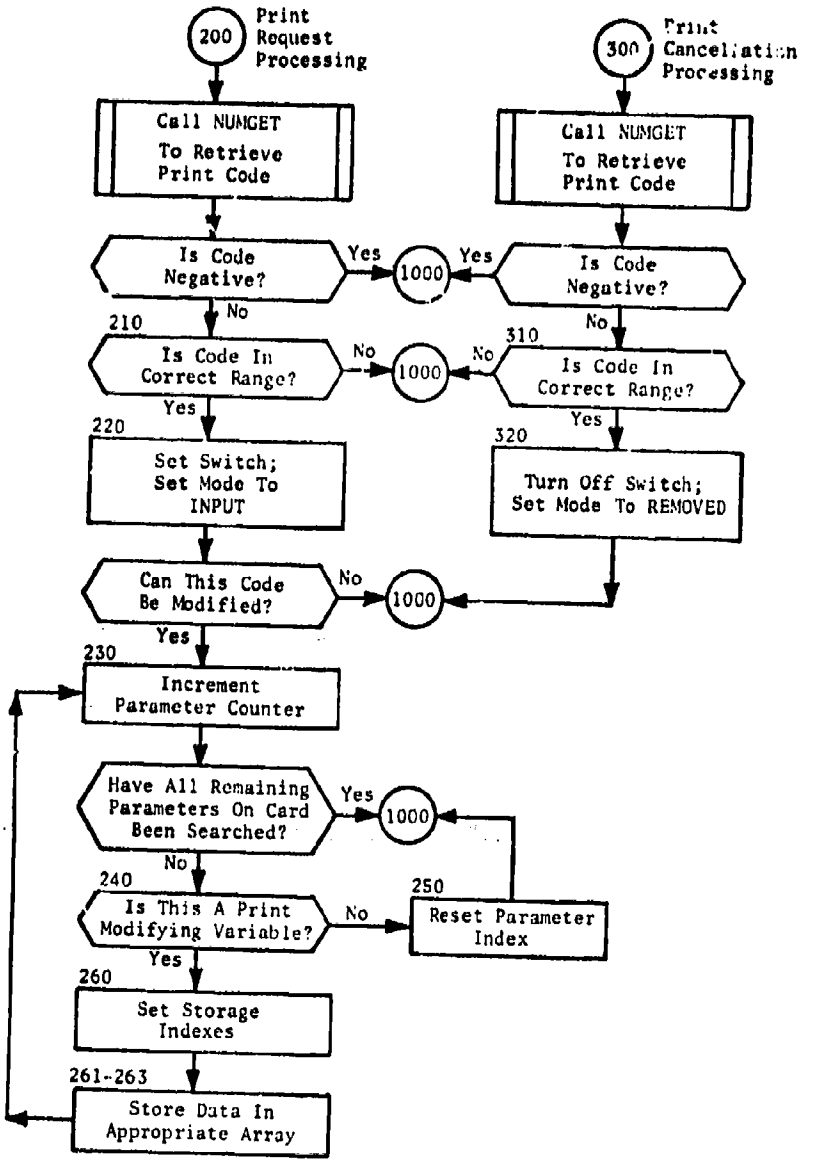


Fig. 20. (cont.)
Part II: Print Control Parameter Processing

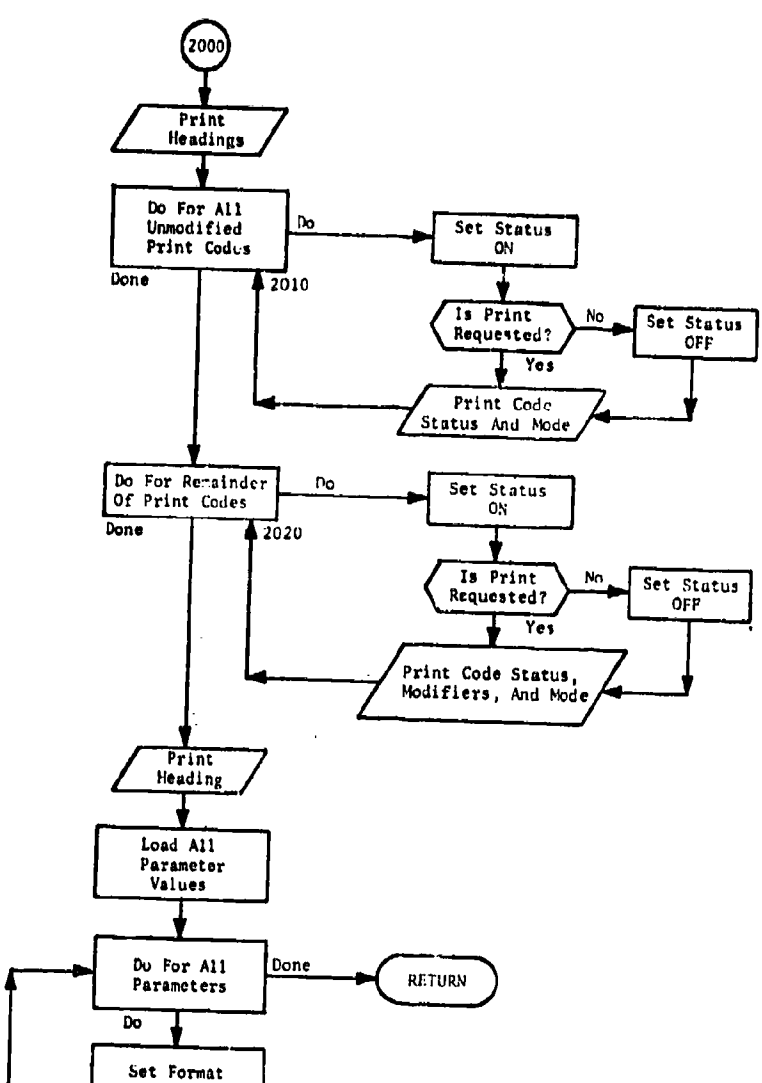


Fig. 20. (cont.)
Part III: Output Processing

SUBROUTINE ROUTING

PURPOSE:	This routine stores the routing data from WINFILE into common block arrays and computes various distances and attrition factors associated with each penetration and depenetration corridor.
ENTRY POINTS:	ROUTING
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, BRKPNT, FILES, PLANTYPE, DPENREF, BOUNDARY, CORRCHAR, HAPPEN, CHARTER, ITP, IFTPRNT, PREALOC, INPSTOR, PRNTCTRL, MYIDENT, TEMPO, BOUND, DUMCORR, RECOVR, NOWFILE, IODUMMY, MACHINE, DEFAULT, SIZES
SUBROUTINES CALLED:	SETREAD, RDARRAY, DISTF, PRINTDAT, ORDER, RDPRCMP
CALLED BY:	PREPALOC

Method

The subroutine begins by calling RDPRCMP to read the user-input parameter cards. The header, penetration and depenetration corridors, routing points, recovery points, refuel points, and zone boundaries are read in from WINFILE and stored in indexed arrays. For each penetration corridor the following data are computed: length of corridor, number of defended areas in corridor, length of each defended area, and the attrition in each of the areas. A corridor is processed through all of its routing points (legs) accumulating the lengths of the legs and locating the corridor entry point. Also, a count is made of the number of defended areas in the corridor, where a defended area is defined as one or more consecutive legs with attrition. The length of each area and its attrition are computed and all of the data computed above are stored in common /CORRCHAR/.

As each corridor is being processed, the coordinates and length of each leg are stored. Also a cell, JAPTYPE, is set to designate whether the leg is the beginning or termination of a defended area. If JAPTYPE is 1, 2, or 3, a defended area begins with this leg. The value of JAPTYPE is the number of the precorridor leg for this leg. If JAPTYPE is 4, 5, or 6, this leg ends a defended area. The value JAPTYPE -3 gives the precorridor leg number. These data are then stored in common /HAPPEN/. Since the data in /HAPPEN/ are not indexed by corridor, for each corridor a pointer to the

/HAPPEN/ arrays and the number of entries for the corridor are stored in common /CHARTER/.

The depenetration corridors are processed in the same manner as the penetration corridors except that they do not have associated attrition. At this stage the recovery bases are linked to the depenetration points. Up to four bases may be associated with each point. For later processors, the bases are ordered according to increasing distance from the end of the depenetration corridor. The length of each corridor leg from entry to recovery is computed, and data representing the leg are inserted in commons /HAPPEN/ and /CHARTER/, as described above. When all the routing data have been processed, the print routine is called; and if print option one was selected, the routing data are listed.

Subroutine ROUTING is illustrated in figure 21.

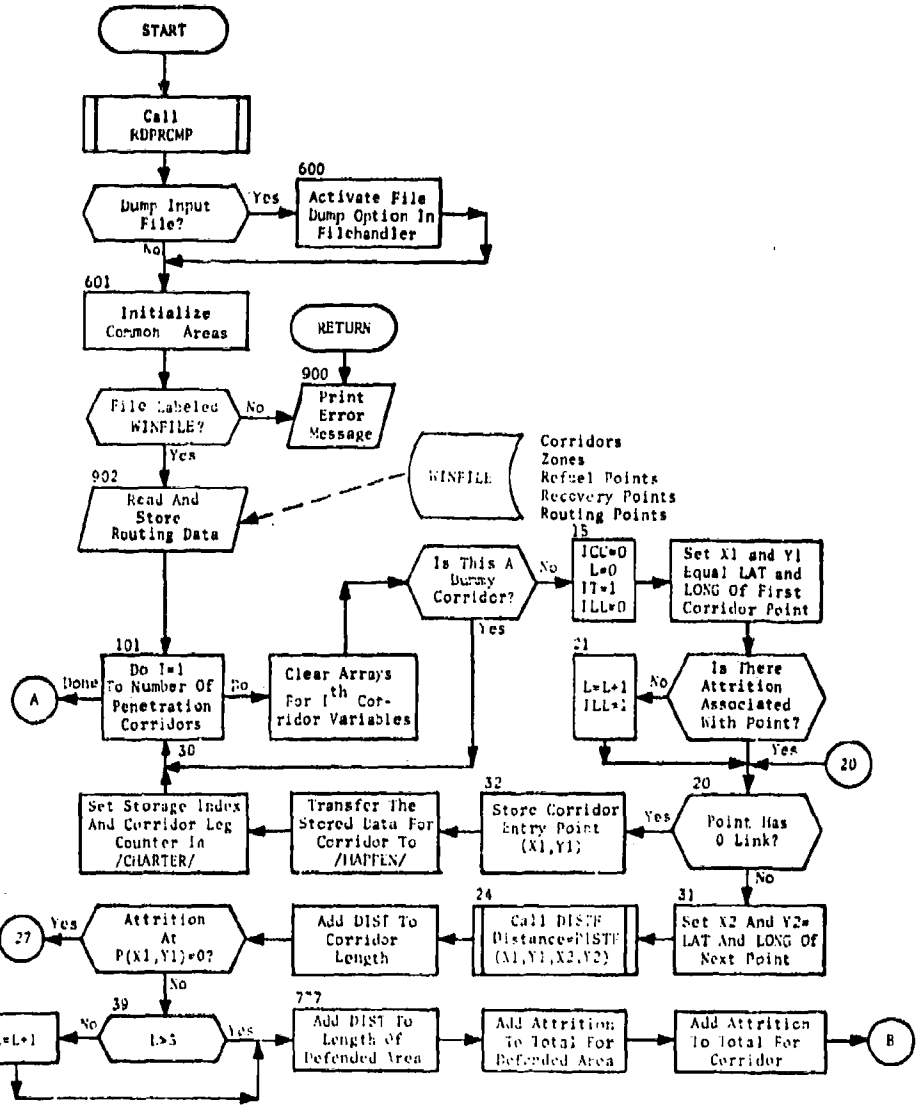


Fig. 21. Subroutine ROUTING
(Sheet 1 of 2)

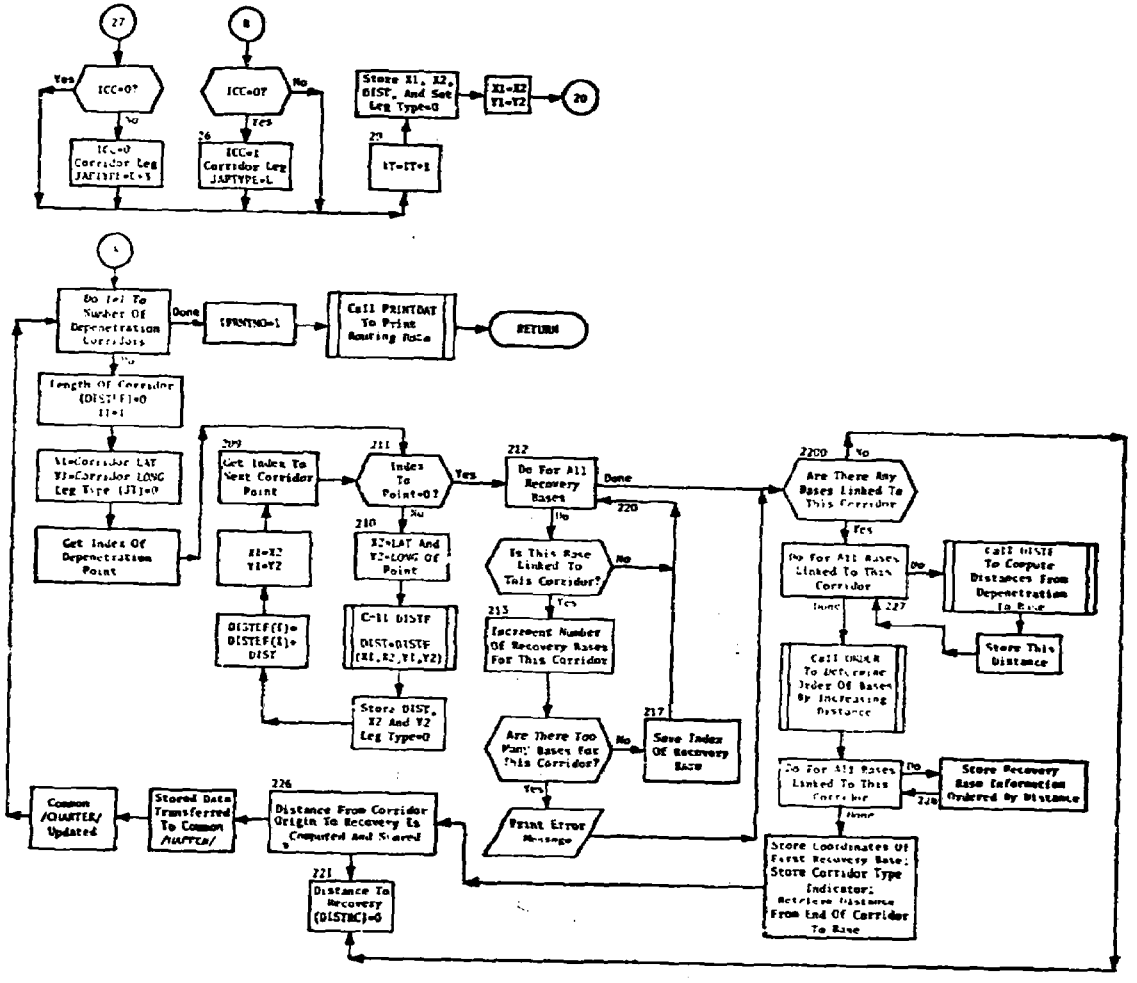


Fig. 21. (cont.)
(Sheet 2 of 2)

SUBROUTINE SETFILE

PURPOSE:	This routine reads the user fixed weapon assignment requests, performs preliminary error checking on this information, and sets up a temporary disk file for storage of this data.
ENTRY POINTS:	SETFILE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	IODUMMY, MACHINE, DEFAULT, FILES, NOWFILE, ITP, MYIDENT, TWORD, WPNDATA, CHANGES, SUMS, OPTION, MASTER
SUBROUTINES CALLED:	SETREAD, SETWRITE, RDARRAY, WRWORD, TERMTAPE, NUMGET, GETVALU
CALLED BY:	PREPALOC

Method

This routine reads the user-input fixed weapon assignment requests, performs preliminary error checking on these requests, and stores the data in modified form on the FIXFILE. This file will be used by subroutine FIXWEAP to place the fixed weapon assignments on the TGTFILE for use by program ALOC.

The potentially large number of requests for fixed weapon assignment precludes the possibility of storing all information for these assignments in core storage at all times. Common blocks /WPNDATA/ and /CHANGES/ are used by subroutine FIXWEAP to process the assignment requests after these blocks have been used by subroutines VALUMOD, WEAPPREP, TGTPREP, and MAKECHG. The temporary scratch file FIXFILE is utilized for temporary storage of the weapon assignment data. Table 7 displays the format of this file.

One word of information is required for each weapon assigned. This word contains the group number of the weapon to be fixed in the first three characters. The last five characters can be used to specify the arrival time (in minutes) of a missile weapon. (If these characters are used for a bomber weapon, they are ignored.) If the last five characters are left blank, the launch timing for the weapon will be calculated in the normal manner. If an arrival time is specified, it takes precedence over all

other launch timing considerations. As an example, a word of 127 1.0 specifies fixing a missile from group 127 to arrive at the target 1.0 minutes after H-hour.

The fixed assignment information is read organized by target. That is, the fixed assignment requests for each target are grouped together. The requests can be viewed as a number of sets of request cards, one set for each target. The first card of each set contains the target identifiers for the set in the first ten columns. These columns are ignored in all later cards in the set.) This identifier may be target designator code (left justified) or index number (justification not applicable). (In some cases it may be target number as assigned by PLANSET. This option, the TGTNUM option, is described later.) Every card of each set contains the fixed assignment requests in columns 11-18, 21-28, 31-38, 41-48, 51-58, 61-68. Each field represents a request to fix one weapon on the target (or one weapon on each component in the case of multiple targets). The eight columns in each field are divided into three columns for the group number and five columns for the arrival time as described previously. A blank field is ignored and later fields are scanned for fixed assignment requests. If the set of requests for a target contains more than one card, every card but the last should contain a nonblank character in either or both columns 71 and 72. The last card in each set must contain blanks in these columns.

Each set of requests is output to the FIXFILE in the format described in table 7. Between the input and the output of this information, some simple error checking occurs as described later.

There are several user options available in this routine concerning the mode of input. The user may request that the target identifiers be considered as target numbers (assigned by program PLANSET). In this case the fixed assignment request indicator, IFIXTGT, in common block /OPTION/, is set to -100. (In the default mode, target identifiers are target designator code or index number. IFIXTGT is set to +100.) The other user options are described briefly as follows:

TAPE: The input requests are on a magnetic tape in filehandler format, one word for each field.

DISK: The input requests are on a disk file in filehandler format, one word for each field.

BCDTAPE: The input requests are on a magnetic tape in BCD card image format, one 80-character record for each card.

The default input option is, of course, the card reader (or standard input device) with one 80-character BCD record per card.

Only two files are active during this subroutine's processing, FIXFILE and the input file (if any). If the user has requested the input file on unit six, which is normally used for the FIXFILE, the FIXFILE is output on unit three. If the input file is a disk file, the Filehandler uses the buffer area for unit three for this file.

Arrays MYVAL and MYTYPE in common /DEFAULT/ are used for temporary storage of information on the user-input options of this subroutine. The processing in the 100 series of statements is concerned solely with reading and interpreting the option commands. The 300 and 400 series initializes the files and prints the option information. If no options were input by the user, the first card read is a fixed assignment request. In this case, the DO-loop beginning at statement 450 will allow the processing of this request. The checks between statements 100 and 110 provide for terminating the reading of the option cards by reading the first request card.

During processing of the assignment request sets, three indicators control processing. NEWCRD designates the beginning of a new set (i.e., a new target). If NEWCRD is 1, the next card read begins a new target. If the next card is merely a continuation of the current set, NEWCRD is equal to 0. The logical variables IGNORET and IGNOREF control processing if there are errors present in the input data. Both variables are initially set false. If an error results in termination of processing of all further fixed assignment requests, IGNOREF is set true. If an error results in termination of all requests after processing the requests for the current target, IGNORET is set true.

During processing, the requests are checked for group numbers within range, and the limits on total number of weapons and targets used for fixed assignment. Any request for fixing a weapon from a group outside the range from one to 200 is ignored. If there are more than 5,000 sets (i.e., targets) requested for fixed assignment, the excess requests are ignored (by setting IGNORET true). If more than 10,000 weapons are requested for fixing, the excess requests are ignored by setting IGNOREF true.

At the end of SETFILE, the routine prints the number of weapon requests for each group. This print, however, shows the number of requests in each group. If a target is a multiple target, a request for a fixed assignment to the target results in fixing a weapon on each component of the multiple target. Thus, one request may result in the assignment of up to five weapons. The actual number of weapons fixed (rather than requests) is printed by subroutine FIXWEAP.

Subroutine SETFILE is illustrated in figure 22.

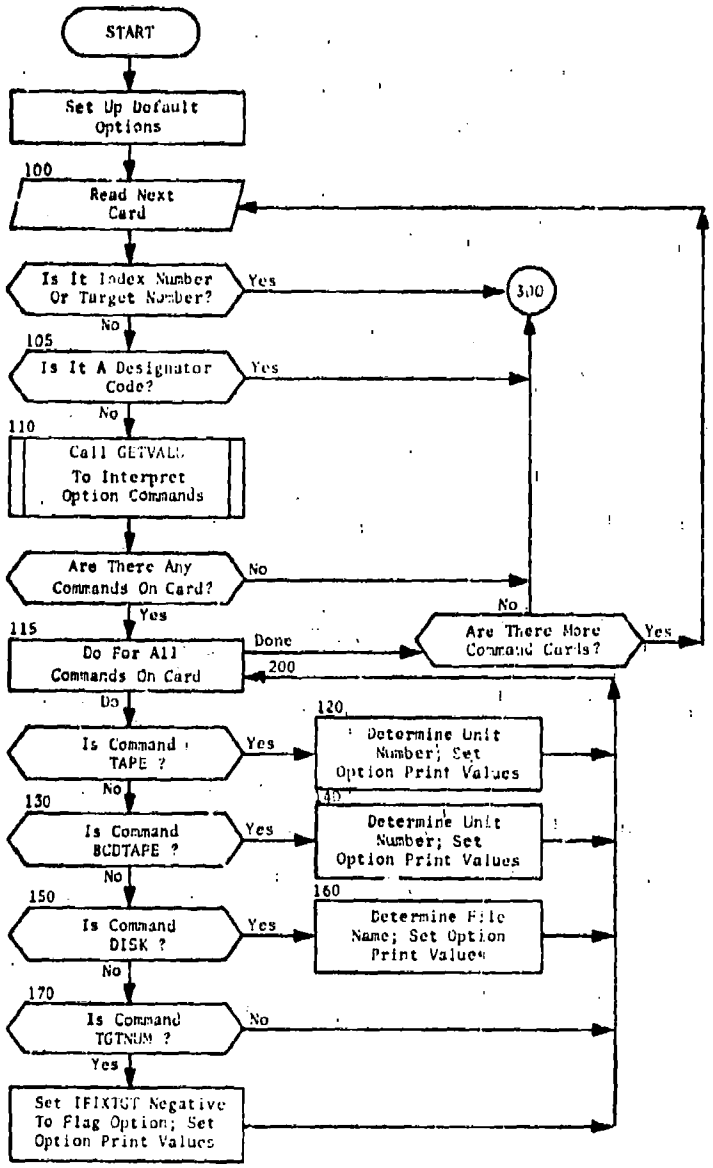


Fig. 22. Subroutine SETFILE
Part 1: Option Command Processing

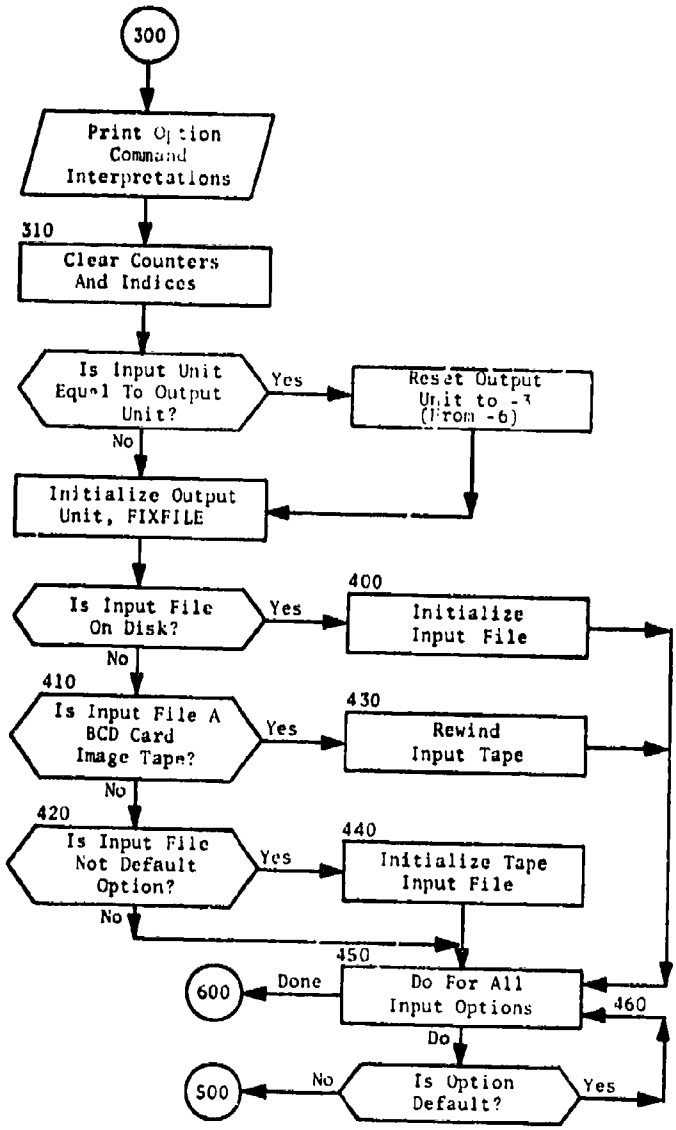


Fig. 22. (cont.)
Part II: File Initialization

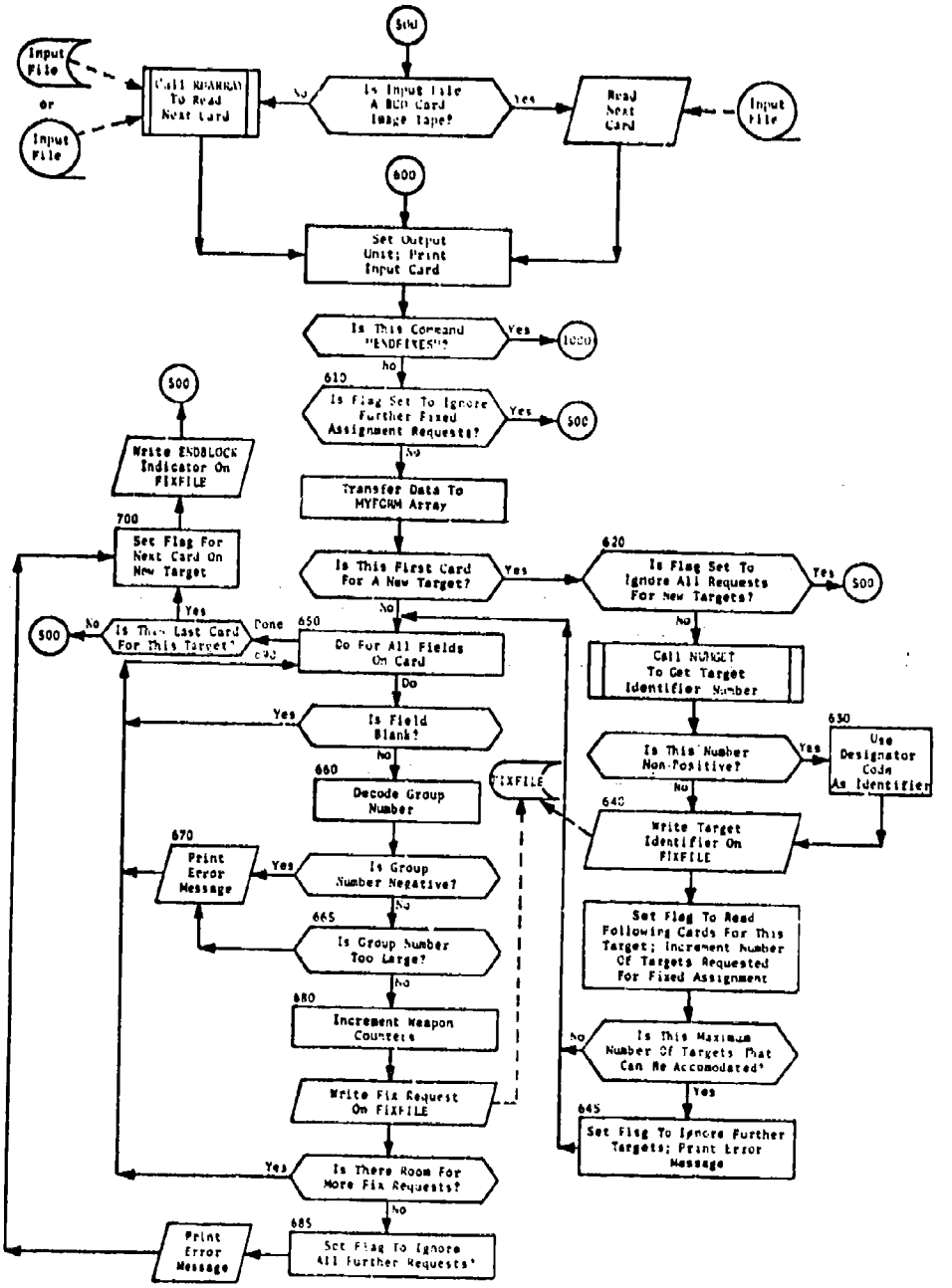


Fig. 22. (cont.)
Part III: Processing of Fixed Assignment Requests

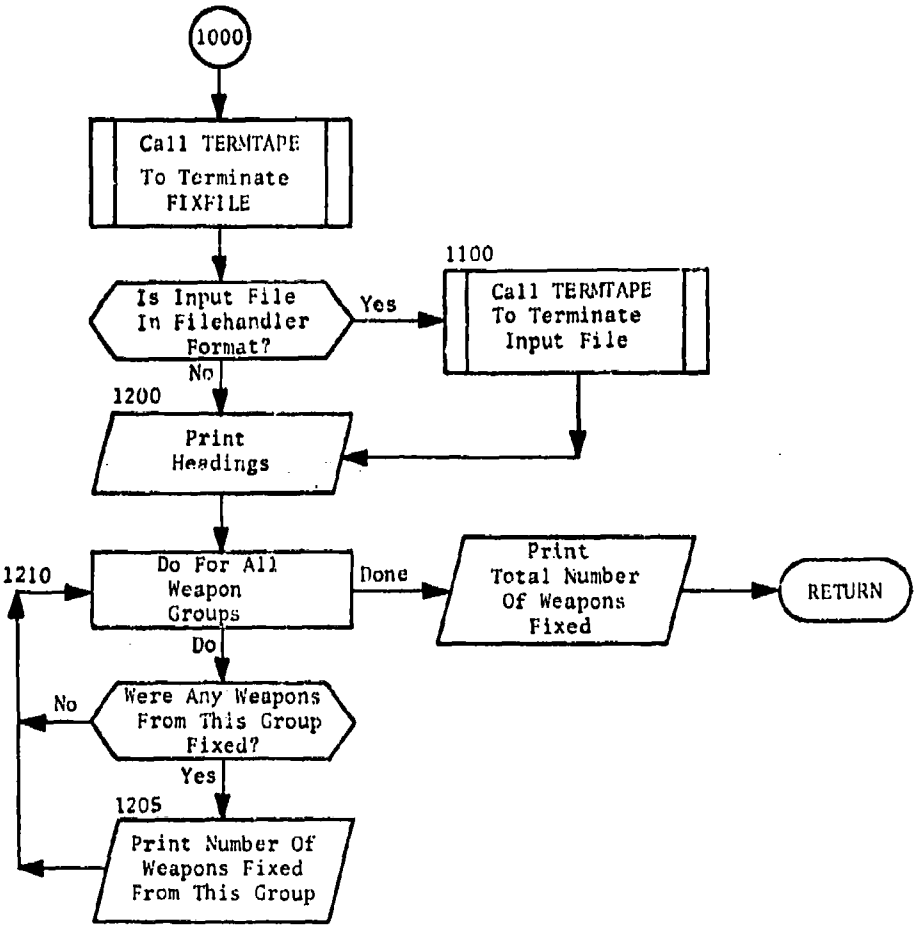


Fig. 22. (cont.)
Part IV: Termination Processing

SUBROUTINE TGTPREP

PURPOSE:	This routine prepares the target file for input to program ALOC and calls the routines which modify target factors, fix-assign weapons, and write the BASFILE.
ENTRY POINTS:	TGTPREP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, FILES, DPENREF, CORRCHAR, ITP, IFTPRNT, PREALOC, INPSTOR, PRNTCTRL, MYIDENT, DUMCORR, NOWFILE, SIZES, IODUMMY, MACHINE, DEFAULT, OPTION, CHANGES, SUMS, MYLABEL, BRKPNT, CTRYCD
SUBROUTINES CALLED:	SETWRITE, SETREAD, RDARRAY, DISTF, PRINTDAT, WRARRAY, TERMTAPE, MAKECHG, ITLE, CHKCHG, BASWRIT, FIXWEAP, NORMALZ
CALLED BY:	PREPALOC

Method

Each target is read into the buffer array BLOCK in /INPSTOR/, and the input data are rearranged and placed in a 95-word output record. The distance from the target through each depenetration corridor to recovery is computed. The index to the corridor having minimum distance, the distance from target to corridor, and the distance from corridor entry to recovery are inserted in the output record. Also the distance from each penetration corridor to the target and the corridor attrition are computed and placed in the output record. Subroutine MAKECHG is called to modify target value, MINKILL or MAXKILL. The fraction of target value in each class is accumulated and a list of all country location codes for the targets is constructed.

If there were fixed weapon requests or requests to change target planning factors, the data is output on a temporary target file, TMPTAR. Otherwise the data is output on the TGTFILE for program ALOC. Prior to output of each target record, the print subroutine is called and the target is listed if option five is activated for this target.

At the end of the processing of the individual targets, TGTPREP calls subroutine CHKCHG to check the results of the target factor change requests. Subroutine BASWRIT is called to write the BASFILE. If there were fixed assignment requests, subroutine FIXWEAP is called to add this information to the TGTFILE. If there were target factor change requests but no fixed assignment requests, subroutine NORMALZ is called to renormalize (to a sum of 1,000) the target values on the TGTFILE. (If there are fixed assignments as well as factor changes, subroutine FIXWEAP performs this renormalization.) Finally the fraction of value in each target class is printed and control returns to the main program.

In computing the depenetration point to be associated with each target, a modified depenetration distance calculation is used. The depenetration point associated with the target is the point which minimizes the sum:

$$(2*DISTD) + DISTR$$

where DISTD is the distance from target to the depenetration point and DISTR is the distance from depenetration point to recovery.

Subroutine TGTPREP is illustrated in figure 23.

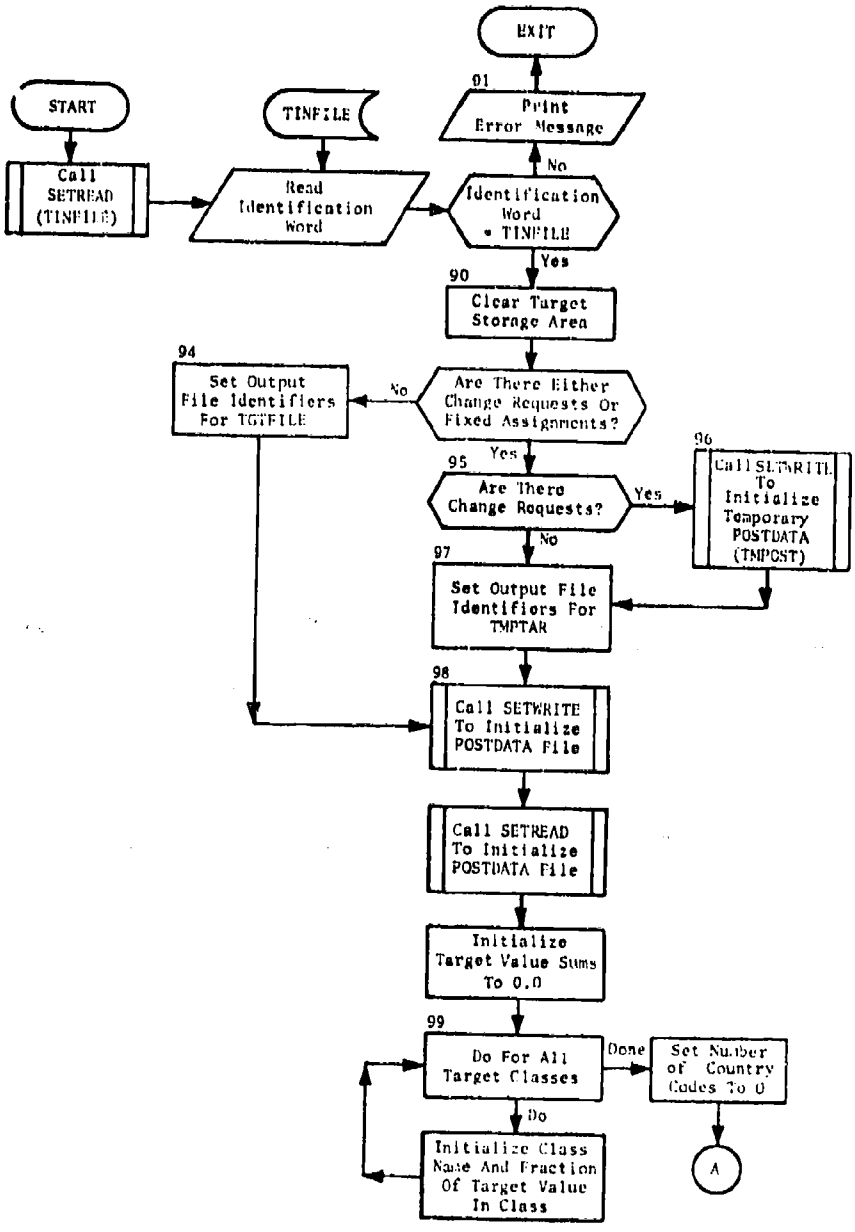


Fig. 23. Subroutine TGTPREP
Part I: Initialization

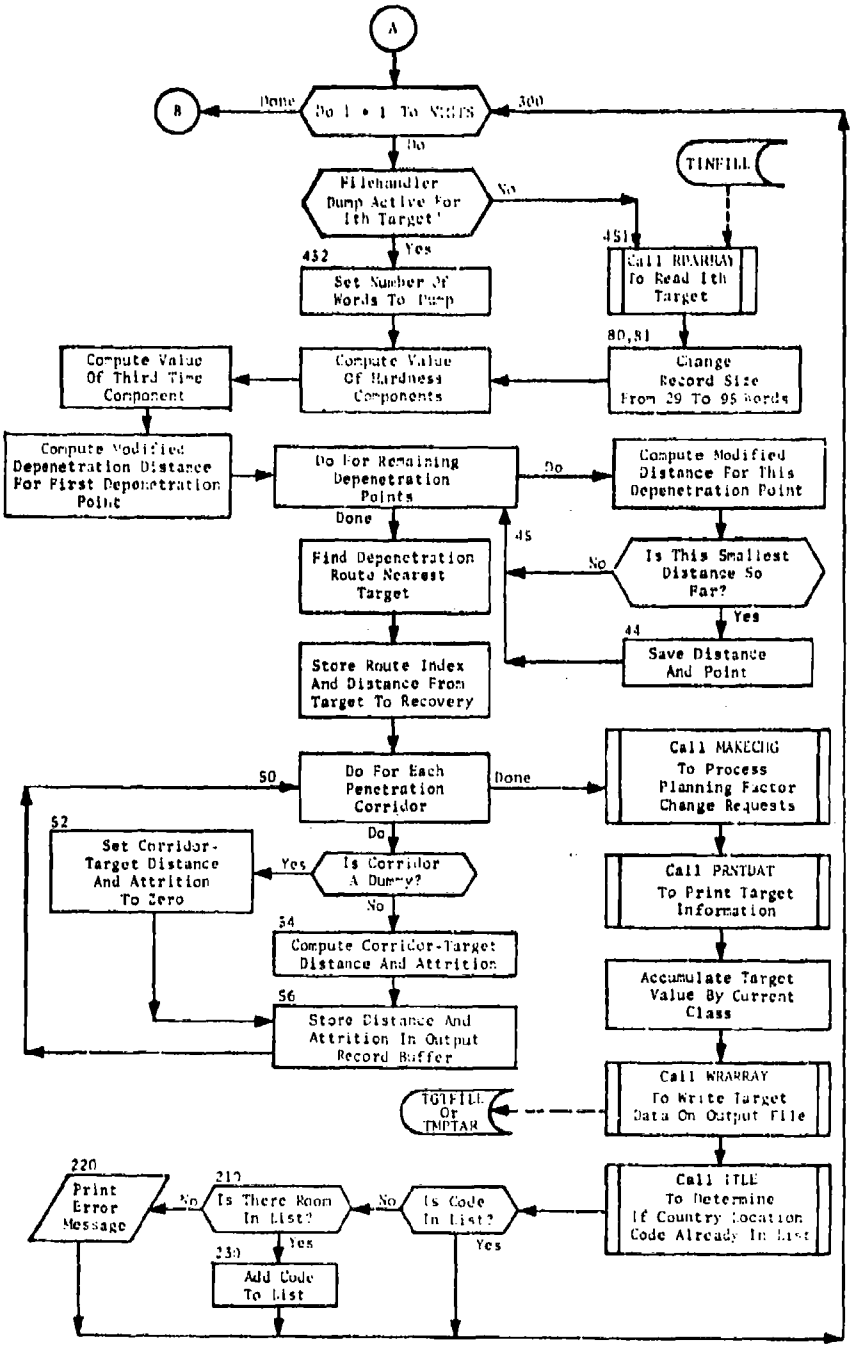


Fig. 23. (cont.)
Part II: Target Processing

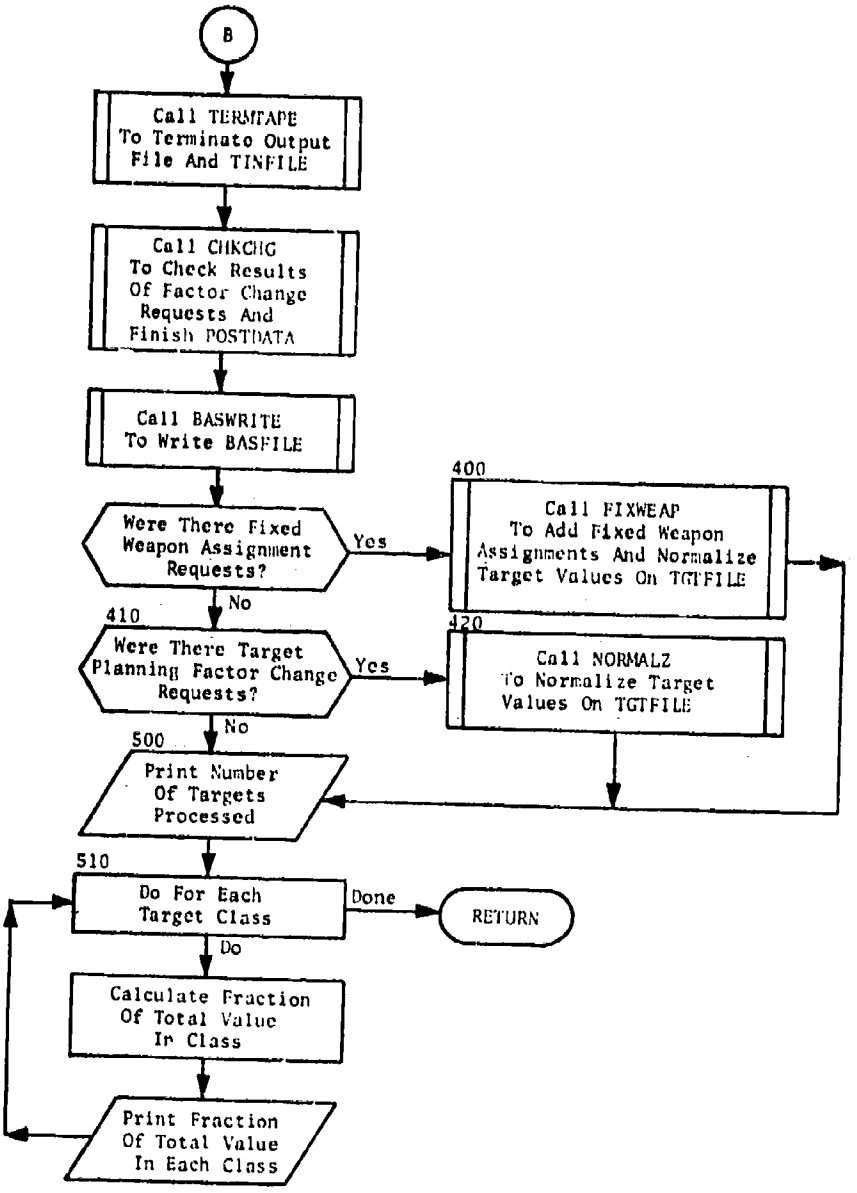


Fig. 23. (cont.)
Part III: Final Processing

SUBROUTINE VALUMOD

PURPOSE:	This routine reads the target planning factor change request cards for changes to target value, MINKILL, or MAXKILL.
ENTRY POINTS:	VALUMOD, MINMOD, MAXMOD
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CHANGES, SUMS, IODUMMY, MACHINE, DEFAULT, OPTION
SUBROUTINES CALLED:	NUMGET
CALLED BY:	PREPALOC

Method

This routine uses three entry points, one each for value change requests, MINKILL change requests, and MAXKILL change requests. All requests are stored in arrays in common block /CHANGES/. ICLASWAN is the class name for which the request is to be effected; ITYPEWAN is the type name for which the request is to be effected; IDENTWAN is the target identifier for the change; and VALUNEW is the value of the new factor (target value, MINKILL, or MAXKILL).

Array IFOUND is a counter which accumulates the number of targets on which each change request is used. Since requests for changes in all three factors are stored in the same arrays, the arrays in common block /OPTION/ are used to specify the starting and ending index for requests for each factor. The arrays are INDVAL for value change requests, INDMIN for MINKILL change requests, and INDMAX for MAXKILL change requests. In each array, the first element is the first location in the arrays in common block /CHANGES/ that is used for the change requests for that factor. The second element is the last location used for that factor. For example, if option MINKLMOD precedes option VALUEMOD which precedes option MAXKLMOD, and there are 11 MINKILL change requests, seven MAXKILL change requests, and eight value change requests, the arrays in common block /OPTION/ are as follows:

```
INDVAL(1) = 12
INDVAL(2) = 19
INMIN(1)  = 1
```

INDMIN(2) = 11
INDMAX(1) = 20
INDMAX(2) = 26

The change requests may be input to cover a set of targets. The user specifies the class name, type name, and target identifier which define the appropriate target set for each change. If the user desires that one of those characteristics not be used in determining the target set, he may:

1. use the word "ALL" as the desired name, or

2. leave the field blank.

For example, if the value of ICLASSWAN is either blank or "ALL", subroutine MAKECHG assumes every target meets the class name criterion for this request regardless of the target class name. However, if a blank card is input as a change request, an error message is printed and the card is ignored.

The new values for the planning factors are checked for the appropriate range of values. If a target value less than zero or a MINKILL or MAXKILL not in the range from zero to one is encountered in a change request, the card is flagged as an error and the out-of-range value is reset to the nearest acceptable value.

For example, negative values are set to 0. Values of MINKILL or MAXKILL greater than 1 are set to 1.

The local variable IOPT is used to determine which entry point was used to enter this routine. For entry VALUMOD, IOPT = 1; entry MINMOD, IOPT = 2; entry MAXMOD, IOPT = 3. This variable is used in the computed GO TO statement preceding statement 310 to determine the correct array to store the last location used for the change requests on the current call to this routine. If IOPT = 1, value changes were requested and INDVAL(2) is set in statement 310. If IOPT = 2, MINKILL changes were requested and INDMIN(2) is set in statement 320. If IOPT = 3, MAXKILL changes were requested and INDMAX(2) is set in statement 330.

If more than 2,000 requests for planning factor changes are input, the excess requests are ignored.

Subroutine VALUMOD is illustrated in figure 24.

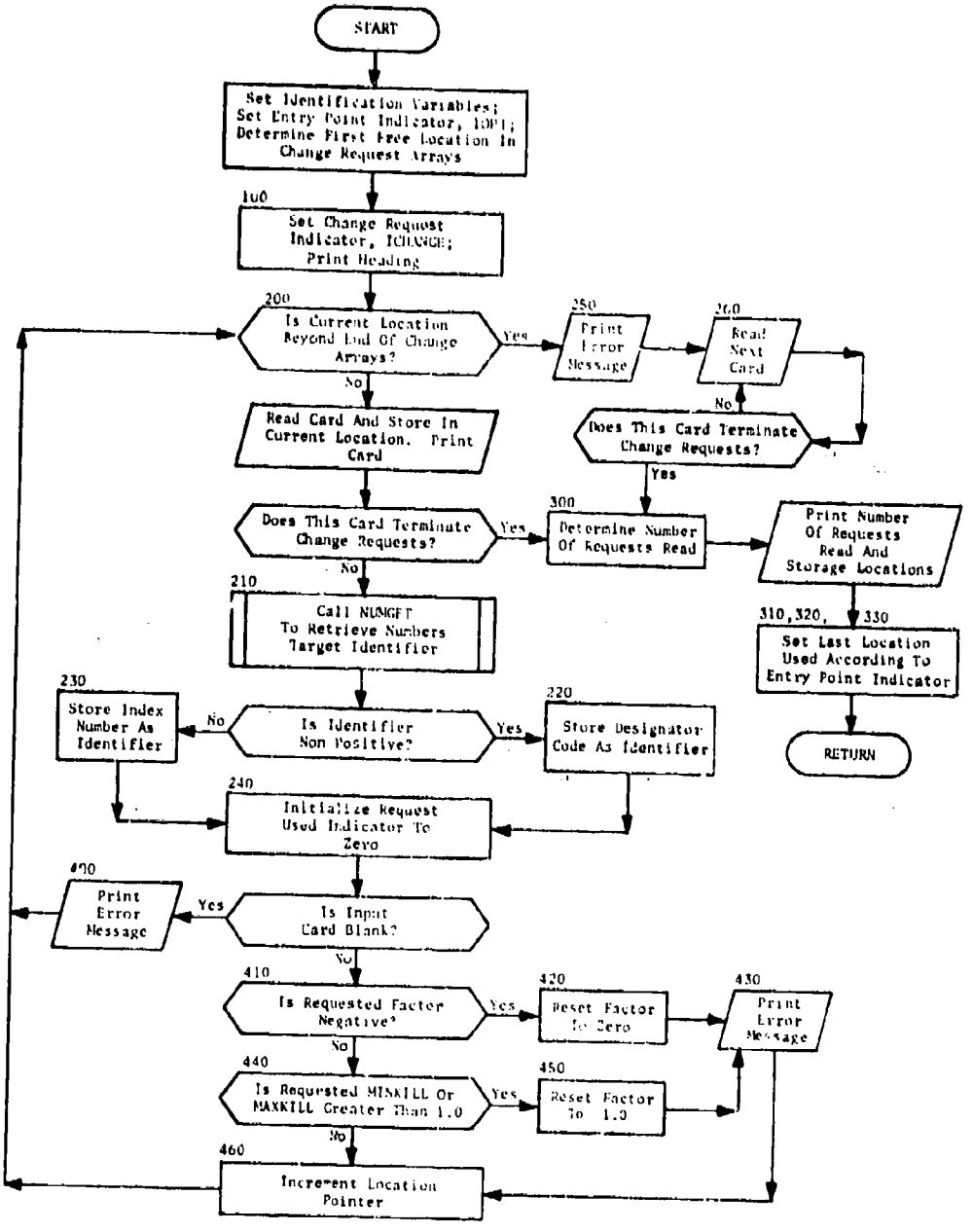


Fig. 24. Subroutine VALUMOD

SUBROUTINE WEAPPREP

PURPOSE:	This routine reads the weapon data from WINFILE and fills weapon arrays to be written on the BASFILE file.
ENTRY POINTS:	WEAPPREP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, FILES, DPENREF, WPNREG, WPNDATA, PLANTYPE, PAYLOAD, ASMTABLE, WARHEAD, CORRCHAR, BOUNDARY, HAPPEN, CHARTER, ITP, IETPRNT, PREALOC, INPSTOR, PRNTCTRL, MYIDENT, TEMPO, RECOVR, EXCESS, NAVAL, DUMCORR, NOWFILE, SIZES, TWORD
SUBROUTINES CALLED:	SETWRITE, RDARRAY, WRARRAY, DISTF, PRINTDAT, TERMTAPE, WRWORD
CALLED BY:	PREPALOC

Method

When WEAPPREP is called, WINFILE is positioned immediately preceding the complex and multiple target data. These targets, along with the end sentinel of Z's, are copies onto the POSTDATA file without modification. This data is transferred by subroutine BASWRIT to the BASFILE. Then the warhead, ASM, payload, weapon region, and weapon type data are read and stored in their respective common blocks. Selected attributes from the weapon types are stored in common /WPNDATA/. The other type data are held in temporary storage until the group attributes are read. Each weapon group is read in and common /WPNDATA/ filled. The weapon group and its corresponding type data are then written on the POSTDATA file.

The number of weapons in each group is increased according to the proportion specified in common /EXCESS/. The destruction before launch probability is then modified by a factor to maintain the same number of expected launched weapons as before.

For each bomber group the distance from the refuel point to each penetration corridor is computed and stored in array DISTAC of /WPNDATA/. If the group does not have a refuel point assigned, the group coordinates

are used instead. Also the time required by each bomber group to reach its refuel point is computed and stored in the same common block.

The first 200 locations of array DISTAC are used for the fraction of weapons in each group that are ASMs rather than bombs (local array EXPASM).

Finally, common blocks /BOUNDARY/, /CHARTER/, /HAPPEN/, and the tanker data from WINFILE are written on the POSTDATA file separated from the weapon data by a word of Y's. The POSTDATA file is terminated with a word of X's. The print subroutine is called and the distances computed above will be listed if option three was chosen.

Subroutine WEAPPREP is illustrated in figure 25.

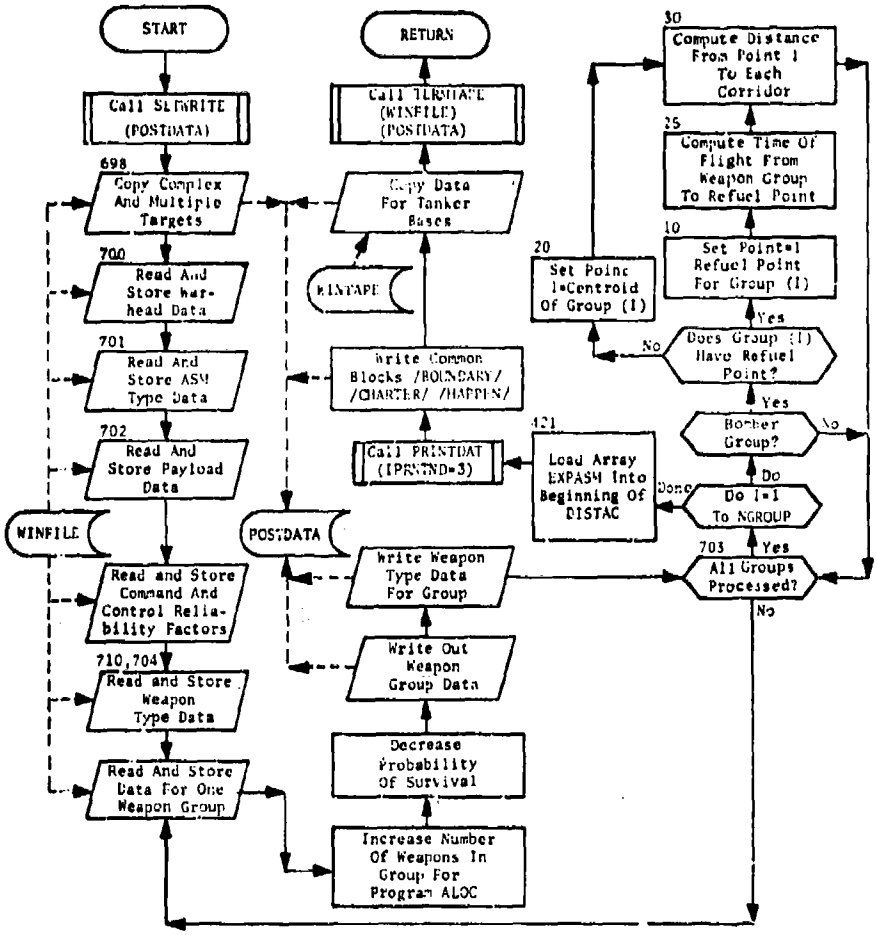


Fig. 25. Subroutine WEAPPREP

CHAPTER 4
PROGRAM ALOC

PURPOSE

The major purpose of this program is to determine the optimal allocation of weapons to targets, using a Lagrange multiplier technique. The weapons are divided into weapon groups. A group contains weapons of the same characteristics which are geographically proximate. Thus, weapons are considered identical within groups. Each target is considered individually for weapon assignment. The order of investigation was determined by program PLANSET when it shuffled the targets from their data base ordering. When all targets have been processed, another pass over the targets begins. This process continues until the Lagrange method has allocated all the weapons to the targets. The assignments are then passed to later processors which calculate the detailed sorties for each weapon.

The program allows the user to specify the assignment of weapons to specific targets. This fixed assignment capability provides for the program to allocate optimally those weapons which are not user-fixed. The input for fixed assignment information is either generated by program PREPALOC for input on the TGTFILE or generated for direct input to program ALOC.

In addition, there are other minor capabilities which allow the user to modify weapon range values, to restrict the use of MIRV weapons by target class, and to restrict the use of any weapon group by flag (which is usually set equal to region) or country.

As input, program ALOC uses the data on the TGTFILE and BASFILE concerning the target system and weapon stockpile (respectively). Using user-input parameter information concerning allocation restrictions (e.g., fixed assignments, range restrictions) program ALOC generates an assignment of weapons to each target. This assignment is used by later processors to generate sortie specifications (attack plans) for each bomber and missile.

INPUT FILES

There are two input files for program ALOC, the BASFILE and the TGTFILE. Both of these files are created by program PREPALOC. The BASFILE data

define the basic data base parameters (/MASTER/), the file utilization information (/FILES/), the penetration corridor descriptions (/CORRCHAR/), the payload information (/PAYLOAD/), the refueling point data (/DPENREF/), the plan timing information (/PLANTYPE/), the command and control reliability data (/WPNREG/), the weapon type data (/WPNDATA/ in BASFILE; /WPNTYPE/ in ALOC), the weapon group data (/WPNDATA/ and /NAVAL/ in BASFILE; /WPNGRP/ and /PKNAVAL/ in ALOC), and the country location code list (/CTRYCD/). This BASFILE information defines the weapon characteristics and the necessary geography to plan the routings of weapons to the targets.

The TGTFILE defines the target characteristics for each target. The target parameters (e.g., latitude, longitude, value) are read into common block /ALOCIN/ in program ALOC. These parameters are used to define the interactions between the weapons and each target. (This information is stored on the WPNTGT temporary scratch files.) The target parameters are read only once from the TGTFILE. After the first pass by program ALOC through the target list, the necessary target data have been stored on the ALOCTAR file and the ALOCT1 scratch file.

OUTPUT FILES

There are two output files from program ALOC: ALOCTAR and MSLTIME. The ALOCTAR file describes each target and lists all weapons allocated to the target. The MSLTIME file contains information on all missiles assigned through the fixed assignment capability of program ALOC.

Both files are produced by the QUICK system filehandler. The description of the physical format of these files is described in the section of this manual pertaining to the filehandler. This current section will describe the logical format of these files.

On the ALOCTAR file, each target is represented by a variable length logical record (see table 11). This record contains 45 words which describe the basic target information. Following these words are data which identify the weapons allocated to the target. Note that the ALOCTAR file is written and rewritten during the course of an ALOC run. This file and the ALOCT1 scratch file are used as dynamic data files. As each pass through the target list progresses, the weapon allocation data on the ALOCTAR file change to reflect the new allocations. Common block /DYNAMIC/ is used for internal storage of the ALOCTAR data.

The MSLTIME file contains information on each missile assigned by the fixed assignment capability. Each missile so assigned generates a five

Table 11. Format for ALOCTAR File Logical Record Data Blocks
(Sheet 1 of 2)

VARIABLE OR ARRAY	DESCRIPTION
TGTNAME	Hollerith target name
INDEXNO	Index number of target
DESIG	Target designator code
TASK	Target task code
CNTRYLOC	Target country location code
FLAG	Target flag code
TGTMULT	Target multiplicity (original)
TGTLAT	Target latitude
TGTLONG	Target longitude
TGTRAD	Target radius
VTO	Original target value
M	Number of hardness components (≤ 2)
H(2)	Hardness of each component
VO(2)	Original value of each component
NK	Number of time periods (≤ 3)
FVAL(3)	Fraction value escaping in each period
TAU(3)	Time ending each period
IHCLASS	Hollerith target class name
ICLASS	Target class number
IHTYPE	Hollerith target-type name
TARDEF	Local bomber defense factor
INDYPEN	Depenetration corridor index
DISTDF	Distance from target to end of depenetration
DISTDG	Distance from target to recovery base
NBLN	= number of terminal ballistic missile interceptors if a STALL allocation = minus the number of interceptors if a DEFALOC allocation
CTMULT	Current target multiplicity

Table 11. (cont.)
(Sheet 2 of 2)

VARIABLE OR ARRAY	DESCRIPTION
VT	Value remaining after allocation of weapons
TGTWT(3)	Target weighting values
PAYOFF	Payoff on this target [VTO-VT-COST]
COST	Sum of Lagrange multipliers of all weapons allocated to target
PROFIT	PAYOFF - COST
DPROFIT	Difference in profit between passes
WRTEST	Test value for weight rates
IHEOT	End of information marker
NUMFIX	Number of weapons allocated by fixed assignment capability
ITGT	Target number
NUM*	Number of weapons assigned
IG(NUM)	Group number of assigned weapons
KORR(NUM)	Weapon penetration corridor
RVAL(NUM)	Relative value of weapon allocation
PEN(NUM)	Weapon penetration probability
TOARR(NUM)	Weapon time of arrival on target

*If there are no weapons assigned, NUM is equal to 0 and none of the remaining arrays are output on the file.

word logical block on the MSLTIME file (see table 12). The records are ordered according to increasing weapon group number. The information on this file is used by program PLNTPLAN to determine precise launch times for all fixed missiles.

Table 12. Format of Records on MSLTIME File

VARIABLE	DESCRIPTION
IGROUP	Group number (the last record in the file has IGROUP = 999)
INDEXNO	Index number of target
DESIG	Target designator code
TASK	Target task code
TIME	Time of arrival information

CONCEPT OF OPERATION

Program ALOC is divided into a series of functions. The major function, ALLOCATE, allocates the weapons to the target system. The constraint functions RANGEMOD, MINRANGE, FLAGREST, LOCREST, and MIRVREST, restrict the use of weapons against certain targets. The convergence control functions, READMUL and PUNCH, help control the convergence of the Lagrange multipliers. The termination functions, STOP and DUMP, determine the mode by which the program is terminated. Since these functions are separated into separate subroutines, their operation will be described by subroutine. Except for the ALLOCATE function, all calculations and operations are simple and straightforward. The only required functions are ALLOCATE and STOP. All the others are optional.

Constraint Functions (Optional)

Subroutine RNGEMOD (RANGEMOD Option): This subroutine will allow the user to specify a new range (both refueled and unrefueled) for any weapon group. This specified range will be in effect only for program ALOC. Later processors will use the range specified in the data base. This function allows the user to restrict weapons by range of the sortie.

Subroutine MINRNGE (MINRANGE Option): This entry in subroutine RNGEMOD will allow the user to specify the minimum range between base and target for any weapon group. Targets closer to the group than the minimum range will not be considered for allocation of weapons from the group.

Subroutine MIRVRST (MIRVREST Option): This subroutine allows the user to restrict the use of each weapon system with a MIRV capability by target class. In addition to listing the classes that may be targeted by each weapon system, the user can also include multiple targets and targets with terminal ballistic missile defense as permitted targets. Each MIRV weapon system can be restricted independently of the others.

Subroutine FLAGRST (FLAGREST Option): This subroutine allows the user to restrict the allocation of weapons from any group according to the attribute FLAG. (This attribute is usually set to be equal to the region, but may be set in the data base to any value the user desires.) Weapon groups may be permitted or forbidden to strike targets according to the FLAG value for the targets.

Subroutine LOCREST (LOCREST Option): This subroutine restricts weapon use in a manner similar to that of subroutine FLAGRST. In this case, the weapon groups are restricted by the country code (CNTRYLOC in the data base).

Convergence Functions (Optional)

Subroutine READMUL (READMUL Option): This subroutine reads the values of the initial Lagrange multipliers to initiate the run. It is useful when the user has some information on the relative worth of the weapon systems. The closer the initial multipliers are to the optimal set, the less time is spent in the allocation process.

Subroutine PUNCHM (PUNCH Option): This subroutine punches the final multipliers from the Lagrange process. It allows the user to reinsert these values in a later similar plan to speed up the allocation.

Termination Functions

STOP: This function is performed in subprogram ALOC and provides normal termination. This function may be replaced by the DUMP function.

DUMP: This optional function replaces the STOP function and uses utility subroutine ABORT to terminate the job with a memory dump.

Allocation Function - ALLOCATE (Required)

The QUICK system is capable of allocating up to 200 distinguishable weapon groups against a target system of up to 5,000 targets. It is clearly not feasible to keep information in memory concerning the capability of all weapons with respect to all targets. Moreover, it would be inefficient to recompute such information each time it was required. Therefore, during the first pass through the targets, subroutine GETDATA prepares files which for each target list the capability of all relevant weapon types. In this way, on later passes, it is possible to examine efficiently any desired allocation of weapons against each target. To minimize the time required for the allocation, as much of the needed information as possible is precomputed and stored on these files. The allocator generates a complete weapon assignment against each target before proceeding to the next. It thus passes through the files one target at a time and then recycles the files as required to obtain a satisfactory allocation.

The design of the weapon-to-target allocator utilizes a hierarchy of eight subroutines operating at different levels of detail. Figure 26 illustrates this hierarchy. The major functions associated with these subroutines are summarized below and related to the overall concept in subsequent paragraphs.

Subroutine MULCON is the first subroutine in the hierarchy and is responsible for the control and adjustment of the Lagrange multipliers. MULCON monitors the rate at which various classes and types of weapons are being allocated to the target system and makes appropriate adjustments in the values of the Lagrange multipliers. In this role, MULCON does not need any detailed information concerning actual allocation. It is concerned only with the actual rate of allocation of the available inventory as the targets are processed. To obtain the assignment of weapons to each successive target, MULCON simply calls subroutine STALL (Single Target Allocator) for targets without missile defenses, or subroutines STALL and DEFALOC if the target is defended. STALL and DEFALOC utilize the current values of the multipliers to make an allocation to the next target, then return control to MULCON. However, since MULCON is the main controlling routine it is convenient for it to handle much of the data input and output operations.

In this second role, it reads and writes the dynamic target data files, ALOCT1 and ALOCTAR (or ALOCT2), which contain the allocation information. Since this information changes from pass to pass, these files are called dynamic. The data on the weapon-target interaction files prepared by subroutine GETDATA do not change from pass to pass. Hence these files are called the static files. (The format of the ALOCT1 file is identical to that of the ALOCTAR file as shown in table 11.) In addition, MULCON processes some of the user-specified fixed assignments. These are specific weapon allocations requested by the user to be used regardless of their profit. For fixed missiles, some information is stored on the ITMPMSL file for use by a later program PLNTPLAN. The ITMPMSL file is reordered and renamed MSLTIME in ALOC. The format of both files is the same (see table 12).

The function of subroutine GETDATA varies according to the number of passes made through the target list. On the first pass, this routine does detailed weapon-target interaction calculations for each weapon against each target. For bombers, it computes the penetration corridor with the lowest total attrition. For missiles, it computes the probability of penetration through area and terminal missile defenses. For all weapons, interweapon correlations in delivery are also computed, to be used in determining the optimal cross targeting of weapons. All of these data are stored on temporary disk files (WPNTGT files) in the first pass. In later passes, GETDATA merely reads the data from these files and reconstructs all the weapon-target interaction parameters. Table 13 presents the format of the WPNTGT files. The number of WPNTGT files used is dynamically determined by subroutine GETDATA. Each WPNTGT file must be less than one million words long.

Subroutine STALL is the next subroutine in the hierarchy when dealing with targets without missile defenses. STALL utilizes the values of the multipliers supplied by MULCON and generates an appropriate allocation of the weapons to the specified single target. It is not responsible for computing payoffs and is not responsible for actually adding or deleting weapons. When STALL has determined that a single weapon should be added, or deleted, it calls the Weapon Addition and Deletion subroutine WAD. WAD then adds or deletes any weapon as specified and corrects the residual target value. In addition, before returning to STALL, WAD examines every other relevant weapon group and calculates the potential change in payoff, if a weapon from that group was added or deleted. This information on potential payoffs is used by STALL in determining whether other weapons are to be actually added or deleted. When STALL has achieved an allocation of weapons to the target appropriate for the current values of the multipliers as supplied by MULCON, it returns control to MULCON.

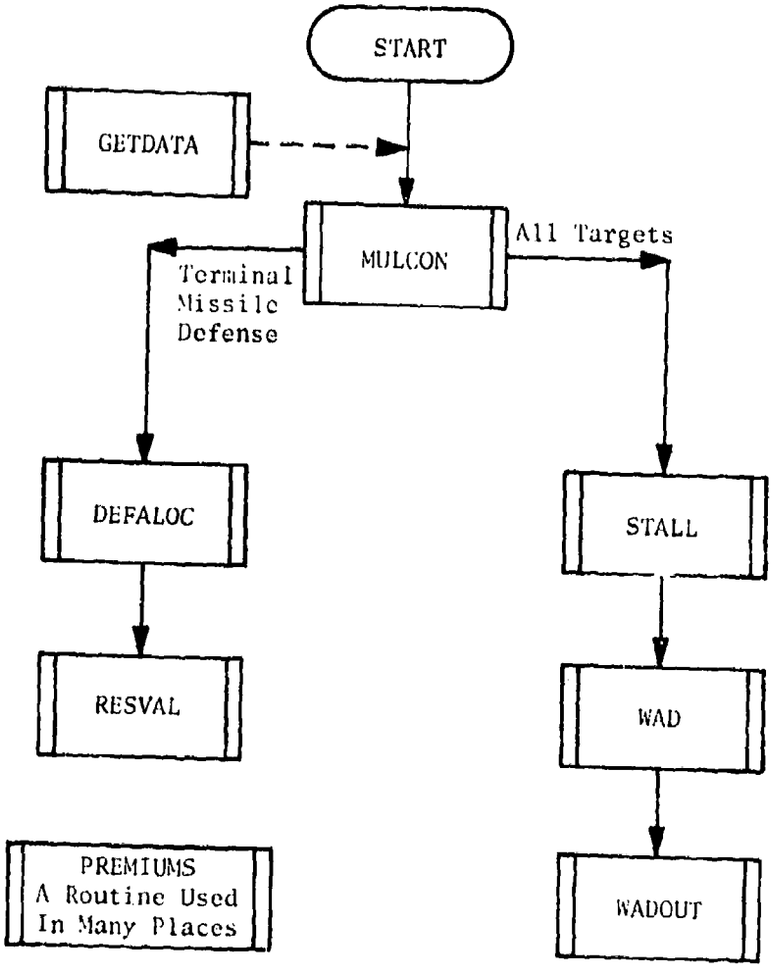


Fig. 26. ALOC Calling Sequence Hierarchy

Table 13. Format For WPNTGT Files

VARIABLE OR ARRAY*	DESCRIPTION
JTGTX	Target number
LNEXT	Length of record for this target
MINKILL	Minimum required fractional kill
MAXKILL	Maximum allowed fractional kill
MAXCOST	Maximum cost ratio allowed to provide MINKILL
MISDEF	Number of terminal ballistic missile interceptors
NACTV	Number of weapon groups active on this target
IGX(NACTV)	Weapon group number
TOA(NACTV)	Weapon time of arrival
MORR(NACTV)	Preferred penetration corridor
PEX(NACTV)	Penetration probability
STK(M*NACTV)	Warhead kill probability (for M hardness components)
STK2(M*NACTV)	Modified kill probability for targets with terminal ballistic missile defense

Each target is represented on a WPNTGT file by one physical record of the preceding format. The files do not use the QUICK filehandler for input/output.

*Parenthetical values are array dimensions. All other elements are single word variables.

While the allocations generated by STALL are determined by the values of the Lagrange multipliers and the target payoff functions, there is no requirement for STALL to be involved in the calculation of these quantities. Thus, the structure of STALL can be independent of the details of the operation of either WAD or MULCON.

Subroutine WAD (weapon addition and deletion) is the next subroutine in the hierarchy and is responsible for the mechanics of addition and deletion of weapons and for the actual calculation of payoff for targets without missile defenses.

Subroutine WADOUT is called by WAD to summarize the output of WAD for STALL. WADOUT calculates an overall benefit for using each weapon by adding the current premium for using weapons from that group to the potential payoffs computed by WAD. These benefits are then compared with the current prices (or Lagrange multipliers) to produce the summary data actually used by STALL. Thus STALL, in fact, is attempting to maximize (PAYOFF + PREMIUM - COST) rather than just (PAYOFF - COST). The introduction of the premium provides a flexibility which is used to accelerate convergence to an allocation that exactly matches the stockpile.

Subroutine DEFALOC performs the same function as STALL for targets with terminal ballistic missile defense. The complication that necessitates a separate subroutine is the nonconcave payoff function for defended targets. DEFALOC determines whether it is more profitable to attack the target with missiles until the interceptors are exhausted than to use the STALL allocation. If it is not profitable to exhaust the defense, then the allocation job is turned over to STALL, after setting the missile penetration parameters to reflect leakage through the defense. In the event an exhaustion tactic is most profitable, DEFALOC calls subroutine RESVAL to calculate the payoff against the defended target with a specified mix of weapons.

Subroutine RESVAL calculates the damage to a ballistic missile-defended target when attacked with a specified mix of weapons. The attacking missile payloads may contain decoys and electronic penetration aids which degrade interceptor effectiveness. The target is defended with a prespecified nominal number of terminal interceptor salvos with kill probability PKTX against unhardened warheads. Uncertainties may be introduced into the number of interceptors by specifying two probabilities PK(1) and PK(2) that the actual number of interceptors will be RX(1) lower or RX(2) higher than the nominal number.

All of those factors discussed under WAD are considered in RESVAL except correlations in weapon failures.

Subroutine PREMIUMS is the final subroutine in the hierarchy. It is called to calculate the current premiums for adding or deleting any weapon. The premium reflects whether the weapon is currently over-allocated or underallocated. The size of the premium and the way it is calculated change as the allocation progresses.

The remainder of this section treats the subroutines one at a time.

Subroutine MULCON: The flow of operations in MULCON is illustrated in figure 27. The diagram is broken into two parts. Part I is the main bookkeeping loop. Part II is the computational loop. (The numbers above the boxes in the figure correspond to the FORTRAN statement numbers in the program; i.e., statements corresponding to box 14 will be between 1400 and 1499.)

There is an initial setup phase in this subroutine which calls subroutine SETABLE to initialize the damage calculation table, subroutine INITGET to initialize subroutine GETDATA, subroutine TIMEME to initialize the clock, and subroutine RDALCRD to process the user-input parameter cards. The operation of this phase is straightforward and will not be discussed further.

The analytic techniques upon which this subroutine is based are discussed in the Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques (Weapon Allocation, Adjustment of Multipliers).

The loop shown at the beginning of Part I of figure 27 has two branches. The left branch is used only on the first pass over the target system. All succeeding passes use the downward branch. On the first pass, the raw data on target characteristics for each target are read in by subroutine GETDATA from the TGTFILE supplied by PREPALOC. Then the basic information on capabilities of each weapon with respect to the target is computed. Since these data are independent of the allocation to the target, they are stored on files (known as the static files) to avoid recomputation of the data on later passes. In block 10, just before weapons are actually allocated to the target, the allocation previously recorded for the target (in the initial pseudo-allocation) is removed by deleting it from the running sum used to estimate allocation rate. That is, the contribution of the target i to $\Sigma_i N(i,J) * W(J)$, known as RUMSUM,* and $\Sigma_i W(i)$, known as WTSUM,* is removed.

---

* See Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques (see Weapon Allocation, Adjustment of Multipliers), and Chapter 3, Calculations (Algorithms, Lagrange Multiplier Adjustment).

In block 14, after STALL and/or DEFALOC has been called for an allocation to the target, the running sums are augmented by the new contribution of the target i using the new (and usually much larger) target weight W(i). Finally, the data describing this allocation are written out on the dynamic file for later reference. (This file is called dynamic because it contains data that change on successive passes, as opposed to the data on the static files which are constant for all passes.)

At the end of the pass, the alternate dynamic file is selected for recording the allocation on the next pass while the dynamic file written on the first pass is rewound. In this way, information on the previous allocation to each target can be read in when each target is considered again.

On succeeding passes (block 12), the data from both the current static file and the dynamic file are read in before proceeding with the bookkeeping for each target. The availability on the dynamic file of the old target weight W(i) and a list of the weapons previously allocated makes it possible on each successive pass (block 13) to remove the prior contribution of each target to the running sums.

Again, after the new allocation has been made, it is added into the running sums and recorded on the dynamic file (blocks 14 and 15). Of course, multiple targets (single target records on the file which represent several identical targets at slightly different locations) are subtracted from and added to the sums as multiple targets. During the closing phase, these targets may be separated to allow different allocations to the separate targets. Provision is made at the end of block 15 to recycle and process later elements of the same multiple target if such a split occurs during the allocation to the target. Before passing on to the next target, the current value of the target weight is revised in block 17.

After every two to four targets, the Lagrange multipliers are updated by transferring control to the multiplier computation loop shown on figure 27. The error in the rate of allocation for each collection of weapons J is estimated (block 20). Three separate estimates are made corresponding to differing rates of increase of the target weights. If all estimates have the same sign, then a small adjustment of the multiplier in the indicated direction is made (blocks 23 and 24). For

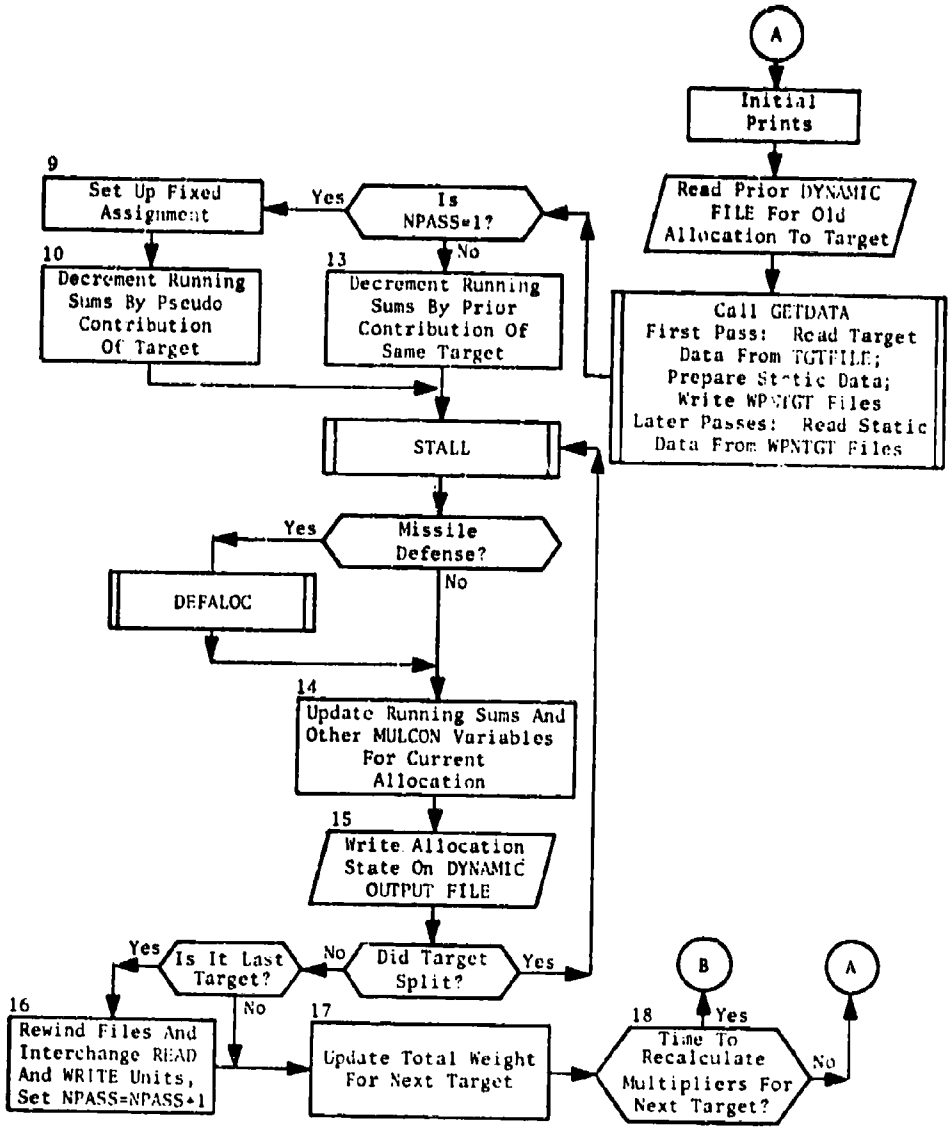


Fig. 27. Subroutine MULCON
Part 1: Bookkeeping Loop

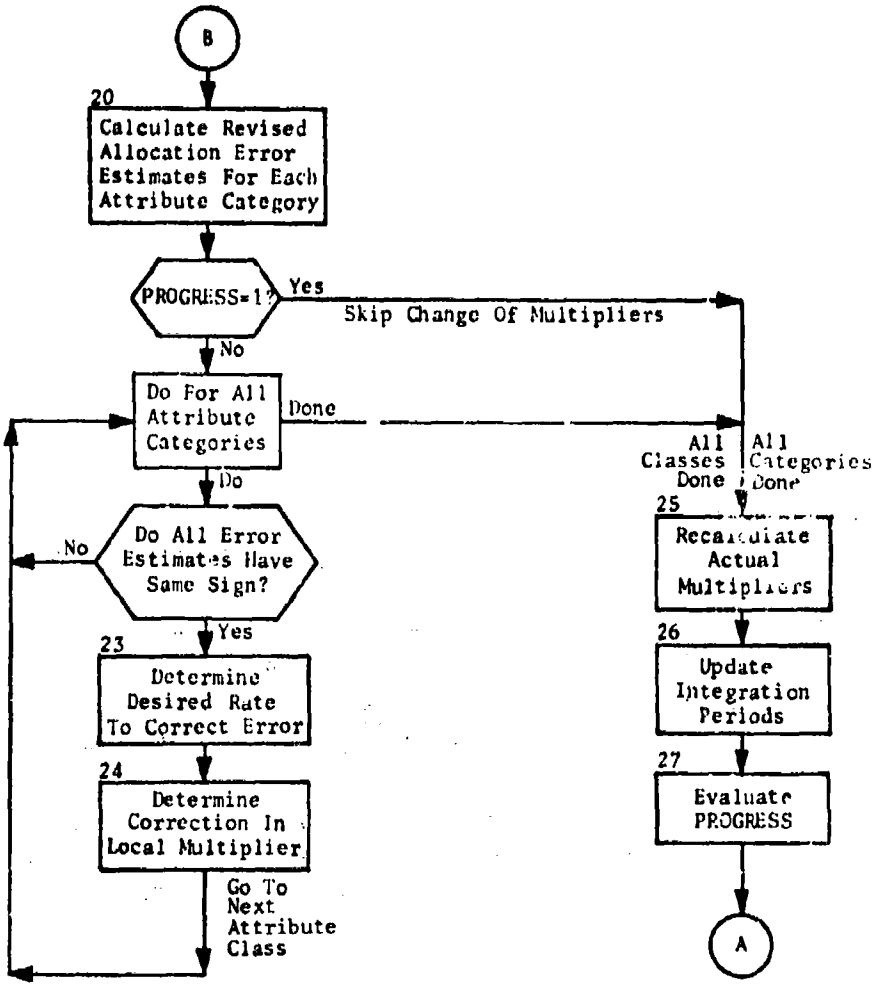


Fig. 27. (cont.)
Part II: Computation Loop

the details of this adjustment see the Analytical Manual* and the more detailed description of subroutine MOLCON in the following section. The revised local multipliers are then used (block 25) to recalculate the Lagrange multipliers. During the closing phase (PROGRESS = 1), the local multipliers are not changed so the actual multipliers, computed in block 25, remain unchanged. In block 26, the rate of change of the target weight is adjusted depending on the apparent size of the current error in the allocation rates.

Finally, in block 27, the progress of the allocation is evaluated and flags are set if the mode of operation is to change.

Subroutine GETDATA: This subroutine calculates the weapon-target interaction data and reads and writes the static WPNTGT files. The calculation of these data is straightforward. The processing in this subroutine is merely the application of the formulas presented in the Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques (see Weapon-Target Interaction and Weapon Correlations).

Subroutine STALL: STALL is basically a very simple routine. It is not responsible for computing payoffs. The payoffs are computed by WAD and summarized for STALL by WADOUT. Thus, the input to STALL consists only of the summary data provided by WADOUT. These data consist of only the following variables:

1. PPMX the maximum potential profit available if a weapon is added, and IPPMX the index to that weapon

2. PVRMX the maximum efficiency for any potential weapon that could be added, and IPVRMX the index to that weapon

3. DPMN the minimum differential profit produced by any weapon on the target, and IDPMN the index to that weapon.

Actually, of course, the profits and efficiencies mentioned above are based on a modified payoff (or BENEFIT) computed by WADOUT, which includes the actual payoff from WAD together with the premiums for staying close to the desired allocation rates. Thus, throughout the allocation, the operation of STALL remains the same; it simply tries to maximize this modified payoff. Changes in the mode of the allocation are thus

---

* Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques (see Weapon Allocation, Adjustment of Multipliers), and Chapter 3, Calculations (Algorithms, Lagrange Multiplier Adjustment).

accomplished in WADOUT simply by changing the way the payoffs are modified, so no change in the logic of STALL is required.

To obtain the initial values of these quantities, STALL makes an initialization call on WAD. Then it adds the weapons which have been fixed to the target by the user. On the first pass, the arrival time of fixed missiles is written on the ITMPMSL file. This file is sorted by subroutine SORTMIS to create the MSLTIME file for use by program PLNTPLAN.

On the basis of the values delivered by WADOUT, STALL decides whether to add a weapon. After each call on WAD to add or delete a weapon, a new set of variables is delivered by WADOUT, and STALL uses this revised information to decide whether more weapons should be added or deleted and finally when to terminate the allocation.

Figure 28 illustrates the operation of STALL. As Part I of the flow chart shows, a special option has been provided so that, in the case where IVERIFY = 2 and PROGRESS = 2, the normal operation is short circuited and STALL simply duplicates the previous allocation to the target so that with one additional pass the allocation can be evaluated with different correlation coefficients.

In all other cases, the normal allocation procedure is used. This procedure consists of four parts:

I. Setup and single weapon allocation phase

II. Fixed weapon processing

III. Multiple weapon laydown loop

IV. Multiple weapon refinement loop.

Parts I, III, and IV are discussed in the Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques, (see Weapon Allocation, Single Target Allocation - Targets Without Terminal Ballistic Missile Defenses). Part II is discussed in the same manual under Weapon Allocation, Combined Fixed Optimum Assignment Capability. Figure 28 presents the flow of operations in subroutine STALL.

The most time-consuming part of subroutine STALL is the multiple weapon refinement loop. This phase tests many permutations of weapon assignments which could be made. The full testing of every single target allocation can considerably slow down the operation of the allocator. Therefore, a way of terminating the testing has been provided, through the user-input parameter QUALITY. The maximum number

of weapons which will be removed in any testing process is not permitted to exceed NUM * QUALITY where NUM is the number of weapons allocated to the target. There QUALITY is a measure of the fraction of the weapons which can be re[illegible]d. If QUALITY is set to 0, the refinement operation is skipped ent[illegible].

Since the program provides only finite arrays to list the number of weapons assigned to a target and for WAD to compute surviving target value for different times of arrival, there is always the possibility that the arrays provided might be exceeded. Therefore, before any weapon is added, a test is made to be sure it will not overflow these arrays.

If the maximum number of weapons would be exceeded, then the least profitable individual weapons are removed to make way for the most profitable weapons. If the number of time-of-arrival columns would be exceeded, the error criterion for treating slightly different times of arrival in the same column is relaxed, but the allocation to this target must be reinitialized and repeated.

Subroutine WAD: When the next target is to be processed, WAD is called with the control variable WADOP set to 1. This results in the initialization of the allocation for that target, starting with zero weapons assigned. STALL examines these data and determines what to do. On all succeeding calls, the print options 9 and 22 are available to print the decision made by STALL, together with the data (previously given by WAD to WADOUT) on which the decision was based. If STALL decides to add a weapon, WAD is called with WADOP = 3 and with the variable G specifying the group from which the weapon is to be added. If STALL later decides to delete a weapon, WADOP is set to 4, and the variable NW is set to specify which weapon in the list of those assigned (the NWth weapon in the list) is to be deleted. Finally, when STALL decides the allocation is complete, a dummy call WADOP = 2 is made to permit a print of the data which caused STALL to terminate the allocation. The option WADOP = 5 should never be used. It is provided simply to catch any erroneous calls (WADOP > 4) on WAD.

Each of the main options (WADOP = 1,3,4) causes control to pass to an internal control routine, which in turn makes calls on appropriate local subroutines in WAD. The structure of these routines is governed strongly by the objective of speed and efficiency, and need not be dealt with here. The techniques underlying the design of WAD are discussed in the Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques, (see Planning Factor Processing and Weapon Correlations).

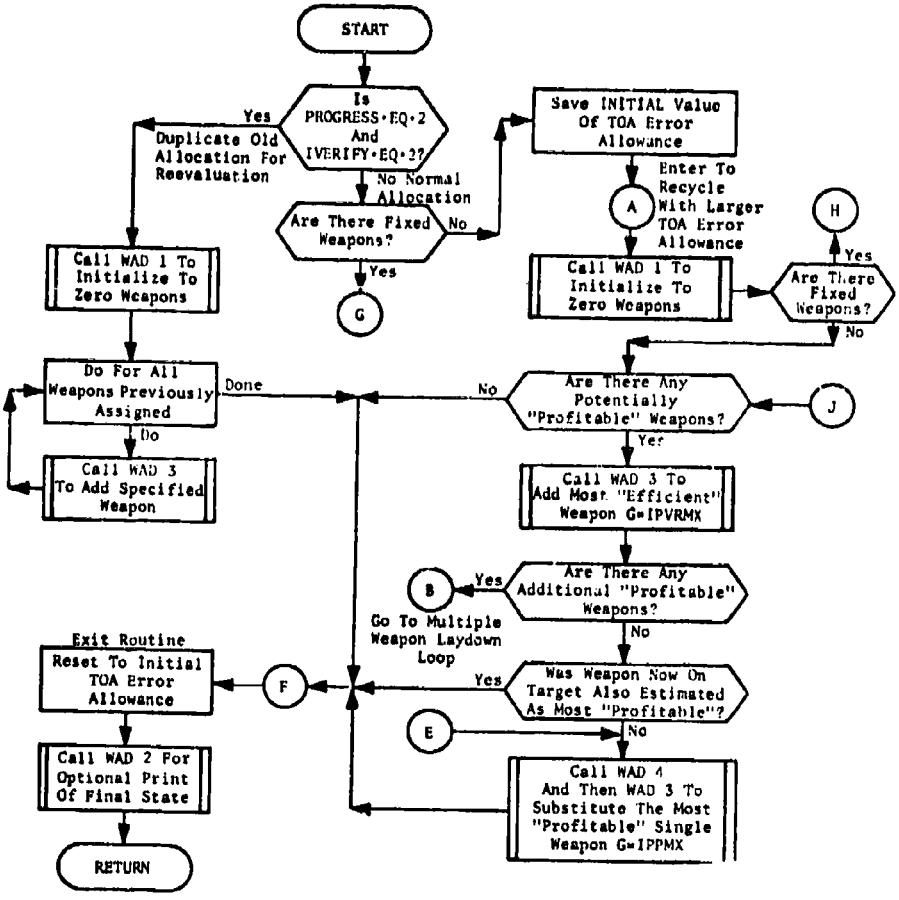


Fig. 28. Subroutine STALL
Part I: Setup and First Weapon

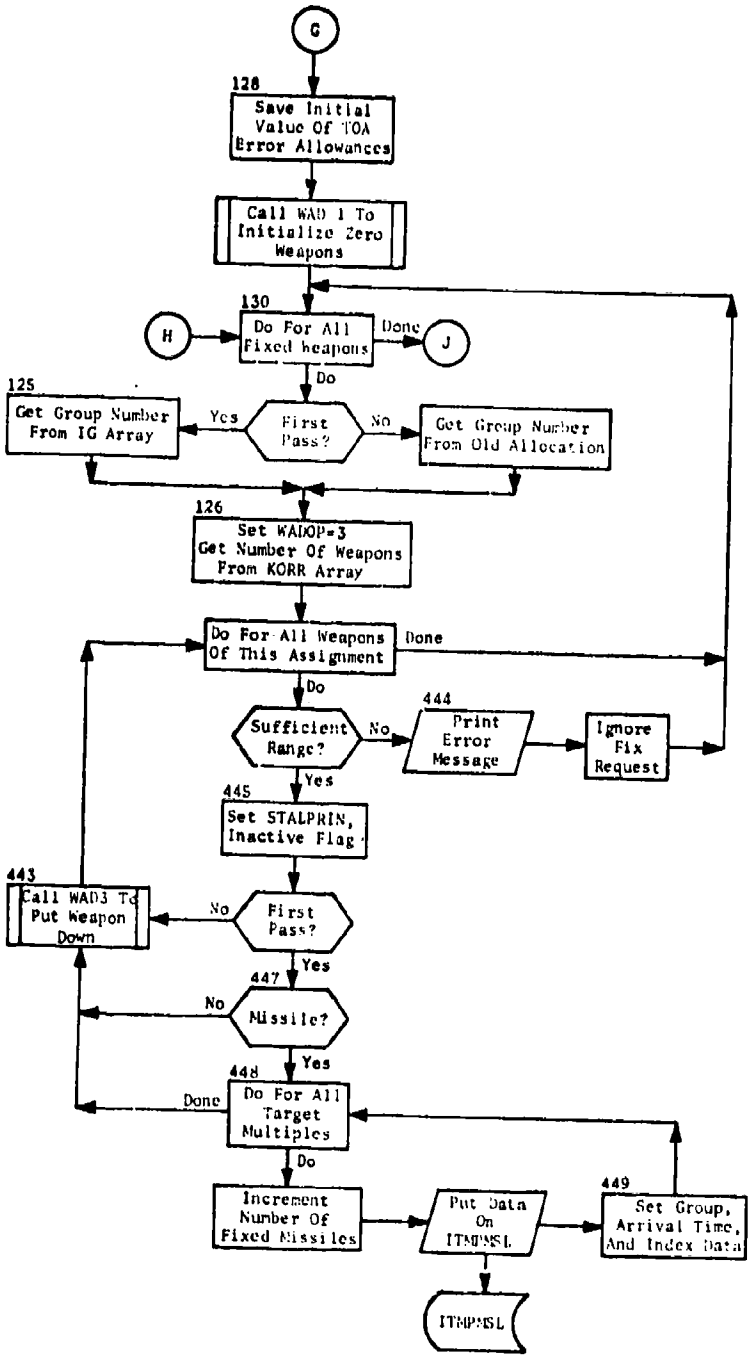


Fig. 28. (cont.)
Part II: Fixed Weapon Assignment Processing

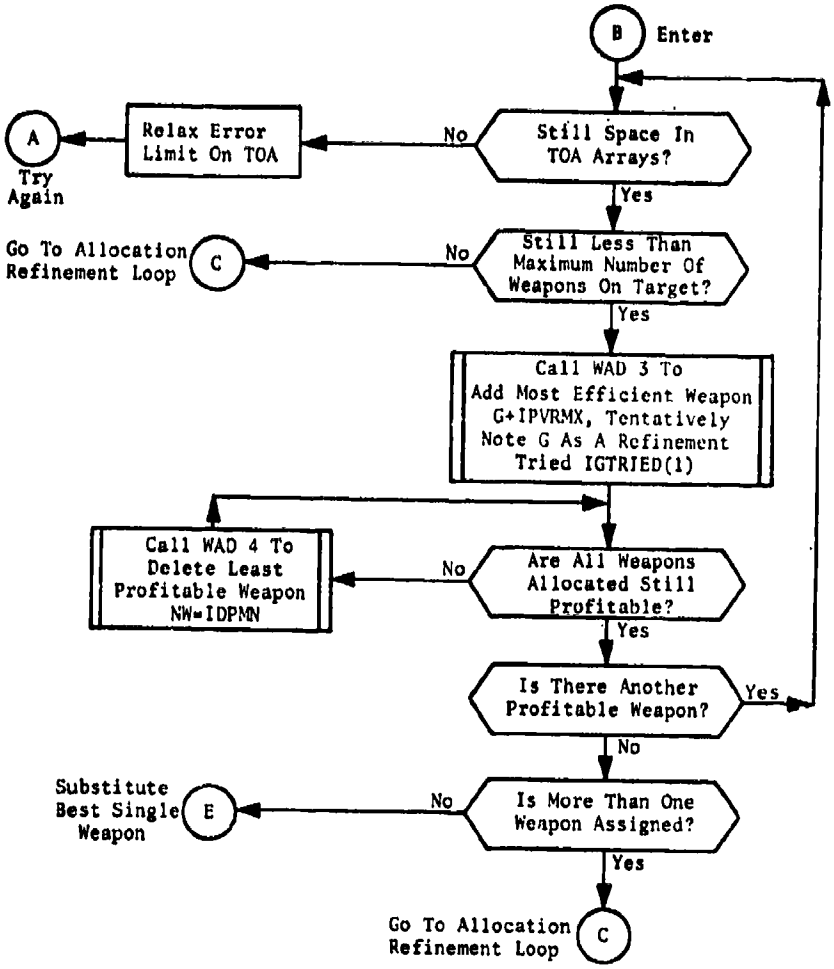


Fig. 28. (cont.)
Part III: Multiple Weapon Laydown

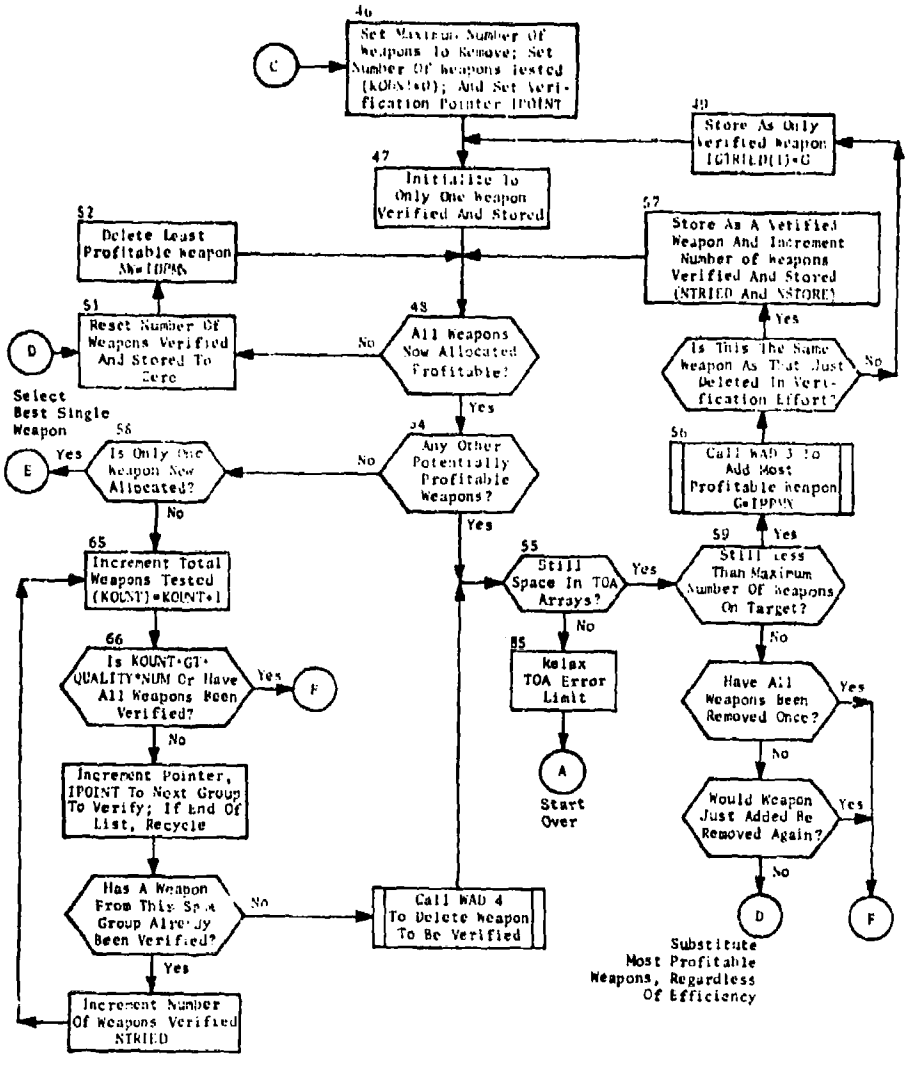


Fig. 28. (cont.)
Part IV: Allocation
Refinement Loop

Subroutine WADOUT: Much of the logic of the subroutine is concerned with the decisions of which weapon groups to make INACTIVE to save time in the computations in WAD. The variable INACTIVE outside of WADOUT is interpreted as true so long as it is not 0. That is, only weapon groups for which INACTIVE = 0 are processed. If INACTIVE = 100, it implies that the range is inadequate, and the weapon will be inapplicable regardless of its Lagrange multiplier. This value of INACTIVE is set permanently where the weapon-target interactions are computed in subroutine GETDATA. INACTIVE = 2000 or 30000 is used to note weapons that are inapplicable because their cost (LAMEF) is too high for the target. These must be reconsidered each time a target is processed. The flow of the program is illustrated in figure 29.

Basically, the subroutine begins with a do-loop over all weapons currently assigned, to compute the marginal BENEFIT associated with these weapons. At the same time, it tags all the weapons assigned by setting INACTIVE to -100 to avoid any possibility that any weapons currently assigned will be made inactive. It then enters a do-loop over all potential weapons to evaluate the marginal potential BENEFIT associated with these. It skips any weapons already set inactive (INACTIVE = 100,2000,30000) and processes the remainder (INACTIVE = -100,0). Those not currently assigned to the target (INACTIVE $\neq$ - 100), and not showing potential profitability are considered to determine whether they should be set inactive.

If a weapon is not potentially profitable when no weapons are assigned to the target, it is presumed safe to set it inactive (INACTIVE = 30000). If there are other weapons on the target that are unprofitable, the decision to make a weapon inactive is postponed until these are removed. Otherwise, any weapon whose cost is a factor of 10 higher than its potential payoff is made INACTIVE -- it being presumed that even with replacement (or substitution) of weapons by STALL such a weapon is very unlikely to become attractive. The value of INACTIVE, however, is set to 30000 (conditionally INACTIVE) rather than 2000. Ordinarily this is treated exactly the same as INACTIVE = 2000. However, if before exiting from WADOUT it is found necessary to recycle (using a revised value of ALPHA in order to achieve a specified MINKILL), then a flag, REEVAL, is set and those weapons with INACTIVE = 30000 are reconsidered. In some cases, they may turn out to still be applicable because of the increase in effective target value. However, if an exit from WADOUT occurs with INACTIVE $\neq$ 0, it must never be set back to 0 during the remainder of this allocation to the target. Otherwise, incorrect computation would occur -- as a result of trying to do various steps in the computation in WAD despite having omitted earlier steps when the weapon was treated as inactive.

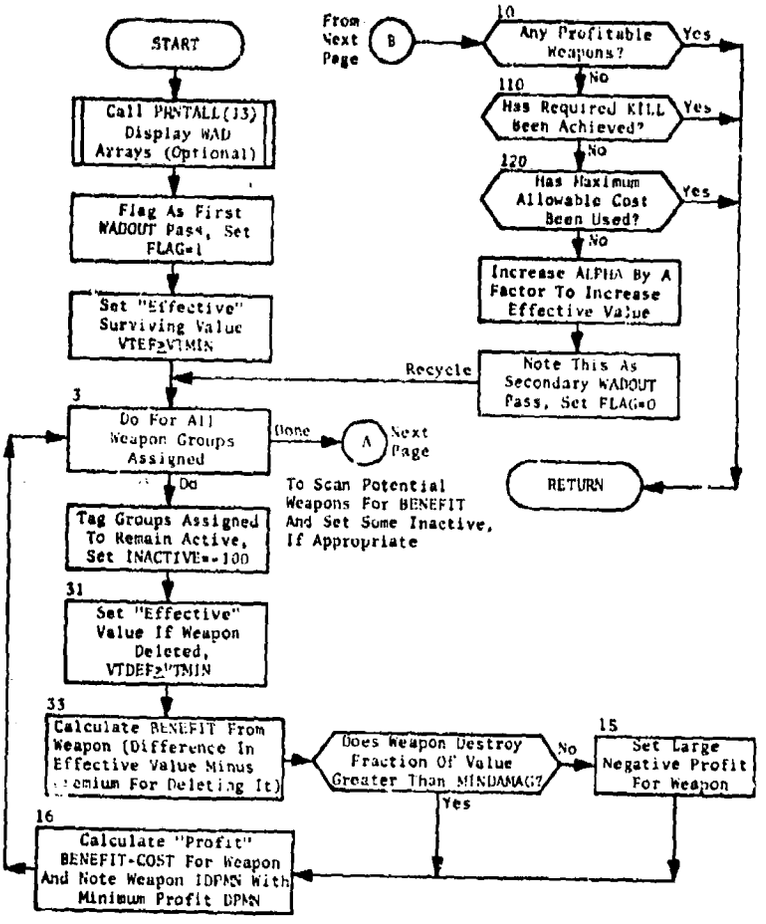


Fig. 29. Subroutine WADOUT
(Sheet 1 of 2)

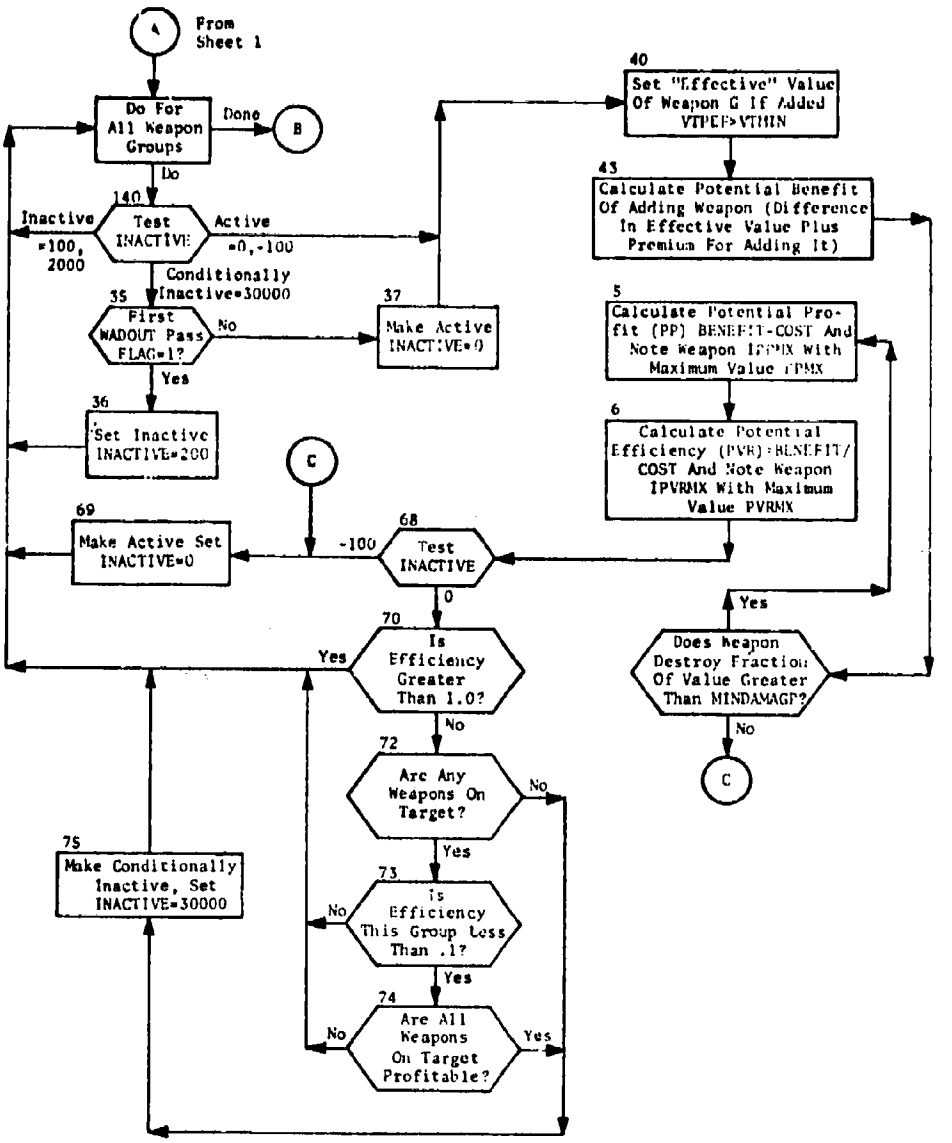


Fig. 29. (cont.)
(Sheet 2 of 2)

Therefore, any weapon group encountered by WADOUT with INACTIVE = 30000 is immediately set to INACTIVE = 2000 unless it is a case of recycling with a new value of ALPHA.

WADOUT, however, has two other subsidiary responsibilities. It is responsible for modifying, when necessary, the six variables transmitted to STALL, so that STALL will not try to exceed the MAXKILL specified for a target, and so that it will continue to add weapons until any specified MINKILL is achieved.

To make it possible for QUICK to match target kill requirements specified for simpler models, where both correlations and time dependence are ignored, a user-input parameter IMATCH is provided in which MINKILL and MAXKILL are interpreted in terms of an oversimplified target kill estimate VTZO.

If the parameter IMATCH is set nonzero by the user, the internal parameters VTMIN and VTMAX are modified. These parameters are defined as:

VTMIN = Original target value * (1 - MAXKILL)

VTMAX = Original target value * (1 - MINKILL).

Their default values are VTMIN = 0, VTMAX = original value.

If the IMATCH parameter is used, VTMIN and VTMAX retain their default values until the 0th order calculation VTZO indicates that MINKILL or MAXKILL have been reached. Then VTMIN or VTMAX is set correspondingly and thereafter operates as usual.

In any case, WADOUT modifies its calculations so that every weapon placed on target destroys at least that percentage of original target value specified by the user-input parameter MINDAMAG.

Subroutine PREMIUMS: The purpose of this subroutine is to compute the payoff premium required by STALL to avoid unduly large deviations from the desired allocation rate. During the closing phase, the premium is also used to cause STALL to close in to an allocation that exactly meets the stockpile constraints. PREMIUMS is called with one parameter which specifies for which group a new evaluation of the premiums (PREMIUM(G) and DPREMIUM(G)) is desired. The call results in the replacement of the old value of these premiums by a new value.

The operation of this routine is a straightforward application of the premium formulae discussed in the Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques, (see Weapon Allocation, Closing Factors - Premiums).

Subroutine DEFALOC and Subroutine RESVAL: The operational concept for these routines is discussed in the Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques, (see Weapon Allocation, Single Target Allocation - Targets With Terminal Ballistic Missile Defenses).

COMMON BLOCK DEFINITION

External Common Blocks

The common blocks used by program ALOC in processing input/output (I/O) files are shown in table 14.

Internal Common Blocks

Table 15 describes the additional common blocks used internally by program ALOC.

Table 14. Program ALOC External Common Blocks
(Sheet 1 of 6)

INPUT FROM BASEFILE

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
MASTER	IHDATE	Date of run initiation
	IDENTNO	Run identification number
	ISIDE	Attacking side
	NRTPT	Number of route points
	NCORR	Number of penetration corridors
	NDPEN	Number of depenetration corridors
	NRECOVER	Number of recovery bases
	NREF	Number of directed refuel areas
	NBNDRY	Number of boundary points
	NREG	Number of command and control regions
	NTYPE	Number of weapon types
	NGROUP	Number of weapon groups
	NTOTBASE	Total number of bases
	NPAYLOAD	Number of payload types
	NASMTYPE	Number of ASM types
	NWHDTYPE	Number of warhead types
	NTANKBAS	Number of tanker bases
	NCOMPLEX	Number of complex targets
	NCLASS	Number of weapon classes (2)
	NALERT	Number of alert conditions (2)
	NTGTS	Number of targets
	NCORTYPE	Number of penetration corridor types
	NCNTRY	Number of distinct country codes
FILES	TGTFILE(2)**	Target data file
	BASFILE(2)	Data base information file
	MSLTIME(2)	Fixed missile timing file
	ALOCTAR(2)	Weapon allocation by targets file
	TMPALOC(2)	Temporary allocation file
	ALOCGRP(2)	Allocation by group file
	STRKFIL(2)	Strike file

*Parenthetical values indicate array dimensions. All other elements are single word variables.

**In two-word arrays, first word is filehandler buffer usage number; second word is maximum file length in words. Single variables are logical tape unit numbers.

Table 14. (cont.)
(Sheet 2 of 6)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
FILES (cont.)	EVENTAPE *	Simulator events tape
	PLANTAPE *	Detailed plans tape
CORRCHAR	PCLAT(30)	Latitude of corridor point
	PCLONG(30)	Longitude of corridor point
	PCZONE(30)	Defense zone in which corridor origin is located
	ZPLAT(30)	Latitude of corridor origin
	ZPLONG(30)	Longitude of corridor origin
	ENTLAT(30)	Latitude of corridor entry
	ENTLONG(30)	Longitude of corridor entry
	CRLENGTH(30)	Distance from corridor entry to corridor origin
	KORSTYLE(30)	Power of y versus x **
	ATTCORR(30)	High altitude attrition per nautical mile unsuppressed
	ATTRSUPF(30)	High altitude attrition per nautical mile suppressed
	HILOATTR(30)	Ratio low to high altitude attrition (less than 1)
	DEFRANGE(30)	Characteristic range of corridor defense (nautical miles)
	NPRCRDEF(30)	Number of attrition sections this corridor
	DEFDIST(30,3)	Distance of precorridor leg
	ATTRPRE(30,3)	Attrition in this precorridor leg
	NDATA	Number of words in common /CORRCHAR/
	LMAX	Maximum number of precorridor legs
PAYLOAD	NOBOMB1(40)	Number of type-1 bombs
	IWHD1(40)	Type-1 warhead index
	NOBOMB2(40)	Number of type-2 bombs
	IWHD2(40)	Type-2 warhead index
	NASM(40)	Number of ASMs
	IASM(40)	ASM index
	NCM(40)	Number of countermeasures
	NDECOYS(40)	Number of terminal decoys
	NADECOYS(40)	Number of area decoys
	IMIRV(40)	MIRV system identification number
	MPAYLOAD	Maximum number of payload types

* These files are output on magnetic tape.

** See program POSTALOC.

Table 14. (cont.)
(Sheet 3 of 6)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
REF*	RFLAT(20)	Refuel point latitude
	RFLONG(20)	Refuel point longitude
PLANTYPE	INITSTRK	Indicator for first or second strike
	CORMSL	Coordination time parameter for missiles
	CORBOMB	Coordination distance for bombers
WPNREG	CCREL(20)	Command and control reliability by command and control region.
WPNTYPE**	RANGE(80)	Range of vehicle (nautical miles)
	CEP(80)	CEP (nautical miles)
	SPEED(80)	Speed (knots)
	ALERTDLY(80)	Alert delay } (hours)
	NALRTDLY(80)	Nonalert delay } (hours)
	RANGEDEC(80)	Low/high altitude fuel consumption ratio
	ICLASS(80)	Weapon class
	RANGEREF(80)	Refueled range (nautical miles)
	REL(80)	Reliability
	IRECMODE(80)	Recovery mode
	IPENMODE(80)	Penetration mode
WPNGRP**	NWPNS(200	Number of weapons
	WLAT(200)	Centroid latitude
	WLONG(200)	Centroid longitude
	IREG(200)	Command and control region
	ITYPE(200)	Type index (LTYPE)
	IALERT(200)	Alert status
	SBL(200)	Probability of survival before launch
	IREFUEL(200)	Indicates refuel code for bombers or payload index for missiles
	YIELD(200)	Weapon yield (megatons)
	REFTIME(200)	Refuel time
	EXPASM(200)	Fraction of weapons in group that are ASMs

*From BASFILE block DPENREF
**From BASFILE block WPNDATA

Table 14. (cont.)
(Sheet 4 of 6)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PKNAVAL*	PKNAV(200)	Single shot kill probability against naval targets
CTRYCD	CTRYCD(150)	List of distinct country location codes on defending side

INPUT FROM TGTFILE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
ALOCIN	TGTNAMZ	Target name
	INDEXNZ	Target index number
	DESIGZ	Target designator code
	TASKZ	Target task code
	CNTRYLCZ	Target country location code
	FLAGZ	Target flag code
	TGTMULZ	Target multiplicity
	TGTLAZ	Target latitude
	TGTLONZ	Target longitude
	TGTRAZ	Target radius (nautical miles)
	VTZ	Original target value
	MZ	Number of hardness components
	HZ(2)	Lethal radius (1MT) by hardness component (nautical miles)
	VOZ(2)	Value by hardness component
	NKZ	Number of time components
	FVAZ(3)	Time component value by time component
	TAZ(3)	Time component time by time component
	IHCLASZ	Target class name
	ICLASSZ	Target class number
	IHTYPZ	Target type name (for complex targets, this is replaced by the number of components in the complex)
	TARDEZ	Local bomber defense level
	MISDEZ	Number of terminal ballistic missile defense interceptors
	MINKILZ	Minimum kill probability required

*From BASFILE block NAVAL

Table 14. (cont.)
(Sheet 5 of 6)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
ALOCIN (cont.)	MAXKILZ	Maximum kill probability desired
	MAXCOSZ	Maximum (weapon cost/target value) acceptable to get MINKILZ
	INDYPEZ	Depenetration corridor index
	DISTDFZ	Distance target to end of depenetration corridor (nautical miles)
	DISTDGZ	Distance target to recovery base (nautical miles)
	DISTCD(30)	Distance corridor origin to target by penetration corridor (nautical miles)
	ATTRCD(30)	Attrition corridor origin to target by penetration corridor
	NFIXES	Number of fixed assignments for this target
	LTG	Length of common block /ALOCIN/

OUTPUT ON ALOCTAR FILE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DYNAMIC	TGTNAME	Hollerith target name
	INDEXNO	Index number of target
	DESIG	Target designator code
	TASK	Target task code
	CNTRYLOC	Target country location code
	FLAG	Target flag code
	TGTMULT	Target multiplicity (original)
	TGTLAT	Target latitude
	TGTLONG	Target longitude
	TGTRAD	Target radius
	VTO	Original target value
	M	Number of hardness components (≤ 2)
	H(2)	Hardness of each component
	VO(2)	Original value of each component
	NK	Number of time periods (≤ 3)
	FVAL(3)	Fraction value escaping in each period
	TAU(3)	Time ending each period

Table 14. (cont.)
(Sheet 6 of 6)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DYNAMIC (cont.)	IHCLASS	Hollerith target class name
	ICLASSN	Target class number
	IHTYPE	Hollerith target type name
	TARDEF	Local bomber defense factor
	INDYPEN	Depenetration corridor index
	DISTDF	Distance from target to end of depenetration
	DISTDG	Distance from target to recovery base
	NBLN	= number of terminal ballistic missile interceptors if a STALL allocation = minus the number of interceptors if a DEFALOC allocation
	CTMULT	Current target multiplicity
	VT	Value remaining after allocation of weapons
	TGTWT(3)	Target weighting values
	PAYOFF	Payoff on this target (VTO-VT-COST)
	COST	Sum of Lagrange multipliers of all weapons allocated to target
	PROFIT	PAYOFF - COST
	DPROFIT	Difference in profit between passes
	WRTEST	Test value for weight rates
	IHEOT	End of information marker
	NUMFIX	Number of weapons allocated by fixed assignment capability
	ITGT	Target number
	NUM	Number of weapons assigned
	IG(30)	Group number of assigned weapons
	KORR(30)	Weapon penetration corridor
	RVAL(30)	Relative value of weapon allocation
	PEN(30)	Weapon penetration probability
	TOARR(30)	Weapon time of arrival on target
	LDN	Number of words from TGTNAME to NUM (inclusive)

Table 15. Program ALOC Internal Common Blocks
(Sheet 1 of 9)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
CONTROL	STALADJ	Determines weight given to profit vs. efficiency in initial allocation
	CLOSE	Controls initial size of step premium
	WADOP	Determines next operation by WAD
	PROGRESS	Measures progress of allocation
	QUALITY	Determines maximum number of cycles of STALL refinement
	NPASS	Keeps count of number of passes allocator makes over targets
	PRM	Controls initial size of linear premium
	DELTVAL	Maximum discrepancy in target value for time-of-arrival bins
	CORR	Controls the size of correlations
	STIME	Not used
	IVERIFY	Verification indicator (0 for no verification, 1 for verification, 2 for test with CORR2)
	CORR2	Correlation used with IVERIFY=2
	IMATCH	If IMATCH ≠ 0, achieve MINKILL for target designating minimum destruction ignoring value-time dependence
	MINDAMAG	Minimum fraction of target value to be destroyed by each weapon
	LAW(2)	Hollerith name of damage law used
	FACMIRV	Correlation array (SMAT) factor used for MIRV systems
	TARFAC	Multiplier for terminal bomber defense levels
CTRYRST	CTRYOK(150,200)	Logical array indexed by (I,J) where I is index of country location code, J is group number. Element (I,J) is true only if weapons from group J can be allocated against targets whose country code is in the Ith position of the country code list, CTRYCD.

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 15. (cont.)
(Sheet 2 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DEFENSE	NTX(3)	Estimates of terminal interceptors present
	PX(3)	Probability that NTX occurs
	PKTX	Terminal interceptor kill probability against unhardened warhead
	RX(2)	NTX/MISDEF for upper and lower deviations from MISDEF
	PRX(2)	Probability there are RX*MISDEF interceptors
	RADPX	Probability of warhead kill by random area defense
	TINTFAC	Multiplier for terminal interceptor defense levels
FIXED	IFTGT	Target number of next fixed target
	ISTORE(8)	Temporary storage area
	IFW(31)	Group numbers of weapons fixed this target
	TIME(30)	Specified arrival time of fixed missiles
	NFIXWPS	Total number of fixed missiles
	SX(5)	Scratch array
	IFIXBEG, IFIXEND	Pointers to input data for fixed weapons
	SAVFIX	Logical variable set true only if fixed missile assignments are to be output on MSLTIME file
	IADD	If negative -- no fixed assignments from TGTFILE If zero -- fixed assignments according to TGTFILE If positive -- fixed assignments from TGTFILE and card input
FIXEDASS	IFIXTEMP(10)	Temporary storage area for fixed assignment data
	IFIXTAPE	Logical unit number of file containing fixed assignment data
	IBCD	Logical unit number of BCD tape containing fixed assignment data
	NLFTAR	Number of fixed assignments for this target on TGTFILE that have not yet been processed

Table 15. (cont.)
(Sheet 3 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
FLAGRST	FLAGOK(9,200)	Logical array indexed by (I,J) where I is flag code, J is group number. Element (I,J) is true only if weapons from group J can be allocated to targets with flag code I.
FORMATS	INWORD	Input word value
	NFORMAT	Hollerith code for appropriate 10-column format for input word
LAMBDA	LAMEF(200)	Value of Lagrange multiplier for each weapon group
	SURPWP(200)	Estimated surplus for weapon group
	PREMIUM(200)	Premium for using weapon group
	DPREMIUM(200)	Premium for deleting weapon group
LOCDEF	VTDX	Surviving target value
	NOWEP(200)	Number of weapons from group
	RATM	Maximum missile rate of return
LOCFIL	WPNTGT	Current WPNTGT file in use
	ALOCT1	Filehandler buffer utilization number for ALOCT1 file
	ITMPMSL	Buffer number for temporary MSLTIME file (unordered)
MACHINE	IREAD	Logical unit number-standard input
	IWRIT	Logical unit number-standard output
	ICOMM	Logical unit number-output comment medium
	IPUNCH	Logical unit number-standard punch
MULADJ	NINTPRD	Number of separate integration periods used
	RINTPRD	Controls ratio between integration periods
	RATIOINT	Ratio of controlling integration period to that implied in allocation rates

Table 15. (cont.)
(Sheet 4 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
MULADJ (cont.)	SNSTVTY	Rate of multiplier convergence relative to the significance of errors
	FSNSTVTY	Final rate of convergence relative to number of targets
	SETTLE	Number of cycles for multipliers to settle to closing
	BPENFAC	Multiplier for bomber attrition rates
NALLY	NALL(200)	Number of weapons allocated this pass from each group
	RNALL(200)	Number of weapons now on all targets from each group
PAYOFF	OPROFIT	Profit for hold allocation (last pass) evaluated with present values LAMEF
	SPAYOFF	Cumulative payoff all targets
	SUMCOST	Cumulative cost all targets
	SPROFIT	Cumulative profit all targets
	SUMPREM	Cumulative premiums all targets
	TBENEFIT	Total benefit this target
	NAT	Number of attribute categories (currently 6)
PEN	PENALT(30)	Temporary storage for penetration probabilities for each corridor
PRINEED	N	Weapon group index
	G	Weapon group index
PRINI	STALPRIN	Print indicator showing location in STALL processing
PRNTCON	IFRSTPR	Initialization trigger for subroutine PRNTCON
	IDO(40)	Contains optional print flags (set to 3 if print active, 1 for inactive)
	INDEXPR(40)	Requested option
	JPASS(40)	First pass to activate request
	JTGTP(40)	First target to activate request

Table 15. (cont.)
(Sheet 5 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PRNTCON (cont.)	LPASS(40)	Last pass to activate request
	LTGT(40)	Last target to activate request
	KTGTFREQ(40)	Print frequency
	ICOUNT(40)	Number of targets processed since last print
	MYPRT(40)	Request setting mode (DEFAULT or INPUT)
	MAXREQ	Maximum number of print requests
	MPRNT	Maximum number of print options
	NREQ	Actual number of print requests
PRNTWADD	IG1	Internal indexes used in optional prints from subroutine WAD
	J1	
	N1	
	XMU	
	S1	
	VSN1	
	IG2	
	J2	
	N2	
	NW2	
PRTMULL	NTATTRIB	Total number of local multipliers
	PROCMULT	Target multiplicity now processed this target
	ERRCLOS	Factor which controls the termination of the allocation
	CLOSER	Controls increase in closing force per pass
	DELTEFF	Increase in profit from last pass divided by VALWPNS
	SDELTEFF	Cumulative DELTEFF
	VALWPNS	Value of all weapons
	VALERR	Value of surplus and deficit weapons combined
RANGE	RANGEMUL(200)	Multiplier for unrefueled range for group
	RANREMUL(200)	Multiplier for refueled range for group
	RANGEMIN(200)	Minimum range (nautical miles) for group

Table 15. (cont.)
(Sheet 6 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
RESTRICT	MYNMIRV	Number of MIRV systems in list
	MYMIRV(40)	IMIRV system number
	NPERMIT(40)	Number of permitted classes
	MYPERMIT(40,16)	Hollerith names of permitted classes
	TBMDEF(40)	Logical array--true if defended targets are permitted
	MULTARG(40)	Logical array--true if multiple targets are permitted
	MAXMIRV	Maximum number of MIRV systems
	MAXNP	Maximum number of permitted classes
SMAT	SMAT(6,5)	Interweapon correlation information
	LSMAT	Length of SMAT array
	SMATMIRV(3)	Storage for SMAT values which change for MIRV systems
	SMNOMIRV(3)	Storage for original values for non-MIRV systems
TABLE	TABLE(1001)	Kill factors for use in square root damage function
WADFINAL	VTP(200)	Total expected undestroyed target value with weapon added from each group
	DELVT(30)	Change in residual target value by weapon allocation
	NUMO	Number of weapons assigned this target on old target
	IGO(30)	Index of weapons assigned on old pass
	IOP	Total number of add and delete operations this target
	IOPS	Grand total number of add and delete operations
	CTSPILL	Number of multiple target elements postponed for later processing

Table 15. (cont.)
(Sheet 7 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
WADOUT	PVRMX	Maximum efficiency for weapon IPVRMX
	IPVRMX	Weapon group index
	PPMX	Maximum profit for weapon IPPMX
	IPPMX	Weapon group index
	DVRMN	Not used
	IDVRMN	Not used
	DPMN	Minimum profit
	IDPMN	Estimated minimum profit
	NUMMAX	Maximum number of weapons allowed per target
	NW	Number of weapons now on target
	TPMX	Largest profit or potential profit so far
	NTOA	Number of time arrival bins used
	NTOAMAX	Maximum number of time arrival bins allowed
	VTMIN	Destruction of target below VTMIN considered of no value
	VTMAX	Maximum acceptable surviving target value
	ALPHA	Value factor required to justify VTMAX
	VTEF	Maximum of either VT or VTMIN
WADWPN*	JTGT	Static target number
	INACTIVE(200)	Nonzero value flags inactive groups
	TOA(200)	Time of arrival at target of weapons from each group
	TVALTOA(200)	Unattrited target value at TOA for weapons in each group
	VTOA(200,2)	Unattrited value for each component at TOA for weapons in group
	MUP(200,2)	Contribution of weapon to mean if added
	RISK(6,200,2)	SQRTF(2*SUMMODE(GAMAMODE* LN(MODEK)/LN(TK)))

*This block is used as an input buffer in subroutines RDALCRD, LOCREST, FLAGRST, MIRVRST, and RNGEMOD. The alternate definition of items in this block follows the original definition.

Table 15. (cont.)
(Sheet 8 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
WADWPN (cont.)	SSIG(200,2)	Square root of -log of s
	MINKILL	Minimum acceptable kill probability
	MAXKILL	Maximum desired kill probability
	MAXCOST	Maximum acceptable cost to achieve MINKILL (weapon cost/target value)
	ILAW	Kill probability law indicator
	MISDEF	Number of terminal ballistic missile interceptors
	MORR(200)	Preferred penetration corridor
	PEX(200)	Penetration probability
	XMUP(200,2)	Single shot survival probability--used only for targets with ballistic missile defenses
	JTGTX*	Static target number for next target
	LNEXT	Number of words in WPNTGT record
	MINKILX	MINKILL for next target
	MAXKILX	MAXKILL for next target
	MAXCOSX	MAXCOST for next target
	NACTV	Number of active groups on next target
	IGX(200)	Group numbers of active groups
	TOAX(200)	TOA for active groups
	MORRX(200)	MORR for active groups
	PEXX(200)	PEX for active groups
	STKX(200,2), STK2X(200,2)	Intermediate interaction calculation variables for active groups
	LSTMAX	Length of WPNTGT I/O buffer
WADWPN (as redefined in RDALCRD, LOCREST, FLAGRST, MIRVRST, RNGEMOD.)	INPUT(10)	Input card storage array
	NVARS	Number of parameters on input card
	NAMES(40)	Parameter names
	INVALU(2,40)	Parameter input values
	INDEX1(40), INDEX2(40), INDEX3(40)	Array indexes for parameters
	MORE	Input termination indicator
	MYNAME(100)	Default parameter names
	MYFORM(100)	Default parameter output format

* This area, from JTGTX to the end of the block, is used as an I/O buffer for the WPNTGT files.

Table 15. (cont.)
(Sheet 9 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
WADWPN (cont.)	MYTYPE(100)	Default parameter setting mode (equivalent to FVAL)
	MYVAL(100)	Default parameter - value
	MYGOTO(100)	Default parameter - index to variable type
	NDEFLT	Number of default parameters
	FILLER(5560)	Dummy array
222	WTFAC(3)	Divide into old target weights for commensurability
	WTRATE(3)	Rate of increase of target weights
	WTSUM(3)	Sum of target weights
	JATTRIB(6,200)	Index to local multipliers for weapon group
	RUNSUM(310,3)	Running sum of target weight times weapons allocated for local multiplier
	ALLEREST(310,3)	Estimate of allocation error
	LA(310)	Value of local multiplier
	MXATTRIB	Maximum number of local attributes (Lagrange multipliers)
333	NWP(10)	Number of weapons in TOA set
	VAL(10)	Unattrited target value at TOA for each set
	V(11,2)	Unattrited component at TOA for each set N and each hardness component
	MU(10,2)	Sum of means through each TOA set
	SIG(10,2)	Sum of variants through each TOA set
	S(10,2)	Component survival probability through each TOA set
	VS(10,2)	= (V(N,JH)-V(N+1, JH))*S(N,JH)
	VSN(11,2)	= VSN(N-1, JH)+VS(N-1, JH)
	ITOA(200)	TOA index for weapons in group if added
	IADDTOA(200)	Set to 1 if new TOA set is required
	SIGP(200,10,2)	Increase in variance for each TOA set N if weapon is added from group
	SIGP(30,10,2)	Change in variance for each TOA set if weapon deleted
	DSIG(200,2)	Variance contribution for weapon vs. specified weapon

PROGRAM ALOC

PURPOSE:	This program processes the user-input command cards and calls the appropriate subprograms to perform the requested operation.
ENTRY POINTS:	ALOC
FORMAL PARAMETERS:	None
COMMON BLOCKS:	WPNGRP, WPNTYPE, WPNREG, REF, PAYLOAD, FILES, FILABEL, MYLABEL, ITP, MYIDENT, NOPRINT, TWORD, LOCFIL, MASTER, IFTPRNT, CTRYCD, MACHINE, CORRCHAR, PLANTYPE, PKNAVAL, 222
SUBROUTINES CALLED:	STORAGE, INITALC, TIMEME, SKIP, RNGEMOD, MINRNGE, MIRVRST, FLAGRST, LOCREST, READMUL, TIMEPRT, MULCON, PUNCHM, INITAPE, SETREAD, RDARRAY, TERMTAPE, DEACTIV, ABORT
CALLED BY:	This is main program; entered by SCOPE System

Method

This main program processes the user-input command cards and controls the initialization routines. It begins by calling STORAGE to print out available core space (see figure 30). The variable NCNTRY is initialized so that the DO loop in INITALC which clears /CTRYCD/ will operate over the correct range. Subroutine INITALC is called to initialize program variables. A call on TIMEME with a parameter of -1 initializes the timing clock. Subroutine INITAPE is called to initialize the filehandler. Subroutine SETREAD is called to initialize reading on the BASFILE. INFORM (block /FILABEL/) is tested to determine if the BASFILE is of the correct format. If not the run aborts with an error message (statement 10). If the format is correct, the external common blocks MASTER, FILES, CORRCHAR, PAYLOAD, REF, PLANTYPE, WPNREG, WPNTYPE, WPNGRP, PKNAVAL, and CTRYCD are filled from the BASFILE (statements 20 and following). Extraneous portions of the BASFILE are skipped over by subroutine SKIP. The BASFILE is terminated and the time to read the file is recorded by a call on TIMEME.

The remainder of the subprogram reads and processes the user-input command cards. Each card is read at statement 100. A series of 18 statements determines the requested option and calls the appropriate

subroutines. The calls on TIMEME record the time spent in each option which precedes weapon allocation.

When the STOP command is encountered, a completion message is output to the printer and console typewriter. Two calls on subroutine DEACTIV are made to remove TINFILE and WINFILE from the list of files to be spilled, as they are no longer needed.

A description of each command and its effect follows. The first six commands may appear in any order, but must precede the ALLOCATE command. The use of any command except ALLOCATE and STOP is optional.

1. RANGEMOD: This command will cause a call to subroutine RNGEMOD. The data for the new refueled and unrefueled weapon ranges will be read, one group per card. These new ranges will remain in effect only for the allocation process. Later processors will use the ranges specified in the data base.

2. MINRANGE: This command will cause a call to subroutine MINRNGE. The data cards will contain the minimum range for each group. There will be one card for each group the user wishes to restrict. Weapons from a group will be allocated to a target only if the distance to be traversed by the weapon exceeds the minimum range. If no minimum range is specified for a group, a zero value will be assumed.

3. MIRVREST: This command will initiate processing by subroutine MIRVRST. The data cards will contain the numbers of permitted target classes for each MIRV system. There may be several cards for each MIRV system. If a MIRV system is restricted, weapons using that system can be allocated only against targets of that class. In addition, the user may specify complex targets, multiple targets, and targets with terminal ballistic missile defenses as permitted target types. If no restrictions are made on a system, it can be allocated to any target.

4. FLAGREST: This command will initiate processing by subroutine FLAGRST. The data cards will allow the restriction of weapon allocations from each group according to the value of the FLAG attribute. There will be one data card for each group the user wishes to restrict.

5. LOCREST: This command will initiate processing by subroutine LOCREST. Its function is much the same as for the previous function except that weapon groups will be restricted according to country location code.

6. READMUL: This command will cause the program to read initial values for the Lagrange multipliers.

7. ALLOCATE: This is the major function command for program ALOC. It must be present in every run and must follow any of the previous six commands that are present. This command will initiate the weapon allocation process. The data cards will contain print requests and user-input parameters.

8. PUNCH: This optional command, which will follow the input parameter set associated with the ALLOCATE command, will cause the punching of cards containing the final Lagrange multipliers used in the allocation process. These cards may be used by the command READMUL to set initial multipliers in a later run.

9. STOP: This command will stop processing in a normal manner.

10. DUMP: This command will terminate processing with a memory dump.

Program ALOC is illustrated in figure 30.

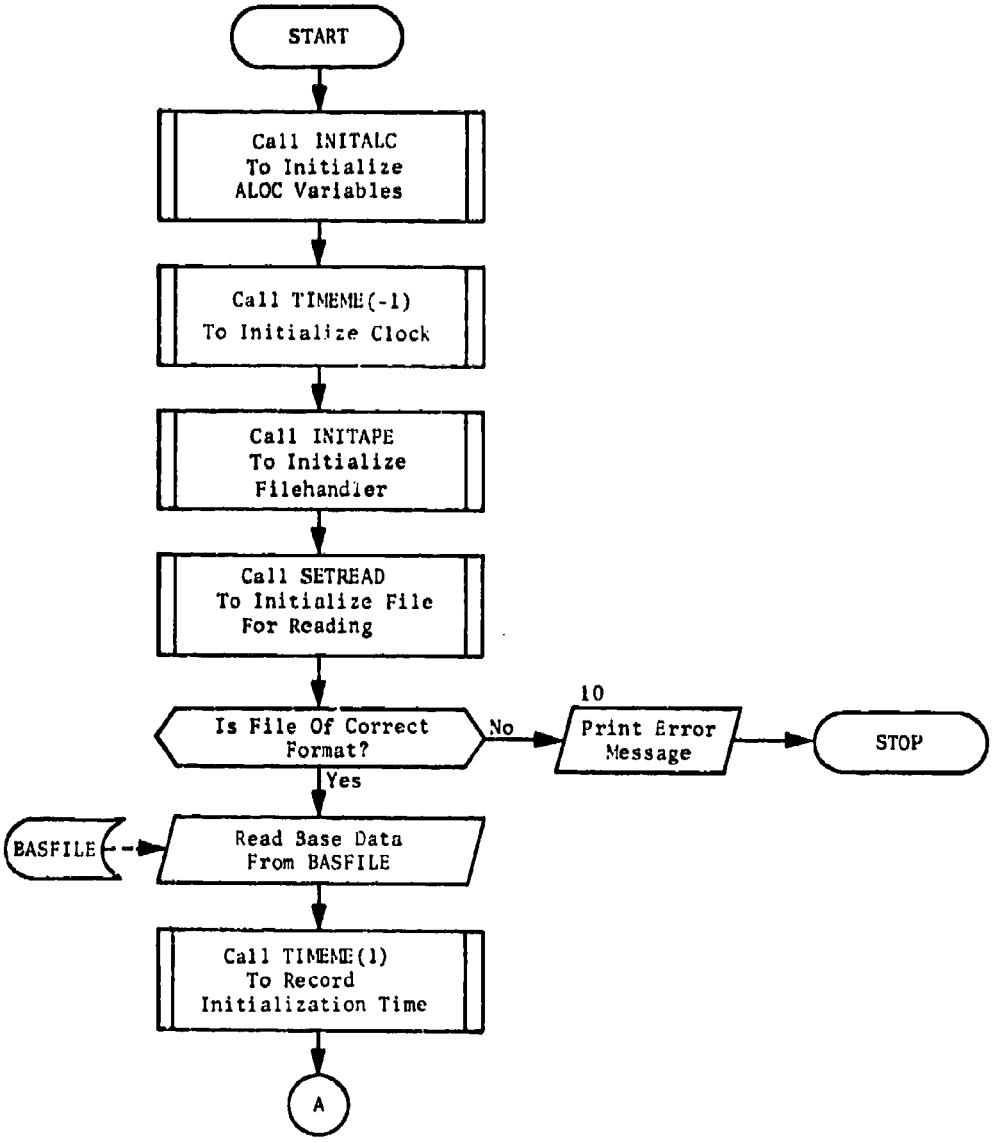


Fig. 30. Program ALOC
(Sheet 1 of 2)

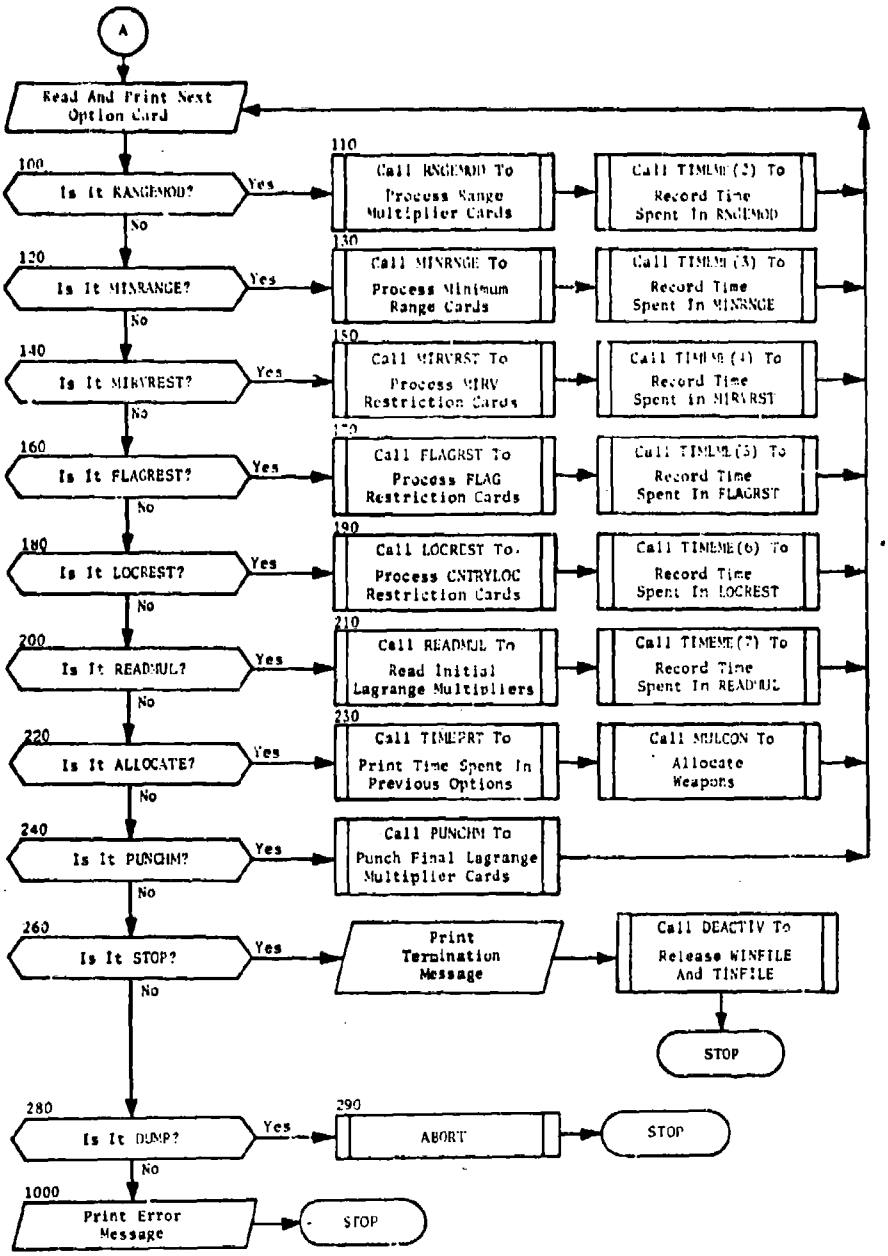


Fig. 30. (cont.)
(Sheet 2 of 2)

SUBROUTINE DEFALOC

PURPOSE:	The purpose of this subroutine is to allocate missiles to an individual target which is defended with terminal ballistic missile interceptors (i.e., MISDEF >0).
ENTRY POINTS:	DEFALOC
FORMAL PARAMETERS:	None
COMMON BLOCKS:	WPNGRP, DEFENSE, WADWPN, LAMBDA, WPNTYPE, WPNREG, MASTER, DYNAMIC, PAYLOAD, LOCDEF, FIXED, FILES, LOCFIL, WADOUT, WADFINAL, FIXEDASS, Filehandler (FILABEL, MYLABEL, ITP, MYIDENT, NOPRINT, TWORD)
SUBROUTINES CALLED:	PREMIUMS, RESVAL, NUMGET, PRNTALL, Filehandler (RDWORD, RDARRAY, WRARRAY)
CALLED BY:	MULCON

Method

When MULCON has read in data associated with a new target, it examines MISDEF to determine if the target is defended with terminal ballistic missile interceptors. If MISDEF = 0, indicating no defenses, it proceeds to call STALL for the allocation of weapons. If MISDEF >0, then there are terminal interceptors present; DEFALOC is called after calling STALL to allocate the missiles to the target, and the most profitable allocation (STALL or DEFALOC) is chosen.

The input variables describing the target's local ABM capability allow uncertainties to be introduced in the number of interceptors present. MISDEF is the nominal number of interceptors on the target, each with kill probability PKTX against unhardened warheads, and RADPK is the random area defense kill probability. In addition, four other parameters are defined (the same for all targets) which introduce uncertainties in MISDEF. RX(1) (input as LOWFAC) is a factor which, when multiplied by MISDEF, gives a lower estimate of interceptors which has probability PX(1) (input as PROBLOW) of occurring. Likewise, RX(2) (input as HIGHFAC) and PX(2) (input as PROBHIGH) define the overestimate of interceptor availability. Thus, if there is imperfect knowledge of the terminal ABM

capability, the allocator can hedge against these uncertainties when assigning weapons.

In addition to the target-associated ABM data, it is possible to describe penetration aids suitable for the various missiles by means of the payload table. For a particular payload index IPAY, the following variables are defined which describe the terminal missile defense penetration aids:

NWHD = Number of warheads per independent re-entry vehicle.

NTDECOYS = The number of aim points the terminal defense sees for each independent re-entry vehicle (in addition to the warheads). These are terminal decoys.

XDEG = A factor by which the PKTX is multiplied to obtain terminal interceptor kill probability against this weapon type. It reflects additional hardening of the warhead or electronic penetration aids which can degrade interceptor effectiveness.

The first decision in DEFALOC concerns the verification pass. If PROGRESS = 2 and IVERIFY = 2, the current call on DEFALOC is a part of verification pass to determine the effect of a new correlation factor. Since DEFALOC does not consider interweapon correlations, no processing is done in this case.

Before allocating any missiles, DEFALOC determines if STALL has been called. STALL will not be called if the number of fixed assignments exceeds 30. If STALL was called, the surplus weapon indicators SURPWP are reset (statement 76) as if STALL were not called. This procedure provides for the correct premium computations in RESVAL.

If there are fixed assignments on this target (statement 77), DEFALOC assigns these weapons first. On the second and later passes (statement 78) the operation is relatively simple. The old allocation contained in the IG and KORR arrays of /DYNAMIC/ is loaded into the IFW array and the NOWEP array. The NOWEP array defines the potential DEFALOC allocation. If the number of fixed assignments exceeds 30, the value of the KORR array is equal to the negative of the number of weapons assigned (see statements 72 to 96). This procedure allows specification of more than 30 weapons within the 30 elements of the IG array. If the fixed assignments are less than 30 in number, each entry in the IG array represents one weapon.

Processing differs if there are fixed assignments on the first pass. The DO loop starting at statement 97 and continuing to statement 85 allocates each fixed weapon. There are several tests made on proposed fixes in this loop. First, if the fixed weapon is a bomber, DEFALOC

cannot be used since it allocates missiles only. Statement 40 tests if STALL were called. If so, the STALL allocation to this target is used without further DEFALOC processing. If STALL was not called by reason of an excess of 30 fixes, statement 41 prints an error message and the fix request is ignored. If the fixed weapon is a missile, its range is tested in statement 50. If the range is insufficient, an error message is printed and the request ignored. If the range is sufficient the weapon is assigned. A group counter is kept (statement 86) to determine the number of unique groups represented on target. Statement 91 assigns the weapon. Following statement increase the total cost and reset the surplus weapon counter. If the user has requested saving the fixed missile information on the MSLTIME file (SAVEFIX = TRUE), this information is written on the ITMPMSL file by DEFALOC (statement 63) if subroutine STALL has not already done so.

After assigning all fixed weapon requests currently input, DEFALOC determines if there are more requests to be read in the first pass. If NLFTAR (/FIXEDASS/) is greater than zero, more fixes are read from the TGTFILE. The existence of more requests is signalled by setting IFW(31) to a nonblank value. If after reading the remaining fixes from the target file, the value of IFTGT (/FIXED/) matches the target number, ITGT (/DYNAMIC/), more requests are read from the user input. The values of IFIXTAPE and IBCD, set by RDALCRD, determine the source of user input fix requests.

Statement 75 is the exit from the fixed assignment processing. The DO loop from statement 75 to statement 74 places each fixed weapon into the IFW and KORR arrays to be saved for later use. In addition, the expected number of objects (warheads and terminal decoys) perceived by the terminal defense is computed and stored in variable NOBJ.

Before allocating any weapons to the target, DEFALOC calculates an approximate maximum rate of return for attacking the target with the best missiles available, using exhaustion tactics (statement 84).

The missile allocation proceeds as follows: first, those missiles with the cheapest terminal objects (warheads and terminal decoys) are allocated until the terminal interceptors are exhausted. Then each missile type in turn is tried to determine which type has the greatest payoff per unit cost when added to this exhausted mix of weapons.

Since the payoff function for a defended target is generally not convex, one cannot look at only the rate of return of the next missile to determine whether the target is profitable to be attacked. Rather, it is necessary to allocate weapons beyond the exhaustion point and then search for that allocation which yields the highest average rate of return. If this average rate is greater than one; i.e., a profit is realized by attacking the defended target, then the allocation can actually proceed.

The exhaustion phase of the allocation is carried out in the statements between statements 400 and 600. The post-exhaustion phase starts at statement 600 and continues to statement 2000.

In all calculations of target damage, subroutine RESVAL is called to determine the residual target value (VTDX) for the specific mix of weapons allocated at the time RESVAL is called. Appearance of VTDX in the flowchart implies a call on RESVAL. The program, however, can only allocate missiles from a maximum of 30 groups in total, which must be kept in mind when specifying target defenses. That is, if 30 groups cannot provide sufficient objects to exhaust the defense, this tactic is excluded by the allocator. In addition, only 40 percent of the weapons in any one group can be allocated to a single defended target.

When DEFALOC has completed its laydown, it compares the resulting profit to the STALL profit. If the STALL profit is greater, DEFALOC sets NBLN = MISDEF, and restores the STALL allocation. If DEFALOC has a greater profit, it sets NBLN = -MISDEF. If only one of the subroutines has produced an allocation which met the required MINKILL, that allocation is chosen regardless of profit. Then DEFALOC loads the IG and KORR arrays. First, the fixed weapons are placed in the arrays and then the nonfixed ones. In all cases, the KORR array contains a negative number corresponding to the number of missiles allocated from the group specified in the corresponding position of the IG array.

This subroutine also calculates a modified value for deleting a weapon from the target. This value, DELVT in /WADFINAL/, is used by MULCON to compute the relative damage caused by each allocation, RVAL. For fixed weapons, the value of DELVT is set to the original target value, VTO. For all other weapons, DELVT is equal to the difference between the final residual value, VTDX, of the entire allocation and the residual value if all nonfixed weapons of the same group were removed.

Figure 31 illustrates subroutine DEFALOC.

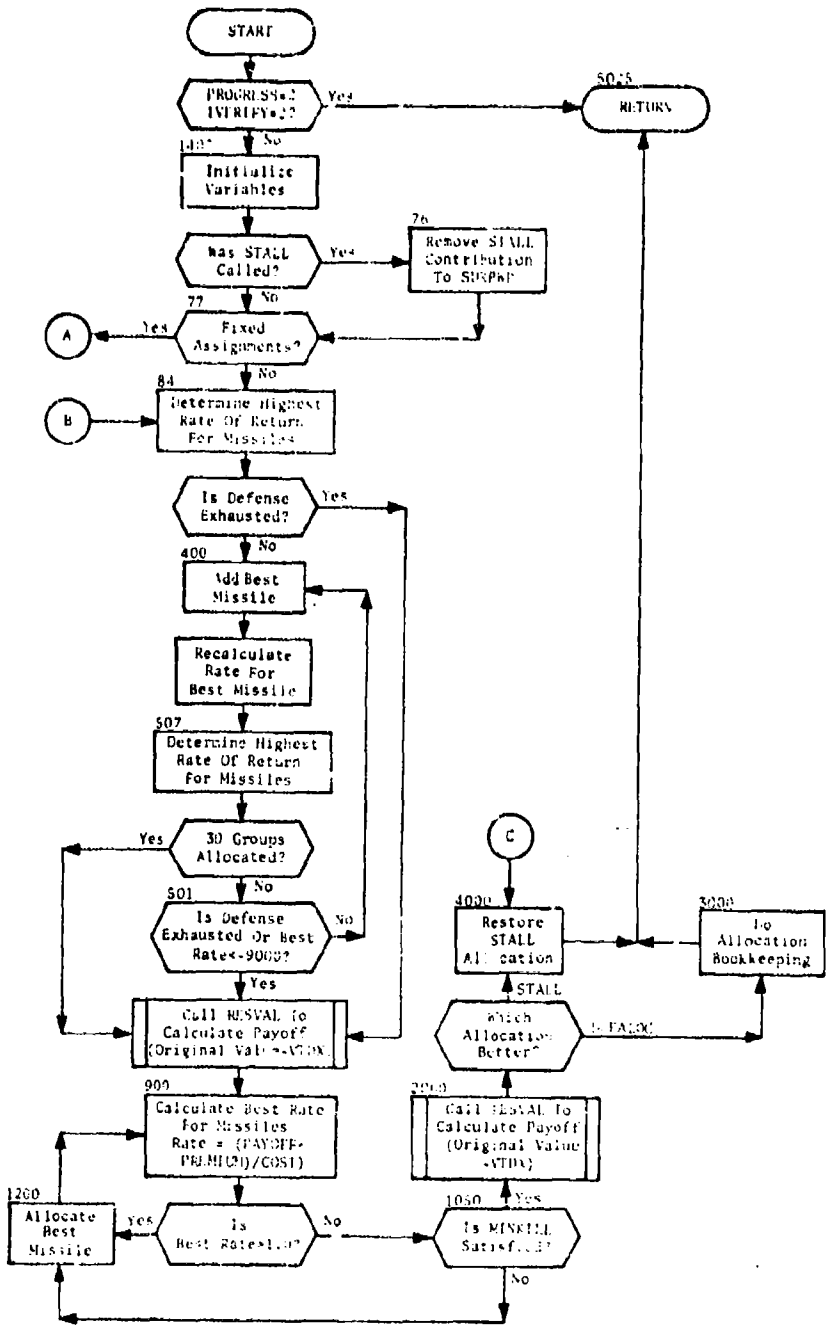


Fig. 31. Subroutine DEFALOC
Part I: Normal Processing

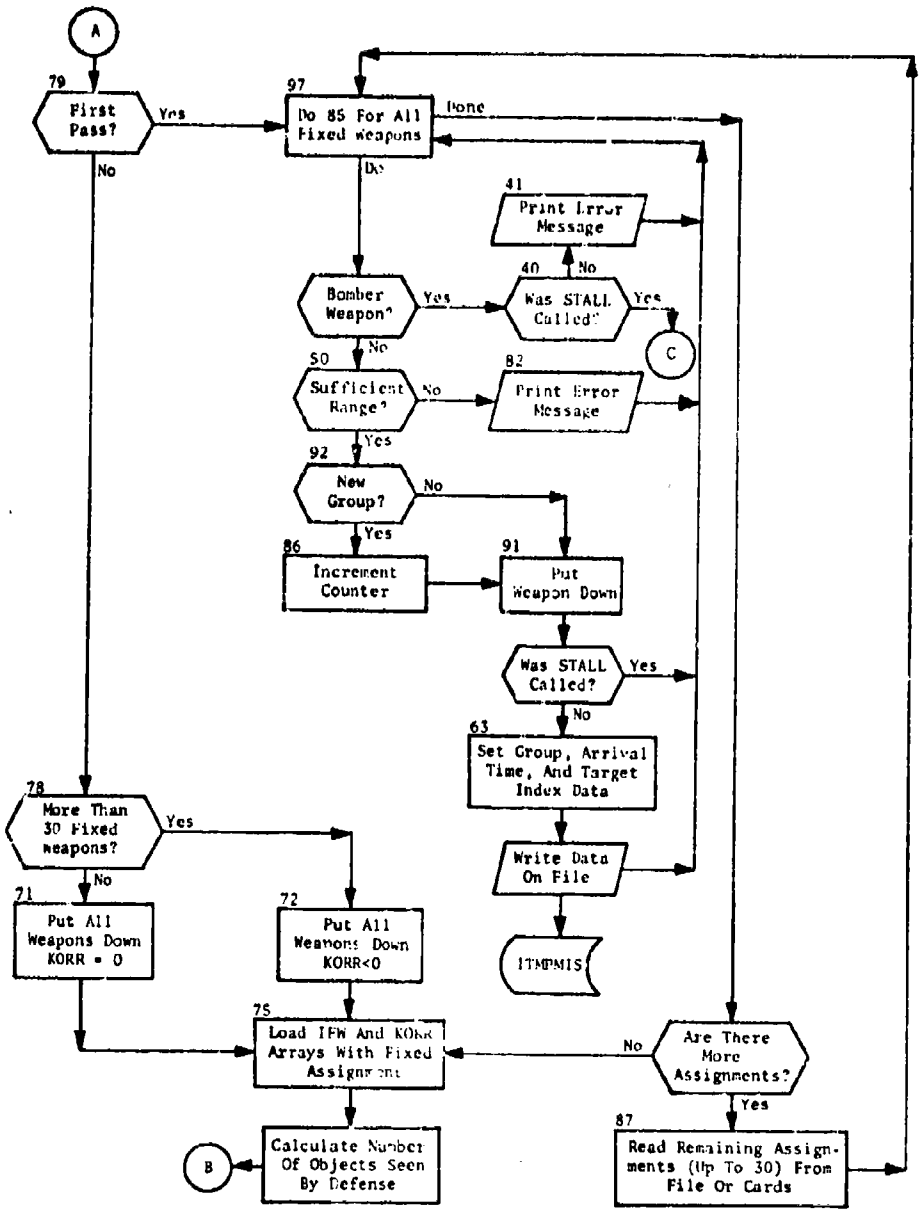


Fig. 31. (cont.)
Part II: Fixed Weapon Assignment Processing

SUBROUTINE FLAGRST

PURPOSE:	This routine processes the flag restriction user-input parameter cards.
ENTRY POINTS:	FLAGRST
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, FLAGRST, MACHINE, WADWPN
SUBROUTINES CALLED:	NUMGET
CALLED BY:	ALOC

Method

This routine performs the flag restriction option operations. Its purpose is to set the flag restriction codes (FLAGOK in common /FLAGRST/ according to user-input parameter cards. (Common block /WADWPN/ is used as an input buffer.)

Array FLAGOK in common /FLAGRST/ is a logical array indexed by flag code (1-9) and group number. The element FLAGOK(I,J) represents the flag I for group number J. If FLAGOK(I,J) is .TRUE., then weapons from group J can be allocated against targets whose flag code is I. If this element is .FALSE. weapons from group J are restricted from targets with code I.

The initial setting of the FLAGOK array is performed in subroutine INITALC, where all values are set .TRUE.. The user has two input options, SELECT and DELETE. In the SELECT option, all elements in FLAGOK for a group are set .FALSE. except for those specified on an input card. In the DELETE option, specific elements are set .FALSE. The local logical variable SET performs this operation. If the user-requested option for a group is DELETE, SET is set .FALSE. (statement 50). If the SELECT option is chosen for a group, the statements from 60 to 80 give SET a value of .TRUE., and set .FALSE. all values of FLAGOK for the requested group.

Statement 80 calls NUMGET to remove the imbedded blanks from the user-input flag codes and convert from BCD code to integer variable. For example, the user-input codes " 1 2 3 4 5" in BCD code would be converted by NUMGET to the number 12,345. A call on the SCOPE library routine

XMODF performs modulo 10 arithmetic on the resulting number. For example, 12,345 modulo 10 is 5. This gives one flag code, the rightmost code. Dividing the initial number (12,345) by 10 removes the rightmost code (12,345/10 = 1,234). Successive applications of modulo 10 arithmetic and divisions by 10 between statements 90 and 110 break down the user input into a list of flag codes. Statement 100 sets the FLAGOK value according to this flag code list.

The subroutine ends by printing the flag codes of all those groups which are restricted in some way (statement 1020).

Subroutine FLAGRST is illustrated in figure 32.

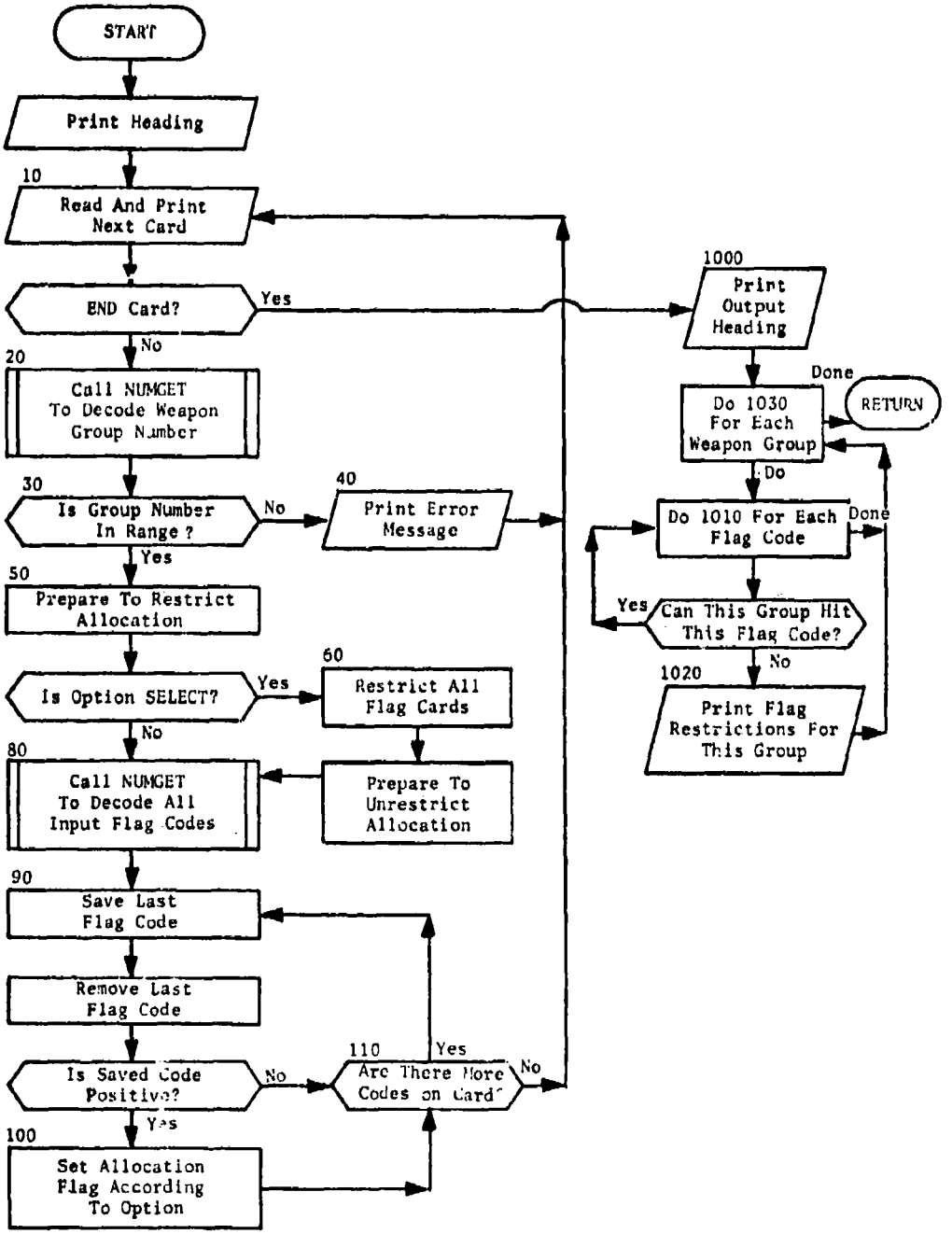


Fig. 32. Subroutine FLAGRST

FUNCTION FMUP

PURPOSE:	This function computes a survival probability given a sum of kill factors for all weapons allocated to the target.
ENTRY POINTS:	FMUP
FORMAL PARAMETERS:	S - A sum of kill factors
COMMON BLOCKS:	WADWPN
SUBROUTINES CALLED:	None
CALLED BY:	WAD, RESVAL

Method

This function is a straightforward application of the two damage laws used in the system. The law used on the current target is determined by the variable ILAW in common /WADWPN/. If this variable was set positive in subroutine RECON (and entry SETUP in that routine), then the square root damage law is used. Otherwise, the exponential law is used.

The input formal parameter S is a sum of the kill factors (each prepared by function TABLEMUP) for all weapons assigned to the target.

The value returned by the function is defined as follows:

Exponential Law

FMUP = exp (-S) (statement 1)

Square Root Law

$$FMUP = (1 + \sqrt{S})*\exp(-\sqrt{S}) \qquad \text{(Statement 2)}$$

Figure 33 illustrates function FMUP.

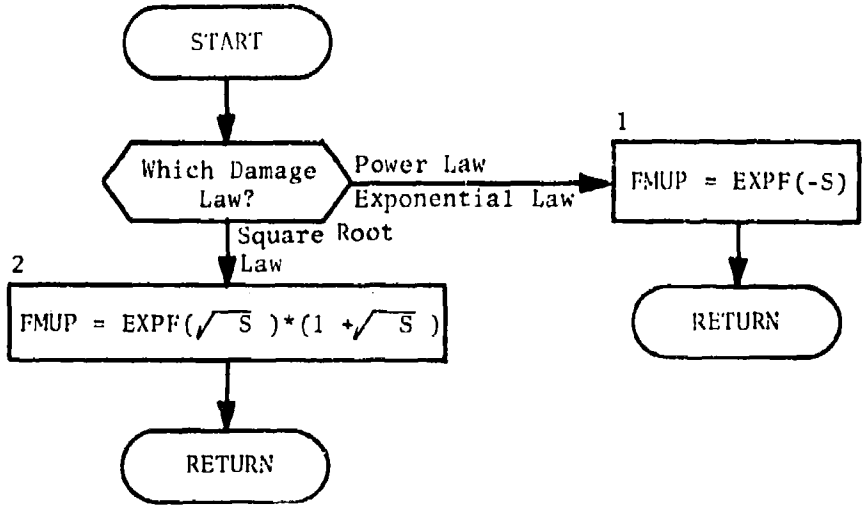


Fig. 33. Function FMUP

SUBROUTINE FORMATS

PURPOSE:	This subroutine determines the best 10-column BCD format for a variable.
ENTRY POINTS:	FORMATS
FORMAL PARAMETERS:	None
COMMON BLOCKS:	FORMATS
SUBROUTINES CALLED:	None
CALLED BY:	PRNTNOW

Method

The variable to be formatted is INWORD, the first word in common /FORMATS/. (This word is equivalenced to WORDIN.) The best 10-column format is returned in NFORMAT the second word of common /FORMATS/. The resulting value of NFORMAT can be used in a FORTRAN ouput statement such as: PRINT NFORMAT, INWORD.

The names WORDIN and INWORD are equivalenced to allow correct specification (real or integer) for any possible input. The name NFORMT is internally equivalenced to the variable name NFORMAT for convenience. Internal to FORMATS, the absolute value of the input variable is kept in INABS, equivalenced to ABSIN for type specification purposes.

To determine if a number is fixed or floating point, the first 18 bits of the absolute value are tested. If a bit is set, the number is assumed to be floating point, since all normalized floating point numbers have at least one bit set in this range. Thus, if an integer quantity greater than 1,073,741,823 is input, it will be treated as if it were a floating point variable. However, no variable input from PRNTNOW can have a value of that magnitude, so this restriction is never a handicap. Table 16 presents the returned formats for each range of input values.

Subroutine FORMATS is illustrated in figure 34.

Table 16: Calculated Formats for Variables

RANGE OF ABSOLUTE VALUE OF VARIABLE, X	FORMAT OF NEGATIVE	FORMAT OF POSITIVE
0 or Integer Variable	I10	I10
$0 < X < 0.0001$	E10.1	E10.2
$0.0001 \leq X \leq 0.999999$	F10.5	F10.5
$0.999999 < X \leq 99.9999$	F10.4	F10.4
$99.9999 < X \leq 9999.99$	F10.3	F10.3
$9999.99 < X \leq 999999.9$	F10.2	F10.2
$999999.9 < X$	E10.1	E10.2

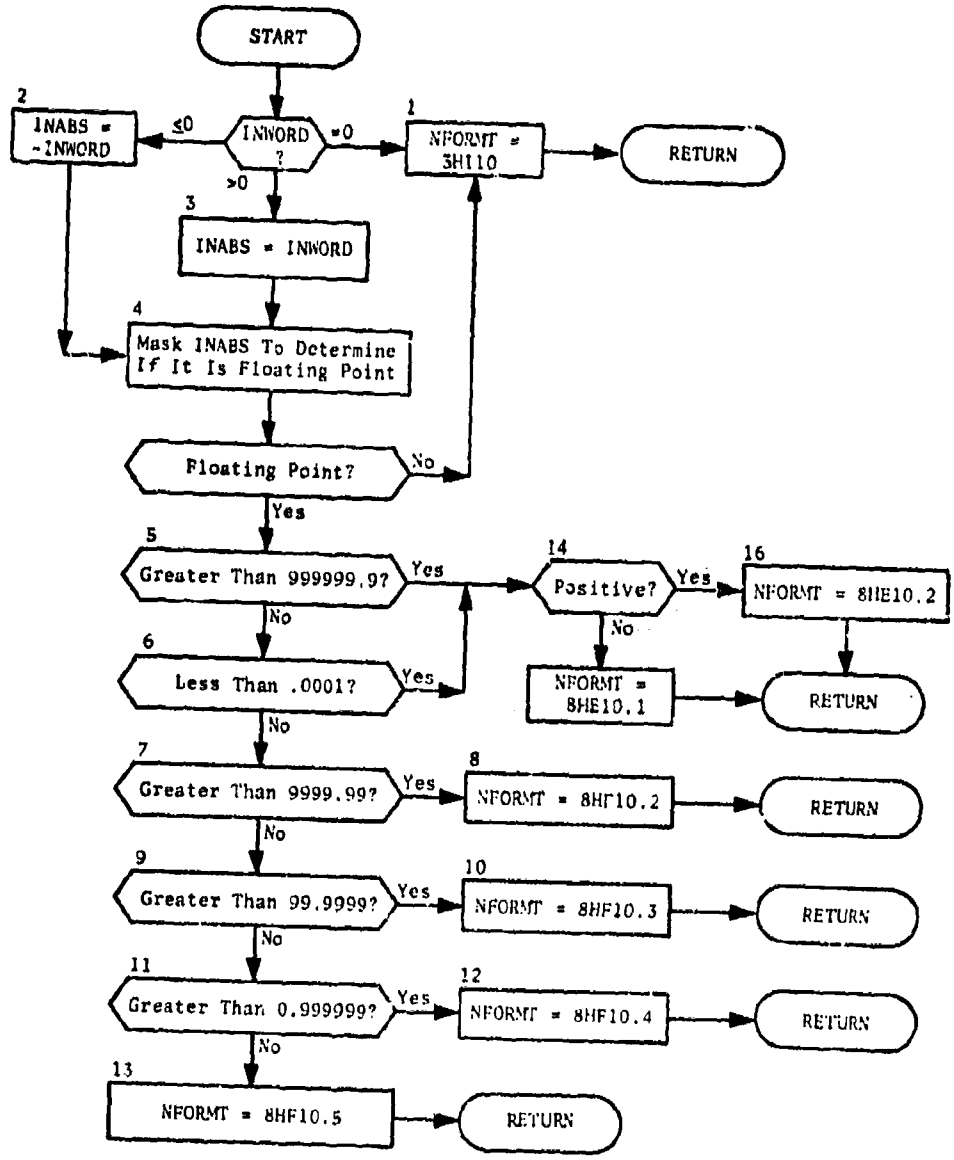


Fig. 34. Subroutine FORMATS

SUBROUTINE GETDATA

PURPOSE:	This routine computes the weapon-target interaction variables and performs all I/O on the WPNTGT files.
ENTRY POINTS:	GETDATA, INITGET
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, FILES, FILABEL, MYLABEL, ITP, MYIDENT, NOPRINT, TWORD, LOCFIL, WPNREG, WPNTYPE, PAYLOAD, WPNGRP, PLANTYPE, REF, CORRCHAR, DYNAMIC, WADWPN, CONTROL, FLAGRST, CTRYCD, CTRYRST, RESTRICT, RANGE, PKNAVAL, FIXED, FIXEDASS, MACHINE, DEFENSE, MULADJ, PRINEED, SMAT, PAYOFF, PEN, ALOCIN, INTFILE, DRC5162
SUBROUTINES CALLED:	TIMEME, RDARRAY, DELLONG, ITLE, DATE, PRNTALL, SETUP, SKIP, RDWORD, NUMGET, ABORT, RECON
CALLED BY:	MULCON

Method

Definitions of local arrays for subroutine GETDATA are:

DISTAD(ICORR)	The distance for a weapon group G from group centroid (or refueling area) to the target via corridor ICORR
ATTRAD(ICORR)	The total high altitude attrition in DISTAD(ICORR)
CROSSDST(ICORR)	The perpendicular distance of the target from the center line of corridor ICORR
PENALT(ICORR)	Penetration probability to the target via corridor ICORR for current group
DPLAT(ICORR)	Latitude of depenetration corridor ICORR
PCLAT(ICORR)	Latitude of penetration corridor ICORR
DPLONG(ICORR)	Longitude of depenetration corridor ICORR
PCLONG(ICORR)	Longitude of penetration corridor ICORR

ATTR(LEG)	Total attrition estimated on (LEG) portion of bomber mission ($1 \leq LEG \leq 4$)
RATE(LEG)	Attrition rate times sortie value estimated on (LEG) portion of mission
DISTT(LEG)	Distance of (LEG) estimated
LASTFT(IF)	Target number of last target on WPNTGT file number IF

The flow of operations for subroutine GETDATA is shown in Parts I through VII of figure 35. The parts of the figure display the Initializing Operations (entry INITGET), Input Filehandling (First Pass), Weapon-Target Calculations, Bomber Penetrability, Output Array Processing, Second Pass Processing, and Fixed Assignment Processing. The flow diagrams are in sufficient detail to be largely self-explanatory. In the following sections, comments are made only where the flow diagrams require some explanation.

Part I: Initialization (Entry INITGET)

Entry INITGET is used to initialize the local variables for GETDATA. The corridor-dependent distances and attrition rates are precomputed for later use. The SMAT arrays for MIRV and nonMIRV systems are computed and the values of attribute RANGEDEC are checked for appropriate range (i.e. ≥ 1.1).

Part II: Input Filehandling - First Pass

This section checks the previous write operation on the WPNTGT file. If there were errors, the write is retried until success or five unsuccessful attempts. In the latter case, the job aborts with an error message. The input data on the target from the TGTFILE are read and transferred to common /DYNAMIC/ from common /ALOCIN/. If GETDATA had to allocate more space for the job as scratch storage, the QUICK filehandler disk directory, common /DRC5162/, is rewritten on the disk to show the new allocations.

Part III: Weapon-Target Calculations

This portion has been described in detail in the Analytical Manual and needs no further comment here. Statements 822 and 7394 apply to the FLAG restriction constraints. Statements 8049 and 1398 apply to the CNTRYLOC restriction constraints. Statement 8051 applies to the minimum range (MINRANGE) constraint. The block preceding statement 822 and statement 970 apply to the restriction of allocations to NAVAL

targets for weapons with PKNAV greater than zero. Statements 909 and 960 apply to the MIRV system restriction (MIRVREST) constraints.

Part IV: Bomber Penetrability

This part calculates the use of low-altitude range and resulting bomber attrition as described in the Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques, Defenses-Bomber. Using a value of recovery equal to one quarter of the total sortie value, the best penetration corridor is chosen.

Part V: Output Array Processing

This section compresses the output arrays in common /WADWPN/. All storage reserved for inactive groups is removed. If the target has only one hardness component, storage in the STK and STK2 arrays for the second component is also removed.

This section also allocates new disk files for the WPNTGT files, if they are required. Variable NOWALOC in common /INTFILE/ gives the number of scratch files currently allocated by the QUICK filehandler. These files are named SCRTCH1, SCRTCH2, SCRTCH 3, etc. Since the ALOCT1 file is SCRTCH1, and the ITMPMSL file is SCRTCH2, the first WPNTGT file must be SCRTCH3. Thus, as new WPNTGT files are needed, NOWALOC is investigated to determine if the space has been allocated. If not, GETDATA uses operating system routine ALLOCATE to reserve one-million words (LINTFL in /INTFILE/) for the WPNTGT file. If GETDATA does allocate extra files, it must change NOWALOC in common /INTFILE/ and rewrite the QUICK filehandler disk directory DRC5162 (see part II).

Part VI: Second-Pass Processing

This part is used on all passes after the first. It reads the data for the next target and checks for read errors. If they exist, the data are reread until there are no errors or there have been three unsuccessful attempts. In the latter case, the job aborts with an error message. The three types of read errors are parity errors, end-of-file errors, or length errors. For a length error, the amount of data read must be less than the amount expected according to LNEXT in /WADWPN/. Subroutine RECON is used to decompress the file data into the form used by the other routines in ALOC. Subroutine RECON moves the data from the lower half of /WADWPN/ to the upper half.

Part VII: Fixed Assignment Processing

This part is used only if fixed assignments are allowed from the TGTFILE and there are fixes for the current target on that file. (This part is used only in the first pass.) If there are more than 30 fixes on the target, the flag NLFTAR in common /FIXEDASS/ is set to the number of fixes remaining on the file for input by subroutine DEFALOC.

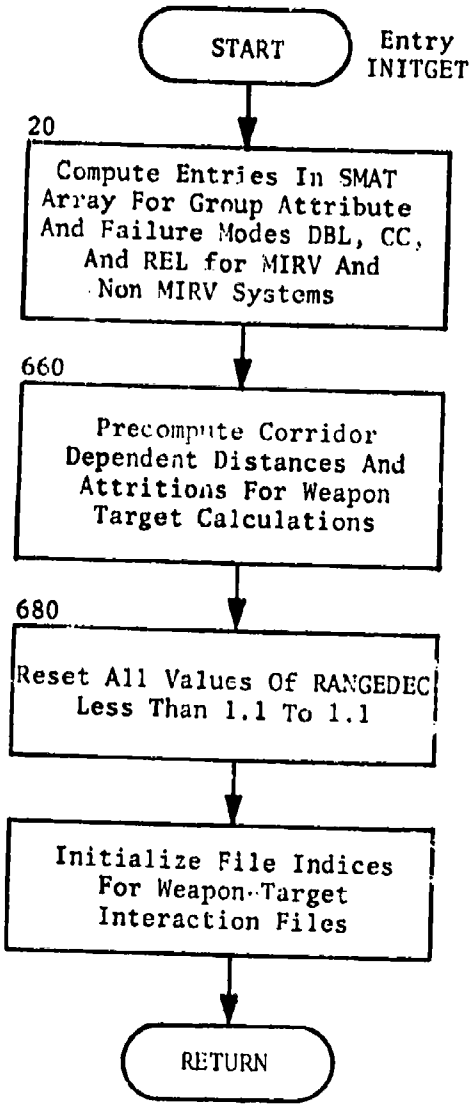


Fig. 35. Subroutine GETDATA
Part I: Initialization,
Entry INITGET

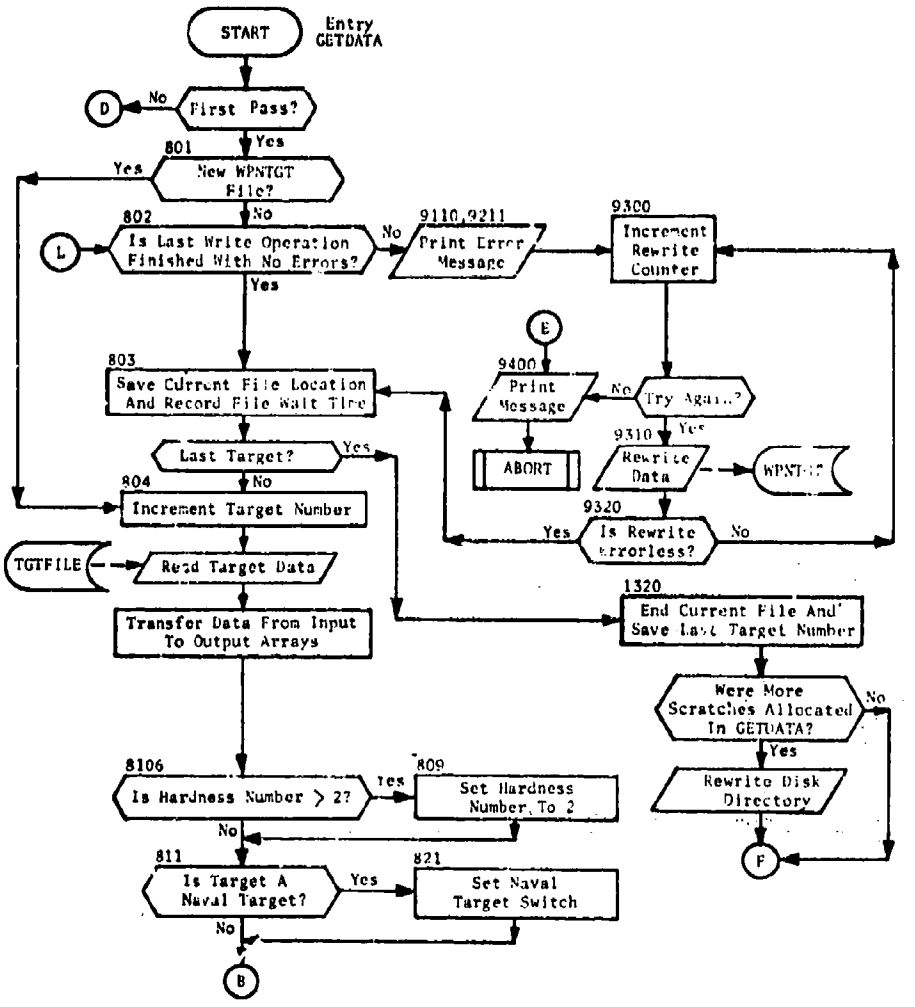


Fig. 35. (cont.)
Part II: Input Filehandling, First Pass

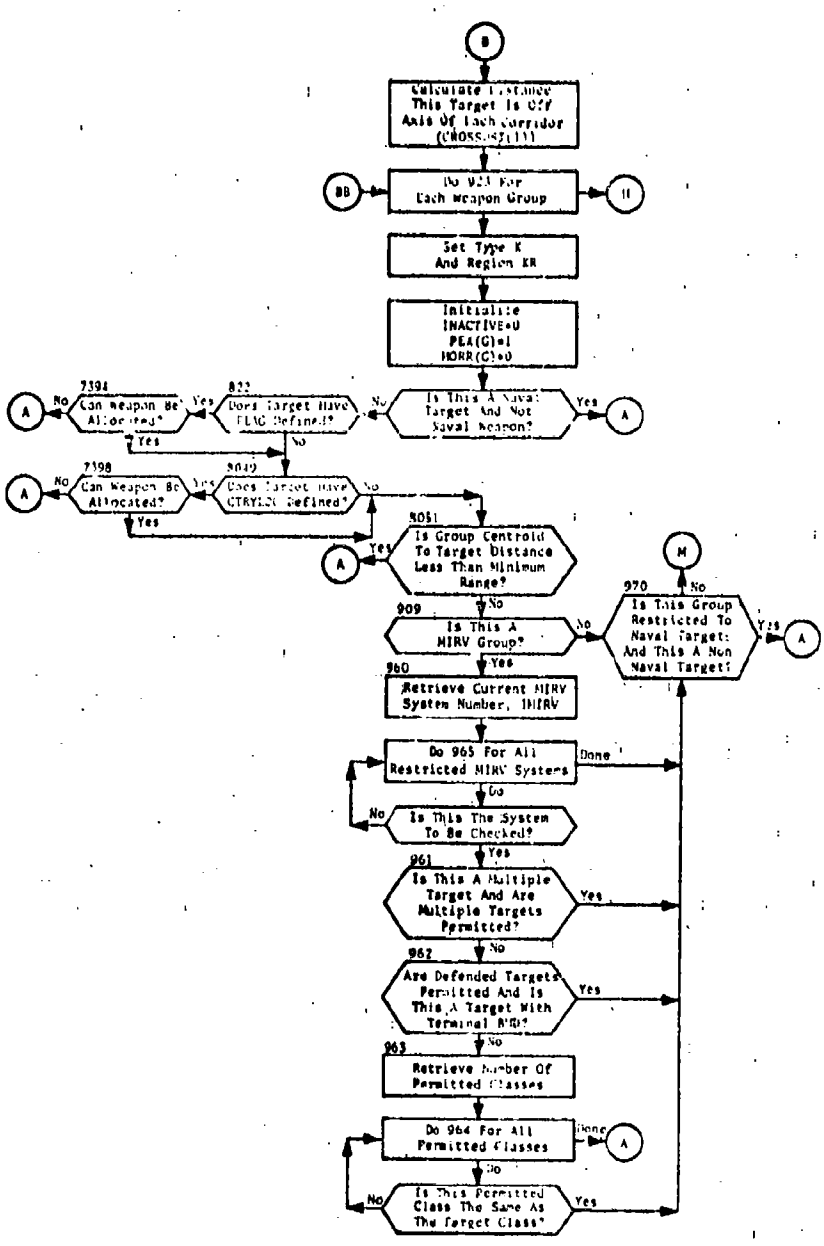


Fig. 35. (cont.)
Part III: Weapon-Target Calculations
(Sheet 1 of 3)

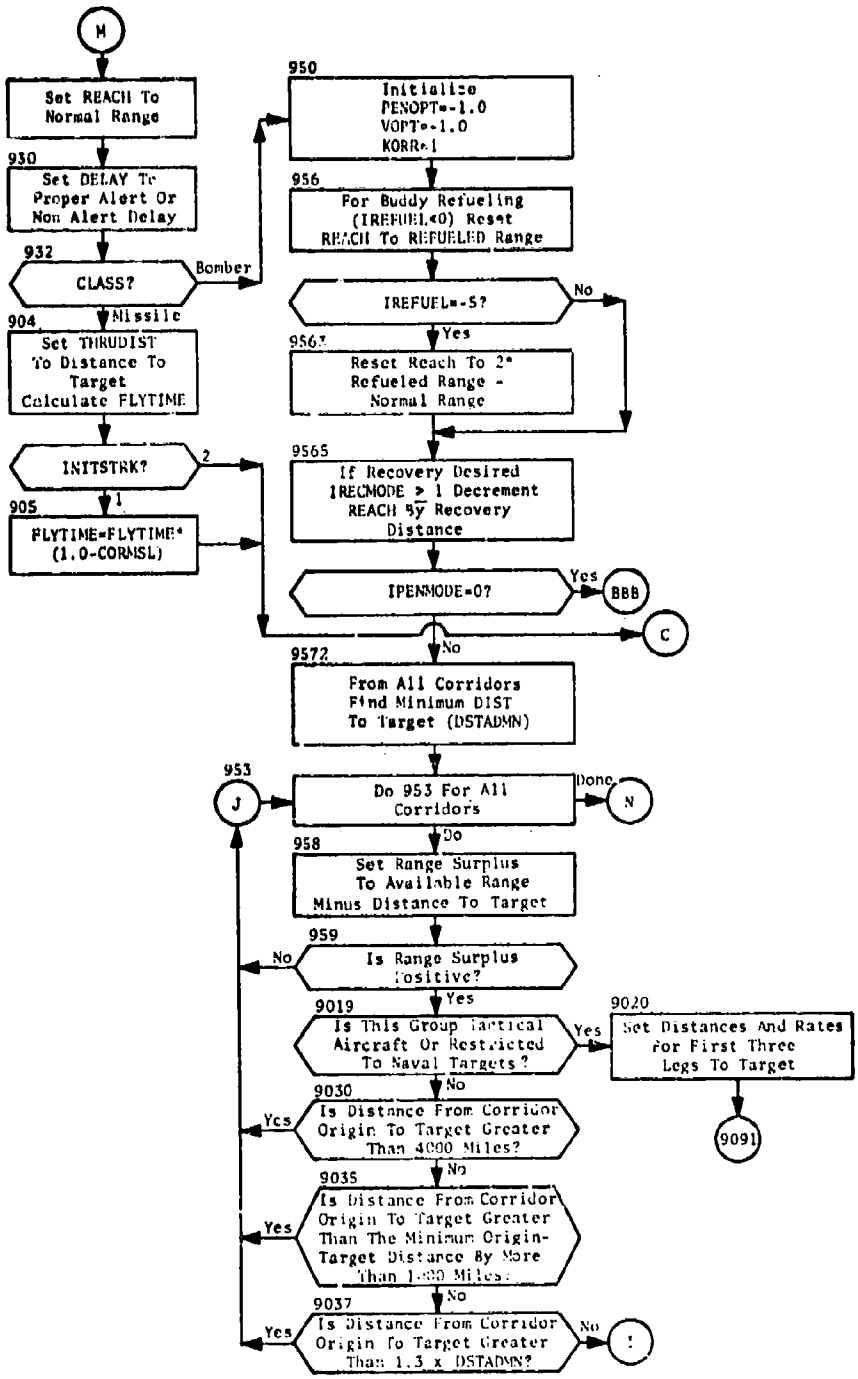


Fig. 35. (cont.)
Part III: (cont.)
(Sheet 2 of 3)

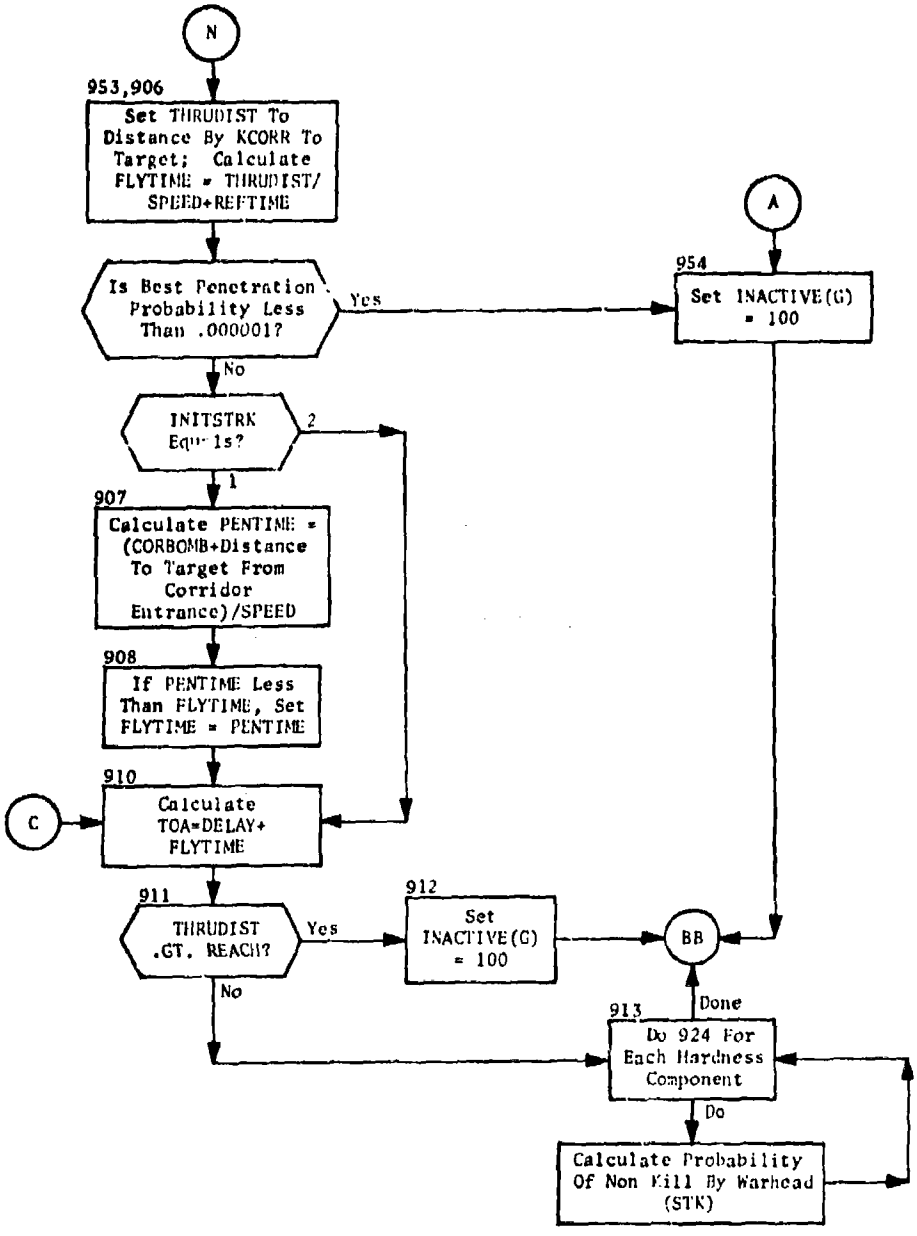


Fig. 35. (cont.)
Part III: (cont.)
(Sheet 3 of 3)

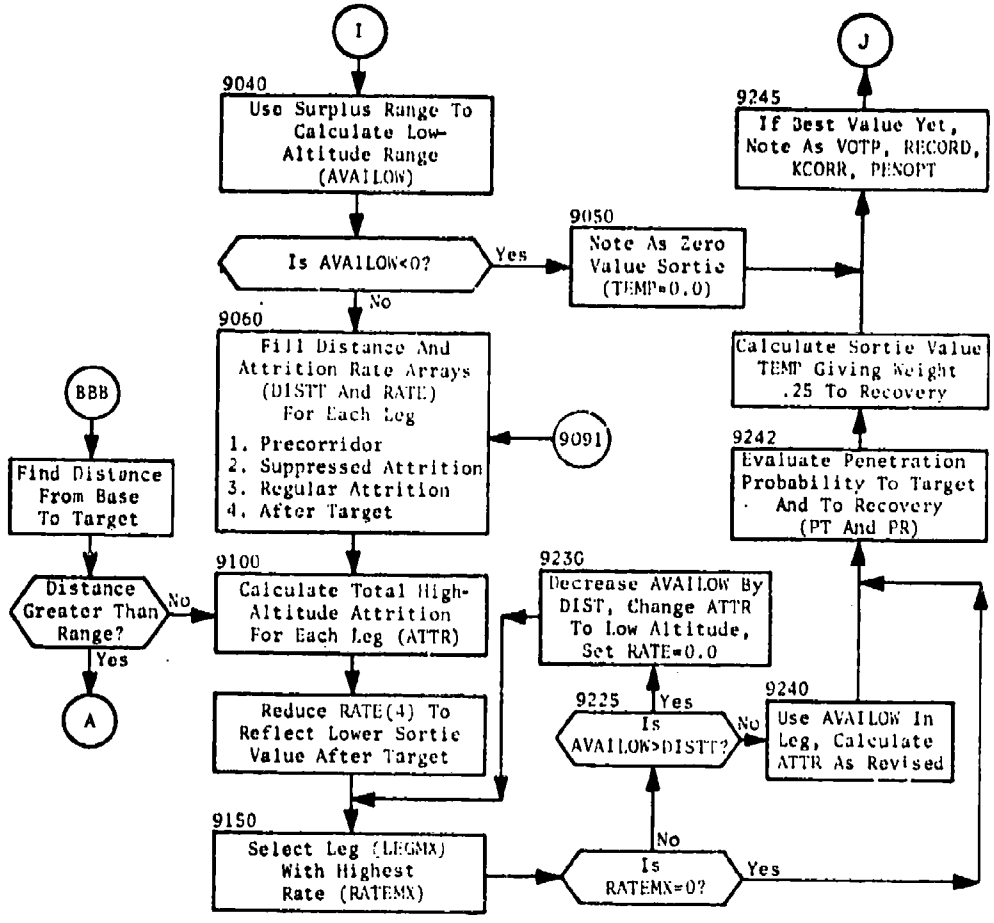


Fig. 35. (cont.)
Part IV: Bomber Penetrability

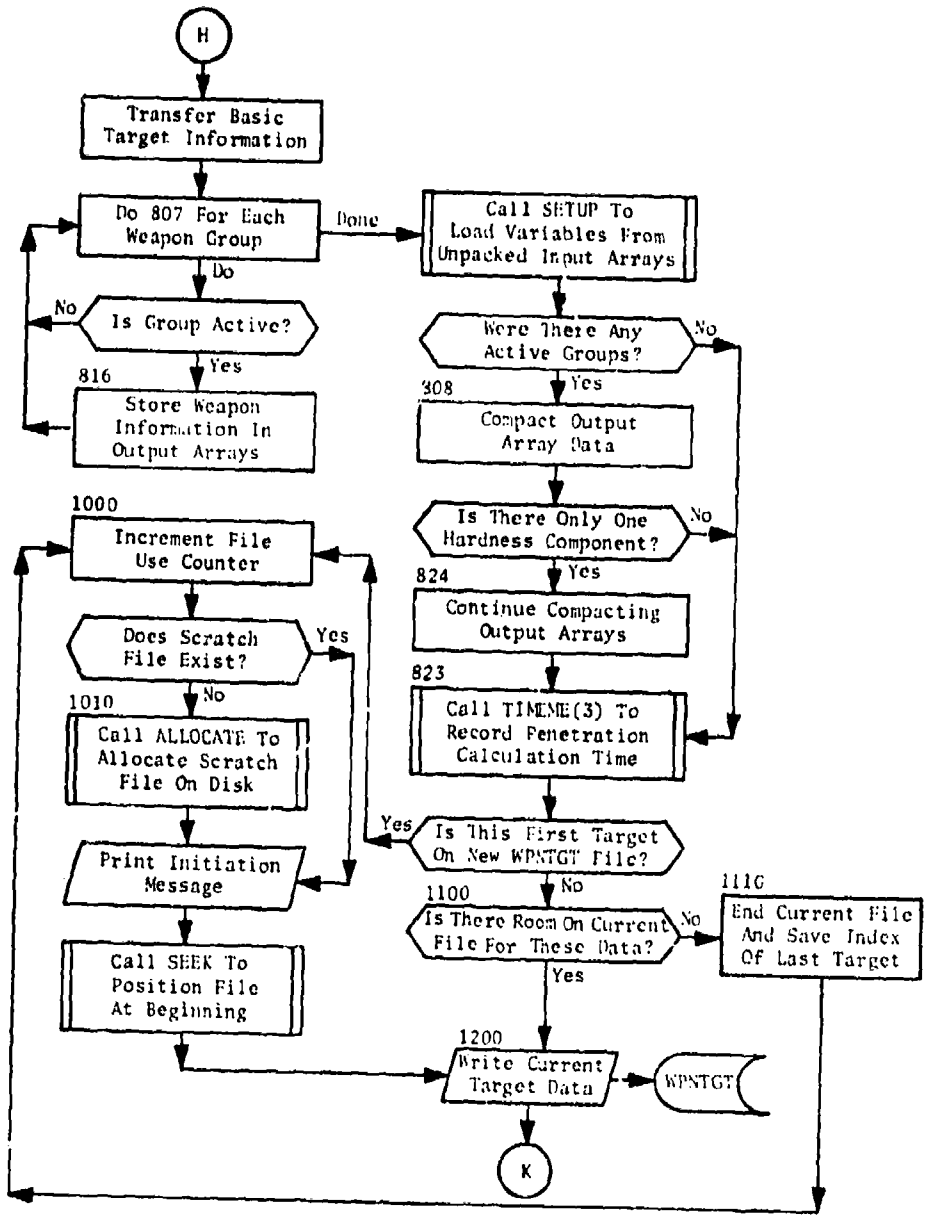


Fig. 35. (cont.)
Part V: Output Array Processing

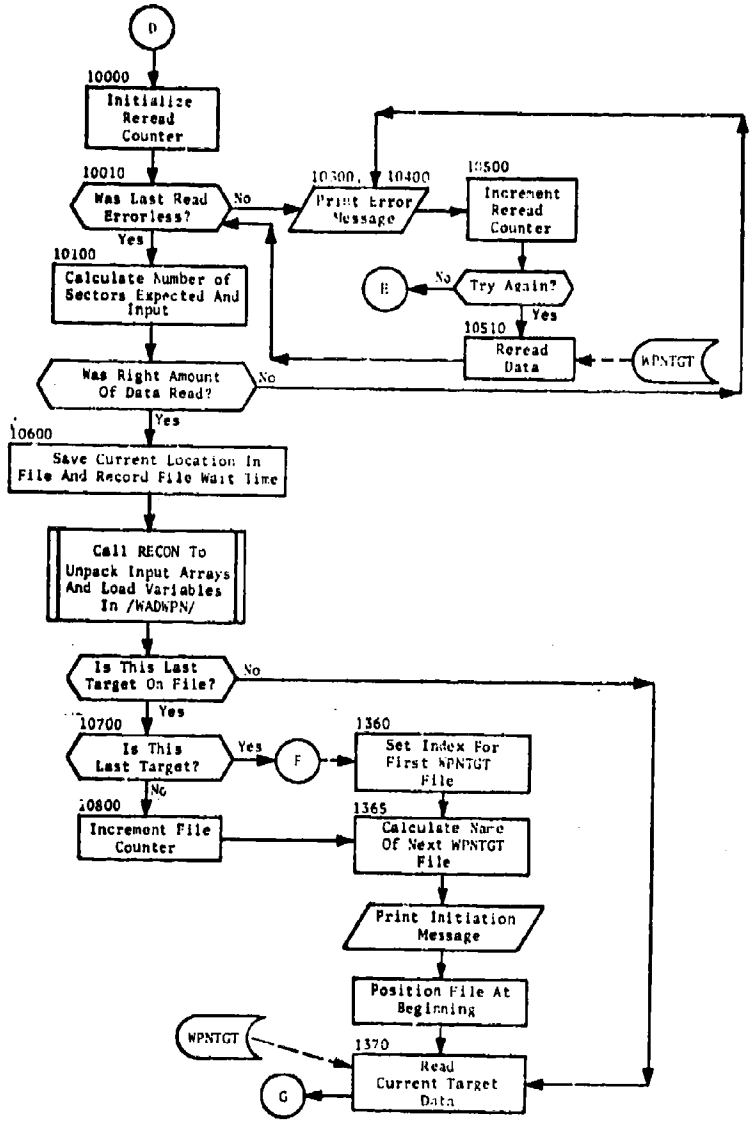


Fig. 35. (cont.)
Part VI: Second Pass Processing

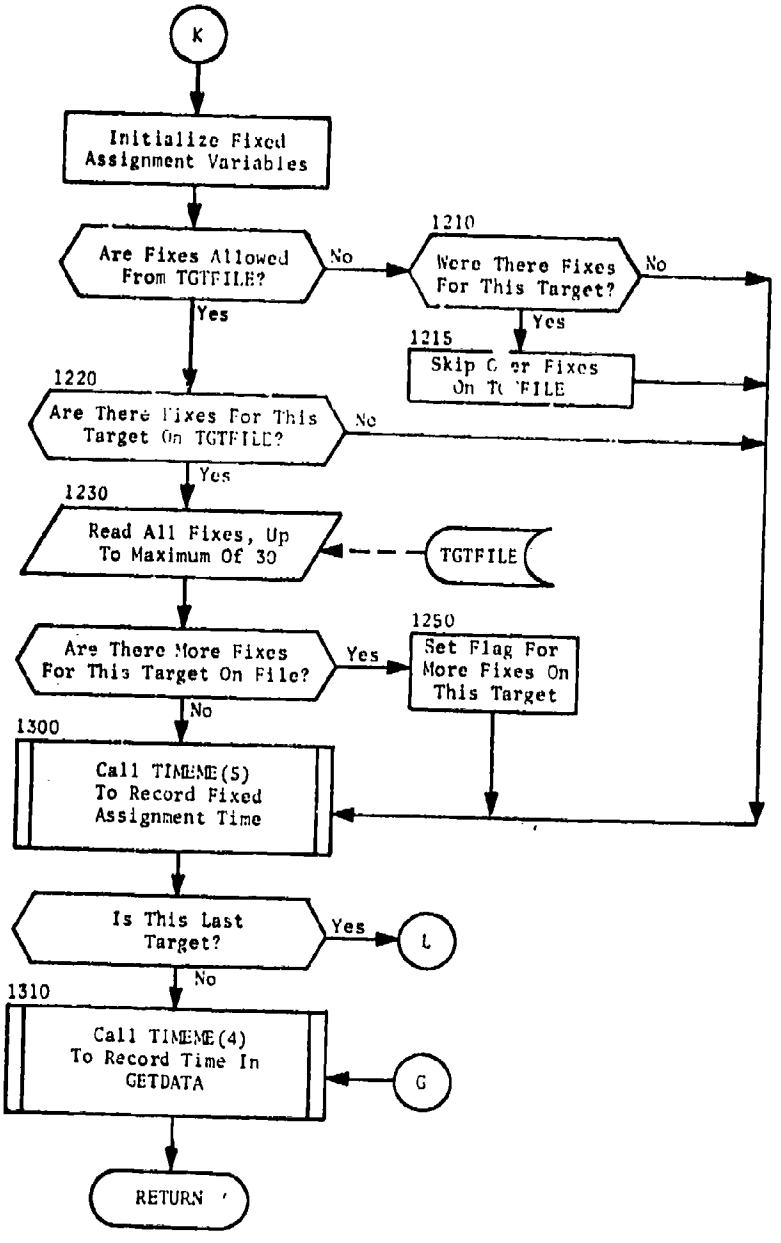


Fig. 35. (cont.)
Part VII: Fixed Assignment Processing

SUBROUTINE INITALC

PURPOSE:	This routine initializes many variables used in program ALOC.
ENTRY POINTS:	INITALC
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, 222, RANGE, RESTRICT, FLAGRST, CTRYRST, MACHINE, WPNGRP
SUBROUTINES CALLED:	None
CALLED BY:	ALOC

Method

This routine initializes the variables in blocks /222/, /RANGE/, /RESTRICT/, /FLAGRST/, /CTRYRST/, and /MACHINE/. Common /MASTER/ is used to retrieve the value of NCNTRY (maximum number of country codes). Common /WPNGRP/ is used to retrieve the value of MGROUP (maximum number of weapon groups).

Subroutine INITALC is illustrated in figure 36.

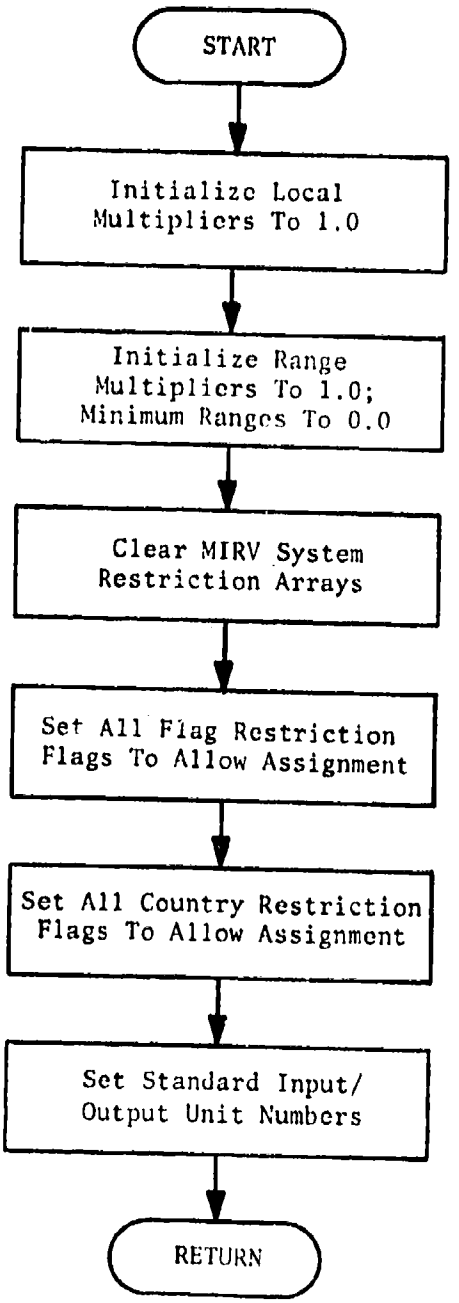


Fig. 36. Subroutine INITALC

SUBROUTINE LOCREST

PURPOSE:	This routine reads and processes the country code restriction user-input parameter cards.
ENTRY POINTS:	LOCREST
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, CTRYCD, CTRYRST, MACHINE, WADWPN*
SUBROUTINES CALLED:	NUMGET, ITLE
CALLED BY:	ALOC

Method

This routine performs the input and processing of the weapon allocation restrictions by country code (CNTRYLOC). Its purpose is to set the country code restriction flags (logical array CTRYOK in common /CTRYRST/) according to the user-input parameter cards.

Array CTRYOK in common /CTRYRST/ is a logical array indexed by a country code index and group number. The element CTRYOK (I,J) represents the restriction of a weapon from group J against a target whose country code occupies position I in the country code list, array CTRYCD in common block /CTRYCD/. The array CTRYCD was filled by program PREPALOC when the TGTFILE was created. It is a list of all the country codes represented in the target system. If CTRYOK (I,J) is .TRUE., then weapons from group J can be allocated against a target whose country code occupies the Ith position in the list. If CTRYOK (I,J) is .FALSE., the weapon from group J cannot be allocated to targets whose code is in the Ith position.

The initial setting of the CTRYOK array is performed in subroutine INITALC where all values are set .TRUE.. The user has two input options for country code restrictions, SELECT or DELETE. In the SELECT option, all elements in CTRYOK for a group are set .FALSE., except for those codes specified on the input card. In the DELETE option, specific elements in FLAGOK are set .FALSE. according to the user input. The SELECT or DELETE option is chosen by the user for each group. The local logical variable SET performs the operation of modifying the CTRYOK array. If the user option for a group is DELETE, SET is set to .FALSE.

*Common block /WADWPN/ used as an input buffer.

(statement 50). If the SELECT option is chosen for a group, the statements from 60 to 80 give SET a value of .TRUE., and all values of CTRYOK for the requested group are set .FALSE..

In the block from statement 80 to the DO 120 loop, the subroutine removes all embedded blanks from the user-input country code list. This is done by breaking down the list into individual characters. The list is reordered by placing all blanks at the end of the list while not changing the order of the nonblank characters. The resulting compressed code list is divided into sections of two-character length, the length of the country code.

The DO 120 loop investigates each code on the input card. When a blank code is encountered, the program realizes the end of the current list has been reached. If the input code is present in the list prepared by program PREPALOC, the appropriate CTRYOK flag is set according to the value of the variable SET.

When the END is encountered, signalling the end of the input, the country code restrictions flags for all groups which have been restricted are printed (statement 1000). For ease in reading by the user, the list of country codes is divided into sections of 35 codes. Within each section, any group restricted from hitting targets with a code in that section is represented by a list of its restriction flags for codes in that section. A printed value of 1 represents a value of .TRUE. for CTRYOK, and a value of 0 represents a value of .FALSE..

Figure 37 illustrates subroutine LOCREST.

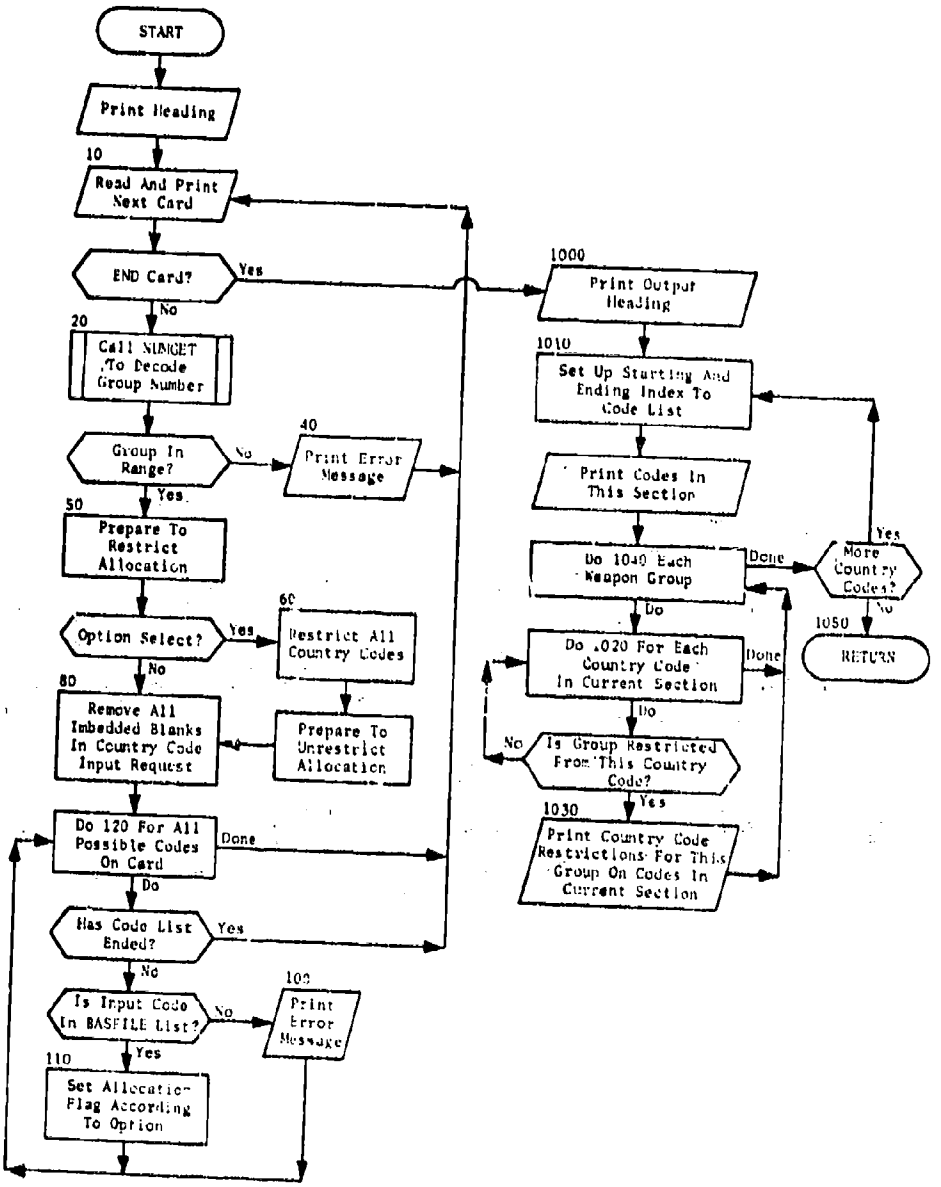


Fig. 37. Subroutine LOCREST

SUBROUTINE MIRVRST

PURPOSE:	This routine reads and saves the data concerning the restriction of weapons with MIRV capabilities to certain target classes. These data are used by subroutine GETDATA to determine whether a weapon system can be used against a target.
ENTRY POINTS:	MIRVRST
FORMAL PARAMETERS:	None
COMMON BLOCKS:	RESTRICT, WADWPN*
SUBROUTINES CALLED:	NUMGET
CALLED BY:	ALOC

Method

This routine reads a set of data cards which list the target classes to which the user wishes to restrict the allocation of MIRV weapons. Each data card contains the MIRV system identification number (IMIRV) and a list of permitted target classes (statement 10). This data set is terminated by a 0 or negative IMIRV number.

The permitted target class name fields are scanned on each card. Only the nonblank fields are processed. The contents of these fields should be the names of the QUICK target classes (e.g., MISSILE, H/I), including COMPLEX and COMPLEXD). In addition, two other target attributes may be put on the permitted list. The field DEFENDED permits targets with terminal ballistic missile defenses; the field MULTIPLE permits multiple targets.

The cards do not need to be in any order, nor is there any required order for the target classes. The same IMIRV number may appear more than once. If so, the permitted classes on the second and succeeding cards are added to the classes permitted on previous cards. The maximum number of permitted classes, not including DEFENDED and MULTIPLE, is 16. There is no limit on the number of restriction cards, either total or per IMIRV number.

If this run option is used and there is no restriction card for a certain system, that system will be unrestricted. Note that the data fields are

*Common /WADWPN/ is used as an input buffer.

permitted classes. Thus, if a card contains an IMIRV number and is blank thereafter, the system cannot be allocated unless there are other cards with that number that do contain permitted classes.

The list of permitted classes (MYPERMIT) is passed to subroutine GETDATA through common block /RESTRICT/. The logical arrays TMBDEF and MULTARG are used for terminal ballistic missile defended target restrictions and multiple target restrictions, respectively. If an element in one of these arrays is .TRUE., the system can hit a target of that type. For example if TMBDEF(3) = .TRUE. and MULTARG(3) = .FALSE., weapons with an IMIRV value of 3 can be allocated against defended targets but not against multiple targets.

Subroutine MIRVRST is illustrated in figure 38.

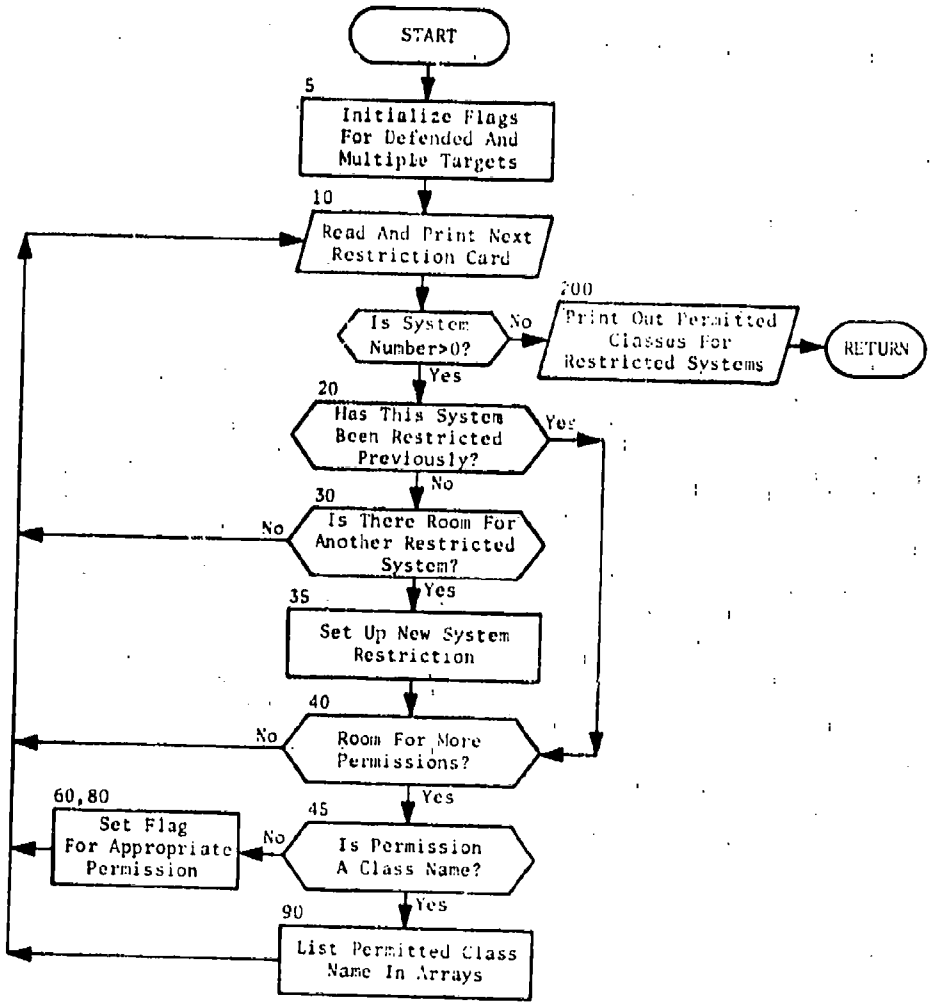


Fig. 38. Subroutine MIRVRST

SUBROUTINE MULCON

PURPOSE:	The primary purpose of MULCON is to adjust the Lagrange multipliers during the allocation. It also monitors the progress of the allocation, controls the flow of the program throughout the allocation, and handles much of the file input/output.
ENTRY POINTS:	MULCON
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, FILES, FILABEL, MYLABEL, ITP, MYIDENT, NOPRINT, TWORD, LOCFIL, WPNREG, WPNTYPE, PAYLOAD, WPNGRP, CONTROL, DYNAMIC, FORMATS, LAMBDA, MULADJ, PRNTCON, 222, NALLY, PAYOFF, PRINEED, PRIN1, PRNTWADD, PRTMULL, 333, WADFINAL, WADOUT, WADWPN, DEFENSE, FIXED, FIXEDASS, SMAT
SUBROUTINES CALLED:	SETREAD, RDARRAY, SETWRITE, WRARRAY, TERMTAPE, TIMEME, SETABLE, RDALCRD, PRNTCON, PRNTALL, INITGET, GETDATA, NUMGET, PRNTNOW, SORTMIS, STALL, DEFALOC, PREMIUMS
CALLED BY:	ALOC

Method

Definitions of variables local to MULCON:

RTGTWT(INTPRD)	The current value of the changing target weight to be used; may differ from TGTWT(INTPRD) since the latter as read in from tape may represent the value on the previous pass
PARTIAL(J)	The estimated dependence of allocation rate on the local multiplier
NOWPS(J)	The total number of weapons associated with the local multipliers (TYPE REAL)
CURSUM(J)	The running sum of target weight multiplied by weapons assigned for local multiplier J since the local multipliers were last evaluated

The processing in MULCON involves two types of files, dynamic and static. The dynamic files are the ALOCT1 and ALOCTAR (or ALOCT2) files. They contain information on the targets and weapon allocation that varies from pass to pass. Hence, the file content is dynamic. The WPNTGT files produced by GETDATA are the static files. The weapon-target interaction data calculated by GETDATA in the first pass and written on these files do not change as the allocation progresses. Thus, the file content is static.

MULCON proceeds by making multiple passes through the target list until the correct number of weapons have been allocated. During this process, the Lagrange multipliers are modified to force the allocation to converge to the given stockpile. The basis for most of the MULCON calculations is discussed in the Analytical Manual for the Plan Generation Subsystem and the Concept of Operation section of this chapter.

The flow of operations in MULCON is illustrated in figure 39. The first sheet represents the flow in five parts of the program: Part I, the initialization phase; Part II, fixed weapon assignment processing; Part III, the main flow; Part IV, processing after allocation; and Part V, multiplier adjustment. The flow diagrams are annotated with statement references to the FORTRAN listing and are in sufficient detail to be largely self-explanatory. In the following sections, comments are made only where the flow diagrams require some explanation.

Part I: Initialization

The routine begins by calling utility subroutine TIMEME to initialize the clock. The dynamic file names are selected and the files initialized for input/output. Subroutine SETABLE is called to initialize the table in common /TABLE/ needed by function FMUP for the square root damage law. Subroutine RDALCRD is called to read and process the user-input parameter cards for the ALLOCATE function. The description of these parameters is contained in the description of subroutine RDALCRD. NPASS and ITGT are initialized so that the print control routine PRNTCON will know what prints to activate, and then PRNTCON is called to do so. This early call on PRNTCON is primarily to allow control of prints during the initialization phase. A call for a summary print (PRNTALL(1)) immediately follows.

In order to determine the allocation rates associated with specific local multipliers LA(J) to compute the actual Lagrange multipliers LAMEF(G) from the local multipliers, and finally to evaluate the correlation cross terms between different weapons, it is necessary to have an index which makes it possible to look up all the local multipliers associated with each weapon group G. This index JATTRIB(IAT, G) is computed in the 100 series of statements.

In the 500 series statements, the initial values of the actual Lagrange multipliers are computed. In addition, the total number of weapons NOWPS(J) associated with each local multiplier LA(J) is determined so that the expected rate of allocation associated with the multiplier can be determined.

In statement 602, the sensitivity of the allocation errors to the value of the local multipliers is estimated. With a fairly large value for the linear premium PRM all sensitivities are about the same independent of the size of the group; thus, PARTIAL(J) is simply set to -1.0.

In statement 603, RUNSUM and WTSUM are initialized as if the starting pseudo-allocation were exact, but the weight given to that allocation is reduced by the square root of the number of targets so that it does not take too long for data on actual allocation rates to produce a significant effect on the estimated rates.

The initialization phase ends with a call on INITGET to initialize the variables in subroutine GETDATA and a call on subroutine TIMEME to record time spent in initialization.

Part II: Fixed Weapon Assignment Processing

This section deals with the fixed weapon assignment processing in the 900 and 9,000 series of statements. The processing begins by clearing the arrays used to store the assignments. Subroutine GETDATA is called to calculate the weapon-target interaction data and write the WPNTGT file. These calculations will determine the validity of the fixed assignments. Subroutine PRNTCON is called to control the printing for this pass and target. Then the target number is checked against the number of the next fixed target to be processed. If they are equal, the fixed assignments are read. Only the first 30 weapons assigned are read. If there are more, subroutine DEFALOC will read the remainder. The fixed assignment data, including the arrival time information, is processed by STALL and DEFALOC. The origin of the fixed assignment input data was selected by the user input options in subroutine RDALCARD. The variables in common blocks /FIXED/ and /FIXEDASS/ describe the origin of this information.

Part III: Main Flow (After First Pass)

In statement 1206, it is necessary to save the old allocation data which have been read in off the dynamic file, since they would otherwise be overwritten by the new allocation.

The test for a new target is necessary since the program might be processing a second component of a multiple target (if PROGRESS ≥ 1.0).

In this case the target index ITGT in the DYNAMIC array would still be the same as JTGT and it would be a mistake to request new static data from subroutine GETDATA.

During the main allocation phase PROGRESS is less than 2 so usually the termination closing phase on the lower right of the sheet is skipped entirely.

When PROGRESS first becomes equal to 2, the allocation is complete and the file then being written on is noted as STRKIN for ALOCOUT (statement 2721 Part V). However, the allocation at this point is not all on that file. Consequently, in order to get the allocation on one file, it is necessary to continue reading the old and simply copy the old allocation to fill out the new file. This is done in two ways. If verification or testing of the allocation is to be done (IVERIFY = 1,2), the old allocation is saved in statement 1220 and the program goes on to carry out the verification, reading in the old allocation to the next target as usual in statement 1201. Since verification always requires a full pass, this process will assure that the STRKIN file is complete. (When the end of the file is reached and files are switched, it continues writing on the next file, but this does no harm.) When the verification is complete all files are terminated (statement 1231) and subroutine SORTMIS is called to write the MSLTIME file for use by program PLNTPLAN. A test is then made to determine if the STRKIN file is the intended file, ALOCTAR. If so, control returns to the main program immediately after a final print of timing data. If not, the ALOCT1 file is copied onto the ALOCTAR file before the timing print and return.

However, assuming that PROGRESS is not yet equal to 2.0, the ermination section is ignored and the program gets ready to generate a new allocation to replace the one just read in statement 1201. On the first pass the old allocation is a pseudo-allocation and the replacement is done elsewhere (see Part II). The replacement is accomplished by removing the contribution of the old allocation to all running sums before the new allocation is generated. This is accomplished in the 1300 series of statements. This can be done in this simple way because the values of COST, PAYOFF, PROFIT, TGTWT, and DPROFIT, then in memory, are those just read in from the dynamic file and so they correspond to the old allocation. Since the quantities RUNSUM and WTSUM were divided by WTFAC at the end of the last pass when the files were interchanged, the old TGTWT must also be divided by this same factor to make it commensurate before it is subtracted out.

The reason that REVCOST must be computed is that the values of the multipliers have probably changed (unless PROGRESS = 1.0) since the prior allocation; consequently a revised cost (REVCOST) of the allocation based on the new multipliers is of interest, and is probably

different than the old cost COST. The reason for the test on PROGRESS before correcting the cumulative differential profit will be discussed in connection with Part IV of the program (Sheet 5 of the figure).

Part IV: Processing After Allocation

Before calling STALL, CTSPILL is set to 0. (If some elements are spilled, WAD will so note by setting CTSPILL equal to the number of elements spilled.) The calls on TIMEME (5, 6, 7) before and after the allocation cause the time spent during the actual allocation to be recorded in columns 6 and 7 of the TIMEME output print (number 23).

Before calling either allocation routine, however, the program must check the number of fixed weapon assignments. The limitation on weapons allocated to one target is 30 weapons on an undefended target and 30 weapon groups on a defended (i.e., terminal ballistic missile defenses) target. Usually, MULCON calls both STALL and DEFALOC on defended targets and chooses the best allocation. If there are more than 30 fixed weapon assignments, STALL should not be called. If the number of fixed assignments is greater than 30, MULCON checks to see if it is a defended target. If not, an error message is printed and the excess assignments are ignored. Then STALL is called. If it is a defended target, MULCON sets a dummy low profit (except for verification) and calls only DEFALOC.

The additional details of the allocation required by later processors are then recorded in /DYNAMIC/. PEN and TOARR are required by EVALALOC, while KORR and VTD (as changed to compute RVAL) are required by ALOCOUT, FOOTPRNT, and POSTALOC.

The various running sums are then calculated in statements following 1402. If DEFALOC has made the allocation, the KORR array gives the number of missiles from each group allocated to the target. If KORR is positive, it represents the corridor. If it is negative, it represents the number allocated. Then the profit and cost data are recorded.

The following quantities are of particular importance:

DPROFIT = PROFIT - OPROFIT

SDPROFIT = Σ DPROFIT

DELTEFF = DPROFIT/VALWPNS

SDELTEFF = SDPROFIT/VALWPNS

These quantities are computed and the last two printed out in the standard ALOC print number 2 to help the user

evaluate the progress of the allocation.* The quantity OPROFIT represents the profit of the old allocation to the target evaluated in terms of the present values of the Lagrange multipliers. DPROFIT is thus a measure of the improvement in profit using the new allocation. Up until PROGRESS = 1.0 this quantity is summed over all targets (one complete pass only) to give SDPROFIT. Thus, when the multipliers have been near the correct values for one full pass the value of SDPROFIT should be small. To provide a standard relative value for interpreting these quantities, they are divided by the value of all the weapons VALWPNS,

$$\text{VALWPNS} = \Sigma\ \text{NWPNS(G)} * \text{LAMEF(G)}$$

to obtain DELTEFF and SDELTEFF which measure changes in profit as a fraction of the total value of all weapons.

The quantity of SDELTEFF, therefore, provides an estimate of how efficient the allocation would have been if the allocation had been terminated one pass earlier. Presumably, the current efficiency is substantially higher, but SDELTEFF does not, at this point, give any indication of how much. It is nevertheless of value in developing experience on how soon the PROGRESS .75 phase can be terminated. When PROGRESS is equal to 1.00 the multipliers are frozen and this role of SDELTEFF ceases to be relevant. The quantity is then reset to 0. Thereafter it provides a measure of the effect on the profit of closing to the exact stockpile. Usually during the closing phase SDELTEFF goes slightly negative. However, since during this phase we continue to replace allocations originally produced with slightly different values of the multipliers, the value may go positive for a while until the closing forces get large enough to force closure even at some loss of profit. Thus the value of SDELTEFF at the end of the closing phase (PROGRESS = 1) measures the loss in profit associated with closing. In the event that closing requires more than one full pass, a test has been inserted just before statement 1304 which causes SDELTEFF to continue to accumulate over more than one pass when PROGRESS = 1.0.

Finally when PROGRESS = 2.0 the quantity is again set equal to 0. If a verification pass is carried out, SDELTEFF then measures any increase in profit in the verification pass relative to the final allocation. In this role it defines an upper limit on the inefficiency of the actual allocation.

Ordinarily after all these calculations are performed the results are stored on the output file in statement 1501. However, if PROGRESS = 2 this step is skipped since the output file already contains the final allocation.

*The column labelled (P-O)/VWPS in print number 2 contains these variables: DELTEFF on the first line, SDELTEFF on the second.

If any elements of a multiple target spilled, the weapons previously assigned to these elements are removed (statement 1505) and the program recycles through point C (statement 1401) without changing the basic target data. One change, however, does have to be made. Later programs assume that where multiple targets exist, the INDEXNO will correspond to the first target of the multiple and that the other values of INDEXNO must be set to correspond to the first element of the second part (statement 1502).

If spilling did not occur, a test is made (statement 1503) to see if all elements of the target have been processed. This is necessary since a multiple target might have split on a prior pass. If so, the program cycles back immediately to point B (statement 1201) to process the remaining elements. Thus, multipliers are never recomputed in the middle of a multiple target.

At the end of each complete target, the target weight is adjusted (statement 1709). If it is also the end of a pass, the target weight is still adjusted by the same amount relative to other targets, but all target weights and the value of RUNSUM and WTSUM are renormalized (statement 1711).

After the target weights are adjusted, a test is made to see if it is time to recompute the multipliers. If so, control is transferred to point D (statement 2001) of Part V.

The details of the file handling at the end of a pass are shown in Part IV. Two other operations, however, require special comment. At the bottom of the diagram a test is made to see if sufficient progress has been made. If after three passes PROGRESS has not reached .75, it is assumed that a problem exists and the run is terminated. In the middle of the diagram, at the end of the first pass, the value of NOWPS(J) is reevaluated omitting any weapon groups that could not reach any targets. This allows the allocator to ignore such weapon groups thereafter, and avoids an endless and fruitless effort to allocate these weapons by reducing their Lagrange multipliers. The array TVALTOA provides a convenient test, since it is initialized to zero and will remain zero only for weapon groups that have never been within range of any target.

Part V: Multiplier Adjustment

Every fourth target or so,* when it is decided to recompute the multipliers, controls passes to this adjustment routine. The first step is to recompute all the allocation error estimates, ALERREST. At the same time SURPWP, the excess allocation estimate, is reevaluated based on the new value of ALERREST. Although SURPWP is continously updated by the operating program it is useful, especially in the early phases

* Every second target for PROGRESS equal to 0.0, 0.1, and 0.5; every fourth target for PROGRESS equal to .75 and 1.00.

of the program, to base it on the projected allocation-rate estimates rather than the actual weapons allocated, which at that time could be very misleading. This provides a more rational basis for calculating the premiums at this early stage of the program.

If PROGRESS = 1.0, the change of local multipliers is omitted so that the same values of the multipliers are retained. Otherwise control passes over the local multipliers to the Do loop (statement 2101). Each multiplier is changed only if all the estimates of error rate have the same sign. In the early phases of the program (PROGRESS .LT. .75) better stability is achieved by requiring, in addition, that the average allocation rate to the last 2 to 4 targets (as computed from CURSUM) show the same sign. This limitation is later removed, since it clearly would not work well for weapon groups with very small numbers of weapons that might only be allocated 20 to 10 times during a pass over the target system.

In the 2300 series of statements an estimate is made of CORRATE, the rate at which it is desired to correct the allocation rate. If the allocation rate is corrected too rapidly there will be a tendency to over correct before the effects of the correction become observable in the values of the allocation error estimates. This can produce oscillations. To estimate how rapidly to correct the error, an estimate is made of the number of targets that would have to be observed before an error of the observed size would be statistically significant. Even if the multipliers were exact and the average allocation rate was correct, statistical fluctuations would be observed in the allocation of each weapon group when the allocation rate was sampled for a small number of targets.

Let n equal the expected or average number of weapons from a group available per target; i.e., n = NOWPS(J)/NTGTS. Then in M targets the expected number of weapons allocated should be just n(M). Suppose the actual number observed however is n'(M). Then our estimate of the error in the allocation rate ALERREST would be

$$\text{ALERREST} = n' - n$$

Assuming a Poisson distribution, the statistically expected error in a number of expected value n(M) is equal to $\sqrt{n(M)}$. That is,

$$(n'(M) - n(M))^2 = n(M)$$

$$(n' - n)^2 = n/M$$

Solving for the number of targets M we have:

$$M = n/(n' - n)^2$$

or

$$M = (NOWPS(J)/NTGTS)/(ALERREST(J))^2$$

as the number of targets we should expect to sample to get a statistical error estimate of size, ALERREST. If we wish to reduce the indicated error by 1 part in M per target, our fractional correction in the allocation rate per target should be:

$$1/M = ALERREST ** 2/(NOWPS(J)/NTGTS)$$

This, multiplied by a sensitivity factor SNSTVTY, is the first term in the value of CORRATE. However, if the entire set of targets were observed, the estimate would not be a sample but would be exact. Therefore, even a very small value of ALERREST becomes statistically significant if it is based on a sample of size NTGTS. Therefore, errors should always be corrected at a rate at least equal to one part in NTGTS. This explains the second term in CORRATE which is just 1.0/NTGTS multiplied by a sensitivity factor FSNSTVTY (final sensitivity). This factor controls the sensitivity of corrections to the allocation rate in the final phase of the allocation where the errors are small. Thus the desired correction rate is just:

$$CORRATE = SNSTVTY * ALERREST ** 2/(NOWPS(J)/NTGTS) + FSNSTVTY/NTGTS$$

This is multiplied by the number of targets processed between corrections MULSTEP to determine the fraction CORFAC of the error to correct. In addition, in statement 2303 a safety limit of 1/2 is used to avoid ever making a correction larger than 1/2 the estimated error rate.

However, even when it is known what fraction of the error in the allocation rate we wish to correct, an estimate must be made of the relationship of the allocation rate to changes in the Lagrange multipliers before the size change to make in the multiplier can be estimated. For this purpose it is useful to have a model of the dependence of the allocation rate on the value of the multipliers. We have assumed a dependence as follows:

$$Rate = k\lambda^{-n}$$

Consider now two rates, the current rate R_0 associated with a multiplier λ_0 and a predicted rate R_1 associated with a new multiplier λ_1. Thus we find

$$R_1\lambda_1^{\,n} = R_0\lambda_0^{\,n} = k$$

or

$$R_1/R_0=(\lambda_1/\lambda_0)^{-n}$$

so

$$\frac{\partial(R_1/R_0)}{\partial(\lambda_1/\lambda_0)} = -n$$

For small differences between λ_0 and λ_1 this implies:

$$\frac{R_1 - R_0}{R_0} = -n \frac{\lambda_1 - \lambda_0}{-\lambda_0}$$

Solving for the new value λ_1 of λ

$$\lambda_1=\lambda_0\left(1 + \frac{(R_1 - R_0)/(-n)}{R_0}\right)$$

If we now identify a new variable R_2 as the ultimately desired allocation rate, R_1 as the new rate we hope to obtain with λ_1, and R_0 as the current allocation rate -- then the above variables can be associated with information already available as follows:

$$R_1-R_0 = \text{CORFAC}*(R_2-R_0)=\text{CORFAC}*\text{ALERREST}$$

$$R_0 = \text{ALERREST} + (\text{NOWPS/NTGTS})$$

If we now associate the FORTRAN variable PARTIAL with n and the local multiplier LA with λ this gives rise to the following procedure for updating LA:

$$LA_1 = LA_0 * 1.0 + \left[\frac{\text{CORFAC} * \text{ALERREST(J, INTPRD)}/(-\text{PARTIAL})}{\text{ALERREST(J, NTPRD)} + (\text{NOWPS(J)/NTGTS})}\right]$$

This formula is well behaved if ALERREST is large and positive, but if it is negative and as large as the expected rate (NOWPS(J)/NTGTS) (i.e., if the actual allocation rate is zero), then the denominator goes to 0. In this case an infinite correction would be indicated. To avoid this, the expected rate in the denominator is multiplied by 2 giving:

$$LA_1 = LA_0 * \left[1.0 + \frac{CORFAC * ALERREST(J, INTPRD)/(\ -\ PARTIAL)}{ALERREST(J, INTPRD) + 2 * (NOWPS(J)/NTGTS)}\right]$$

This is the function used in statement 2401.

In the present version of the program the value of PARTIAL(J) has been set equal to 1.0 for all the local multipliers LA(J). This choice is based on the effect of the premium on the sensitivity of the allocation rate to the value of LAMEF or λ. When the multipliers are almost correct, it is usually the case that most weapon groups are in close competition with many other groups with very similar properties. Then a small change in the multiplier LAMEF will produce a very large change in the allocation rates, as the weapon group in question almost totally replaces, or is replaced by, its competitors.

However, such a large error in the allocation rate will not actually occur because as the error builds up the estimated value of the payoff will be automatically changed by the premium. Thus, for constant values of LAMEF, when an equilibrium allocation rate is reached, it must be approximately true that the error in LAMEF is compensated by the premium. That is, if λ_0 is the correct value for LAMEF then:

$$LAMEF\ -\ PREMIUM \cong \lambda_0\ .$$

Since

$$PREMIUM = PRM * LAMEF * \frac{SURPWP\ -\ .5\ *\ CTMULT}{NWPNS}$$

we can define a relation between LAMEF and (SURPWP/NWPNS).

$$LAMEF * (1\ -\ PRM * \frac{SURPWP\ -\ .5\ *\ CTMULT}{NWPNS}) \cong \lambda_2$$

Since this relationship is the same for all groups it is reasonably simple to use the same value 1.0 of partial derivative for all local multipliers.

In the 2500 series of statements the values of LAMEF(G) are recomputed using the new values of the local multipliers LA(J). At the same time it is necessary to recvaluate the summation of the value of all the weapons VALWPNS = $\sum$LAMEF(G) * NWPNS(G) and the summation of the value of the error in weapons allocated.

VALERR = $\sum$LAMEF(G) * ABSF(SURPWP(G)) using the updated values of LAMEF.

The average number of targets over which allocation rates are averaged (the integration period) is determined by the rate at which the target weights are increased. The 2600 series of statements are concerned with the adjustment of this integration period.

In estimating the rate with which to correct multipliers, it was computed on a statistical basis that even if the allocation rates were correct an estimated error of size ALERREST would be expected if the allocation rates were monitored only over a small sample of M targets where:

$$M=(NOWPS(J)/NTGTS)/(ALERREST(J))^2$$

Thus, if separate integration periods could be used for each local multiplier, M, as defined above, might provide a reasonable basis for determining the period. However, in fact, the same three periods (INTPRD=1, 2, 3) must be used for all local multipliers LA(J). Consequently, the value of the integration period used might be based on an estimate of overall error rate. The corresponding relation is:

$$M=(\sum_G NOWPS(J)/NTGTS)/\sum_G (ALERREST(J))^2$$

where the summations are taken over all weapon groups. The quantity, $\sum_G NOWPS(J)$, is identical with NOWPS(2) and so for efficiency the variable NOWPS(2) is used*. While the expected value of $(ALERREST(J))^2$ is the same as $\sum_G (ALERREST(J))^2$ the variance of the later version is much less and it is therefore preferable as an estimator of the expected integration period, EXPINTPD.

To allow the possibility of using integration periods either longer or shorter than the theoretical EXPINTPD, a desired longest integration period DESINTPD is defined in statement 2604:

```
DESINTPD=EXPINTPD * RATIOINT
```

where RATIOINT is an adjustable input parameter.

If this period were used exactly in setting the rate of change of the target weight; i.e., WRATE=1.0/DESINTPD, and WRATE would never become exactly 0 as is required for a constant target weight. Obviously when the change in the target weight becomes small over a full pass, the WRATE should be allowed to go to 0. Therefore, in statement 2605:

---

* LA(2) is used for all weapon groups.

WRATE = (1.0/DESINTPD) - (2.0/(NTGTS * RATIOINT))

the term 2.0/(NTGTS * RATIOINT) is subtracted and if the resulting WRATE is negative it is set to zero. To avoid a situation where large errors cause the integration period to become ridiculously small, a limit that WRATE $\leq$ .07 is set.

Moreover, after the allocation is well underway, PROGRESS $\geq$ .5 the value of WRATE is not allowed to increase (statement 2615). In the program WTRATE(INTPRD) is used as a multiplier of the target weight; therefore, we add 1.0 to WRATE to obtain a suitable multiplier for the longest period NINTPRD.

The values of the WRATE for the shorter periods are then derived from this value to give a ratio of integration periods (roughly equal to RINTPRD) another input parameter.

The evaluation of progress is shown in the final sheet for the subroutine MULCON. The procedure is very straightforward, and should be obvious from the flowchart. It may be worth noting, however, that when the allocation is finally complete, the index IFINTGT of the last target, the final pass IFINPASS, and the logical unit number for the file STRKIN are all recorded so that the file with the final allocation can be recognized later.

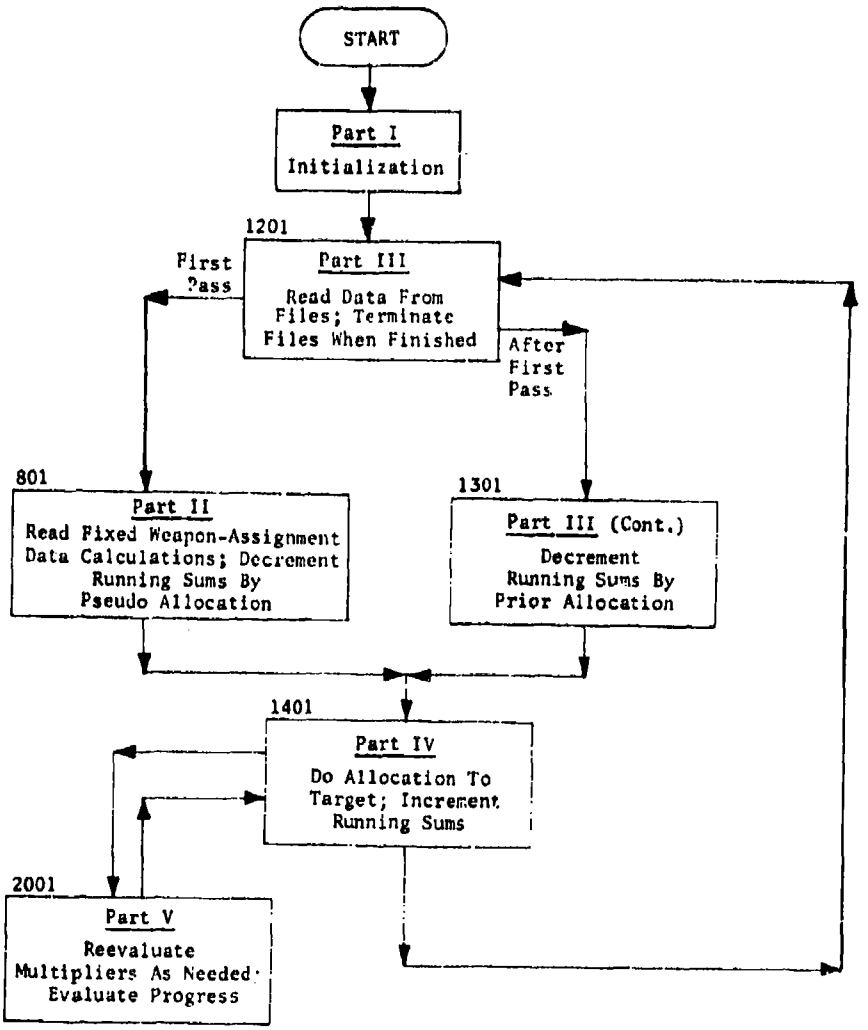


Fig. 39. Subroutine MULCON Summary Flow

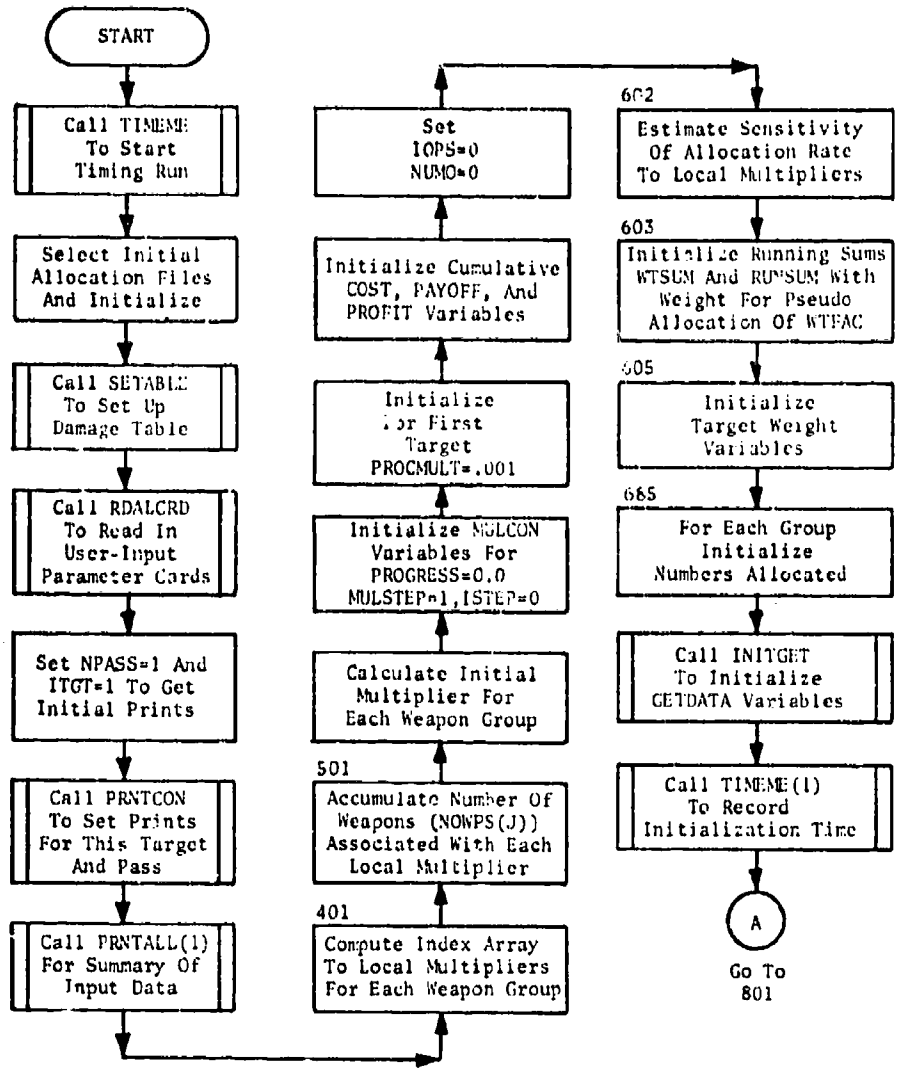


Fig. 39. (cont.)
Part I: Initialization

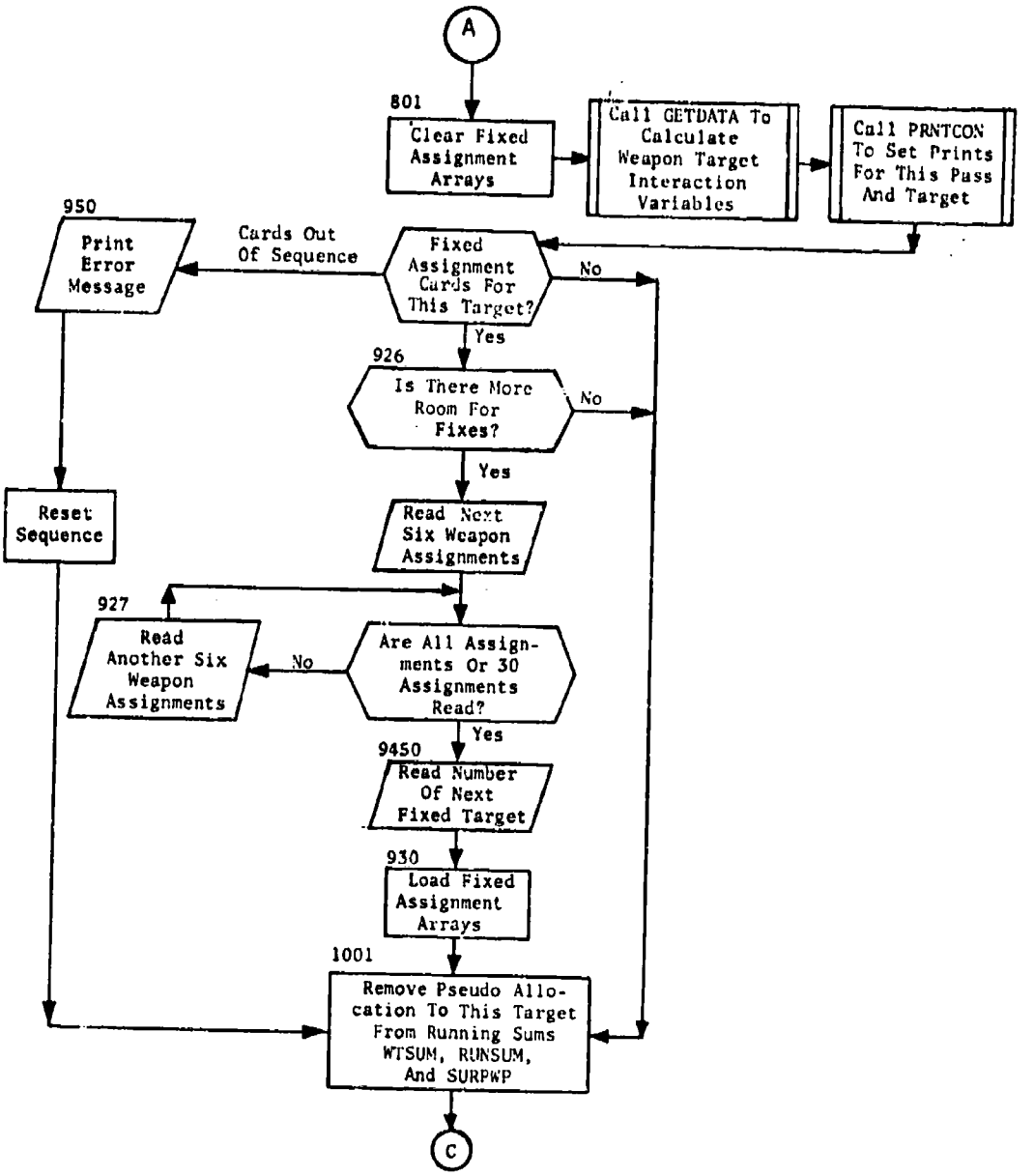


Fig. 39. (cont.)
Part II: Fixed Weapon Assignment Processing

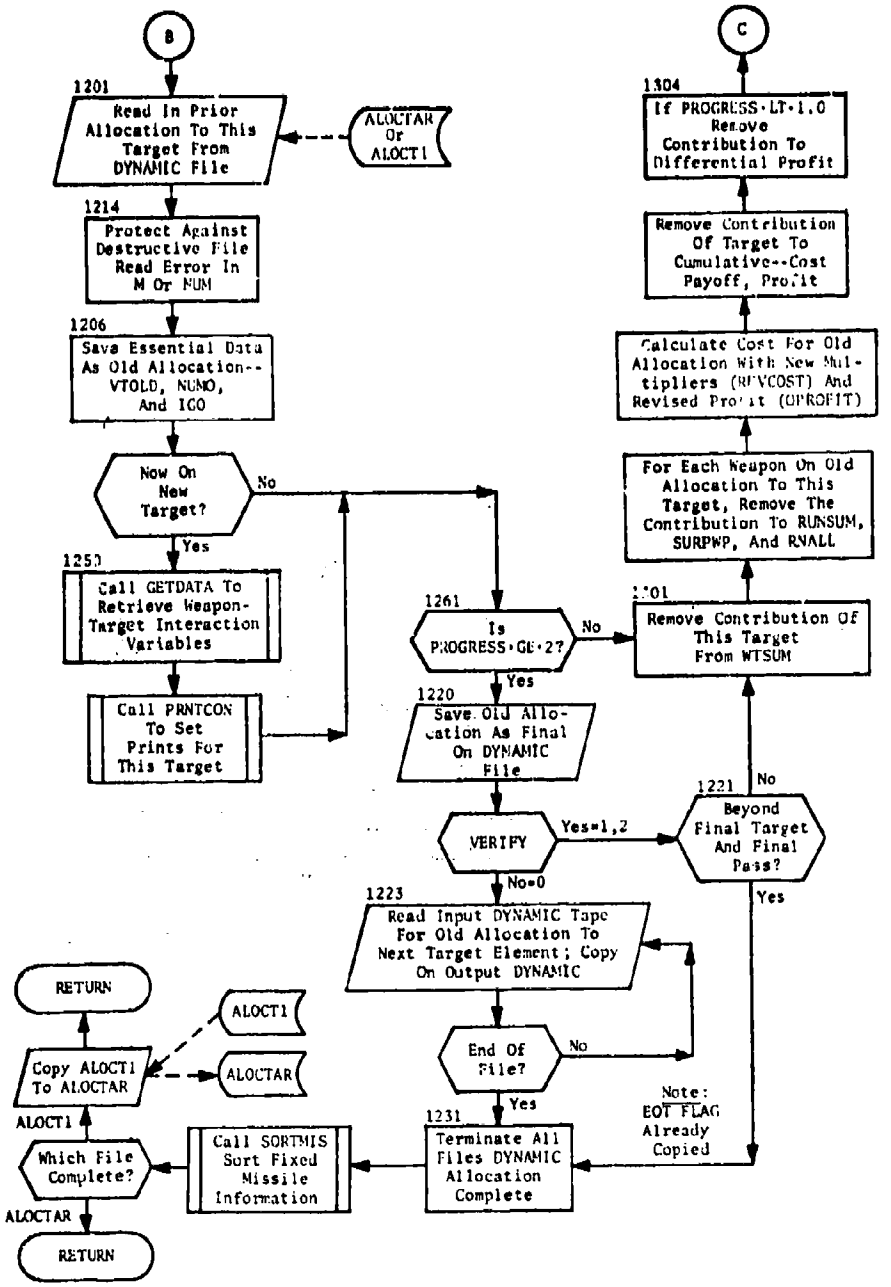


Fig. 39. (cont.)
Part III: Main Flow
(After First Pass)

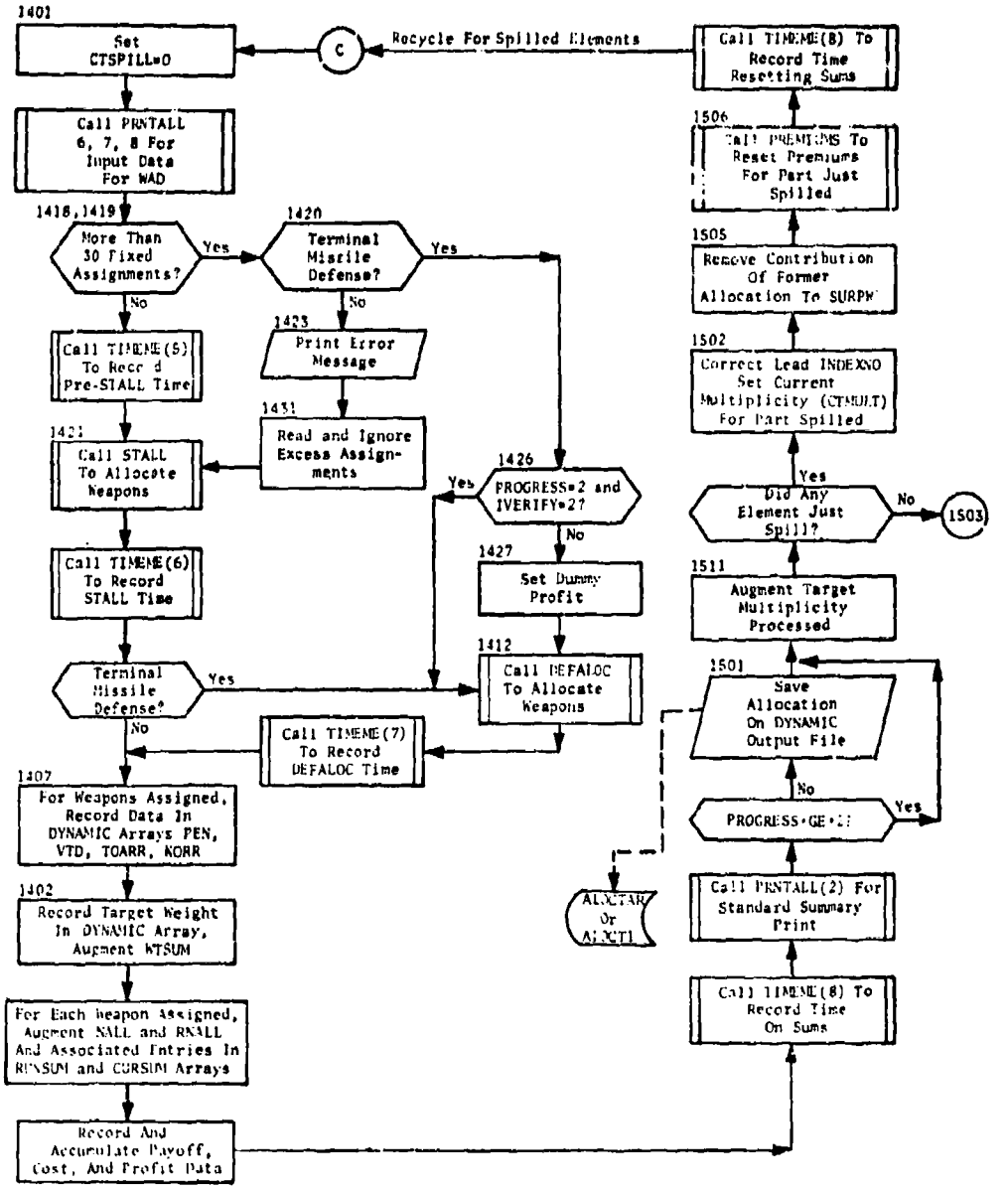


Fig. 39. (cont.)
Part IV: Processing After Allocation
(Sheet 1 of 3)

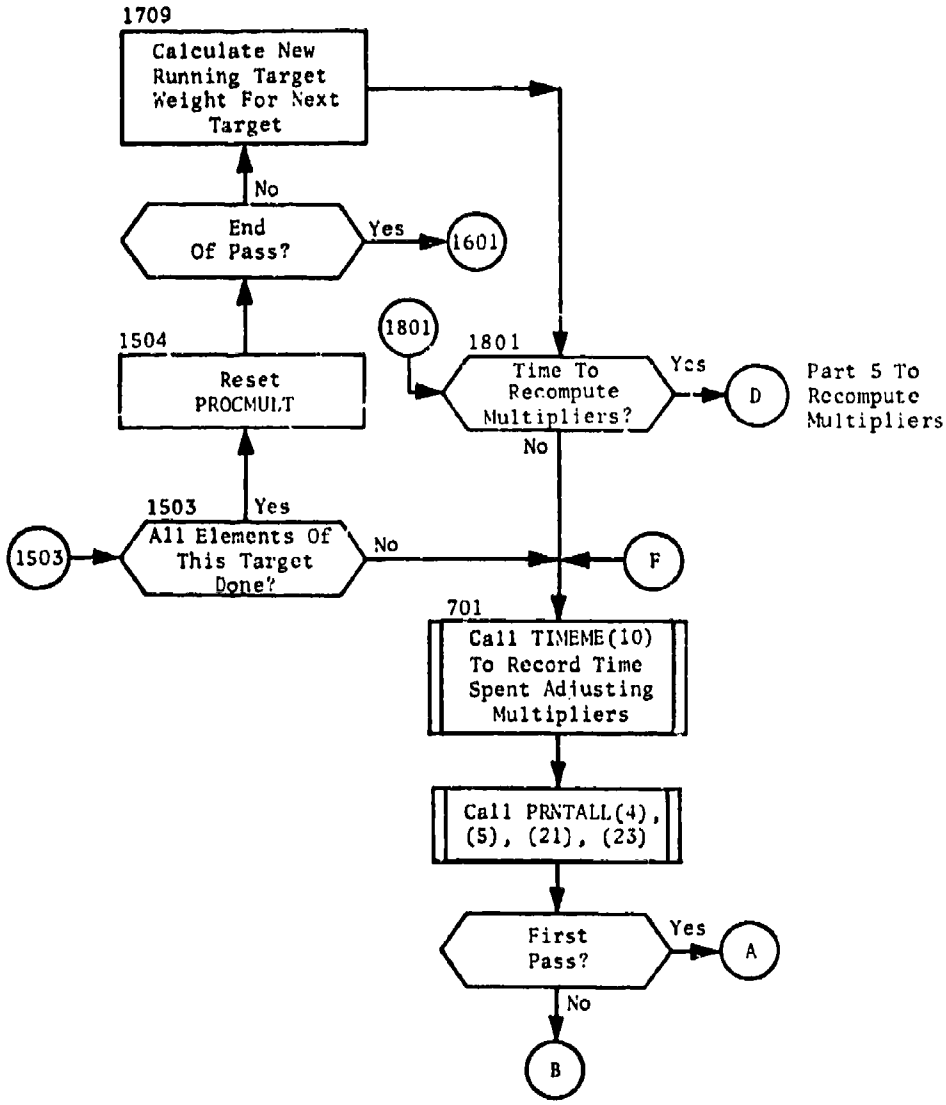


Fig. 39. (cont.)
Part IV: (cont.)
(Sheet 2 of 3)

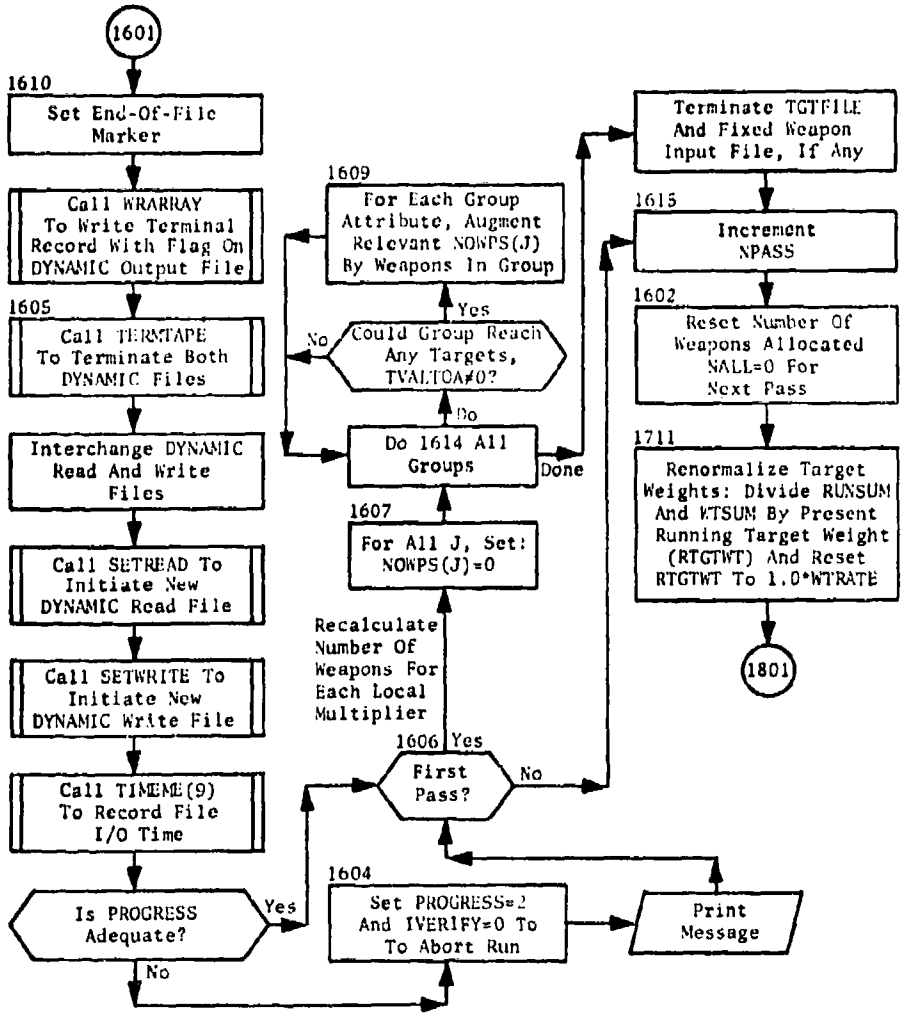


Fig. 39. (cont.)
Part IV: (cont.)
(Sheet 3 of 3)

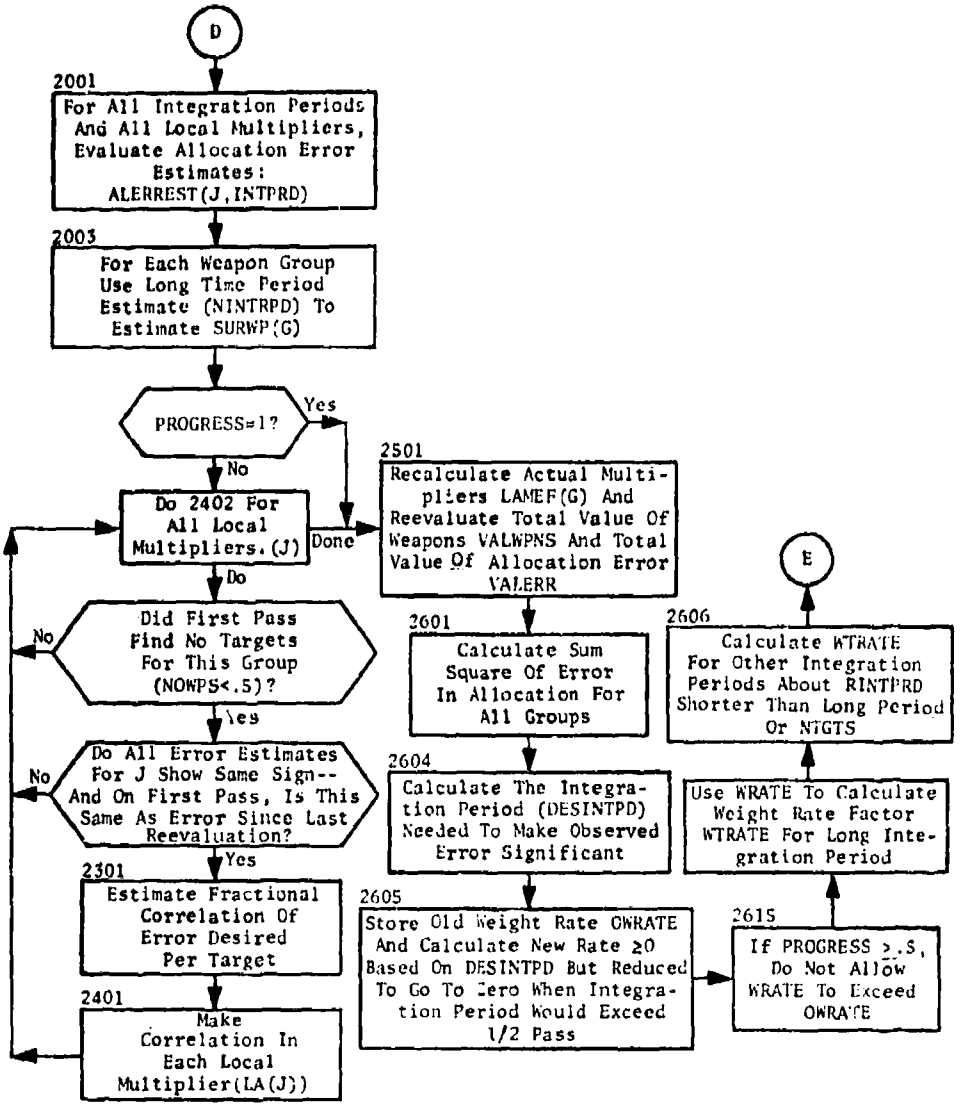


Fig. 39. (cont.)
Part V: Multiplier Adjustment and Progress Evaluation
(Sheet 1 of 2)

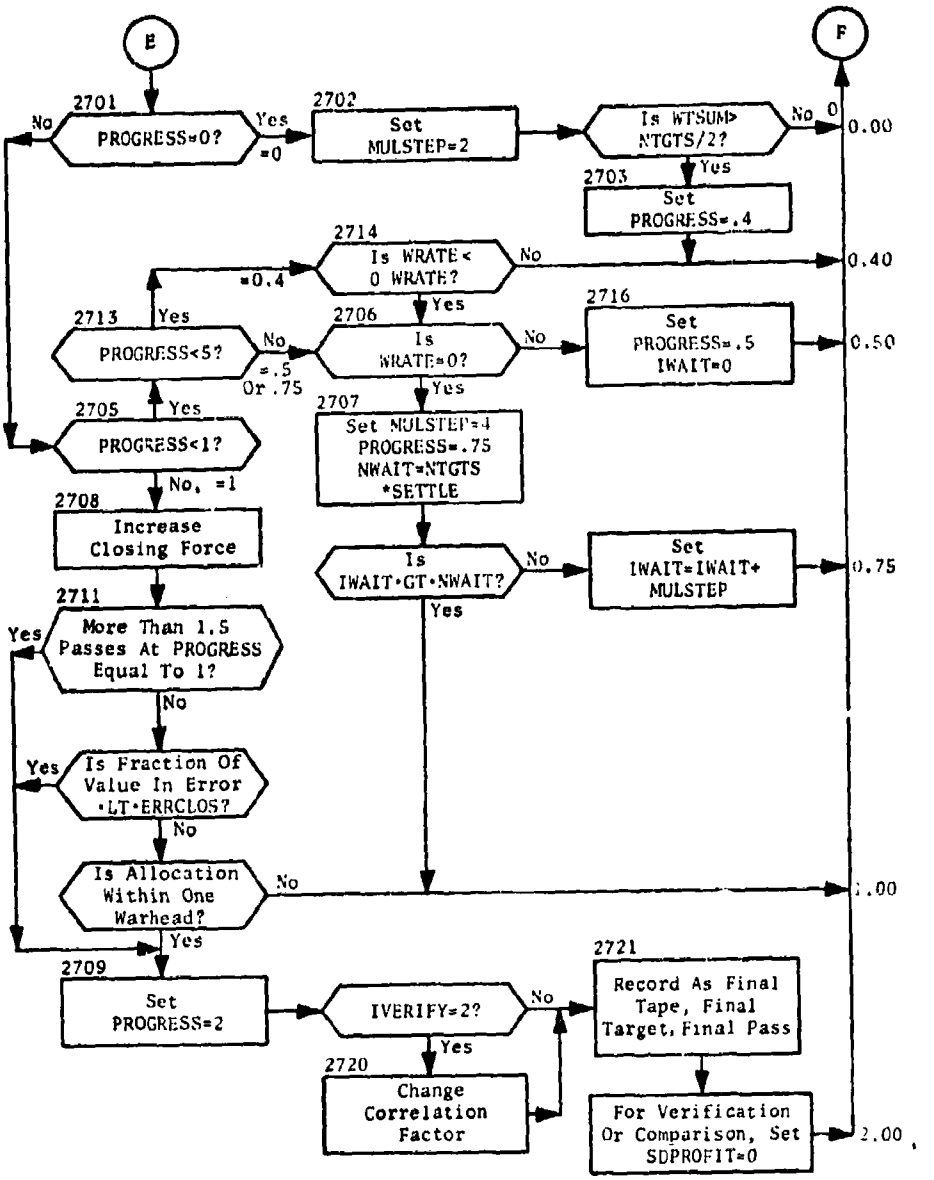


Fig. 39. (cont.)
Part V: (cont.)
(Sheet 2 of 2)

SUBROUTINE PREMIUMS

PURPOSE:	This routine calculates the premium used by WADOUT in evaluating the benefit of using or not using weapons from specific groups.
ENTRY POINTS:	PREMIUMS
FORMAL PARAMETERS:	G - An integer group number
COMMON BLOCKS:	CONTROL, LAMBDA, MASTER, WADWPN, WPNGRP, WPNREG, WPNTYPE, DYNAMIC
SUBROUTINES CALLED:	None
CALLED BY:	WAD, DEFALOC, MULCON

Method

The formal parameter G specifies for which weapon group the premium is to be recomputed. The mathematics of the calculation are described in the Analytical Manual.* Three modes are provided for the computation of the premiums, depending on the value of PROGRESS.

PROGRESS < 1.0	A normal linear premium is computed which keeps the Allocator from producing allocations with unnecessarily large deviations from the desired allocation rate (statement 12).
PROGRESS = 1.0	The normal premium is augmented by a step function which strongly motivates the allocator to exactly match the stockpile (statement 1).
PROGRESS > 1.0	The subroutine exits with a zero premium for use in verification allocations (statement 10).

Since the calculation of the step function premium requires the quantity SMALLAM = .5 * LAMEF(G) for the lowest value of LAMEF, this quantity is evaluated only on the first call of PREMIUMS after PROGRESS is 1.0 (statement 18). Thereafter (since the values of LAMEF are frozen while PROGRESS = 1.0) there is no need to recompute this quantity.

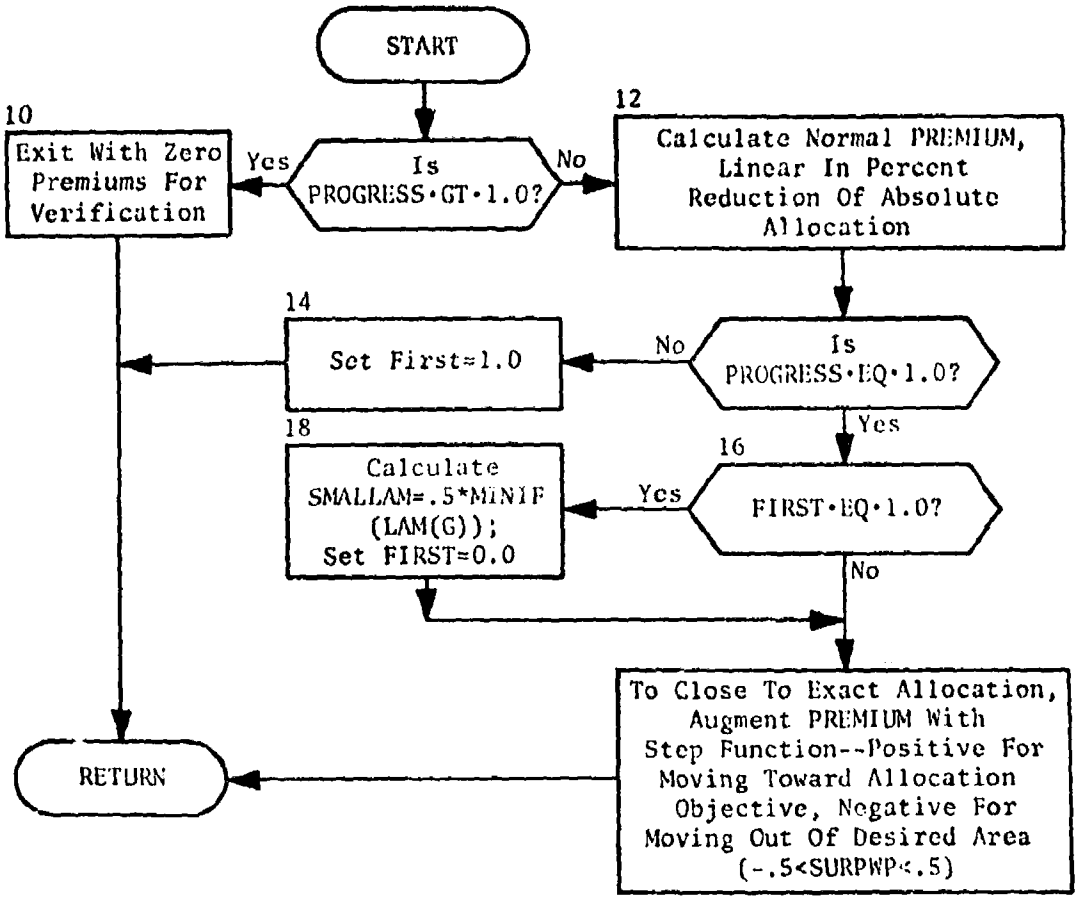
PREMIUM is subtracted from the cost of adding a weapon. DPREMIUM is subtracted from the cost of deleting a weapon.

---

* Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques (see Weapon Allocation, Closing Factors - Premiums).

Where a weapon surplus exists (i.e., SURPWP(G)>0), PREMIUM(G)> 0.0 and DPREMIUM(G)≤ 0.0. Where a weapon deficiency exists (i.e., SURPWP(G)<0), the reverse is true.

Figure 40 illustrates subroutine PREMIUMS.



Note

--Where a weapon surplus exists, (PREMIUM) is positive, and the premium for deleting (DPREMIUM) is negative.

--Where a deficiency exists (SURPWP<0), the reverse is true.

Fig. 40. Subroutine PREMIUMS

SUBROUTINE PRNTALL

PURPOSE:	This routine provides a way of calling the print subroutine PRNTNOW that is conditional on the print control flags set by PRNTCON.
ENTRY POINTS:	PRNTALL
FORMAL PARAMETERS:	IOPT - Print option number
COMMON BLOCKS:	PRNTCON
SUBROUTINES CALLED:	PRNTNOW, TIMEME
CALLED BY:	MULCON, WAD, WADOUT, GETDATA, RESVAL, DEFALOC

Method

To provide convenient control over prints in program ALOC almost all print statements are contained in subroutine PRNTNOW. They are activated by calling PRNTNOW(IOPT) for the appropriate print option IOPT. If it is desired to place the print under data-input control so that the print will not appear unless a specific print request is included in the data deck, this can be accomplished by calling PRNTNOW via a call on PRNTALL. PRNTALL executes the request on PRNTNOW only if the print control subroutine PRNTCON has set the corresponding print control flag IDO(IOPT) active (i.e., = 3).

For each call PRNTALL first checks to see if the print has been set active by PRNTCON. If not, it immediately RETURNs (statement 2) to minimize time wasted on inoperative print calls.

If the particular print is active, PRNTALL immediately calls TIMEME (statement 1) to stop the clock which records active time in the program. This makes it possible to do a test run with an unusual number of prints and still obtain a valid estimate of what the running time would be without such prints. After the call on PRNTNOW (statement 300), PRNTALL reactivates the clock before returning to the main program.

Before each print option (except 3, 18, and 26), PRNTALL prints a heading identifying the optional print (statement 450).

Subroutine PRNTALL is illustrated in figure 41.

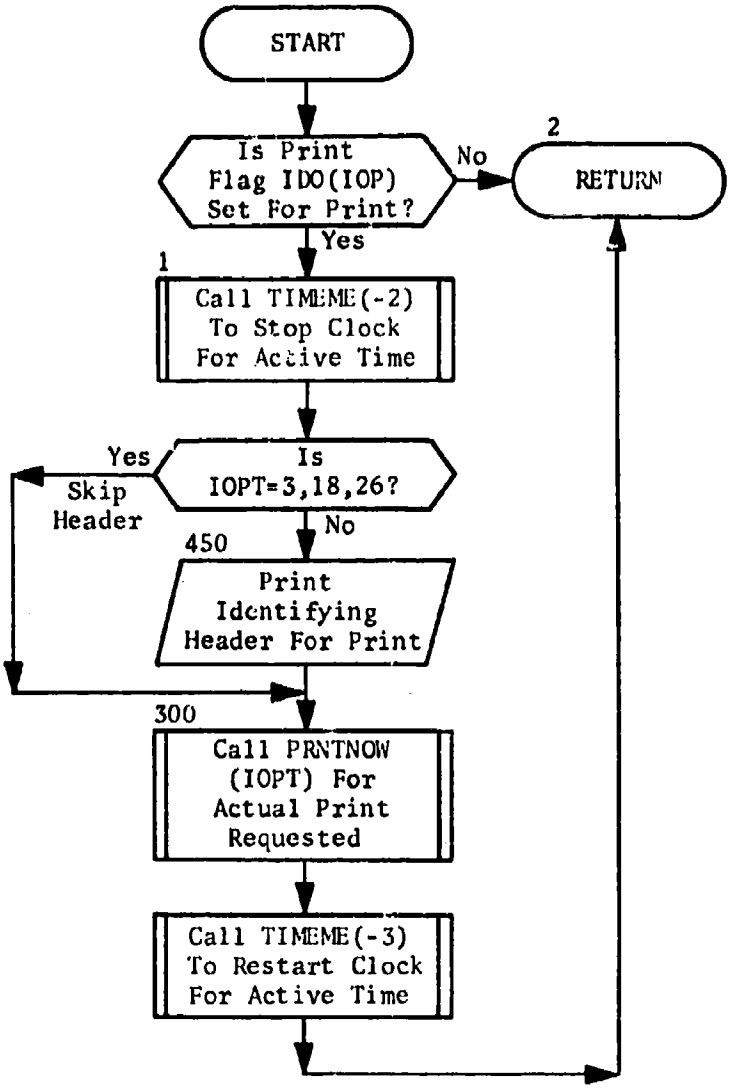


Fig. 41. Subroutine PRNTALL

SUBROUTINE PRNTCON

PURPOSE: This routine sets the print control flags which determine whether a given print request made through PRNTALL will be executed (either by the filehandler or PRNTNOW).

ENTRY POINTS: PRNTCON

FORMAL PARAMETERS: None

COMMON BLOCKS: PRNTCON, CONTROL, DYNAMIC, UFTPRNT

SUBROUTINES CALLED: None

CALLED BY: MULCON

Method

The input arrays for the print requests were read in subroutine RDALCRD. These arrays are:

INDEXPR	The index to the print requested
JPASS	The first pass (value of NPASS in /CONTROL/) on which the request is to operate
LPASS	The last pass on which the request is to operate
JTGT	The first target (value of ITGT in /DYNAMIC/) on which the request is to operate on each pass
LTGT	The last target on which the request is to operate
KTGTFREQ	The frequency with which the print is to operate (e.g., KTGTFREQ = 5 implies every fifth target). In the case of filehandler prints this is interpreted as number of words to print from each end of the file buffer.

PRNTCON is called by MULCON before proceeding to process each new target. PRNTCON first reinitializes all print control flags (IDO(INDEXPR) for regular prints and IFTPRNT(INDEXPR - 100) for FILEHANDLER prints) to a nonprint state (IDO = 1, IFTPRNT = 0). It then examines the list of print requests to see if any are operative for this target on this pass. For each operative print the flags are set to print (IDO = 3, IFTPRNT = KTGTFREQ).

This arrangement makes it possible to request the same print at different targets or passes with separate independent print requests. In the case of file prints, if more than one request on the same LTN are simultaneously operative, the last request will control the number of words printed. If regular prints are requested with KTGTFREQ greater than 1, the first print will not occur at JTGT but at KTGTFREQ - 1 targets later, and thereafter the print will occur every (KTGTFREQ)th target.

Figure 42 illustrates subroutine PRNTCON.

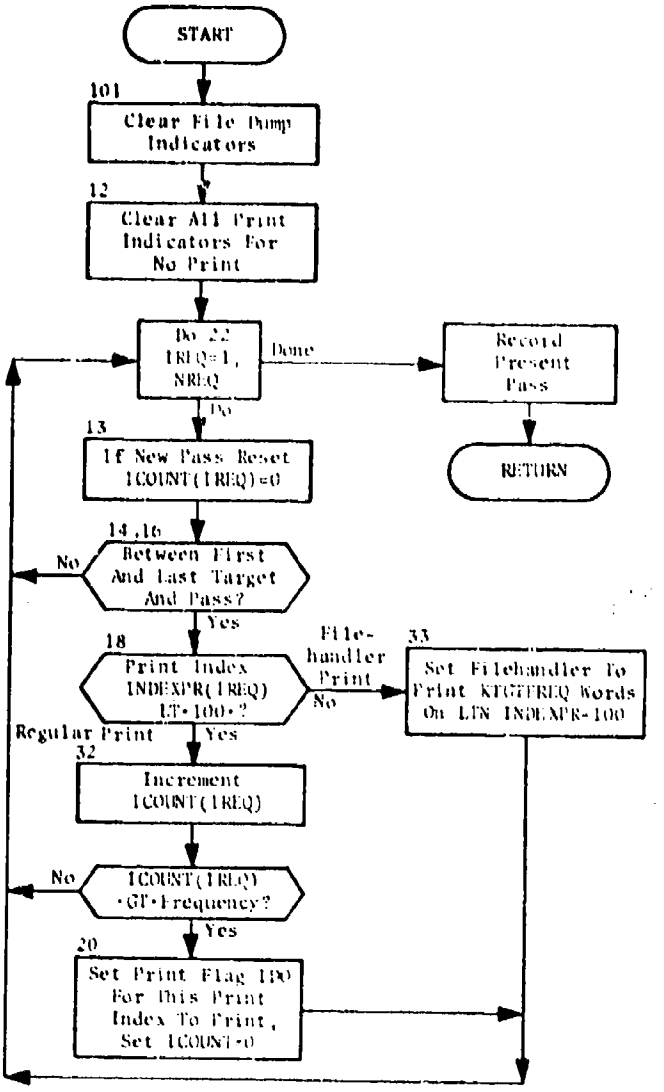


Fig. 42. Subroutine PRNTCON

SUBROUTINE PRNTNOW

PURPOSE:	This subroutine contains all the optional prints for the ALLOCATE function of program ALOC.
ENTRY POINTS:	PRNTNOW
FORMAL PARAMETERS:	IOPT - Print option number
COMMON BLOCKS:	MASTER, CONTROL, DYNAMIC, WPNREG, WPNTYPE, WPNGRP, LAMBDA, MULADJ, 222, NALLY, PAYOFF, PRINEED, PRIN1, PRNTWAD, PRTMULL, 333, WADFINAL, WADOUT, WADWPN, FORMATS, DEFENSE, LOCDEF, PEN
SUBROUTINES CALLED:	FORMATS, TIMEME, ABORT
CALLED BY:	PRNTALL, MULCON, PUNCHM

Method

The formal parameter IOPT determines which section of this print routine is to be exercised. The result of the 28 alternatives are illustrated in the User's Manual.* The subroutine requires almost all the common blocks in the program because it is responsible for prints throughout the program that ordinarily would be local to each subroutine.

Subroutine PRNTNOW is illustrated in figure 43.

---

* Volume II, Chapter 3, Plan Generation Subsystem, Program ALOC, Output, ALLOCATE Function - Weapon Allocation.

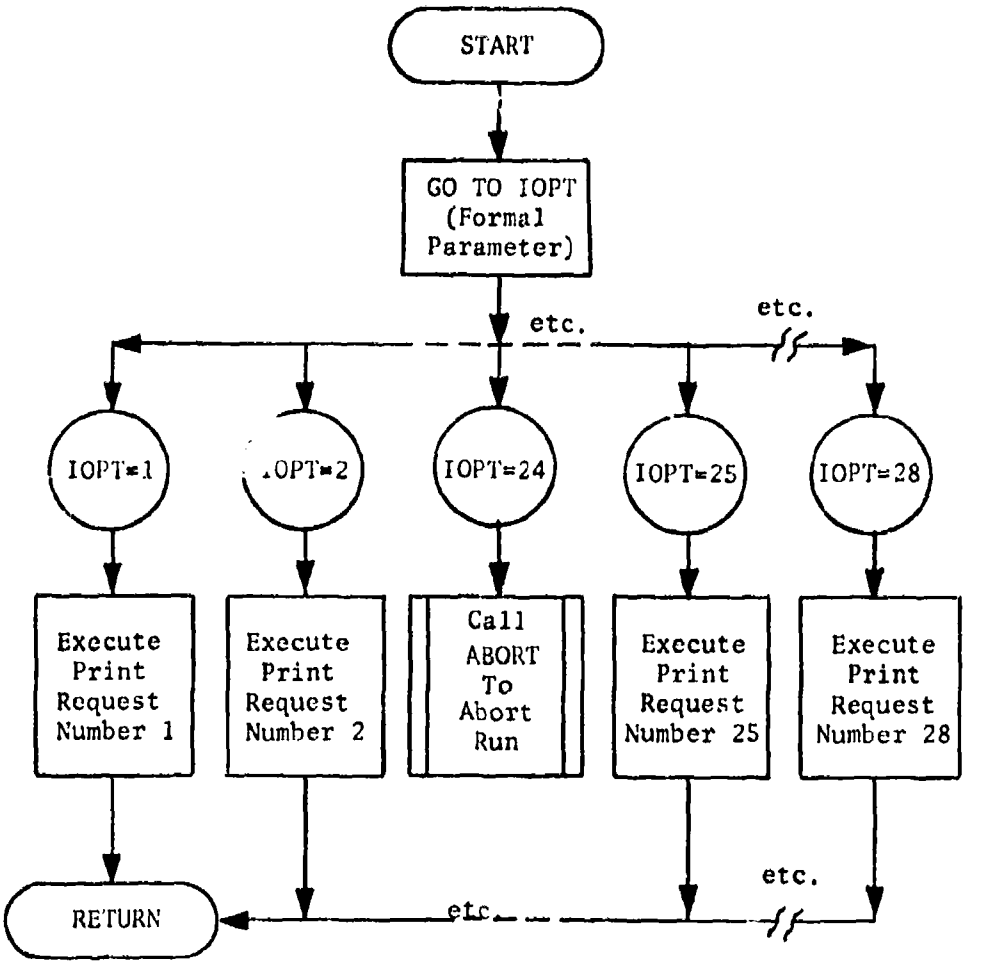


Fig. 43. Subroutine PRNTNOW

SUBROUTINE PUNCHM

PURPOSE:	This routine punches and prints the values of the local Lagrange multipliers for the PUNCH option.
ENTRY POINTS:	PUNCHM
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, 222, MACHINE
SUBROUTINES CALLED:	PRNTNOW
CALLED BY:	ALOC

Method

This subroutine is called for the PUNCH option. It merely prints and punches the values of the local Lagrange multipliers in array LA of common 222. These local multipliers represent the multipliers for each weapon characteristic (group, region, class, type, and alert status). The cards are punched in a format that can be read by subroutine READMUL for the READMUL option. The format is displayed in the User's Manual.*

The subroutine proceeds through a series of Do loops which print and punch each multiplier. Each multiplier is identified by its name (ALL, GROUP, REGION, CLASS, TYPE, and OTHER) and its order within the name. (The ALL multiplier is used for every weapon. It is stored in the second location of LA. There is no multiplier stored in LA(1) which is used for temporary storage in the ALLOCATE function.)

At the end of PUNCHM, subroutine PRNTNOW is called to print the final values of the product of the multipliers for each group. Since the PUNCH option must follow the ALLOCATE function, these values were the last values used in allocating weapons. They may be input via the READMUL option into a later run of program ALOC to speed convergence during the ALLOCATE function.

Figure 44 illustrates subroutine PUNCHM.

---

* Volume II, Chapter 3, Plan Generation Subsystem, Program ALOC, Input, READMUL Option - Read Initial Values of Lagrange Multipliers.

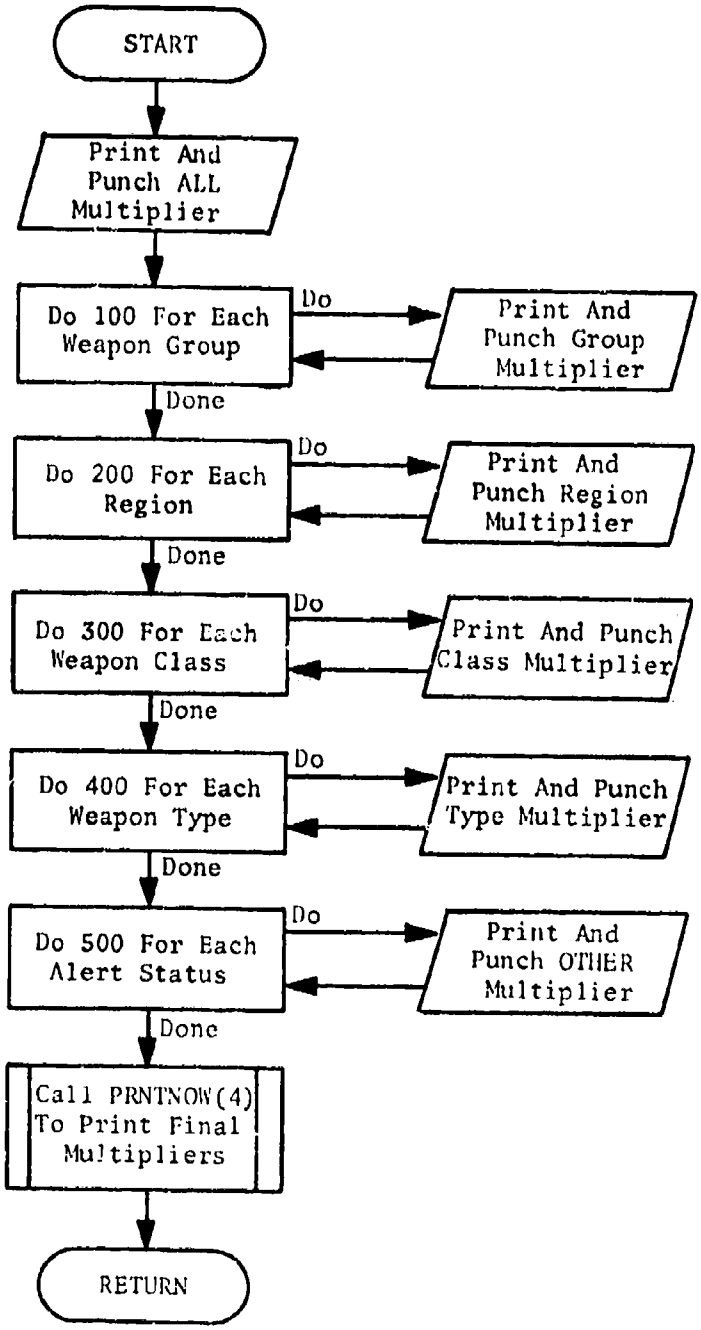


Fig. 44. Subroutine PUNCHM

SUBROUTINE RDALCRD

PURPOSE:	This routine reads and processes the user-input parameter cards for the ALLOCATE function.
ENTRY POINTS:	RDALCRD
FORMAL PARAMETERS:	None
COMMON BLOCKS:	FILES, FILABEL, MYLABEL, ITP, MYIDENT, NOPRINT, TWORD, LOCFIL, MACHINE, WADWPN*, CONTROL, MULADJ, PRNTCON, PRTMULL, DEFENSE, IFTPRNT, FIXED, FIXEDASS, SMAT
SUBROUTINES CALLED:	GETVALU, ITLE, NUMGET, SETREAD, RDARRAY
CALLED BY:	MULCON

Method

This subroutine reads the user-input cards for the ALLOCATE function, except for the fixed assignment specification cards which are read by MULCON and DEFALOC. The format of the user-input parameter cards is free field as described in the User's Manual.** Utility subroutine GETVALU is used to convert the input information into variable-value format.

The user-input parameters recognized by RDALCRD are as follows:***

1.	IMATCH = 0	If IMATCH ≠ 0, weapons will be allocated to the target until the designated MINKILL is achieved, not considering time-dependent target value.
2.	RINTPRD = 2.0	Approximate ratio between rate of change of target weights between different integration periods.

---

* Common block WADWPN used as an input buffer.

** Volume II, Appendix E, Free-Field Format.

*** The number preceding the parameter name is its position in the parameter name list. The value following the parameter is the default value.

3. RATIOINT = 2.0 — Ratio of longest integration period used to the theoretical -- a low value allows higher sensitivity without oscillations, but too low a value makes convergence sensitive to statistics of target list.

4. SNSTVTY = 0.1 — Controls sensitivity of multiplier adjustment during early phases. Too high a sensitivity can cause oscillations in multipliers.

5. FSNSTVTY = 1.0 — Controls sensitivity of multiplier adjustment during latter part of allocation.

6. CLOSE = 1.05 — Must be greater than 1.00. Excess over 1.0 determines magnitude of closing force relative to Lagrange multipliers at start of closing phase (PROGRESS = 1.0).

7. DELTVAL = .005 — The maximum fractional difference in time-dependent target value permitted in the same time-of-arrival cell. (Will be automatically increased if available cells are exceeded. A high value allows slightly faster operation; a low value increases accuracy of time-of-arrival calculations.)

8. PRM = .5 — Controls value of quadratic premium before PROGRESS = 1.0. Must lie between 0 and 1.0. Higher values give more stable performance.

9. STALADJ = .5 — Determines extent to which STALL favors high-unit profit vs. efficiency in selecting weapons for initial laydown on each target. Should be adjusted to minimize IOPS for run so long as it does not adversely affect total payoff.

10. CLOSER = 4.00 — Controls rate of increase in CLOSE or closing force per pass over target system. High values will close allocation to exact stockpile more rapidly but will cost more in payoff to do so.

11. QUALITY = 0.5 — Controls extent to which STALL will attempt to refine allocation for each target. Should be set as low as possible for fast operation so long as total payoff is not reduced.

12.	CORR = .5	Controls magnitude of correlations assumed by allocator. Since assumptions of correlation treatment give an upper limit of possible correlations if CORR = 1.0, .5 is probably the highest value that should be used.
13.	TARFAC = .1	Multiplier for terminal bomber defense level (TARDEF) in terminal bomber attrition formula.
14.	IVERIFY = 0	Zero used for normal run. If verification of payoff as optimum is desired, use 1, and 2 can be used to evaluate same allocation using different value CORR2 of CORR.
15.	CORR2 = 0.0	Used as value of CORR in verification pass if IVERIFY = 2.
16.	SETTLE = 1.00	Controls number of passes at PROGRESS = .75 before PROGRESS = 1.00 and closing begins. Larger numbers give more exact multipliers. However, 1.00 usually seems quite adequate. Numbers less than 1.00 can give quite bad results if defective allocations prior to PROGRESS = .75 are not replaced in closing.
17.	BPENFAC = 1.00	Multiplies attrition rates for bomber penetration given in data base. Normally should be one, but can be used to test alternative assumptions without changing data base.
18.	ERRCLOS = .001	Factor which controls allocation termination.
19.	PKTX = .95	Probability of warhead kill by one terminal ballistic missile defense interceptor.
20.	RADPX = 0.00	Probability of warhead kill by a random area ballistic missile defense.
21.	MINDAMAG = 0.	Minimum fraction of original value of undefended target that must be destroyed by each weapon ($0. \leq$ MINDAMAG < 1.0).
22.	LOWFAC = 0.0	NTX/MISDEF for lower deviation from MISDEF.
23.	HIGHFAC = 0.0	NTX/MISDEF for upper deviation from MISDEF.
24.	PROBLOW = 0.0	Probability there are LOWFAC * MISDEF interceptors.

25. PROBHIGH = 0.0 — Probability there are HIGHFAC * MISDEF interceptors.

26. LAW = POWER — If this field contains the characters SQUAREROOT, the square root damage function will be used on area targets. Any other characters will cause use of the exponential law.

27. TINTFAC = 1.0 — This variable is a multiplier of the number of terminal ballistic missile interceptors on defended targets. It is used to increase or decrease the total number of interceptors in the target set without changing their proportional allocation. If this field is blank or 0, the terminal defenses are omitted.

28-57. SMAT* — Failure mode-attribute correlation assumption array.

58. FACMIRV = 0.0 — This is the MIRV correlation factor. It should be in the range 0.0 to 1.0. The program takes this factor and uses it as a multiplier of the INDEPENDENT attribute of the SMAT array for the failure modes survival before launch, SBL, command and control reliability, CC, and system reliability, REL. The resulting products are added to the GROUP attributes for their respective failure modes for MIRV systems only. The effect of this parameter is to increase intragroup correlation for MIRV groups in these failure modes. A value of 0 means no increase in the intragroup correlations. A value of 1.0 means the maximum intragroup correlation. This factor has no effect on the correlations for nonMIRV weapons.

In addition there are 28 optional print requests. These requests are described in the User's Manual.**

At the end of reading the changes to the above user-input parameters for this option, the use of the fixed assignment capability must be described by the user.

---

* See User's Manual, Volume II, Chapter 3, Plan Generation Subsystem, Program ALOC, Input, ALLOCATE Function - Weapon Allocation for default values.

** Volume II, Chapter 3, Plan Generation Subsystem, Program ALOC, Output, ALLOCATE Function - Weapon Allocation.

In order to provide more flexibility in the fixed assignment capability, there are three subcommands to control this capability. One of these commands must be present at the end of the input parameter data. The subcommands are:

FIXTGT: This command will cause the fixed assignment data to be read from the TGTFILE. No further fixed assignment information may be placed in the rur deck.

NOFIXES: This command will prevent the use of the fixed assignment capability. Its use will be to force the program to ignore any fixed assignment information on the TGTFILE. Of course, no further fixed assignment information will be needed in the run deck.

FIXNOW: This command specifies that the fixed assignment information is to be read at execution time of program ALOC. The fixed assignment data cards are in the same format as those used in program PREPALOC, except that the target identifier must be the target number, and the cards must be in order of increasing target number. This subcommand may be modified by the following optional subcommands:

TAPE:	Specifies a tape in filehandler format as input medium for fixed assignment data
BCDTAPE:	Specifies a BCD card image tape as input medium for fixed assignment data
DISK:	Specifies a disk file in filehandler format as input medium for fixed assignment data
NOSAVE:	Specifies that no fixed missile information is to be placed on the MSLTIME file. The file will always be created normally in the absence of this command.
ADDONLY:	Specifies that the fixed assignment data read by program ALOC is to be used in addition to the data on the TGTFILE.

The use of the ADDONLY command involves the merging of two streams of data. The weapons fixed by the TGTFILE data will be first allocated to the target. Then, the weapons fixed by the immediate input medium will be added. The same checks performed on the fixed assignment data in program PREPALOC will also be performed on the immediate data stream. In addition, both sets of data will be checked to insure that no fix violates range constraints or other restrictions.

This routine's processing is quite straightforward. First the default values are loaded into the parameters and the standard print requests. Statement 100 begins a loop which reads, processes, and prints each card. The loop is exited to statement 2000 when either:

1. The input card has no parameters
2. The first parameter on the card is a fixed assignment command (statement 110)
3. The termination parameter; i.e., $, (see subroutine GETVALU) appears.

For each parameter on the card, the name list, MYNAME in common /WADWPN/, is searched. If a match is found, the input value is loaded (statements 150, 151, 152, 153). The computed GO TO statement following statement 150 uses the MYGOTO array in common /WADWPN/. This array has an element for each parameter and the value of MYGOTO specifies the parameter input format as follows:

1. Implies floating point numeric format (real variables)
2. Implies fixed point numeric format (integer variables)
3. Implies BCD code format (alphameric variables).

Print requests, cancellations, and modifications are simply processed according to the methods described in the User's Manual.* A print request is processed starting at statement 200 where a test is made to determine if the request is a valid option. If not, the request is ignored. The request is also ignored if more than MPRNT (=40) requests have been read (statements 220, 230). If there is room for a valid request, it is processed at statement 240. A print cancellation is processed starting at statement 300 by determining if it is in range (statement 300) and the requested option is active (statement 320). If both conditions are met, the print is deactivated in statement 340. Print modification information (determined at statement 130) is simply loaded at statement 280.

When all the input cards have been read (statement 2000) the values are printed and loaded into the appropriate common blocks. If the input deck was not terminated by a fixed assignment subcommand, this command is now read and printed at statement 3000. Subroutine GETVALU is used to process the subcommands.

The subcommands modify the following variables in common blocks /FIXED/ and /FIXEDASS/:

---

* Volume II, Chapter 3, Plan Generation Subsystem, Program ALOC, Input, ALLOCATE Function - Weapon Allocation.

IFTGT - Target number of first target on which fixes will be input from card reader

IADD - Positive only for ADDONLY option

SAVEFIX - Fixed missile information saved only if value is .TRUE. (Set .FALSE. only by NOSAVE or NOFIXES command.)

IFIXTAPE - ITP value for input fixed assignment file in Filehandler format (TAPE or DISK commands)

IBCD - Logical unit number for input fixed assignment file on BCD input tape (BCDTAPE command).

At the end of the option processing, if the fixed assignment command is FIXNOW, the first input card for fixed assignment is read (statement 5000).

Subroutine RDALCRD is illustrated in figure 45.

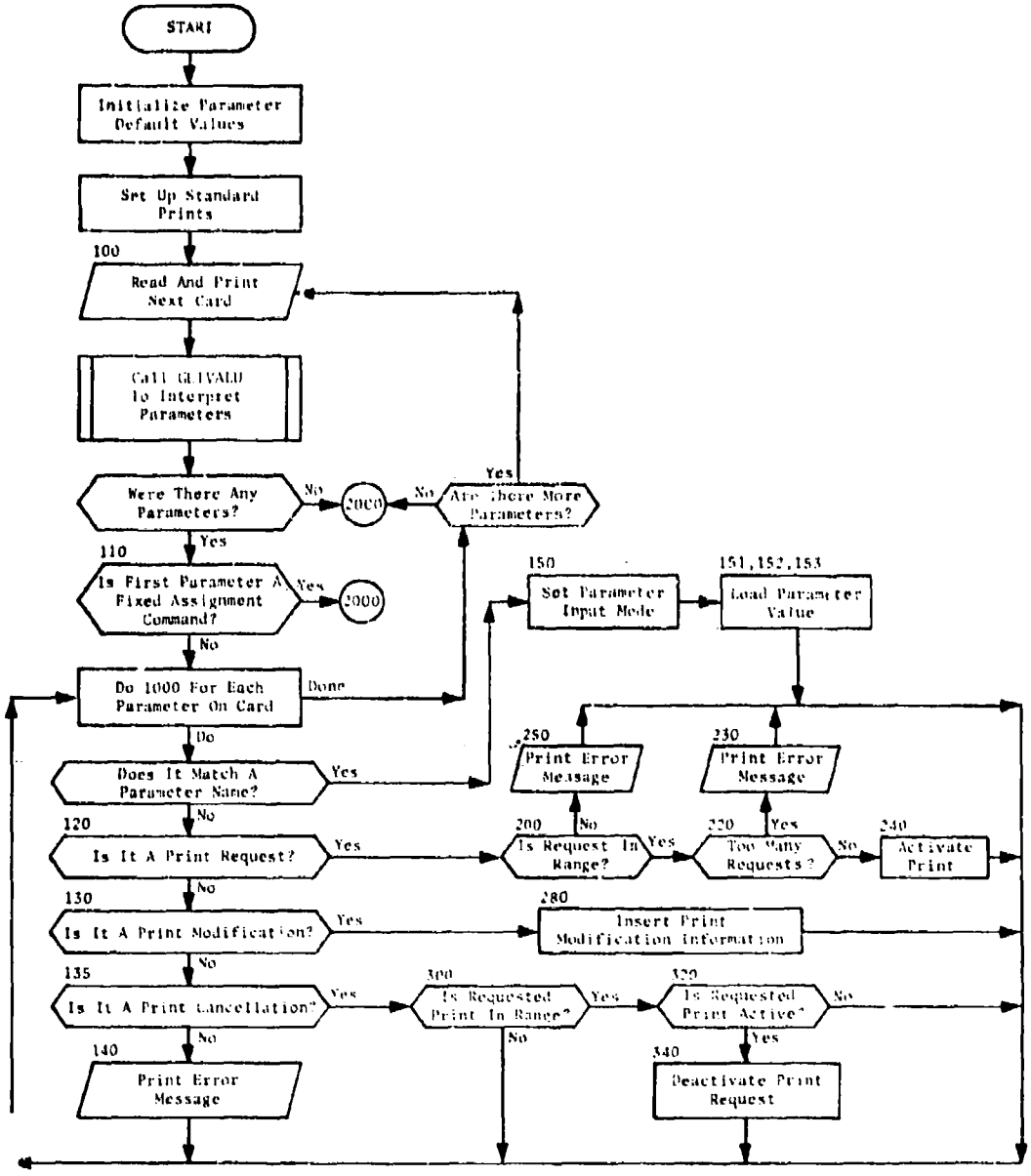


Fig. 45. Subroutine RDALCRD
Part I: Input Card Processing

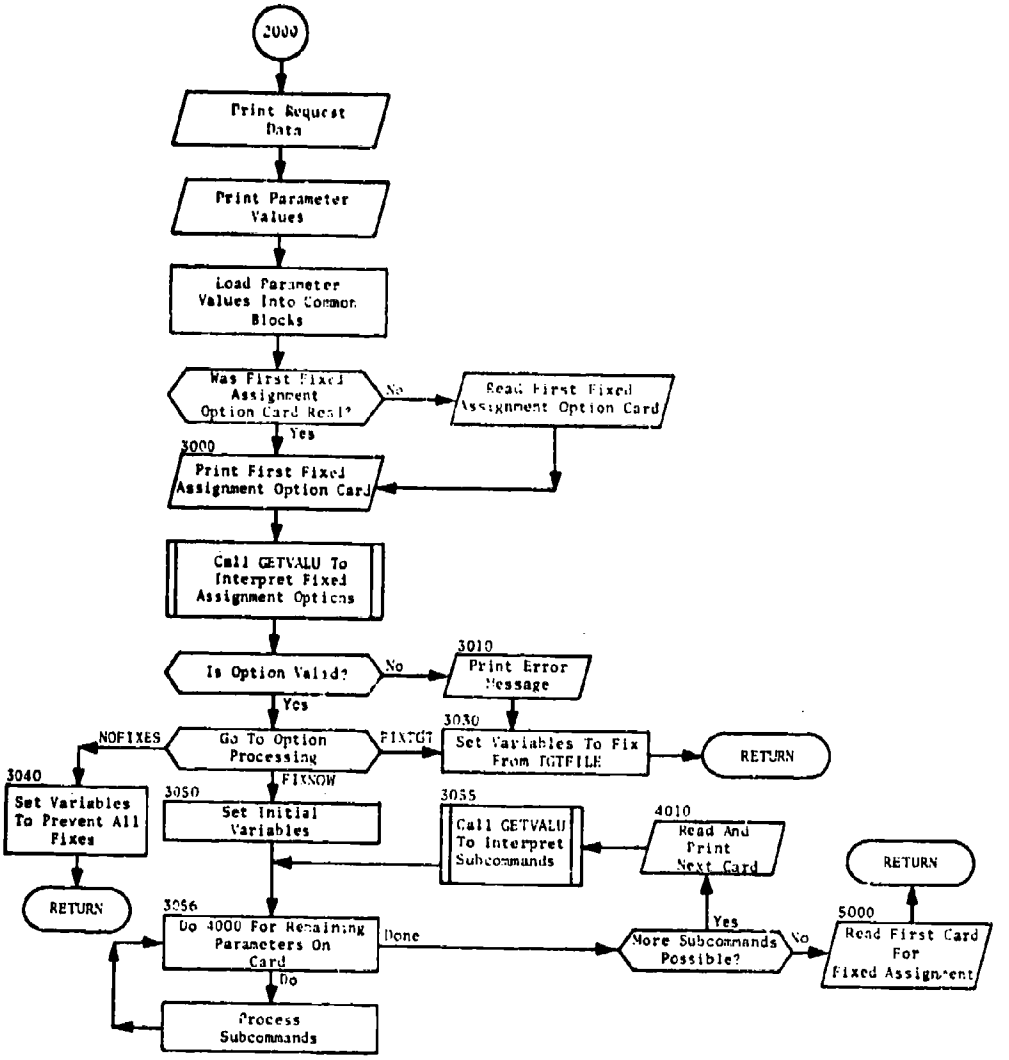


Fig. 45. (cont.)
Part II: Post Input Processing

SUBROUTINE READMUL

PURPOSE:	This routine processes the user-input initial settings for the Lagrange multipliers in the READMUL option.
ENTRY POINTS:	READMUL
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, 222, MACHINE
SUBROUTINES CALLED:	None
CALLED BY:	ALOC

Method

This routine reads and processes the user-input parameter cards for the READMUL option. The format of these cards is displayed in the User's Manual.*

The operation of the routine is straightforward. Each card is read and printed (statement 100). If it reads END LAM the routine exits with no further processing (statement 1000).

Otherwise the first field on the card is tested for a match on ALL, GROUP, REGION, CLASS, TYPE, or OTHER (statements 200, 220, 240, 260, 280, 300). If the first field is none of these, an error message is printed (Statement 1100) and the next card is read (statement 100). If there is a match on the first field, the correct index to the placement of the local multipliers is computed (statements 210, 230, 250, 270, 290, 310). The first multiplier, ALL, occupies position 2. (Position 1 is used for temporary storage in the ALLOCATE function.) The succeeding positions are occupied of the GROUP, REGION, CLASS, TYPE, and OTHER multipliers in that order. After the index, LOU, is calculated, the input value is loaded into the local multiplier at statement 900. Control then returns to statement 100 for the reading of another card.

Subroutine READMUL is illustrated in figure 46.

---

* Volume II, Chapter 3, Plan Generation Subsystem, Program ALOC, Input, READMUL Option - Read Initial Values of Lagrange Multipliers.

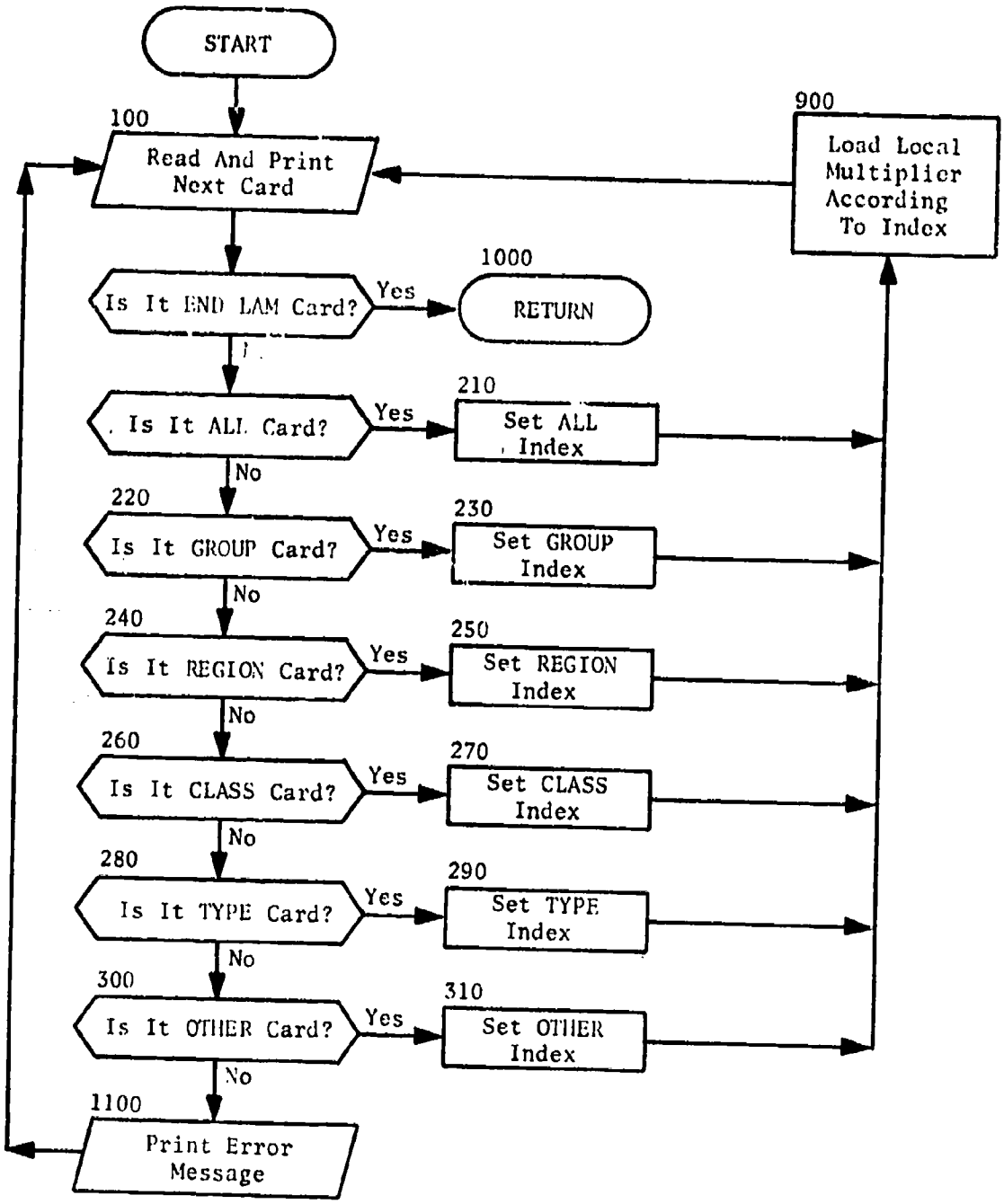


Fig. 46. Subroutine READMUL

SUBROUTINE RECON

PURPOSE:	This routine reconstructs the weapon target interaction data from the compressed format used on the WPNTGT files. Entry SETUP is used in the first pass when no decompression is necessary.
ENTRY POINTS:	RECON, SETUP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	WADWPN, DYNAMIC, WPNTYPE, WPNGRP, WPNREG, PAYLOAD, CONTROL, DEFENSE, PKNAVAL, SMAT, PAYOFF
SUBROUTINES CALLED:	TABLEMUP, TIMEME
CALLED BY:	GETDATA

Method

The first part of this routine is used only after pass one. Before writing the weapon target interaction data on the WPNTGT files, subroutine GETDATA compresses the data by removing spaces reserved for inactive groups. If the target has only one hardness component, space reserved in the STK and STK2 arrays in common /WADWPN/ for the second component is also removed. The first part of RECON, therefore, merely decompresses the data. The VTOA and XMUP arrays of common /WADWPN/ are used for temporary storage for the STK and STK2 data, since these later arrays may be overwritten by the data input from the WPNTGT files.

Entry SETUP is used on all passes (but not explicitly on passes after the first). The MUP, XMUP, SSIG, and RISK arrays are constructed according to the formulae presented in the Analytical Manual, Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques, Weapon-Target Interaction.

The processing in statements 945 and 960 is used to load the correct SMAT array. Through the use of the user-input parameter FACMIRV, the SMAT array may differ for MIRV and nonMIRV weapons.

In statement 1925 the data base attribute PKNAV is used for the single shot kill probability for weapons which have nonzero values for the attribute.

Subroutine RECON is illustrated in figure 47.

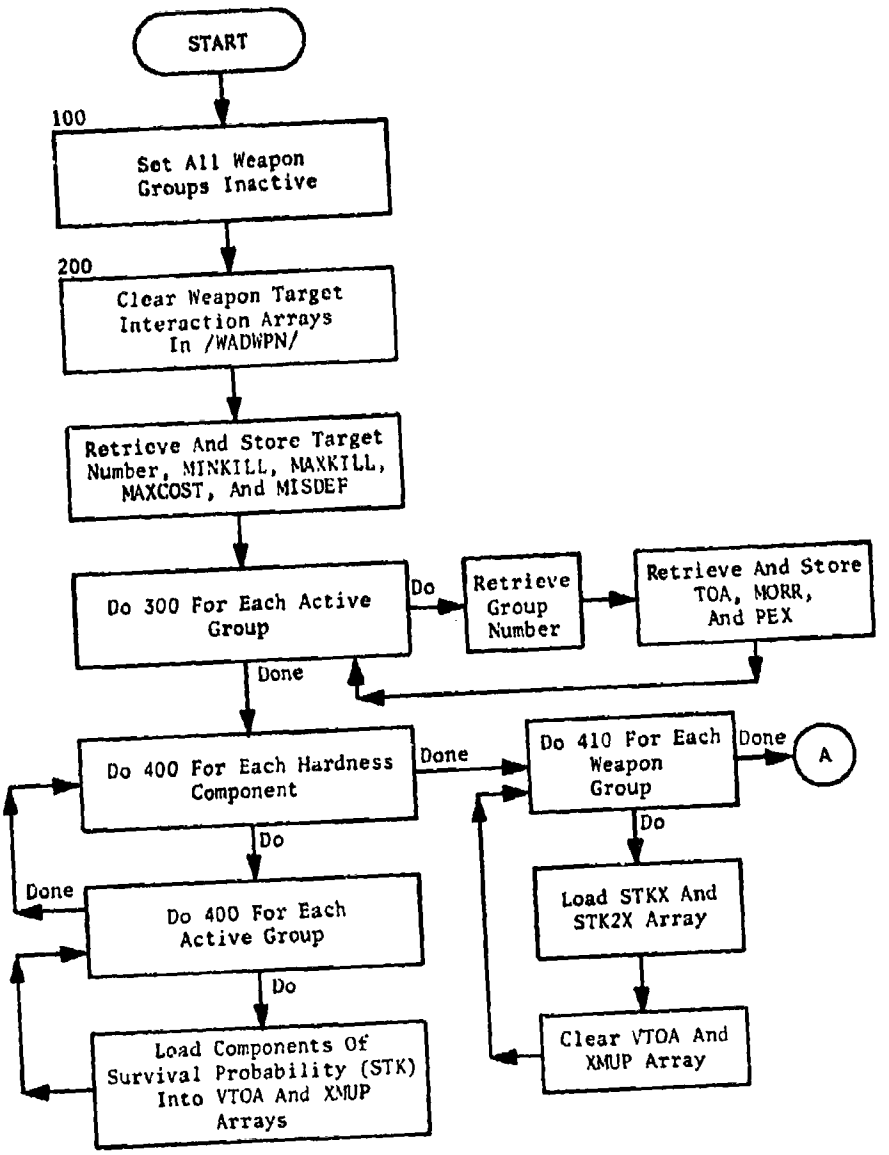


Fig. 47. Subroutine RECON
Part I: Entry RECON

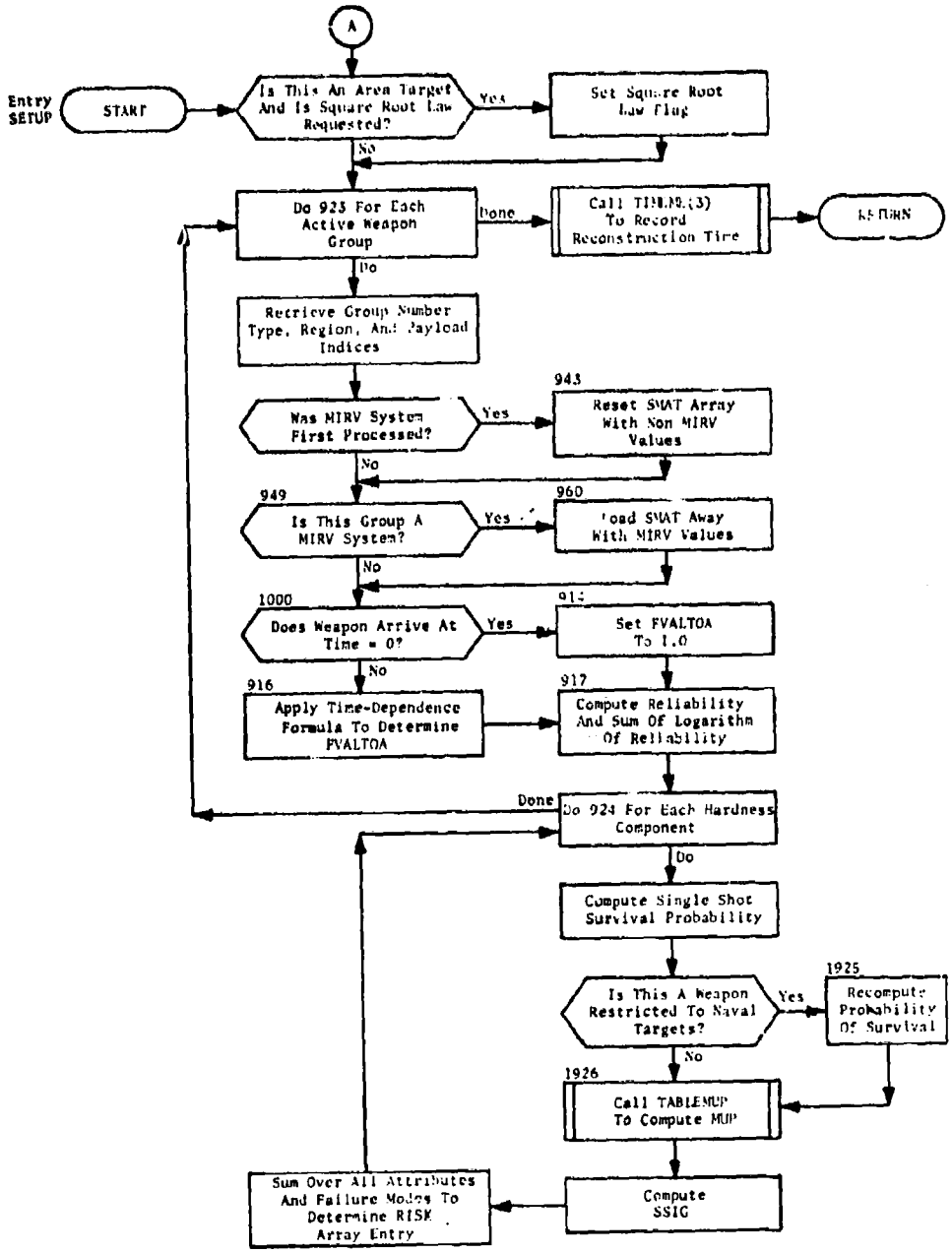


Fig. 47. (cont.)
Part II: Entry SETUP

SUBROUTINE RESVAL

PURPOSE:	This routine calculates the surviving value of a target defended with terminal ballistic missile interceptors when attacked with missiles with or without penetration aids.
ENTRY POINTS:	RESVAL
FORMAL PARAMETERS:	None
COMMON BLOCKS:	WPNREG, WPNGRP, DEFENSE, WADWPN, LAMBDA, WPNTYPE, MASTER, DYNAMIC, PAYLOAD, LOCDEF
SUBROUTINES CALLED:	FMUP, TABLEMUP, PRNTALL
CALLED BY:	DEFALOC

Method

RESVAL first orders the weapons by time of arrival on the target and then computes the total number of expected terminal objects contained in the weapons specified by NOWEP(G) (common /LOCDEF/). (NOWEP(G) = number of weapons of group G allocated.) The single shot survival probability of the target from a weapon from group G on hardness component J is equal to the previously calculated XMUP(G,J) (common /WADWPN/). This survival probability must be modified for multiple weapon attacks. As the number of attackers exceeds the number of defenders, the single shot survival probability will decrease.

Three different levels of terminal interceptors (NTX(I)) are calculated for each defended target, and a probability of the occurrence of each is given by PX(I),* such that

$$\sum_{I=1}^{3} PX(I) = 1 \ .$$

*PX(1) is the user-input parameter PROBLOW. PX(3) is the user-input parameter PROBHIGH. PX(2) = 1-PX(1)-PX(3). RX(1) is the user-input parameter LOWFAC. RX(2) is the user-input parameter HIGHFAC.

These values are calculated from MISDEF, [illegible]nal number of terminal interceptors at the target, as follows:

NTX(1) = MISDEF * RX(1)*

NTX(2) = MISDEF

NTX(3) = MISDEF * RX(2)*

The probability that a warhead from the weapon G is killed by the terminal defense is then given by:

$$\left.\begin{aligned} PWK(I) &= PKTX * XDEG(G) \quad \text{if NOBJ} \leq NTX(I) \\ &= \frac{NTX(I)}{NOBJ} * PKTX * XDEG(G) \quad \text{if NOBJ} > NTX(I), \end{aligned}\right\} \text{for } I=1,2,3$$

where NOBJ is the number of warheads plus decoys in the attack and the XDEG factor (common /PAYLOAD/) degrades PKTX for weapon group G. Hence, the probability that target component J survives NOWEP(G) weapons from group G is given by a calculation involving the use of the functions TABLEMUP and FMUP, which are described in other sections of this chapter.

The former function takes as input the modified single shot survival probability,

MSSSP(G,J,I) = PWK(I) + ((1-PWK(I))*XMUP(G,J))

and computes the kill factor,

KF(G,J,I) = TABLEMUP(MSSSP(G,J,I)).

The kill factors for all the weapons allocated to the target from each group are summed to generate the group total kill factor,

GTKF(G,J,I) = KF(G,J,I)*NOWEP(G)*NWHD(G).

(NWHD(G) is number of warheads per weapon from group G.) This factor is input to the function FMUP to generate the probability that target component J survives NOWEP(G) weapons from group G; i.e.,

S(J,G,I) = FMUP(GTKF(G,J,I)) .

---

*PX(1) is the user-input parameter PROBLOW. PX(3) is the user-input parameter PROBHIGH. PX(2) = 1-PX(1)-PX(3). RX(1) is the user-input parameter LOWFAC. RX(2) is the user-input parameter HIGHFAC.

Hence, the total surviving target value is calculated from:

$$\text{Surviving Target Value} = \sum_{I=1}^{3} PX(I) \left\{ \sum_{J=1}^{M} \sum_{N1=0}^{NN} [VTOA(N1,J) - VTOA(N1+1,J)] * \prod_{G=1}^{N1} S(G,J,I) \right\}$$

where

VTOA(N1,J) = value of component J when weapon N1 arrives

NN = total number of weapon groups

VTOA(0,J) = VO(J) = value of hardness component J

and

VTOA(NN + 1,J) = 0.

The innermost sum over N1, the weapon groups, must be carried out in order of the weapons' time of arrival; i.e., the first term corresponds to the N1 with shortest time of arrival, etc.

Hence the residual target calculation in RESVAL takes into account (1) uncertainties in the terminal interceptor stockpile, (2) target value dependence on time, (3) multiple hardness components of the target, (4) various penetration aids and decoy capabilities of attacking weapons, and (5) a detailed target-warhead interaction calculation.

This apparently complicated manner of calculating the target survival probability is required by the optional use of two damage laws. The functions TABLEMUP and FMUP determine which damage law is being used on the current target and modify their calculations accordingly. Since subroutine RESVAL is called a very large number of times for each missile-defended target, certain intermediate results are not saved in order to decrease execution time. In particular, the variables for the modified single shot survival probability, MSSSP(G,J,I), the kill factor, KF(G,J,I), and the total survival probability, S(G,J,I), are never explicitly saved. The group total kill factor, GTKF(G,J,I), is saved in a temporary storage variable, S. Thus, these four intermediate variables do not appear explicitly in the program.

Subroutine RESVAL is illustrated in figure 48.

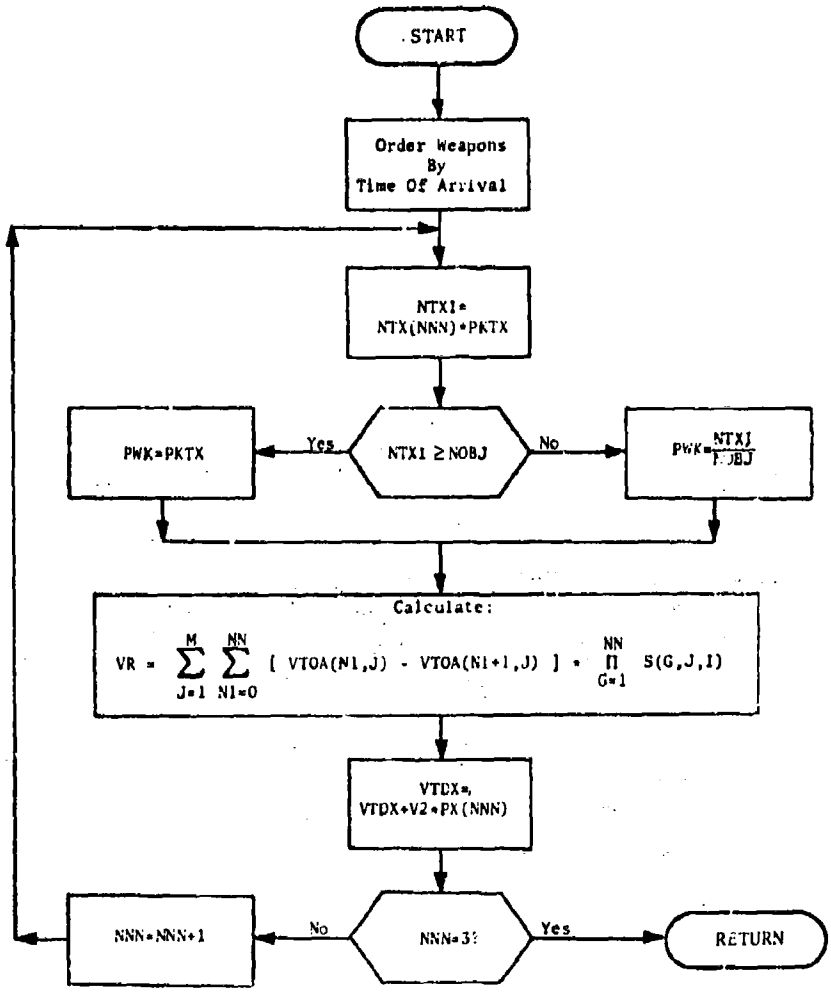


Fig. 48. Subroutine RESVAL

SUBROUTINE RNGEMOD

PURPOSE:	This routine changes range (RANGE) and refueled range (RANGEREF). Entry MINRNGE sets a minimum range (RANGEMIN) by weapon group. These new ranges are used by program ALOC in assigning targets to weapons. The minimum range specifies the minimum distance allowed between group centroid and target.
ENTRY POINTS:	RNGEMOD, MINRNGE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	WPNTYPE, RANGE, WADWPN*, WPNGRP
SUBROUTINES CALLED:	NUMGET
CALLED BY:	ALOC

Method

The range data produced by this subroutine is processed by subroutine GETDATA. Definitions of variables passed to GETDATA through common /RANGE/ are as follows:

RANGEMUL(G)	Range multiplier for group G
RANRFMUL(G)	Refueled range multiplier for group G
RANGEMIN(G)	Minimum range (nautical miles) for group G

This routine allows the user to change the range of all vehicles in any specific weapon group. These new ranges will be used only during weapon allocation in program ALOC. Later programs (i.e., POSTALOC, FOOTPRNT, PLNTPLAN) will use the range specified in the data base.

The user inputs two parameters, RANGEMUL and RANRFMUL for each group for which he wants to modify range. Subroutine GETDATA will multiply the unrefueled range, RANGE(K)** (in common /WPNTYPE/), by the user specified RANGEMUL(G). The refueled range, RANGEREF(K) (in common

*Common /WADWPN/ used as an input buffer.

**K is the weapon-type index retrieved from array ITYPE in common /WPNGRP/.

/WPNTYPE/), is handled similarly. If the user does not specify range multipliers for a group, they are assumed to be equal to 1.0. These parameters are input on cards, each card specifying one set of values for G, RANGEMUL(G) and RANRFMUL(G).

This data card set is terminated by a card with a nonpositive integer value for G. If the user wishes the same multiplier for both refueled and unrefueled range, he need only specify RANGEMUL(G) and leave the RANRFMUL(G) field blank on the card. Subroutine RNGEMOD will set the refueled range multiplier equal to the unrefueled range multiplier (statement 40).

Notes:

1. If conflicting range change requests are input, the last request input will take precedence.

2. If the user wishes to remove a group from consideration for weapon allocation, a change request with both multipliers equal to 0.0 will prevent allocations from that group.

Entry MINRNGE

This entry reads cards which specify minimum ranges for groups. Each input card specifies the group number (G) and the minimum range, RANGEMIN(G). Any target that is closer to the group centroid than the minimum range for that group will be eliminated by subroutine GETDATA from consideration for weapon assignments from that group. The cards are terminated by a nonpositive group number.

Subroutine RNGEMOD is illustrated in figure 49.

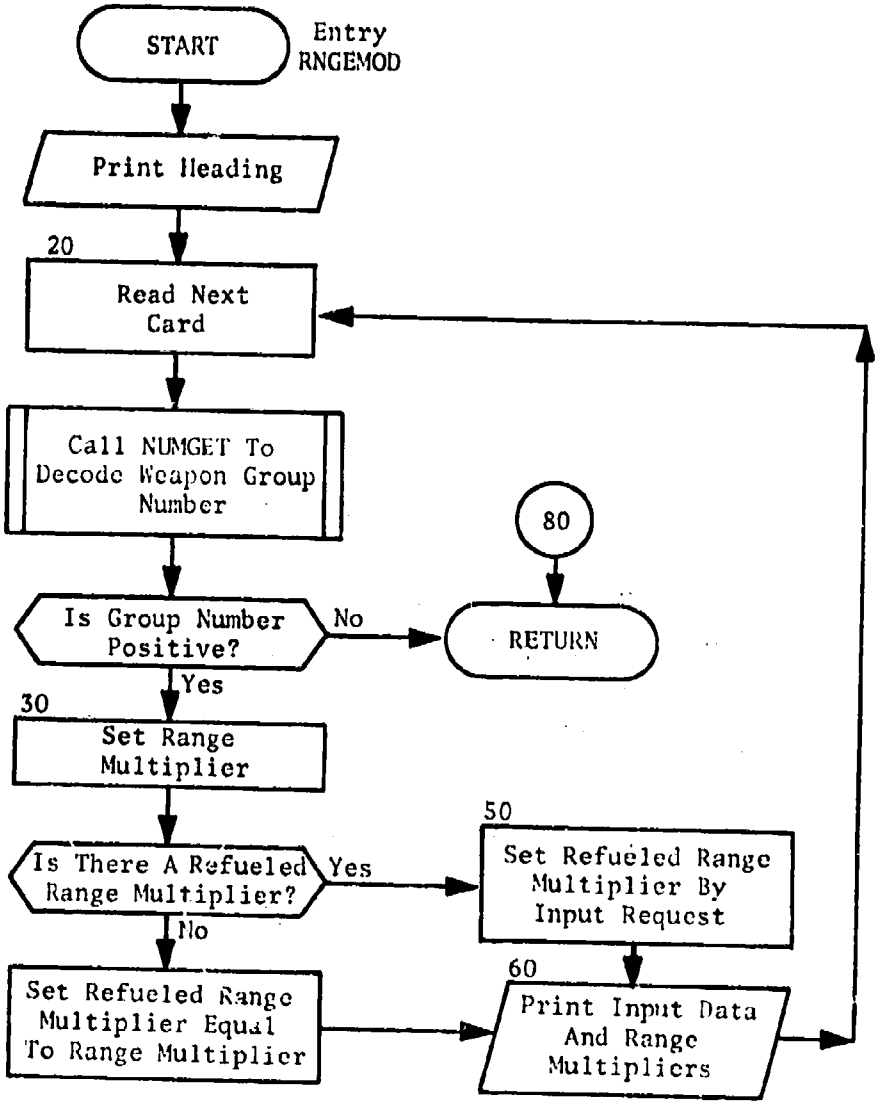


Fig. 49. Subroutine RNGEMOD
Part I: Entry RNGEMOD

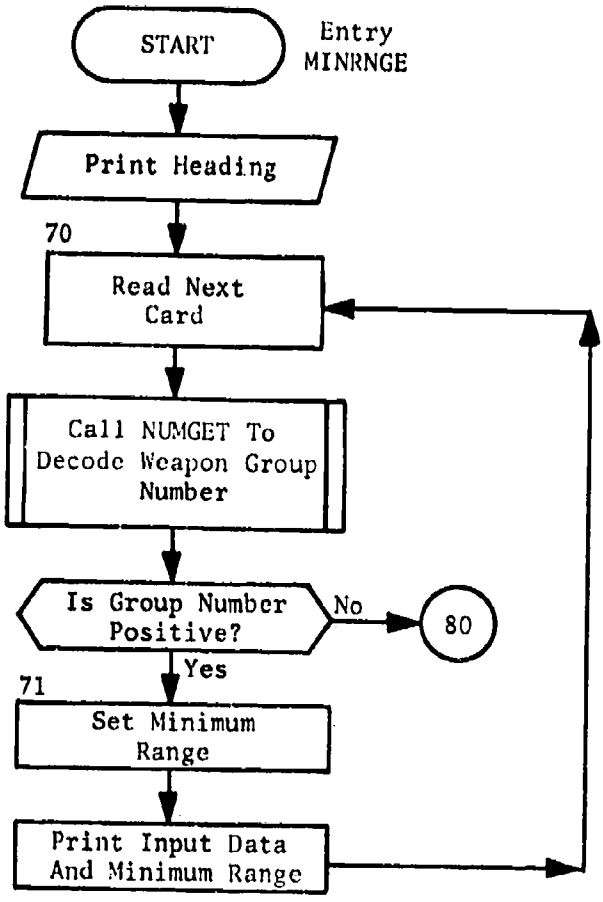


Fig. 49. (cont.)
Part II: Entry MINRNGE

SUBROUTINE SETABLE

PURPOSE:	This routine initializes the table which is used to calculate the weapon kill factors used in the square root damage law.
ENTRY POINTS:	SETABLE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	TABLE
SUBROUTINES CALLED:	None
CALLED BY:	MUICON

Method

This subroutine fills common /TABLE/ with the data needed for the square root damage law. The array, TABLE, contains values for weapon kill factors which will produce single shot survival probabilities between 0.0 and 1.0. The table entries are defined as follows:

TABLE(i) is the square root kill factor for a single shot survival probability of (i-1)* .001. There are 1001 entries in the table. Let

SSSP = Single shot survival probability
TB = TABLE array entry

Then:

SSSP = (1+TB)*exp(-TB).

During processing, function TABLEMUP will use this table to compute weapon kill factors. A simple heuristic root finder is used by SETABLE to construct the table. The procedure is as follows:

Define $x_o = 1.0$

SSSP = Input single shot survival probability

$S_i = (1+x_i)*\exp(-x_j)$ (see statement 1).

The procedure iterates on x_i such that

$$x_{i+1} = x_i + \frac{(1+x_i)}{x_i} * \text{ERR} \quad \text{(see statement 2)}$$

where ERR = $(S_i - SSSP)/S_i$.

The procedure ends when ERR<.000001. The table value is the x_i which produced the S_i such that ERR met that condition (see statement 3).

Subroutine SETABLE is illustrated in figure 50.

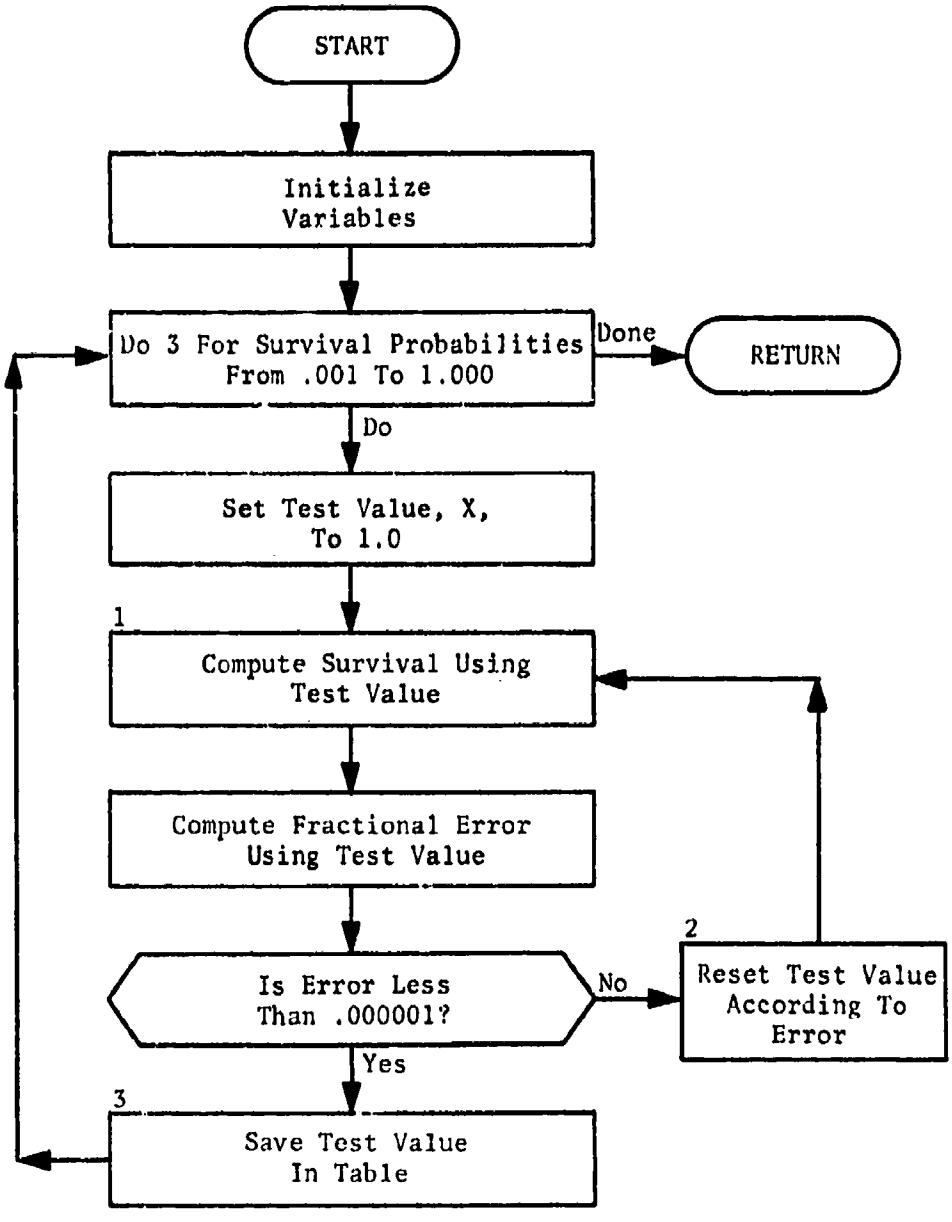


Fig. 50. Subroutine SETABLE

SUBROUTINE SORTMIS

PURPOSE:	This routine sorts the information on the ITMPMSL file in order of ascending group number. The reordered information is then output as the MSLTIME file.
ENTRY POINTS:	SORTMIS
FORMAL PARAMETERS:	None
COMMON BLOCKS:	FIXED, FIXEDASS, WADWPN, FILES, FILABEL, MYLABEL, ITP, MYIDENT, NOPRINT, TWORD, LOCFIL
SUBROUTINES CALLED:	SETREAD, RDARRAY, SETWRITE, WRWORD, WRARRAY, TERMTAPE, ORDER, REORDER
CALLED BY:	MULCON

Method

The ITMPMSL file prepared by STALL and DEFALOC contains the same information as the MSLTIME file. The order of the fixed assignments on the ITMPMSL file, however, is by target number. Program PLNTPLAN, which reads the MSLTIME file, requires the information in order of increasing group number.

Common block /WADWPN/ is used for temporary sorting storage for this subroutine. The first 6,000 words of this block are redefined as the following variables local to SORTMIS:

FIXTOA	Time of arrival (equivalenced to IFIXTOA)
IGFIX	Weapon group number
NDEXFIX	Index number of target
DFIX	Target designator code
TFIX	Target task/subtask code
INDX	Sequence array for sorting

The length of each of these arrays is 1,000 words. Thus, information for up to 1,000 fixed missiles can be accommodated in this area at any one time. If there are more than 1,000 fixed missiles, their information must be sorted in several passes. Information for the first thousand missiles is read into core from the ITMPMSL file, sorted by group number and written on a scratch file. Information for the next thousand missiles is then read from the ITMPMSL file and sorted by group number. The data in core and on the first scratch file are merged and written on

a second scratch file. The merging process merely writes from core or the first scratch file, depending on which origin has the lower group number. The process is repeated, switching the two scratch files from reading to writing. On odd-numbered passes the first scratch is written and the second is read. On even passes the roles are reversed. The last pass occurs when the number of missiles for which information remains on the ITMPMSL file is less than 1,000. In this case, two switches are set. The logical file name for writing, MYDENSAV, is changed from SCRATCH to MSLTIME. This insures that the MSLTIME file will be written in the last pass. The second switch is the logical variable IPRT. This switch is initially set .FALSE. to suppress printing of the sorted missile information. On the last pass, it is set .TRUE. to print the sorted fixed missile information on the standard output.

The subroutine begins by terminating the ITMPMSL file written during the first pass by STALL and DEFALOC. A test is then made to determine if any fixed missile information was saved on the ITMPMSL file (i.e., NFIXWPS greater than 0). If not (statement 100) a null MSLTIME file is written. This null file contains one fixed missile information record with group number 999. A message is printed which states that no information was saved and control returns to MULCON.

If there is fixed missile information (statement 200), the number of sorting passes needed (NTIMES) is computed and the scratch files are initialized. The first scratch file uses ITP=10 and the second file (ITP=9) is initiated before the first pass by writing a fixed missile record for group 999. Statement 220 provides for skipping over the multiple pass sorting phase when the number of fixed missiles is less than 1,000 (i.e., NTIMES=0). The multiple pass coding begins at statement 300 where the scratch files are interchanged and prepared for input/output. Then the next 1,000 fixed missile records are read from ITMPMSL. The time of arrival is in BCD code on the ITMPMSL file. If it is blank for a record, it is reset to -99999.999 to signal program PLNTPLAN that no arrival time was specified by the user. If it is not blank, it is decoded from BCD code to floating point format. Following statement 330, utility subroutines ORDER and REORDER are used to sort the block of fixed missile information in core by group number. Then the merging process begins. The first record is read from the read scratch file. If its group number is less than the first group number in core, it is written on the write scratch file. The next record is read from the read file and the process is repeated. If the group number for the record in core is lower, it is written on the write scratch file. The next record in core is then compared to the file record. This merging process continues until all records have been processed.

Statement 540 is used to set the switches for the last pass as previously described. If the number of fixed missiles is an exact multiple of 1,000,

the statements following 510 merely copy the last written scratch file onto the MSLTIME file. The subroutine ends by terminating the UTMPMSL file at statement 900.

Figure 51 illustrates subroutine SORTMIS.

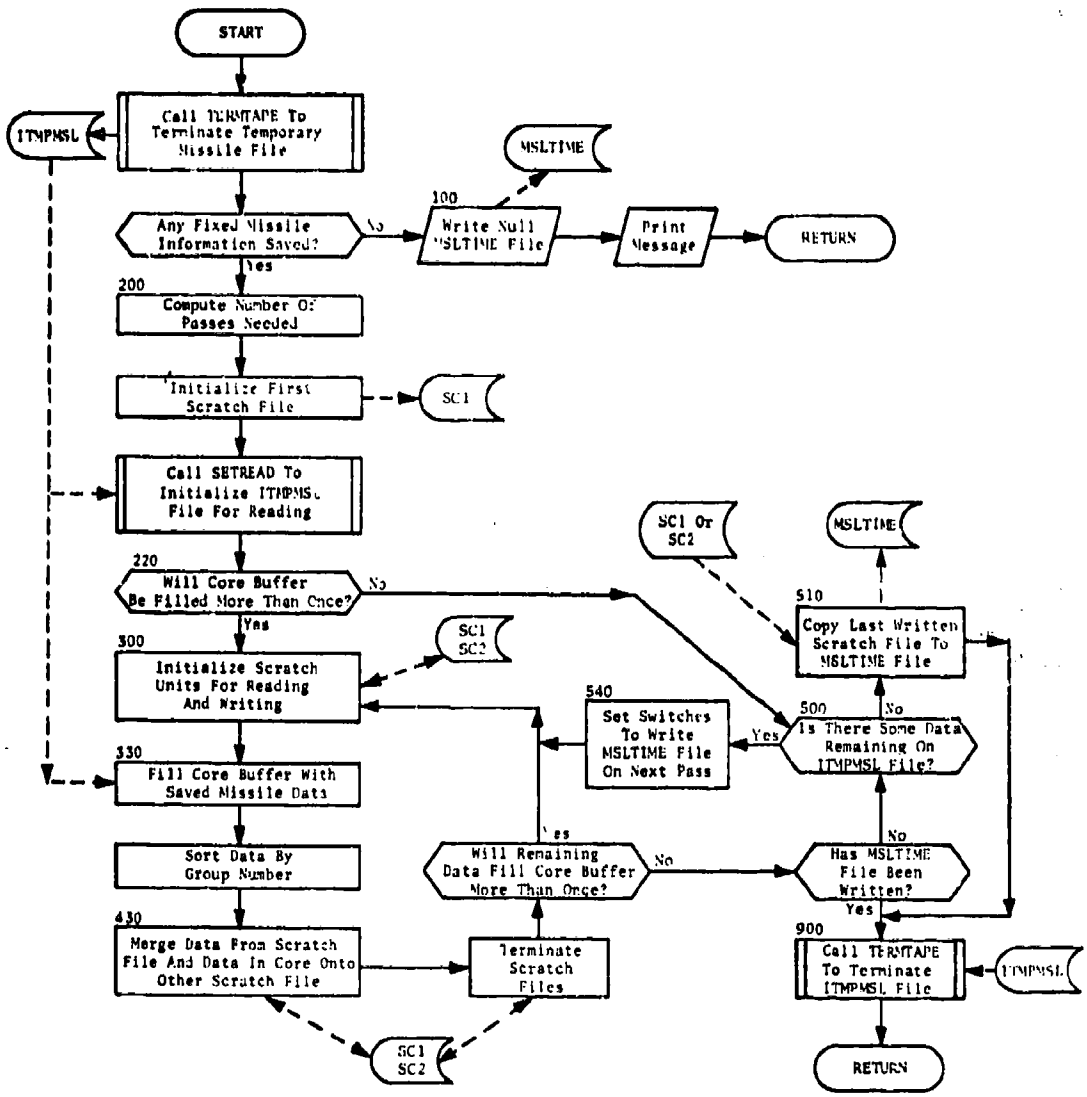


Fig. 51. Subroutine SORTMIS

SUBROUTINE STALL

PURPOSE:	This routine determines the sequence of weapon additions and deletions required to achieve a near optimum allocation of weapons to a target.
ENTRY POINTS:	STALL
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CONTROL, DYNAMIC, MASTER, PRIN1, WADOUT, PRINEED, WADFINAL, WADWPN, FIXED, WPNGRP, WPNTYPE, FILES, FIXEDASS, FILABEL, MYLABEL, ITP, MYIDENT, NOPRINT, TWORD, LOCFIL
SUBROUTINES CALLED:	WAD, WRARRAY
CALLED BY:	MULCON

Method

STALL controls WAD's addition and deletion of weapons by setting the values of the following three variables:

WADOP	in/CONTROL/
G	in/PRINEED/
NW	in/WADOUT/

WADOP has the following options:

WADOP = 1	Initialize allocation
WADOP = 2	Finalize allocation (optional print of results)
WADOP = 3	Add weapon from group (G)
WADOP = 4	Delete (NW)th weapon now on target

To facilitate monitoring the operation of STALL, the variable STALPRIN is also set to provide a unique indicator of the position in STALL where WAD is called. This variable is printed under the print option 22.

The input data for STALL consists of the following six variables supplied by WADOUT in /WADOUT/:

PPMX and IPPMX - the maximum potential increase in effective profit for any weapon, and the index G to that weapon group, respectively.

PVRMX and IPVRMX - the maximum effective efficiency for any weapon, and the index to that weapon group, respectively.

DPMN and IDPMN - the minimum marginal effective profit for any weapon now assigned, and the index to that weapon in the list of weapons assigned, respectively.

The flowchart for STALL is in four parts. The first part contains the setup and single weapon allocation phase. This phase provides a prompt exit from STALL if the indicated allocation consists of one weapon or less. This part also includes a dummy version of STALL which is used to reproduce the prior allocation independent of current payoff data. This option is used in place of the usual verification phase (when PROGRESS =2) if IVERIFY =2. This mode of operation checks the effects of an alternate level of interweapon correlation, CORR2.

Before going into the normal allocation phase, the initial value of the time-of-arrival error allowance DELTVAL is saved, so that it can be restored if it is necessary to change it. This quantity determines the maximum fractional difference in target value at the time-of-arrival of weapons that are allowed to use the same time-of-arrival bin in the calculations by WAD. If the indicated allocation would result in an overflow of the available time-of-arrival bins, this quantity is increased by STALL and the allocation is reinitialized for another attempt.

The second part of STALL processes the fixed assignment data. It puts the weapons down on target and, if they are missiles and the fixed missile information is to be saved (i.e. SAVEFIX = .TRUE.), writes the group number, target identifiers, and specified arrival time on the ITMPMSL file. After initializing WAD, the routine checks the pass number. On the first pass, the fixed assignments were placed in array IG by MULCON. In later passes, they are in the IGO array. The statements after 126 determine the number of weapons the assignment represents. If DEFALOC has made an allocation on the previous pass, the number of missiles allocated from each group is shown as a negative number in the KORR array. If the KORR entry is positive, there is only one weapon assigned from the group. STALL then checks the INACTIVE flag to see if the weapon can reach the target. If not, an error message is printed and the assignment request is ignored. In statement 443, WAD is called to actually put the weapon on target. During the first pass, the group number, specified arrival time, and target information (designator code,

task-subtask code and index number) for fixed missile assignments are output to the ITMPMSL file. This file will be processed by subroutine SORTMIS which creates the MSLTIME file to be used by PLNTPLAN to specify missile launch times.

The third part of the subroutine provides an initial laydown of weapons if multiple weapons are indicated against the target. As this laydown progresses two types of array overflows could occur. The overflow of available time-of-arrival bins has just been discussed. It results in simply restarting the allocation. The other possibility is that the total number of weapons assigned could exceed the maximum number (30) permitted. In this event, control is passed to the refinement loop just as it would be in the normal exit at the bottom of Part III.

Part IV independently checks for such a potential overflow (near statement 59) and if it is threatened, a cycle of operation is generated through the connector (D) that results in removal of the least profitable weapons (statement 52) and replacement by the most profitable available weapons (statement 56). This sequence is terminated either by entering the normal refinement process when the residual target value is reduced so that no potential weapons remain profitable (statement 54), or by using exit (F) after statement 59 if no combination of 30 weapons can be found which will reduce the target value sufficiently.

The operation of the normal refinement loop, which cycles through the branch at statement 71 until the tests at statement 66 are satisfied, is discussed in detail in the Analytical Manual* and will not be repeated here.

Figure 52 illustrates subroutine STALL.

---

* Volume II, Plan Generation Subsystem, Chapter 2, Analytical Concepts and Techniques, (see Weapon Allocation, Single Target Allocation - Targets Without Terminal Ballistic Missile Defenses).

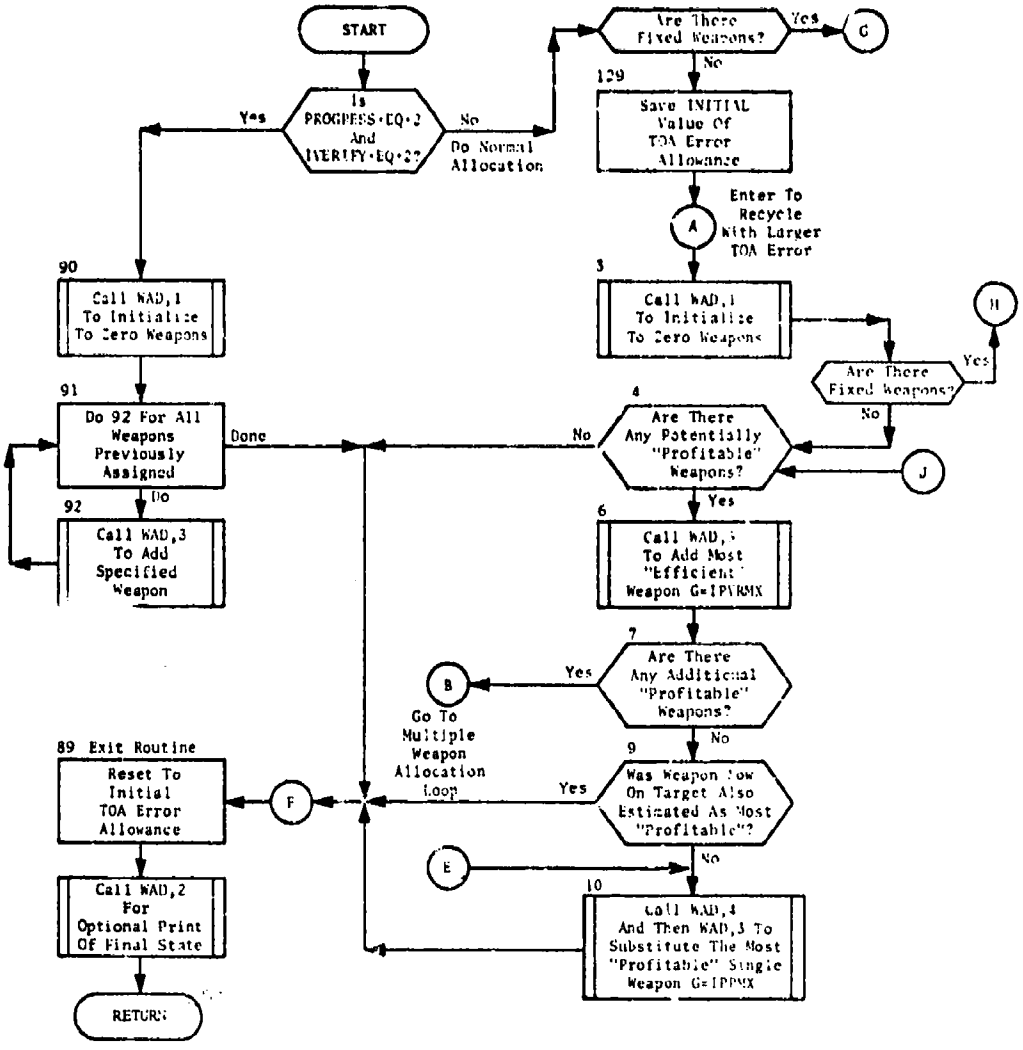


Fig. 52. Subroutine STALL
Part I: Setup and First Weapon

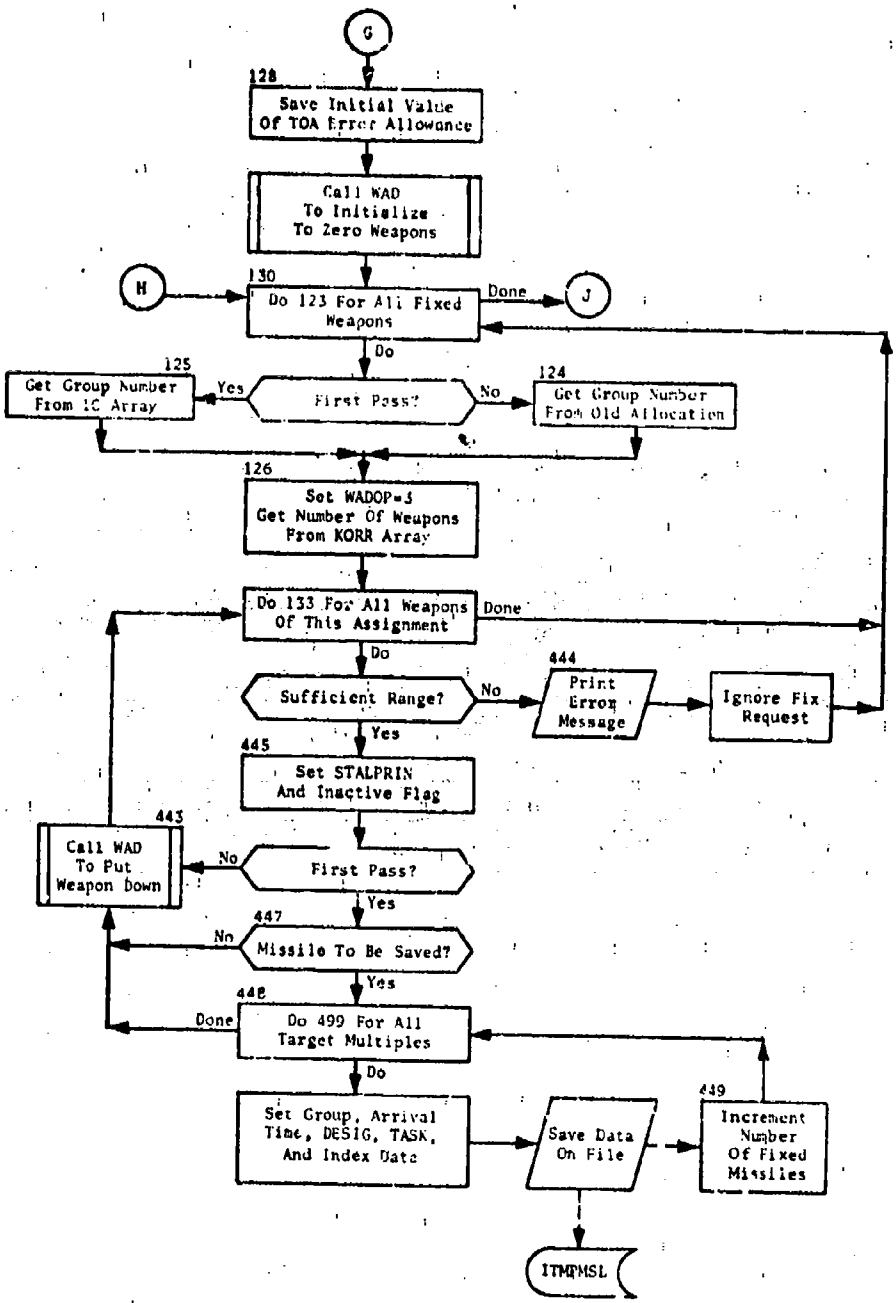


Fig. 52. (cont.)
Part II: Fixed Weapon Assignment Processing

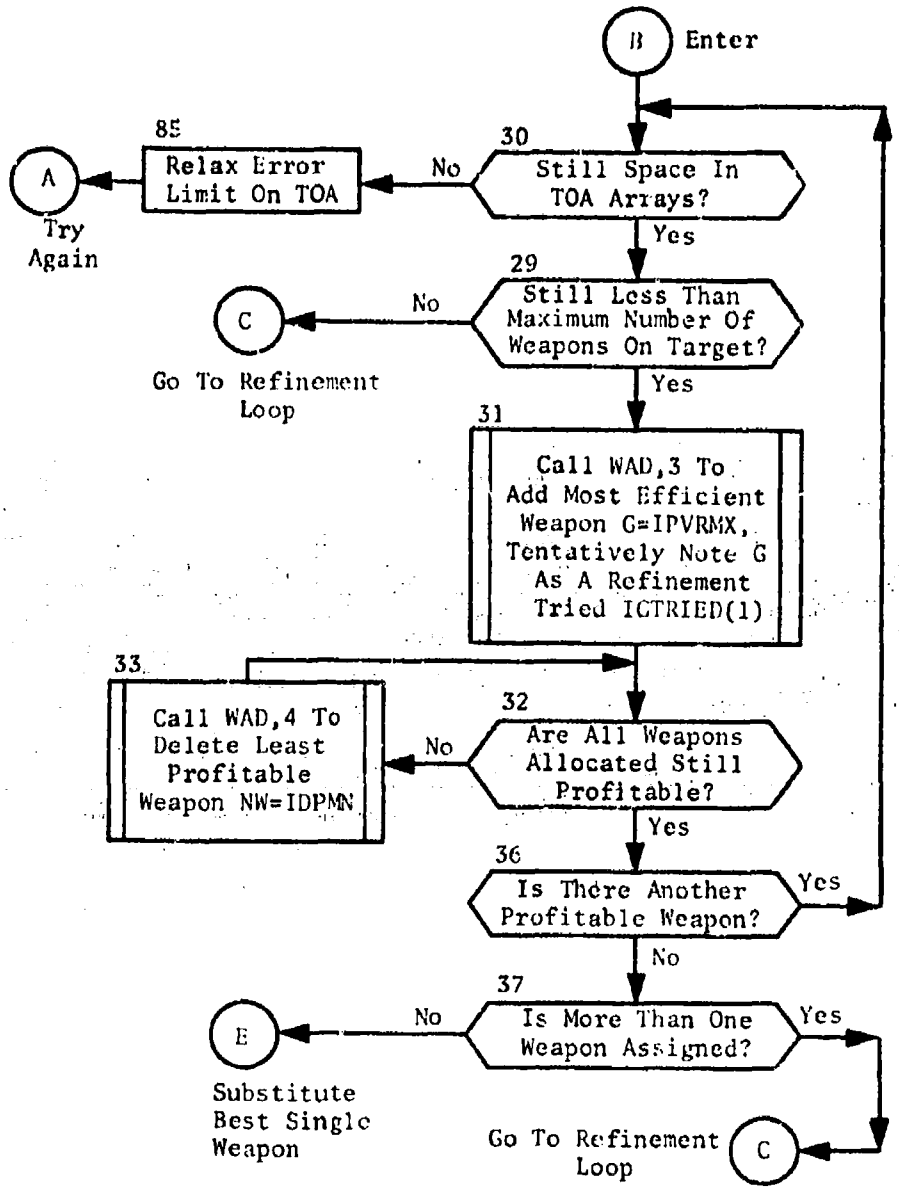


Fig. 52. (cont.)
Part III: Multiple Weapon Laydown

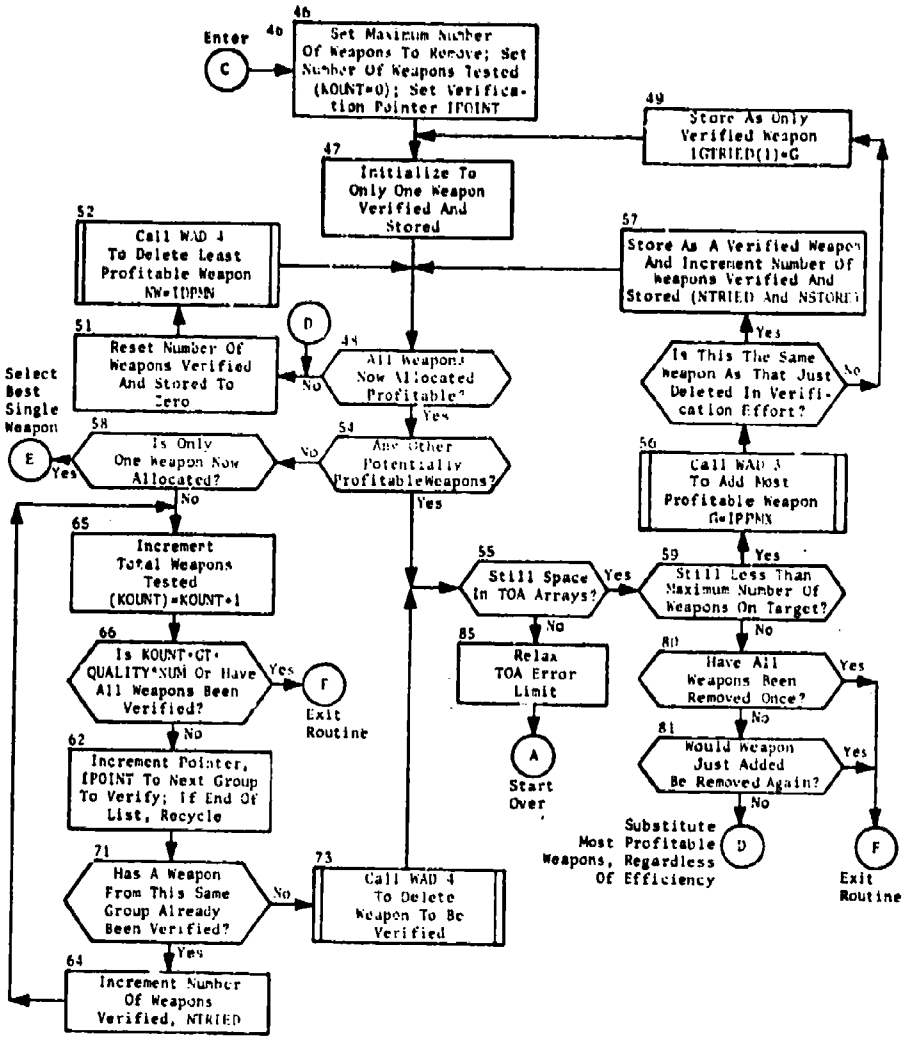


Fig. 52. (cont.)
Part IV: Multiple Weapon Refinement Loop

FUNCTION TABLEMUP

PURPOSE:	This function calculates weapon-target kill factors for either the exponential or square root damage laws as a function of an input single shot survival probability.
ENTRY POINTS:	TABLEMUP
FORMAL PARAMETERS:	S -- a single shot survival probability
COMMON BLOCKS:	WADWPN, TABLE
SUBROUTINES CALLED:	None
CALLED BY:	RESVAL, RECON

Method

This function computes weapon kill factors by computation for the exponential damage law and by a table lookup for the square root law. The function uses the square root law only if the option is selected by the user and the target has a radius greater than 0.

If the square root law is used on a target, the variable ILAW in common /WADWPN/ is set to 100 by subroutine RECON. This variable is checked by TABLEMUP to determine which damage law is used. The exponential law use causes ILAW to be set to 0.

The input formal parameter is a single shot survival probability, S. If the exponential damage law is selected, the function returns the value -LOG(S). If the square root law is selected, the function performs a table lookup technique on the array TABLE in common /TABLE/. This array was preset by subroutine SETABLE. The function performs linear interpolation between the entries of the table. It returns the square of the interpolated value. (The method of determining the kill factors used in subroutine SETABLE is too slow for use in TABLEMUP.)

Function TABLEMUP is illustrated in figure 53.

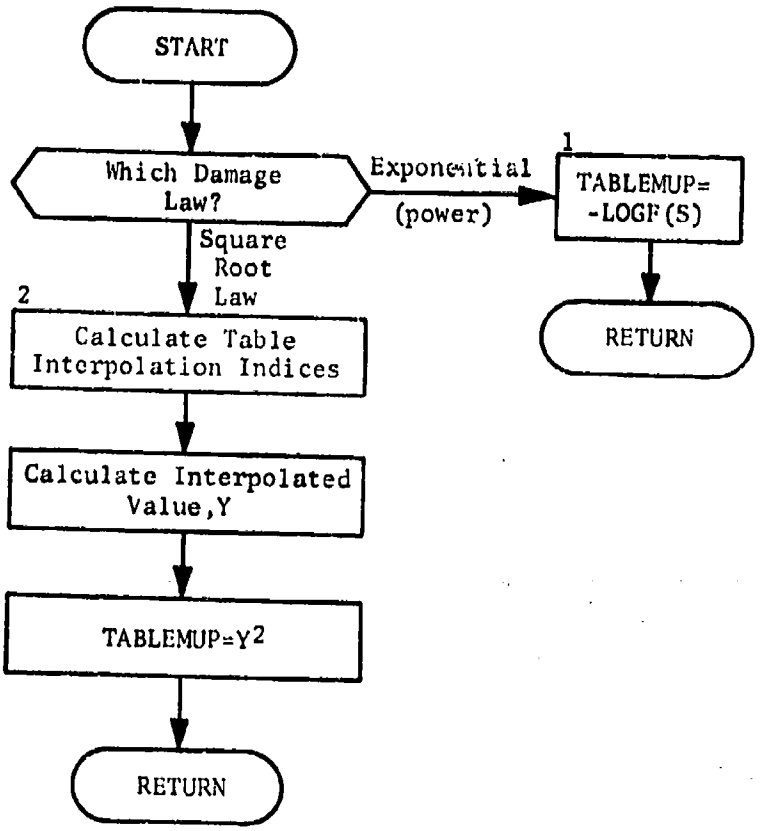


Fig. 53. Function TABLEMUP

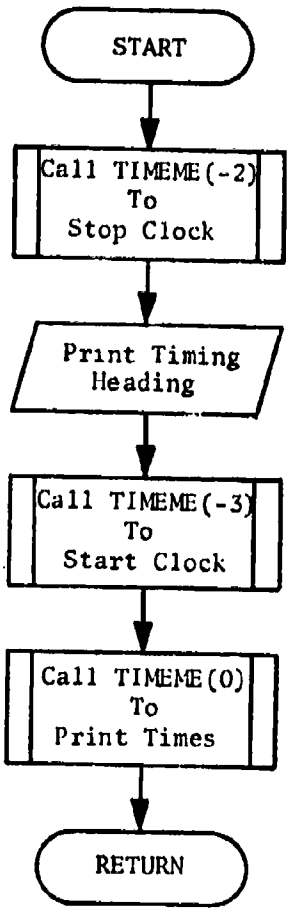
SUBROUTINE TIMEPRT

PURPOSE:	This routine prints the amount of time spent in the options which precede the ALLOCATE function.
ENTRY POINTS:	TIMEPRT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	None
SUBROUTINES CALLED:	TIMEME
CALLED BY:	ALOC

Method

This routine first calls utility subroutine TIMEME with a -2 argument to stop the clock while the heading is being printed. After the heading is complete, two calls on TIMEME are made to restart the clock (argument of -3) and print the time spent in each option (argument of 0).

Subroutine TIMEPRT is illustrated in figure 54.



Subroutine TIMEPRT

Fig. 54. Subroutine TIMEPRT

SUBROUTINE WAD

PURPOSE:	This routine carries out the addition and deletion of weapons as specified by STALL. After each change in the allocation to a target WAD computes the surviving target value VT; for each potential weapon G, the potential surviving value VTP(G) if a weapon for that group were added; and for each weapon currently on the target, the potential surviving value VTD(NW) if a weapon from that group were deleted.
ENTRY POINTS:	WAD
FORMAL PARAMETERS:	None
COMMON BLOCKS:	222, 333, CONTROL, DYNAMIC, LAMBDA, MASTER, PAYOFF, PRINEED, PRNTWADD, WADFINAL, WADOUT, WADWPN
SUBROUTINES CALLED:	FMUP, PREMIUMS, PRNTALL, WADOUT
CALLED BY:	STALL

Method

The surviving target value VT is given by:

$$VT = \sum_{J=1}^{J=M} \sum_{N=0}^{N=NN} [V(N,J) - V(N+1,J)] * S(N,J)$$

where

$$S(N,J) = FMUP\ [(MU(N,J)\ **2)/(MU(N,J) + SIG(N,J))].$$

The function FMUP(X) is defined as follows:

Exponential damage law:

$$S(N,J) = FMUP(X) = \exp(-X).$$

Square root damage law:

$$S(N,J) = FMUP(X) = (1 + \sqrt{X})(\exp(-\sqrt{X})).$$

The index J is over the hardness components, and the index N is over the time of arrival bins. V(N,J) is the unattrited value of the Jth hardness component at the time corresponding to the Nth time-of-arrival bin.

MU(N,J) is the summation of MUP(G,J) for all weapons arriving at the target through the Nth time-of-arrival bin.

SIG(N,J) is the summation of all the cross terms (relative to the Jth hardness component) between all the weapons on the target by the Nth time-of-arrival bin.

The calculations indicated above are carried out by WAD in sets of scratch arrays and computation arrays that make it possible to retain all intermediate values in the calculations. The resulting scratch pad is then referred to and used wherever possible in computing the modified value of VT,VTP(G) if a weapon G is added, and the modified value of VT,VTD(NW) if the (NW)th weapon is deleted.

Table 17 may help to visualize the scratch pad results that are stored. This illustrates the calculation at a time when there are three weapons on the target (NUM = 3), which use two time-of-arrival bins (NTOA = 2), and there are two hardness components (M = 2).

The first time-of-arrival bin always corresponds to 0 time and reflects the maximum target value, VAL(1) = 100, in the example. Bin number 2 corresponds to the first actual time of arrival and it contains two weapons NWP(2) = 2, but the unattrited target value at that time is lower, VAL(2) = 60. Thus, 40 units of value are assumed to escape before the first two weapons arrive. Similarly, an additional 20 units are assumed to escape before the last weapon arrives. The total value, 100 units, is assumed to be distributed 80-20 between the two hardness components (VO(1) = 80, VO(2) = 20). The calculation is carried out in parallel for the two hardness components JH = 1, JH = 2. The total surviving and escaping value for each component, VSN(4,J), is added to obtain the total surviving value:

$$VT = 57.6 = 48.0 + 9.6$$

The intermediate computation values not previously defined, VS and VSN, are defined as follows:

$$VS(N,J) = [V(N,J) - V(N + 1,J)] * S(N,J).$$

Table 17. Illustrating Calculation of Actual Payoff on Target

TIME OF ARRIVAL CELL		1	2	3	4	5
	NWP(TOA)	0.0	2.0	1.0	0.0	0.0
	VAL(TOA)	100.0	60.0	40.0	0.0	0.0
Hardness Element						
JH = 1	V(TOA,J)	80.0	48.0	32.0	0.0	
V0(1) = 80	MU(TOA,J)	0.0	.90	1.70		
	SIG(TOA,J)	0.0	.274	.385		
	S(TOA,J)	1.0	.50	.25		
	VS(TOA,J)	32.0	8.00	8.00		
	VSN(TOA,J)	0.0	32.00	40.00	48.0	
Hardness Element						
JH = 2	V(TOA,J)	20.0	12.0	8.0	0.0	
V0(2) = 20	MU(TOA,J)	0.0	2.0	3.0		
	SIG(TOA,J)	0.0	.48	.915		
	S(TOA,J)	1.0	.20	.10		
	VS(TOA,J)	8.0	0.80	.80		
	VSN(TOA,J)	0.0	8.00	8.80	9.60	

VT = 57.6
NUM = 3
NTOA = 2
M = 2

$$VSN(N+1,J) = \sum_{NI=1}^{NI=N} VS(NI,J)$$

These calculations which constitute the core of the calculations of WAD are carried out in the local subroutine CALPAY (statements 400 through 406) of WAD.

It is useful to visualize how this computation (as illustrated in table 17) would be revised if a new weapon were added. The new weapon might have a time of arrival between bins 2 and 3. In this case a new column 3 would have to be created for the weapon and the contents of columns 3 and 4 would have to be moved over. MU, SIG, and S for column 2 would be unaffected by a weapon arriving at a later time and would remain unchanged. However, the value of VS would be changed since the value of V(N + 1,J) should now reflect the target value for the new column 3 and would be higher than the 32.0 now shown for column 3,JH = 1. The value of MU and SIG in the new column 3 would be the same as that in column 2 except that MU would be augmented by MUP(G,J) for the new weapon and SIG would be augmented by the cross term between the new weapon and all other weapons on the target at that time. The same rule of course applies for all succeeding time-of-arrival bins. In each following column (including old column 3, now 4) the previous value of MU is increased by MUP for the weapon added, and the value of SIG is increased by the cross terms between all weapons previously on the target and the weapon added.

Of course, the new weapon might fit into one of the existing time-of-arrival bins. In this case it would be unnecessary to make a new column, and it would be unnecessary to recompute VS for the previous column. The value of MU and SIG would simply be augmented in the corresponding column and in all succeeding columns as before. Naturally, after the values of MU and SIG are revised, the value of S must be recomputed and the VS and VSN must be revised in the columns affected.

We now recall that WAD is required to provide potential weapon added target value VTP(G) for every weapon group each time a weapon is added or deleted. Obviously the calculation every time of all the above cross terms could be very time consuming. Moreover, precalculation of all individual cross terms for 200 weapon groups would be equally impractical. The technique adopted, therefore, was to calculate cross terms for each weapon group, but only with the weapons already on the target. Since the process of augmenting the values of SIG must be carried out individually for each time-of-arrival bin, the resulting data are stored by time-of-arrival bins. That is, for each weapon group G and each time-of-arrival column, N, data are stored which indicates the amount by which SIG would be increased in that column if the weapon G

were added. These data are stored in the array SIGP(G,J,N). Using these data, the effective augmentation of SIG to calculate VTP(G) for each weapon group G can be accomplished simply by adding SIGP for the appropriate column, and the augmentation of MU is accomplished simply by adding MUP for each weapon G.

Table 18-A may help to visualize how these data are used. Each potential group G is tagged with the index ITOA(G) of the time of arrival column it would occupy if it were added. In addition it is also noted whether the weapon would generate a new column (IADDTOA=1) or share the column with the weapons already there (IADDTOA=0). The situation illustrated here in table 18 corresponds to the same one illustrated in table 17. Notice that for each weapon group G the array SIGP(G,J,NI) contains a significant (usually nonzero) data for NI ITOA(G) - IADDTOA. The extra term in the column NI = ITOA(G)-1 for rows 1 and 5, where the weapon group would require a new column (IADDTOA=1), is to provide a term required for the new columns if this weapon were added. Since addition of this weapon would move all columns (including its own) one position to the right, the resulting term would be in the proper position after moving, even though it is in an incorrect column at present.

Just as the information in table 18-A is used to provide values of SIG for computing VTP(G), the array SIGD in table 18-B is used to provide values of SIG for the computation of VTD(NW). This table contains an entry for each weapon currently on the target. The array IG(NW) in this case indicates that three weapons have been assigned, first from group 2, then group 3, finally another from group 3. The role of SIGD exactly parallels SIGP; that is, to obtain the potential value of SIG if a weapon were deleted SIGD is added to SIG in each column. Since SIGD is negative, this has the effect of cancelling out the cross terms for the weapon that would be removed. Of course, if removal of the weapon would reduce the number of weapons in a time-of-arrival column to 0, the following columns would be spaced back to avoid unnecessary columns.

In summary, table 17 contains the scratch pad data used to calculate the actual payoff. Table 18-A contains the corrections SIGP for SIG needed to calculate the corresponding weapon-added estimate VTP(G) for each weapon group G. Table 18-B contains the corrections SIGD for SIG needed to calculate the weapon-deleted estimate VTD(NW) for each weapon NW now on the target.

These arrays SIGP and SIGD are kept continuously up-to-date as weapons are added and deleted. For example, as illustrated in table 18, the last weapon added was from group 3. Thus the last set of cross terms computed would have been the cross terms between group 3 and every other group. These cross terms are shown in table 18-A in the array DSIG(G,J). When the last weapon from group 3 was added, these terms were computed, and for each weapon G they were added into the array SIGP in all time of

Table 18. Illustrating Quantities Calculated for Potential Weapon Added and Deleted Payoffs

A. - Data on All Potential Weapons

G	VTP (G)	ITOA (G)	IADDTOA (G)	J	DSIG, 3	SIGP(G, J, NI) NI=1	NI=2	NI=3	NI=4
1		2	0	1	.021	0.0	.042	.063	
				2	.103	0.0	.206	.659	
2		3	0	1	.052	0.0	0.0	.45	
				2	.660	0.0	0.0	4.23	
3		2	0	1	.274	0.0	.548	1.653	
				2	.780	0.0	1.560	1.880	
4		3	0	1	.005	0.0	0.0	.010	
				2	.020	0.0	0.0	.040	
5		4	1	1	.003	0.0	0.0	.007	
				2	.009	0.0	0.0	.018	

B. - Data on Weapons Now on Target
(As Candidates for Possible Deletion)

NW	IG(NW)	VTD(NW)	J		SIGD(NW, J, NI) NI=1	NI=2	NI=3	NI=4
1	2		1		0.0	0.0	- .105	
			2		0.0	0.0	-1.32	
2	3		1		0.0	- .274	- .052	
			2		0.0	-1.560	- .660	
3	3		1		0.0	- .274	- .052	
			2		0.0	-1.560	- .660	

arrival columns where both the weapon G and the weapon 3 would be present. In the array SIGD the same quantity:

$$DSIG\left(IG(NW),J\right)$$

is subtracted out for each column where both weapons are present, thus removing the contribution of the weapon, IG(NW), to SIG in the calculation of VTD(NW).

Whenever it is decided to add or delete a weapon from a group, KG, the local subroutine CALSG is called to calculate the array DSIG(G,J) to obtain the cross terms between KG and all potential weapon groups G. CALSG is contained in statements 100 through 108 of WAD.

Table 19 (only partly filled out) illustrates some of the input data required by WAD. The array RISK(IAT,G,J) is used in calculating the cross terms DSIG. However, those elements contribute where the particular attribute (class, type, etc.) is shared.

The shared attributes between weapon groups G1 and G2 are determined by checking whether JATTRIB(IAT,G1) = JATTRIB(IAT,G2).

The flowchart for WAD consists of 14 parts. Part I shows the overall flow of the subroutine. The main processing by the subroutine is controlled by one of the control programs depending on the WADOP option chosen (Initialize, Add, or Delete). The following three parts each illustrate the operation of one of these control programs. Each control routine utilizes a number of other local subroutines, as well as external routines (see figure 55).

The Add Weapon Control routine (Part III) will be used as a vehicle to illustrate the operation of the program. Once the operations of this routine are understood, the corresponding operations in the other routines should be obvious.

The routine first checks to be sure that the number of weapon additions and weapon deletion operations, IOP, on this target does not exceed 100. If it does, it is assumed that STALL and WAD are caught in an endless loop probably repeatedly adding and deleting the same weapon, so the processing of the target is terminated. In principle, such looping should not occur. However, it has been found that errors in reading file data for one target, or a random machine malfunction, or inconsistencies in the data supplied to the program can sometimes result in such a situation. This makes it possible for the program to proceed to the next target rather than aborting the entire run. However, when this happens, the LOOP flag is set nonzero. This causes the print in

Table 19. Illustrating Quantities Pre-Calculated for Each Potential Weapon Before WAD is Called

G	INAC-TIVE (G)	TOA (G)	TVAL-TOA (G)	PEX (G)	MORR (G)	J	VTOA (G,J)	MUP (G,J)	SSIG (G,J)	RISK(IAT, G, J) IAT=1 ALL	IAT=2 GROUP	IAT=3 REGION	IAT=4 CLASS	IAT=5 TYPE	IAT=6 ALERT
1	0	.38	50			1	40.0								
						2	10.0								
2	0	.54	40.0			1	32.0								
						2	8.0								
3	0	.23	60.0			1	48.0								
						2	12.0								
4	0	.55	39.7			1	31.8								
						2	7.9								
5	2	1.5	10.0			1	8.0								
						2	2.0								

statement 41 to appear during the initialization of every succeeding target, so that the user is sure to notice that the difficulty occurred.

However, assuming that no such loop has occurred, the routine adds the Lagrange multiplier for the weapon added to the COST of the allocation and also updates SUMPREM, the sum of the premiums for the target. (This last variable SUMPREM, as well as the variables TBENEFIT and TPMX near statement 14, are computed only to provide a consistency check on the treatment of premiums. These variables are not essential to the operation of the program.) Notice that these variables and almost all other variables used by WAD such as VT, VTP, VTD, PAYOFF, PROFIT, etc. are computed (even for multiple targets) as if the program were dealing only with a single simple target. It is necessary to take target multiplicity into account only when dealing with variables which accumulate the total cost, total payoff, or total consumption of specific weapon groups over the whole target system.

The PREMIUMs used by the allocator, however, depend on just such a variable -- namely SURPWP(G), the available surplus (positive or negative) of unused weapons in each group G. Consequently, when a weapon G is added, the PREMIUM for that weapon group must be recalculated. Before it is recalculated, SURPWP(G) must be revised as it is in statement 7 to reflect the multiplicity of the target; i.e.:

```
SURPWP(G) = SURPWP(G) - CTMULT
```

The variable CTMULT, current target multiplicity, is used rather than the initial multiplicity TGTMULT, because when PROGRESS is equal to 1.0 a multiple target can be split into several parts of reduced multiplicity. The local subroutine SPLIT (Part VI) which begins with statement 9 is responsible for determining if such a split is needed, and for carrying out the adjustment of bookkeeping on the arrays SURPWP and PREMIUM if it is. To avoid unnecessary computation by WAD, SPLIT is designed to minimize the number of times multiple targets are split up. The intent is to avoid separating multiple targets, unless retaining the full multiplicity of the target during the allocation of the weapon indicated would cause a weapon surplus > .5 weapons. Such a change in SURPWP should cause the step premium for the groups to change from positive to negative. Obviously it would be a mistake to keep allocating as if the same premium would apply. Therefore, when this happens the target is split into two parts. One part, CTMULT, containing the largest multiplicity that could be allocated a weapon from one group without causing the premium to go negative, the other part, CTSPILL, containing the remainder.

After this is done, it is necessary to correct the value of SURPWP and recompute the premiums. To understand what is required to do this, we must recall that when MULCON began the allocation to the

target on this pass, it removed the weapons previously assigned to all the elements of the multiple target and thus increased SURPWP by the multiplicity for each weapon previously assigned. If a decision is now to be made that only part of the target is to be dealt with, the old allocation should in effect be restored for the remainder of the target elements. Thus, in statement 911, SURPWP is decreased by the change in multiplicity for each weapon previously assigned. Conversely, while the present allocation was proceeding, SURPWP was being decremented by the multiplicity for each weapon assigned. If we now intend to interpret the allocation as applying only to a part of the total multiplicity, then the value of SURPWP must be increased as in statement 908 by the change in multiplicity for each weapon already assigned. Finally, since the value of PREMIUM(G) depends not only on SURPWP(G) but also on CTMULT for the target all premiums are recomputed (statement 930). This completes the bookkeeping corrections made by SPLIT. A slightly different version of SPLIT beginning at statement 21 is used by the deletion control routine. In this case the question is whether deletion of the weapon group for the full multiplicity would cause a deficit > .5. Aside from this obvious change in sign, however, the operation is essentially identical.

When SPLIT has completed its work, the program proceeds as usual to update SURPWP for the weapon now being added and PREMIUMS(G) is called as usual.

After the call on subroutine PREMIUMS(G), the main business of WAD begins. This business consists of updating all the arrays in tables 17 and 18 to reflect addition of the weapon G. The process is accomplished by calling a series of local subroutines.

First CALSG (Part VII) is called to compute the cross term array, DSIG, in table 18-A. Then ADDSIG (Part VIII) is called to update the arrays SIGP and SIGD. For maximum efficiency, the revision of the arrays SIGP and SIGD is done one column at a time, working from right to left and dealing only with those columns affected by the weapon G. If the weapon G adds a new column, the column NI = NWRT where the augmented results are written, will be displaced one column to the right of NI = NRD where the original data were read. However, before actually updating these two arrays in each column, the data currently in SIGP in that column are used to update SIG in table 17. Since this value of SIGP was required to add the weapon G, the negative of it would be required to delete it. Consequently, at the same time SIGP is simply negated and stored in SIGD to produce a new row (NW = 4) for the weapon G in table 18-B. All of these operations required to update SIG, SIGP, and SIGD are done in series for the same read and write columns, one column at a time until the updating is complete.

Next, ADDIND (Part IX) is called to update all the indices to reflect the addition of the new weapon. The indices which must be updated are in table 17, NUM, NTOA, NUWP, VAL, in table 18, ITOA, IADDTOA.

When this has been done the payoff computations begin. CALPAY (Part XII) is called first to calculate the actual payoff (table 17). Since this involves only one calculation it is done in a very straightforward way and is completely recalculated for all columns. Thus, the subroutine CALPAY is very straightforward and easy to understand.

The updated information in table 17 is then used as a basis for calculating the weapon added payoffs (CALPOT, Part XIII) and the weapon deleted payoffs (CALDEL, Part XIV). To avoid any need for additional arrays to store intermediate results these calculations are done (both by CALPOT and CALDEL) one weapon group and one hardness component at a time working straight down the lists in table 18. For each group and each hardness component the calculations then work from left to right dealing only with columns affected by the new weapon. For the columns in table 17 that would be affected, revised values S1 of S and VSN1 of VSN are computed working toward the right to obtain the final values which can be added to obtain the revised value VTD for the weapon deleted calculations.

When all payoffs have been calculated, WADOUT is called summarizing the results for STALL, and finally the actual PROFIT, and PAYOFF, BENEFIT, etc., are computed and stored so that they are available to be printed if such information is requested. WAD then returns control to STALL.

The control routine for weapon deletion exactly parallels the above procedures except that SUBSIG and SUBIND replace ADDSIG and ADDIND. SUBSIG parallels ADDSIG almost exactly except that the order of processing columns is reversed to avoid writing over essential data as columns are spaced back. SUBIND differs from ADDIND mainly in that indices are decremented instead of incremented.

The initialize control routine (Part II), and the local subroutine INITIALIZE (Part V), are concerned with establishing the starting state for the allocation to a target. After the discussion of the add and delete routine, the flow diagrams should be self-explanatory. However, a couple of comments may be appropriate. For computing efficiency the initialization is limited to the minimum required to provide the starting state. Many cells in SIGP and SIGD are not initialized and irrelevant data will remain in many cells. Thus on targets after the first target during the allocation, many cells shown in table 18 with irrelevant zeros, may in fact contain irrelevant data that will not be referenced. Particular attention is

called to rows containing inactive weapons. These rows will not be reinitialized at all and thus will contain much irrelevant data for prior targets.

On the initialization call, the only nontrivial payoffs that need to be computed are the potential weapon payoffs VTP(G). Since at this stage these all involve only one weapon, the correlation cross terms are irrelevant. Consequently the simple formula,

```
S = FMUP(+MUP(G))
```

can be used.

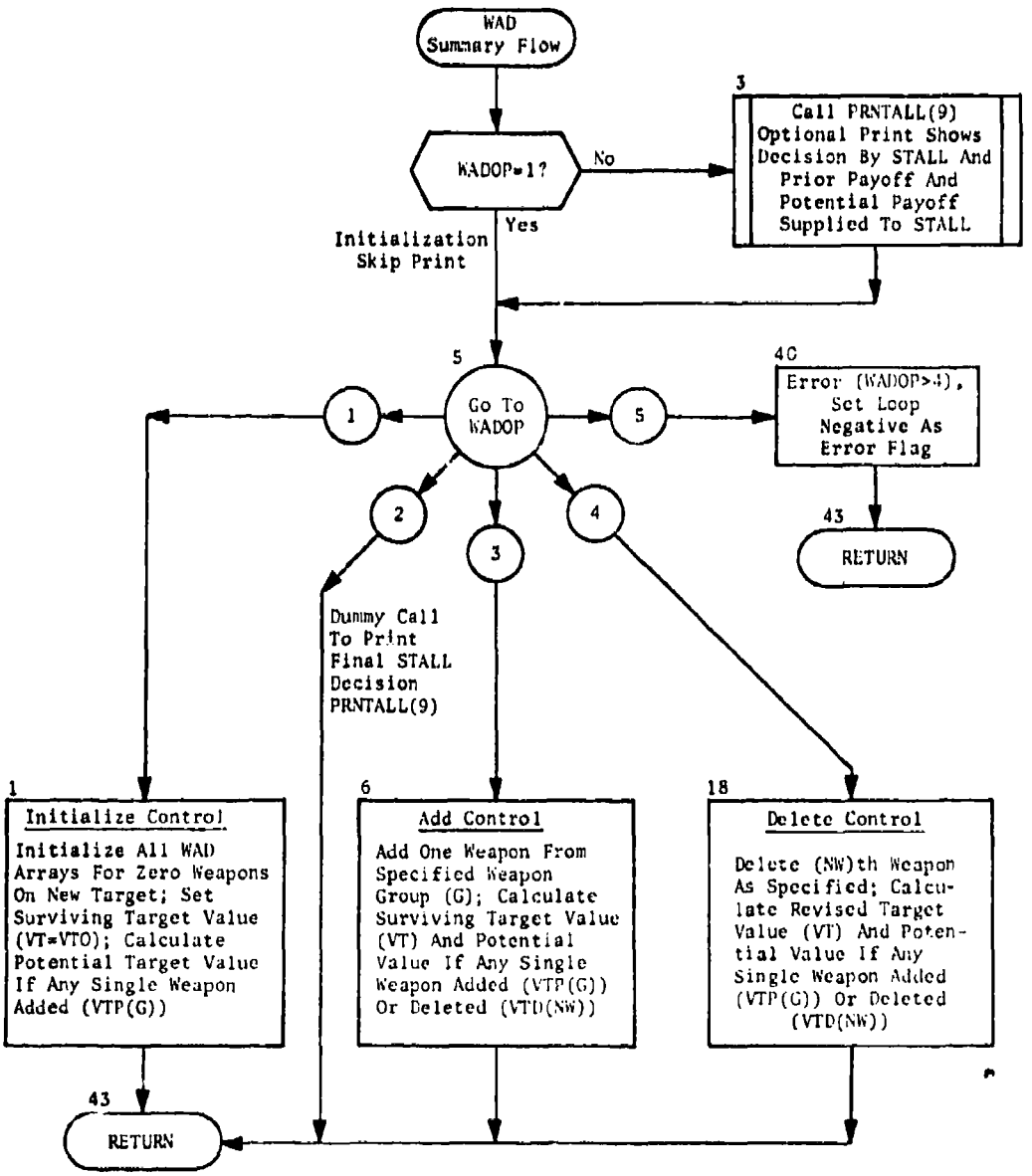


Fig. 55. Subroutine WAD
Part I: Summary Flow

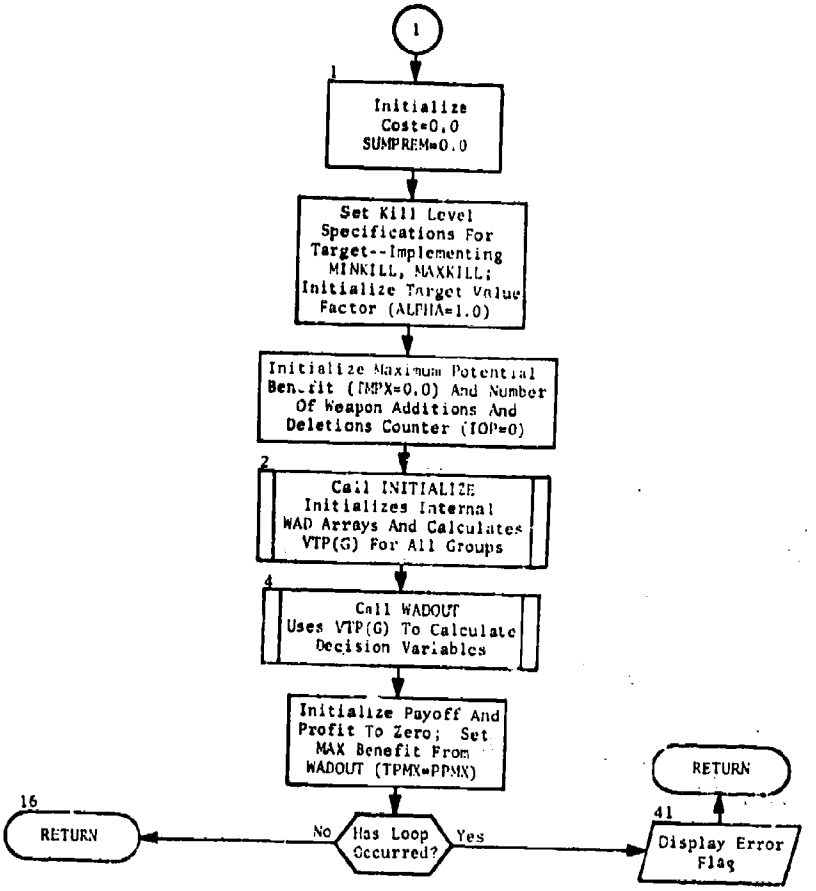


Fig. 55. (cont.)
Part II: Initialize Control

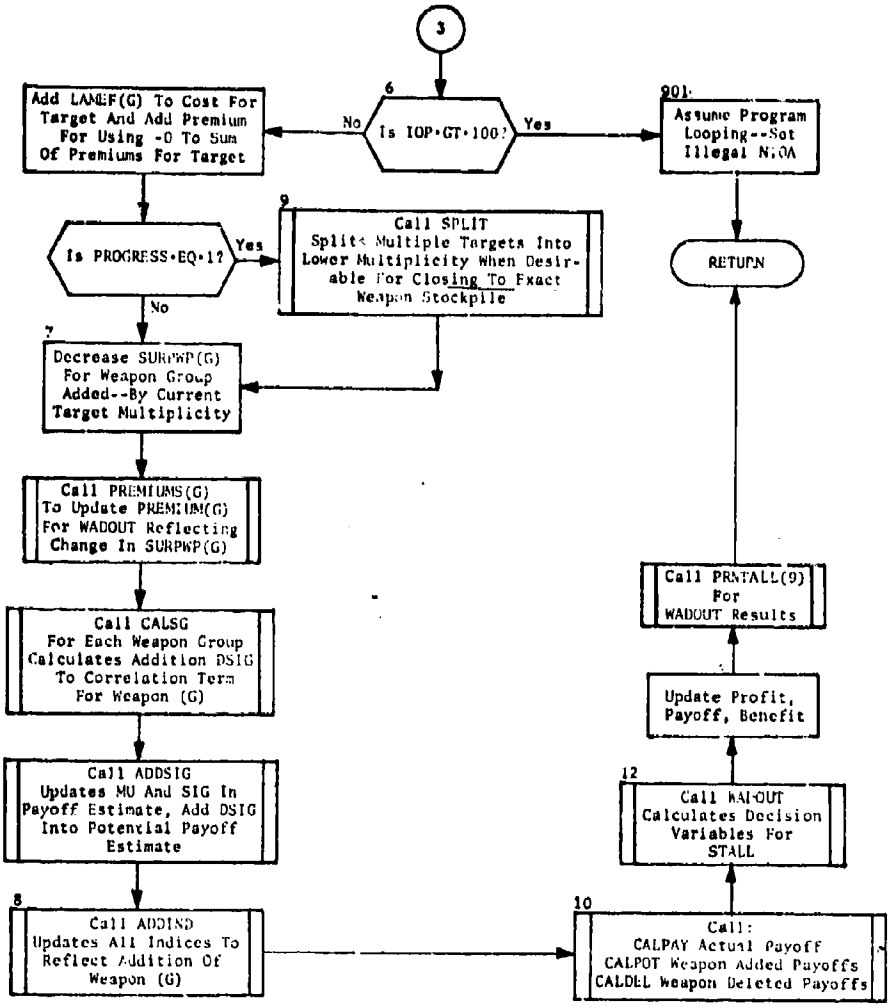


Fig. 55. (cont.)
Part III: Add Weapon Control

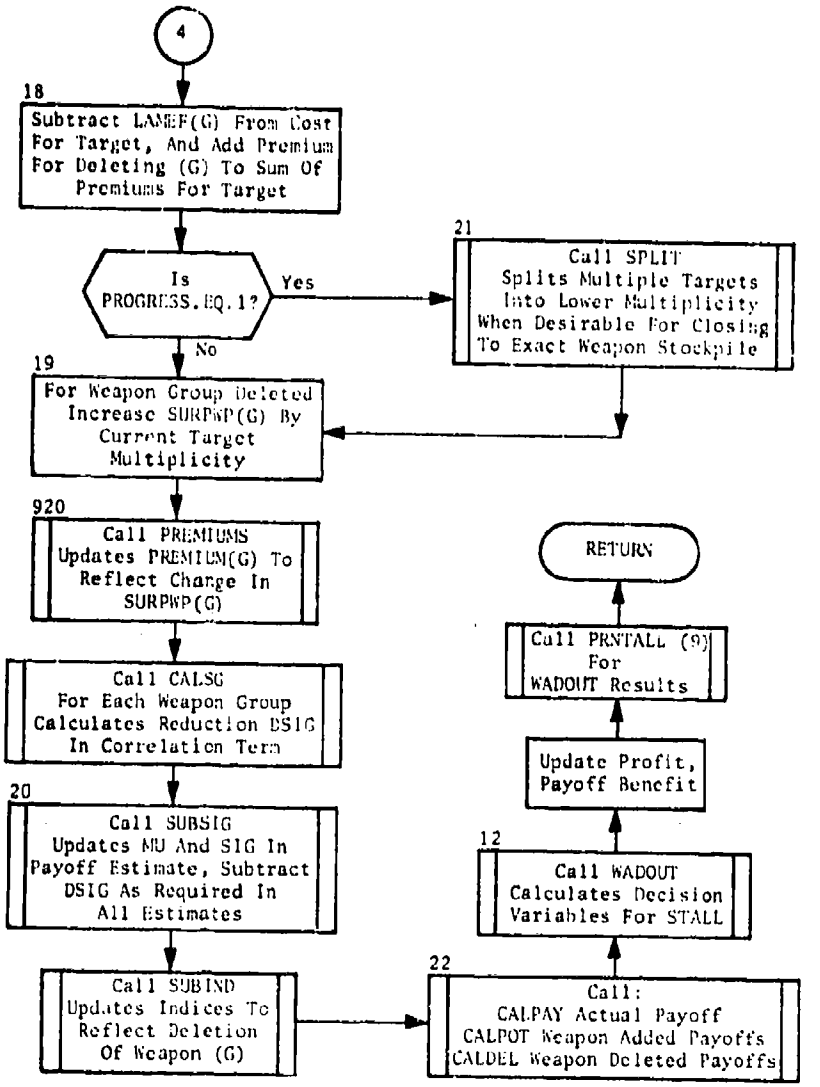


Fig. 55. (cont.)
Part IV: Delete Weapon Control

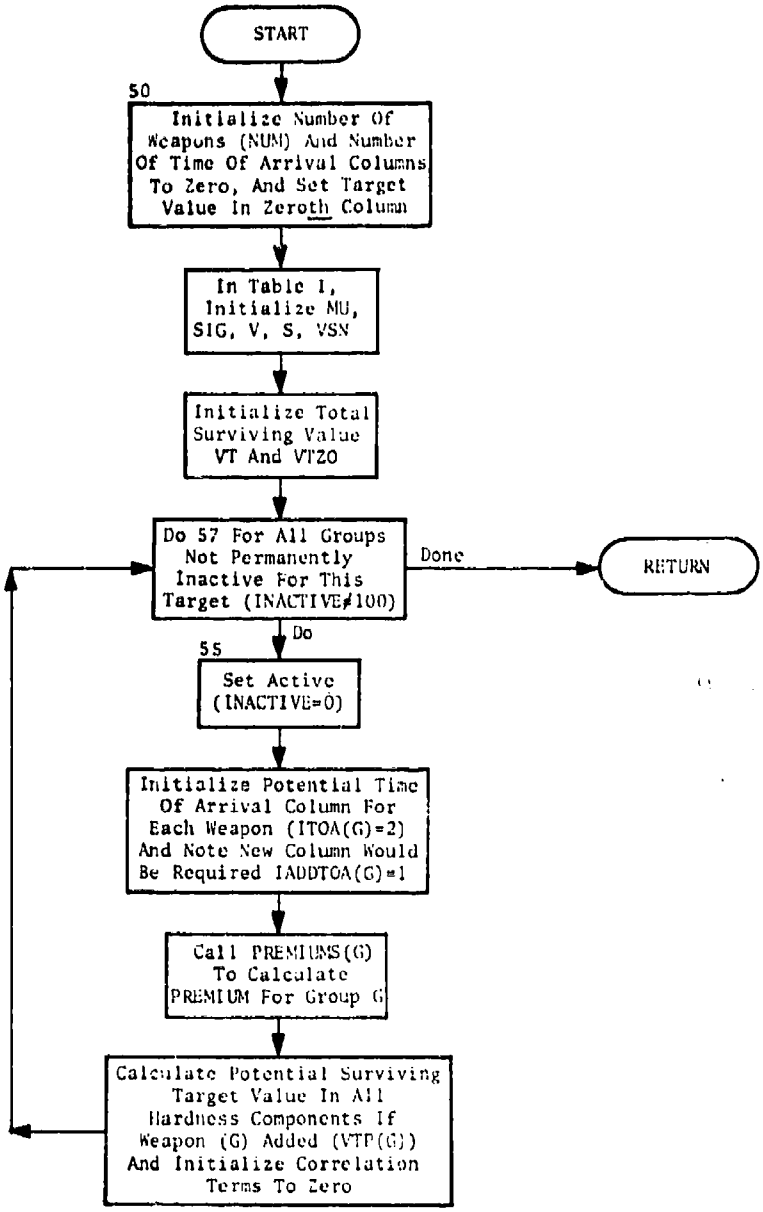
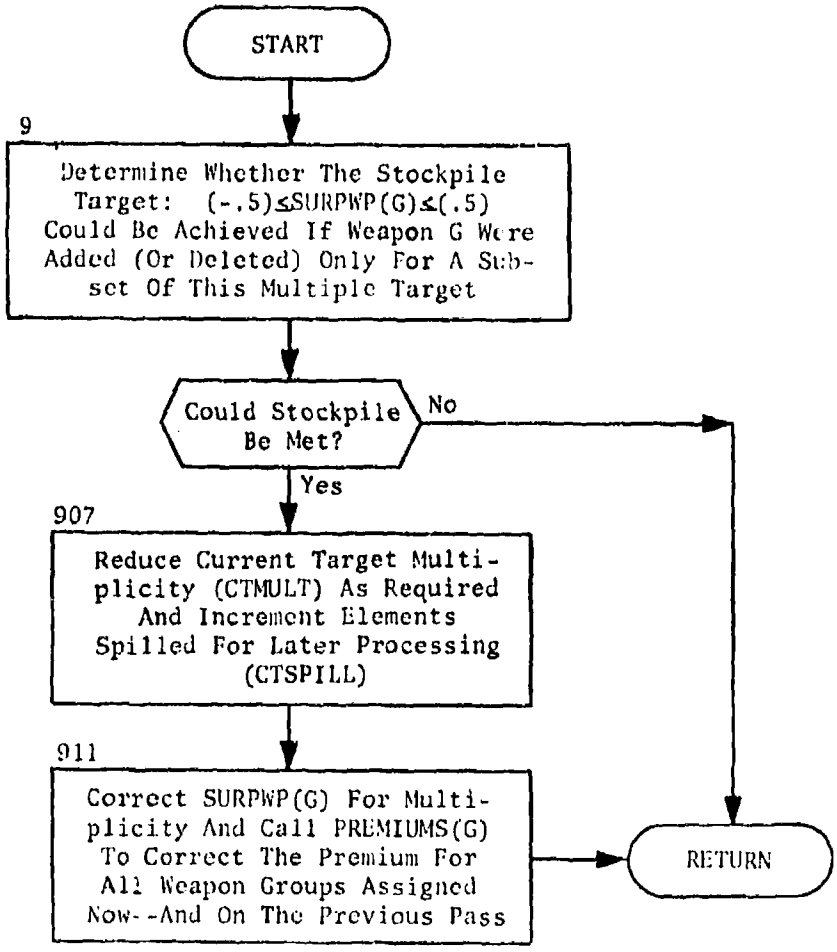


Fig. 55. (cont.)
Part V: Local Subroutine INITIALIZE

[Called in Closing Phase of Allocation, Before Adding (or Deleting*) a Weapon--Splits Multiple Targets into Lower Multiplicity When It Will Help to Meet Exact Stockpile]



* Separate Copy in Program with Change of Signs for Weapon Deletion.

Fig. 55. (cont.)
Part VI: Local Subroutine SPLIT

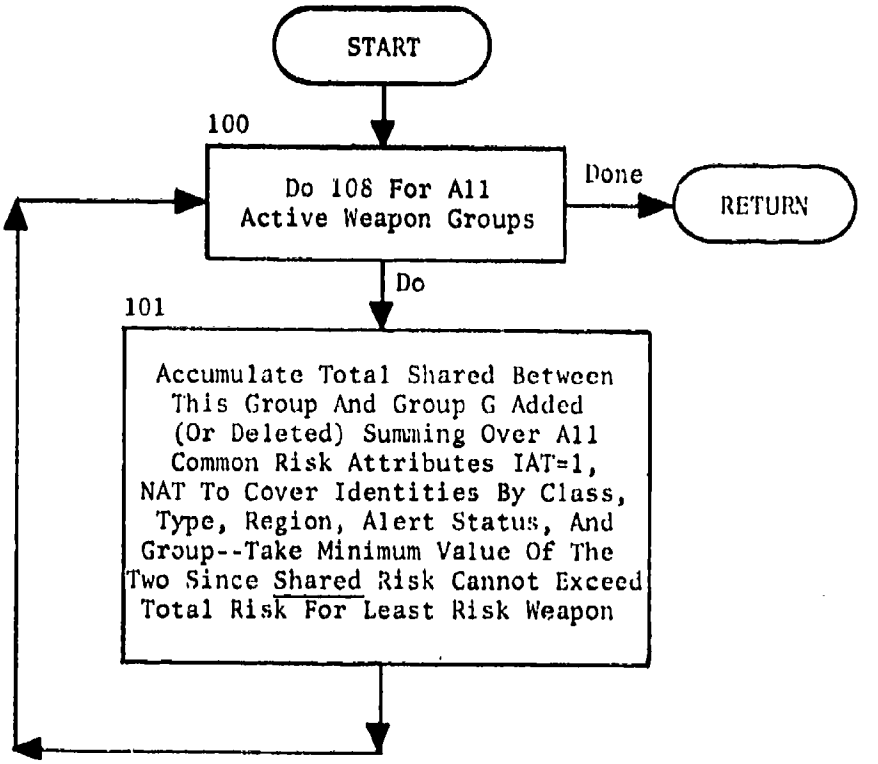
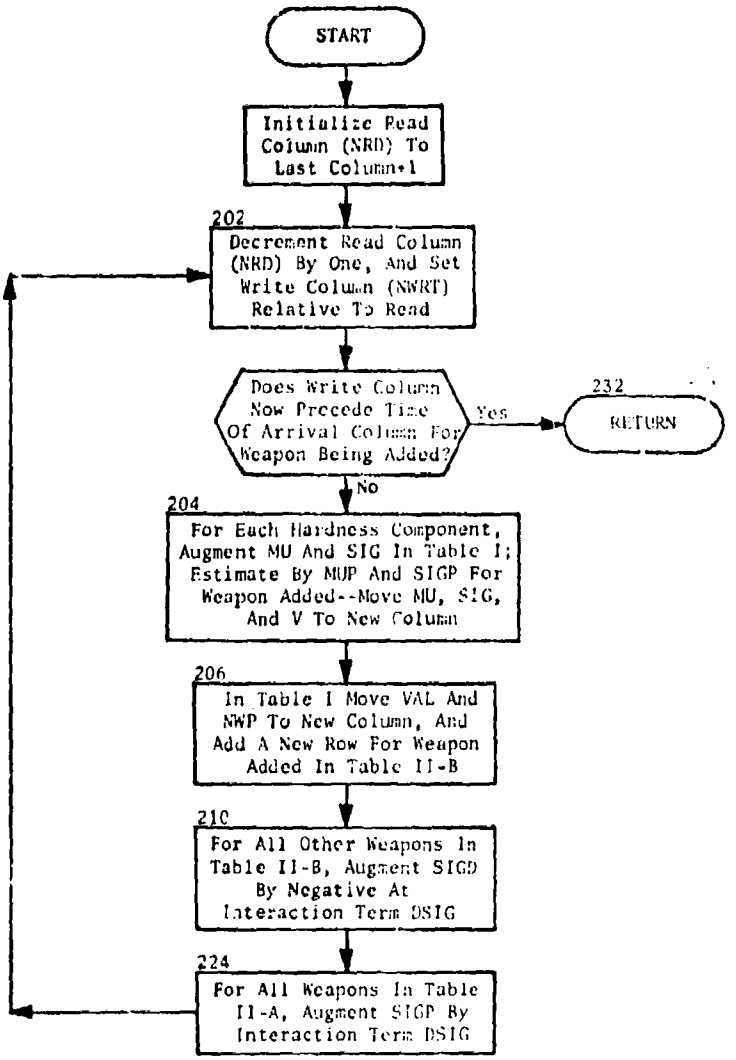


Fig. 55. (cont.)
Part VII: Local Subroutine CALSG



This subroutine updates values for MU and SIG in all Time of Arrival columns affected by new weapon G. If weapon G adds a new column, it also moves the data one column to the right by writing in a column with an index one larger than the index of the column being read.

Fig. 55. (cont.)
Part VIII: Local Subroutine ADDSIG

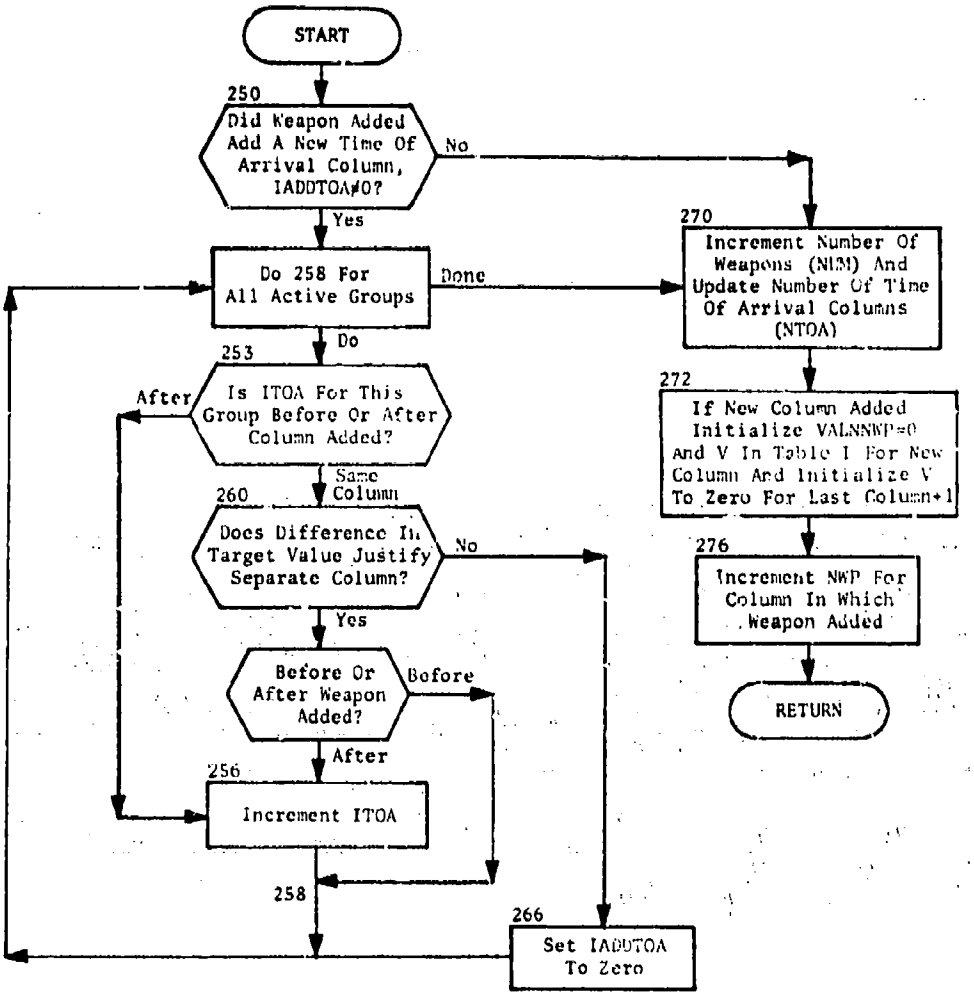


Fig. 55. (cont.)
Part IX: Local Subroutine ADDIND

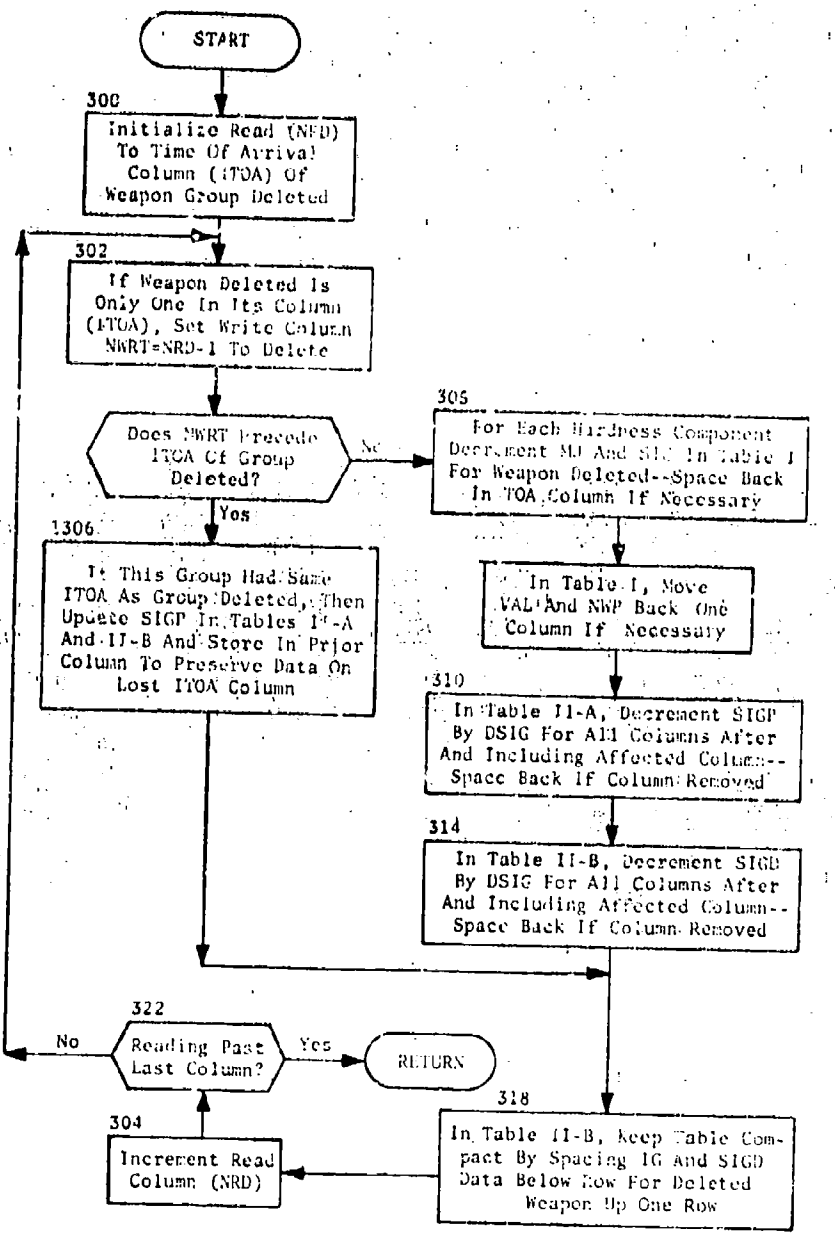


Fig. 55. (cont.)
Part X: Local Subroutine SUBSIG

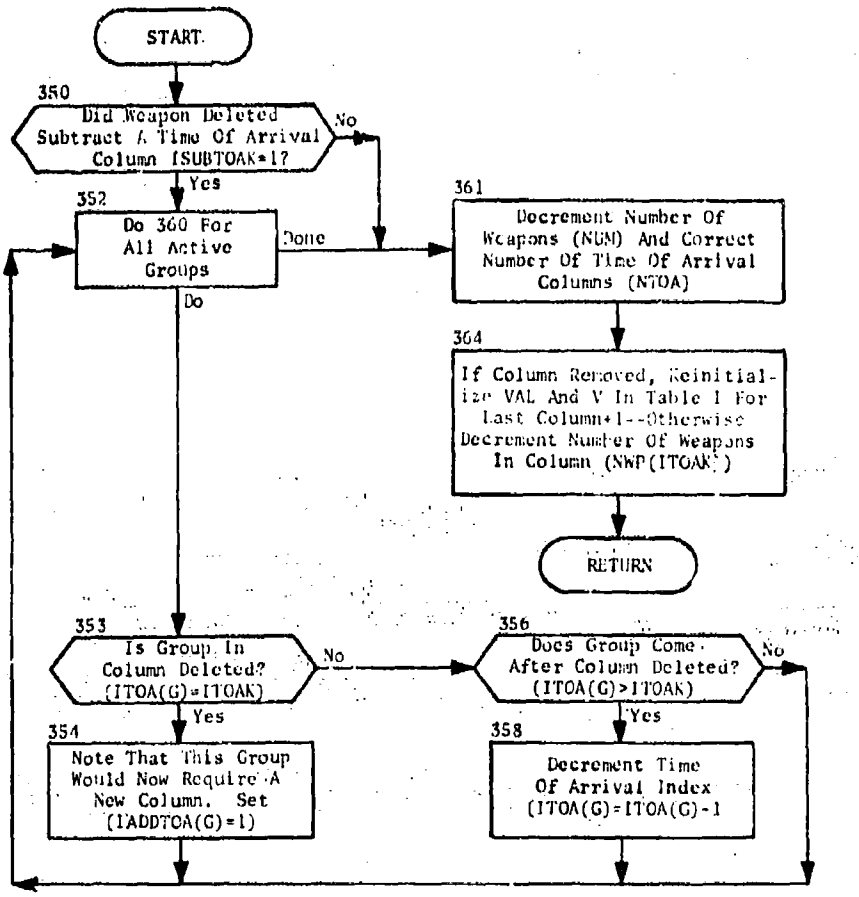
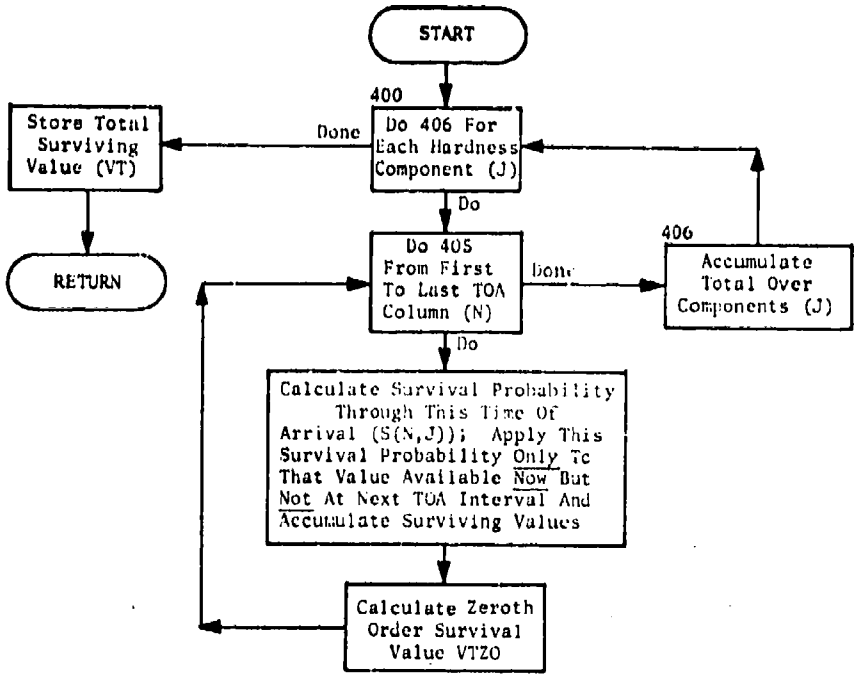


Fig. 55. (cont.)
Part XI: Local Subroutine SUBIND



Calculates Actual Surviving Target Value, VT:

$$VT = \sum_{J=(I,M)} \sum_{N=(1,NTOA)} S(N,J) * [\, V(N,J) - V(N+1,J) \,]$$

and Zeroth Order Surviving Value, VTZO;

$$VTZO = \sum_{J} S(NTOA,J) * VO(J)$$

Fig. 55. (cont.)
Part XII: Local Subroutine CALPAY

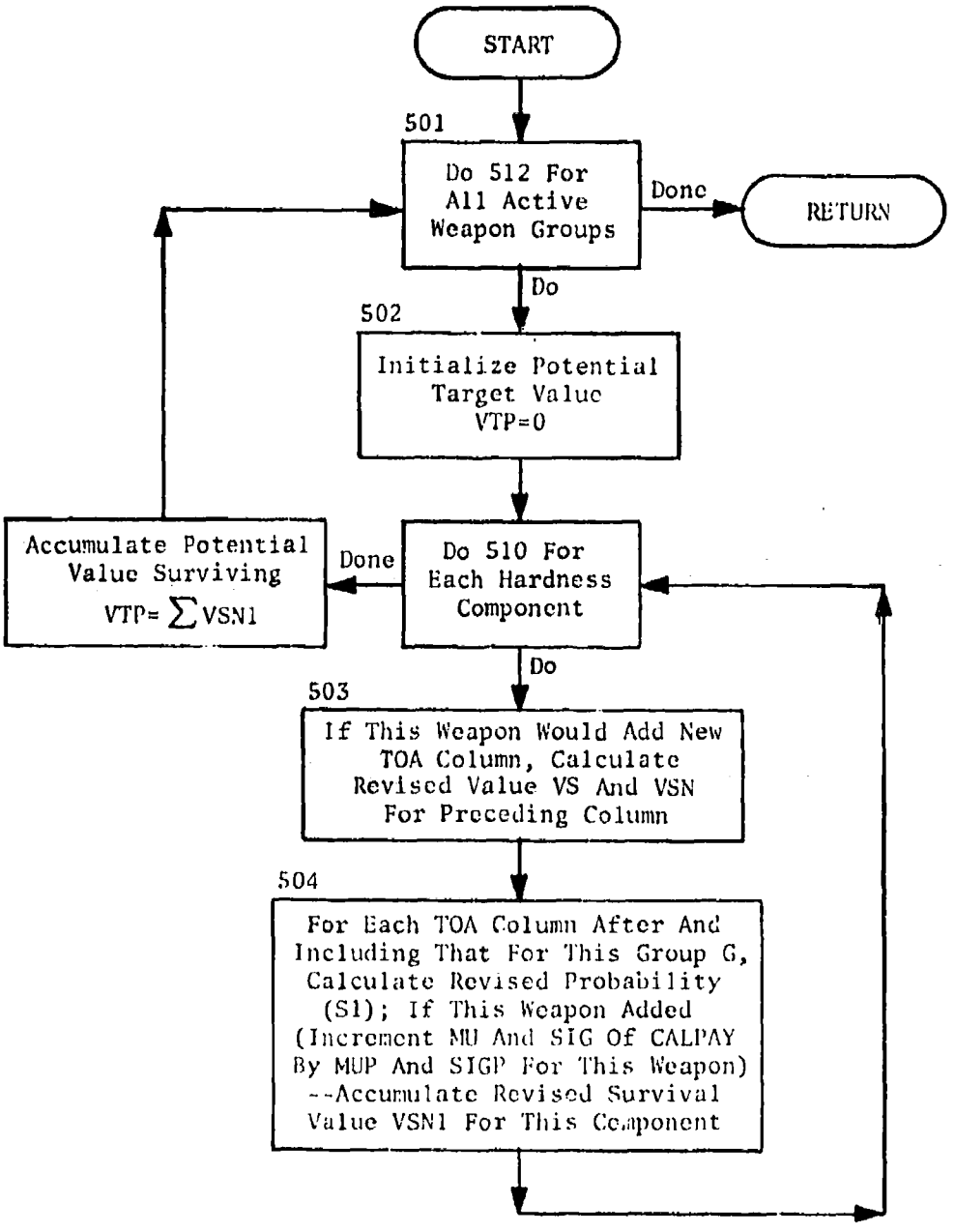


Fig. 55. (cont.)
Part XIII: Local Subroutine CALPOT

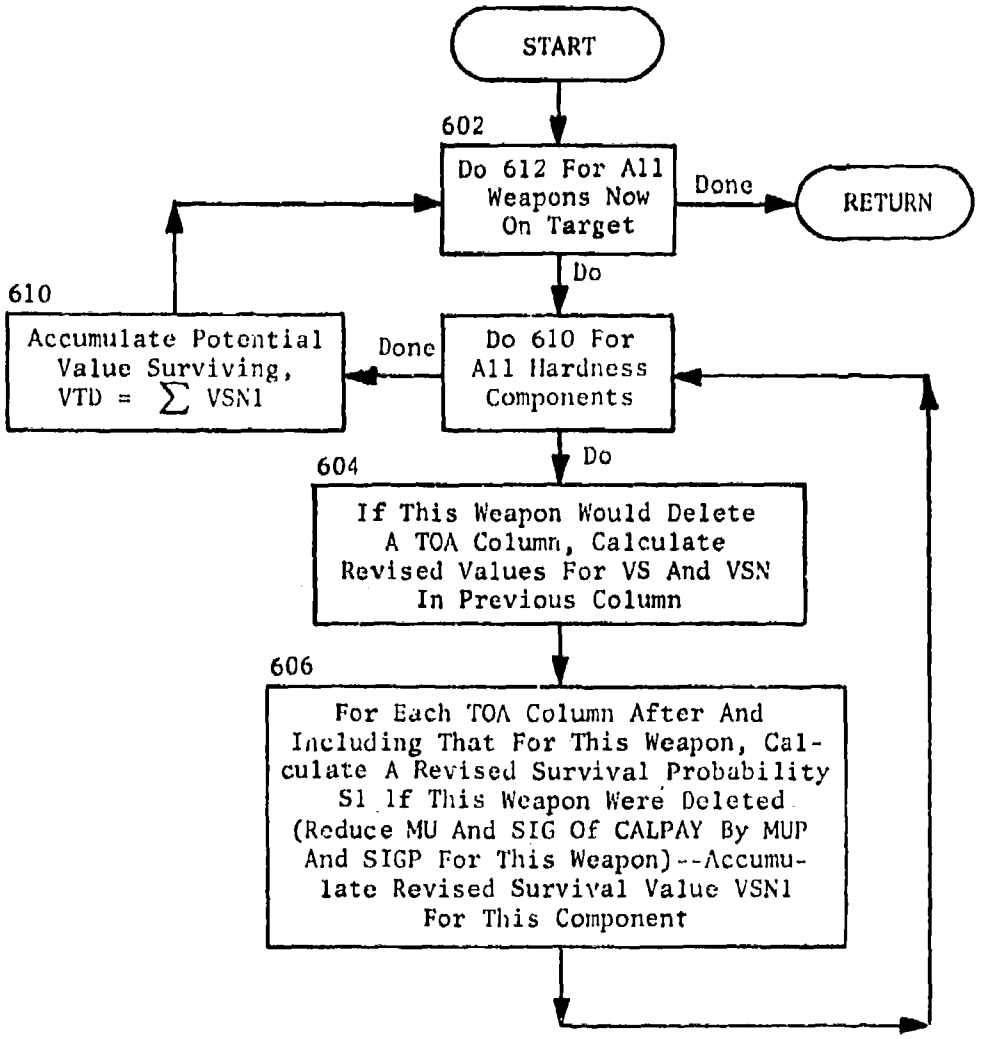


Fig. 55. (cont.)
Part XIV: Local Subroutine CALDEL

SUBROUTINE WADOUT

PURPOSE:	This routine summarizes decision alternatives for STALL by combining payoff data produced by WAD with the weapon cost data (Lagrange multipliers minus premiums). It also contributes to the efficiency of WAD by making inappropriate weapons inactive.
ENTRY POINTS:	WADOUT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	MASTER, CONTROL, DYNAMIC, LAMBDA, WADFINAL, WADOUT, WADWPN
SUBROUTINES CALLED:	PRNTALL
CALLED BY:	WAD

Method

The output data produced by WAD consist of the following parameters for STALL that are recorded in /WADOUT/:

PPMX and IPPMX - the maximum potential increase in effective profit for any single weapon; and the index G to that weapon group, respectively

PVRMX and IPVRMX - the maximum effective efficiency of any potential weapon; and the index G to that weapon group, respectively

DPMN and IDPMN - the minimum marginal effective profit for any weapon now assigned; and the index NW to that weapon in the list of weapons assigned, respectively.

It also produces the array INACTIVE(G) in /WADWPN/ which is used by WAD to determine which weapons groups need not be processed.

The input data from WAD consist of:

VT - the surviving target value in /DYNAMIC/

VTD(N) - the potential weapon deleted surviving target value (also in /DYNAMIC/ (equivalenced to RVAL)

VTP(G) - the potential weapon-added surviving target value in /WADFINAL/.

The input data on weapon costs consist of LAMEF(G), PREMIUM(G), and DPREMIUM(G) in /LAMBDA/.

WADOUT also initializes VTMAX and VTMIN of /WADOUT/, and MAXCOST of /WADWPN/ which reflect the MINKILL, MAXKILL specifications for each target.

The quantities ALPHA and VTEF of /WADOUT/ are essentially local variables for WADOUT. They are included in this common block for use by PRNTNOW, and in the case of ALPHA, to allow WAD to reinitialize it to 1.0 for each new target.

The flow diagram, figure 56, is in three parts. In Part I, the user-input parameter IMATCH is used to determine the method of computing MINKILL and MAXKILL. If IMATCH is 0, then the damage calculations used to determine residual target value for purposes of MINKILL and MAXKILL use time dependence of target value. If IMATCH is nonzero, then MINKILL and MAXKILL are computed relative to the original target value. In addition, if IMATCH is 100, then the routine prints the WADOUT variables, VTO, VT, VTZO (original target value), FLGMN, FLGMX, SVTMIN, SVTMAX, VTMIN, VTMAX, and ALPHA.

The Do loop ending at statement 18 is used to flag all groups with fixed weapons on this target as active. This prevents their being removed during processing and maintains the validity of the damage calculations.

In Part II, the processing begins by evaluating the actual effective surviving target value VTEF. It then scans all weapons currently assigned to calculate the output quantities DPMN and IDPMN. If any weapon now on target fails to destroy a fraction of the original value greater than MINDAMAG, the weapon is flagged for immediate removal (statement 15). At the same time, the groups already assigned are flagged with INACTIVE = -100 to eliminate any possibility that they would be erroneously set inactive. (WADOUT never exits with INACTIVE set negative. A weapon group flagged with a -100 is always reset to INACTIVE = 0 before the routine exits.) (Do 14 loop)

Basically, the processing in Part III is concerned with scanning all potential weapons to calculate the output quantities PPMX, IPPMX, PVRMX, and IPVRMX. Any weapon which would fail to destroy a fraction of the original value greater than MINDAMAG will be ignored in these calculations. Thus, it could never be allocated to the target.

At the same time, however, the values for the array INACTIVE(G) are established. The INACTIVE array for each target is permanently stored on the WPNTGT files, as it was originally computed by GETDATA, with only two values - zero for weapons in range of the target, and 100 for weapons out of range. Consequently, when these data are read on successive passes for each target, these initial values are automatically restored.

External to WADOUT the INACTIVE array is treated as if it has only two values - zero for active weapon groups, and nonzero for inactive groups. WADOUT makes weapons temporarily inactive relative to a specific target by setting INACTIVE equal to either 2,000 or 30,000.

If WADOUT exits with either value, 2,000 or 30,000, the weapon is treated as temporarily inactive in exactly the same way. The difference between 2,000 and 30,000 is relevant only if WADOUT recycles without exiting.

WADOUT will recycle if, after all potential weapons are examined, it is found (upon return to Part 2) that there are no potential weapons that appear profitable and, moreover, that the required kill probability, MINKILL, has not been achieved. In this case the value of ALPHA is increased to make the target seem more valuable and the evaluation is repeated. When this occurs, weapons tentatively set to INACTIVE = 30,000 are reset to 0 and the decision to inactivate them is reexamined.

If WADOUT exits with INACTIVE = 30,000, it is always set to 2,000 if WADOUT is called again.

The operation of Part III of WADOUT can now be summarized as follows. In the execution of this Do loop inactive weapons, INACTIVE = 100, 2,000, and 30,000 (30,000 except on a recycling pass), are skipped. All active weapons, INACTIVE = 0, -100 (or 30,000 on a recycling pass), are evaluated. Those for which INACTIVE is not negative are then considered to determine whether they should be made inactive.

This consideration (lower half of the flowchart) is as follows:

1. Any weapon that still shows a positive potential profit remains active.

2. Any weapon which does not show a positive profit against the full target value (in the absence of other weapons) is made inactive.

3. If there are other weapons on the target the weapon remains active unless its efficiency is less than .1. This reduces the chance that an inactive weapon could be come profitable if some other weapon currently on the target were removed.

4. If the efficiency is less than .1, it is made inactive unless there are already unprofitable weapons assigned. In this case the decision to make it inactive is postponed until these unprofitable weapons are removed.

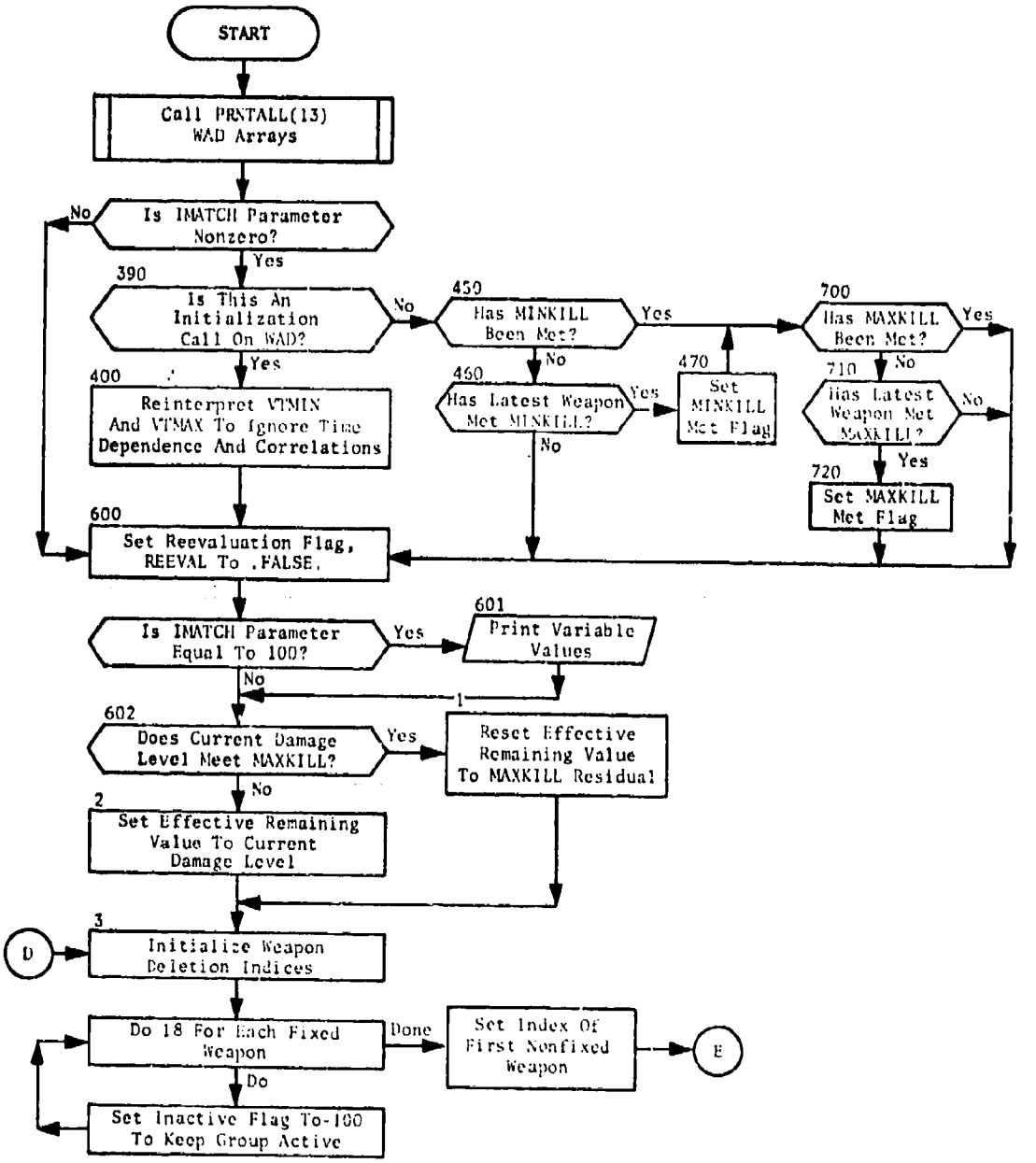


Fig. 56. Subroutine WADOUT
(Sheet 1 of 3)

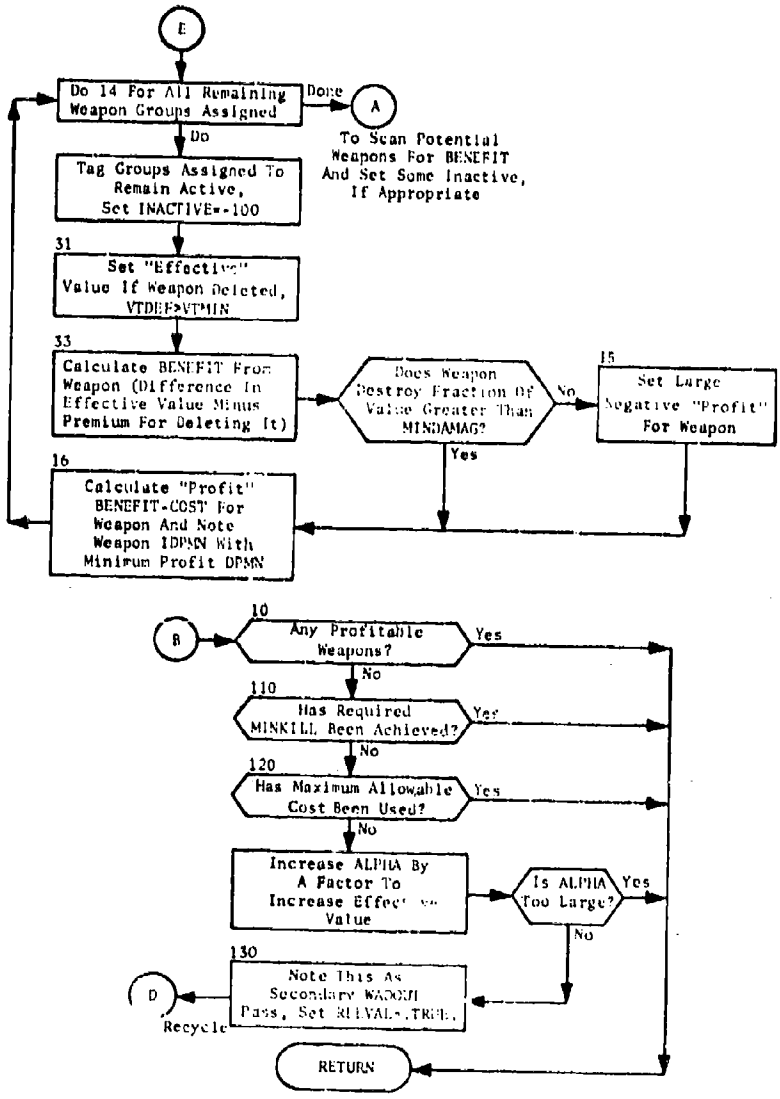


Fig. 56. (cont.)
(Sheet 2 of 3)

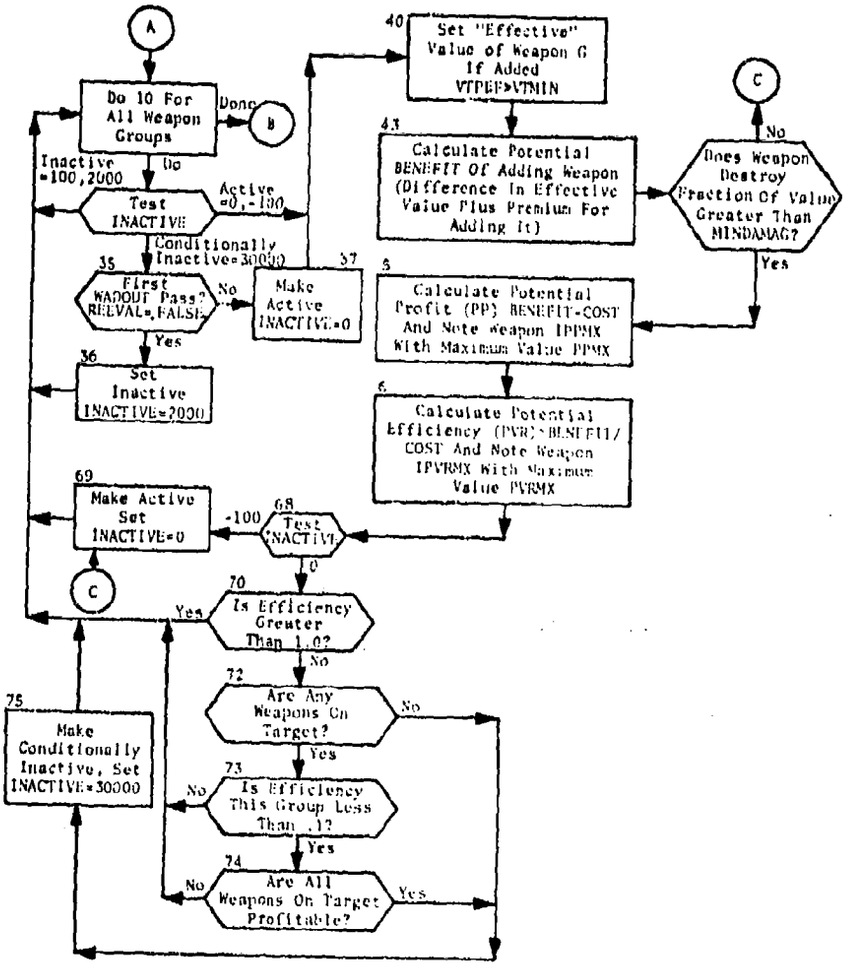


Fig. 56. (cont.)
(Sheet 3 of 3)

CHAPTER 5
PROGRAM ALOCOUT

PURPOSE

The main purpose of ALOCOUT is to rearrange the output of program ALOC into a convenient input form for programs FOOTPRNT and/or program POSTALOC. In addition, ALOCOUT is responsible for selecting optimum DGZs (desired ground zeros, also called weapon aim points) for weapons allocated to target complexes.

Program ALOC supplies ALOCOUT with a file, ALOCTAR, which provides data for each target, specifying the weapon groups assigned to each target together with associated targeting data. ALOCOUT extracts data from these records and reorganizes the extracted data by weapon group, giving for each weapon group the number of strikes and the specific targets assigned through each penetration corridor, plus associated data relating to these targets. ALOCOUT is also responsible for computing any aiming offsets required by the plan. In the case of simple targets, or multiple targets, these offsets are simply set to zero. In the case of complex targets, which can have several elements at slightly different coordinates, ALOCOUT employs subroutine DGZSEL (desired ground zero selector) to select optimum aim points within the target complex.

INPUT FILES

The input files used by program ALOCOUT are the BASFILE and ALOCTAR files. The BASFILE is prepared by program PREPALOC and is used to obtain the weapon data and target characteristics of multiple and complex targets, which are used in ALOCOUT processing. The ALOCTAR file is output by program ALOC and contains allocation of weapons to targets (in target order).

OUTPUT FILES

Program ALOCOUT produces only one output file, the TMPALOC file. This file contains nearly the same data as the ALOCTAR file, except that the aim point offsets for complex targets are included, and the file is ordered by the weapon group. The TMPALOC record formats are shown in table 20.

Table 20. TMPALOC File Format
(Sheet 1 of 3)

HEADER RECORD FOR EACH WEAPON GROUP

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
STRKSUM	1	KGROUP	Weapon group index number
	1	NTSTRK	Total number of strikes for weapon group
	1	NCORR	Number of corridors used by weapon group
	30	NSTRK	Number of strikes per corridor (in decreasing order)

HEADER RECORD FOR EACH CORRIDOR USED BY WEAPON GROUP

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
/2/	1	NT	Number of targets to be attacked through the corridor
	1	JGROUP	Weapon group index number
	1	JCORR	Corridor index number

BOMBER RECORD FOR EACH CORRIDOR AND BOMBER GROUP

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
/2/	NT	INDEXNO	Target index number
	NT	TGTLAT	Target latitude
	NT	TGTLONG	Target longitude
	NT	TIMEPREM	Time premium factor

Table 20. (cont.)
(Sheet 2 of 3)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
/2/ (cont.)	NT	IDEPEN	Depenetration corridor index
	NT	DISTOUT	Distance from target to point of depenetration
	NT	DISTREC	Distance from target to recovery point
	NT	ATTRLOC	Local target defense potential
	NT	RVAL	Relative value of target
	NT	DELAT	Offset longitude for weapon delivery
	NT	DELONG	Offset longtidue for weapon delivery
	$\frac{(NT-1)}{32}+1$	IBFIX	Weapon fixed assignment indicator (logical array)
	NT	DESIG	Target designator code
	NT	TASK	Target task code
	NT	CNTRYLOC	Target country location code
	NT	FLAG	Target flag code

MISSILE RECORD FOR EACH MISSILE GROUP

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
/2/	NT	INDEXNOM	Target index number
	NT	TGTLATM	Target latitude
	NT	TGTLONGM	Target longitude
	NT	RVALM	Relative value of target
	NT	DELATM	Offset latitude for weapon delivery

Table 20. (cont.)
(Sheet 3 of 3)

ASSOCIATED COMMON	LENGTH	VARIABLE OR ARRAY	DESCRIPTION
/2/ (cont.)	NT	DELONGM	Offset longitude for weapon delivery
	$\frac{(NT-1)}{32}+1$	IBFIX	Weapon fixed assignment indicator (logical array)
	NT	DESIGM	Target designator code
	NT	CNTRYLCM	Target country location code
	NT	FLAGM	Target flag code

CONCEPT OF OPERATION

ALOCOUT has two essentially separate data processing phases, ALOCOUT and ALOCOUT2. ALOCOUT reads the entire target file (ALOCTAR) supplied by program ALOC, one record at a time, calling subroutines PROCSIMP, PROCMULT, or PROCCOMP to process each simple, multiple, or complex target, respectively. These routines extract the essential data from the input record and, for each weapon strike, cause a record to be written by PROCSIMP on an intermediate file in a standard form that can later be sorted in ALOCOUT2. All of this processing, including that done by subroutine DGZSEL for complex targets, takes place during the ALOCOUT phase of the program while the ALOCTAR file is being read in.

The main function of PROCSIMP is the writing of this intermediate file which is the input for ALOCOUT2. Subroutine ALOCOUT2 sorts the intermediate file by penetration corridor within weapon group, does the final processing, and writes the TMPALOC file (see figure 57). Figure 58 is the detailed flowchart for ALOCOUT.

Phase One Processing: ALOCOUT

Recall that the subroutines PROCSIMP, PROCMULT, and PROCCOMP are used to process simple, multiple, and complex target data (read from the ALOCTAR file), respectively (see figure 57). These routines extract the necessary data from an ALOCTAR input record and for each weapon strike cause a record to be written by PROCSIMP on an intermediate file in a standard form.

The main function of PROCSIMP is the writing of this intermediate tape which is the input for ALOCOUT2. For simple targets, which need no special processing, PROCSIMP is called directly by ALOCOUT. PROCSIMP writes a 23-word data block for each weapon allocated to the target, containing the necessary data relating to the target and the weapon group. Control then returns to ALOCOUT where the next target block is read from ALOCTAR.

A multiple target represents two or more identical targets whose geographic locations are in the same vicinity (and whose index numbers, as game objects, are consecutive). These targets are represented as a "multiple target" in the input to the allocator, so that program ALOC can save time by making only one assignment of weapons for all elements of the multiple target. This assignment then represents an identical allocation for each of the targets. However, for the detailed plans generated by POSTALOC, separate coordinates must be specified for each target and specific missiles or aircraft must be assigned to each from weapon groups specified.

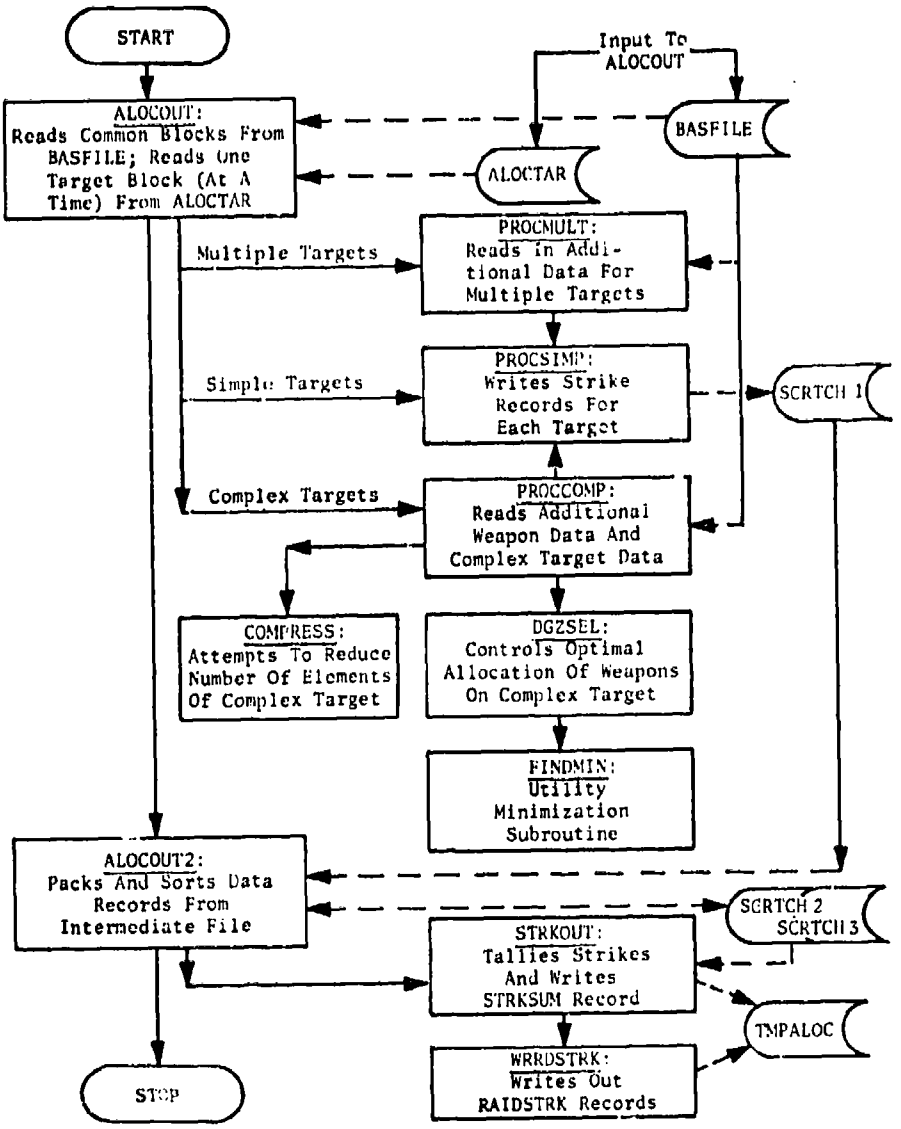


Fig. 57. Calling Hierarchy of ALOCOUT Subroutines

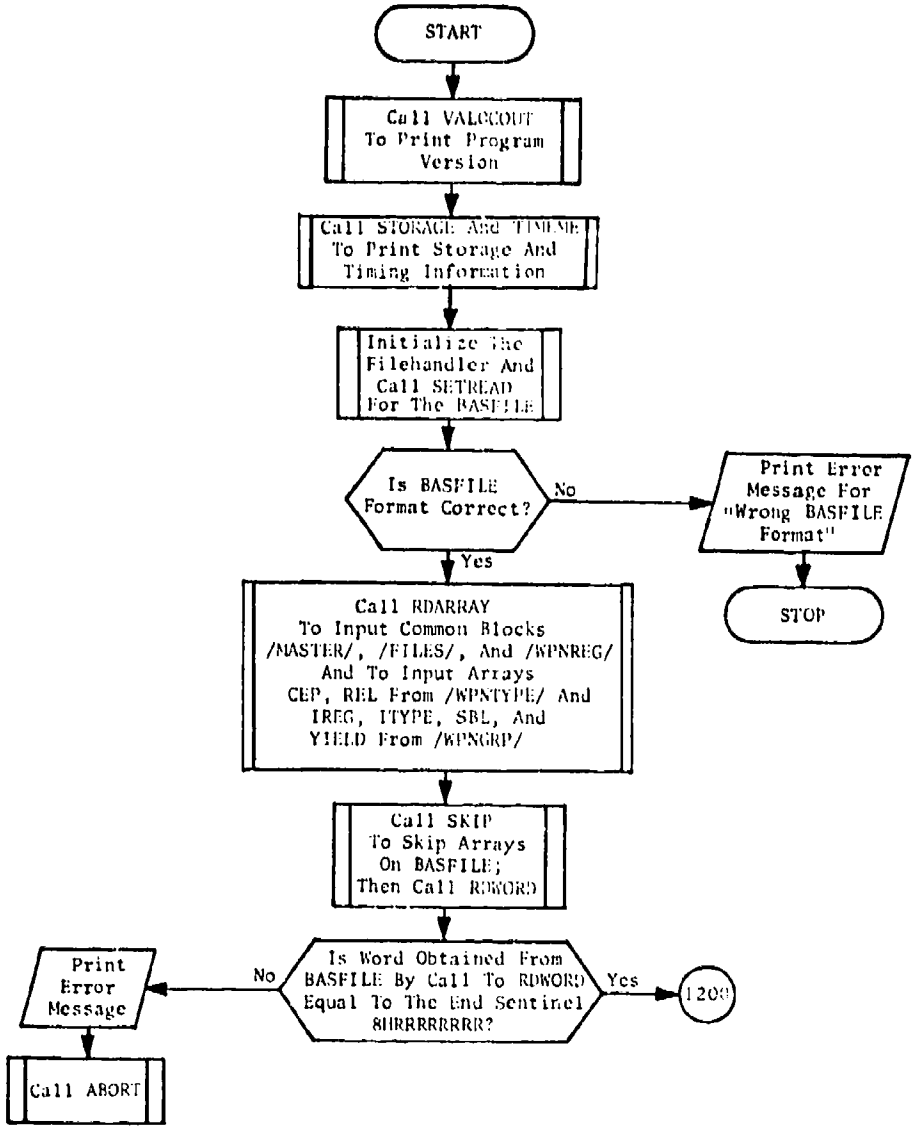


Fig. 58. Program ALOCOUT
(Sheet 1 of 6)

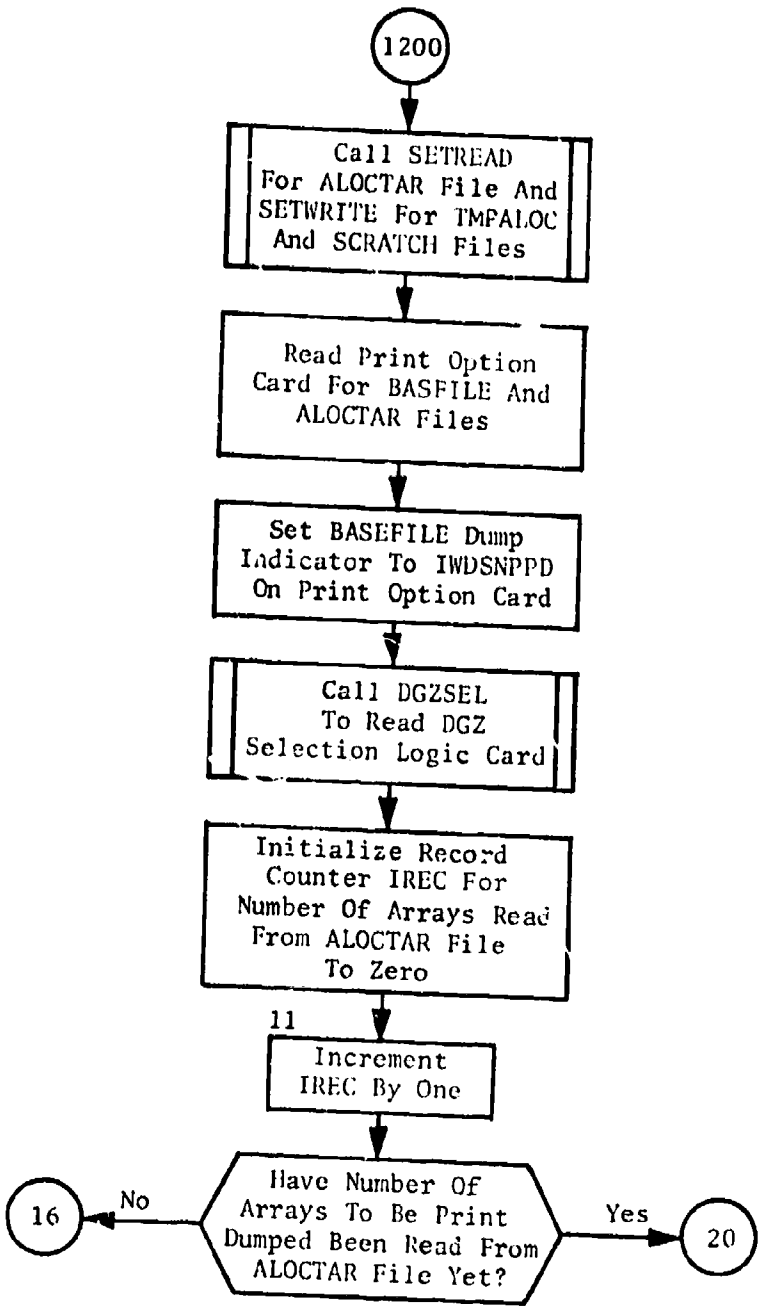


Fig. 58. (cont.)
(Sheet 2 of 6)

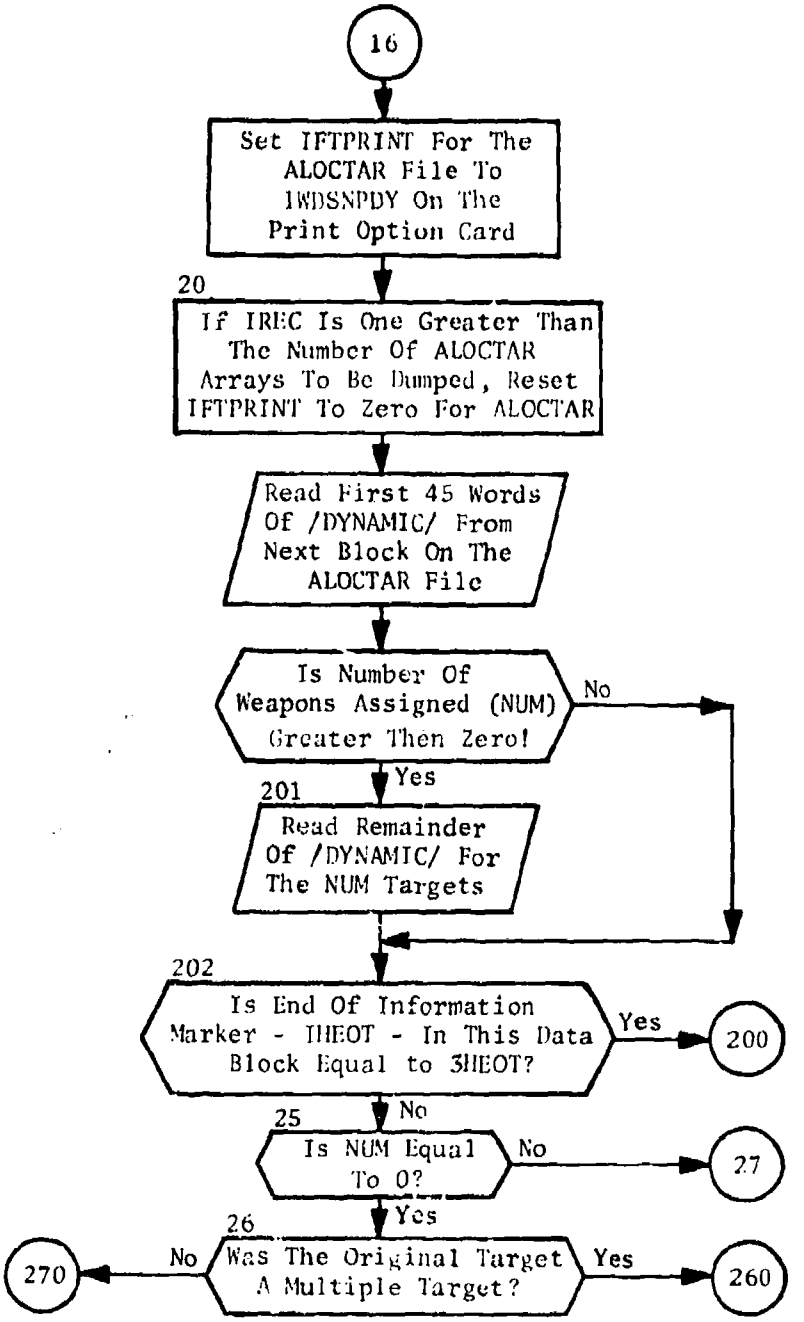


Fig. 58. (cont.)
(Sheet 3 of 6)

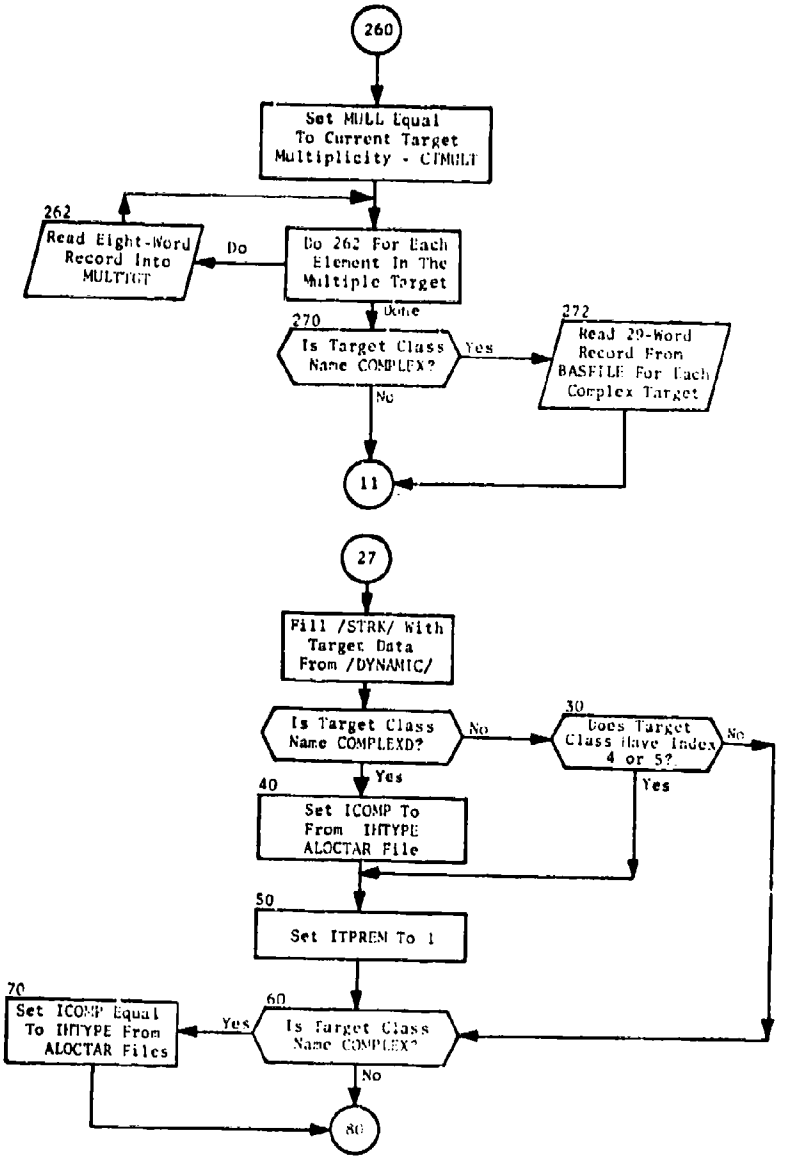


Fig. 58. (cont.)
(Sheet 4 of 6)

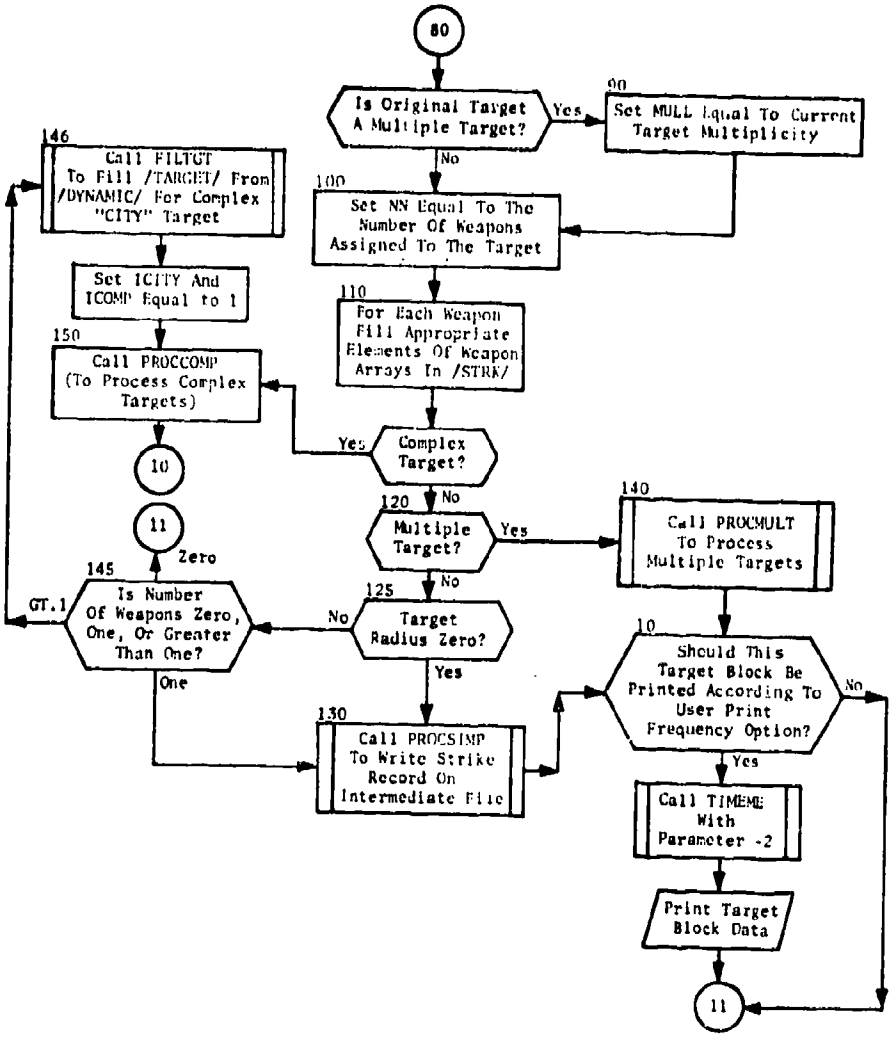


Fig. 58. (cont.)
(Sheet 5 of 6)

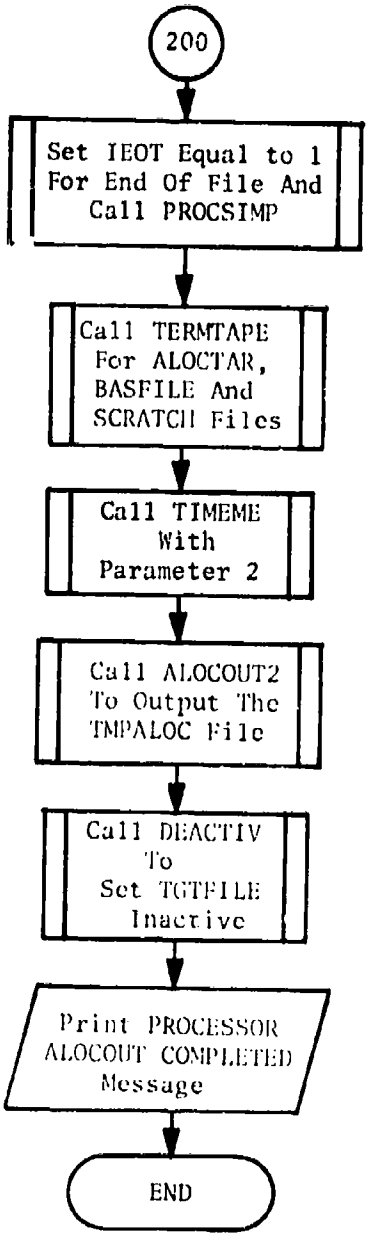


Fig. 58. (cont.)
(Sheet 6 of 6)

When ALOCOUT encounters a multiple target, PROCMULT is called to obtain the additional information needed. For each target element of the multiple target PROCMULT reads from the POSTDATA portion of the BASFILE one data block to obtain the index number and the actual coordinates of the target element. (These records for both multiple and complex targets were written by program PREPALOC on the BASFILE in the same order of occurrence as on the ALOCTAR file, so that no problems exist in file positioning.) PROCMULT then calls PROCSIMP to write the strike data records for each individual target separately; from this point on in the data flow, the individual targets of a multiple target are treated just as if they were separate simple targets.

A complex target (or target complex) is a combination of target elements sufficiently close in geographic location that a weapon on any one of them will have some probability of killing other elements in the complex. Such target complexes are targeted as a unit by the allocator which allocates weapons against their total value, using one set of coordinates. In order to maximize targeting efficiency against such a complex, one must select optimum aim points among the target elements. These aiming offsets are specified relative to the first target element only and are passed on in that form to subsequent programs.

When ALOCOUT encounters a complex target, PROCCOMP is called. PROCCOMP is responsible for assembling the data on a complex target in a form that can be efficiently used for DGZ selection. Each target component of the complex generates a standardized "target element" in the arrays used by DGZSEL. (Targets with more than one hardness component generate more than one such target element, and targets with a specified target radius will generate several elements spread over the area of the target to represent a value spread over the area.)

ALOCOUT has already read weapon group data such as YIELD and CEP which are required for the DGZ selection processing from the BASFILE. PROCCOMP reads a data record for each of the components of the complex target (second section of the BASFILE). If the number of target elements so generated reaches the maximum program dimensions (40), COMPRESS is called to combine elements with similar properties and coordinates. In any case, for efficiency in DGZSEL a call is made on COMPRESS just prior to calling DGZSEL. DGZSEL then uses the data stored in the target element array to select specific aim points (or aiming offsets) for each weapon allocated to the complex. On return from DGZSEL, PROCCOMP calls PROCSIMP to write the intermediate records for the target.

Subroutine COMPRESS recombines those target elements that are near one another and that have approximately the same lethal radius. If maximum program dimensions are reached, COMPRESS loosens its tolerances to assure enough recombination of elements to eliminate this problem.

When ALOCOUT has processed all the targets from the ALOCTAR file and has written a record on the intermediate file for each strike, ALOCOUT2 is called to sort these strike records. At this point, all processing by PROCMULT, PROCCOMP, PROCSIMP, and DGZSEL has been completed. The intermediate file is essentially the only link between ALOCOUT and ALOCOUT2.

Phase Two Processing: ALOCOUT2

ALOCOUT2 (see figure 58) reads each record from the intermediate file and packs data into the appropriate arrays. Since there may be several strikes for each target, to conserve space the target data is stored only once for each target in an array which is kept in core, while the data for each strike is kept in another array. (Included in this data is an index for retrieving the proper target data after sorting.) Because the strike data may exceed the limits of its array in core, two files have been reserved for sorting and merging this data. The first time the strike array is filled, the data is sorted by weapon group and corridor and written out on the first sort file. The program then continues reading records from the intermediate file and packing the data until it either reads all the records on the file, or again fills the strike arrays. It then sorts the strike data currently in core, and proceeds to read the data back in, one word at a time, from the first sort file, and writes the data from the sort file and from the sorted data in core onto the second sort file, merging the data into the proper order.

Processing continues in this manner until all the strike data is contained on one file in order by weapon group and by penetration corridor within each weapon group. At this point, control is turned over to STRKOUT which writes the TMPALOC file (the input file for FOOTPRNT).

STRKOUT reads the sorted data back into core in several batches as storage permits. For each weapon group it tallies the number of strikes assigned through each penetration corridor. When all the data for a single weapon group has been tallied, STRKOUT sorts the penetration corridors in descending order of the number of strikes, and writes a record, common /STRKSUM/, on the intermediate tape, TMPALOC. This record gives the weapon group number, the total number of strikes for that group, the number of penetration corridors used, and the number of strikes through each corridor.

STRKOUT then calls WRRDSTRK for each corridor, which writes a record of the target data for all strikes through that corridor (common /RAIDSTRK/). Thus, the TMPALOC file consists of a STRKSUM record heading the data for each weapon group, followed by a series of RAIDSTRK records, one for each penetration corridor used by that weapon group.

COMMON BLOCK DEFINITION

The external common blocks used by program ALOCOUT in processing input/output (I/O) files are shown in table 21. Table 22 describes the additional common blocks used internally by program ALOCOUT.

Table 21. Program ALOCOUT External Common Blocks
(Sheet 1 of 4)

INPUT DATA FROM BASFILE

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
MASTER	IHDATE	Date of run initiation
	IDENTNO	Run identification number
	ISIDE	Attacking side
	NRTPT	Number of route points
	NCORR	Number of penetration corridors
	NDPEN	Number of depenetration corridors
	NRECOVER	Number of recovery bases
	NREF	Number of directed refuel areas
	NBNDRY	Number of boundary points
	NREG	Number of command and control regions
	NTYPE	Number of weapon types
	NGROUP	Number of weapon groups
	NTOTBASE	Total number of bases
	NPAYLOAD	Number of payload types
	NASMTYPE	Number of ASM types
	NWHDTYPE	Number of warhead types
	NTANKBAS	Number of tanker bases
	NCOMPLEX	Number of complex targets
	NCLASS	Number of weapon classes (2)
	NALERT	Number of alert conditions (2)
	NTGTS	Number of targets
	NCORTYPE	Number of penetration corridor types
	NCNTRY	Number of distinct country codes
FILES	TGTFILE(2)**	Target data file
	BASFILE(2)	Data base information file
	MSLTIME(2)	Fixed missile timing file
	ALOCTAR(2)	Weapon allocation by targets file
	TMPALOC(2)	Temporary allocation file
	ALOCGRP(2)	Allocation by group file
	STRKFIL(2)	Strike file

*Parenthetical values indicate array dimensions. All other elements are single word variables.

**In two-word arrays, first word is filehandler buffer usage number; second word is maximum file length in words. Single variables are logical tape unit numbers.

Table 21. (cont.)
(Sheet 2 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
WPNREG	CCREL(20)	Command and control reliability by command and control region
WGROUP	IREG(200)	Command and control region
	ITYPE(200)	Type index (LTYPE)
	SBL(200)	Probability of survival before launch
	YIELD(200)	Weapon yield (megatons)
WTYPE	CEP(80)	CEP (nautical miles)
	REL(80)	Reliability
TARGET‡	TGTNAME	Target name
	INDEXNO	Target index number
	DESIG	Target designator code
	TASK	Target task code
	CNTRYLOC	Target country location code
	FLAG	Target flag code
	TGTMULT	Target multiplicity
	TGTLAT	Target latitude
	TGTLONG	Target longitude
	TGTRAD	Target radius (nautical miles)
	VTO	Original target value
	M	Number of hardness components
	H(2)	Lethal radius by hardness component
	FVALH1	Fraction of value of first hardness component
	NK	Number of time components
	FVAL(2)	Fraction of value escaping in each time period
	TAU(3)	Time ending each time component
	IHCLASS	Target class name
	ICLASS	Target class number
	IHTYPE	Target type name
	TARGET(25)-TARGET(29)	Not used

‡TARGET is the 29-word record contained on the BASEFILE for each complex target component.

Table 21. (cont.)
(Sheet 3 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
MULTTGT‡‡	NAME	Target name
	INDEX	Target index number
	DSIG	Target designator code
	TSK	Target task code
	CNTRLC	Target country location code
	FLG	Target flag code
	TLAT	Target latitude
	TLONG	Target longitude

INPUT DATA FROM ALOCTAR

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DYNAMIC	TGTNAME	Hollerith target name
	INDEXNO	Index number of target
	DESIG	Target designator code
	TASK	Target task code
	CNTRYLOC	Target country location code
	FLAG	Target flag code
	TGTMULT	Target multiplicity (original)
	TGTLAT	Target latitude
	TGTLONG	Target longitude
	TGTRAD	Target radius
	VTO	Original target value
	M	Number of hardness components (≤2)
	H(2)	Hardness of each component
	VO(2)	Original value of each component
	NK	Number of time periods (≤3)
	FVAL(3)	Fraction value escaping in each period
	TAU(3)	Time ending each period
	IHCLASS	Hollerith target class name
	ICLASSN	Target class number
	IHTYPE	Hollerith target type name
	TARDEF	Local bomber defense factor
	INDYPEN	Depenetration corridor index

‡‡MULTTGT is the 8-word record contained on the BASFILE for each multiple target element.

Table 21. (cont.)
(Sheet 4 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DYNAMIC (cont.)	DISTDF	Distance from target to end of depenetration
	DISTDG	Distance from target to recovery base
	NBLN	= number of terminal ballistic missile intereceptors if a STALL allocation = minus the number of intereceptors if a DEFALOC allocation
	CTMULT	Current target multiplicity
	VI	Value remaining after allocation of weapons
	TGTWT(3)	Target weighting values
	PAYOFF	Payoff on this target [VFO-VT-COST]
	COST	Sum of Lagrange multipliers of all weapons allocated to target
	PROFIT	PAYOFF - COST
	DPROFIT	Difference in profit between passes
	WRTEST	Test value for weight rates
	IHEOT	End of information marker
	NUMFIX	Number of weapons allocated by fixed assignment capability
	ITGT	Target number
	NUM	Number of weapons assigned
	IG(30)	Group number of assigned weapons
	KORR(30)	Weapon penetration corridor
	VID(30)	Relative value of weapon allocation
	PEN(30)	Weapon penetration probability
	TOARR(30)	Weapon time of arrival on target

Table 22. Program ALOCOUT Internal Common Blocks
(Sheet 1 of 5)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
STRK		This common block primarily contains data moved from /DYNAMIC/ for further processing by PROCSIMP, PROCMULT or PROCCOMP
	NAME	Hollerith target name
	INDEX	Target index number
	DSIG	Target designator code
	TSK	Target task code
	CNTRLC	Target country location code
	FLG	Target flag code
	JHCLASS	Hollerith target class name
	JCLASS	Target class number
	JHTYPE	Hollerith target type name
	TLAT	Target latitude
	TLONG	Target longitude
	IATLOC	Local bomber defense factor
	ITPREM	1 if complex target; 0 otherwise
	IDPN	Depenetration corridor index
	DISTF	Distance from target to end of depenetration
	DISTG	Distance from target to recovery base
	IGG(30)	Group number of assigned weapons
	KOR(30)	Weapon penetration corridor
	DLAT(30)	Latitude of target aim offset
	DLONG(30)	Longitude of target aim offset
	TOA(30)	Weapon time of arrival on target
	RELVAL(30)	RELVAL(I)=VTD(I)/PEN(I) from /DYNAMIC/ block
	PENN(I)	Weapon penetration probability
	MULL	Current target multiplicity if multiple target; zero otherwise
	ICOMP	IHTYPE from /DYNAMIC/ block if complex target; 1 if target is city (area target); 0 otherwise
	N	Number of weapons assigned
	ISTAPE	1 if BASFILE is used; 0 if program in debug mode
	IEOT	1 when last record read from ALOCTAR file; 0 otherwise
	NNFIX	Number of weapons allocated by fixed assignment capability

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 22. (cont.)
(Sheet 2 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
SCRATCH	ISCRATCH	The logical unit number of the intermediate file written by PROCSIMP and read by ALOCOUT2
CITY	ICITY	1 if the target is a city (or area target); 0 otherwise
ISKIPDGZ	ISKIPDGZ	Use indicator for DGZSEL. Normally it is 0. COMPRESS resets it to 1 if more than 20 calls to it are made to reduce the number of target elements for a complex target; DGZSEL is not used again for the target in this case.
STRKTGT		STRKTGT contains the variables output to the intermediate file by PROCSIMP. The variables are the same as those in STRK but pertain to only a single target/weapon combination.
	NAMEX	Target name
	INDEXX	Target index number
	DSIGX	Target designator code
	TSKX	Target task code
	CNTRLCX	Target country location code
	FLGX	Target flag code
	JHCLASSX	Target class name
	JCLASSX	Target class number
	JHTYPEX	Target type name
	TLATX	Target latitude
	TLONGX	Target longitude
	ATLOCX	Local bomber defense factor
	TPREMX	1 if complex target; 0 otherwise
	IDPEN	Depenetration corridor index
	DOUT	Distance from target to depenetration
	DREC	Distance from target to recovery base
	IGX	Weapon group number
	KORRX	Weapon penetration corridor
	DLATX	Latitude of weapon delivery aim offset

Table 22. (cont.)
(Sheet 3 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
STRKTGT (cont.)	DLONGX	Longitude of weapon delivery aim offset
	TOAX	Weapon time of arrival on target
	RELVALX	VTD(I)/PEN(I) from /DYNAMIC/ for a given weapon I
	IIFIX	Fixed assignment flag
1		As used by PROCCOMP, DGZSEL and associated subroutines
	XO(J), YO(J)	Coordinates of target element J
	VI(J)	Initial target element values
	VTOA(J,I)	Value of target element J immediately following arrival of weapon I
	S(J,I)	Survival probability of target element J relative to weapon I
	VEFF(J,I)	Effective value of target element J relative to weapon I
	X(I), Y(I)	Offset coordinates of weapon I
	PDEL(I)	Probability of delivery of weapon I
	ERDEL(I)	Error in delivery of weapon I
	YDSCL(I)	Scaled yield for weapon I
	VESC(J)	Intermediate computational value used in subroutine VAL in determination of total escaping target value
	RADL(J)	Lethal radius of target element J
	NI	Number of weapons for complex
	NJ	Number of target elements for complex
	DUM1(18)	Not used
1		As used by ALOCOUT2 and STRKOUT
	ITD1(4000)	Storage for packed target data
	IWD1(900)	Storage for packed weapon data
11		As used by FINDMIN
	H(60,60)	H matrix used during minimization procedure

Table 22. (cont.)
(Sheet 4 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
11 (cont.)	X1(60), X2(60), X3(60), X4(60)	First, second, third and fourth trial aim point offset vectors
	SIG(60)	Offset aim point increment vector
	S(60)	Modified gradient direction vector
	GO(60)	Initial gradient component vector
	G(60)	Current gradient component vector
	DX(60)	Trial offset aim point increment vector
	Y(60)	Gradient component increment vector
	DUMX(700)	Not used
		as used by ALOCOUT2 and STRKOUT
	ITD2(4000)	Storage for packed target data
	IWD2(900)	Storage for packed weapon data
12		as used by ALOCOUT2 and WRRDSTRK
	ITD3(4000)	Storage for packed target data
13	ITD4(4000)	Storage for packed target data
2 (formerly RAIDSTRK)		as used by WRRDSTRK
	NT	Number of strikes in corridor
	JGROUP	Group index number
	JCORR	Corridor index number
	INDEXNO/INDEXNOM	Target index numbers
	TGTLAT/TGTLATM	Target latitudes
	TGTLONG/TGTLONGM	Target longitudes
	TIMEPREM	"COMPLEXD" target indicators
	IDEPEN	Depenetration corridor indices
	DISTOUT	Distances from targets to depenetration corridors
	DISTREC	Distances from targets to recovery points
	ATTRLOC	Local target defense potentials
	RVAL/RVALM	Relative values of targets

Table 22. (cont.)
(Sheet 5 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
2 (formerly RAIDSTRK) (cont.)	DELAT/DELATM	Target offset latitudes
	DELONG/DELONGM	Target offset longitudes
	DESIG/DESIGM	Target designator codes
	CNTRYLOC/CNTRYLCM	Target country location codes
	FLAG/FLAGM	Flag codes for targets
	IBFIX	Fixed assignment indicators for targets
	INDEX	Array used by ORDER and REORDER for storage
ALOC	INOWPNS	Total number of strikes on final sort file
	ITAPEW	Logical unit number of final sort file
KEYS		Keys used for packing data, using subroutines IPUT and IGET
RAID	NS	Number of strikes in corridor
	KOR	Corridor index number
	LOC	Index where this corridor begins, in sorted, packed weapon data
	IGRP	Group index number
DATA	IOUTDAT2	Number of words in filehandler snap on first RAIDSTRK write
	IOUTDAT3	Number of words in filehandler snap on second RAIDSTRK write
	JOUTDATP	Frequency of print of RAIDSTRK data, i.e., if JOUTDATP=3, prints every third record
STRKSUM	KGROUP	Group index number
	NTSTRK	Number of strikes in corridor
	NCORR	Internal index for corridor
	NSTRK(30)	Number of strikes by corridor
WGROUP		as used by ALOCOUT2
	ISEQ(900)	Storage for subroutine ORDER

SUBROUTINE ALOCOUT2

PURPOSE:	To sort data written on the intermediate tape by PROCSIMP and call STRKOUT to summarize and write the output.
ENTRY POINTS:	ALOCOUT2
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ALOC, FILABEL, IFTPRNT, ITP, KEYS, MYIDENT, MYLABEL, SCRATCH, STRKTGT, TWORD, WGROUP, 1, 11, 12, 13
SUBROUTINES CALLED:	ABORT, IPUT, KEYMAKE, ORDER, RDARRAY, RDWORD, REORDER, SETREAD, SETWRITE, STRKOUT, TERMTAPE, TIMEME, WRWORD
CALLED BY:	ALOCOUT

Method

ALOCOUT2 (figure 59) reads each record from the intermediate tape and packs the data into the appropriate arrays. Since there are a series of strikes on each target, the target data are stored only once for each target in an array which is kept in core, while the data for each strike are kept in the arrays IWD1 and IWD2 to conserve space. (Included in these data is the index for retrieving the proper target data after sorting.) It is possible that the strike data may exceed the limits of its array. Therefore, two files have been reserved for sorting and merging these data. The first time the strike array is filled, the data are sorted by weapon group and corridor and written out on the first sort file. The program then continues reading records from the intermediate file and packing the data until it either reads all the records on the file, or again fills the strike arrays. It then sorts the strike data currently in core and proceeds to read the data back in, one word at a time, from the first sort tape and writes the data from the sort tape and from the sorted data in core onto the second sort tape, merging the data into the proper order.

Processing continues in this manner until all the strike data are contained on one tape in order by weapon group, and by penetration corridor within each weapon group. At this point, STRKOUT assumes control and writes the TMPALOC output file.

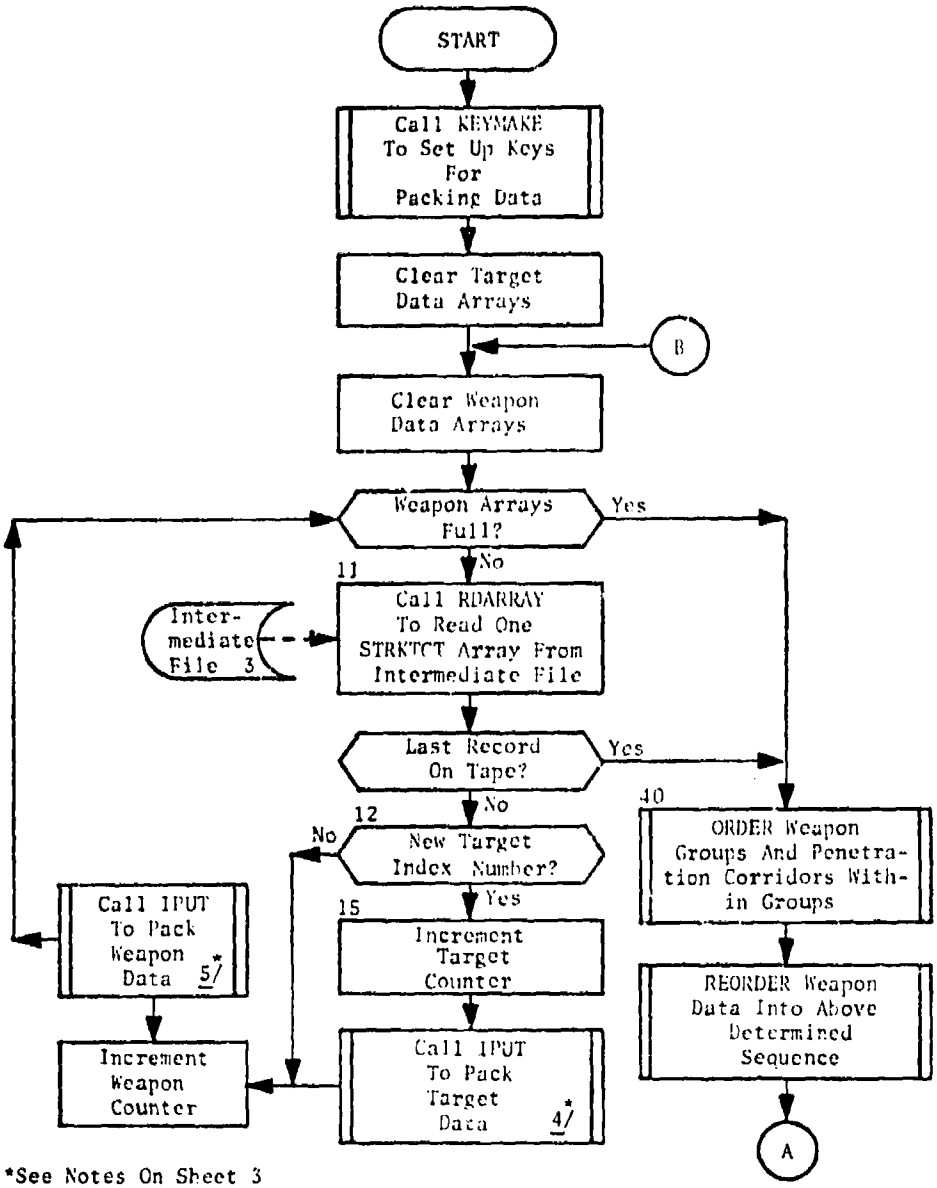
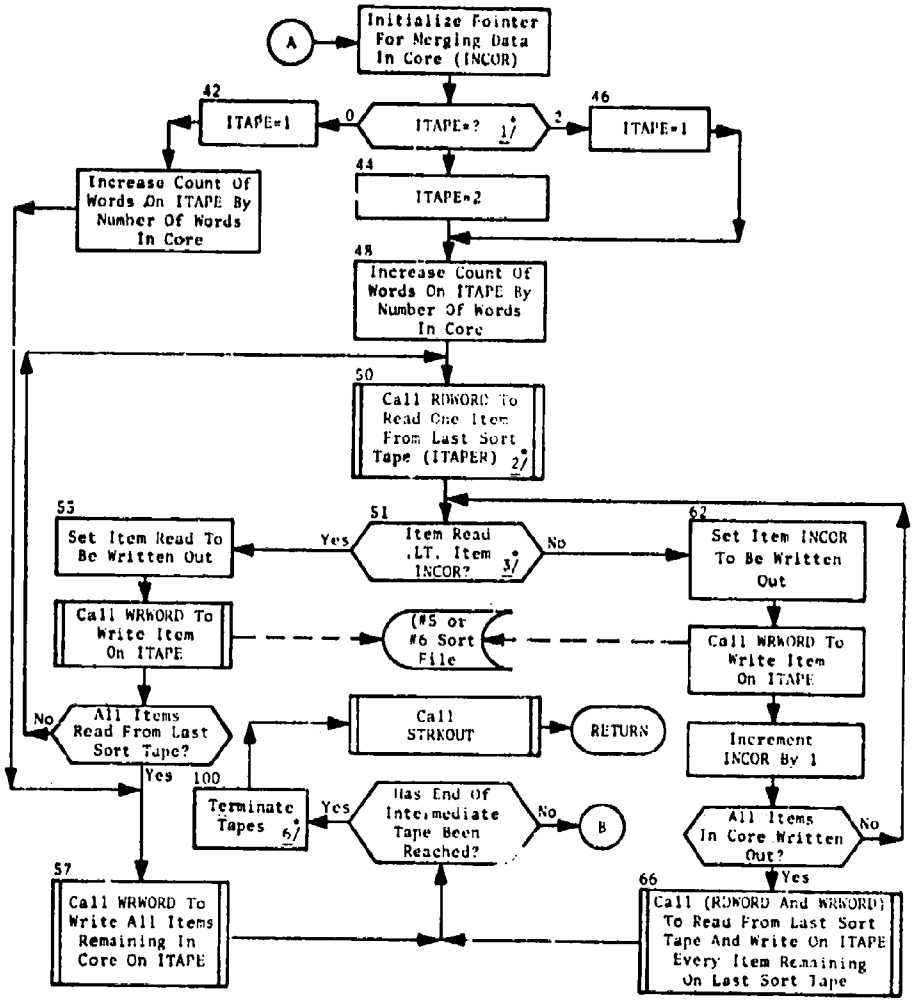


Fig. 59. Subroutine ALOCOUT2
(Sheet 1 of 3)



*See note on Sheet 3

Fig. 59. (cont.)
(Sheet 2 of 3)

Notes:

1. ITAPE is used as a pointer to the (first or second) sort tape being written on. It is used as an index for NWDTAP(2) which gives the number of items currently on each tape. ITAPE1 and ITAPE2 contain the logical unit numbers of the two sort tapes. ITAPEW is set to either ITAPE1 or ITAPE2, whichever tape is being written on, and ITAPER is set to the tape being read.

2. The item read from tape, during sorting and merging, is stored in INWORD(2). NREAD keeps count of the number of words read from tape.

3. INCOR contains the index of the item in IWD1 and IWD2 which is to be written next.

4. IBETA is a running index for storing target data.

5. NEXT is a running index for storing weapon data and, during the sorting and merging phase, is equal to the number of items in IWD1 and IWD2.

6. INOWPNS is set to NWDTAP(ITAPE) when the last tape has been written and thus passes on to STRKOUT the total number of strikes.

Fig. 59. (cont.)
(Sheet 3 of 3)

SUBROUTINE COMPRESS

PURPOSE:	For computational efficiency and/or to avoid exceeding maximum program dimensions, COMPRESS recombines those target elements which are near one another and have approximately the same lethal radius.
ENTRY POINTS:	COMPRESS
FORMAL PARAMETERS:	OPENTOL (type INTEGER). If OPENTOL is 0, distance and lethal radius tolerances will not be eased to decrease the number of target elements. If OPENTOL is 1, the tolerances will be eased.
COMMON BLOCKS:	1, ISKIPDGZ
SUBROUTINES CALLED:	IMIN
CALLED BY:	PROCCOMP

Method

When OPENTOL is zero, COMPRESS merely recombines target elements which are close enough together that their lethal radii nearly coincide. COMPRESS in this mode is called by PROCCOMP just prior to calling DGZSEL in order to improve the efficiency of DGZSEL.

In the event that maximum program dimensions are reached, OPENTOL is set to 1 by PROCCOMP; COMPRESS will then loosen its tolerances, if necessary, to assure enough recombination of target elements to eliminate the problem, at least temporarily. A print is also issued in this case which gives the number of times the tolerances were doubled.

The flowchart for COMPRESS is shown in figure 60.

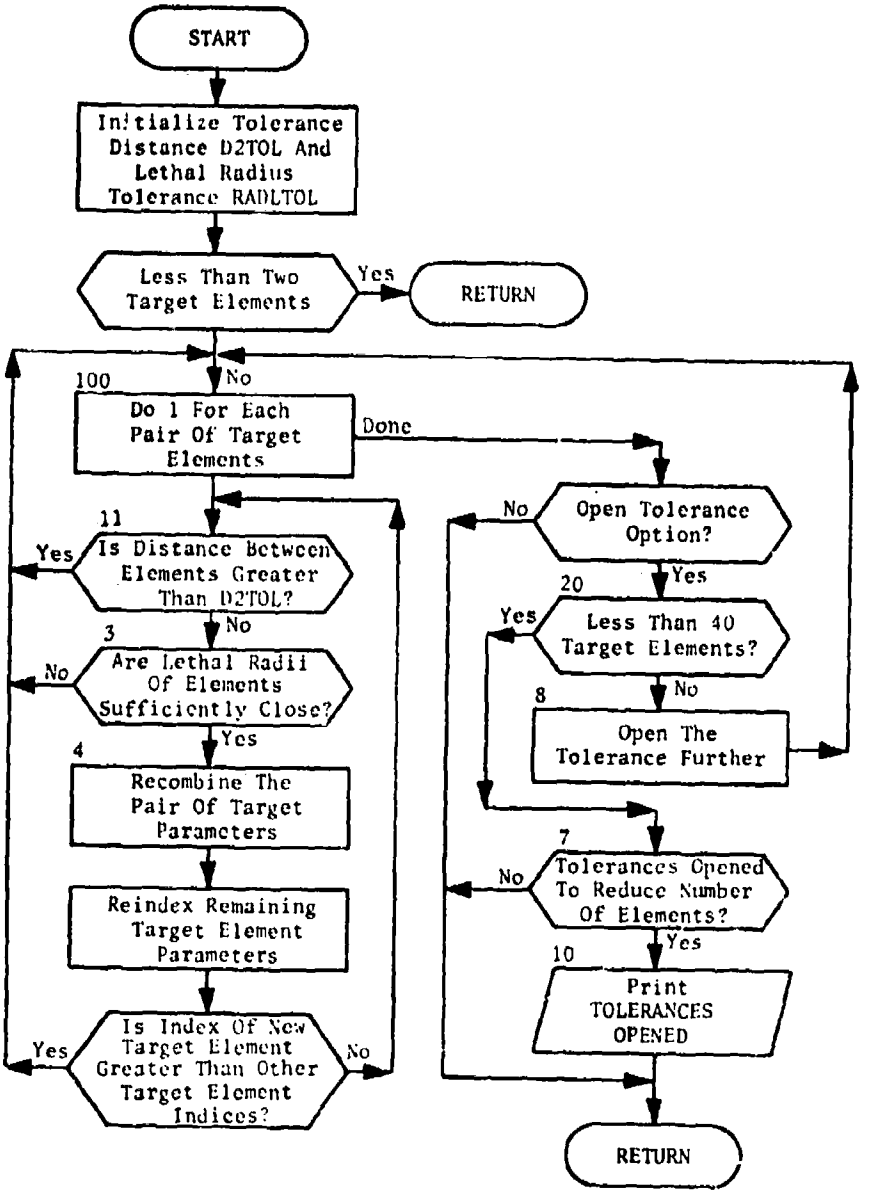


Fig. 60. Subroutine COMPRESS

FUNCTION CUMINV

PURPOSE:	To determine the value X such that Z is the probability that $x \leq X$.
ENTRY POINTS:	CUMINV
FORMAL PARAMETERS:	Z - The probability that $x \leq X$
COMMON BLOCKS:	None
SUBROUTINES CALLED:	None
CALLED BY:	PROCCOMP

Method

Function CUMINV is illustrated in figure 61. By definition,

$$Z = P[x \leq X] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{X} e^{-\frac{t^2}{2}} \cdot dt \text{ for } 0 < Z < 1$$

CUMINV uses the following approximation X^1 for X:

$$X^1 = \pm \left[V - \left(\frac{A_1 + A_2 \cdot V + A_3 \cdot V^2}{1 + B_1 \cdot V + B_2 \cdot V^2 + B_3 \cdot V^3} \right) \right]$$

where

$$V = \sqrt{\ell n \ (1/Q^2)}, \quad Q = Z \quad \text{or} \quad 1 - Z \quad \text{such that } 0 \leq Q \leq .5$$

and

$A_1 = 2.515517$	$B_1 = 1.432788$
$A_2 = .802853$	$B_2 = .189269$
$A_3 = .010328$	$B_3 = .001308$

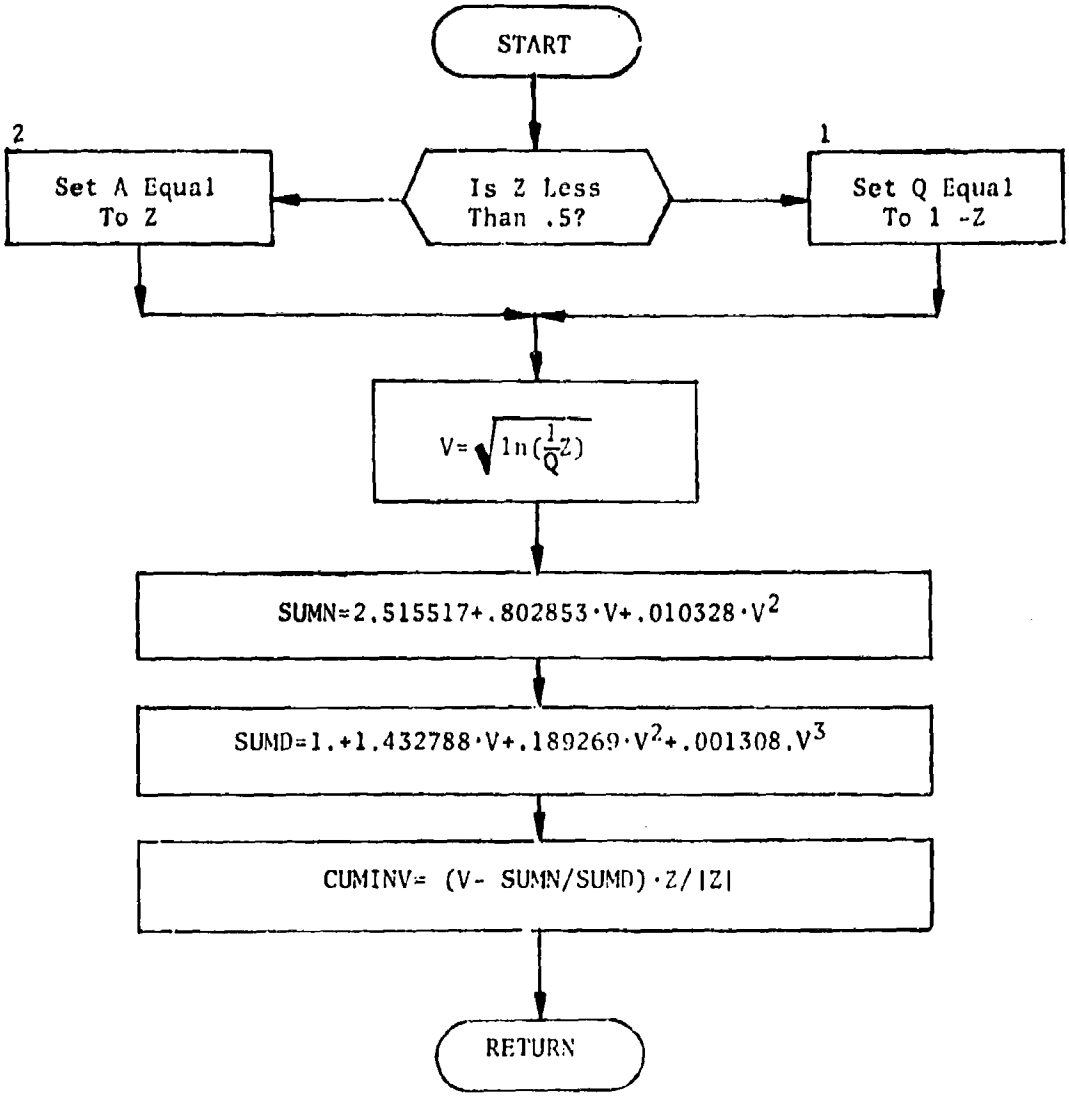


Fig. 61. Function CUMINV

SUBROUTINE DGZSEL

PURPOSE:	DGZSEL is the controlling subroutine for the optimal selection of DGZs for weapons allocated to complex targets.
ENTRY POINTS:	DGZSEL
FORMAL PARAMETERS:	None
COMMON BLOCKS:	1
SUBROUTINES CALLED:	FINDMIN, MOVE, PERTBLD, SEECALC, SEEINPUT, TIMEME, VAL, VMARG
CALLED BY:	PROCCOMP

Method

The optimization of DGZs explicitly considers the time dependence of target value and the time of arrival of warheads. It does not reanalyze the correlation of delivery probabilities, which is assumed to have been treated in the cross targeting provided by the allocator. The selection of DGZs is a two-step process. First, the prescribed warheads are assigned initial coordinates through a "laydown" process in which each successive warhead is targeted directly against that target element where the highest payoff is achieved, taking into account collateral damage to all other target elements. Second, a general-purpose function optimizer FINDMIN is called which calculates the derivatives of the payoff as a function of x and y coordinates of each weapon and adjusts the coordinates to minimize the surviving target value. FINDMIN will exit either after a maximum number of iterations (which are specified on an input card), or after it finds that it can no longer make significant improvements in the payoff. The mathematical representation is given in the Analytical Manual, Volume II, Chapter 3, Calculations.

One data card specifies the options peculiar to DGZSEL. This card input includes specification of the maximum number of iterations to use in FINDMIN, including whether FINDMIN is to be called, and whether various combinations of intermediate prints are desired. Figure 62 illustrates the DGZSEL calling hierarchy; the subroutines are grouped according to how they are used in the selection process. Figure 63 is the DGZSEL flowchart.

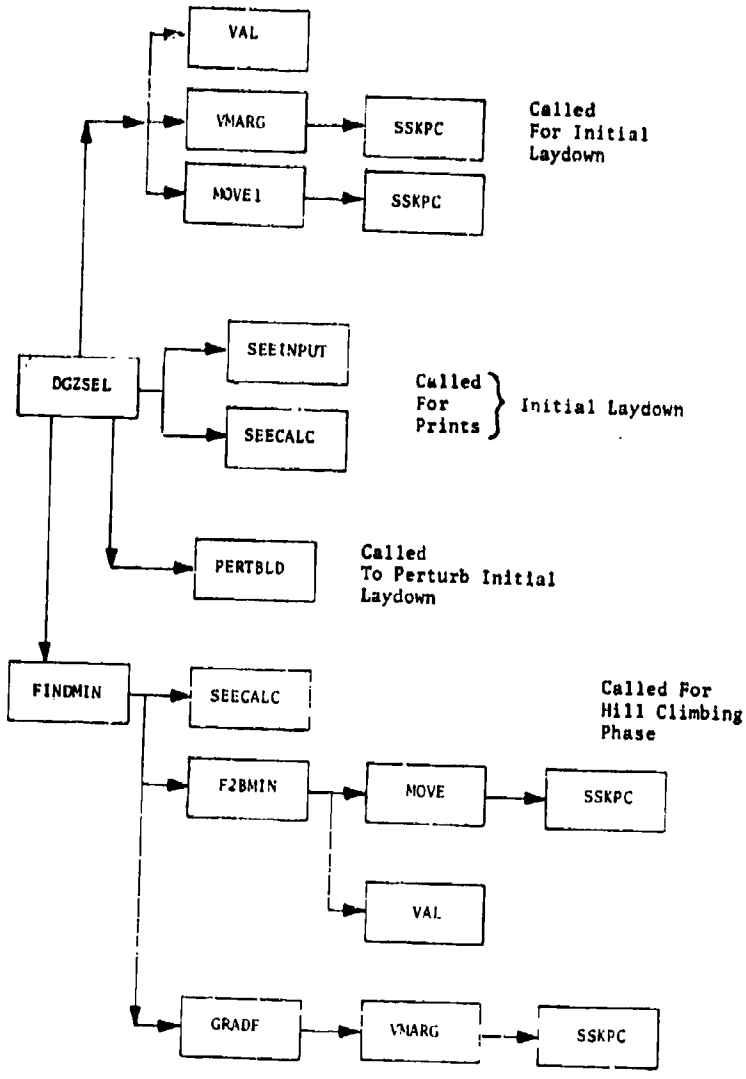


Fig. 62. DGZSEL Calling Hierarchy

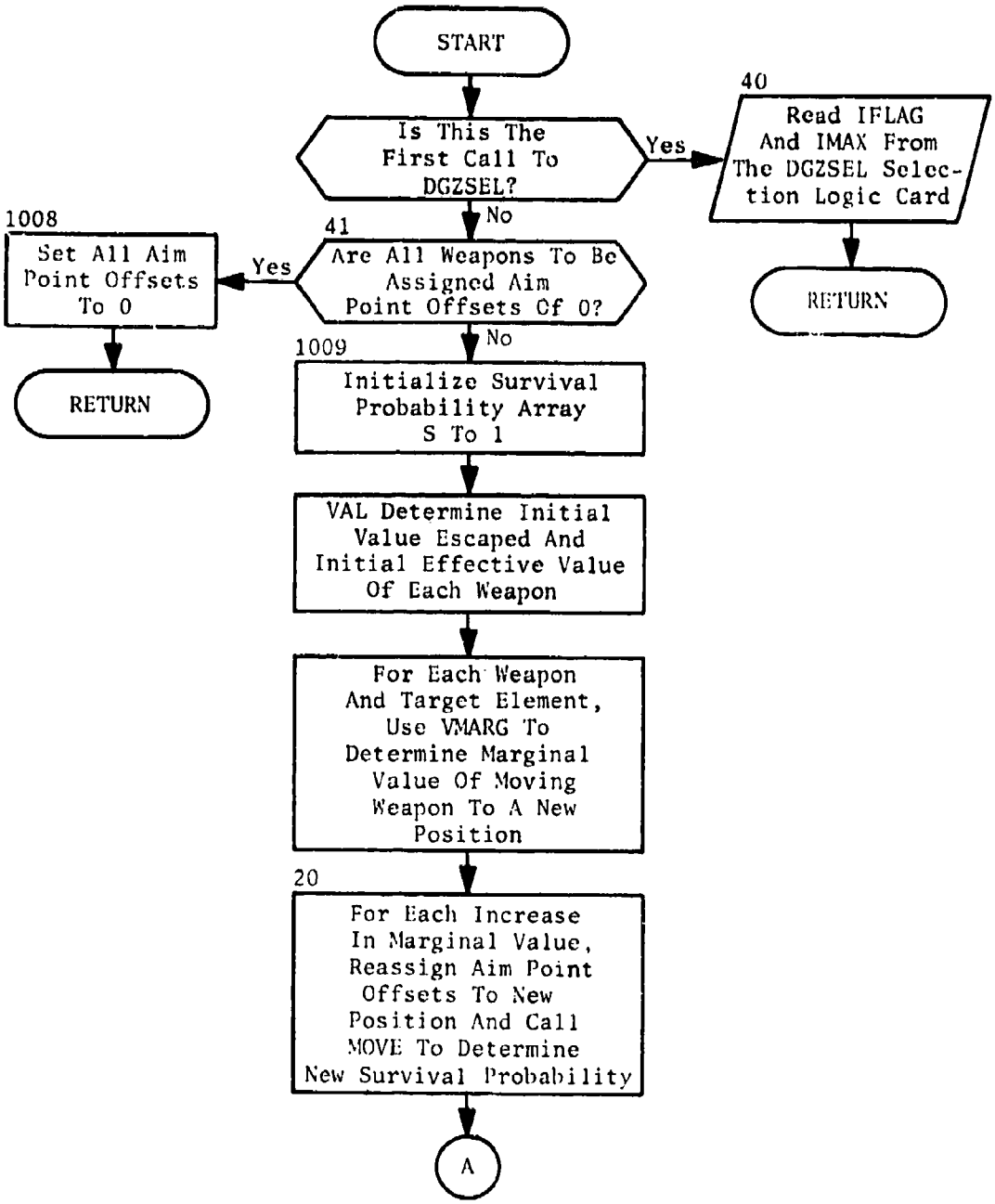


Fig. 63. Subroutine DGZSEL
(Sheet 1 of 3)

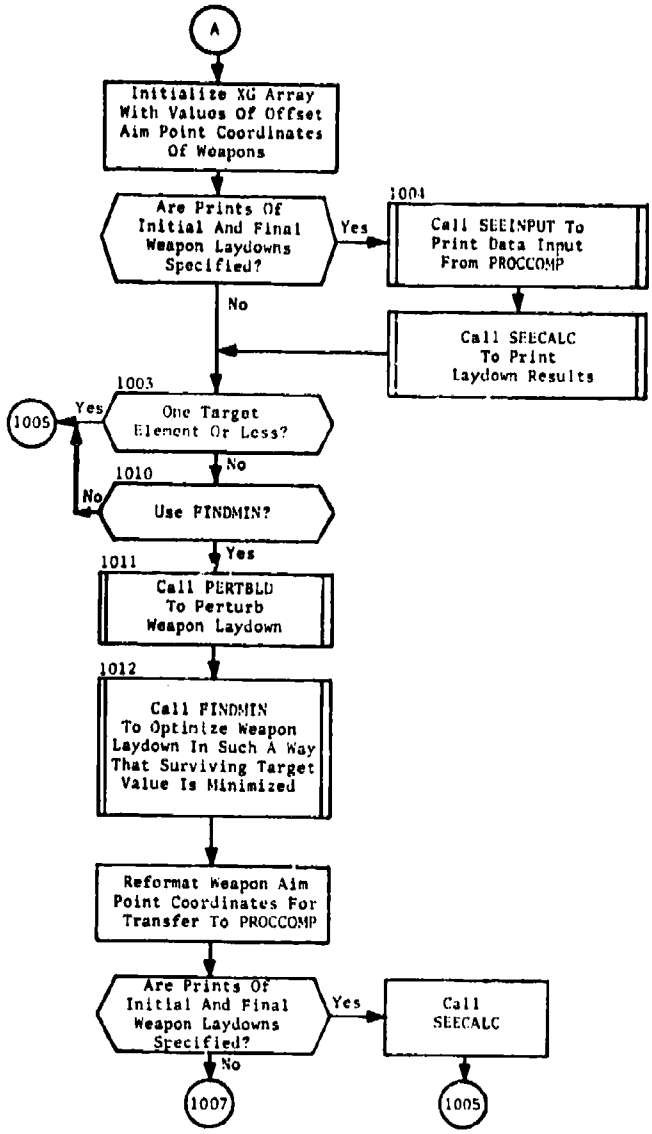


Fig. 63. (cont.)
(Sheet 2 of 3)

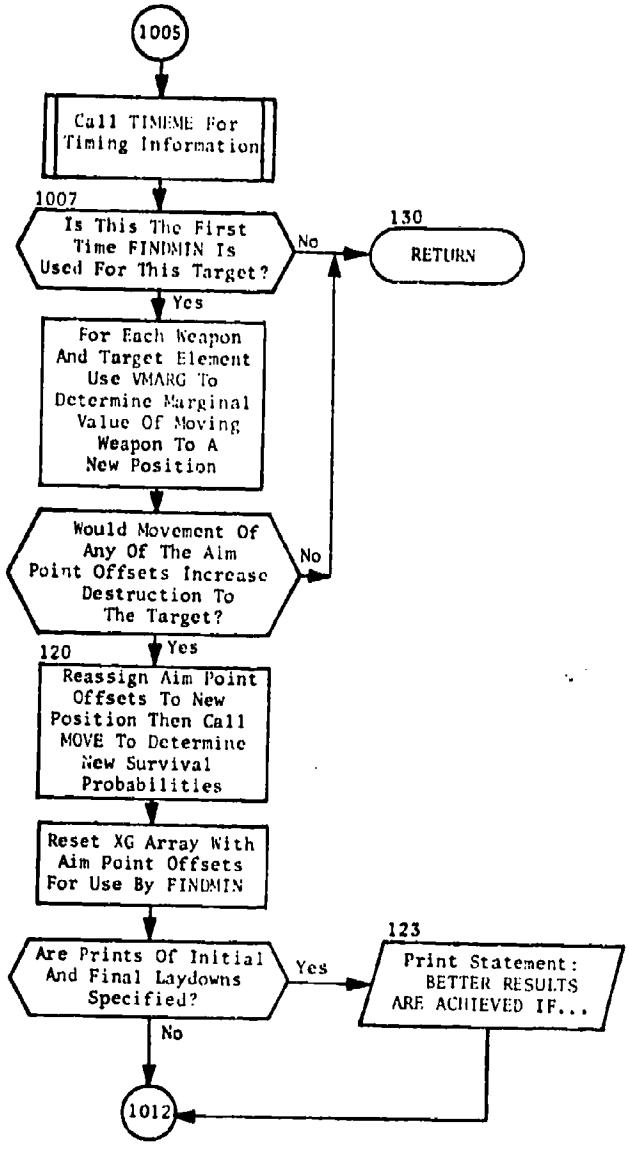


Fig. 63. (cont.)
(Sheet 3 of 3)

FUNCTION ERGOT1

PURPOSE:	To return next number in most uniform ergodic series. (Numbers for up to 10 distinct series can be called for concurrently.)
ENTRY POINTS:	ERGOT1, ERGOT2, ERGOT3
FORMAL PARAMETERS:	I - Index of the series for which the next number is desired
COMMON BLOCKS:	None
SUBROUTINES CALLED:	None
CALLED BY:	PROCCOMP

Method

Depending on whether the entry ERGOT1, ERGOT2, or ERGOT3 is used, the index L is set to 1, 2, or 3, respectively. Then the next number in the Ith ergodic series is calculated. This function is illustrated in figure 64.

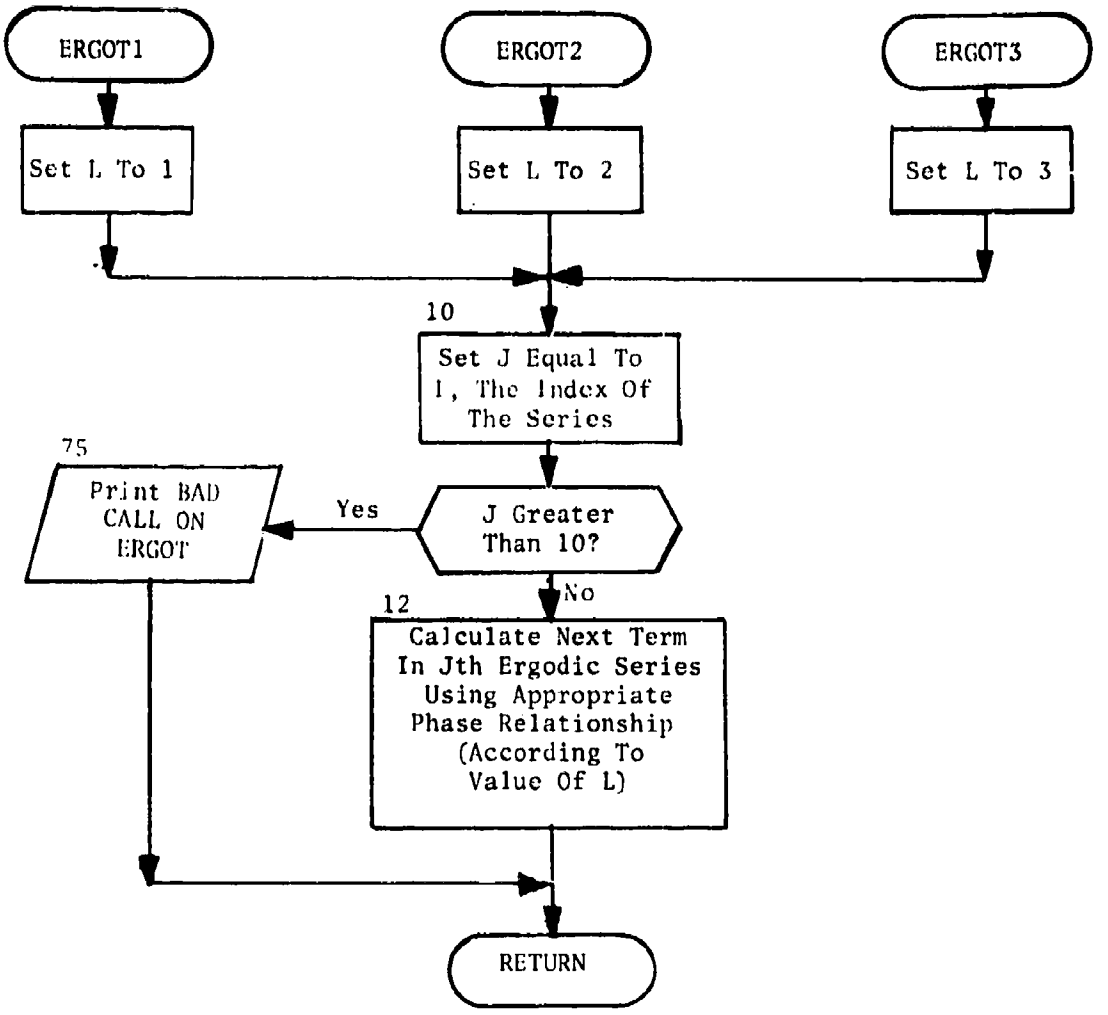


Fig. 64. Function ERGOT1

SUBROUTINE FILTGT

PURPOSE:	To fill /TARGET/ common block for multiple target which is a city or area target, so it can be processed as a complex target by PROCCOMP.
ENTRY POINTS:	FILTGT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	DYNAMIC, TARGET
SUBROUTINES CALLED:	None
CALLED BY:	ALOCOUT

Method

FILTGT transfers data from common block /DYNAMIC/ to common block /TARGET/, as shown in figure 65.

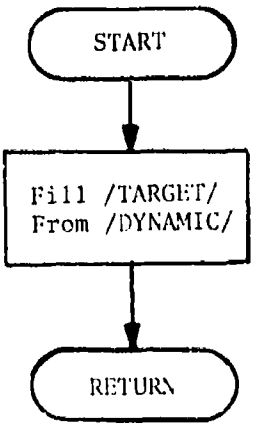


Fig. 65. Subroutine FILTGT

SUBROUTINE FINDMIN

PURPOSE: This subroutine uses a steepest descent method to determine a local minimum of a function of several variables. An initial estimate of the minimum position is input, together with various tolerances. FINDMIN uses two auxiliary routines, F2BMIN and GRADF, which define the function to be minimized and its gradient, respectively. DGZSEL uses FINDMIN to find the DGZs for complex targets.

ENTRY POINTS: FINDMIN

FORMAL PARAMETERS:

- XO - Initial guess at aim point offsets
- N - Length of XO vector
- IMAX - Maximum number of iterations for FINDMIN
- E1,E2 - Tolerances for the minimization
- X - Best aim point offsets as determined by FINDMIN
- F1 - Minimum value found for escaped target value
- IFLAG - Print control flag
  - -2 : DGZSEL computation value print is produced
  - >0 : FINDMIN debug prints will be produced

ENTRY POINTS: FINDMIN

COMMON BLOCKS: 11

SUBROUTINES CALLED: F2BMIN, GRADF, SEECALC

CALLED BY: DGZSEL

Method

Given a function $F(X_1, X_2)$, the gradients $G_1 = \frac{\partial F}{\partial X_1}$, and $G_2 = \frac{\partial F}{\partial X_2}$, and an initial guess (XO_1, XO_2) at the aim point offsets, FINDMIN finds the local

escaped target value F and its associated aim point offset coordinates (X_1, X_2). Each iteration consists of a function, F, and gradient, (G_1, G_2), evaluation followed by determination of the minimum function value along the line associated with the modified steepest descent direction. F is redetermined at each iteration and is defined in such a way that it converges after two iterations. FINDMIN uses two subroutines during its processing. The first, F2BMIN, defines the escaped target value function F in terms of aim point offsets X_1 and X_2. The second, GRADF, defines the gradient components G_1 and G_2. In addition, at the user's option, subroutine SEECALC is called after each iteration to print the results of the optimization.

Subroutine FINDMIN is illustrated in figure 66.

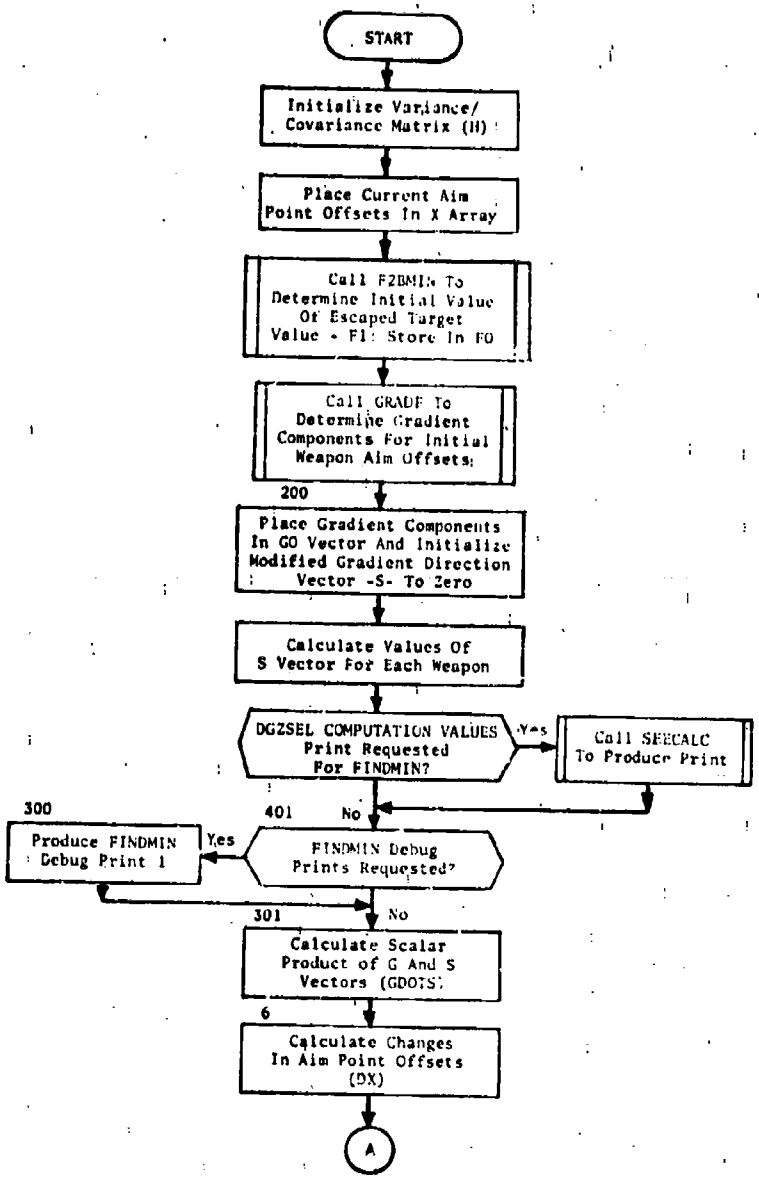


Fig. 66. Subroutine FINDMIN
(Sheet 1 of 4)

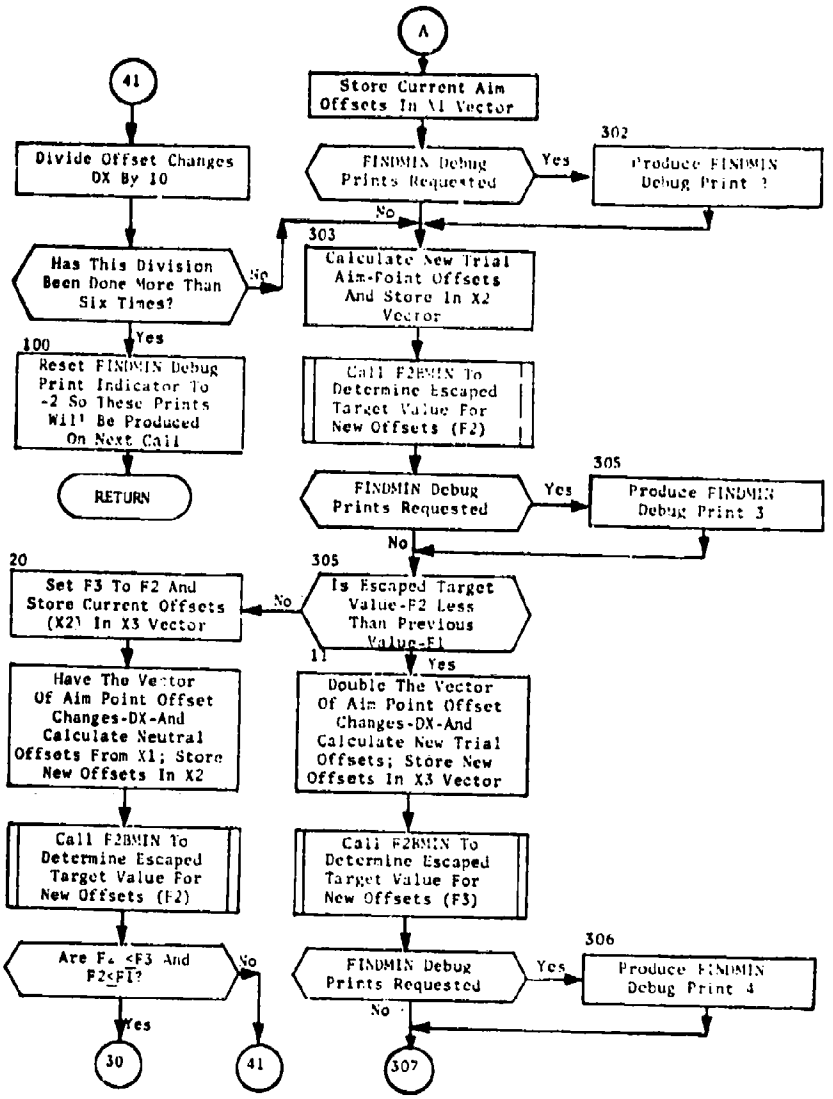


Fig. 06. (cont.)
(Sheet 2 of 4)

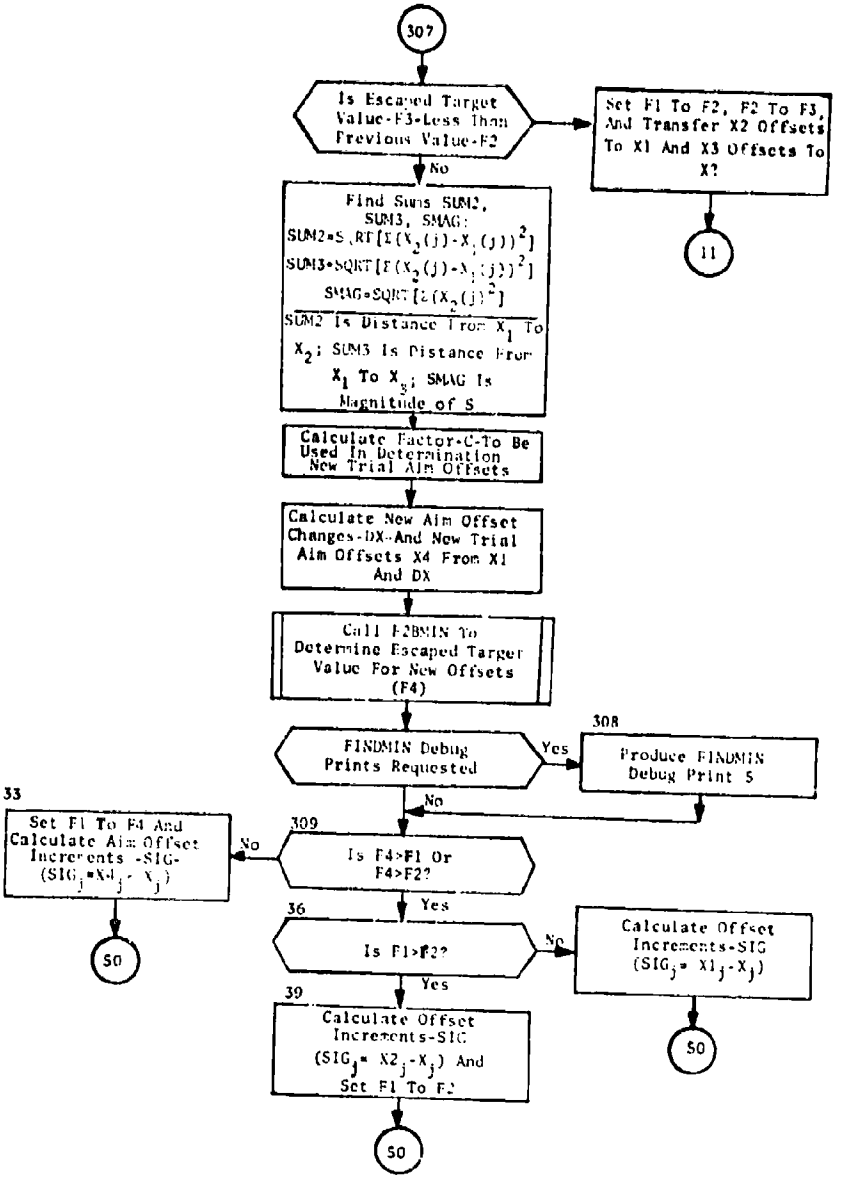


Fig. 66. (cont.)
(Sheet 3 of 4)

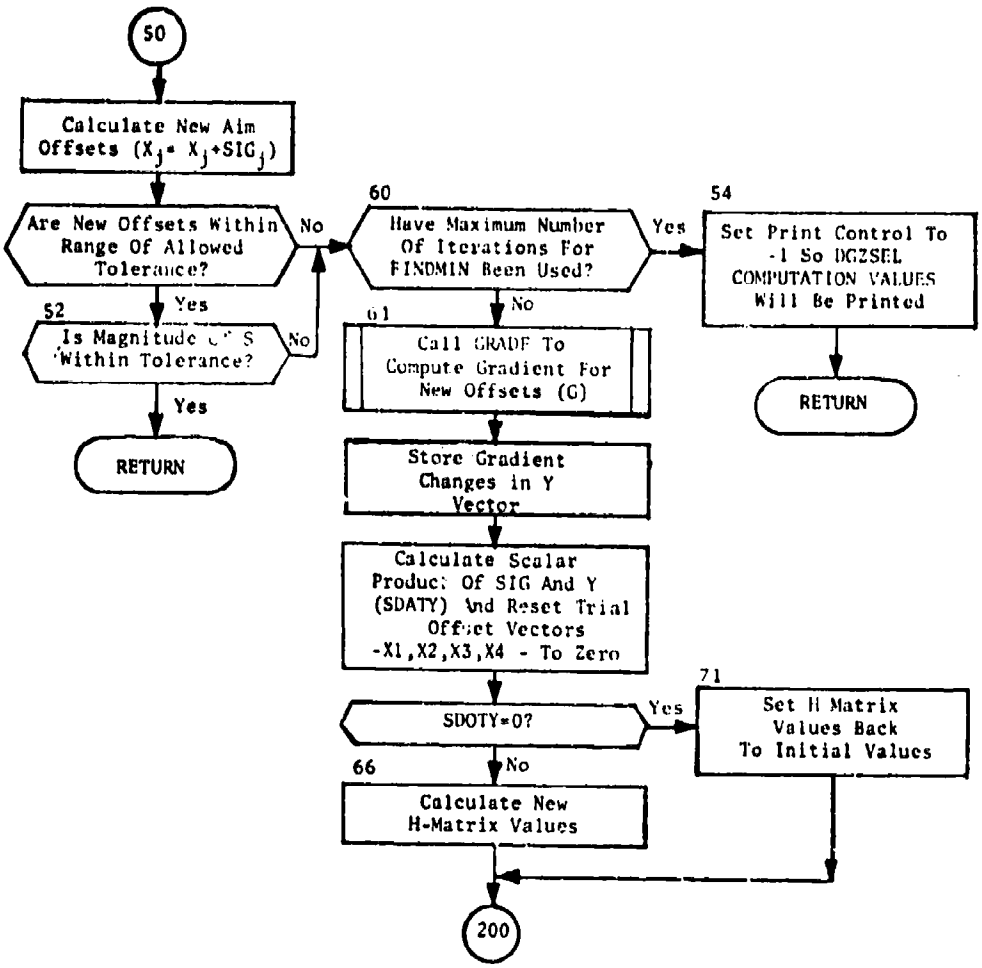


Fig. 66. (cont.)
(Sheet 4 of 4)

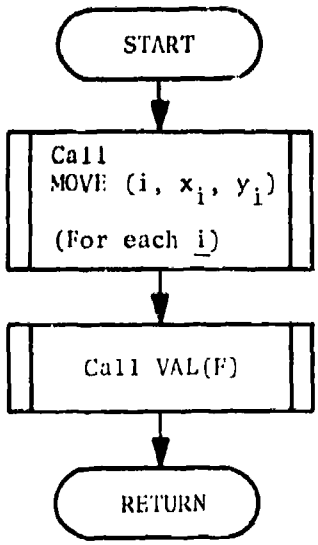
SUBROUTINE F2BMIN

PURPOSE:	F2BMIN defines the function which is to be minimized by FINDMIN. (FINDMIN minimizes survival probabilities of the target element and total escaping target value.)
ENTRY POINTS:	F2BMIN
FORMAL PARAMETERS:	XX - Vector containing offset coordinate for weapons F - Total escaping target value for this weapon configuration
COMMON BLOCKS:	1
SUBROUTINES CALLED:	MOVE, VAL
CALLED BY:	FINDMIN

Method

The offset coordinates for all weapons are input and a call on MOVE is made for each weapon to determine new survival probabilities. Then a call on VAL gives the new function value (total escaping target value) as well as the new effective values. The x, y coordinates of weapon I are given, respectively, by XX(2*I-1) and XX(2*I).

Subroutine F2BMIN is shown in figure 67.



(Determines Survival Probability for all Target Elements Relative to Each Weapon.)

(Determines Effective Values of Target Elements for this Weapon Configuration and Gives Total Escaping Target Value.)

(x_i,y_i) = Coordinates Assigned to Weapon i.

Fig. 67. Subroutine F2BMIN

SUBROUTINE GRADF

PURPOSE:	GRADF determines the components of the gradient associated with the function which is to be minimized by FINDMIN.
ENTRY POINTS:	GRADF
FORMAL PARAMETERS:	XX - Vector giving weapon offset coordinates G - Vector computed by GRADF giving gradient components for each weapon
COMMON BLOCKS:	1
SUBROUTINES CALLED:	VMARG
CALLED BY:	FINDMIN

Method

Two calls on VMARG for each weapon (one for each coordinate) are made to generate the gradient components.

Subroutine GRADF is illustrated in figure 68.

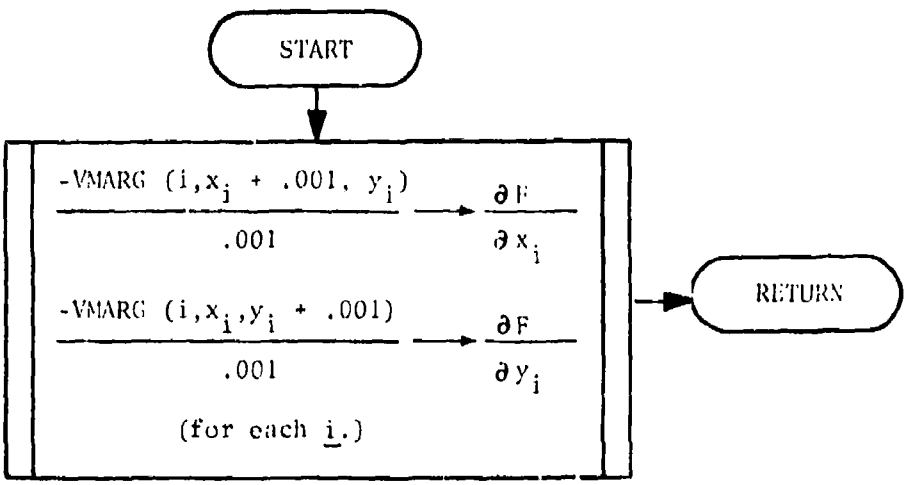


Fig. 68. Subroutine GRADF

FUNCTION IMAX

PURPOSE:	To determine the index of the element in the array ARRAY which has the largest (or smallest, in the case of the IMIN entry) value.
ENTRY POINTS:	IMAX, IMIN
FORMAL PARAMETERS:	ARRAY - The array whose maximum or minimum is to be found NARRAY - The length of ARRAY
COMMON BLOCKS:	None
SUBROUTINES CALLED:	None
CALLED BY:	COMPRESS

Method

The flowchart for IMAX is shown in figure 69. Depending on whether entry IMAX or IMIN is used, A is put equal to 1 or -1. Then the index, M, for the largest or smallest element in the array is initialized to 1. Finally, for each value of I from 2 through NARRAY, ARRAY(I) is compared with ARRAY(M); if the difference (ARRAY(I) - ARRAY(M)) when multiplied by A is greater than zero, M is set equal to I. Then, before control is returned to the calling subprogram, IMAX is set equal to M.

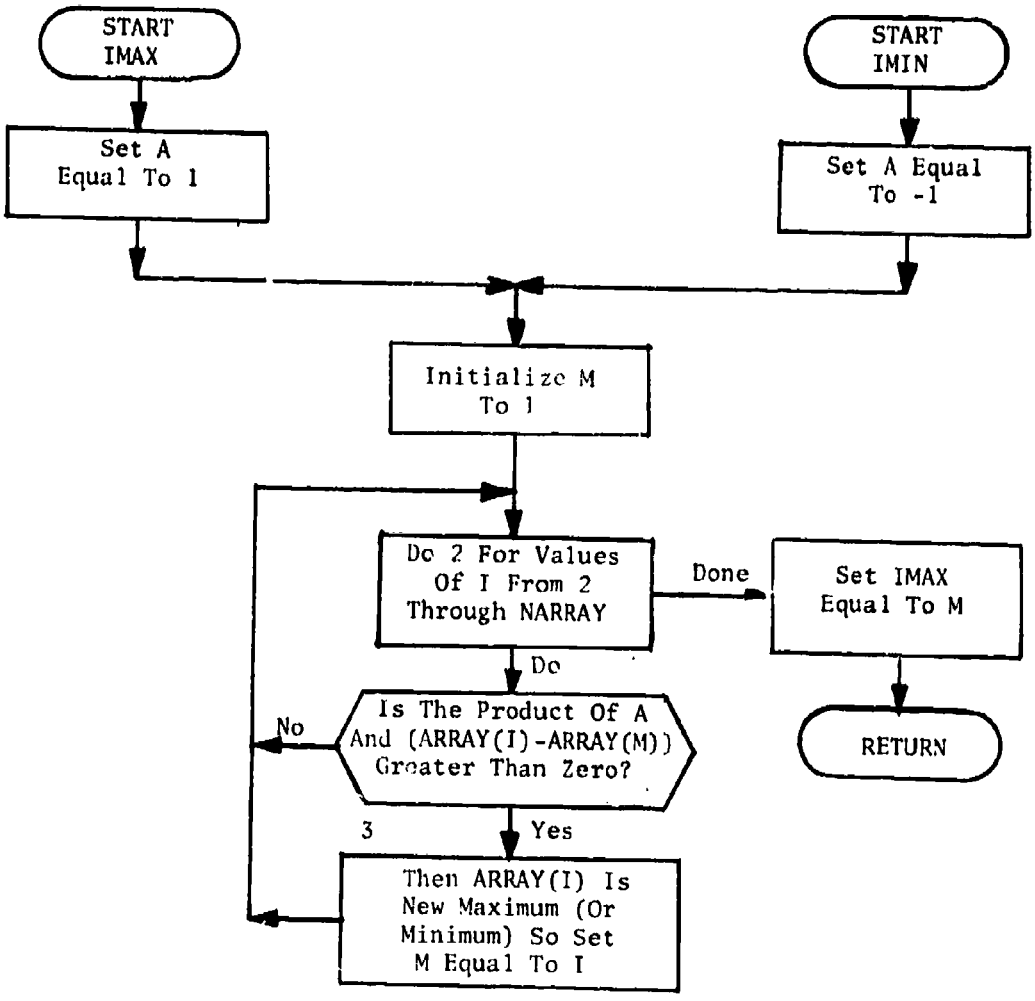


Fig. 69. Function IMAX

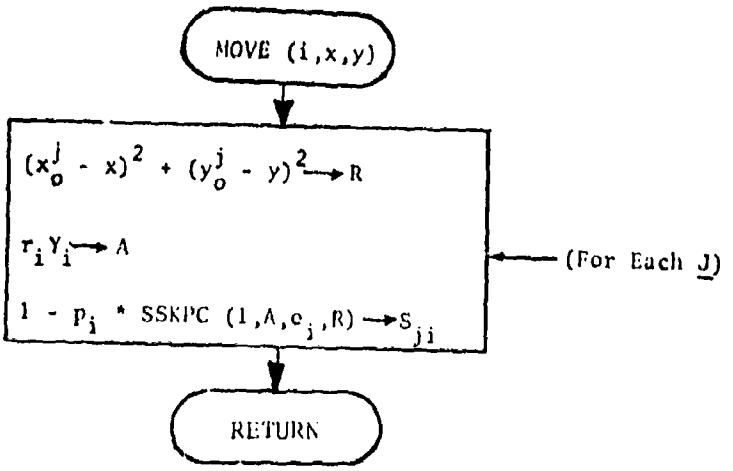
SUBROUTINE MOVE

PURPOSE:	Subroutine MOVE determines the survival probability, for all target elements, for a specific weapon moved to a given position.
ENTRY POINTS:	MOVE
FORMAL PARAMETERS:	IM - Index for weapon XM - X coordinate of weapon aim point offset YM - Y coordinate of weapon aim point offset
COMMON BLOCKS:	1
SUBROUTINES CALLED:	SSKPC
CALLED BY:	DGZSEL, F2BM1N

Method

For weapon IM with aim offset (XM, YM) the survival probabilities, S(J,IM), for each target element J, are redetermined using SSKPC.

Subroutine MOVE is illustrated in figure 70.



(x,y) = Offset coordinates for weapon I.

r_j = Lethal radius of target J.

(x_o^j, y_o^j) = Coordinates of target J.

Y_i = Scaled yield of weapon I.

p_i = Probability of delivery of weapon I.

e_i = Error in delivery of weapon I.

Fig. 70. Subroutine MOVE

SUBROUTINE PERTBLD

PURPOSE:	PERTBLD perturbs the weapon coordinates assigned by the laydown algorithm in such a manner as to assure a unique treatment by FINDMIN for each weapon.
ENTRY POINTS:	PERTBLD
FORMAL PARAMETERS:	None
COMMON BLOCKS:	1
SUBROUTINES CALLED:	ERGOT1
CALLED BY:	DGZSEL

Method

There is the possibility that, from some point on in time, all target element values become constant. In this case, all weapons input to FINDMIN with identical characteristics and later delivery times, which have been assigned to the same target element by the laydown procedure, would remain together. To eliminate this problem, subroutine PERTBLD is called just prior to calling FINDMIN.

Subroutine PERTBLD is shown in figure 71.

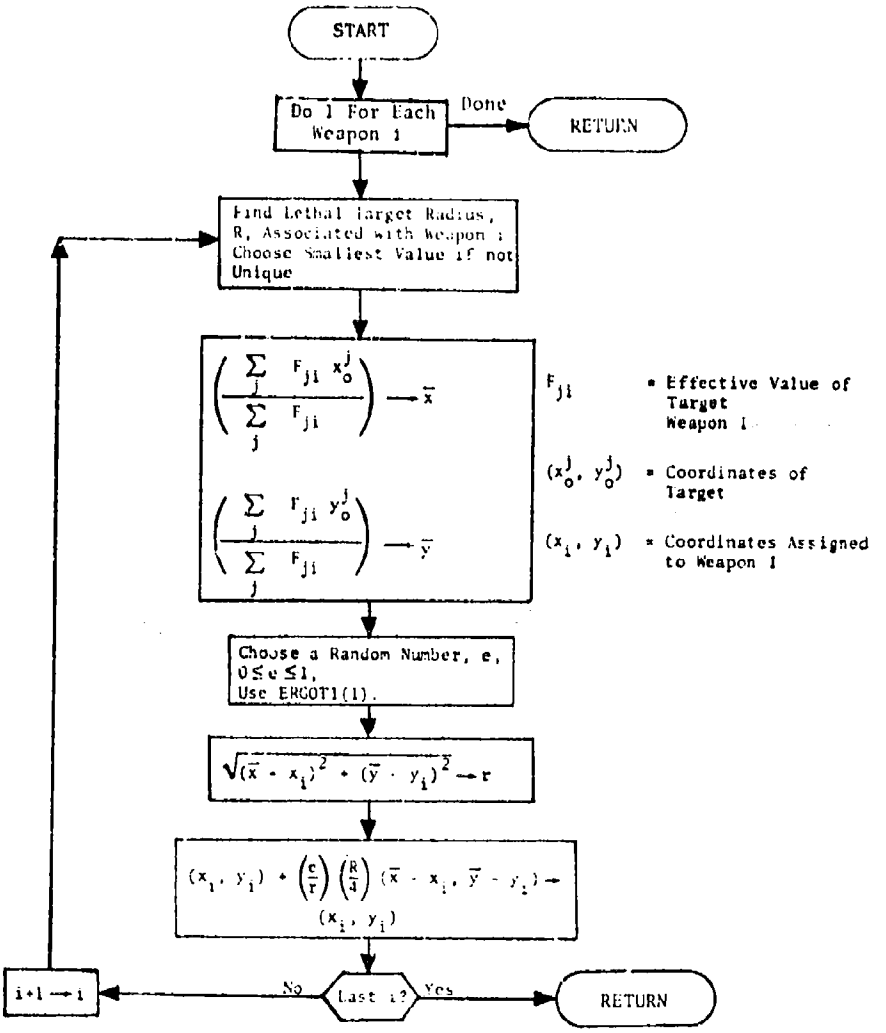


Fig. 71. Subroutine PERTBLD

SUBROUTINE PROCCOMP

PURPOSE:	To set up arrays in common block /1/ for the complex target so that subroutine DGZSEL can use the arrays during the selection of optimal aim point offsets for the weapons allocated to the target; and to output the aim point offset data by target element to subroutine PROCSIMP where it is written on the intermediate file.
ENTRY POINTS:	PROCCOMP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	STRK, FILES, MASTER, WPNREG, WTYPE, WGROUP, TARGET, 1, ITP, IFTPRNT, CITY, ISKIPDGZ
SUBROUTINES CALLED:	1, CITY, FILES, IFTPRNT, ISKIPDGZ, ITP, MASTER, STRK, TARGET, WGROUP, WPNREG
CALLED BY:	ALOCOUT

Method

A complex target (or target complex) is a combination of target elements sufficiently close that a weapon on any one of them will have some probability of killing other elements in the complex. Such target complexes must be targeted as a unit--not as individuals. Thus the allocator treats them as a unit, allocating weapons against their total value, using one set of coordinates. In order to maximize targeting efficiency against such a complex, one must select optimum aim points among the target elements. The aiming offsets are specified relative to the first target element only and are passed on in that form for the post-allocator and the Simulator.

When ALOCOUT encounters a complex target (ICOMP $\neq$ 0), PROCCOMP is called (figure 72). PROCCOMP is responsible for assembling the data on a complex target in a form that can be efficiently used for DGZ selection. Each target component of the complex generates a standardized "target element" in the working arrays used by DGZSEL (common /1/). Targets with more than one hardness component generate more than one such target element, and targets with a specified target radius will generate several elements spread over the area of the target to represent a value over the area. First PROCCOMP fills the weapon data arrays in common /1/, which is used by the DGZ selection (DGZSEL). Then, for each element of the complex target, it reads a data record /TARGET/ from the BASFILE and sets up the

target element value and location arrays for use by DGZSEL. For area targets, PROCCOMP generates target elements. If the number of target elements so generated exceeds the maximum program dimensions (50), COMPRESS is called to recombine target elements near each other having nearly the same lethal radius. In any case, for efficiency in DGZSEL, a call to COMPRESS is made just before calling DGZSEL. On return from DGZSEL, PROCCOMP converts the target aim point offsets to differences in latitude and longitude and calls PROCSIMP to write the records for the target onto the intermediate file.

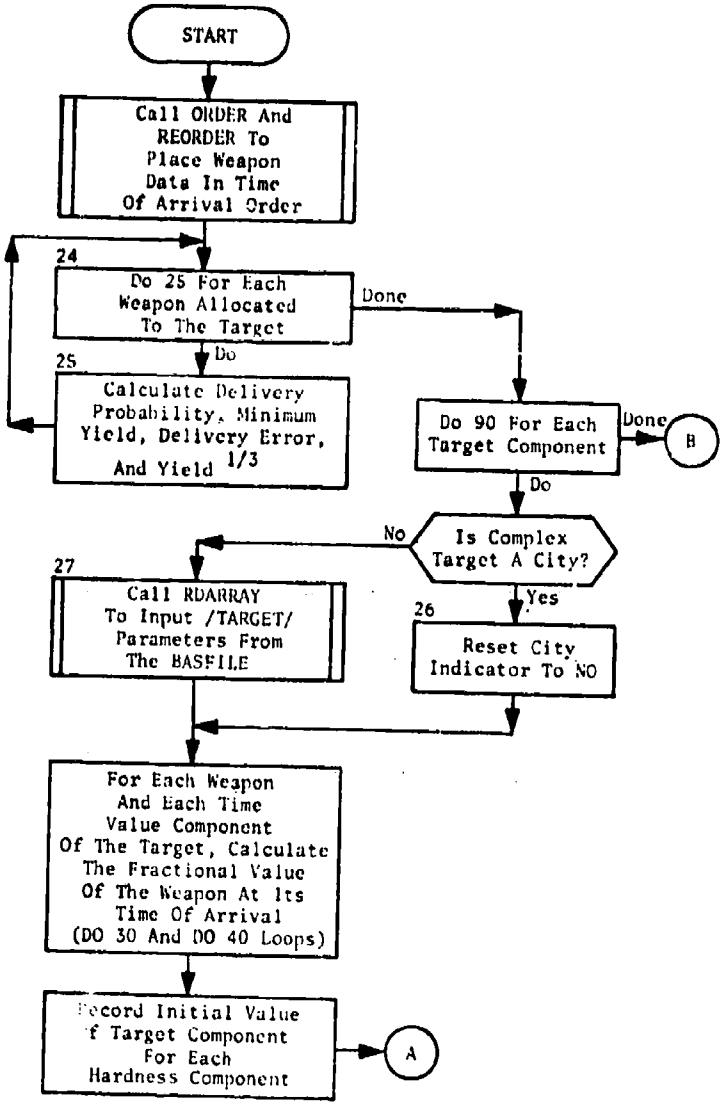


Fig. 72. Subroutine PROCCOMP
(Sheet 1 of 4)

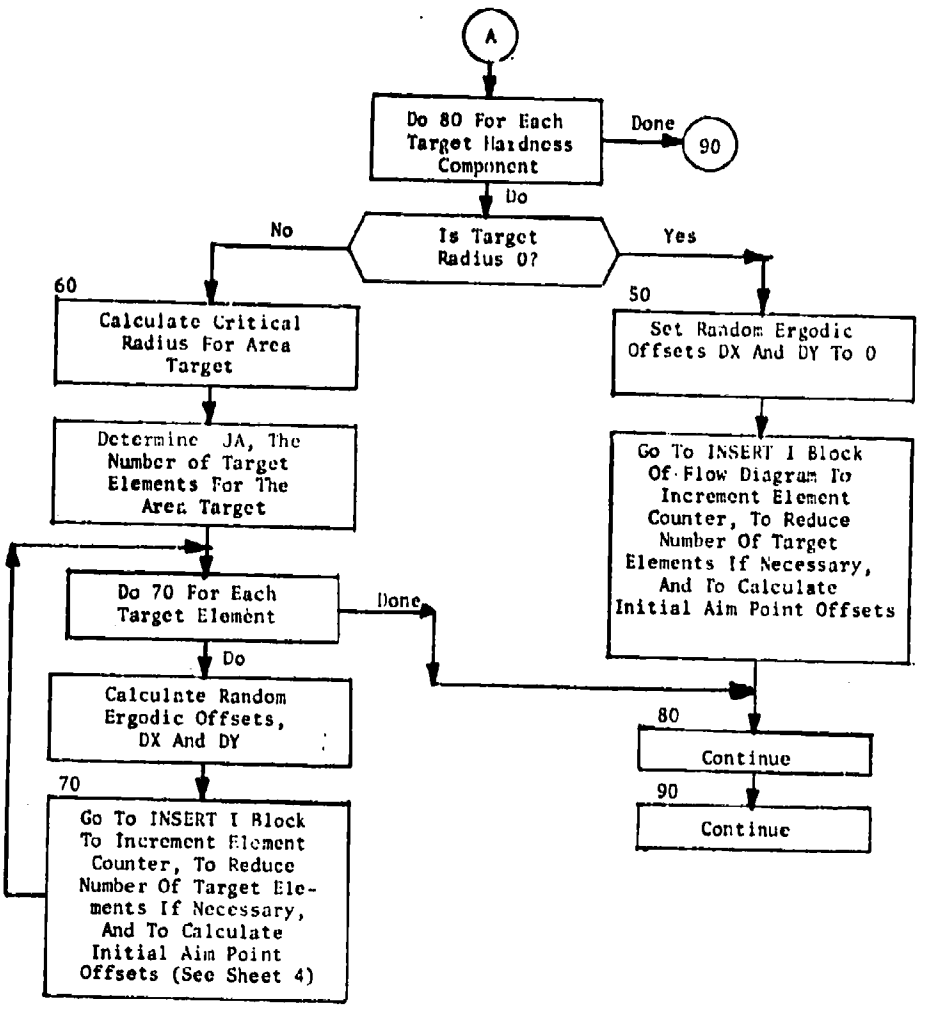


Fig. 72. (cont.)
(Sheet 2 of 4)

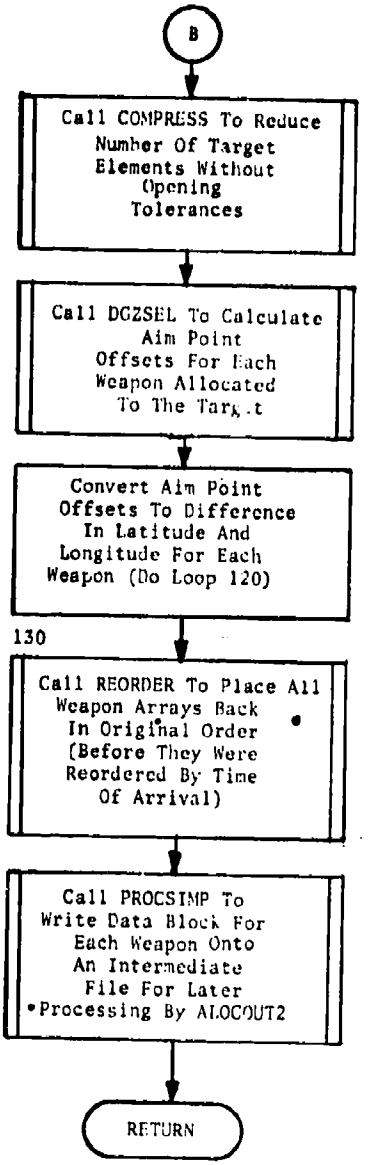


Fig. 72. (cont.)
(Sheet 3 of 4)

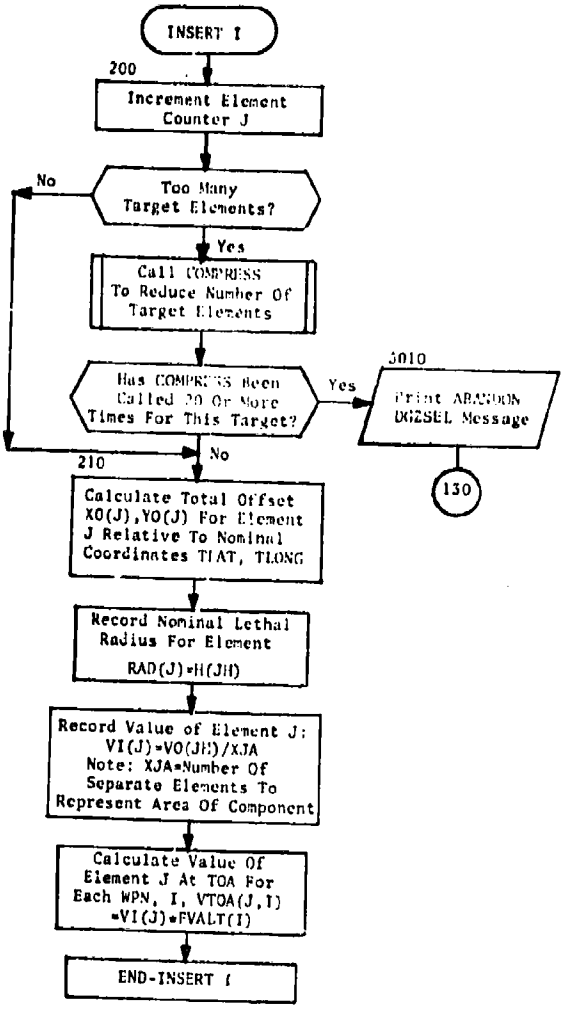


Fig. 72. (cont.)
(Sheet 4 of 4)

SUBROUTINE PROCMULT

PURPOSE:	To break up a "multiple target" as received from the allocator into individual targets.
ENTRY POINTS:	PROCMULT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	FILES, IFTPRNT, ITP, MULTTGT, STRK
SUBROUTINES CALLED:	RDARRAY, PROCSIMP
CALLED BY:	ALOCOUT

Method

A multiple target represents two or more identical targets whose geographic locations are close (and whose index numbers, as game objects, are consecutive).

Program ALOC (the allocator) may make only one assignment of weapons for all the elements of the multiple target where the allocation is identical for each target. However, the post-allocator (program POSTALOC) requires separate coordinates and weapon assignments for each target.

Hence, when ALOCOUT reads a "multiple target" data block from ALOCTAR, it calls PROCMULT (figure 73) to break up the target into individual (single element) targets. PROCMULT reads one record /MULTTGT/ for each of the target elements from the BASFILE to obtain the actual index number, name, coordinates, etc., of the target. (These multiple target records are on BASFILE in the same order as they occur on ALOCTAR so no problems with file positioning occur.) PROCMULT then calls PROCSIMP to write the strike data records for each individual target separately, so that from this point on in the data flow the individual targets of a multiple target can be treated as separate simple targets.

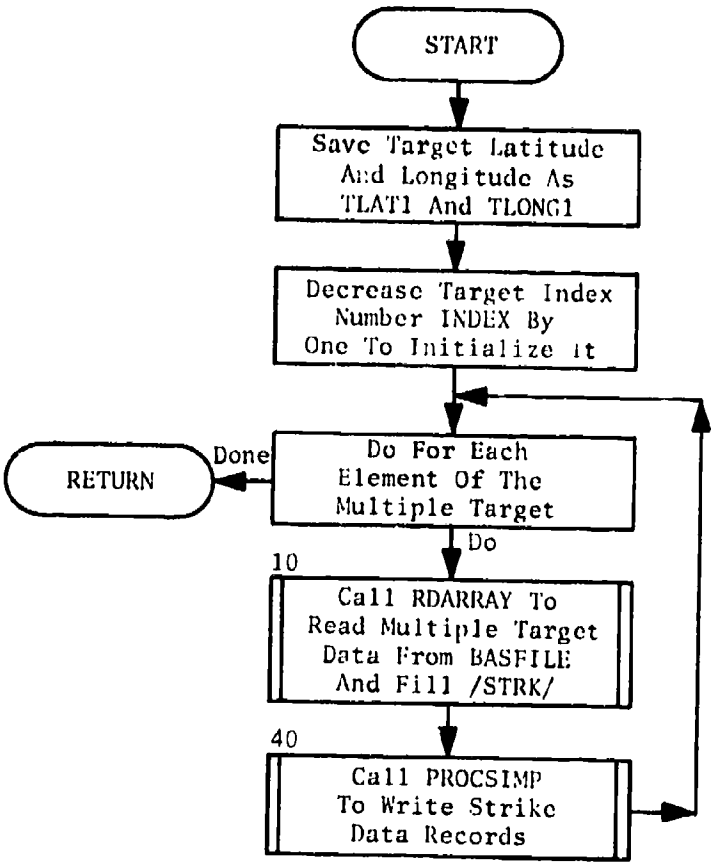


Fig. 73. Subroutine PROCMULT

SUBROUTINE PROCSIMP

PURPOSE	To write a 23-word data block onto an intermediate file ISCRATCH for each weapon allocated to the current target.
ENTRY POINTS:	PROCSIMP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ITP, MYIDENT, SCRATCH, STRK, STRKTGT
SUBROUTINES CALLED:	ABORT, WRARRAY
CALLED BY:	ALOCOUT, PROCCOMP, PROCMULT

Method

The main function of PROCSIMP (figure 74) is to write one data block for each weapon onto the intermediate file ISCRATCH which is later rearranged by ALOCOUT2 to produce the main output of ALOCOUT - namely, the file TMPALOC. PROCSIMP is called by ALOCOUT, PROCMULT, or PROCCOMP depending on the nature of the current target (i.e., depending on whether it is simple, multiple, or complex; multiple and complex targets require additional processing before PROCSIMP can be used).

If ALOCOUT encountered an end sentinel on the ALOCTAR file before this call to PROCSIMP, PROCSIMP writes an end-of-file record on ISCRATCH and returns. Otherwise, the variable NFIX is set equal to NNFIX, the number of weapons allocated to the target using the fixed assignement capability in program ALOC. PROCSIMP then transfers target data from the first LDATAT (data set to 16) words of /STRK/ to the first LDATAT words of /STRKTGT/. Next, the weapon allocation data are transferred from /STRK/ to /STRKTGT/ one weapon at a time (six words per weapon, one word from each of the /STRK/ arrays IGG, KOR, DLAT, DLONG, TOA, and RELVAL). For each set of weapon data, the penetration corridor index KORRX is checked to determine if the data apply to only one weapon (KORRX $\geq$ 0) or to two or more missile-delivered weapons (KORRX<0). A negative value indicates that KORRX missiles from the weapon group IGG(I) have been allocated to the target*.

*Technique used by ALOC in processing targets with terminal ABMs.

If KORRX is negative, PROCSIMP sets NN equal to the number of missiles allocated to the target and resets the corridor index KORRX to 0. The subroutine then evaluates the variable NFIX. If NFIX>0, the fixed weapon assignment flag IIFIX is set to 1. If NFIX is 0, the weapons are not fixed and IIFIX is set to 0. Subroutine WRARRAY is then called to write the /STRKTGT/ data block onto the ISCRATCH file once for each missile-delivered weapon.

If KORRX is equal to or greater than 0 (indicating a single missile- or bomber-delivered weapon), PROCSIMP merely sets the fixed-assignment flag IIFIX and writes the /STRKTGT/ data block once onto the ISCRATCH file for each weapon assigned to the target.

For each fixed-assigned weapon, the value of NFIX is decreased by one. If, in processing subsequent weapon data NFIX is found to be negative, an error message is printed, and the run is terminated by a call to subroutine ABORT.

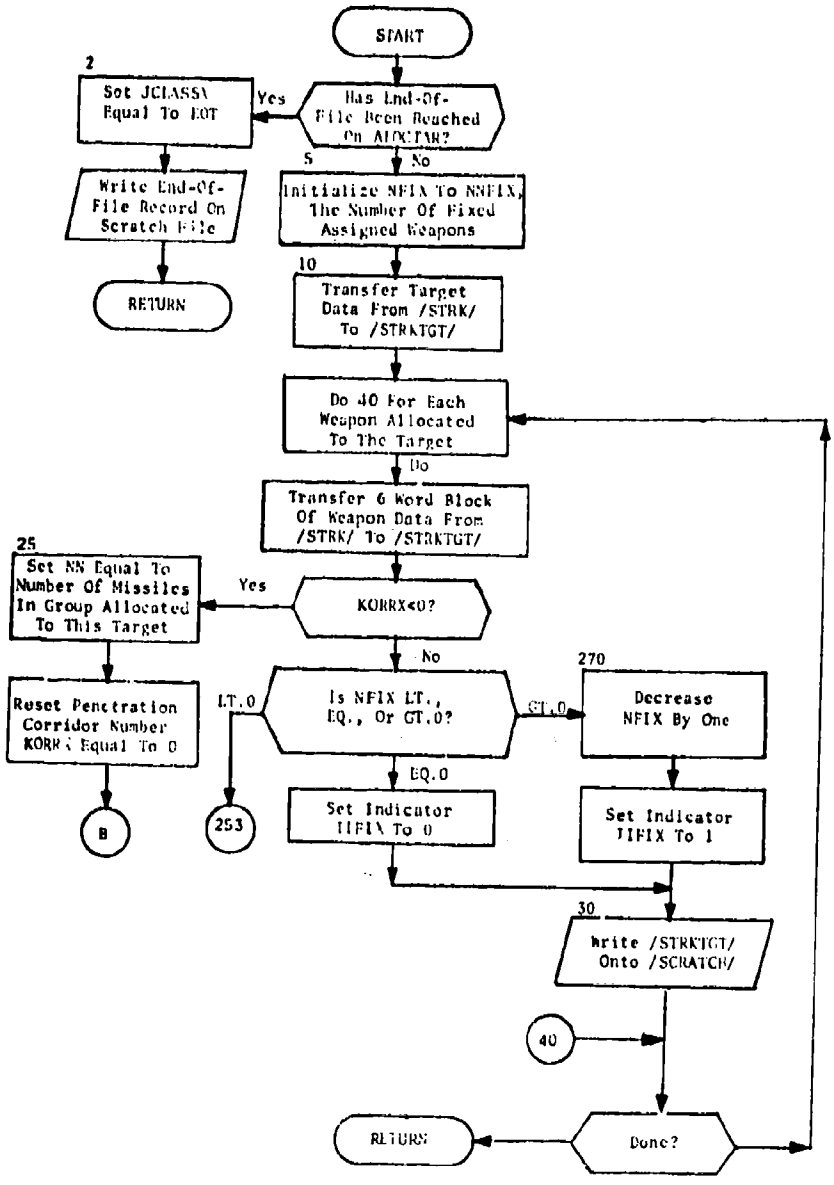


Fig. 74. Subroutine PROCSIMP
(Sheet 1 of 2)

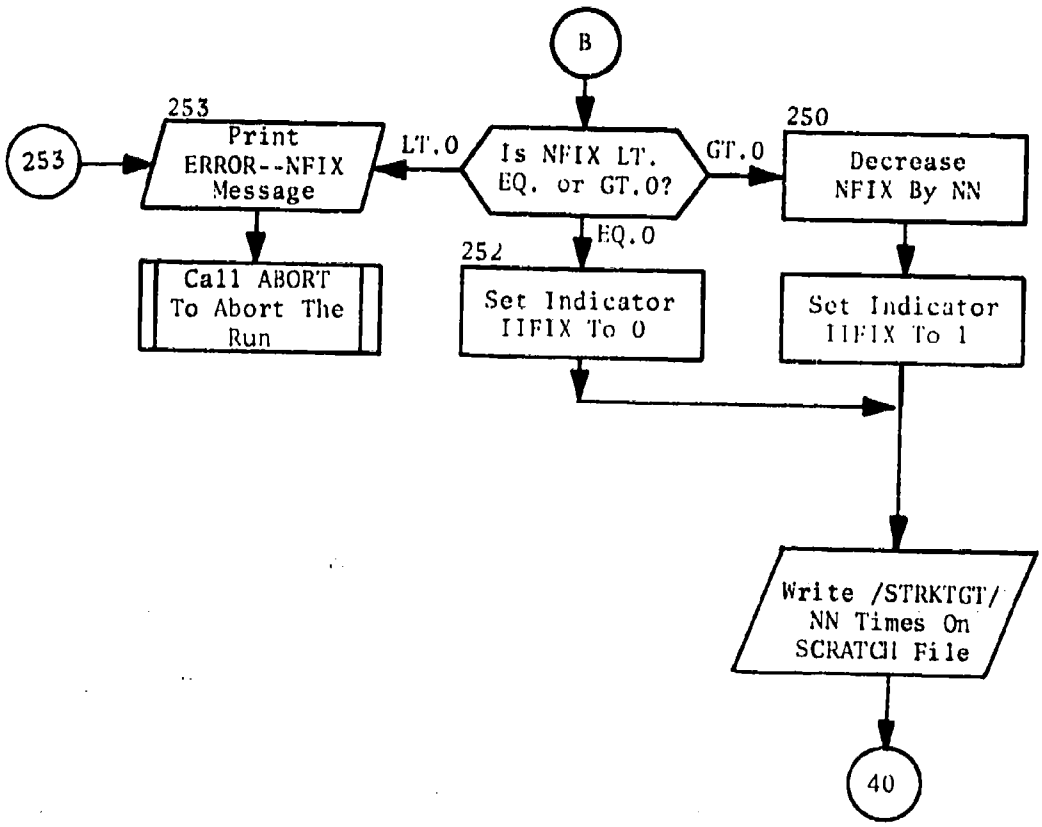


Fig. 74. (cont.)
(Sheet 2 of 2)

SUBROUTINE SEECALC

PURPOSE:	To print the computation values relevant to the selection of aim point offsets at various points within the DGZSEL subarea of program ALOCOUT.
ENTRY POINTS:	SEECALC
FORMAL PARAMETERS:	VESCTOT : Total escaping target value XX : Vector containing the aim point offset positions for the weapons
COMMON BLOCKS:	1
SUBROUTINES CALLED:	TIMEME
CALLED BY:	DGZSEL, FINDMIN

Method

When called by DGZSEL or FINDMIN subroutine SEECALC prints the title DGZSEL COMPUTATION VALUES and column headings. Then for each weapon allocated to the target, SEECALC prints the internal weapon number, the aim point offsets, and the survival of each target element relative to the weapon. At the end of the print for each target, the total escaped target value is printed.

Subroutine SEECALC is illustrated in figure 75.

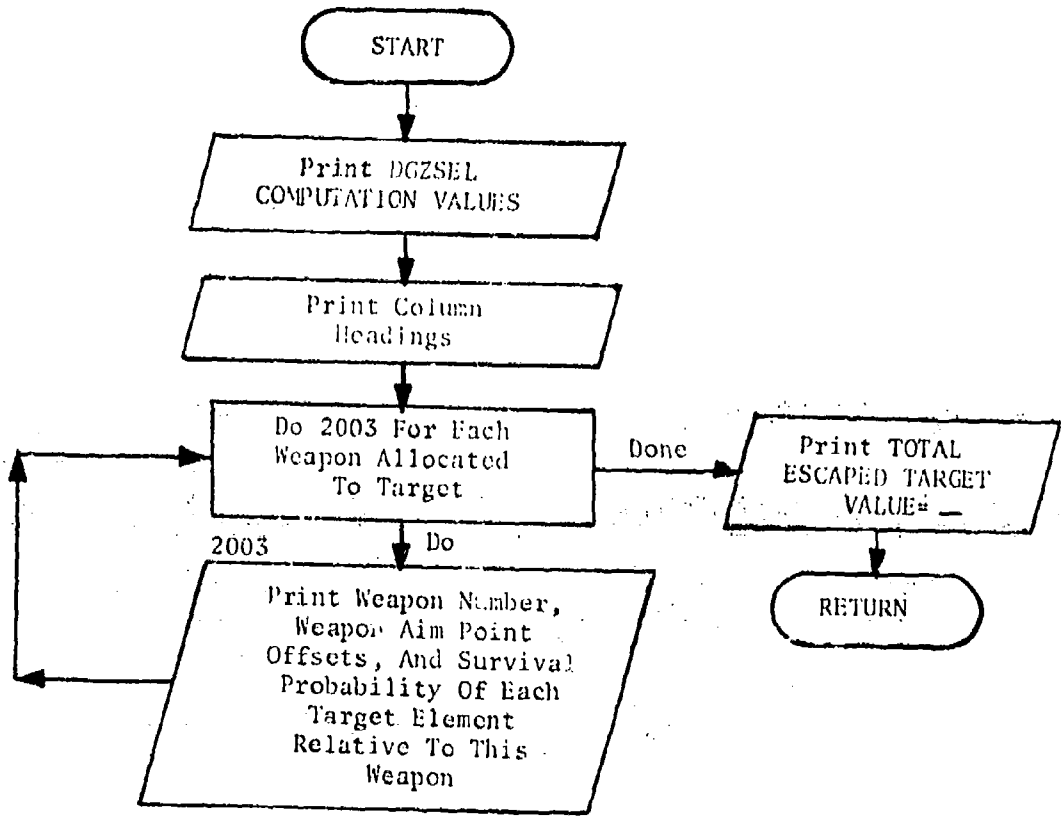


Fig. 75. Subroutine SUBCALC

SUBROUTINE SEEINPUT

PURPOSE:	SEEINPUT prints part of the input to DGZSEL from PROCCOMP. For each complex target, the print by SEEINPUT will precede the first SEECALC print. Its use is largely for debugging and to give the user some feel for the computation.
ENTRY POINTS:	SEEINPUT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	1
SUBROUTINES CALLED:	TIMEME
CALLED BY:	DGZSEL

Method

When called by DGZSEL, subroutine SEEINPUT first prints the title TGT DATA INPUT TO DGZSEL and the column headings for the print. Then it prints the target coordinates, the initial target value, and the target lethal radius by target element. This is followed by a print of the value of the target after the arrival of the weapon by weapon and target element. Finally, SEEINPUT prints the probability of delivery, the delivery error, and the scaled yield for each weapon allocated to the target.

Subroutine SEEINPUT is illustrated in figure 76.

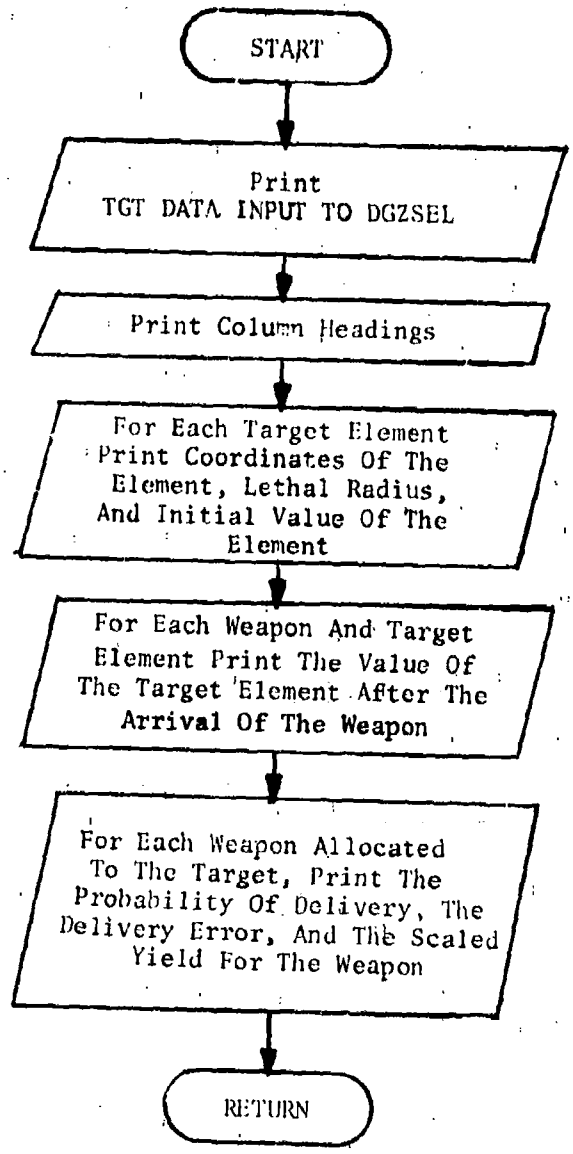


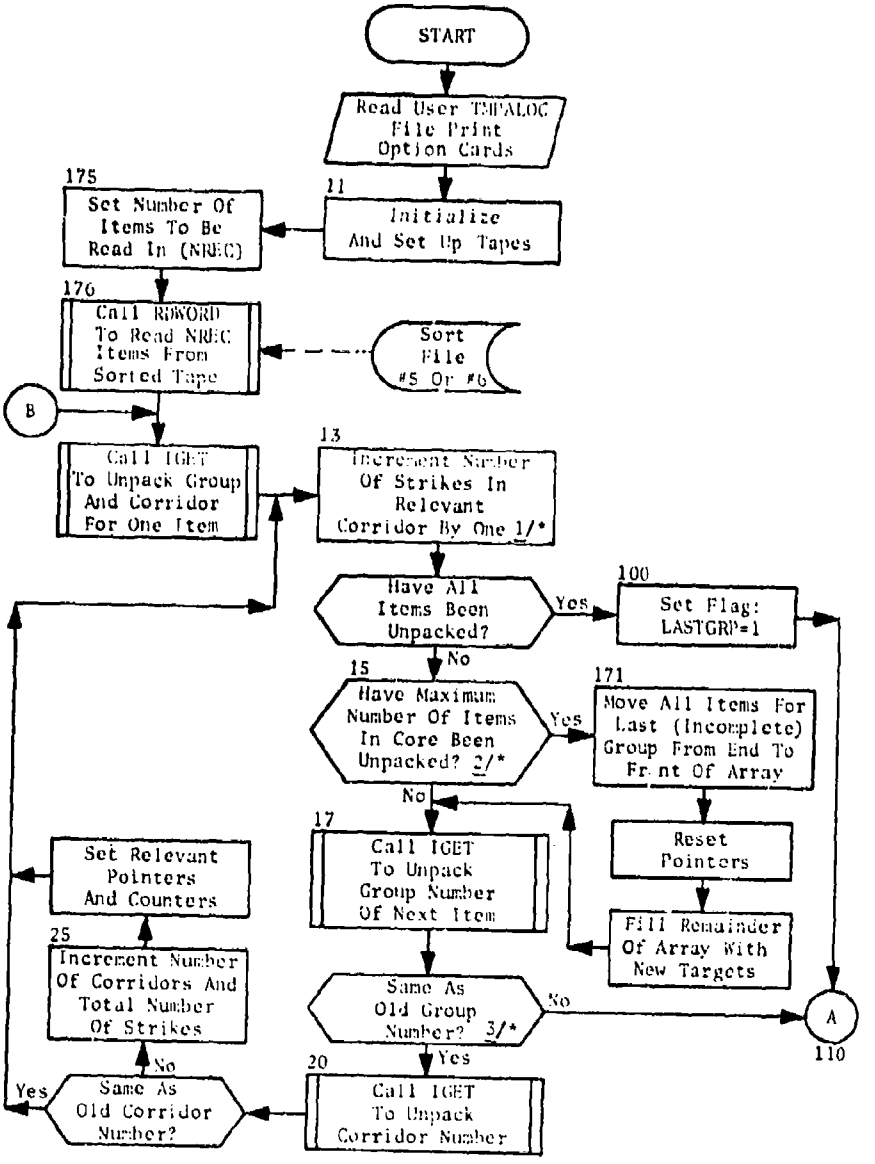
Fig. 76. Subroutine SEEINPUT

SUBROUTINE STRKOUT

PURPOSE:	To unpack and summarize by weapon group the number of strikes through each corridor and write the summary record STRKSUM.
ENTRY POINTS:	STRKOUT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	1, 11, ALOC, DATA, FILABEL, FILES, IETPRNT, ITP, KEYS, RAID, STRKSUM, TWORD
SUBROUTINES CALLED:	SETREAD, IGET, TERMTAPE, ORDER REORDER, WRARRAY, TIMEME, WRRDSTRK, RDWORD
CALLED BY:	ALOCOUT2

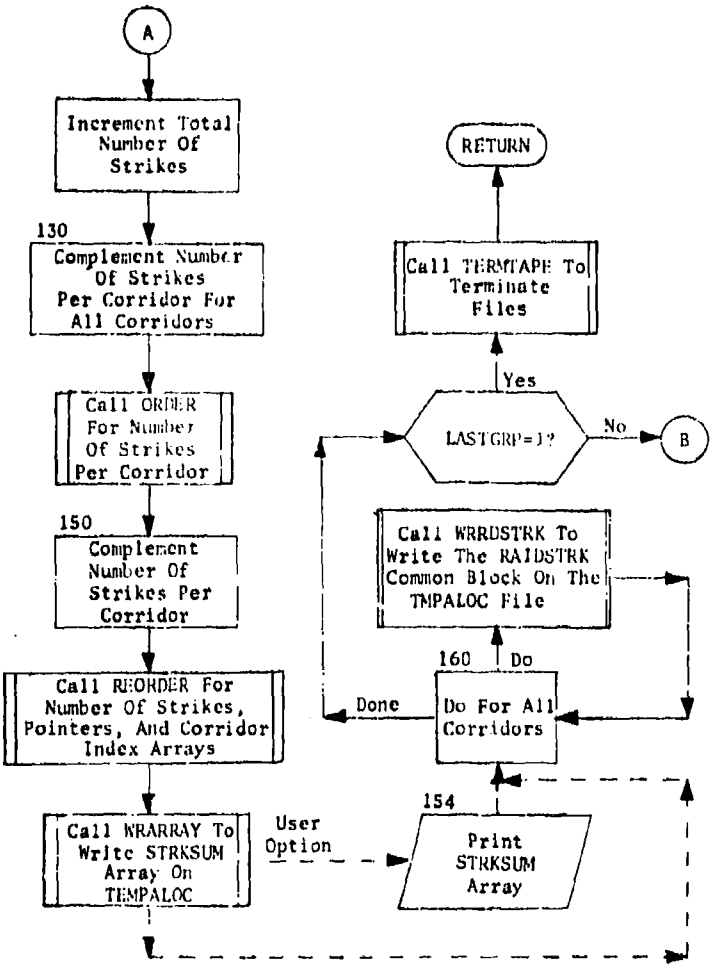
Method

STRKOUT (figure 77) reads the sorted data back into core in several batches as storage permits. For each weapon group, it makes a tally of the number of strikes assigned through each penetration corridor. When all the data for a single weapon group have been tallied, STRKOUT sorts the penetration corridors in descending order of the number of strikes and writes a record, common /STRKSUM/, on the final output file TMPALOC. This record gives the weapon group number, the total number of strikes for that group, the number of penetration corridors used, and the number of strikes through each corridor. It then calls subroutine WRRDSTRK for each corridor to write a record of the target data for all strikes through that corridor, common/RAIDSTRK/ (which is part of common /2/ in subroutine WRRDSTRK).



*See notes on sheet 2.

Fig. 77. Subroutine STRKOUT
(Sheet 1 of 2)



Notes:

1. ICORIN(NC) is an array of the corridor index numbers. IPT(NC) contains the I-index in the IWD array of the first item for the corridor. Both arrays are later reordered in the sequence determined by ordering NSTKK.

2. When all items in the IWD array have been processed, those for the current group (i.e., those that have not yet been written on the output tape) are moved to the beginning of the array, and the rest of the array is filled from the sort tape.

3. As the items are processed and when a new weapon group number is encountered, the tallying, sorting, and tape writing for the computed group is done. Unless ILASTGRP=1, the program then returns to process the remaining items.

Fig. 77. (cont.)
(Sheet 2 of 2)

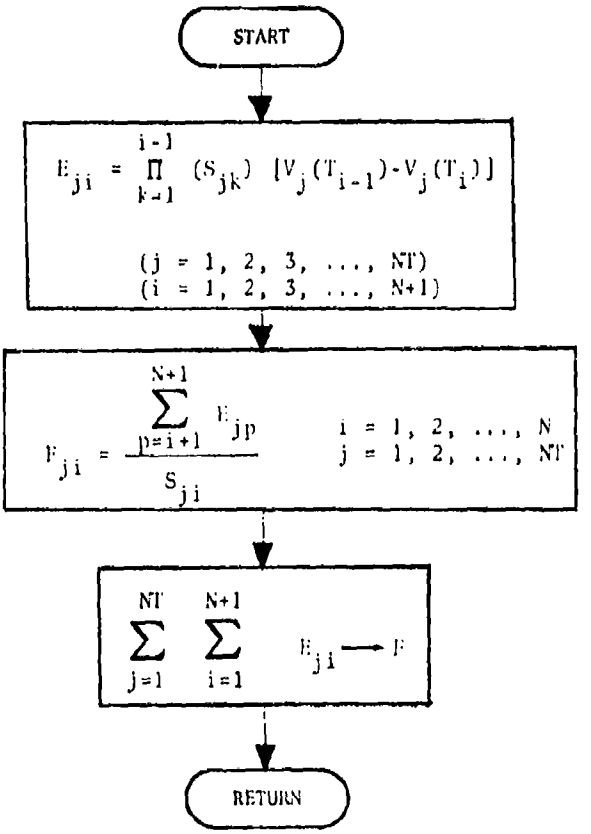
SUBROUTINE VAL

PURPOSE:	VAL determines the target value which has escaped for a given weapon configuration and also determines the effective value, F_{ji}, for each target element as seen by each weapon.
ENTRY POINTS:	VAL
FORMAL PARAMETERS:	VESCTOT
COMMON BLOCKS:	1
SUBROUTINES CALLED:	None
CALLED BY:	DGZSEL, F2BMIN

Method

This computation uses the effective values, VEFF(J,I), the survival probabilities, S(J,I), and the time dependent target values.

Subroutine VAL is illustrated in figure 78.



E_{ji} - Value of target element j after arrival of all weapons

$V_j(T_i)$ - Value of target element j immediately after arrival of weapons 1 through i

S_{jk} - Survival probability of target element j relative to weapon k

F_{ji} - The effective value of target element j as seen by weapon i

F = Total escaping target value=VESCTOT

Fig. 78. Subroutine VAL

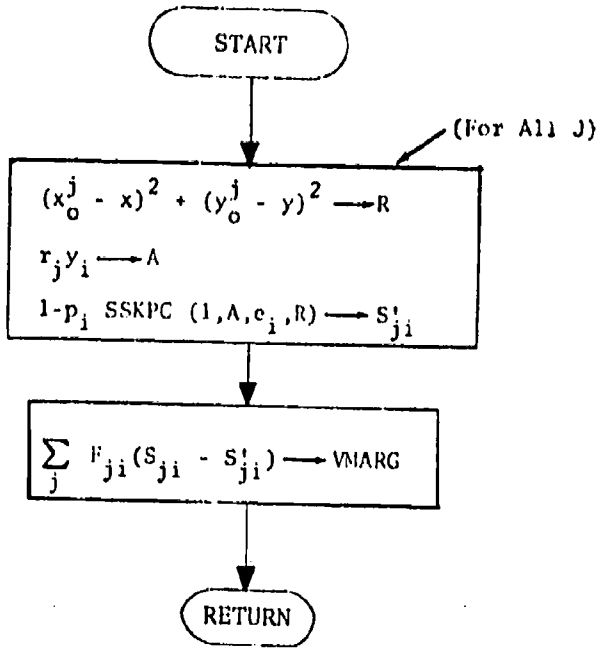
FUNCTION VMARG

PURPOSE:	Given a particular weapon configuration, function VMARG determines the marginal value of moving a specific weapon to a new position.
ENTRY POINTS:	VMARG
FORMAL PARAMETERS:	IT - Index weapon XT - X coordinate of weapon aim point offset YT - Y coordinate of weapon aim point offset
COMMON BLOCKS:	1
SUBROUTINES CALLED:	SSKPC
CALLED BY:	DGZSEL, GRADF

Method

A modified set of survival probabilities for all target elements for this weapon is used to determine the marginal value.

Function VMARG is shown in figure 79.



(x,y) = Position of Weapon I

(x_o^j, y_o^j) = Coordinates of Target Element J

r_j = Lethal Radius, Target J

y_i = Scaled Yield, Weapon I

p_i = Probability of Delivery, Weapon I

e_i = Error in Delivery, Weapon I

S_{ji} = Survival Probability of Target J Relative to Weapon I

S'_{ji} = Survival Probability of Target J Relative to Weapon I when it is Assigned to Position (x,y)

Fig. 79. Function VMARG

SUBROUTINE WRRDSTRK

PURPOSE:	To write the /RAIDSTRK/ common block on the TMPALOC file for a given corridor and weapon group.
ENTRY POINTS:	WRRDSTRK
FORMAL PARAMETERS:	None
COMMON BLOCKS:	1, 2, DATA, FILES, IFTPRNT, ITP, KEYS, RAID
SUBROUTINES CALLED:	IGET, LREORDER, ORDER, REORDER, TIMEME, WRARRAY
CALLED BY:	STRKOUT

Method

Subroutine WRRDSTRK (figure 80) unpacks the strike and target data and writes a variable length record in the format of common /RAIDSTRK/ on the TMPALOC file. For missile records, it sorts the target data in order of decreasing relative values (RVALM array) before writing it on the file. It can optionally print the data, as indicated on the user's input card.

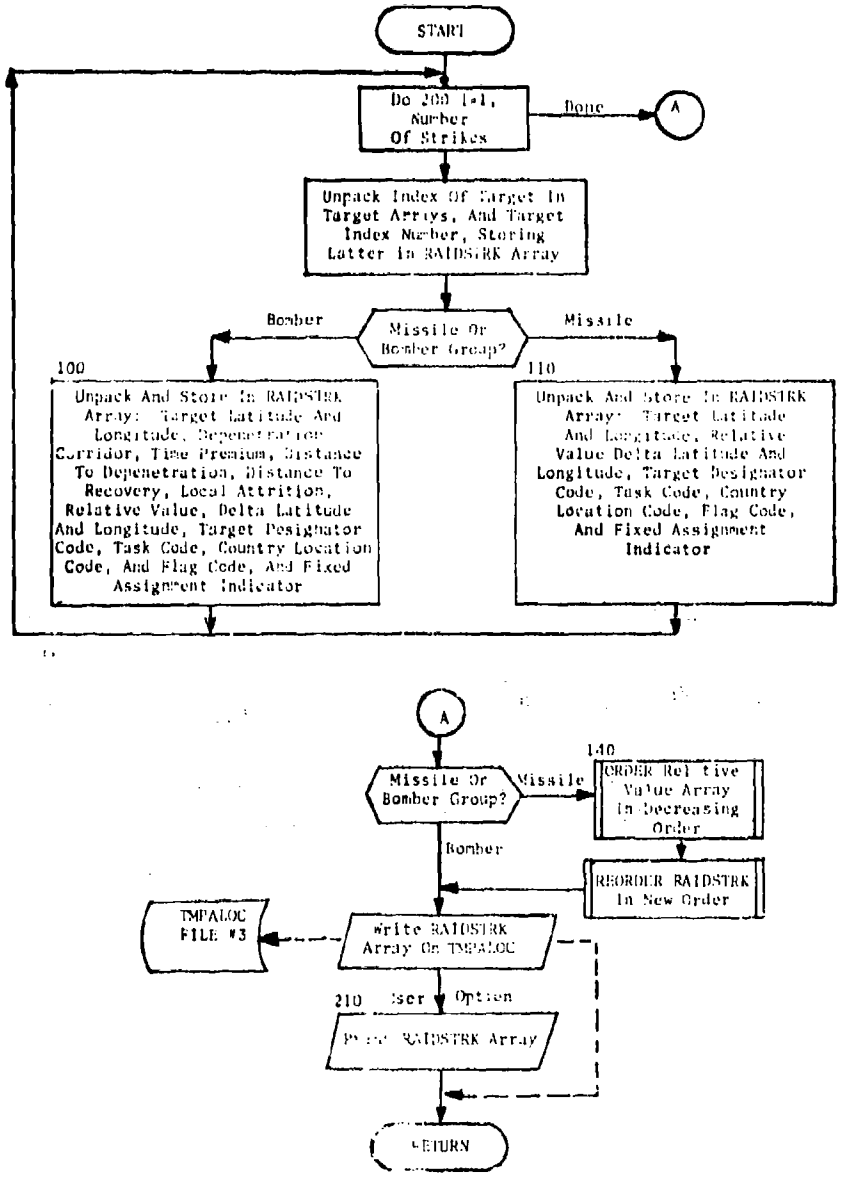


Fig. 80. Subroutine WRRDSTRK

DISTRIBUTION

Addressee	Copies
NMCSSC Codes	
B121	3
B122 (stock)	6
B200	1
B210	2
B220	28
B230	1
B600	1
DCA Codes	
260 (original document only, no subsequent changes)	1
920	1
950	1
OJCS	
Studies, Analysis and Gaming Agency, ATTN: SFD, Room 1D957, Pentagon, Washington, D.C. 20301	5
Commander-in-Chief, North American Air Defense Command ATTN: NPPG, Ent Air Force Base, Colorado 80912	2
Commander, U.S. Air Force Weapon Laboratory (AFSC) ATTN: AWL, Kirtland Air Force Base, New Mexico 87117	2
Director, Strategic Target Planning Offutt Air Force Base, Nebraska 68113	2
Chief of Naval Operations, ATTN: OP963G Room 5E531, Pentagon, Washington, D.C. 20350	2
Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314	12
	70

DDC

DEFENSE COMMUNICATIONS AGENCY
NATIONAL MILITARY COMMAND SYSTEM
SUPPORT CENTER
WASHINGTON, D. C. 20301

12

IN REPLY
REFER TO B221

7 September 1972

AD-742785

TO: DISTRIBUTION

SUBJECT: Change 1 to Programming Specifications Manual CSM PSM 9A-67, Volume II, Plan Generation Subsystem, Part A

1. This set of change pages reflects several computer programming changes to the Quick-Reacting General War Gaming System (QUICK) operational at the National Military Command System Support Center (NMCSSC). The changes were necessary in order to reduce the internal storage requirements of program PLANSET, of QUICK, in conjunction with a change to the Operating System of the NMCSSC CDC 3800 computer. These programming changes do not enhance the production capabilities of QUICK. Consequently, they should not be made unless a reduction in the PLANSET storage requirements is desired.

2. A list of Effective Pages to verify the accuracy of the QUICK Manual is enclosed. This list should be inserted before the title page and an appropriate entry made in the Record of Changes for the manual.

FOR THE COMMANDER:

Tony Lima, LTC
for J. DOUGLAS POTTER
Chief, Military Personnel and
Administrative Services Office

27 Enclosures
Change 1 pages

EFFECTIVE PAGES - 4 August 1972

This list is used to verify the accuracy of Programming Specifications Manual CSM PSM 9A-67, Volume II, Plan Generation Subsystem, Part A after change 1 pages have been inserted. Original pages are indicated by the letter O, change 1 by the numeral 1.

Page No.	Change No.
Title Page, Part A	O
ii-xi, Part A	O
1-8	O
9	1
10-11	O
12	1
13-45	O
46-56	1
56.1-56.2	1
57-59	1
60-78	O
79-80	1
81-450	O
451	1
452-452A	O
Title Page, Part B	O
ii-xv, Part B	O
453-1118A	O
Title Page, Part C	O
ii-vi, Part C	O
1119-1414	O
Title Page, Part D	O
ii-vi, Part D	O
1415-1852	O
Title Page, Part E	O
ii-vii, Part E	O
1853-2346A	O
Title Page, Part F	O
ii-vii, Part F	O
2347-2803	O

All programs of the Plan Generation subsystem use the filehandler in conjunction with input/output operations. The filehandler subroutines and their functions are summarized below. A detailed description of the QUICK filehandler is contained in chapter 2 of the Programming Specifications Manual, Volume 1.

SUBROUTINE	FUNCTION
ALOCDIR	Initializes disk file directory
INITAPE	Initializes filehandler
DEACTIV	Removes file name from active list
SETREAD	Prepares file for reading
RDWORD	Transfers one word from file input buffer to common /TWORD/
RDARRAY	Transfers block of words from file input buffer to user-specified core storage area
SETWRITE	Prepares file for writing
WRWORD	Transfers one word from common /TWORD/ to file output buffer
WRARRAY	Transfers block of words from user-specified core storage area to file output buffer
TERMTAPE	Terminates files after reading or writing and releases buffer area for use by other files

COMPUTER STORAGE REQUIREMENTS

The NMCSSC CDC 3800 computer provides a maximum of 65,534 words of core storage. Excluding the requirements of the operating system, the core storage requirements of the programs of the Plan Generator are as follows.

PLANSET	51,900*
PREPALOC	50,400
ALOC	53,500
ALOCOUT1	27,300
ALOCOUT2	50,000
FOOTPRNT	48,200
POSTALOC	50,400
PLNTPLAN	45,700
EVALALOC	44,800
INTRFACE	18,500
TABLE	13.500

* Decimal words.

CHAPTER 2
PROGRAM PLANSET

PURPOSE

PLANSET prepares the data files required by the Plan Generator to develop a plan for one side. It forms weapon groups, prepares the target list, computes and normalizes the class value factors, calculates the representative attributes for complex targets, and creates the WINFILE (weapon input file) and TINFILE (target input file) required by program PREPALOC. Note that program PLANSET must be processed by program DECLARES before being executed.

INPUT FILES

The principal input to PLANSET consists of the indexed data base, INDEXDB OR INMODDB, generated by program INDEXER or program BASEMOD of the Data Input subsystem. Several user-option data cards are also accepted, which specify:

1. Vulnerability data to be used for blast damage calculations
2. The command and control reliability factor for each region in the plan
3. A request that missile retargeting be used for all missiles with reprogramming capability
4. A range multiplier RANGEMOD to be used when determining whether a weapon is sufficiently within range of a weapon group to be added as a group member
5. The SIDE for which a plan is to be generated
6. Names of each attacking weapon type
7. The maximum absolute difference in PBI probability MAXDPL that is allowed between the first and last weapons of a weapon group
8. The class name and value of an exemplar target for each class in the current plan

9. The values of the attribute TASK to be given priority when assigning the lead target of a complex

10. The alphabetic portion of the values of the attribute DESIC to be given priority when assigning the lead target of a complex

11. Print options allowing a print of the target list weapon group list, and/or complex target list.

OUTPUT FILES

Program PLANSET prepares the TINFILE (target input file) and WINFILE (weapon input file) to be used in program PREPALOC. The TINFILE (see table 2) contains a 29-word block of descriptive information for each target to be considered in the current plan. The targets are placed on the TINFILE in a random order which facilitates evaluation of the allocation process used in program ALOC. The target input file (TINFILE) includes three types of targets:

1. Simple target: one target element

2. Complex target: several target elements either exactly collocated or within the lethal radius of a single weapon (a one-megaton weapon is considered) so that they must be treated as a single target complex

3. Multiple targets: actually several independent identical targets such as separate missile silos in a Minuteman squadron that are close together (relative to the range of the weapon), but far enough apart that each target element must be treated as an independent aim point.

Each complex target is represented on the TINFILE in an aggregated form representing the total value of the complex as required by ALOC. This aggregated representation on TINFILE is paralleled by auxiliary detailed target data on WINFILE which includes a specific representation of each target element as a separate simple target. Similarly each multiple target is represented on TINFILE by a single representative target (of the appropriate multiplicity) as required by ALOC. This representative target is also paralleled by a list of specific coordinates for each target element in the auxiliary target data on WINFILE.

Table 2. TINFILE Format
(Sheet 1 of 2)

BLOCK TYPE	MAXIMUM LENGTH	VARIABLE OR ARRAY	DESCRIPTION
Header	5	TAPETYPE	Run or data base identification number
		DATE	Date of run initiation
		IDENTNO	Run identification number
		ISIDE	Side
		NTGTS	Number of targets in data base
Target Block 2 through (NTGTS+1)	29	TGTNAME	Target name
		INDEXNO	Target index number
		DESIG	Target designator code
		TASK	Target task code
		CNTRLOC	Target country location code
		FLAG	Target flag code
		TGTSTATUS1	Target Status: = 1 for simple target; = multiplicity for multiple target element; = complex number for complex target element
		TGTLAT	Target latitude
		TGTLONG	Target longitude
		TGTRAD	Target radius
		VTO	Total original value of target
		M	Number of hardness components
		H1	Lethal radius first hardness component (1MT)
		H2	Lethal radius second hardness component (1MT)
		FVALH1	Fractional value of first hardness component

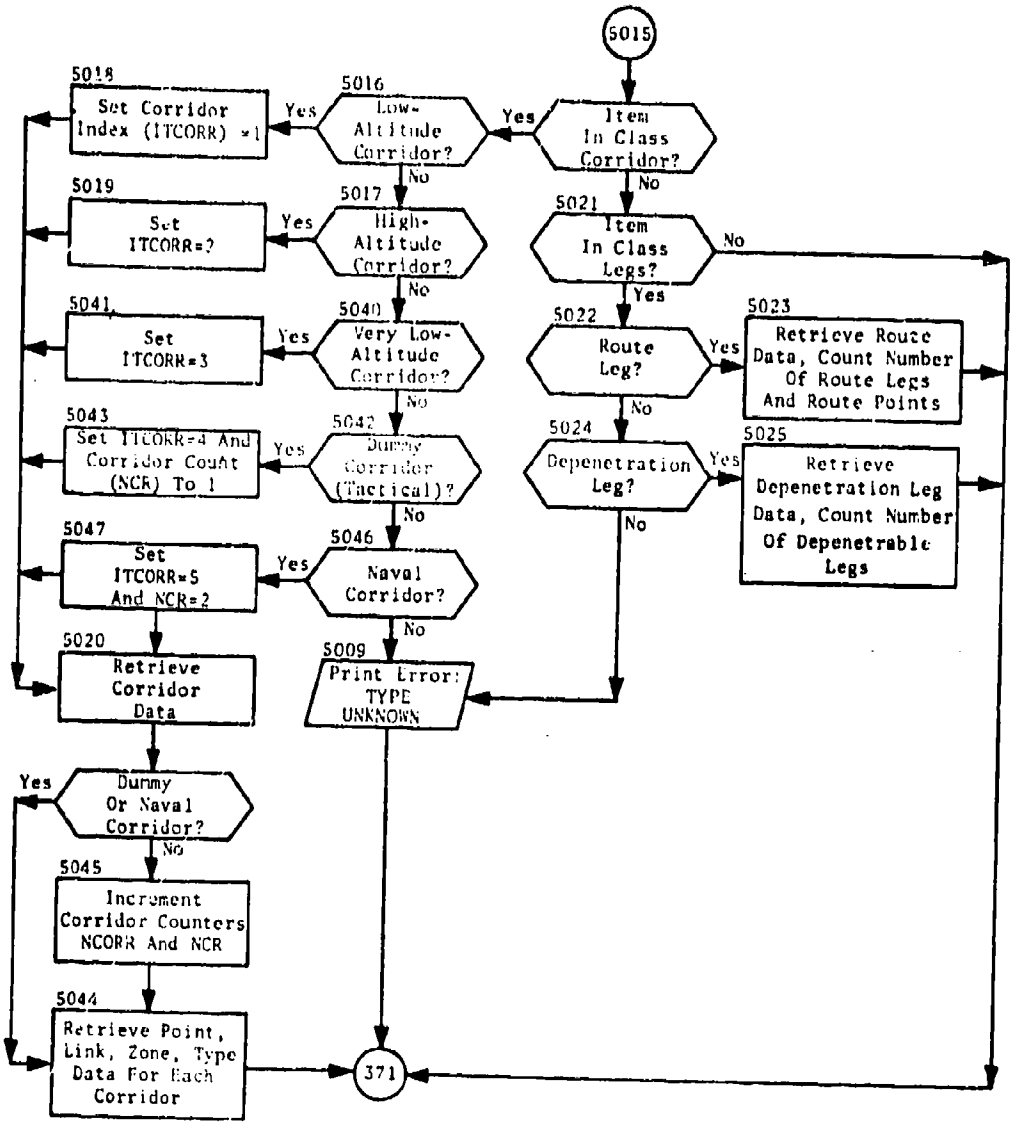


Fig. 2. (cont.)
(Sheet 20 of 20)

Table 4. Program PLANSET Common Blocks
(Sheet 1 of 15)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
CLASNAME	CLASNAME(15)	Hollerith name for Class I
	CLASVAL(15)	Relative value for Class I
	CUMVAL(15)	Total value for Class I
	VALFAC(15)	Fraction of total data base value for Class I to be applied in the current plan
DIRECTRY	MLTCOMP/ FMLTCOMP(I,J) (I = 8, J = 315)	Multiple target data array, where for target J: I = 1: target name 2: target index number 3: target designator code 4: target task code 5: target country location code 6: target flag code 7: target latitude 8: target longitude
	LCPX(2500)	Index to ICPLX array for first element of complex I
DPOOL	IB(200)	Index to point table for boundary point I
	LINKB(200)	Index of a leg linked to the current point (by side)
	ZONEB(200)	Defense zone enclosed by a set of linked boundary points
	NEXTZB(200)	Adjacent zone
	ICHKFLG(20)	Names of arrays which have overflowed
	ICHKNUM(200)	Number of items in overflowed arrays
	IC(30)	Index to point table for corridor point I
	LINKC(30)	Index of a leg linked to the current corridor point

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 4. (cont.)
(Sheet 2 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DPOOL (cont.)	ZONEC(30)	Defense zone in which corridor is located
	ITY(30)	Corridor type
	ID(50)	Index to point table for depenetration point I
	LINKD(50)	Index of a leg linked to the current depenetration point
	KORSTY(5)	Parameter to adjust mode of corridor penetration. Power of y versus x
	HILOAT(5)	Ratio low to high altitude attrition (less than 1)
	DEFR(5)	Characteristic range of corridor defense
	ATTRS(5)	Suppressed high altitude attrition per nautical mile
	ATTRC(5)	Unsuppressed high altitude attrition per nautical mile
	IL(200)	Index to point table for leg I
	LINKL(200)	Index to leg linked to current leg point
	ATRL(200)	Attrition parameter for leg ending with this point
	TMASW(10,10)	Time data for DBL data tables
	DBLASW(10,10)	Probability data for DBL data tables
	NTIMES(10)	Number of entries in Ith table
	TIMESTRT(200)	First launch time for group I
	NALLOW(200)	Maximum number of weapons that can be added to group I
	BLAT(200)	Zone latitude

Table 4. (cont.)
(Sheet 3 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DPOOL (cont.)	BLONG(200	Zone longitude
	RLAT(200)	Latitude of route point I
	RLONG(200)	Longitude of route point I
	LINKR(200)	Index to the leg linked to recovery point I
	RECLAT(200)	Latitude of recovery base I
	RECLON(200)	Longitude of recovery base I
	IRECPCTY(200)	Capacity of recovery base I
	INDREC(200)	Index of recovery base I
	RFLAT(20)	Latitude of refuel point I
	RFLONG(20)	Longitude of refuel point I
	CUMNO(15)	Total number of types in Class I
	BTYPES(15)	Number of BLUE types in Class I
	INDCLAS(15)	Beginning index number for Class I
	INDBEG(250)	Beginning index number for type I
	TYPENAME(250)	A list of types in the order referred to by INDBEG
	NTYPS	Total number of missile and bomber types
	CHK(250)	Contains flags for types in this plan; corresponds to TYPENAME array
	PG(12)	Vulnerability data for VLRAD
	PA(12)	Vulnerability data for VLRAD
	QG(8)	Vulnerability data for VLRAD
	QA(8)	Vulnerability data for VLRAD
	ITEMP(5000)	Temporary array used to hold target numbers assigned during first pass in subroutine TGTSORT

Table 4. (cont.)
(Sheet 4 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
DPOOL (cont.)	ITANK(I,J) (I = 12, J = 200)	Each column J contains the 12 words of tanker squadron description as in array TANK of common /3/. The data stored is for tankers which are pre-assigned to refuel areas
	JTANK(I,J) (I = 12, J = 200)	Same as ITANK, except used for tankers to be automatically assigned to a refuel area by the plan generator
GROUP	GRP/IGRP(I,J) (I = 14, J = 210)	Data for weapon group J I = 1: total number weapons in group 2: total number vehicles in group 3: latitude averaged (group centroid) 4: longitude averaged (group centroid) 5: geographical region index 6: type index 7: alert status (1 = alert, 2 = nonalert) 8: DBL probability for alert vehicle 9: payload index for missiles, refuel index for bombers 10: yield of bombs averaged 11: starting index number for group 12: number of bases in group 13: index to time dependent DBL data table 14: single shot kill probability against naval targets
	INDGRP(210)	Group breakpoint table
	NWDSGRP	Number of words per group

Table 4. (cont.)
(Sheet 5 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION**
GROUP (cont.)	JTGT(2500)	The TGTSORT-assigned target number for elements in complex I
MAX (Contains the QUICK maximum limits)	MAIRDEZ	Area ABM defense zones (20)
	MALERT	Alert conditions (2)
	MASMTYP	ASM types (20)
	MBNDRY	Boundary (200)
	MCCREGN	Command/control (20)
	MCLASS	Weapon classes (2)
	MCNTRYS	Country codes (250)
	MCORR	Penetration corridors (30)
	MCORTYP	Corridor types (5)
	MDPEN	Depenetration corridors (points) (50)
	MDEPNLG	Depenetration legs (50)
	MGROUP	Weapon groups (200)
	MPAYLOD	Payload types (per side) (40)
	MRECOVR	Recovery bases (points) (200)
	MRECVLG	Recovery legs (60)
	MREF	Refuel points (directed) (20)
	MRTLEG	Route legs (200)
	MRTPT	Route points (200)
	MSPERMI	Sites per multiple target (5)
	MTANKBS	Tanker bases (50)

** Parenthetical values indicate the maximum QUICK capabilities.

Table 4. (cont.)
(Sheet 6 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
MAX (cont.)	MTARCLS	Target classes (15)
	MTARCOL	Targets, collocated (4000)
	MTARCPX	Target complexes (total) (4000)
	MTARERC	Targets per collocation island (100)
	MTARGET	Targets (allocator) (5000)
	MTARIND	Target index numbers (12000)
	MTARSEC	Targets per earth sector (4000)
	MTARTEI	TGTS with terminal interceptors (ABM) (500)
	MTARTYP	Target types (total) (250)
	MTARVAL	Target complex with value > 0 (2500)
	MTELMCM	Target elements per complex (40)
	MTOTBAS	Weapon bases per group (150)
	MTYPE	Weapon types (missiles + bombers per side) (80)
	MVULN	Unique target vulnerabilities within the game base (50)
	MWEAPGP	Weapons per group (missiles + bombers) (1000)
	MWHDTPE	Warhead types (50)
	MZONEPT	Zone points (200)
	MZONES	Zones (63)
	MTARPCL	Target types per class (40 = missiles + bombers/20 = others)
MISC	IN/FIN(5)	Array to which exemplar target input option cards are read

Table 4. (cont.)
(Sheet 7 of 15)

BLOCK	VARTABLE OR ARRAY	DESCRIPTION
MISC (cont.)	ITTAPE(5)	Array containing data for TINFILE header record
MLTX	MULT(21)	Array containing multiple target data. Entries correspond to those for common block /TD/
	MLTX/FMLTX(I,J) (I = 8, J = 5)	I words of data for the Jth multiple target element I = 1: target name 2: target index number 3: target designator code 4: target task code 5: target country location code 6: target flag code 7: target latitude 8: target longitude
	NMULT	Number of elements in the current multiple target
PRCNTL	JJJTGTS	Contents of target entry on print option card
	JJJGP	Contents of group entry on print option card
	JJJCPX	Contents of complex entry on print option card
PRIOR	IPTASK(35)	Array into which TASK priority option cards are read
	IPDES(96)	Array into which DESIG priority option cards are read
	MPTASK	Number of TASKs in IPTASK array
	MPDES	Number of DESIGs in IPDES array
	ISUBT	Task flag: If 0, only 1 character in TASK
RETARG	REDUCE(40)	The factor to be applied to NOPERSQN for missile type I when retargeting is considered

Table 4. (cont.)
(Sheet 8 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
RETARG (cont.)	RDBL(40)	The new ALERTDBL for missile type I when retargeting is considered
TAPES	LTWIN	Logical unit for WINFILE
	LTTIN	Logical unit for TINFILE
	LTTGT	Logical unit for intermediate target file
	LTGRP	Logical unit for intermediate group file
	LTDB	Logical unit for INDEXBD input
	LSRTA	Logical units for sorting complex target information for sequenced printout
	LSRTB	
TAU	TAU(90)	Time components of all elements of a complex
	HC(60)	Hardness components of all elements of a complex
	V(90)	Values corresponding to TAU or HC
	INDEXT(90)	Array containing ordered indices
	FV(3)	Fractions of total value corresponding to V
	T(3)	Array containing time components for single target
	TBOX(3)	Not used
	VBOX(3)	Not used
	H(2)	Array containing hardness components for single target
	FVH(2)	Array containing fractions of values lost for single target

Table 4. (cont.)
(Sheet 9 of 15)

BLOCK	VARIABLE OR ARRAY		DESCRIPTION
TD	TD/ITD(31)		Target data array, where the information contained for each target is:
		TGTNAME	1 = 1: target name
		INDEXNO	2: target index number
		DESIG	3: target designator code
		TASK	4: target task code
		CNTRYLOC	5: target country location code
		FLAG	6: target flag code
		TGTSTATUS1	7: target status =1 for simple target, =multiplicity for a multiple target element, =complex number for a complex target element
		TGTLAT	8: target latitude
		TGLONG	9: target longitude
		TGTRAD	10: target radius
		VTO	11: total original value of target
		M	12: number of hardness components
		H1	13: lethal radius first hardness component (1MT)
		H2	14: lethal radius second hardness component (1MT)
		FVALH1	15: fractional value of first hardness component
		NK	16: number of time components
		FVALT1	17: fractional value of first time component
		FVALT2	18: fractional value of second time component
		TAU1	19: first time component
		TAU2	20: second time component
		TAU3	21: third time component
		IHCLASS	22: target class name
		ICLASS	23: target class number
		IHTYPE	24: target type name (or number complex target elements if class is complex)

Table 4. (cont.)
(Sheet 10 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
TD (cont.)	TARDEF	25: state of local bomber defense
	MISDEF	26: state of terminal missile defense
	MINKILL	27: minimum kill probability required
	MAXKILL	28: maximum kill probability desired
	MAXCOST	29: maximum (weapon cost/ target value) acceptable to achieve MINKILL
	TGTSTATUS2	30: target status =0 if single or multiple target, =1 if lead element of a complex target, =2 if additional element of a complex target
	TGTNUM	31: target number assigned by subroutine TGTSORT
WT	WT	Dummy - filled as needed
	IDATE	Hollerith date
	IDENTNO	Data input identification number
	LSIDE	Side
	NRTPT	Number of route points
	NCORR	Number of corridors
	NDPEN	Number of depenetration corridors
	NRECOVER	Number of recovery points
	NREF	Number of refuel points
	NBNDRY	Number of boundary points
	NREG	Number of regions
	NTYPE	Number of weapon types
	NGROUP	Number of weapon groups
	NTOTBASE	Total number of bases

Table 4. (cont.)
(Sheet 11 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
WT (cont.)	NPAYLOAD	Number of entries in payload index table
	NASMTYPE	Number of ASM types
	NWHDTYPE	Number of warhead types
	NTANKBAS	Number of tanker bases
	NCOMPLEX	Number of complex targets
	NCLASS	Number of weapon classes
	NALERT	Number of alert conditions
	NCORTYPE	Number of corridor types
1	MAXICOMP	Total number of complex targets
	NCPX(2500)	Number of elements in complex I
	ICNUM(105)	List of complex numbers currently stored in ICPLX
	ICPNT(105)	First column of ICPLX containing the complex number stored in ICNUM (I) for complex I
	ICNDX(105)	Indices to ICNUM in ascending complex number order
	LTYPE(250)	Weapon type number assigned by PLANSET. Corresponds to TYPENAME array in DPOOL. Equivalenced to array ICNUM.
2	WHD/IWHD(I,J) (I = 3, J = 50)	Warhead table, where for each warhead type J: I = 1: yield 2: dud probability 3: fission fraction
	ASMT/IASMT(I,J) (I = 5, J = 20)	ASM table, where for each ASM type J: I = 1: ASM warhead type 2: ASM range 3: ASM reliability 4: CEP of the ASM 5: ASM speed

Table 4. (cont.)
(Sheet 12 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
2 (cont.)	PLD/IPLD(I,J) (I = 10, J = 40)	Payload table, where for each payload type J: I = 1: number of bombs of type 1 (for MIRVs, the number of IRVs) 2: type index of first bomb 3: number of bombs of type 2 4: type index of second bomb 5: number of ASMs 6: ASM type

Table 4. (cont.)
(Sheet 13 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
2 (cont.)		I = 7: number of countermeasures (bombers). Degradation factor (missiles) 8: number of decoys (for MIRVs, the number of terminal decoys per IRV) 9: number of area decoys 10: MIRV system identification number
	RGN(20)	Reliability factors for each command-and-control region I
	WTP,IWTP(I,J) (I = 20, J = 80)	Weapon table, where for each weapon type J: I = 1: weapon type name 2: weapon range (nautical miles) 3: weapon CEP (averaged) (nautical miles) 4: weapon speed (knots) 5: weapon delay before launch when on alert status (hours) 6: weapon delay before launch when not on alert status (hours) 7: weapon range decrement for low altitude flight 8: weapon class index (1 for missiles, 2 for bombers) 9: number weapons per squadron (missiles), number in commission (bombers) 10: speed at high altitude (knots) 11: speed at low altitude (knots) 12: dash speed (knots) 13: weapon range with refueling (nautical miles)

Table 4. (cont.)
(Sheet 14 of 15)

BLOCK	ABLE OR ARRAY	DESCRIPTION
2 (cont.)		I = 14: weapon reliability 15: number of weapons per site 16: reprogramming index (for missile squadron) 17: recovery mode index (1 for normal recovery probability, 0 for no recovery, -1 for low recovery probability) 18: penetration mode index (1 for normal use of corridors, 0 for corridors not used) 19: weapon type index used by the simulator 20: weapon function code (Hollerith)
3	TANK/LTANK(12)	Tanker data array, where for each tanker base: I = 1: tanker index number 2: tanker base latitude 3: tanker base longitude 4: refuel area assigned to tanker base 5: number of tankers per squadron or base 6: number of alert tankers per base 7: tanker speed in knots 8: alert delay 9: nonalert delay 10: total time on station 11: tanker type index 12: tanker range
	GRPX/IGRPX(6)	Weapon group data for output to LTGRP, where for each group element: I = 1: group number 2: base index number

Table 4. (cont.)
(Sheet 15 of 15)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
3 (cont.)		I = 3: base latitude 4: base longitude 5: base payload index 6: number of first vehicle/ number per base
12	TAR/ITAR(I,J) (I = 29, J = 210)	Array in which target data is stored during subroutine TGTSORT. For each target table entry J, I = 1 to I = 29 correspond to the same variables in common block /TD/
	CPLX/ICPLX(I,J) (I = 30, J = 210)	Array in which complex target data is stored during subroutine TGTSORT. For each table entry J, I = 1 to I = 29 correspond to the same variables in common block /TD/. ICPLX(30, J) is a copy of the complex number
	LOOK(30)	Hold area for LSRTA and LSRTB data
	GRPCOMP/IGRPCOMP(I,J) (I = 5, J = 2500)	Array containing data for each weapon in a group. For each group element J, I = 1: base index 2: base latitude 3: base longitude 4: base payload index 5: number of first vehicle/ number per base

SUBROUTINE AROVRFLW

PURPOSE:	To print an error message in the case of an array overflow during PLANSET execution.
ENTRY POINTS:	AROVRFLW
FORMAL PARAMETERS:	None
COMMON BLOCKS:	DPOOL
SUBROUTINES CALLED:	ABORT
CALLED BY:	PLANSET

Method

The array ICHKFLG contains the name of the type of item which has exceeded the maximum number of that type allowed in the current version of QUICK. (See description of common block MAX.) The corresponding element in the ICHKNUM array contains the number of that item type encountered. AROVRFLW (see figure 3) examines each element of ICHKFLG, and prints the error message ARRAY OVERFLOW * * (ICHKNUM(I)) (ICHKFLG(I)) for each nonzero word it encounters. If any errors are found, the program then aborts.

The flowchart for subroutine AROVRFLW is shown in figure 3.

SUBROUTINE TGTSORT

PURPOSE:	To rearrange the list of targets so that classes are more evenly distributed throughout.
ENTRY POINTS:	TGTSORT
FORMAL PARAMETERS:	NTAR (number of targets)
COMMON BLOCKS:	CLASNAME, 1, DPOOL, DIRECTRY, GROUP, 12, IFTPRNT, ITP, MAX, MLTX, MYIDENT, PRCNTL, TAPES, TWORD, TD
SUBROUTINES CALLED:	ABORT, CALCOMP, ORDER, RDARRAY, SETREAD, SETWRITE, TERMPAPER, WRARRAY, WRWORD
CALLED BY:	SHUFFLE

Method

TGTSORT begins by assigning an index IND to each target as it is read from the target tape. This index is used to distribute the targets uniformly throughout the target list by cycling through them as follows: a sorting index (LEAD), which is a function of the total number of Targets NTAR), is calculated by the formula:

$$LEAD = \frac{NTAR\ (3 - \sqrt{5})}{2}$$

To start a cycle, a beginning index (IBEG) is designated and assigned to the target being read. Initially IBEG = LEAD. The index for the next target (IND) then is found by incrementing the previous index (LAST) by LEAD. If the result exceeds NTAR, the cycle is reset by subtracting NTAR from IND. When a cycle is completed (i.e., when IND = IBEG), the next cycle is begun by incrementing IBEG by one and proceeding as above. Thus, a unique nonsequential index is assigned to each target as it is read.

Because of storage limitations on the arrays ITAR, ICPLX, and MLTCOMP several passes through the target list may be necessary before all data can be processed. In order to determine which targets are to be considered during a given pass, the index limits MIN (lowest target number in this pass minus one) and MAX (maximum number of targets which can be processed per pass) are established. These limits, together with the index IND, are

used to find an index (IT) to the column in ITAR in which data corresponding to IND are to be stored. The procedure is as follows.

As each target is encountered, the value of MIN for that pass is subtracted from the index IND corresponding to the target. If the result (IT) lies between one and MAX (210), the target data is stored in column IT of array ITAR, as required. Otherwise, the target is ignored for the remainder of the pass.

To indicate that a target was not processed during a pass, a zero is placed in the second word of each column of ITAR when the pass begins. If data for a target are stored, the zero is replaced by the attribute INDEXNO. Then, after the pass is completed, a check is made on the content of ITAR (2,I), $1 \leq I \leq 210$. If a positive value is found, target data for the corresponding target are processed as necessary and written onto TINFILE. The first time a zero is encountered, the index MIN is incremented by the number of targets successfully processed (I - 1), and another pass is begun.

When the index check indicates that data for a complex or multiple target are to be saved during a pass, data for all other elements of the target must also be included in the same pass. Therefore, the number of elements is tested against available storage in the complex ICPLX or multiple MLTCOMP target array. If the required data will not fit into the array, the value of MAX is decreased by 10, the variable MAXCHNG is set to two, and the entire pass is restarted. The process continues until the number of targets to be considered during the pass is sufficiently decreased to permit the storage of all necessary data in ICPLX or MLTCOMP. When this occurs, the variable MAXCHNG will still equal one after the last target has been read from the target tape, thereby indicating that a new pass is to begin. If data for the first target in a complex are not stored during a pass, data for additional complex members also are ignored.

In order to avoid repeating index assignments, the value of IND calculated during the first pass is stored in the array ITEMP for all targets. Thus, as each target is encountered during subsequent passes, the appropriate IND is retrieved from ITEMP for use in computing the new value of IT. In addition to calculating IND on the first pass, certain target data (including NAME, INDEXNO, multiplicity, TYPE, complex indicator, VALUE, and NTINT) are printed for each target if the target prints were requested by the user.

When the above testing finds a target to be processed, TGTSORT proceeds in the following manner. The class name and value are stored in the appropriate locations of the target array ITD. For multiple and individual targets, the first 29 items of data then are read into column IT of array ITAR, and the value (VAL) is set at a minimum of 1×10^{-12}.

DISTRIBUTION

Addressee	copies
NMCSSC Codes	
B121	3
B122 (stock)	6
B200	1
B210	2
B220	29
B600	1
DCA Codes	
260 (Original document only, no subsequent changes)	1
920	1
950	1
OJCS	
Studies, Analysis and Gaming Agency, ATTN: SFD, Room 1D957, Pentagon, Washington, D.C. 20301	5
Commander-in-Chief, North American Air Defense Command ATTN: NPPG, Ent Air Force Base, Colorado 80912	2
Commander, U.S. Air Force Weapon Laboratory (AFSC) ATTN: AWL, Kirtland Air Force Base, New Mexico 87117	2
Director, Strategic Target Planning Offutt Air Force Base, Nebraska 68113	2
Chief of Naval Operations, ATTN: OP963G Room 5E531, Pentagon, Washington, D.C. 20350	2
Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314	12
	70