UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CAA-D-84-14	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Operator's and Programer's Guide to the Analysis of Force Potential System (AFPSYS)		5. TYPE OF REPORT & PERIOD COVERED System Documentation January 83/November 84
		6. PERFORMING ORG. REPORT NUMBER CAA-D-84-14
7. AUTHOR(s) Mr. Gerald E. Cooper LTC Raymond K. Elderd Ms. Beverly M. Knox MAJ Edgar E. Stanton III Mr. John W. Warren		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Concepts Analysis Agency ATTN: CSCA-RQ 8120 Woodmont Avenue, Bethesda, MD 20814-2797		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Concepts Analysis Agency ATTN: CSCA-RQ 8120 Woodmont Avenue, Bethesda, MD 20814-2797		12. REPORT DATE November 1984
		13. NUMBER OF PAGES 347
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

N/A

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Combat Potential, Static Measures of Combat Potential, Equipment Potential, Force Potential

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

The documentation provides a brief general description of the Analysis of Force Potential System and detailed descriptions of all the modules and submodules of the system. Input data, runstream generators, runstreams, source programs, and system output are all described, usually with examples.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DOCUMENTATION
CAA-D-84-14

AD

OPERATOR'S AND PROGRAMER'S GUIDE TO THE ANALYSIS OF FORCE POTENTIAL SYSTEM (AFPSYS)

NOVEMBER 1984

PREPARED BY

US ARMY CONCEPTS ANALYSIS AGENCY
8120 WOODMONT AVENUE
BETHESDA, MARYLAND 20814-2797

DISCLAIMER

The findings of this report are not to be construed as an official Department of the Army position, policy, or decision unless so designated by other official documentation. Comments or suggestions should be addressed to:

Director
US Army Concepts Analysis Agency
ATTN: CSCA-RQ
8120 Woodmont Avenue
Bethesda, MD 20814-2797

DOCUMENTATION
CAA-D-84-14

OPERATOR'S AND PROGRAMER'S GUIDE TO THE ANALYSIS OF FORCE POTENTIAL SYSTEM (AFPSYS)

NOVEMBER 1984

PREPARED BY
US ARMY CONCEPTS ANALYSIS AGENCY
8120 WOODMONT AVENUE
BETHESDA, MARYLAND 20814-2797

DEPARTMENT OF THE ARMY
US ARMY CONCEPTS ANALYSIS AGENCY
8120 WOODMONT AVENUE
BETHESDA, MARYLAND 20814-2797

REPLY TO
ATTENTION OF:

CSCA-SPP 27 NOV 1984

MEMORANDUM FOR RECORD

SUBJECT: Analysis of Force Potential (AFP)

1. In October 1983 CAA initiated a major developmental effort to provide new static measures of the combat potentials of equipment and major organizations. The resulting AFP System is central to a new method for Measuring Improved Capabilities of Army Forces (MICAF).

2. This guide provides a detailed description of the AFP System and its components.

1 Incl
as

E. B. VANDIVER III
Director

ANALYSIS OF FORCE POTENTIAL SYSTEM (AFPSYS)	STUDY SUMMARY CAA-D-84-14

THE REASON FOR PERFORMING THE STUDY is primarily widespread dissatisfaction with previous combat potential estimation methods that do not give enough attention to influences noted below in the study objectives.

THE PRINCIPAL FINDINGS during AFP System development and implementation and as evidenced by illustrative examples in the Operator's and Programer's Guide to the AFP System and by the parallel MICAF Study application are:

(1) All modules, submodules, and special processors of the AFP System for estimating the static combat potential of equipment and organizations have been tested and perform as designed.

(2) AFP estimates of static combat potentials depend on input to the AFP System and are sensitive to opposing sides' weapon characteristics, weapon quantities, type-on-type engagement preferences, environmental conditions, and combat support and combat service support levels.

(3) Full application of the AFP System is labor, data, and computer intensive.

THE MAIN ASSUMPTIONS for purposes of estimating static combat potentials:

(1) The large-scale battlefield may be decomposed into separate firepower-counterfirepower, combat support, and combat service support processes. These processes may be analyzed largely independently. Their separate results may be combined afterward to yield estimates of combat potentials.

(2) Total division firepower-counterfirepower processes may be decomposed into pure weapon type on pure weapon type engagements. The engagements may be further decomposed into still smaller matchups in which at least one weapon opposes one or more weapons. Only indirect, area fire weapons may impinge on the interaction of otherwise pure type-on-type "duels." The usual techniques of dynamic modeling and simulation need not be applied except to the independent duels of relatively short duration.

(3) Movement and maneuver need not be represented within the firepower-counterfirepower process. Tactical mobility may be treated adequately within the combat support and combat service support processes. Duels are distributed to fixed ranges.

THE PRINCIPAL LIMITATIONS

(1) Like all static indicators, AFP combat potentials may be inappropriate bases for estimating prolonged, fluid combat.

(2) Because AFP combat potentials depend on weighted averages for 16 distinct combat environments, the potentials may not be useful estimators for differently weighted or different environments. For example, interest in just one of the combat environments implies a vastly different weighting: just one 1.0 and 15 0.0's.

(3) AFP combat potentials are estimates of achievement for the very special circumstance in which one's own weapons are 50 percent attrited. (This is why AFP combat potentials are often called "half-life potentials.") In general, the potentials do not correspond to any one common moment in projected real time because different weapon types do not reach 50 percent survival at the same instant.

(4) In its current implementation, the AFP System does not represent suppression nor the effects of echelons above division (other than some nondivisional artillery and some fixed wing aircraft).

THE SCOPE OF THE STUDY included development and implementation of the AFP System and parallel support of the MICAF Study. The Operator's and Programer's Guide to the AFP System provides a wealth of information needed in maintaining and applying the AFP System. Some applications of the AFP System have been made in support of other studies. In particular, the MICAF I and II Studies depended heavily on AFP, and AFP "results" may be found in the MICAF I and II reports.

THE STUDY OBJECTIVES are to develop and demonstrate (via the parallel MICAF application study) a new method for estimating the static combat potential of equipment and organizations. That method is to depend more directly on quantitative data, full division inventories of opposing equipment, combat support, combat service support, and wider range of combat environments than in previous approaches.

THE BASIC APPROACH of AFP is to begin with a highly stylized abstraction of the battlefield, decompose the battlefield into separate processes, provide extensive input data to drive those processes, and then operate a system of specially developed computer programs which replicate estimates of kills and losses for 16 different combat environments, project those estimates to half-lives, modify the estimates in accord with support levels, and roll up everything into final estimates of combat potential.

THE STUDY SPONSOR is the Director, CAA.

THE STUDY EFFORT was directed by Mr. Gerald E. Cooper, Strategy, Concepts and Plans Directorate. All directorates contributed.

COMMENTS AND QUESTIONS may be directed to US Army Concepts Analysis Agency, ATTN: Assistant Director for Requirements and Resources, 8120 Woodmont Avenue, Bethesda, MD 20814-2797

Tear-out copies of this synopsis are at back cover.

CONTENTS

FIGURES

FIGURE		Page

FIGURE | Page

FIGURE		Page

FIGURE | Page

FIGURE | Page

FIGURE		Page

FIGURE | Page

TABLES

OPERATOR'S AND PROGRAMER'S GUIDE TO THE ANALYSIS OF FORCE POTENTIAL SYSTEM (AFPSYS)

I. INTRODUCTION

1. The purpose of this volume is to provide operator/programer documentation of the Analysis of Force Potential System (AFPSYS) data, computer programs, and procedures for estimating and often comparing the combat potential of Army equipment and organizations (currently up to and including division level).

2. The complete AFPSYS includes a wide variety of input data files from several sources; utility programs for converting input to forms better suited for use within AFPSYS; major program modules for estimating combat, combat support, and combat service support results, and for merging such results for a single combat environment; a major program module for merging the results of 16 combat environments into complete estimates of combat potential; modules for comparing combat potentials among equipment types and among different organizations; and utility programs for managing files and generating runstreams. AFPSYS runs under the normal UNIVAC 1100-series computer operating system. AFPSYS modules and utilities are written in ASCII FORTRAN, COBOL, and SYMSTREAM, a language peculiar to the UNIVAC SSG processor.

II. SCOPE

3. This volume provides reference material for the execution of AFPSYS. It includes information necessary for an operator/programer to modify existing data, generate data for "new" equipment or organizations, create new file names and the corresponding files, and execute utilities and program modules. The main part of this volume is very short; it provides little more than a short description of AFP in general and an index to the volume's appendices which provide:

a. A diagram of the complete system with some discussion of the notation and implications of the diagram: Appendix B. The diagram provides the "road map" for the following appendices.

b. A key to the data preprocessing utility programs: Appendix C.

c. A description of the Combat Module: Appendix D.

d. A description of the Combat Support/Combat Service Support Module and its preprocessor program: Appendix E.

e. A description of the module for the merge of combat/combat support/combat service support results for a single combat environment: Appendix F.

f. A description of the module for merging a single organization's results over all 16 combat environments: Appendix G.

g. A description of the programs for comparing different organizations: Appendix H.

h. A description of a special procedure for estimating the combat potential of an organization "between" two organizations for which results are already known (i.e., AFP interpolation): Appendix I.

i. A key to the various reports producible throughout execution of AFPSYS: Appendix J.

j. A key to the AFPSYS approach to runstream management via the SYMSTREAM language of the UNIVAC SSG processor: Appendix K.

III. VIEWPOINT

4. It is assumed that an operator/programer is going ahead with an AFPSYS application. This volume does not argue for or against the AFP approach. It does not replay consideration of the relative merits of static and dynamic approaches to the estimation of combat potential. It is assumed that the operator/programer is familiar with the UNIVAC operating system, and with the FORTRAN, COBOL, and SYMSTREAM programing languages. Even then, the new operator/programer should have the services of someone already experienced in operation of AFPSYS and in the interpretation of results.

5. Many AFPSYS parameters were set at the beginning of system development with the prospect of changing their values in order to improve the reliability of results and/or speed of system execution. However, debugging of the AFP Combat Module consumed so much time and energy that too little time and resource were left for "improvements." The AFP team believes that improvements are desirable and achievable. For example, if run times of the main modules can be reduced enough, the AFP interpolation process can be discarded. In a sense, a change of an AFPSYS system parameter defines a new AFPSYS. Indeed, in that view, there is a family of imaginable AFP systems. Certainly not all members of that family would be equally useful and efficient. There may be some optimal member of the family. Unfortunately, no good procedure exists at this time for finding that best member. However, some effort should be devoted to finding parameters that permit the system to evolve to steadily better performance.

IV. AFP IN GENERAL

6. AFP provides a new way to estimate the combat potential of weapons and units. However, like several earlier measures, AFP estimates of combat potential fall in that broad class known as "static measures." Among other things, that means that relatively little of the dynamics of warfare is treated explicitly within AFP. Indeed, the AFP method is not now nor was it ever intended to involve what is usually called "combat simulation." On the other hand, the AFP method does estimate attrition, survival, and, hence, exchange ratios from very short duration sampling of highly stylized duels. Still, CAA prefers to insist that AFP is not a simulation nor a model in the usual senses of those terms.

7. The AFP approach is asserted to be superior to earlier "static" efforts because explicit attention is given to the level and mix of weapons in both the US and threat units or forces faced off in each of 16 distinct combat environments. Also, AFP explicitly approximates impacts of combat support and combat service support (CS/CSS). The CS/CSS approximation remains cruder than CAA would like it to be, but it is the best that can be done given the current state of knowledge of the matter.

8. AFP has been limited to divisions versus divisions with only some aircraft included from among all the resources above division level. In the long run, CAA is committed to address many more higher level resources.

9. AFP, then, is certainly more broadly based than its static measure predecessors. In practice, as shown in Figure 1, it has been necessary to split apart the total combat/CS/CSS process and address those pieces separately before combining the combat and CS/CSS results into the grand estimate of combat potential--in the last step of the AFP process. This bold splitting of the total process is reflected by the three blocks within the larger AFP block in Figure 1. The next paragraph begins a little deeper look into each of the three blocks within that larger AFP block.

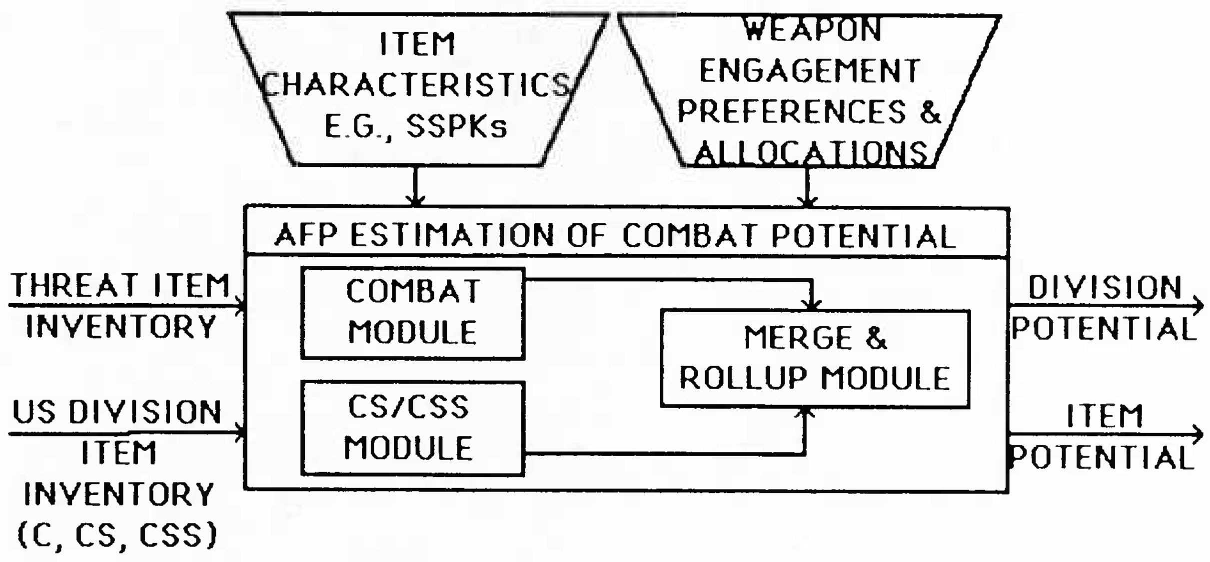


Figure 1. The Analysis of Force Potential (AFP) System in Outline

a. First, the AFP Combat Module. Only firepower and counterfirepower (less suppression) are represented in the AFP Combat Module. As suggested in the upper left corner of Figure 2, combat is represented by a matrix of possible matchups. Up to 60 weapon types on each side may be paired against opposing weapons--against just one opponent type or against all types. Weapon types are assigned to 12 weapon categories, largely as a bookkeeping convenience, but, in principle, a weapon from any category may engage or be engaged by a weapon in any other category--subject, of course, to AFP-imposed constraints. Each cell of the matchup matrix may lead to a distinct type-on-type battle. One such battle is suggested in the center of the diagram. One model of US tanks opposes one model of threat tanks. (Different models of tanks are permitted within AFP, but each model-versus-model matchup leads to a separate matrix cell and, hence, a separate minibattle. A so-called minibattle may engage from a few to several hundred tanks.) In the minibattle, the tanks are subdivided over up to five ranges from 250 to 2,500 meters. During the course of a minibattle nothing moves from one range to another. However, the weapons are "fired" subject to a probabilistic treatment of detection, firing time, and "killing." By now everyone is probably wondering why the artillery symbols are included in the tank-on-tank snapshot. The AFP Combat Module subjects the direct fire weapons to the possibility of attrition from indirect fire within the scope of minibattles. Some weapons may be assigned to a sixth, so-called "deep range." Tanks so assigned cannot fire at one another because of the great range, but they may be targets of indirect fire weapons. Units in reserve or moving up to engage occupy the "deep range." The tanks are at a disadvantage inasmuch as they are not permitted to engage the up to 10 types of indirect firers that may be pounding away. However, some of the indirect fire weapons are committed to counterbattery minibattles which may, in the long run, reduce the amount of indirect fire inflicted on the tank-versus-tank and other direct fire minibattles. Subject to AFP-imposed constraints, survivors may engage in later minibattles. The results of minibattles are estimates of survival, loss, and exchange ratio as suggested by the two-dimensional sample plot at the lower right of the figure. Starting strengths and results of all minibattles for all matchups and repetitions of matchups are saved for use in the later steps of the AFP process.

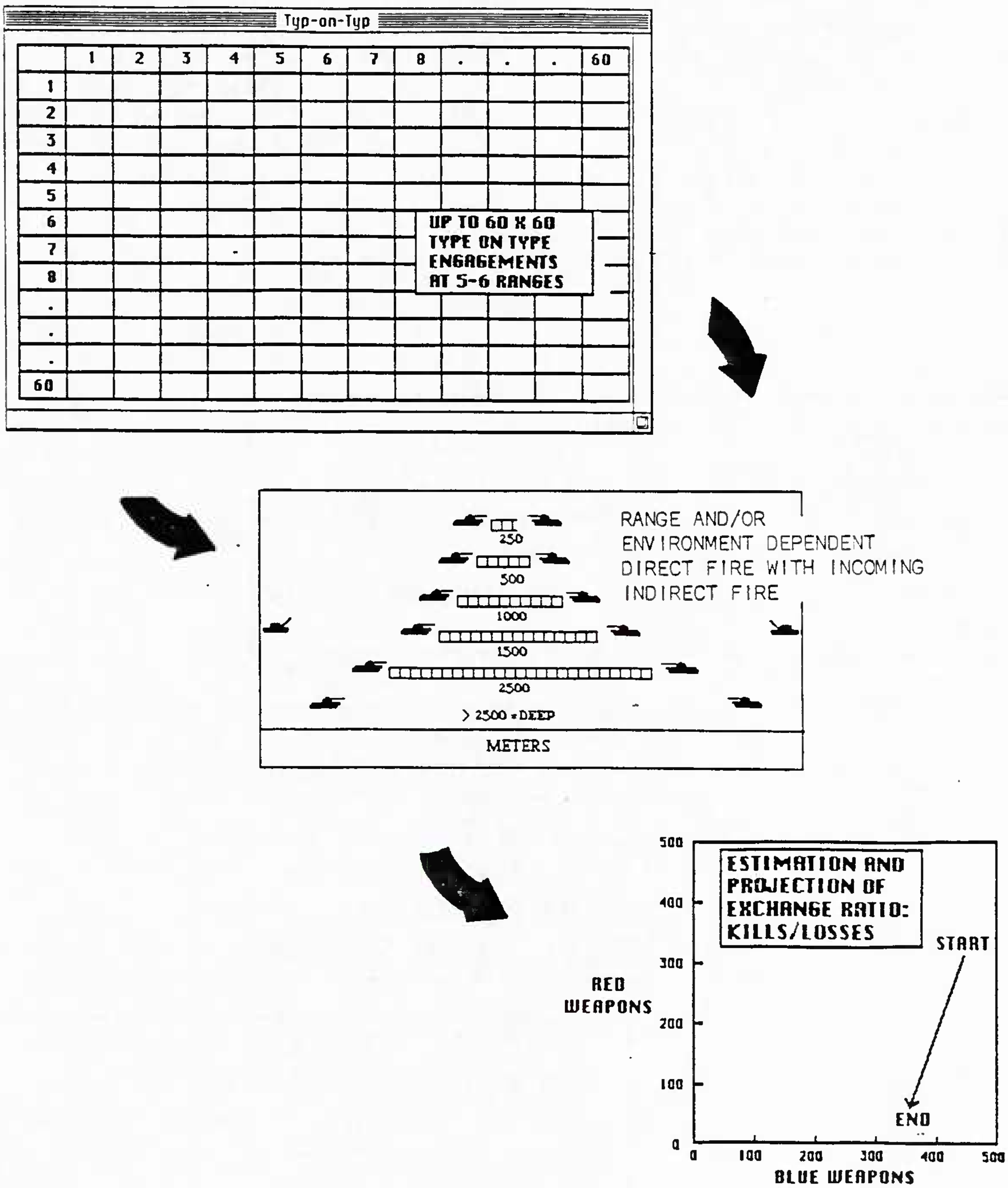


Figure 2. Major Steps in the AFP Combat Module

b. In the meantime, a separate AFP module is busy generating CS/CSS factors for the functions shown in the upper left corner of Figure 3. Depending on the specific function, both measures and countermeasures for both sides may be estimated as suggested in the center of Figure 3. The estimates, depending on the function, are based on independently derived capabilities and requirements, as appropriate. A requirement may be regarded as primarily one-sided in the sense that a maintenance requirement depends mostly on the quantity and quality of one's own assets. Or it may be regarded as two-sided in the sense that the quantity and quality of the opponent's assets and activities are the primary drivers. Capability depends on the quantity of one's people or equipment (also on the quality of equipment). The factors for the separate CS/CSS functions may be indexed or normed relative to some base unit or year and are then combined dependent on weapon-versus-weapon considerations and on environment as suggested at the lower right corner. The combined factors are identified by the somewhat esoteric term "CS/CSS moduli." These moduli are saved for combination with results of the Combat Module in the later steps of the AFP process.

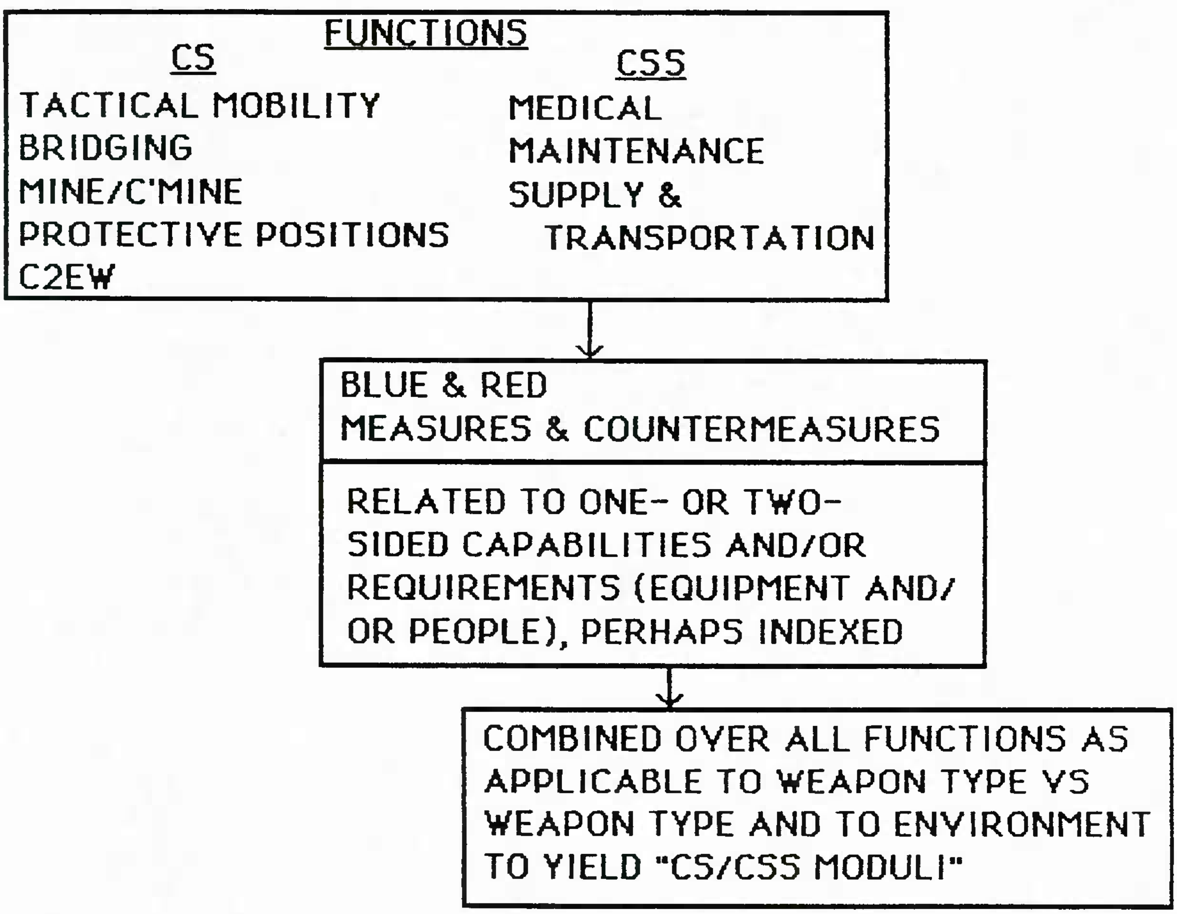


Figure 3. Major Steps in the AFP Combat Support/Combat Service Support(CS/CSS) Module

c. Figure 4 is intended to suggest how the previously generated AFP partial results become summed into final measures of combat potential of the "Blue" side. In principle, for summation over all weapons and environments, there may be more than 50,000 expressions such as the one shown here. It is the job of the final AFP module to retrieve the previously saved partial results, combine the terms correctly, and sum the results of the expression shown for all type-on-type engagements and for 16 distinct environments and possibly many replications. Summation over replications implies division by the number of replications. Because the AFP process is fully two-sided, computations similar to those suggested in Figure 4 are also performed for the "Red" side.

(1) The term, RED LOSSES, is an estimate fed from the Combat Module for the particular type-on-type engagement (here symbolized by "TT").

(2) That term is multiplied by the CS/CSS modulus for that type-on-type duel from the CS/CSS Module.

(3) In turn, that product is multiplied by the number of Blue weapons allocated; this term and all the others like it come from the Combat Module. All estimates of threat losses are projected to the same assumed fractional, clock, or resource lifetime of the friendly weapon type. In AFP work to date, combat potential has been estimated through the first half of total lifetime, the first 2 weeks of combat, or the expenditure of a basic load of munitions, depending on the equipment types.

(4) Combat potentials consist of five components: personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar. The kill of a target, in general, contributes target-type dependent amounts to the five components. The multiplier "target value" makes the appropriate adjustment. On option, certain other weapons (e.g., small arms) may be included with the "personnel" components of potentials.

(5) Because not all combat environments are equally important or likely, another multiplier expressing the relative weight by environment is applied. On option, the four-valued measure may be weighted by target values, and the "personnel" component may include other weapons.

(6) And finally, the preceding intermediate result is divided by the Blue losses of the given type over all engagements involving that Blue weapon type.

MULTIPLE SUMMATION (ΣΣΣΣ)
OVER DIVISION FOR ALL TYPE VS TYPE COMBINATIONS,
ALL ENVIRONMENTS, AND ALL REPLICATIONS
FOR DIRECT FIRE SCORES

$$\sum_{\text{REPLI-CATION}} \sum_{\text{ENVIRON-MENT}} \left(\text{ENVIRONMENTAL WEIGHT}\right) \sum_{\text{FRIENDLY TYPES}} \frac{\left(\begin{array}{l}\text{\# OF BLUE}\\ \text{TYPE}\\ \text{ALLOCATED}\end{array}\right) \times \left(\begin{array}{l}\text{BLUE}\\ \text{FRACTIONAL}\\ \text{LIFETIME}\end{array}\right) \sum_{\text{OPPOSING TYPES}} (\text{RED LOSSES})_{TT} \times \left(\begin{array}{l}\text{CS/CSS}\\ \text{MODULUS}\end{array}\right)_{TT} \times \left(\begin{array}{l}\text{TARGET}\\ \text{VALUE}\end{array}\right)}{\text{MAXIMUM}\left(\text{BLUE LOSSES}_T, 1.0\right)}$$

ZERO LOSS IS ALWAYS REPLACED BY 1.0.

WITH SIMILAR TERMS FOR INDIRECT FIRE AND SAMS

["TT" IMPLIES A VALUE SPECIFIC TO A TYPE-
ON-TYPE ENGAGEMENT.
"T" IMPLIES SUMMATION OVER OPPOSING TYPES]

THEN --
DIVISION COMBAT POTENTIAL =

DIRECT FIRE SCORES + INDIRECT FIRE SCORES + SAM SCORES

= POTENTIAL KILLS OF
PERSONNEL,
LIGHT ARMOR,
HEAVY ARMOR,
AIRCRAFT,
SCALAR (WEIGHTED SUM OF 1ST FOUR)

Figure 4. Symbolic Representation of AFP Merge (CBT with CS/CSS) and Rollup (Over 16 Combat Environments) Process

d. There is more to the last AFP module than is shown in Figure 4. Some averaging over multiple engagements may be performed and the calculation shown in Figure 4 for a Blue division combat potential has a counterpart for the threat. Other system features permit scores for individual weapons to be generated. Also, scores without CS/CSS modulation may be produced. Either partial or total scores may be produced for personnel, light armor, heavy armor, and aircraft target classes separately in a four-valued static measure or rolled up into a single-valued weighted measure.

10. Sixteen combat environments may seem an unncessarily large variety to include within the scope of normal AFP estimation of combat potentials. On the other hand, 16 combat environments provide a rather limited set of combinations of the conditions and their levels which may influence combat outcomes within a specific theater.

a. From among all the possible light levels, AFP represents only two: day and night.

b. From among all the possible levels of obscuration, AFP represents only two: clear and degraded.

c. From among all commonly defined postures, AFP represents only four: Red Attack of Blue Prepared Defense (RAPD), Static, Red Attack against Blue Delay (RADE), and Blue Attack of Red Prepared Defense (BAPD).

d. From among all possible starting division ratios, AFP represents just four (Red divisions:Blue divisions): 4:1 3:1, 1:1, and 1:3.

e. AFP weapons may be shooters or targets or both. As a shooter or as a target, a weapon may be open or in defilade, and it may be stationary or moving.

f. From among all possible ranges at which weapons may engage, AFP represents just five at which direct fire weapons may shoot plus another deep range at which they may be targeted without returning fire.

g. Although, in principle, almost any kind of weapon can engage almost any kind of weapon, few AFP weapons engage more than 15 opposing types. Nevertheless, there could be 60x60 distinct type-on-type matchups.

11. An exhaustive examination of all the combinations of factors identified in paragraph 5 could, in the extreme, lead to as many as 2x2x4x4x2x2x 2x6x60x60x2 $\approx$ 20 million "combat outcomes." Examination of all these is at once impossible and unnecessary. Clearly the AFP approach must ignore the vast majority of combinations. Indeed, the official table of 16 combat environments is as shown in Figure 5 The environmental characteristics shown in Figure 5 reflect AFP policy, not experimental results or international rules of engagement. Notice, for example, that the RAPD posture is always represented at a 3:1 Red to Blue division starting ratio. Although not shown in Figure 5, range distribution and type-on-type weapon preferences do depend on posture but do not vary for the same type-on-type pairing within a posture. Nevertheless, the data demands of AFP are enormous. Detection and/or single shot probability of kill (SSPK) data vary by open/defilade, day/night, moving/stationary, shooter/weapon/round, range, and target types. On the other hand, such basic data as refire times depend only on the shooter/weapon/round type. The basic data do not depend on the numbers of targets (except for indirect fire). Hence, volume of AFP fire may be expected to depend directly on the numbers of firers with powerful effects on killing rates, which in turn affect survival rates (doing unto the opponent before it can do unto you), and hence, exchange ratios.

CBT ENV	CONDITION			BLUE		RED		DIVISION
	VISIBILITY	DAY/ NIGHT	POSTURE	SHOOTERS	TARGETS	SHOOTERS	TARGETS	R/B RATIO
1	CLEAR	DAY	RAPD	D S	D S	O M	O M	3:1
2			STATIC	D S	D S	O S	O M	1:1
3			RADE	O S	O S	O S	O M	4:1
4			BAPD	O M	O M	D S	D S	1:3
5		NIGHT	RAPD	D S	D S	O M	O M	3:1
6			STATIC	D S	D S	O S	O M	1:1
7			RADE	O S	O S	O S	O M	4:1
8			BAPD	O M	O M	D S	D S	1:3
9	DEGRADED	DAY	RAPD	D S	D S	O M	O M	3:1
10			STATIC	D S	D S	O S	O M	1:1
11			RADE	O S	O S	O S	O M	4:1
12			BAPD	O M	O M	D S	D S	1:3
13		NIGHT	RAPD	D S	D S	O M	O M	3:1
14			STATIC	D S	D S	O S	O M	1:1
15			RADE	O S	O S	O S	O M	4:1
16			BAPD	O M	O M	D S	D S	1:3

D= DEFILADE
O= OPEN
M= MOVING
S= STATIONARY

Figure 5. Characteristics of the 16 AFP Combat Environments

12. The following paragraphs highlight some worrisome aspects of the AFP approach.

a. In AFP, a weapon type can run up a high score only by killing much more than it loses. And because kills are credited at different values, killing tanks and aircraft earns higher scores than does killing small arms. Weapons do not, within AFP, earn points for suppression or diversion of opposing fires. In particular, counterbattery fire typically gets little credit in AFP analyses. Counterbattery fire only rarely kills an opposing indirect fire weapon. The fact that counterbattery fire forces opposing firers to move more frequently is neither credited nor represented explicitly. The refire times input to AFP presumably reflect some averaged effect of counterbattery activity, but slower enemy firing rates do not directly accrue credit to counterbattery weapons. To a degree, counterbattery fire by one side forces an opponent to engage in similar activity. Hence, counterbattery fire diverts some weapons from killing softer, perhaps valuable, other targets. AFP does not credit counterbattery weapons with equipment and lives so "saved." However, the weapons saved do, in principle, achieve higher scores because they survive longer. Thus, some of the value of counterbattery weapons shows up in the scores of other weapons.

b. The AFP process may be viewed as a sampling of many of the events possible between opposing divisions. But in AFP practice, only a small fraction of the possible events may be represented. On the theoretical AFP battlefield, some weapon type might meet at least one of each of 60 opposing weapon types at each of five or six ranges. Just one each of these possibilities requires 360 distinct AFP duels. Yet all AFP direct fire duels are mutually exclusive in the sense that no direct fire weapon can perform in two or more AFP duels concurrently. In this sense, there must be 360 of a weapon type before 360 duels may be sampled one each. It is not necessary that all 360 possibilities be tried. However, many division weapons exist at less than 360 items per division. A statistician would like to have many duels against each weapon type and at each range of interest A seemingly modest request for 20 duels against 10 weapon types at each of 4 ranges requires 20 x 10 x 4 = 800 weapons before the statistician begins to feel comfortable. Very few major weapon systems populate divisions to the level of 800 items. The replicative approach does not eliminate all such problems.

c. Low density weapons produce another AFP difficulty. The problem is clear in a trivial extreme. Suppose a division contains only one item of weapon type Z. Weapon type Z may only kill opposing weapon type A and may only be killed by weapon type B. The solitary weapon of type Z can participate in only one duel. If Z duels A, Z cannot be killed. If Z duels B, Z cannot kill. Because of AFP weapon allocation rules, replication does not help solve this problem. Even if there are 10 weapons of type Z, a problem remains. The AFP analyst/user must know in advance how to divide those 10 weapons between Z-A and Z-B duels. This problem is not exactly the same for direct and indirect fire weapons. Indirect firers can be killed only as a result of counterbattery duels. Indirect firers kill best in other than counterbattery roles. Let Y be a modern Blue indirect fire weapon system; a typical division may have only nine of weapon Y. The AFP process works best for weapons numbering in the hundreds and which can kill each other directly. Only nine of a weapon provides an allocation dilemma.

13. Because AFP estimated combat potentials depend strongly on exchange ratios, the AFP method can be very sensitive to weapon qualities and quantities--up to a point. For some weapons against some targets at some ranges in some combat environments, the result may be very one-sided. It may be so one-sided that more or better weapons for one side may make no or only little difference in outcome. Within the constraints of static analysis, AFP has been developed to be sensitive to very many, but hardly all, the possibly crucial aspects of combat. In short, AFP is often said to be context sensitive. Indeed it is, but that also means that it may be more or less sensitive to the same input changes also depending on context, e.g., combat environment with all that implies about posture, division ratio, range, weapons, and the like. Figure 6 summarizes some of the factors which are and are not represented and reflected in any modules and in the AFP indices of combat potential. The extent to which disregard of the listed factors degrades the usefulness of AFP indices of combat potential remains unknown. Disregard of the same factors for both sides almost certainly does not "average out" for different divisions in different numbers, with different objectives and doctrine, and in different postures. Although the AFP method is symmetric in its treatment of both sides, the input values are not identical for both sides. AFP indices then must be regarded as being displaced in unknown directions by unknown amounts. Hence, the ordering of some output values differing by only small amounts may be reversed. Replications of the AFP Combat Module yield estimates of variances of many components of AFP combat potentials. Inasmuch as so many possible sources of variance are disregarded in the AFP process, the variances output by AFP probably grossly underestimate battlefield variations.

AFP INDICES DO REFLECT	AFP INDICES DO NOT REFLECT
• QUANTITY AND QUALITY OF MANY KINDS OF ITEMS, E.G., TANKS, ARTILLERY	• NUMBER OR QUALITIES OF SOME OF ITEMS, E.G., MAINT SHOP SETS
• QUANTITY OF PERSONNEL MANNING/ SUPPORTING OPERATION OF ITEMS	• QUALITY, MORALE, LEADERSHIP, AND TRAINING
• SOME PARTS OF ABILITY TO ACCOMPLISH FUNCTIONS (I.E., COMMO, EW)	• COMMAND ASPECTS OF CONTROL FUNCTIONS
• PERFORMANCE CHANGES RESULTING FROM ENVIRONMENTAL CONDITIONS (CRUDELY)	• DETAILED PERFORMANCE DIFFERENCES RESULTING FROM SUPPORT OF COUNTER-MEASURES, ETC.
• EFFCTS OF SUPPORT ON SHORT-TERM SUSTAINMENT	• EFFECTS OF SUPPORT ON LONG-TERM SUSTAINMENT
• SOME FACTORS SUCH AS WPN-TO-TGT ALLOCATIONS WHICH CAN CRUDELY REPRESENT TACTICAL DOCTRINE	• DIFFERENCES IN TACTICAL DOCTRINE IN DETAIL

Figure 6. Indices -- "Do's and Dont's"

14. The AFP approach depends on many assumptions. The development team inherited a variety of guidelines that appeared to have been based on someone's combination of experience and intuition. The origins of these guidelines were never documented. AFP's inclusion of specific combat, combat support, and combat service related factors seems beyond dispute. AFP's exclusion of many probably related factors remains questionable. AFP's treatment of the included factors has not been proved correct by the preferred standards of scientific inquiry. On the other hand, its methods and results have been scrutinized in varying degrees by analysts of considerable and significant experience. No one has been fully satisfied. However, the collective and probably balanced opinion to date is that AFP treats a wider range of combat environments (16) including division ratios (4:1 to 1:3), postures (4), brightness (2), including division ratios (4:1 to 1:3) postures (4), brightness (2), and visibility (2) with more reasonable results in the large than other known efforts at estimation of static combat potential. The following subparagraphs list many of the AFP assumptions, guidelines, and axioms.

a. The usual techniques of dynamic modeling and simulation are not applied except to very small engagements over short time intervals.

b. Large-scale combat may be decomposed into separate firepower-counterfirepower, combat support, and combat service support processes. These processes may be analyzed largely independently, and their results may be combined largely afterward.

c. Gross direct fire division combat may be decomposed into pure weapon-type-on-weapon-type engagements. Such engagements may be further decomposed into smaller matchups in which at least one weapon opposes one or more weapons. Only indirect fire weapons may impinge on the interaction of otherwise pure type-on-type "duels."

d. In AFP work to date, higher echelon fires and support have been ignored. (Later generation AFP is to include echelons above division.)

e. Movement and maneuver are not represented within the Combat Module. Duels are distributed to fixed ranges. Opposing weapons that survive are assumed to remain at those ranges throughout an AFP "day." SSPKs are adjusted for moving targets. However, the difficulties of detecting aircraft within a lobed three-dimensional space are consistently underestimated. The AFP battle space remains essentially one-dimensional for all direct fire weapons.

f. An AFP combat day consists of four 2.01-minute intervals. An AFP combat run consists of 2 AFP combat days.

g. Direct fire weapons may not fire until normal detection occurs or until they have been fired upon a specified number of times by the opposing direct fire weapon(s).

h. A direct fire weapon which annihilates its opponents in a duel cannot shift its fire to similar opponents at the same range until the next duel. (A duel occupies one 2.01-minute interval.) A direct fire weapon may not shift its fire to another range or to another type opponent until the next AFP day. This scheme for holding and switching targets tends to mix the squared and linear law flavors of simplest Lancester theory. The extent of the mixing depends on the rates of fire and lethalities of rounds. Quickly terminating duels tend more toward linear flavor over successive duels. Duels terminating near 2.01 minutes or more tend more toward squared flavor in the long run.

i. Input engagement preferences and the opposing inventories at the beginning of AFP days determine the numbers of weapons of each type meeting enemy weapons of each type. Type-on-type duels are then forced to occur at very nearly those type-on-type odds. Hence, duels at much worse than average odds tend to be rare. The effect usually benefits relatively slow-firing weapons with relatively high SSPKs. Replication of the AFP Combat Module does not affect the tendency for duels to occur at very nearly average odds.

j. Weapons are assumed to replenish ammo between duels. The effect is negligible for weapons which do not expend much of their on-board ammo between during a duel. The assumption is to the advantage of one- or two-shot weapons, for example.

k. AFP combat potentials depend, among other things, on both estimated kills and target values. Estimated kills are an intermediate AFP result. Target values are an AFP input. To date it has not been practical to implement a method in which target values, too, are an AFP result.

l. AFP combat potentials depend on both kills and losses. Indeed, exchange ratios are key terms in the computation of potentials. Notice, that an exchange ratio is an estimate of how many kills may be achieved if the weapon type represented in the second part of the ratio is run down to zero survival. In that sense, the exchange ratio is a measure (usually extrapolated) of the lifetime killing potential of a weapon. There are obvious objections to measures of full lifetime potential. Few forces fight to their own total depletion. For many weapons, projection to full lifetime potential involves imprudent extrapolation to a distant future far beyond the limits of known results. AFP employs estimates of fractional lifetimes for most weapons. For those weapons, AFP potentials provide an estimate of kills achievable for loss of half of one's weapons. Indeed, the term half-life potential is sometimes employed within AFP. Depending on the weapon, half-life estimation may involve extrapolation or interpolation from results of the AFP Combat Module. It is important to note that different weapons will not reach their half lives at the same battlefield moment. For this reason, AFP potentials are sometimes called asynchronous potentials--i.e., estimates for different weapons at constant life fractions but different calendar time. Some potentials (for indirect fire and SAMs, for example) are estimated differently. Indirect fire potentials reflect estimated kills achievable within two weeks of combat. SAM kills represent estimates of kills achievable by the in-theater supply of missiles.

m. Within the AFP Combat Module, indirect fire weapons "fire" only during the standard 2.01 minute intervals. That means those weapons fire only 8.04 minutes per AFP day at very nearly intense rates. AFP makes a post-Combat Module adjustment to indirect fire results under the assumption that indirect fire weapons would have fired their normal daily loads and identical effects per round.

n. Final AFP combat potentials are weighted sums across 16 combat environments. A result from a single combat environment is called a partial combat potential. Weighted summation may produce the same final potential for divisions with different partial potentials. From the AFP potential point of view, the two divisions are equally "good" or "bad" on (weighted) average. Yet no one would want to use the divisions interchangeably, given any better prior knowledge of environment on one's own or the enemy's part. Here, "to use" involves not just battlefield choice but also choice in further combat analyses and force comparisons where combat environments differ markedly.

15. This section has been about AFP in general and is not a summary of this guide. The complete guide is primarily a collection of appendices and annexes for AFP system operators and programers. The contents are organized by major and minor module. Unclassified examples of input, output, runstreams, and programs are included. Production versions of some parts must necessarily differ from the examples shown in order to achieve secure processing. Examples shown may include obsolete user IDs; it was easier to retain the IDs current at documentation/example time than to issue new examples every time a user changed.

V. ORGANIZATION OF GUIDE

16. Application of the complete AFP System involves preparation of many different data sets specific to the System's different modules, preparation of module runstreams, execution of modules, and review of output. Some of the output of a module may become input to another module. Some modules may be executed many times during a "single" system application and within a single runstream. Finding particular information about these AFP matters within this guide to AFP may be difficult. Several reference aids are provided.

a. At the highest level, the guide is organized by module and submodule (with some extras). The tables of contents reflect this macrostructure.

b. Figure 7 is the official "road map" to the AFP System and its documentation. It goes a step beyond macrostructure. Figure 7 shows which parts of the documentation correspond to which parts of the AFP System. Similar figures appear at the beginnings of many appendices to help orient the reader. Figure 7 includes many clues to the names of programs, data sets, runstream generaors, runstreams, and reports. The clues are not described here, but they will gain in value as an operator or programer uses the guide.

c. Although each AFP module is different from the others, the descriptions of the modules follow a common pattern; hence, this guide's principal appendices and annexes are similarly organized in the following order: OVERVIEW, INPUT, OUTPUT, RUNSTREAM, and PROGRAM.

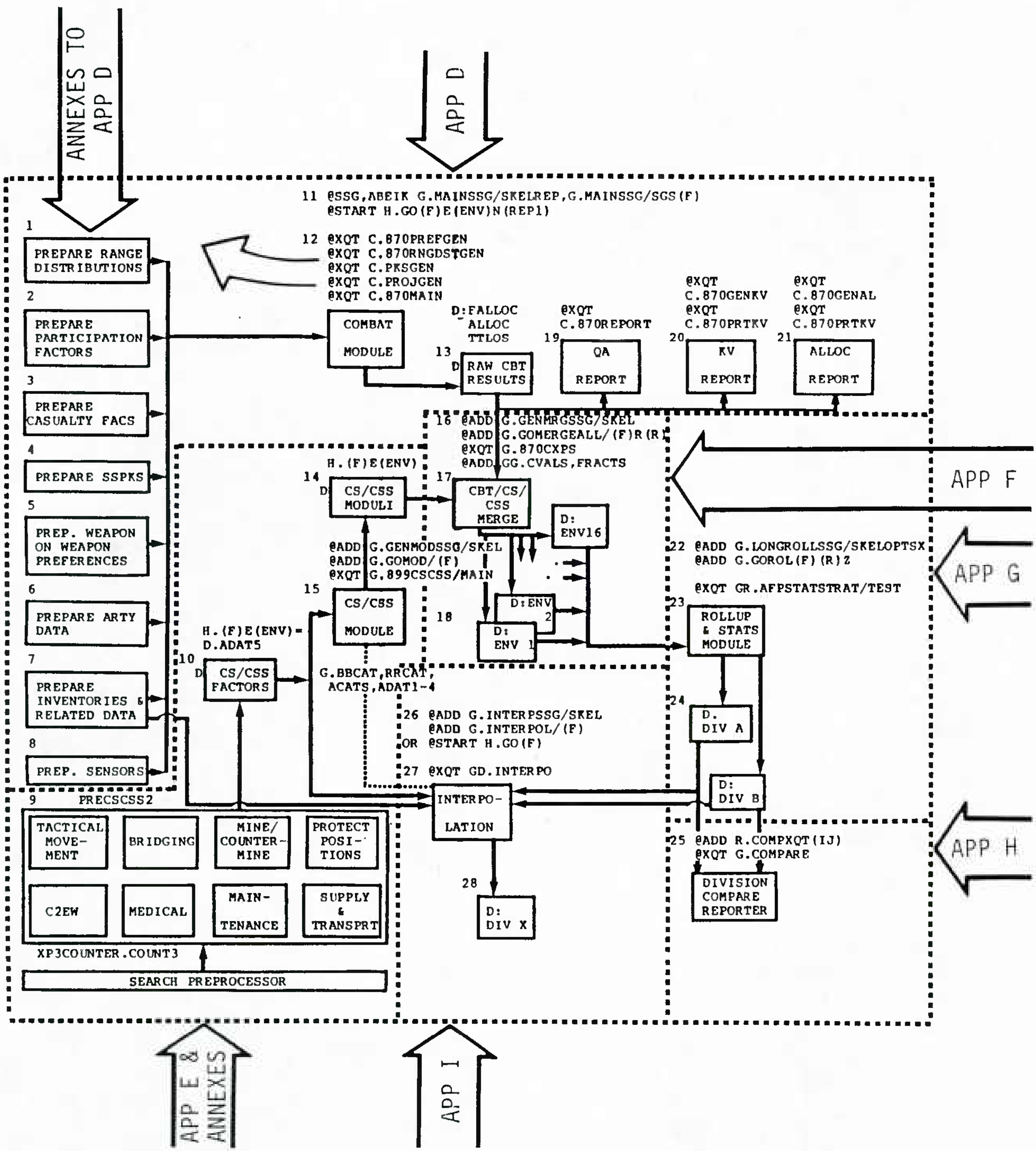


Figure 7. Key to Descriptions of Principal Data, Modules, Other Components, and Products of the Analysis of Force Potential (AFP) System

(1) Overview. A module (or submodule or special processor) overview describes a module's position to the AFP System in total. Some of the overviews include background information. For example, the overview for the Firepower/Counterfirepower Module (Combat Module) in Appendix D delves into a philosophy of combat representation and interpretation especially useful for AFP purposes.

(2) Input. A module's input data are described (apart from "input" to the modules's runstream generator) in more or less detail as appropriate. Examples of input records may be included. Input generated as the output of another module or process may be described more fully for the generating program than for the receiving program. Appendix C provides a key to the locations of preprocessor descriptions. Most of the AFP System's preprocessors involve preparation of input data. Hence, Appendix C is also a key to discusssions of much of AFP input. The central table of Appendix C is included here as Table 1. The input to the Combat Module is so extensive and involves so many preprocessors that the "input" paragraphs of Appendix D (on the Combat Module) are limited to a keyed guide to separate annexes containing the details of the corresponding data and preprocessors.

(3) Output. A module's output data are described (apart from output of the module's runstream generator) in more or less detail. In most cases, examples are provided. Some output are optional. Output may include intermediate or final results, diagonostic information, or simply copies for record. Appendix J provides a key to the locations of descriptions or examples of principal AFP output. The central table of Appendix J is reproduced here as Table 2.

(4) Runstream. Most of the AFP System's modules may be executed using runstreams producible by runstream generators written for the UNIVAC SSG processor. A generator, if it exists for a given module, is described in the runstream section. As a minimum, the SGS section of the generator is discussed. In some cases, the SKELeton section is also described. In ideal AFP production, only SGSs need be changed. In practice, many situations arise in which it is most convenient to modify the SKELeton section as well. In other words, the generators shown are not perfectly general and should be regarded as specific examples subject to change as necessary or convenient. The UNIVAC system editor is the obvious choice for making global changes of user IDs, for example. Most runstream descriptions include one or more examples of generated runstreams. Any example runstream is necessarily specific to some single run. Some runstream generators include an option to produce a runstream as an @ADD or an @START element. Single execution of some runstream generators may produce several runstreams for separate executions of one or more modules. Appendix K provides a key to the locations of descriptions and examples of runstream generators and generated runstreams. The central table of Appendix K is reproduced here as Table 3.

Table 1. Key to Preprocessor Descriptions

Preprocess(or)	Block # in Fig 7	Parent module	Location of main description or other related material
Prepare range distributions	1	Combat	Annex III to Appendix D
Prepare participation factors	2	Combat	Annex V to Appendix D
Prepare casualty factors	3	Combat	Annex IV to Appendix D
Prepare SSPKs	4	Combat	Annex VI to Appendix D
Prepare weapon-on-weapon preferences	5	Combat	Annex II to Appendix D
Prepare artillery data	6	Combat	Annex I to Appendix D Annex VIII to Appendix D
Prepare inventories and related data	7	Combat	Annex I to Appendix D
Prepare sensor data	8	Combat	Annex I to Appendix D
PREFGEN	12	Combat	Annex VII to Appendix D
RNGDSTGEN	12	Combat	Annex VII To Appendix D
PKSGEN	12	Combat	Annex VII to A ^ndix D
PROJGEN	12	Combat	Annex VII to Appendix D
PRECSCSS	9	CS/CSS	Annex I to Appendix E

Table 2. Key to AFP Output Record Copy and Report Examples and Descriptions (page 1 of 2 pages)

Record copy or report	Block # in Fig 7	Related process	Location of descriptions and/or examples
Basedata	6-8	Combat preproc	Annex I to Appendix D Annex VIII to Appendix D
Weapon on weapon preferences	5	Combat preproc	Annex II to Appendix D
Range distribution	1	Combat preproc	Annex III to Appendix D
Casualty factors	3	Combat preproc	Annex IV to Appendix D
Participation factors and engagement characteristics	2	Combat preproc	Appendix V to Appendix D
SSPKs	4	Combat preproc	Annex VI to Appendix D
Allocation "scoreboard"	21	Combat preproc	Appendix D, paragraph D-4c
Killer/victim scoreboard	20	Combat preproc	Appendix D, paragraph D-4d
Quality assurance report (QAREP)	19	Combat	Appendix D, paragraph D-4e
CS/CSS input	9	CS/CSS preproc	Annex I to Appendix E
CS/CSS factors	10	CS/CSS preproc	Annex I to Appendix E Annex II to Appendix E, Figure E-II-9
Special CS/CSS Module	15	CS/CSS Module	Annex II to Appendix E, Figures E-II-2-8
CS/CSS moduli	14	CS/CSS Module	Annex II to Appendix E, Figure E-II-10

Table 2. Key to AFP Output Record Copy and Report Examples and Descriptions (page 2 of 2 pages)

Record copy or report	Block # in Fig 7	Related process	Location of descriptions and/or examples
Special Merge Module input (CVALS and FRACTS)	16	CBT/CS/ CSS Merge Module	Appendix F, Figures F-3 and F-4
Raw combat report	17	CBT/CS/ CSS/Mearge Module	Appendix F, Figure F-5
Partial combat potentials	18	CBT/CS/ CSS/Merge Module	Appendix B, Figure B-7 Appendix F, Figure F-6
Final combat potentials	24	Rollup & Stats Module	Appendix B, Figure B-6 Appendix G, Figure G-3
Statistical reports	24	Rollup & Stats Module	Undocumented
Division comparison	25	Division Compare Reporter	Appendix H, Figure H-3
Interpolated final combat potentials	28	Interpo-lation Module	Appendix I, Figure I-3

Table 3. Key to AFP Runstream Generator Descriptions and Examples

Runstream	Block # in Fig 7	Related process	Location of descriptions and/or examples
Prepare combat module input (many)	1-8	Combat preproc	Annexes I-VI to Appendix D
Prepare CS/CSS input	9	CS/CSS preproc	Annex I to Appendix E
GENMRGSSG/SKEL	16	CS/CSS	Annex II to Appendix E
MAINSSG/SKELREP MAINSSG/SGS	11	Combat Module	Appendix D, paragraph D-4
GENMRGSSG/SKEL	16	CBT/CS/ CSS Merge	Appendix F, paragraph F-4
LONGROLLSSG/SKELOPTSX data	22	Rollup & Stats	Appendix G, Section IV
Example only	25	Division Compare Reporter	Appendix H, paragraph H-4
INTERPSSG/SKEL	26	Interpo-lation Module	Appendix I, Section IV

(5) **Program.** The final sections or paragraphs of an appendix or annex on a module or submodule are usually a description of the corresponding computer program in whole or in part. The description may include a general discussion, a logic flow diagram, source listing, definitions and indexing of principal arrays, line-by-line comments, and MAP element, as appropriate.

APPENDIX A

STUDY CONTRIBUTORS

1. STUDY TEAM

a. Study Directors

Mr. Gerald E. Cooper, Strategy, Concepts and Plans Directorate
LTC Thomas W. Lott, Requirements and Resources Directorate

b. Team Members

Dr. Steve Bravy
MAJ James L. Hubbard
MAJ William R. Milliron
MAJ Edgar E. Stanton, III
Mr. John W. Warren

2. TASK FORCE

a. Leader

COL John L. Rafferty

b. Task Force Members

MAJ Herbert C. Clifton
LTC Raymond K. Elderd
LTC Desmond W. Flanigan
MAJ John E. Justice
Ms Beverly M. Knox
MAJ Thomas B. Lingan
MAJ James R. Methered
CPT Jeffrey V. Rogers
Ms Cynthia Trimble
MAJ Tony Tyson
Mr. John W. Warren

3. PRODUCT REVIEW BOARD

Mr. Peter C. Byrne, Chairman
MAJ Thomas C. Wegleitner
MAJ Lester C. Roth

APPENDIX B

THE AFP SYSTEM

B-1. OVERVIEW

a. Figure B-1 presents the AFP system at the level of nondetail usually sufficient for executives and system users. At that level, the AFP approach may be viewed simply as a one-time decomposition of combat from combat support and combat service support, independent one-time processing within the corresponding modules, and finally a merge of CBT/CS/CSS results to yield equipment and organizational combat potentials. There is a clue in Figure B-1 to something more complicated. The term "ROLL UP" within the MERGE and ROLL UP Module implies that just maybe the results of several applications of the Combat Module (routinely, 16 distinct executions and sometimes as many as 160), and the CS/CSS Module (usually only four executions) must be combined to produce complete estimates of combat potential for a single friendly and a single threat division and their constituent equipment. Also from Figure B-1, it appears that an AFP "customer" need do little else than provide an equipment inventory, sit back for a while, and then receive results. It also appears that a threat inventory and sufficient information about item engagement characteristics are available off-the-shelf just as needed for easy AFP exploitation. Any such impressions are, of course, gross oversimplications. AFP operators and programers must face the grim reality of AFPSYS as it really is. That reality is presented in the following paragraphs and appendices of this volume. The presentation begins with a (very low) resolution view of Figure B-1.

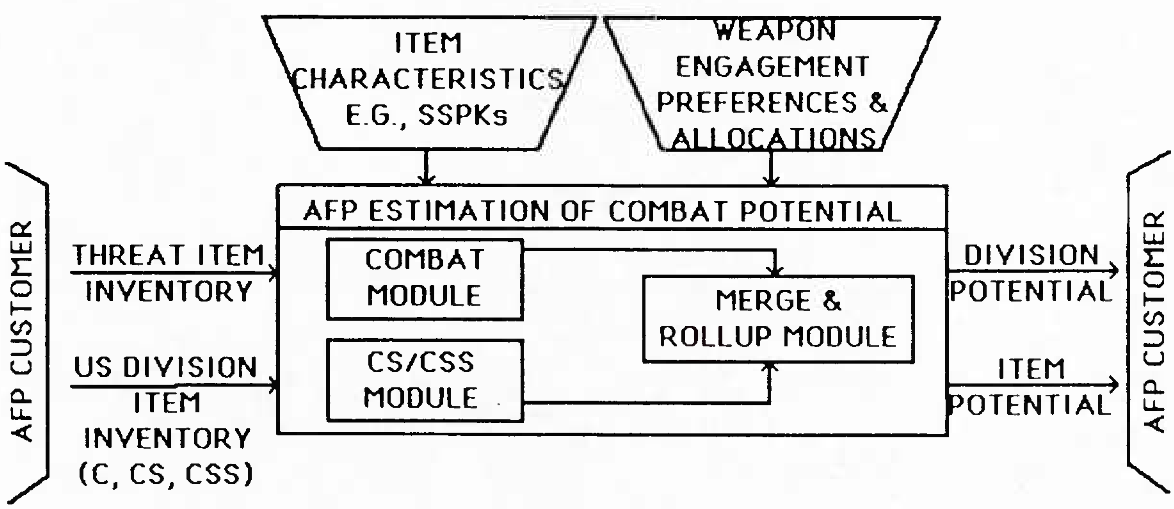


Figure B-1. A Simplified View of the Analysis of Force Potential (AFP) System and Its Relation to Customers, Data and Products

b. Figure B-2 represents the principal parts of the AFP System at higher resolution in a stylized, compact form. Major input, intermediate, and output data files are represented. Major program modules and some utility programs are also shown. In addition, @-symbols key special runstream generation steps and some of the principal runstream examples. Names of many principal absolute elements are included. The purpose of this appendix is to introduce Figure B-2 as a map of the AFP System. The following paragraphs of this appendix provide examples of the notation and conventions used and of the kinds of considerations the operator/programer should keep in mind throughout the remainder of the volume. The following appendices describe the parts of Figure B-2 in more detail.

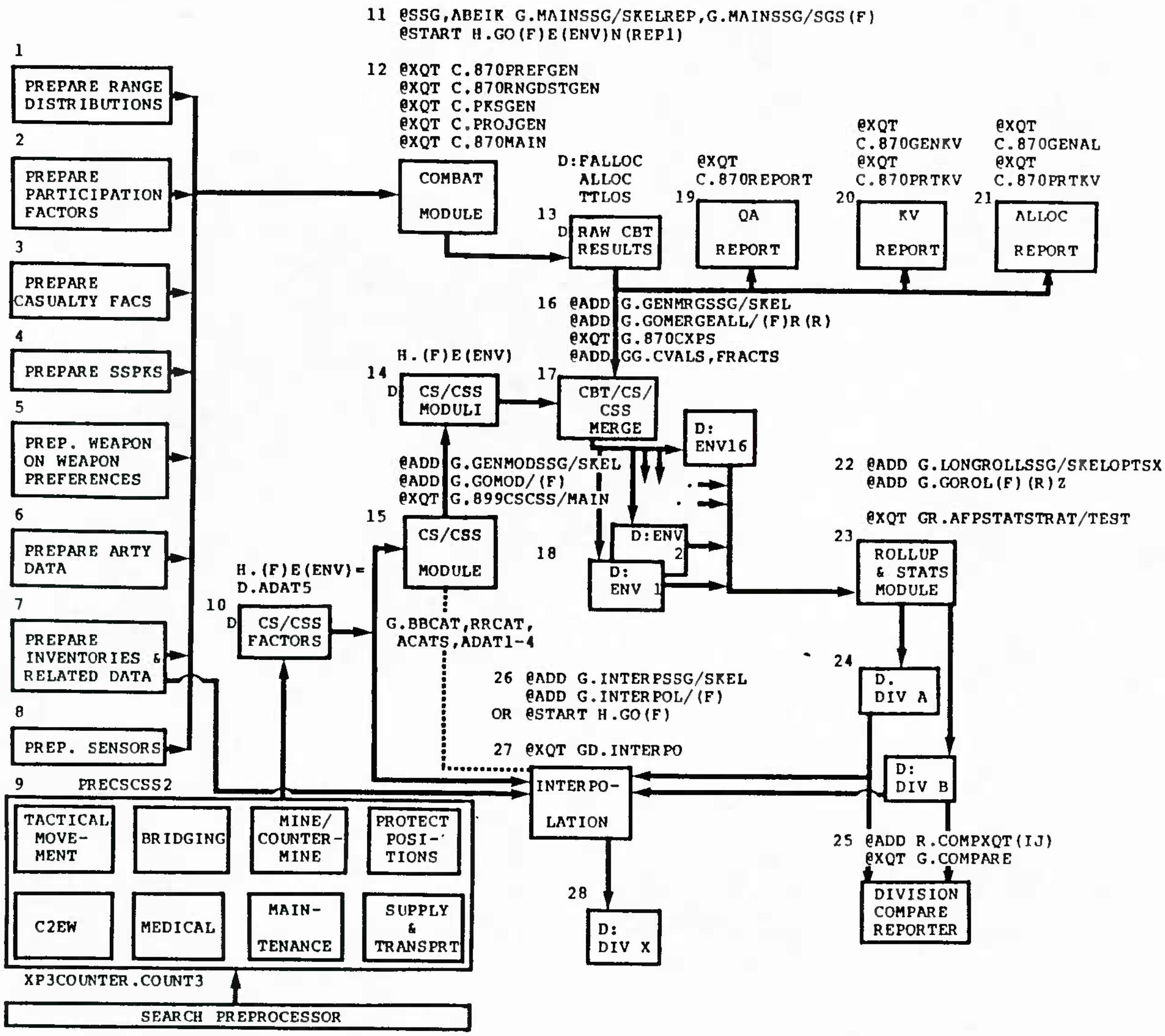


Figure B-2. The Relation Among Principal Data, Modules, Other Components, and Products of the Analysis of Force Potential (AFP) System

c. Figure B-2 does not portray a single, unambiguous system entry. Indeed, scrutiny of the diagram reveals a number of possible entry points. The ambiguity is intentional. Depending on the task, the system may have to be entered many times and perhaps at different points. The arrows in Figure B-2 make it appear that everything necessary may be accomplished in a "single pass." This is not the correct impression. For the sake of simplicity, this diagram has been kept free of the several "loops" usually required to make full-scale application of AFPSYS.

(1) Estimation of the combat potential of a single division requires that much of the AFP System be applied at least 16 times, once for each of the 16 combat environments treated by AFP. In order to achieve greater statistical significance within each environment and overall, the system operator may apply the Combat Module several times using different seeds for the pseudorandom number generator. For example, two seeds and, hence, two executions of the Combat Module, were routine for each environment through most of the system development and testing. In experimental production, 10 replications were performed for each combat environment for each of two stylized US divisions. Among other things, the 10-seed approach meant that the Combat Module (.870MAIN, block 12) and CBT/CS/CSS Merge Module (.870CXPS, block 17) were executed 160 times for each division considered. Once the 160 (16 environment x 10 seeds) ENV files had been produced for a division, the ROLLUP & STATS Module, block 23, was executed only once to produce the combat potentials and statistical analysis for that division, symbolized as DIV A (block 24) in Figure B-2.

(2) The AFP system is intended for estimation of the combat potentials of different divisions. The differences may involve unmodernized and modernized versions of the otherwise same division. The differences may involve alternative inventories of weapons at essentially the same level of technology. Also, of course, there may be interest in the differences between friendly and threat divisions or among US and allied divisions. In general, each different division analysis requires exercise of all or nearly all parts of the AFP System. Hence, interest in a new division, say the one symbolized as DIV B (block 24) in Figure B-2, implies generation of some but not necessarily all new data and execution of the Combat and Merge Modules 16 to 160 or more.

(3) At such time as two or more sets of division potentials have been developed (represented as D:DIV A and D:DIV B in Figure B-2, block 24), then the Division Compare Module (.COMPARE, block 25) may be executed to produce a comparative summary of the potentials of the equipment in two divisions.

(4) Full system execution with 16 (to hundreds of) applications of the Combat and Merge Modules remains more labor and computer intensive than desired. If two divisions have already been processed and if those divisions (again let us say they are DIV A and DIV B) are related as extremes on an inventory continuum, then the AFP Interpolation Module (.INTERPO, block 27) provides a means to approximate the combat potential of an "intermediate" division, say DIV X (block 28), in a few minutes instead of in several hours of computer time. Division combat potentials for an evolving division at five different times have been "interpolated" in roughly 10 minutes, wall time.

d. A runstream to produce results for a single combat environment from the Combat and Merge Modules may include 150 statements. Sixteen such runstreams involving critical differences from runstream to runstream are required to produce results for all 16 combat environments for a single division. Thus, 2,400 runstream control statements must all be correct to do just part of the work for a single division. Block 11 in Figure B-2 presents a highly abbreviated reference to one of the utility functions within AFPSYS. The first statement, @ADD,ABEIK..., invokes an SSG program that generates all runstreams for the appropriate number of random number seeds for a division. The second statement in block 11, @START..., is a generic representation of a statement to start the runs.

e. Blocks 1-8 provide highly oversimplified references to the seven broad categories of data required by the AFP Combat Module. Over all 16 AFP combat environments, blocks 1-8 may entail generation of hundreds of thousands of data elements.

B-2. AFP PRODUCT

a. Because the only purpose of AFP is to produce estimates of combat potential, the only AFP product of concern must be combat potentials. The remainder of this volume should be viewed in the perspective of producing the intended product. Toward that view, the rest of this appendix is devoted to a preview of the types and form of AFP products.

b. All AFP combat potentials provide estimates of the killing or casualty-producing capability of equipment or organizations relative to one or more measures of the resources lost or expended in attaining that capability. The resource lost may be a weapons platform, a quantity of munitions, or time. For any one weapon, the resource is just one of these types. For an organization, because it usually includes many types of weapons, the organization's potential probably involves all three types of resource expenditure.

c. AFP combat potentials are expressed in two ways:

(1) The first expression of combat potential is in terms of four components. This representation of combat potential is often called the four-valued or four-vector potential. On option, these components may include target value weights. If so, the "personnel" component then includes weighted other weapons (e.g., small arms).

(a) Personnel. The personnel component provides an estimate of personnel kills and casualties including dismounted troops and the crews or passengers of weapon- or troop-carrying platforms. In general, the kill or damage of a target contributes to the personnel component of potential depending on input factors that may vary by both target and shooter.

(b) Light Armored Vehicles. The second component of the four-valued potential provides an estimate of killed or damaged equipment, including the range of lightly armored vehicles from personnel carriers to self-propelled artillery and many air defense systems. Within the AFP System, mobility, firepower, and total kills are not differentiated.

(c) Heavy Armored Vehicles. The third component of the four-valued potential provides an estimate of killed or damaged tanks of all types. Mobility, firepower, and total kills are not differentiated.

(d) Aircraft. The fourth component of the four-valued combat potential provides an estimate of killed or damaged aircraft including both rotary and fixed wing. As for the second and third components, the types of kills are not differentiated.

(2) The second form of expression of combat potential is single-valued. It is often called the scalar combat potential. The scalar potential is a weighted sum of estimated kills and damage to target categories. Each of up to 60 target types may correspond to a distinct target category, but in practice, about a dozen categories seem sufficient. The weight (or "value") of a target is an input to AFP; it is not a result derived by the AFP System. Let us be clear about an important distinction. The values of targets are input. The combat potentials of shooters are derived by the AFP System but do depend on the input values of targets. Combat potentials are, in effect, values of shooters. A possible source of confusion is that most shooters can also be targets. Hence, a weapon may have one value as a shooter and a different value as a target.

(3) Both the four-valued and scalar forms of combat potential are frequently presented within a single computer or printed record. Because both forms so often appear side by side, it has become common to refer to a five-valued or five-vector combat potential. However, because this involves nothing more than appending the scalar form to the four-valued form, there is no need to define a third means of expressing combat potential.

d. AFP combat potentials are expressed at two levels:

(1) The potentials of a weapon type may be expressed as combat scores or combat item potentials. The latter are often referred to as CIPs.

(a) A weapon score is the sum of the potentials of all the weapons of that type within a division.

(b) The potential of a single weapon of given type is the CIP of that weapon.

(c) If there are N weapons of a given type within a division,

$$SCORE = N * CIP$$

(d) The CIP of a weapon is the same for all weapons of that type within the division. In other words, the CIP is a mean potential for all weapons of that type within the division.

(2) The potentials of an entire organization (division) are referred to as combat organizational potentials or COPs. A COP is simply the sum of the scores of all weapons within the organization.

e. Taken strictly, the terms "scores," "CIPs," and "COPs" imply that combat support and combat service support effects have been accounted by the process called CS/CSS modulation. However, because quantities in the format of scores, CIPs, and COPs are available at an intermediate stage before CS/CSS modulation, it has become common practice to save and report such intermediate values along with the final ones. The terms "unmodulated" and "modulated" have been prefixed to scores, CIPs, and COPs in order to provide the corresponding, necessary distinctions.

f. Also in strict terms, "scores," "CIPs," and "COPs" refer to combat potentials weighted over 16 distinct combat environments. However, because numbers in exactly the same formats appear for each combat environment before the weighted summation over all environments is performed, it has become common practice to refer to the results for a single environment as scores, CIPs, and COPs, too.

g. The foregoing few paragraphs make it clear that AFP provides many opportunities for confusion in terminology and among computer and printed records of "potential." To avoid ambiguity, AFP assigns numeric identifiers to combat potential records. An identifier appears in the first field of each combat potential record. Figure B-3 provides the keys to safe recognition of all related records. Note that separate identifiers are provided for friendly and threat records. Note too that there is no need (or use) for records at divisional level that simply sum CIPs. At divisional level, only sums of scores are significant.

h. With the preceding several paragraphs as preparation, the reader should now be prepared for a first look at an example of AFP output at the very end of the AFP process, that is, following a rollup across all 16 combat environments for a single division and its opposing threat. Figure B-4 includes two ovals showing where such results emerge in relation to the total system. The ovals enclose three possible data files. D:DIV A and D:DIV B emerge as the result of a full system application. D:DIV X emerges at the point where interpolation is possible only because two previously generated files (A and B) "bound" the inventory of DIV X. The next paragraph presents extracts from an example file that could be any one of D:DIV A, D:DIV B, or D:DIV X. However, the results of interpolation do not include Red side potentials.

i. Figure B-5 displays records extracted from a sample output file of AFP combat potential for a notional division inventory and the opposing threat division inventory. Only a subset of the 192 records of the complete file is shown in Figure B-5. In the example, the first four components of potential do not include target values (other than 1.0).

	AFP OUTPUT RECORD TYPE IDENTIFIERS (FIELD 1=ISCNT)			
	FOR EACH VISIBILITY, POSTURE,DAY/NITE		OVER ALL VISIBILITIES, POSTURES,DAY/NITE	
INFORMATION TYPE	BY WPN	COP	BY WPN	COP
UNMODULATED BLUE SCORE	10	11	110	111
UNMODULATED RED SCORE	20	21	120	121
UNMODULATED BLUE CIP	30		130	
UNMODULATED RED CIP	40		140	
MODULATED BLUE SCORE	50	51	150	151
MODULATED RED SCORE	60	61	160	161
MODULATED BLUE CIP	70		170	
MODULATED RED CIP	80		180	

Figure B-3. Identifiers of the Principal Type Records (and their contents) for Partial and Final Combat Potential Output of the AFP System

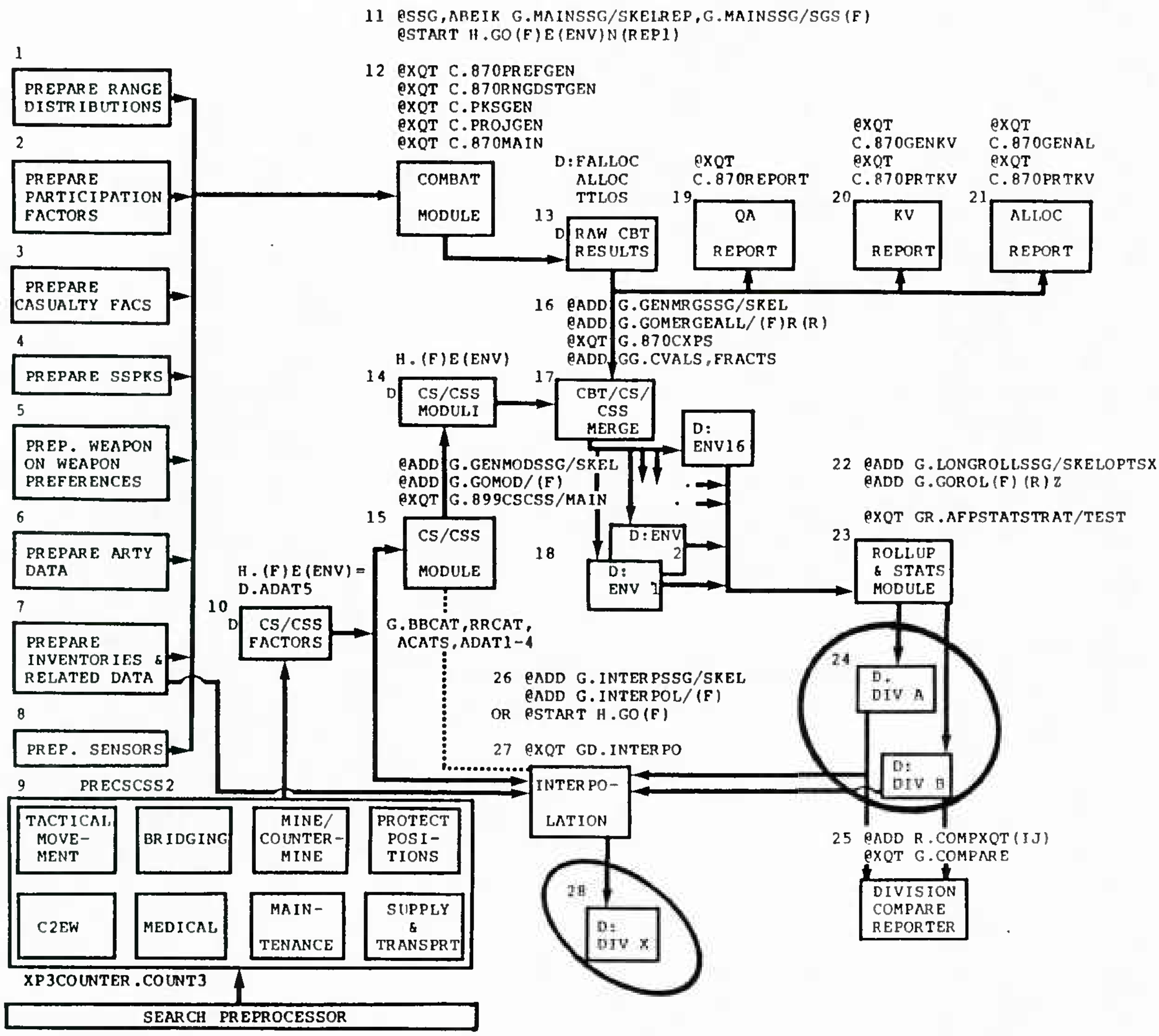


Figure B-4. Points Within the AFP System at Which Final Combat Potentials Are Produced

FIELD

	1	2	3	4	5	6	7	8	9
25.	110	E 1 0 0 0	16	991.096	67.506	90.345	.000	21.586	1 1 1
26.	130	E 1 0 0 0	16	4.737	.327	.436	.000	.104	1 1 1
27.	150	E 1 0 0 0	16	990.053	67.287	90.312	.000	21.555	1 1 1
28.	170	E 1 0 0 0	16	4.779	.325	.435	.000	.104	1 1 1
29.	110	E 1 0 0 0	17	524.247	35.921	45.623	.000	11.156	1 1 1
30.	130	E 1 0 0 0	17	4.628	.318	.400	.000	.098	1 1 1
31.	150	E 1 0 0 0	17	523.781	35.855	45.722	.000	11.161	1 1 1
32.	170	E 1 0 0 0	17	4.621	.317	.401	.000	.098	1 1 1
33.	110	E 1 0 0 0	20	2669.735	157.594	180.532	3.791	51.092	1 1 1
34.	130	E 1 0 0 0	20	7.992	.473	.538	.011	.153	1 1 1
35.	150	E 1 0 0 0	20	2673.525	157.388	180.920	4.131	51.470	1 1 1
36.	170	E 1 0 0 0	20	8.000	.472	.539	.012	.154	1 1 1
37.	110	E 1 0 0 0	26	105.556	4.346	.000	8.111	8.841	1 1 1
38.	130	E 1 0 0 0	26	3.175	.131	.000	.244	.265	1 1 1
39.	150	E 1 0 0 0	26	107.007	4.320	.000	8.807	9.538	1 1 1
40.	170	E 1 0 0 0	26	3.218	.130	.000	.264	.286	1 1 1
					*				
					*				
					*				
169.	120	E 1 0 0 0	51	3.160	.074	.000	.000	.016	1 1 1
170.	140	E 1 0 0 0	51	.129	.003	.000	.000	.001	1 1 1
171.	160	E 1 0 0 0	51	3.329	.079	.000	.000	.017	1 1 1
172.	180	E 1 0 0 0	51	.136	.003	.000	.000	.001	1 1 1
173.	120	E 1 0 0 0	52	45.216	7.383	.013	.000	.902	1 1 1
174.	140	E 1 0 0 0	52	.741	.121	.000	.000	.015	1 1 1
175.	160	E 1 0 0 0	52	47.456	7.748	.013	.000	.947	1 1 1
176.	180	E 1 0 0 0	52	.778	.127	.000	.000	.015	1 1 1
177.	120	E 1 0 0 0	56	19.989	2.611	.047	.000	.335	1 1 1
178.	140	E 1 0 0 0	56	.169	.022	.000	.000	.003	1 1 1
179.	160	E 1 0 0 0	56	20.952	2.749	.049	.000	.352	1 1 1
180.	180	E 1 0 0 0	56	.178	.023	.000	.000	.003	1 1 1
					*				
					*				
					*				
189.	111	E 1 0 0 0	0	11916.452	897.693	489.232	103.601	293.738	1 1 1
190.	151	E 1 0 0 0	0	11919.645	894.350	488.403	112.592	302.274	1 1 1
191.	121	E 1 0 0 0	0	898.871	184.892	12.760	89.643	112.274	1 1 1
192.	161	E 1 0 0 0	0	975.283	193.200	13.275	106.670	131.446	1 1 1

Figure B-5. Example Extract Records from Sample File of Final Combat Potentials Produced by the AFP System

(1) Field 1 contains the identifiers corresponding to the keys tabulated in Figure B-3. Note that the file must be a representation of potential relative to all 16 combat environments because the identifiers are greater than 100.

(2) Records 25 through 40 present potentials for four Blue or friendly weapons. Records 169 through 180 present potentials for three Red or threat weapons. There are four records for each weapon. Records 189 through 192 present the COPs for the Blue and Red divisions; respectively.

(3) The subfields of fields 2 and 9 may be filled with blanks or zeros. These subfields provide the means to include special identifiers that may be applied but are not needed if files are carefully named in the first place.

(4) Field 3 contains a number corresponding to a specific weapon type. Although it is not apparent from Figure B-5, Blue and Red weapons may have the same number. However, because the identifiers in field 1 uniquely specify Blue and Red records, there is no ambiguity with respect to weapon type.

(5) Field 4 contains the first or personnel component of the four-valued potential for each weapon or division.

(6) Field 5 contains the second or light armored vehicle component of the four-valued potential for each weapon or division.

(7) Field 6 contains the third or heavy armored vehicle component of the four-valued potential for each weapon or division.

(8) Field 7 contains the fourth or aircraft component of the four-valued potential for each weapon or division.

(9) The literal interpretation of the values in fields 4 through 7 is as estimates of kills, casualties, or damage given the expenditure of the corresponding resource (half-life, basic load, or fortnight).

(10) Field 8 contains the scalar combat potential for each weapon or division.

(11) As noted earlier, because the four-valued and scalar potentials appear in sequence, fields 4 through 8 are considered together as a five-valued or five-vector representation of combat potential.

j. The next example of an AFP product is an intermediate result. Figure B-6 contains an oval enclosing symbols representing similar files for up to 16 distinct combat environments. If the Combat Module is exercised M times for each random number seed, then M * 16 files would be enclosed in the oval within Figure B-6. Figure B-7 presents extracted records from just one of the 16 (or M * 16) example files so symbolized. Note that Figure B-7 is very much like Figure B-5. The format is the same. The values in fields 4 through 8 differ from those in Figure B-5. Such differences reflect dependence on combat environment and on the uncertainties underlying the stochastic steps in the Combat Module. And of course, the entries in field 1 differ from those in Figure B-5. Figure B-7 represents a single combat environment; therefore, all the identifiers are less than 100. Note that entries in the subfields of fields 2 and 9 are usually not blank or zero. The nonblank, nonzero characters may identify the specific posture, visibility, and day/night condition to which the file corresponds. In Figure B-7, the first four components of partial potentials do not include target value weights (other than implicit 1.0s); however, an AFP option permits weighting of those entries by target values.

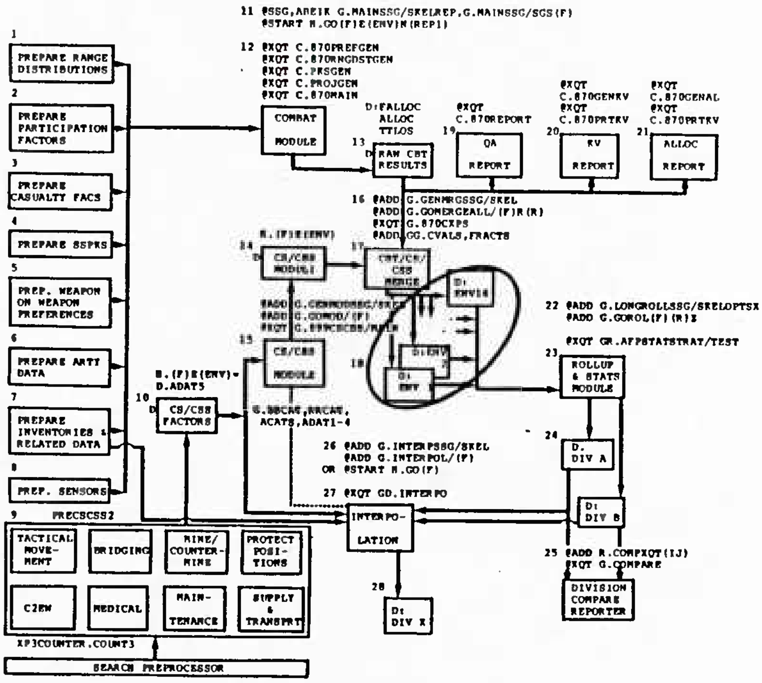


Figure B-6. Point Within the AFP System at Which Partial Combat Potentials Are Produced

FIELD

	1	2	3	4	5	6	7	8	9
17.	10	E 1 1 1 1	16	1191.458	84.925	113.229	.000	26.959	1 1 1
18.	30	E 1 1 1 1	16	5.784	.412	.550	.000	.131	1 1 1
19.	50	E 1 1 1 1	16	1197.489	85.355	113.802	.000	27.096	1 1 1
20.	70	E 1 1 1 1	16	5.813	.414	.552	.000	.132	1 1 1
21.	10	E 1 1 1 1	17	644.042	47.542	48.500	.000	13.075	1 1 1
22.	30	E 1 1 1 1	17	5.776	.426	.435	.000	.117	1 1 1
23.	50	E 1 1 1 1	17	647.302	47.782	48.746	.000	13.141	1 1 1
24.	70	E 1 1 1 1	17	5.805	.429	.437	.000	.118	1 1 1
25.	10	E 1 1 1 1	20	3252.010	234.271	345.906	5.000	83.696	1 1 1
26.	30	E 1 1 1 1	20	9.855	.710	1.048	.015	.254	1 1 1
27.	50	E 1 1 1 1	20	3272.702	235.457	347.657	5.448	84.554	1 1 1
28.	70	E 1 1 1 1	20	9.917	.714	1.054	.017	.256	1 1 1
29.	10	E 1 1 1 1	26	118.500	6.625	.000	8.000	9.006	1 1 1
30.	30	E 1 1 1 1	26	3.703	.207	.000	.250	.281	1 1 1
31.	50	E 1 1 1 1	26	120.453	6.659	.000	8.717	9.732	1 1 1
32.	70	E 1 1 1 1	26	3.764	.208	.000	.272	.304	1 1 1
					•				
					•				
					•				
157.	20	E 1 1 1 1	51	1.706	.000	.000	.000	.004	1 1 1
158.	40	E 1 1 1 1	51	.059	.000	.000	.000	.000	1 1 1
159.	60	E 1 1 1 1	51	1.780	.000	.000	.000	.005	1 1 1
160.	80	E 1 1 1 1	51	.061	.000	.000	.000	.000	1 1 1
161.	20	E 1 1 1 1	52	51.161	6.395	.000	.000	.810	1 1 1
162.	40	E 1 1 1 1	52	.839	.105	.000	.000	.013	1 1 1
163.	60	E 1 1 1 1	52	53.377	6.672	.000	.000	.845	1 1 1
164.	80	E 1 1 1 1	52	.875	.109	.000	.000	.014	1 1 1
165.	20	E 1 1 1 1	56	21.396	1.297	.000	.000	.192	1 1 1
166.	40	E 1 1 1 1	56	.181	.011	.000	.000	.002	1 1 1
167.	60	E 1 1 1 1	56	22.322	1.353	.000	.000	.201	1 1 1
168.	80	E 1 1 1 1	56	.189	.011	.000	.000	.002	1 1 1
					•				
					•				
					•				
177.	11	E 1 1 1 1	0	16413.689	1193.812	704.285	112.000	373.358	1 1 1
178.	51	E 1 1 1 1	0	16494.709	1197.939	705.834	122.042	384.262	1 1 1
179.	21	E 1 1 1 1	0	1052.358	180.632	13.746	125.153	148.850	1 1 1
180.	61	E 1 1 1 1	0	1144.605	187.630	14.106	150.793	175.517	1 1 1

Figure B-7. Example Extracts Records from Sample File of Partial Combat Potentials Produced by the AFP System

APPENDIX C

KEY TO AFP PREPROCESSOR DESCRIPTIONS

C-1. OVERVIEW. The AFP System consists of many processes: computer programs, runstream generators, runstreams, and input, intermediate, and output data. Among all processes and programs, AFP draws somewhat arbitrary distinctions between major and minor modules. This short appendix provides a key to those minor modules considered preprocessors. In AFP terms, a preprocessor does not identify something preliminary to AFP. Instead, a preprocessor is part of AFP but involves something preliminary to a major module. Figure C-1 presents the standard view of the AFP System as displayed in many other appendices of this document. The major modules are labeled:

a. Combat Module

b. CS/CSS Module

c. CBT/CS/CSS Merge Module

d. Rollup and Stats Module

e. Interpolation

C-2. PREPROCESSOR KEY. Table C-1 lists the AFP preprocessors and the locations of the principal corresponding descriptive material within this documentation. Many descriptions include material on the related data, both input and output, of preprocessors. Any one preprocessor may consist of more than one computer program. Runstream generators have not been included among the AFP preprocessors. The AFP System's runstream generators are separately keyed in Appendix K.

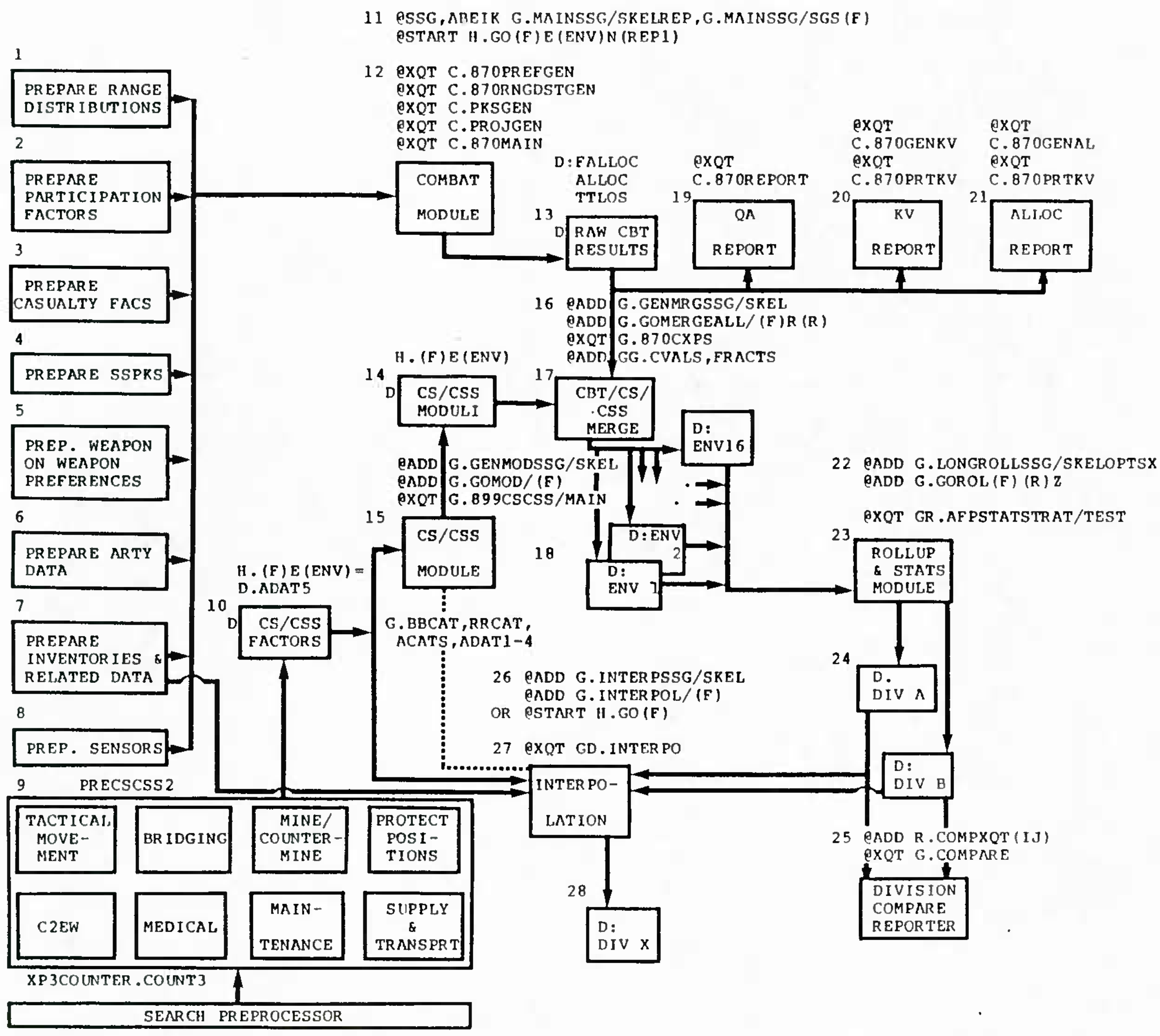


Figure C-1. The Relation Among Principal Data, Modules, Other Components, and Products of the Analysis of Force Potential (AFP) System

Table C-1. Key to Preprocessor Descriptions

Preprocess(or)	Block # in Fig C-1	Parent module	Location of main description or other related material
Prepare range distributions	1	Combat	Annex III to Appendix D
Prepare participation factors	2	Combat	Annex V to Appendix D
Prepare casualty factors	3	Combat	Annex IV to Appendix D
Prepare SSPKs	4	Combat	Annex VI to Appendix D
Prepare weapon on weapon preferences	5	Combat	Annex II to Appendix D
Prepare artillery data	6	Combat	Annex I to Appendix D Annex VIII to Appendix D
Prepare inventories and related data	7	Combat	Annex I to Appendix D
Prepare sensor data	8	Combat	Annex I to Appendix D
PREFGEN	12	Combat	Annex VII to Appendix D
RNGDSTGEN	12	Combat	Annex VII to Appendix D
PKSGEN	12	Combat	Annex VII to Appendix D
PROJGEN	12	Combat	Annex VII to Appendix D
PRECSCSS	9	CS/CSS	Annex I to Appendix E

APPENDIX D

AFP FIREPOWER AND COUNTERFIREPOWER MODULE

D-1. OVERVIEW

a. Purpose. This appendix and its annexes are intended to help clarify many of the concepts applied within the AFP System, especially in the AFP Firepower and Counterfirepower Module. Although the AFP Firepower and Counterfirepower Module represents little more than detection and the exchange of fire, the module, for the sake of brevity, is usually labeled the "Combat Module." The relation of the AFP Combat Module, and its pre- and postprocessors, to the AFP System in general is portrayed in Figure D-1.

(1) The AFP Combat Module is a computer program for estimating kills achievable by the weapons of two opposing sides under very special circumstances. By the usual definitions, the Combat Module is not a combat model or a combat simulation. However, the Combat Module does involve the application of some data and techniques of the kinds typically applied in combat modeling and simulation. To a modest degree, the Combat Module does represent many of the aspects of combat addressed in most combat models and simulations. Final products of the AFP System are measures of the static combat potentials of equipment and units. Although heavily "static" in its emphasis, the AFP System in general, and the Combat Module in particular, do not completely disregard battlefield time. The AFP treatment of time is much less sophisticated than would be required if the AFP System were to produce "dynamic" measures of combat potential.

(2) The AFP approach decomposes the military battlefield in several ways. Detection and firing are separated from other combat and the combat support and combat service support (CS/CSS) operations and functions. This separation led to the development of the separate Combat Module and CS/CSS Module within the AFP System. Such separation forced development of a third module, the CBT/CS/CSS Merge Module, to then combine the results of the Combat and CS/CSS Modules. Detection and firing were further decomposed in several ways leading to a special hierarchical representation. Many of the steps and elements involved in and resulting from this process have been given their own names and definitions. The names are old words with new meanings. The old words were chosen to be suggestive, but inevitably, some of the names suggest too much and others suggest too little. Terms involving or implying time tend to be particularly awkward because of the primarily "static" nature of AFP.

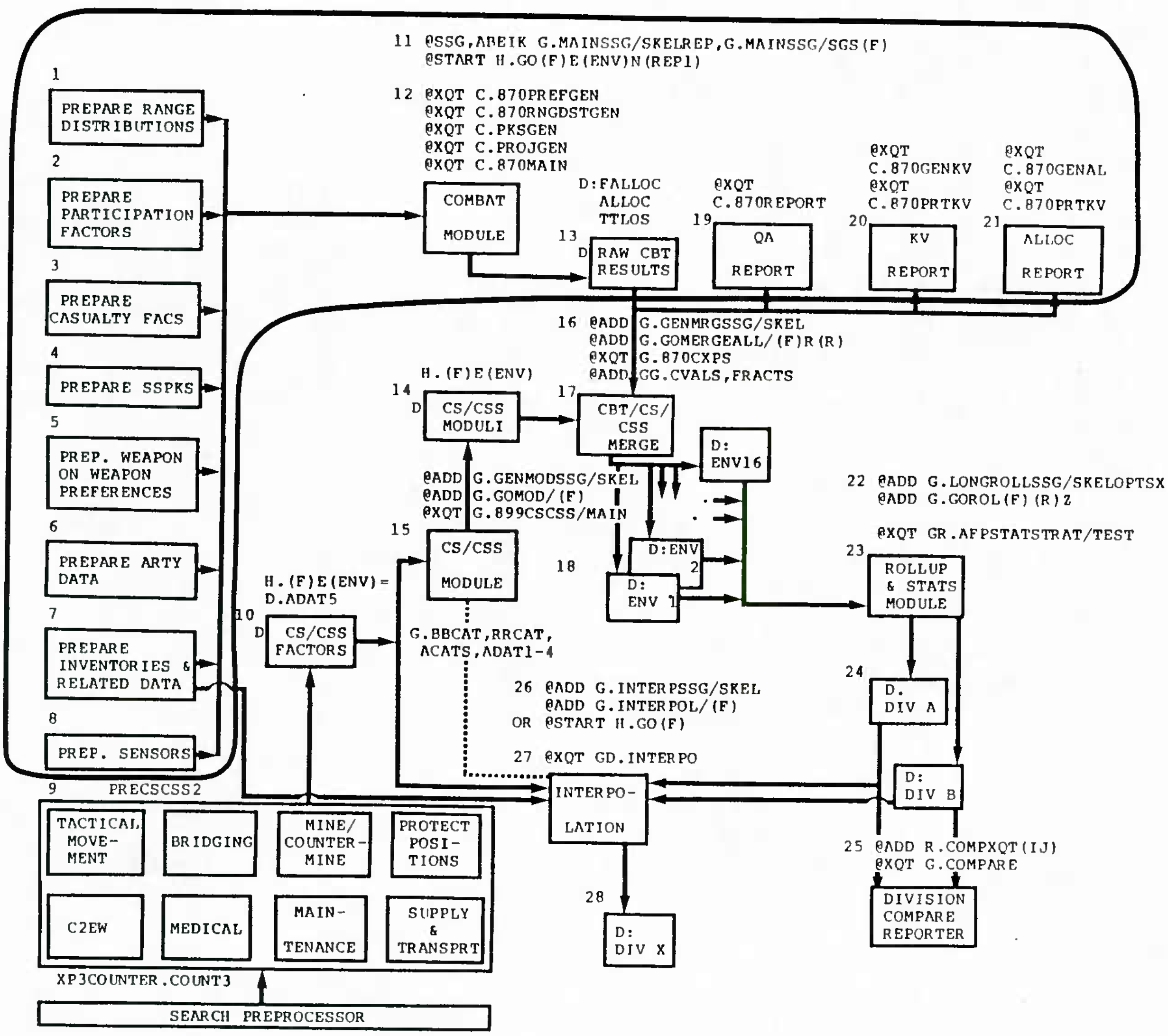


Figure D-1. The Relation of the AFP Firepower and Counterfirepower Module and Its Pre- and Postprocessors to the AFP System in General

(3) The developer of a combat model, a combat simulation, or an AFP System has to figure how best to carve up or decompose the battlefield whole into logically separable parts. Part of the decomposition usually involves invention of a scheme of taxonomy or categorization. Because of many kinds of limitations, the developer usually accepts fewer than the "ideal" number of categories and then struggles to fit enormous variety into too few categories. The combat analyst's art is very much involved with choosing necessary and sufficient categories. The categories must balance the often conflicting demands of specificity and generality. And the assignments of particular entities must satisfy everyone as reasonable--no entities fall between categories and no entity belongs in two or more categories at the same time--everything maps uniquely and correctly.

b. **Detection and Firing Engagements.** The Combat Module must permit weapons to engage one another as both shooters and targets. There are only two ways in which weapons may engage targets within the Combat Module: as direct firers or as indirect firers. The basic engagement is a direct fire engagement. A direct fire engagement lasts an AFP "day."

(1) A direct fire engagement is one in which one or more direct fire weapons of a single given type engage one or more opposing direct fire weapons of a single given type. In AFP terms, if weapons of each of 10 different types engage opposing weapons of each of 10 different types on a given day, then 10 x 10 = 100 distinct engagements would be generated. In usual AFP practice, something less than 100 engagements would occur because of a type-on-type preference and allocation scheme that may preclude some pairings of weapon types. There may be too few weapons of some types to generate a full set of nonzero engagements. Although preferencing, participation, and allocation steps precede engagements in the Combat Module sequence, these processes are not described until after many of the features of engagements have been introduced. Even though the Combat Module must allocate specific numbers of weapons to engagements before the engagements begin, it is the author's choice to describe engagements first as though weapons allocation has already occurred. This is partly to emphasize first those aspects of the Combat Module closest to the killing of targets--the only way that weapons can earn combat potential. This emphasis helps demonstrate the importance of the earlier allocation process. The allocation of too many superior weapons to an engagement can "deny" those weapons full opportunity to earn potential if the weapons run out of targets early. The allocation of too few weapons to an engagement may cause them to be destroyed largely as the result of a locally adverse force ratio. Even though the formal allocation process is described much later, the reader is invited to begin to think about the two-sided problem of balancing weapon allocation among possibly hundreds of different weapon-type-on-weapon-type engagements.

(a) The case of 10 TOWs and 15 T-62 tanks firing at one another comprise a permissible engagement. It is an ordinary direct fire engagement in that each weapon must detect a target before firing.

(b) Two M-1 tanks firing at one unarmed command vehicle also define a permissible engagement, even though that command vehicle cannot be a shooter. Every "direct fire" platform is a target, even though it may be a weaponless platform. Nevertheless, AFP still refers to such platforms as direct fire weapons because they are subject to the normal direct fire processing logic of the Combat Module. Again, all AFP direct fire weapons are always targets but are not always shooters.

(c) Six (Blue) 155mm howitzers and six (Red) 122mm howitzers firing at one another are also a permissible "direct fire" engagement--though of a special kind. Counterbattery fire is considered direct fire in AFP terms. A counterbattery engagement is special in that a weapon does not "detect" before firing. It is assumed that the firing mission has been defined and given beforehand, i.e., it is assumed that detection occurred sometime before the engagement begins. Weapons opposing one another in the counterbattery version of direct fire engagements are both shooters and targets.

(d) The case of 10 TOWs and 10 T-62 and 5 T-64 tanks firing at one another cannot be a permissible single AFP engagement because the Red side consists of two different weapon types. "10 - X" TOWs versus 10 T-62s and "X" TOWs versus 5 T-64s are permissible as two separate direct fire engagements.

(e) Direct fire weapons, then, may be any of the usual small arms, free AT rockets, ATGMs, tanks, IFVs/CFVs, aircraft, CLGP, fire-and-forget rockets, tube AAA, SAMs, and counterbattery assigned artillery.

(f) Direct fire engagements usually are complicated by having "indirect fire" fall upon them. The mortars and artillery (both tube and rocket) not engaged in direct fire (counterbattery) missions may fire on the full range of direct fire engagements including counterbattery direct fire engagements. If the Blue side possesses 10 types of indirect fire weapons and the Red side also possesses 10 types of indirect fire weapons, then 20 different types of indirect fire weapons may all fire on a single direct fire engagement in which only two types of direct fire weapons are permitted to fire. In this extreme case, the Combat Module manages the fire of 22 weapon types in a single "direct fire" engagement. The direct fire weapons shoot only at one another; they do not fire at the indirect fire weapons. The indirect fire weapons are totally "immune" in this role; they can kill but cannot be killed. This concept, permitting indirect fire weapons to fire upon the direct fire engagements, extends to the counterbattery direct fire engagements. In this special case, the extreme permits (for the above example) only a maximum of 20 weapon types to fire. However, two of those types may be firing in two senses--as counterbattery and as indirect weapons. Any one weapon, of course, cannot be both a counterbattery and indirect firer in that engagement. Some weapons of its type fire exclusively in the counterbattery role; some other weapons of that type fire exclusively in the indirect role. Both the counterbattery and indirect fire weapons may kill the counterbattery weapons; but, as usual, no one can kill the indirect fire weapons (until perhaps, on another "day," some of them become assigned counterbattery roles). The weapons in the counterbattery role are both shooters and targets in the counterbattery version of direct fire engagement. The weapons (which may be of the same types as those in the counterbattery role) in the indirect fire role are shooters but not targets in the counterbattery (as well as the normal) version of direct fire engagements.

(2) The direct fire engagement is not the smallest combat action represented within the AFP Combat Module. The next smaller action is called a "conflict." And the action next smaller than a conflict is called a "duel." An AFP duel is the smallest combat action at which direct fire is represented. In a sense, a duel is the basic building block of AFP combat representation. AFP may take liberties with the term "duel." The standard definition limits fighting to two persons or parties. AFP duels may generalize the classic one-on-one encounter all the way to a 50-on-1 extreme. (That extreme is an alterable program parameter.) In review, an engagement may consist of one or more conflicts; and a conflict may consist of one or more duels. In the AFP scheme of things, engagements are decomposed into duels, duels are distributed by range and environment, and all the duels at a given range in a given environment are combined as a conflict. Conflicts and duels are described at greater length in the following paragraphs. A conflict cannot last longer than an input-specified maximum time (2.01 minutes is the standard maximum). The survivors of a conflict may fight one another again in another conflict on that day at the same range in the same environment. Provided that there continue to be survivors, conflicts can continue on that day to an input-specified maximum number (four conflicts is the standard maximum). Survivors may be assigned to different opponents, ranges, or environments only at the beginning of a following "day."

(3) The weapons allocated to a direct fire engagement may not all find their way into duels. Some weapons may be lost to other (nonengagement) causes as "external losses" imposed in accord with input-specified factors. Weapons not lost may be diverted or delayed in accord with input-specified participation factors. With allowance for external losses and nonparticipation, suppose net quantities of m Blue and n Red weapons (each of single type) assigned to an engagement are available for assignment to duels. The m and n weapons are grouped by the Combat Module into min(m,n) distinct duels. This is an AFP rule not an international rule of engagement. That is, there are as many duels as there are weapons on the less numerous side. In AFP terms, in an S:1 duel (S weapons opposing one weapon), "S" is called the "odds class" of the duel. Inasmuch as there are to be as many duels as there are weapons on the less numerous side, the less numerous side provides the "1" in every S:1 odds class. For any pair of numbers, m and n, there need be no more than two odds classes.

(a) Let $q = INT(max(m,n)/min(m,n))$

$r = max(m,n) - q \times min(m,n)$

q and r are both integers.

(b) There need be, at most, the two odds classes: q and (q+1).

(c) The number of duels at q:1 is (min(m,n) - r).

The number of duels at (q+1):1 is r.

Clearly, the second of these numbers may be zero.

(4) Once the number of duels by odds class has been determined, the duels must be distributed by range and environment. In all work to date, only one environment per Combat Module run has been represented. Each combat environment of interest forced a separate Combat Module run. For the rest of this description, it is assumed that there is always only one combat environment per Combat Module run. Then the duel distribution problem reduces to one of distribution to six ranges. The standard AFP practice is to limit direct fire shooting engagements to the first five ranges: 250, 500, 1,000, 1,500, and 2,500 meters. The sixth range, well beyond 2,500 meters, is reserved as a "deep area" subject only to indirect fire against mid- to rear-area targets. Figure D-2 presents a graphic representation of the standard range protocol. The symbols represent tank-on-tank engagements with the usual indirect fire. The figure oversimplifies indirect fire in the sense that indirect fire, much as for direct fire weapons, must be distributed by range. For all possible direct fire versus direct fire weapon type engagements, input to the Combat Module specify the intended fractional distributions of duels by range for each type-on-type pairing. The duels previously assigned to the, at most, two odds classes are distributed roughly in proportion to the input-specified fractions by range. Only whole duels are distributed. This means, for example, that a solitary duel can be assigned to only one range leaving five ranges unoccupied. Such a solitary duel would be assigned to the range with the largest fraction within the input distribution. In the event of equal fractions (ties), a duel is assigned to the shorter (or shortest) range with that fraction. Each odds pool is distributed independently of the other. If two odds pools contain exactly one duel each, both of these duels will be assigned to the same range band. Now suppose an odds pool contains two duels. Again, tied fractions are broken by assignments to shorter before longer ranges. The first duel is assigned as already described. The second duel is assigned to the range with the second highest fraction. Obviously, an odds pool must contain at least six duels before each range can receive at least one duel. The fractional range distributions are input as two-place decimal values. Thus, something seemingly as innocent as a "uniform" distribution may produce surprises. Six equal nonnegative two-place decimal fractions cannot be made to total 1.0. In practice, input for the so-called uniform distribution consists of four 0.17s and two 0.16s. The shortest range with 0.17 receives singleton duels. For very large engagements with dozens of duels in each odds class, the result of distributing duels comes close to the input-specified distribution. Once the duels have been distributed by range, all the duels at a given range comprise a conflict, with an obvious maximum of six "simultaneous" conflicts, one for each range. From one conflict to the next on the same day, survivors must remain at their starting ranges, but survivors may be shifted between odds classes at the fixed range.

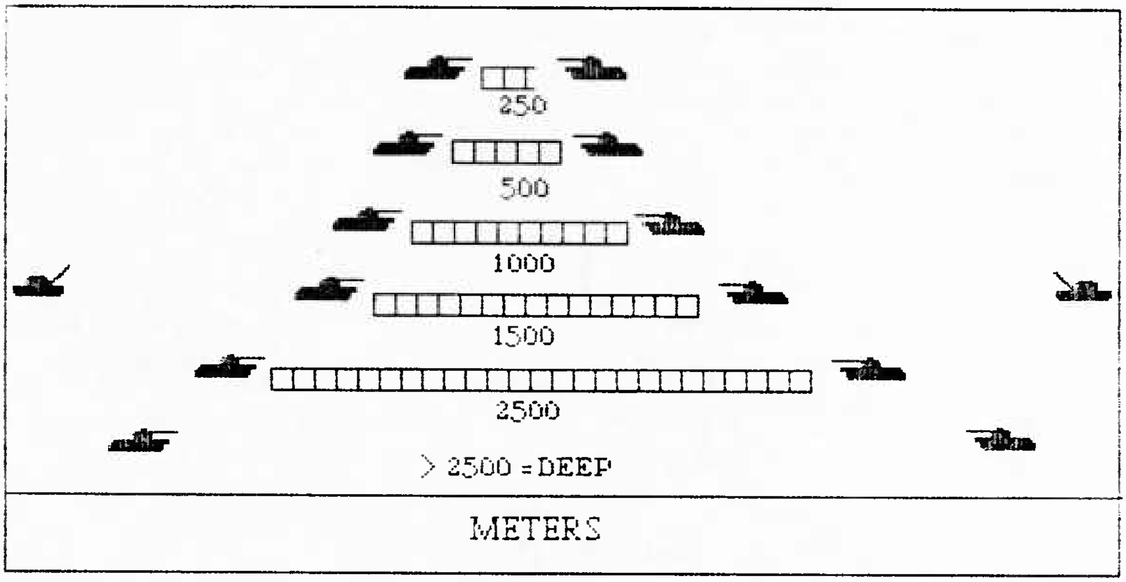


Figure D-2. AFP Standard Ranges of Engagement

(5) For an ordinary (noncounterbattery) direct fire duel, detection or an input-specified number of opponent's shots must precede any firing by a direct fire weapon. At the beginning of the duel, the Combat Module "draws" a detection time "at random" for each direct fire weapon from a detection distribution. For the detection time to be finite, the potential shooting weapon must be able to resolve a target. This means that the potential shooter's detection system must be able to achieve at least an input-specified number of resolvable cycles for the given target size in the given posture and under the given seeing conditions. A weapon gets one detection draw per target up to an input-specified maximum number of targets within the duel. The distribution is implemented within the Combat Module by a subroutine developed by Night Vision Labs as part of its detection/acquisition modeling. Arguments of the detection routine include target size, range, brightness, contrast, attenuation, and sensor type. With other arguments being equal, the more numerous side almost always should have the weapon with the shortest detection time, and hence, get off the first shot. The expected minimum of S draws ($S > 1$) is less than the expected minimum of fewer draws (from the same distribution) whenever the distribution has nonzero variance. Within broad practical limits of resolvable cycles, the detection routine is rather insensitive to target size. In AFP practice, the outnumbered side in a duel is usually in defensive positions and should, in principle and if equipped with the appropriate sensors, make the vast majority of first detections. For sufficient resolution, the routine returns a time-to-detect. Nondetection is flagged.

(6) Notice how time has crept into the Combat Module even though it is not a combat model or combat simulation. "Statics" is about to receive yet another temporal shock. The shooting sequence is important enough to real combat outcomes to force attention to shot sequencing into the Combat Module. The Combat Module depends on a sequencing index. (Perhaps no one will notice that a sequencing index is really just time by another name.) The events in the sequence are direct fire shots or bursts masquerading as "stages" or a little more openly as the critical events in "shot cycles." In general, the underlying times between consecutive stages are not equal; so the sequencing index is not a fixed time unit after all.

(7) How long should a conflict last? Input to the Combat Module imposes one limit: an actual time limit, typically 2.01 minutes. (That 0.01 minute is there to avoid some awkward counting problems that have never been observed but remain a theoretical worry. The 0.01, though, may be a cure for a nondisease.) The Combat Module does keep book against that maximum time. But it also keeps book on shot cycles. Before shooting begins, the Combat Module computes a permissible number of shot cycles within the conflict.

SHOT CYCLES = T * (1/R(INF) + MAX(S1,S2)/R(SUP))

where

T	is the maximum time of a conflict
R(INF)	is the refire time of the weapon in use by the numerically inferior side
MAX(S1,S2)	is the larger, nonempty odds class
R(SUP)	is the refire time of the weapon in use by the numerically superior side

All of which simply gives an estimate of how many direct fire rounds would be fired if no weapon were killed and if firing occurred at the mean refire times. Weapons may be killed and refiring times are drawn from a log-normal distribution with the result that the computed SHOT CYCLES alone often does not limit duels to maximum time duration T. Even without kills, duels at the smaller odds class would tend to exceed the time limit T. These are the principal reasons that the Combat Module keeps book on both shot cycles and time. At most, only one direct fire round (or burst) is fired per shot cycle. If no direct fire weapon has detected a target, no direct fire shot is taken, and no "time" elapses against maximum direct fire time. (Indirect fire may occur on the direct fire cycle even though no direct shot is taken.)

(8) The direct fire weapon that shoots first in a duel is the one with the shortest drawn or arbitarily set detection time. That weapon shoots at the first shot cycle, which "occurs" in real time, at the detection time. The round (or burst) has zero flight time. The SSPK of the weapon/round combination is compared with a randomly drawn number to determine whether a kill occurred. If a kill occurred, a randomly chosen "live" target is tallied as a kill. That target cannot detect or shoot evermore. A refire time is drawn for the weapon that just fired; that increment is saved for reference in determining at which later shot cycle that weapon gets to fire again if it is not killed in the meantime. Before the current shot cycle is completed, weapons which have not yet detected a target again have detection times drawn from the detection distribution. A successful detection puts that weapon in line to fire at a later shot cycle. A nondetection leaves the weapon flagged as unable to fire; it will get another chance at detection in the next shot cycle. A weapon that has not detected a target by the input-specified nth shot by an opposing side is then assumed to detect (n is weapon specific). Notice that the weapon that just fired may draw a very short refire time from the refire distribution. If its refire time is short enough, it may capture the next shot cycle and fire again before any other weapon fires. In the "long" run, weapons with shorter mean refire times as input will draw shorter refire times from the distribution and capture more of the shot cycles than will equal numbers of slower firing weapons. On the other hand, very numerous slow firing weapons, by their sheer numbers, get many draws from their refire distribution with the increased chance that one of the weapons will draw a short time and thereby capture an early shot cycle. Both for detection and refire times, sheer numbers of weapons increase the chance that some of them may preempt time from technically superior but numerically inferior weapons. During a single shot cycle, all the duels of a conflict are processed. Although the sequence index is the same for all the duels, each duel may be at a different clock time at that cycle. Notice that the probability distributions for detection and refiring are referenced in ways that depend on the numbers of weapons and the results of prior drawings. The net effect should be for the shots by both sides to interweave (subject to chance) properly over the shot cycles. Although direct fire ceases in a duel if all of one side's weapons are lost, the shot cycles continue in order that indirect fire continue to be represented. As noted above, indirect fire weapons may fire at the dueling direct fire weapons. Weaving indirect fire into the direct fire shot cycle sequence is the subject of the next paragraph.

(9) The current version of the AFP System in general, and Combat Module in particular, permits up to 10 types of indirect fire weapons per side to fire on the two weapon types of a normal or counterbattery version of a direct fire engagement. Each of the 20 indirect fire weapon types may have a refire time different from the rest. Hence, the Combat Module must be capable of interweaving 20 different indirect fire rates within the framework of direct fire cycles. Some indirect fire weapons may fire more rapidly than either of an engagement's two direct fire weapon types. Of course, some indirect fire weapons may fire more slowly than the direct fire weapons. And if the direct fire weapons fire at different rates, there may be some indirect fire weapons which fire at intermediate rates. All three of these cases may apply within a single direct fire engagement. Indirect fire is latched to the normal direct fire shot cycles. Whereas only one direct fire shot is permitted per direct fire cycle, indirect fire weapons of a type may fire zero, one, or more volleys within a single direct fire shot cycle. An indirect fire weapon type with refire times longer than the average time between direct fire shot cycles is permitted to fire only every R(IND)/R(DIR) shot cycles. Inasmuch as this ratio is not necessarily an integer, some compromise has to be made by the Combat Module. An indirect fire weapon with refire times shorter than the average time between direct fire shot cycles is permitted to fire R(DIR)/R(IND) times per shot cycle. This ratio, too, is not necessarily an integer; so again some compromise is necessary. In principle, an indirect fire weapon does not fire its first shot until its input-specified refire time has elapsed. However, an indirect fire weapon with refire time less than or equal to the mean time between direct fire shot cycles will be latched to fire on the first direct fire shot cycle. But because the first direct fire shot depends only on detection time and not the direct fire/refire time, the "time" to the first indirect fire shot(s) may be much shorter than the mean indirect refire time. In fact, if no direct fire weapon has detected a target on the first direct fire shot cycle, no direct weapon will fire nor will any direct fire time elapse. Thus, in the extreme, one or more volleys by an indirect weapon type may occur without so much as the first tick of the "direct fire clock." For all known weapons, this anomaly may be of theoretical interest, but it is not a practical concern.

(a) Consider 1:1 direct fire duels with mean refire times of 1.9 minutes for each of the direct fire weapon types. Also consider an indirect fire weapon type with a mean refire time of 1.5 minutes. Let the conflict duration be 2.01 minutes. Combat Module logic will assign two shot cycles to the duel. The implied average time between shot cycles is 2.01/2 = 1.005 minutes. Because 1.005 minutes is less than the indirect refire time of 1.5, the indirect fire weapons do not fire on the first shot cycle. The implied time of the second shot cycle is greater than 1.5 minutes; therefore, the indirect fire weapons do fire on the second shot cycle. The "real" time of the second shot cycle may be much less than 1.5 minutes; nevertheless, the indirect fire weapons do fire on the second shot cycle. The net effect of all these times is that the indirect fire weapons of the assumed type fire once during the conflict.

(b) Consider the above example with one change. Now let the refire time of the indirect fire weapons be 0.9 minutes. There are still just two direct fire shot cycles with the implied average time of 1.005 minutes between them. But because 0.9 minutes is less than 1.005 minutes, the indirect fire weapons fire on the first shot cycle. And because 2 x 0.9 is less than 2 x 1.005, the indirect fire weapons also fire on the second shot cycle. The net effect of all these times is that the indirect fire weapons of the assumed type fire twice during the conflict.

(c) Next consider the first example with a different kind of change. Suppose the duel occurs at 2:1 odds. Refire times remain unchanged. The Combat Module logic now assigns three shot cycles. The implied average time between direct fire shot cycles is now 2.01/3 = 0.67 minutes. The input-specified refire time of the indirect fire weapons is 1.5 minutes. The implied time of the first shot cycle is 0.67 minutes. Because 0.67 is less than 1.5 minutes, the indirect fire weapons do not fire on the first shot cycle. And because 2 x 0.67 is less than 1.5 minutes, the indirect fire weapons do not fire on the second shot cycle either. Yet, on the third shot cycle, 3 x 0.67 is greater than 1.5 minutes; thus, the indirect fire weapons do fire on the third (the last) shot cycle. The net effect of the interweaving of all these implied and real times is that the indirect fire weapons fire only once during the conflict.

(d) Finally, consider the second example with one change. Suppose the duel occurs at 2:1 odds. As in the third example, the Combat Module assigns three shot cycles with the implied average time between shot cycles of 0.67 minutes. The input-specified refire time is 0.9 minutes. The implied time of the first shot cycle is 0.67 minutes; because this is less than 0.9 minutes, the indirect fire weapons do not fire on the first direct fire shot cycle. Because 2 x 0.67 is greater than 0.9 minutes, the indirect fire weapons do fire on the second direct fire shot cycle. And because 3 x 0.67 is greater than 2 x 0.9 minutes, the indirect fire weapons fire again on the third shot cycle. The Combat Module logic is such that two, and exactly two, volleys are fired even if the conflict has 10 shot cycles; of course, the cycles on which the volleys are fired does depend on the number of shot cycles.

(e) The examples just described may seem like separate, unrelated conflicts of more theoretical than practical concern. Rounding to integral numbers of indirect fire volleys is a practical requirement and may be expected to induce some differences in the number of rounds fired from engagement to engagement. But as just shown, it is possible for the number of volleys to change between conflicts within the same engagement. An early conflict may involve entirely 1:1 duels. Attrition in the early conflict may cause a later conflict on the same day to occur at 2:1 (or higher odds). A change in the odds class changes the number of shot cycles and can (but usually does not) change the number of indirect fire volleys. In the example above, the odds increased in the "later" conflict. An example in which the odds decrease can be contrived just as easily. The point is that the number of indirect fire volleys may change upward or downward or may remain the same for conflicts on the same day within a single type-on-type engagement. Such changes are primarily artifacts of Combat Module logic for latching indirect fire to direct fire shot cycles. That logic is a potential source of variation in AFP results largely unrelated to the usual issues addressed in comparing weapons and units. The bottom line: BE CAUTIOUS.

(10) Consider a direct fire shot cycle with indirect fire latched to it. Because of the possibly different refire times of different indirect fire weapon types, less than all indirect fire weapon types are likely to be eligible to fire in this direct fire shot cycle. Or, if noneligibility to fire is considered to be eligibility to fire zero rounds, then each indirect fire weapon type is "eligible" at each direct fire shot cycle. Within a shot cycle, indirect firing is processed before the direct firing. Each indirect fire weapon type is considered in turn. Each of the previously assigned (the assignments are not described until later) indirect fire weapons of that type fires its "eligible" number of rounds: zero if its refire time ratio to the direct refire times does not "hit" this shot cycle, one round if the refire cycles match, more than one round if the indirect weapon is a rapid firer. All assigned weapons of that indirect fire weapon type fire their eligible number of rounds, say E rounds each. The net effect, if there are A assigned weapons, is for weapons of that type to fire E volleys of A rounds at the beginning of the otherwise direct fire shot cycle. Each such round is credited with an input-specified fractional kill capability K. Thus, the assigned indirect fire weapons of the type under consideration are "expected" to kill A x E x K = T targets. Usually T is not an integer. INT(T) kills are assessed directly against the targets within the conflict. INT(T) pointers to duels within the conflict and to targets within the duels are generated. If the target "pointed to" had not been killed previously in an earlier or this shot cycle, it is flagged as killed, and a kill is credited to the indirect weapon type. If the target "pointed to" had been killed previously, no credit for a kill is given, and the round is, in effect, wasted. In one sense, INT(T) is a deterministic number of kills with only the identity of the victim(s) to be determined. But as just explained, there is an element of uncertainty inasmuch as there may be fewer that INT(T) live targets. The fractional part, if any, of T must be assessed. Let T' = T - INT(T) be the fractional part of T. Recall that only whole targets can be killed and credited. T' is compared with a randomly drawn number to determine whether 1.0 or 0.0 kill is to be attempted to be credited. There is obviously no need to attempt to credit 0.0 kill. However, an attempt to assign 1.0 kill may succeed or fail. A pointer to a conflict and duel is generated randomly; if the pointer points to a live target, a kill is flagged and credited; if the pointer points to a killed target, no kill is credited. Indirect fire weapon types are processed in numerical index order. The first type has the greatest opportunity to score kills in the sense that following types may face fewer live targets. Fortunately, most of the indirect fire weapons yield such small T values that "who shoots first" is more of a theoretical than practical concern. But note that all the indirect fire kills are taken off the "top" in a shot cycle, possibly reducing the opportunities for the direct fire weapons to score kills, and thereby earn combat potential. Again, the small T values of typical indirect fire should only rarely deny opportunity to the direct fire weapons by killing them or their targets "prematurely." Even after one side's direct fire weapons have been killed (by indirect or direct fire of their combination), indirect fire will continue to latch on following shot cycles. Thus, a surviving side may continue to receive indirect fire and suffer losses until SHOT CYCLES is reached. Indirect fire is not limited by MAXIMUM TIME. In principle, indirect fire should stop at MAXIMUM TIME, but book is not kept on time after the direct firing has stopped. The preceding discussion is keyed to indirect fire falling on direct fire duels in the first five ranges where direct fire weapons may fire on one another on shot cycles. At the deep range, the so-called direct fire weapons do not fire on one another; hence, indirect fire is latched to artificially generated shot cycles.

c. Weapon Preference and Allocation. The preceding paragraphs have described many of the features of direct fire engagements given the numbers of engaging direct fire weapons and the numbers of indirect fire weapons firing on the direct fire weapons. This paragraph describes the processes by which those supposedly "given" numbers are determined. The underlying problem is very close to ones frequently described as assignment or allocation problems--so close that it seems useful to borrow some of the terms and illustrations of that field. The description also has a bit of game theoretic flavor, even though the Combat Module itself does not.

(1) Suppose a sort of two-sided game in which one side possesses two kinds of resources (U and V), and the other side possesses two kinds of resources (X and Y). Let there be a game board divided into quarters marked 1, 2, 3, and 4.

Game board or matrix

1	2
3	4

Side 1 may put some of U in cell 1 and some in cell 2. Side 1 may put some of V in cell 3 and some in cell 4. Side 2 may put some of X in cell 1 and some in cell 3. Side 2 may put some of Y in cell 2 and some in cell 4.

Allocation matrix

U1,X1	U2,Y2
V3,X3	V4,Y4

All quantities must be nonnegative, and they should satisfy some simple relations.

$$U1 + U2 <= UTOT$$
$$V3 + V4 <= VTOT$$
$$X1 + X3 <= XTOT$$
$$Y2 + Y4 <= YTOT$$

i.e., neither side may assign more of a resource than it has in total.

(2) Next suppose that Side 1 earns points in accord with some formula:

$$S1 = Q1(U1,X1) + Q2(U2, Y2) + Q3(V3,X3) + Q4(V4,Y4)$$

and that Side 2 earns points in accord with some formula:

$$S2 = R1(U1,X1) + R2(U2,Y2) + R3(V3,X3) + R4(V4,Y4)$$

(3) Presumably, Side 1 would like to assign its resources (i.e., pick the values U1, U2, V3, and V4) in a way that gives high assurance of achieving a good score S1 for itself but with some regard for the score S2 achieved by the other side. And presumably, Side 2 would like to assign its resources (i.e., pick values X1, X3, Y2, and Y4) in a way that gives high assurance of achieving a good score S2 for itself but with some regard for the score S1 achieved by the other side. Game theorists know how to make these assignments provided some very special circumstances apply. In AFP those special circumstances do not apply. Nevertheless, the AFP Combat Module must make resource (weapon) assignments in cases with up to 60 kinds of resources for each side, implying up to a 60 x 60 game board. In AFP terms, a side may assign only one type of direct fire weapon to a cell, but it may assign all of its indirect fire weapon types to a cell. There is no AFP assignment/allocation algorithm for maximizing a side's combat potential much less while simultaneously minimizing the opponent's combat potential--or vice versa--or both together in a two-sided min-max sense. Nevertheless, there is an AFP algorithm that does make all the assignments. More accurately, there are two algorithms, one for direct and another for indirect fire weapon assignment/allocation. The most that is ever said for the algorithms is that they usually lead to "reasonable" allocations. And in those cases where allocation results are objectionable, the AFP customer is invited to suggest better rules--hardly a trivial matter.

(4) Consider again the four-celled game board appropriate for the special case in which each side possesses only two types of resources. And for the time being, suppose that there are equal (or standard) quantities of all resources. Then, it seems helpful to think in terms of "preferences."

Fractional preference matrix

p(U,X),p(X,U)	p(U,Y),p(Y,U)
P(V,X),p(X,V)	p(V,Y),P(Y,V)

p(U,X) represents Side 1's "fractional preference" to have resource U engage Side 2's resource X. p(X,U) represents Side 2's fractional preference to have resource X engage Side 1's resource U. All p()'s should be nonnegative. For ordinary direct fire weapons, it is intended that:

$$p(U,X) + p(U,Y) = 1.0$$

For the special counterbattery version of direct fire, the corresponding relation is:

$$p(U,X) + p(U,Y) = f(U) \leq 1.0$$

with the implication that (1.0 - f(U)) x UTOT of weapon type U will be assigned to the indirect fire role. At first, it may seem satisfactory to let, for example (here again assuming ordinary direct fire):

$$U1 = p(U,X) \times UTOT$$

In the case where weapons are equal in quantity, this simple expression may be sufficient. But the allocation method should also produce reasonable results under some other special cases. If XTOT = 0, then Side 1 should let U1 = 0. In other words, do not waste weapons by assigning them to cells empty of opponents. Also, if Side 2's weapon X is a greater killer of Side 1's weapon U, Side 1 must be prevented from escaping free simply by choosing $p(U,X) = 0$. Consider the generalizations to handle these special cases in reverse order.

(a) The first generalization is intended to provide some reconciliation of two sides' differences in preference. The intent is to force at least some "reluctant" weapons to engage difficult opponents and to preclude at least some "eager" weapons from engaging easy targets. This is accomplished by modifying fractional preferences in such a way that two different preferences move half-way toward their mean value. For example:

$$p'(U,X) = (\text{ mean }) + (\text{half the difference from mean})$$

$$= 0.5(p(U,X) + p(X,U)) + 0.5(p(U,X) - 0.5(p(U,X) + p(X,U)))$$

$$= 0.75p(U,X) + 0.25p(X,U)$$

Incidentially, this expression is correct whether $p(U,X)$ is greater than, equal to, or less than the mean of $p(U,X)$ and $p(X,U)$. It is also the mean of the original preference with the mean of the two original preferences.

$$p'(U,X) = 0.5(p(U,X) + 0.5(p(U,X) + p(X,U)))$$

The result of performing this step for all cells will usually lead to a relation:

$$p'(U,X) + p'(U,Y) <> 1.0 \ (\text{or} <> f(U) \text{ in the general case})$$

Therefore, the first modified preferences are modified again by renormalizing them to yield the original totals. For example:

$$p''(U,X) = (p'(U,X) \times f(U))/(p'(U,X) + p'(U,Y))$$

The renormalized, modified preferences satisfy:

$$p''(U,X) + p''(U,Y) = f(U)$$

The Combat Module performs these steps for up to 60 weapon types on each side.

(b) The second generalization makes allocation depend not only on the fractional preferences but also on the opponent's inventory quantities. The intent is to allocate more than the simple fraction of one's weapon type if the opponent has more than "average," and to allocate less than the simple fraction of one's weapon type if the opponent has less than the "average" number of the type in question. This is why it was suggested that the original fractional preferences be thought of in the special context of equal numbers (or standard numbers) of weapons. Analysts seem to waver between notions of equal and standard numbers of opponents as they "fill in the blanks" on a fractional preferences input forms. Whatever those analysts' feelings were during input data generation, the Combat Module adjusts for opposing inventories; for example:

$$U1 = UTOT \times \frac{(p''(U,X) \times XTOT)}{(p''(U,X) \times XTOT) + (p''(U,Y) \times YTOT))}$$

$$U2 = UTOT \times \frac{(p''(U,Y) \times YTOT)}{(p''(U,X) \times XTOT) + (p''(U,Y) \times YTOT))}$$

These allocations have many of the intended properties.

1. U1 + U2 = UTOT

2. The renormalized, reconciled fraction preferences p"() are used

3. If there is no opposing weapon (e.g., if XTOT = 0), none of resource U will be allocated to engage it

4. If the original fractional preferences have been picked with an underlying notion that all types of opposing weapons were equal in number (i.e., XTOT = YTOT = ATOT), the expression, for example, for U1 above becomes:

$$U1 = UTOT \times \frac{(p''(U,X) \times ATOT)}{(p''(U,X) \times ATOT) + (p''(U,Y) \times ATOT))}$$

$$U1 = UTOT \times \frac{p''(U,X)}{p''(U,X) + p''(U,Y)}$$

where for the usual direct fire weapons

$$p''(U,X) + p''(U,Y) = 1.0$$

yielding the original "intuitive" form

$$U1 = UTOT \times p''(U,X)$$

Hence, the generalized procedure includes the original special case.

(5) The allocated weapons need not all get into conflicts. As noted later, the numbers allocated may be reduced as the results of external losses or participation factors less than 1.0.

(6) In general, application of the allocation procedures described so far does not yield integers. Therefore, the Combat Module includes additional steps. The integral parts of allocations are left "in place." The fractional parts are redistributed in integer parts to the engagements in order of descending size of the former fractional parts.

(7) The allocation of indirect fire weapons is similar to the process described above for the direct fire weapons. For indirect fire weapons, fractional preferences and participations are input as a single combined number--one for each direct fire weapon (target) type. One constraint is relaxed, however. The same indirect fire weapons may be considered to fire on different engagements because different types of direct fire weapons may be collocated within the same target area even though the Combat Module has artificially segregated all direct fire weapons into pure-type-on-pure-type engagements: conflicts and duels. Recall that some weapons of indirect fire type may be unavailable as indirect firers because they have been treated as counterbattery (i.e., special direct fire) weapons. As usual, the early steps of allocation may produce numbers that are not pure integers. The Combat Module includes logic to distribute fractional parts so that only integral numbers of indirect firers are finally allocated.

(a) Not all of the indirect fire weapons should fire on every engagement cell involving the given type of direct fire weapons. If W1 weapons of a total of WT weapons of the direct fire type have been allocated to engagement cell 1, then only W1/WT of the indirect fire weapon type are allocated to cell 1. The fraction W1/WT is determined during execution of the Combat Module. At this stage, (W1/WT), of the indirect fire weapons have been allocated to engagement cell 1. The Combat Module distributes those indirect fire weapons across ranges in proportion to the input-specified range distribution for each corresponding indirect fire weapon type. If a fractional number of indirect fire weapons is assigned to a range, the number is rounded up or down stochastically in proportion to the fractional part of the assignment.

(b) The indirect fire weapons assigned to fire on engagement cell 1 do so under the assumption that rounds fall within a doctrinal target area against targets at doctrinal densities. The preprocessor derived fractional kills per round for this assumption. As the first conflict of a day runs its course, some of the direct fire weapons may be killed by direct or indirect fire weapons. The killed weapons remain within the implied array, thereby maintaining target densities consistent with the assumptions. Hence, it is appropriate to continue to use the same fractional kills per round and the same number of direct fire weapons throughout the first conflict. The Combat Module continues to distribute hits in accord with the original densities but does not give credit for killing the same target more than once. If subsequent conflicts occur on the same day, the Combat Module removes all targets killed in the preceding conflict. One effect is to reduce the target density below that assumed during the original derivation of the fractional kills per round input to the Combat Module. The Combat Module continues to use the same fractional kills per round throughout all conflicts during a day. However, the Combat Module compensates by reducing the number of indirect fire weapons firing on the ensuing conflicts. The factor W1/WT defined above is generalized such that W1 represents only the survivors of the preceding conflict. Although the adjustment is made to the number of indirect fire weapons assigned, the result is equivalent to having made the adjustment to the fractional kills per round. One systematic error does remain, however. No attrition occurs to the indirect fire weapons during the course of a Combat Module day. The counterbattery fire weapons are attrited during the course of a day, but such attrition cannot affect the number of indirect fire weapons until the beginning of the following day.

d. Census Space Formalism

(1) The "normal viewpoint" for considering the estimation of AFP combat potential is that of an observer of the census of up to 120 weapon types: up to 60 Blue types and up to 60 Red types. Column A of Table D-1 represents the initial number of m Blue weapon types by the symbols x0(1), x0(2),...,x0(m) and the initial numbers of n Red weapon types by the symbols y0(1), y0(2),...,y0(n). Together these symbols define a point in a census space of (m+n) dimensions. Column A can be considered the position vector of the starting point for a run of the AFP Combat Module.

Table D-1. AFP Combat Potential Estimation in "Census Space"

Col A	Col B	Col C	Col D
x0(1)	x1(1)	(1-f)x0(1)	X1'(1)
x0(2)	X1(2)	(1-f)x0(2)	x1'(2)
.	.	.	.
.	.	.	.
.	.	.	.
x0(m)	x1(m)	(1-f)x0(m)	x1'(m)
y0(1)	y1(1)	y1'(1)	(1-f)y0(1)
y0(2)	y1'(2)	y1'(2)	(1-f)y0(2)
.	.	.	.
.	.	.	.
.	.	.	.
y0(n)	y1(n)	y1'(n)	(1-f)y0(n)

(2) Column B of Table D-1 represents the numbers of weapons remaining or surviving at the end of a run of the AFP Combat Module. Column B can be considered the position vector for the ending point of a Combat Module run. The Combat Module does not generate much more information than the equivalent of Columns A and B. (Actually, the Combat Module does produce the equivalent of Columns A and B for each "day" of a Combat Module run.

And because the module decomposes combat into weapon-type-on-weapon-type duels, information about the starting and ending points for duels is also available. However, this section formally introduces decomposition in later paragraphs. For the time being, the reader should wait patiently for decomposition to appear in logical order.)

(3) One of the problems faced in estimating combat potential is to make the measure of potential depend on both enemy and friendly losses. The losses generated in an AFP Combat Module run are given by the difference between Columns A and B of Table D-1 as the ordinary vector difference A-B. Some analysts favored making combat potential a weighted sum of selected elements of the vector A-B. It was clear that the result of such a computation would, in general, depend too strongly on the number of "days" examined in an AFP Combat Module run. A result so strongly dependent on a number of "days" was objectionable on the grounds that the measure retained to much time-like flavor to be considered a static measure. Also, static or dynamic, no one offered a sound basis for picking the "correct" number of days. Although not necessarily objectionable, the fact that the side whose potential was being estimated generally lost different fractions of its weapon types did seem to confuse interpretation of the resulting measure. To have unequal fractional losses for all or nearly all weapons on both sides makes division or force comparisons that much more confusing. Letting both the numerators and denominators of exchange-like ratios float implied an unspecified standard of comparison. The need to consider, at least in part, exchange-like ratios arose with the commitment to treat both losses inflicted and losses suffered.

(4) The concepts and operations described in this and the following subparagraphs are applied in the AFP CBT/CS/CSS Merge Module, not in the Combat Module. The Column B vector in Table D-1 represents a point in census space in which all surviving strengths correspond to a single instant in Combat Module "time," the end of the last module "day." That point is a natural one in the sense of the Combat Module. There are other natural points in census space--natural from other points of view. One such point is the one representing an estimate of Red (y) surviving strength given that exactly (1-f) of each Blue (x) weapon strength survives, i.e., the point for which all Blue coordinates are of the form (1-f)x0(i) for i=1 to m. The obvious question is, "What are the corresponding Red (y) coordinates y1'(j) for j=1 to n of the point?" The position vector of this point is represented in Column C of Table D-1. In general, the coordinates y1'(j) must be functions of the coordinates (1-f)x0(i). The combat potential (without CS/CSS modulation) for the entire Blue force (symbolized as BCOP) is then calculable as follows (with v(j) the assigned value of target type j):

$$\text{BCOP} = \sum_{j=1,n} v(j)\ \left(y0(j) - y1'\left(j;(1-f)x0(1),(1-f)x0(2),\ldots,(1-f)x0(m)\right)\right)$$

This formula is simply the weighted difference between the y portions of the vectors in Columns A and C. As yet, nothing has been revealed about the form of the functional dependence of the y1' on the fixed coordinates (1-f)x0(i); two families of functional forms are presented in paragraph D-1e. An important assumption of AFP is that the foregoing expression for BCOP can be decomposed as follows:

$$BCOP = \sum_{j=1,n} v(j) \sum_{i=1,m} (y0(j,i) - y1'(j,i;(1-f)x0(i)))$$

The decomposition reflected in this expression corresponds to the way in which the AFP Combat Module decomposes total combat into engagements between single types of direct fire weapons. The generalized notion splits the total of y-type j y(j) into components, into the portions engaging Blue type i y(j,i). Similarly, x(i,j) represents the portion of Blue type i engaging Red type j.

(5) CS/CSS modulation involves multiplying the terms in the preceding expression by the moduli appropriate to the particular weapon type i and j pairings. The modulated Blue COP is symbolized as BMODCOP and is calculated in accord with the following expression:

$$BMODCOP = \sum_{j=1,n} v(j) \sum_{i=1,m} (BCSCSS(i,j)) \; (y0(j,i) - y1'(j,i;(1-f)x0(i)))$$

(6) The determination of unmodulated and modulated Red COPs is perfectly symmetrical with that for the Blue COPs above. The corresponding expressions may be generated by replacing "B" by "R" and exchanging y and x wherever they appear above. In census space, the operation corresponds to taking the difference between the x portions of the vectors in Columns A and D of Table D-1.

(7) The unmodulated Red COP is given by (with u(i) the assigned value of target type i):

$$RCOP = \sum_{i=1,m} u(i) \; (x0(i) - x1'(i;(1-f)y0(1),(1-f)y0(2),\ldots,(1-f)y0(n)))$$

(8) Decomposition of the foregoing expression to correspond to the AFP module's weapon-type-on-weapon-type duels yields:

$$RCOP = \sum_{i=1,m} u(i) \sum_{j=1,n} (x0(i,j) - x1'(i,j;(1-f)y0(j)))$$

(9) Modulation of the unmodulated Red COP requires inclusion of the CS/CSS moduli corresponding to the specific j and i type weapon pairings.

$$\text{RMODCOP} = \sum_{i=1,m} u(i) \sum_{j=1,n} (\text{RCSCSS}(j,i))\ (x0(i,j) - x1'(i,j;(1-f)y0(j)))$$

e. Some Examples for Two-dimensional Vector Fields

(1) The AFP Combat Module is based, among other things, on an assumption that combat in a census space of up to 120 dimensions may be decomposed to a satisfactory degree of approximation as independent direct fire duels (with some intrusion by indirect fire) in up to 3,600 two-dimensional subspaces. At its most fundamental level, AFP combat analysis reduces to the consideration of highly specialized, two-dimensional vector fields. The AFP Combat Module generates a single estimate of exchange ratio for each day for each two-dimensional, type-on-type engagement. In this discussion, indirect fire is ignored for purposes of simplifying the basic explanation. The fundamental data describe just two points in the two space. The starting point of an engagement is represented by the ordered pair: y0, x0. The ending point of an engagement is represented by the ordered pair: y1, x1. Hence, y1 and x1 simply represent the surviving strengths at an engagement's end. Then the gross exchange ratio is simply: $(y0 - y1)/x0 - x1)$. The AFP Combat Module does not extend or shorten an engagement to force any particular fractional lifetime engagement on either side. That is, both $(y0 - y1)/y0$ and $(x0 -x1)/x0$ are unconstrained apart from the obvious requirement that $y1 >= 0$ and $x1 >= 0$.

(2) The current AFP method imposes a fixed fractional lifetime (or lifespace) point at which to estimate combat potential. All combat potentials are estimated for a fixed fraction:

$$f = (x0 - x)/x0 = (x0 - (1-f)x0)/x0$$

for the x side. A similar rule applies to the y side:

$$f = (y0 -y)/y0 = (y0 - (1-f)y0)/y0$$

Current AFP work applies $f = 0.50$, i.e., attention is confined to combat potential over half-lifetimes or lifespaces.

(3) Because the Combat Module does not automatically halt engagements at y strengths of $(1-f)y0$ or at x strengths of $(1-f)x0$, the AFP method must estimate exchange ratios for the surviving strengths $(1-f)y0$ and $(1-f)x0$ points based strictly on the only known points: y0, x0 and y1, x1. For example, a contribution to the x-side COP should have the form:

$$z(x0,y0;f) = ((y0 - y((1-f)\ x0))/(x0 - (1-f)x0))fx0 = y0 - y((1-f)x0)$$

This result depends on the path y(x) presumed to emanate from the starting point y0,x0. The AFP module does not help the analyst decide which to choose among the many possible paths passing through the two points y0, x0 and y1, x1, the only two points assured by the Combat Module. The following paragraphs give examples for two assumed one-parameter families of vector fields and corresponding paths in the x,y plane.

(4) Assumed Vector Field Family I. Consider the time-derivative vector field:

$$dx/dt = -(ay)^n$$

$$dy/dt = -(bx)^n$$

This two-component vector is the derivative of positions with respect to "time." In the strictest sense, consideration of time derivatives violates the AFP principle of limiting attention to static measures. Less strictly, AFP philosophy permits examination of a time derivative at some fixed time (a "snapshot" is permissible) as long as no attempt is made to integrate with respect to time. The vector field is shown first in this time-snapshot form, primarily to clarify the origin of what follows, which does revert to purely spatial considerations with integration limited to space (not time) coordinates. Inasmuch as neither a nor b varies with time, the vector field itself is already "static." This is a very remarkable and limiting assumption. Much more general vector fields can be imagined. Indeed, learning and logistic phenomena often imply not only that a and b should vary with time, but also that they should depend on the path and time spent on that path in reaching the particular point x,y. The time-derivative vector field generates paths across the x,y plane. Those paths have tangents:

$$dy/dx = (bx/ay)^n$$

The tangents are, of course, also vectors. Hence, there is still a direction and length associated with every point in the x,y plane. The paths corresponding to the tangent field have the form:

$$(y0^{(n+1)} - y^{(n+1)}) = k(x0^{(n+1)} - x^{(n+1)})$$

If the value of n is known from some authoritative source outside AFP, then a path depends on the single parameter, $k = (a/b)\ \ n$. Given the two points y0, x0 and y1, x1 generated by the AFP Combat Module, the parameter k is determinable as:

$$k = (y0^{(n+1)} - y1^{(n+1)})/(x0^{(n+1)} - x1^{(n+1)})$$

With the appropriate substitutions, the contribution to the x-side COP for a particular x,y engagement becomes:

$$z(x0,y0;x1,y1;n,f) =$$

$$y0 - \ (y0^{(n+1)} - \ ((y0^{(n+1)} - y1^{(n+1)})/(x0^{(n+1)} - x1^{(n+1)})))*$$

$(1 - (1-f)^{(n+1)}) * x0^{(n+1)})^{(1/(n+1))}$

(5) Assumed Vector Field Family II

(a) Consider the time-derivative vector field:

$dx/dt = -(axy)^n$

$dy/dt = -(bxy)^n$

(b) As above, the time-derivative vector field corresponds to a field of tangents and paths across the x,y plane. As above, a and b are assumed independent of position, time, and path. Regardless of the value of n, the paths are all straight lines:

$(y0 - y) = k(x0 - x)$

(c) Given the two points y0, x0 and y1, x1 generated by the AFP Combat Module, the parameter k is determinable as:

$k = (y0 - y1)/(x0 - x1)$

(Cases occur with x0 = x1, i.e., the x side may suffer no losses. In that event and if the y side does suffer losses, AFP sets the x-side loss to an arbitrary 1.0. If neither side suffers losses (i.e., 0/0), k is set to 0.0.)

(d) With the appropriate substitutions, the contribution to the x-side COP for a particular x,y engagement becomes:

$z(x0,y0;x1,y1;f) = ((y0 - y1)/(x0 - x1)) * f * x0$

This expression is independent of n, which affects the rapidity with which a path would be traversed, but not the path proper. Recall that the time at which x and y occupy a particular point on the path is of no concern in so-called static analysis. The parameter n need not be established (or guessed) for this vector field family--unlike the situation for the example vector field family I above. On the other hand, application of vector field family II depends on the very strong assumption of straight-line paths in the x,y space.

(e) It is basically the vector field family II approach that is applied within current AFP practice. The actual approach is generalized to admit indirect fire losses to the otherwise pure type-on-type, x,y direct fire engagements. Also, if suffered, some nonbattle losses are included in the computation. With up to 60 weapon types on each side and with usually multiple engagements, the COP and CIP scoring involves some special averaging procedures that introduce some complexities ignored in the above examples.

(6) The above two examples cover only two popular one-parameter tangent fields. The first is Lanchester's "square" law; the paths in the x,y plane are quadratic. The second is Lanchester's "linear" law; the paths are straight lines. There are, of course, many other imaginable one-parameter fields and even more multiparameter fields. As currently employed, the AFP Combat Module is not useful for determining the best field family; it is limited to a single parameter estimation for an assumed field law. In the longer run, some changes in the application of the AFP Combat Module would be more revealing about its underlying vector fields. In the meantime, the linear law reigns supreme.

f. Combat Decomposition Revisited

(1) AFP begins by decomposing the totality of CBT/CS/CSS into the artificially separated areas of CBT and CS/CSS. Both CBT and CS/CSS are further decomposed. The results determined from the separate elements are then combined to yield estimates of combat potentials at several levels of aggregation.

(2) CBT is represented as conflict among as many as m Blue and n Red weapon types. In current practice, m=n=60. Principal concerns are the numbers of weapons allocated, participating, surviving, and "killed." All these numbers are expressible as "points" in a "space" of m+n dimensions. Hence, in current AFP practice, the underlying combat framework is a space of 120 dimensions. Inasmuch as AFP is devoted to the development of "static" measures of combat potential, the extent to which different points in the 120 space may correspond to the elapse of time must be regarded merely as an AFP artifact. Even though AFP may track paths of points in the 120 space, it must do so largely for "static" reasons. That some of the same results may be determined by integration, with respect to time or by integration with respect to the coordinates of the 120 space, is a fortunate coincidence. However, such equivalences should serve as a reminder that some of the distinctions between static and dynamic may be matters more of form than substance.

(3) In 120 space, the value of the ith coordinate corresponds to the surviving quantity of the ith item type. If track is to be maintained of "live," "disabled," and "killed" items, the ith coordinate becomes three coordinates. That is, an original one-dimensional subspace becomes a three-dimensional subspace of a new 360 space. However, the only admissible points in a three-dimensional subspace lie on the three plane with:

#live + #disabled + #killed = starting strength

It is assumed that all assets are present initially within the starting strength.

(4) Many of the matters of interest to AFP may be shown to be equivalent to concern with special transformations of the 120 space into itself. The narrowest view is that AFP need be concerned with how and why a single point in 120 space becomes another point in that space, e.g., how, through attrition, engagements transform the point corresponding to starting strengths into the point corresponding to surviving strengths.

This narrow view leads us to think that we may have to consider a different transformation for each different starting point. A more general, and in many respects, more powerful and useful viewpoint is to consider, as suggested in the first sentence of this paragraph, transformations of the whole space into the whole space. The starting and ending points in the 120 space become some of the principal intermediate "results" on which final estimates of combat potential are based. The standard AFP practice of examining 16 distinct environments corresponds to dependence on 16 sets of starting and ending points in 120 space. If the different environments correspond to different force levels or ratios, the starting points (and almost certainly, too, the ending points) would differ.

(5) In the following paragraphs, we shall consider stochastic and deterministic transformations, dynamic and static transformations, and decompositions of the 120 space into as many as 3,600 2-spaces. All these considerations are relevant to AFP. That is not to say that AFP achieves all the generality implicit in such consideration. Rather, we will try to put the specializations and simplicifications of AFP in a "mature" perspective. However, our so-called mature perspective will depend much more on intuition than on rigor.

(6) Everyone agrees that combat is uncertain, even for many supposedly lopsided conflicts. Realism then should lead us to consideration and application of stochastic transformations. AFP represents some but relatively little of the uncertainty of combat. AFP introduces some uncertainty to the detection, firing, and killing processes. AFP does not reflect any uncertainty in the starting strengths of engagements. Apart from a need to round off (or up) fractional weapon assignments randomly, the AFP engagement preference and allocation process is purely deterministic. Hence, AFP transformations may be characterized as transforming completely fixed initial points into uncertain end points. In a single standard execution, the AFP Combat Module transforms a single fixed initial point into a single uncertain final point; the result is not a distribution of final points. However, the AFP Combat Module can be operated in a replicative mode; the collective result of many replications is a distribution of final points. In principle, generalization of AFP replication to admit different starting points (i.e., some randomization of allocations by weapons and ranges for the same engagement preferences and range fractions) could reduce some difficulties arising from relatively small numbers of weapons.

(7) Let us consider a very "small" deterministic transformation of the 120 space into itself. We associate with each point in the 120 space a very short (that is what we mean by "small") arrow pointing to where the point will "go" under the transformation. The 120 space is filled with such arrows. We call the totality of these arrows in 120 space the "vector field." In this field, each arrow (vector) possesses position, direction, and length. If the directions and lengths of the vectors do not change with time, we say the vector field is static.

(8) Now let us drop a marble onto the vector field. The marble will land exactly where we want it to; thus, its initial position is deterministic. Once the marble has landed, it will move in 120 space in accord with the directions imposed by the vectors which now serve as direction signs. Note that the marble moves even though the vector field is static. There, indeed, can be marble dynamics in a space that is static. Much more general vector fields can be imagined. The directions and lengths of the vectors may change with time relative to the start of a clock, to where a marble lands, or to the path traversed by a marble in reaching the point. Even within the class of static vector fields there may be many kinds. The direction and length of vectors may or may not depend on their position. There may be many different kinds of dependence on position. Depending on the nature of the field, dropped marbles may roll in the 120 space with constant or changing velocity along paths of constant or changing direction. In general, we will want to ascribe the curvature of paths of dropped marbles to a kind of curvature of the underlying vector field (even though we have not defined exactly what we mean by the curvatures of paths and fields). Without an exact definition of curvature, however, we can state that the "observation" of a single starting and a single ending point in 120 space will not be sufficient to estimate the curvature of the vector field. And we should not fool ourselves about what happens when we decompose 120 space into an many as 3,600 2-spaces. It may appear that we have observed 3,600 pairs of starting and ending points, and that we should have enough information to estimate curvature. Unfortunately, we have nothing more than the equivalent of the original pair of points in 120 space. Also unfortunately, years of combat analysis have not given us widely accepted empirical or theoretical evidence about the curvature of the vector space. If there were an accepted notion of curvature, then a single pair of points might be sufficient (depending on the number of parameters needed to specify curvature). In particular, one pair of points might suffice in the case of a known, one-parameter curvature. AFP, lacking a sound theoretical or empirical foundation, relapses to estimating paths of zero curvature in a vector field of zero curvature. Lest it seem that all this "straightness" and "flatness" of paths and fields reduce the problem to perfect transparency, note that any change in engagement preferences or range fractions may correspond to a different static vector field.

(9) A useful analog of the AFP "curvature problem" is the following: Given only that a traveler has started in Washington, DC and stopped in Baltimore, MD, estimate where the traveler was, is, or will be when half of the fuel remains. It has not been established what the traveler's vehicle is, if any. The traveler's final destination has not been established. Does it help to know that Rte 29, Rte 95, Rte 1, and the Baltimore-Washington Parkway are all popular routes between Washington and Baltimore? In general, AFP would know that the traveler started with full fuel in Washington, and AFP would know how much fuel remained when the traveler reached Baltimore. All other possibly relevant information would remain unknown. In quiet desperation, AFP would draw a straight line between Washington and Baltimore and choose an interpolated or extrapolated point in direct proportion to the known fuel consumption.

(10) In general, we may suspect that CS/CSS should influence the curvatures of the vector field and paths. AFP does modulate the combat paths in 120 space for CS/CSS. Although modulation does alter the direction and length of paths, such modulation does not alter the assumed zero curvature of the paths and underlying vector field.

(11) AFP does visualize the components of the "real" vector field as expressible in terms of functions of position in 120 space. Those functions are considered to consist of several terms of possibly increasing complexity. The first and perhaps simplest term contributes nothing to curvature. Hence, any approximations involving only first terms of the components of the vector field are necessarily approximations at zero curvature. For the simple approximation to be useful, it is necessary that higher order terms contribute relatively little to the real field. When AFP decomposes the 120 space into up to 3,600 2-spaces, the first terms of the vector components are assumed to be independent of position with respect to any direct fire weapons other than the two associated in the 2-space. The effect is to disregard combined arms and synergistic influences! Combined arms and synergistic effects could appear in spaces of zero curvature, but in many respects, it seems that combined arms and syngerism would most naturally introduce to curvature. But here, too, AFP lacks sufficient theoretical and empirical evidence on which to introduce curvature into the zero-curvature approximation of the combat vector field. To the extent that the presence of different weapon types influences the allocation of weapons among engagements, the starting points in the two spaces are combined arms-dependent. However, it would be misleading to suggest that AFP provides much combined arms sensitivity.

(12) Once AFP has decomposed the 120 space into 2-space minibattles, it reintroduces indirect fire to the 2-space direct fire minibattles. Thus, the direct fire minibattles are represented in a higher-dimensioned space. Nevertheless, the practical AFP viewpoint is one of 2-space direct fire minibattles. The minibattles are decomposed into "duels." Each duel involves the smallest number of weapons consistent with the force ratio. A 10-on-10 minibattle decomposes into 10 one-on-one duels. A 15-on-10 minibattle decomposes into 5 two-on-one and 5 one-on-one duels. The direct fire duels allocated to a 2-space minibattle are distributed over five range points from 250 m to 2,500 m. (Thus, with the so-called "deep area," a maximum of six range points may be involved.) There is no interaction among weapons at different ranges. The weapons separated by 1,000 m behave as though there are no weapons at 500 m or 1,500 m. Weapons at the same range separation are then allocated to microbattles. Implicit in this scheme is the notion that force ratio may be important but that force mass is unimportant. If force mass is important, then all weapons should be allocated to all range points with results then averaged over range points. If something about attrition is known in advance, then the allocation from longer to shorter ranges could become $A(deep)=1.0 \geq A(2,500) \geq \ldots \geq A(250)$. The current APF method of averaging results for CIPs would work as is. The current AFP method of averaging for COPs would require change.

D-2. INPUT

a. Of all AFP modules, the Combat Module requires the most diverse and voluminous input data. Many "raw" data undergo transformations before being read by the main program of the Combat Module.

b. Figure D-3 presents a highly condensed schematization of the files and procedures involved in preparing input to the Combat Module. In Figure D-3, the total effort is shown as eight separate tasks.

(1) RANGE

(2) ENGAGE

(3) PCAS

(4) PKS

(5) PREFS

(6) ARTY

(7) INVENTORY

(8) SENSOR

c. Notice that the first five tasks shown in Figure D-3 lead to the generation of mutually exclusive files but that tasks 6-8 all provide information to a commonly named file, H7BASEDATA. The AFP authors chose to organize the description of AFP input data and their preparation in the context of the files shown at the right of each task in Figure D-3. Those descriptions are provided in the annexes to this appendix. In Figure D-3, the products of tasks are files given names in the generic form M-H7aaaaa.(a). The M is intended to signify "machine-readable" files from some other file with L prefixes signifying "labelled human-readable" files. In the following paragraphs, the "M-H7" portions of files names are omitted without ambiguity. Figure D-3 includes some other symbols intended to serve as reminders: (P) is a reminder that different data are usually required for different combat (p)ostures--usually because open/defilade and moving/stationary status differs among postures for both shooter and targets. (D/N) is a reminder that different data are usually required for (d)aytime an (n)ighttime conditions. (C/D) is a reminder that different data are usually required for (c)lear and (d)egraded seeing conditions. (F) is a reminder of dependence on the quantity of systems in a (f)orce. (A) is a reminder of probable dependence on the (a)rea of the world considered.

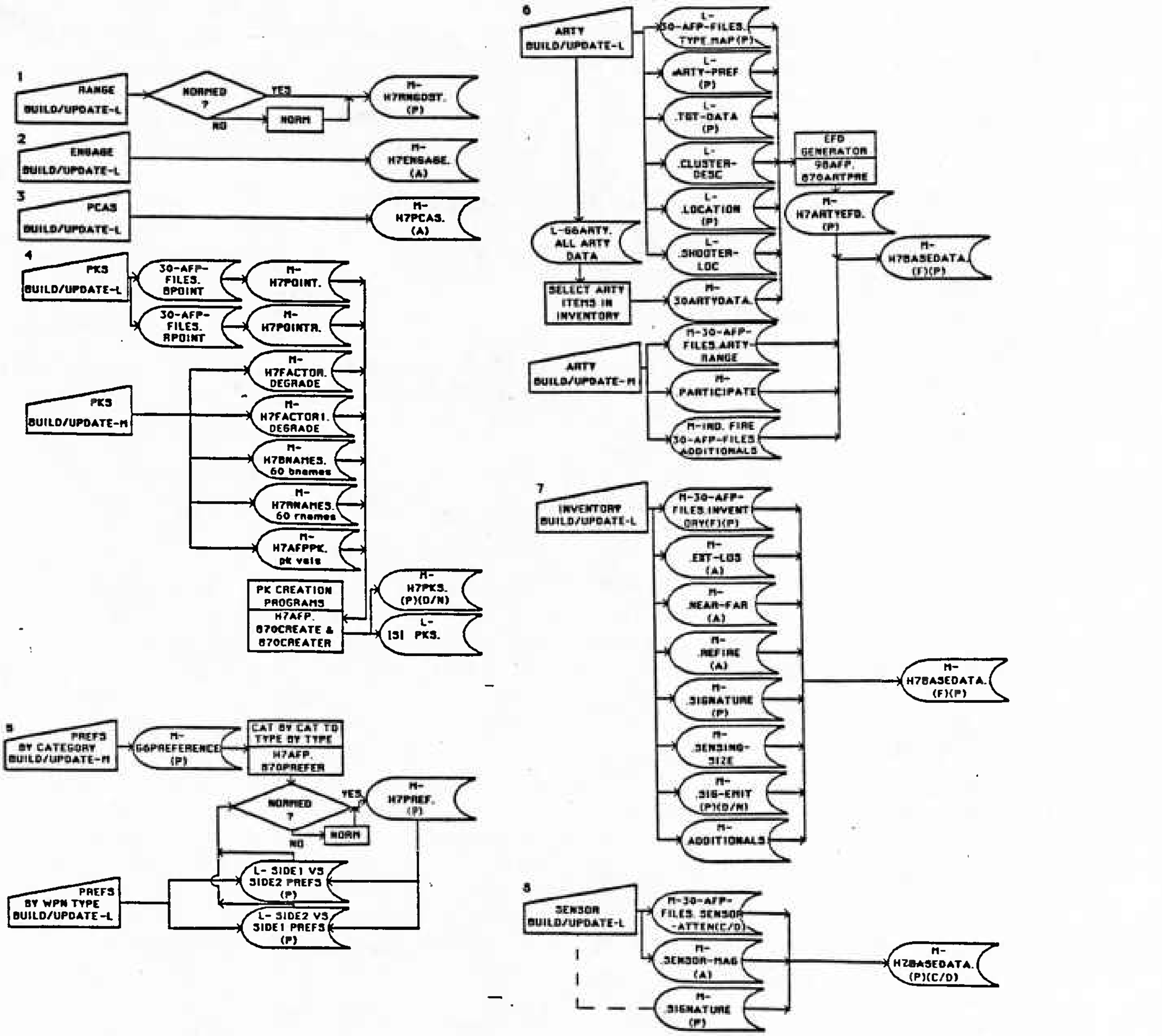


Figure D-3. A Task and File Oriented View of Preparation of Input to the AFP Firepower and Counterfirepower Module

d. The annexes of this appendix are arranged as follows.

(1) Annex I treats the BASEDATA file involving Tasks 6-8.

(2) Annex II treats the PREFerences file involving Task 5. The included data provide the Combat Module with information about the "preferences for targets" that different weapons systems have been assigned by military planners.

(3) Annex III treats the RNGDST file involving Task 1. The included data convey to the Combat Module information about the ranges at which weapon types are intended to engage.

(4) Annex IV treats the PCAS file involving Task 3 and the target categories and crew loss factors assigned to different weapon systems.

(5) Annex V treats the ENGAGE file involving Task 2.

(6) Annex VI treats the PKS file involving Task 4. The included data provide the Combat Module with SSPKs by range for needed combinations of direct fire shooters and targets.

(7) Annex VII describes several programs (preprocessors) that operate between the Combat Module and several of the products of the tasks identified above.

(8) Annex VIII describes many of the details of the program which generates expected kills per artillery round within Task 6 and is directly related to the Task 6 portions of Annex I.

e. It is discouragingly easy to go astray in the preparation of input data. Unfortunately, the input data steps cannot be approached as mutually exclusive tasks. As a general rule, the overburdened analysts must keep all tasks and data in mind at all times--well, almost. A weapon with all other data but invalid sensor specification is doomed to never get off a first shot. Inasmuch as the default refire time is "large," a "weapon" can become little more than someone else's target unless it is given its correct refire time. It may not be obvious from Figure D-3, but both Blue and Red data must be provided. It should be easy to remember this requirement during the initial preparation of a data base. However, experience has shown how easy it is to forget to make two-sided changes during updates. Someone may decide to extend the variety of targets admissible to a given shooter type, insert that shooter's SSPKs against those targets, and forget to let those targets become "symmetric" shooters as well. The inclusion of both labeled and unlabeled versions of many files has made data preparation easier than it once was. But it would be grossly misleading to describe data preparation as a user-friendly process. There is even a pitfall in human-readableness--it is possible to update the labeled version of a file and forget to execute the utility program required to convert the new human-readable version into a new unlabeled machine version.

D-3. OUTPUT. The output of the AFP Combat Module falls into two broad categories. First, each Combat Module run produces three data elements (or files) usually referred to as TTLOS, ALLOC, and FALLOC. (In practice, each element is given a unique name identifying the force, year, combat environment, and replication.) Those elements exist in order to transfer information about weapon allocations and kills by type-on-type engagement and by "day" to the AFP CBT/CS/CSS Merge Module for development of partial combat potentials. Second those same elements plus several others with detailed shot and range results are processed optionally within a Combat Module run to produce human-readable summaries of type-on-type weapon allocations and losses in the form of allocation and killer/victim scoreboards or much more highly detailed reports. All the human-readable output is available primarily for diagnostic review. Routine production of all the reports is impractical; a full set for 10 replications of each of 16 combat environments would require over 100 boxes of computer paper! A (26) full set of TTLOS, ALLOC, and FALLOC data consists of 480 uniquely named elements; in the interest of other computer users and the site files manager, these elements are retained no longer than absolutely necessary.

a. Notes on ALLOC, FALLOC, and TTLOS Files

(1) File Name: ALLOC **Unit Number:** 3 (1U9)

(a) Type of File: direct, formatted

(b) Description: This file contains the numbers of weapons allocated to each type-versus-type conflict at the start of each day. Under some circumstances, the Combat Module may not write some records to the file, which will cause I/O errors when another program attempts to read those records. An error of this type can be intercepted by use of the ERR= clause in the READ statement, and one can confirm that it was due to an attempt to read a dummy record by testing that IOC() is equal to 1053. In such a case, processing should proceed as if all the data in the record were zero.

(c) Pointer Calculation: pointer = repsx + (daysx-1) * ireps,
where repsx = the combat module internal replication number (1)
daysx = the day number (1-2)
ireps = the total number of internal replications (1)

(d) Written by: OUTPT in Main Program of the Combat Module

(e) Read by: ALLOCN, GETLOS in postprocessors

(f) **Description of Fields:**

Name	Format	Type	Description
FORCES	500(500I6)	1	The FORCE array, dimensioned (2,60,60) and written in row

FORCES(sidex, typel, type2) = Major order (rightmost subscript varies most rapidly): the number of weapons of weapons of type "typel" on side "sidex" allocated against weapons of type "type2" on the other side

(2) **File Name:** FALLOC **Unit Number:** 4 (IU10)

(a) **Type of File:** direct, formatted

(b) **Description:** This file contains the participation of weapons in each type-versus-type conflict at the start of each day. Under some circumstances, the Combat Module may not write some records to the file, which will cause I/O errors when another program attempts to read those records. An error of this type can be intercepted by use of the ERR= clause in the READ statement, and one can confirm that it was due to an attempt to read a dummy record by testing that IOC() is equal to 1053. In such a case, processings should proceed as if all the data in the record were zero.

(c) **Pointer Calculation:** pointer = repsx + (daysc-1) * (typs1x-1) * n101 where repsx = the combat module internal replication number (1)
daysx = the day number (1-2)
typs1x = the number of the weapon on side 1
n100 = 1
n101 = 2

(d) **Written by:** MAIN in Main Program of Combat Module

(e) **Read by:** DPART, GETLOS in postprocessors

(f) **Description of Fields:**

Name	Format	Type	Description
D	500(500I6)	1	The array D, dimensioned (2,60) and written in row major order (rightmost subscript varies most rapidly)

D(sidex, typs2x) = the participation of the weapon on side "sidex" in the conflict between weapon type "typs1x" (on side 1) and weapon type "typs2x"(on side 2).

(3) **File Name:** TTLOS **Unit Number:** 9 (IU13)

(a) **Type of File:** sequential, unformatted (binary)

(b) **Description:** This file contains the number of weapons lost in a type-versus-type conflict due to both direct and indirect fire, subdivided by range band. The programs which read this file require the records in a different order than that in which they were originally written, so a preliminary sort is required. If a record would contain only zero losses (i.e., if the TTLOS array described below contains only zeros), the record is not written to the file, in order to conserve file space.

(c) **Written by:** MAIN in Main Program of Combat Module

(d) **Read by:** DIRLOS, INDLOS in postprocessors

(e) **Description of Fields:**

Name	**Word**	**Type**	**Description**
RESPX	1	I	Combat Module internal replication number, normally 1
DAYSX	2	I	The day in which the losses occurred (1-2)
TYPS1X	3	I	Side 1 weapon number
TYPS2X	4	I	Side 2 weapon number
TTLOS	5-136	I	The TTLOS array, dimensioned (2,11,1,6) and written in column major order (leftmost subscript varies most rapidly)

TTLOS(sidex,1,envirx,rangex) = the number of direct fire losses to side "sidex" in the type-versus-type conflict in environment "envirx" (always 1) in range band "rangex" (1-6).

TTLOS (sidex,1 + iext,envirx, rangex) = the number of indirect fire losses to side "sidex" in the type-versus-type conflict in environment "envirx" (always 1) in range band "rangex" (1-6) due to indirect fire shooter "iext" (1-10).

b. Notes on EXT, SHOTS, and TIMES Files

(1) **File Name:** EXT **Unit Number:** 24 (IU5)

(a) **Type of File:** sequential, unformatted (binary)

(b) Description: This file contains the allocations of indirect fire weapons to individual conflicts, and their kills. In order to conserve file space, records which would contain only zero allocations and kills are not written to the file. When the file is first written, the records are not in the order required by the routines which read the file, so a preliminary sort is required.

(c) Written by: MAIN in Main Program of the Combat Module

(d) Read by: INDALC in postprocessor

(e) Description of Fields:

Name	Word	Type	Description
REPSX	1	I	Combat Module internal replication number (always 1)
DAYSX	2	I	The day (1-2)
TYPS1X	3	I	Side 1 weapon number (1-60)
TYPS2X	4	I	Side 2 weapon number (1-60)
EXTN	5-604	I	The EXTN array, dimensioned (2,10,1,6,5) and written in column major order (leftmost) subscript varies most rapidly)
EXTK	605-1204	I	The EXTK array, dimensioned (2,10,1,6,5) written in column major order (leftmost subscript varies most rapidly)

EXTN(sidex,iex,envirx,rangex,cnfx) = the number of indirect fire weapons of type "iex" (1-10) allocated on side "sidex" (1-2) in environment "envirx" (always 1), range band "rangex" (1-6), conflict "cnfx" (1-5).

EXTK(sidex,iex,envirx,rangex,cnfx) = the number of kills by indirect fire weapons of type "iex" (1-10) ON SIDE "sidex" (1-2) in environment "envirx" (always 1), range band "rangex" (1-6), conflict "cnfx" (1-5).

(2) File Name: SHOTS **Unit Number:** 33 (IUOUT3)

(a) Type of File: sequential, unformatted (binary)

(b) Description: This file contains the numbers of shots fired by both direct and indirect fire weapons in each type-versus-type conflict, by range band. In order to conserve file space, records which would contain only zero shot counts are not written to the file. When it is initially written, the file is not in the order required by the routines which read it, so a preliminary sort is necessary.

(c) Written by: MAIN in Main Program of the Combat Module

(d) **Read by:** DIRSHT, INDSHT in postprocessors

(e) **Description of Fields:**

Name	Word	Type	Description
REPSX	1	I	The Combat Module internal replication number (always 1)
DAYSX	2	I	The day (1-2)
TYPS1X	3	I	The side 1 weapon number (1-60)
TYPS2X	4	I	The side 2 weapon number (1-60)
ESHOTS	5-124	I	The ESHOTS array, dimensioned (2,10,1,6) and written in column major order (leftmost subscript varies most rapidly)
SHOTS	125-136	I	The SHOTS array, dimensioned (2,1,6) and written in column major order (leftmost subscript varies most rapidly)

ESHOTS(sidex,iex,envirx,rangex) = the number of shots fired by indirect fire shooter number "iex" (1-10) on side "sidex" (1-2) in environment "envirx" (always 1) on the conflict in range band "rangex" (1-6).

SHOTS(sidex,envirx,rangex) = the number of shots fired by the direct fire shooter on side "sidex" in environment "envirx" (always 1), range band "rangex" (1-6).

(3) **File Name:** TIMES **Unit Number:** 32 (IU15)

(a) **Type of File:** sequential, unformatted (binary)

(b) **Description:** This file contains statistics on detection and refire times for type-versus-type conflicts. In order to conserve file space, records which would contain only zero statistics are not written to the file. When the file is first written it is not in the order required by the routines which read it, so a preliminary sort is necessary.

(c) **Written by:** MAIN in Main Program of Combat Module

(d) **Read by:** DUELTM in postprocessors

(e) **Description of Fields:**

Name	Word	Type	Description
REPSX	1	I	The Combat Module internal replication number (always 1)
DAYSX	2	I	The day (1-2)
TYPS1X	3	I	The Side 1 weapon number (1-60)
TYPS2X	4	I	The Side 2 weapon number (1-60)
NOCONF	5	I	The number of conflicts which took place (1-5)
TMS	6-305	R	The TMS array, dimensioned (2,1,6,5,5) and written in column major order (leftmost subscript varies most rapidly)

In the following, "sidex" is the shooting side (1-2), "envirx" the environment (always 1), "rangex" the range band (1-6), and "cnfx" the cnflict number (1-5):

TMS(sidex,envirx,rangex,cnfx,1) = the number of detections attempted by the shooting side
TMS(sidex,envirx,rangex,cnfx,2) = the number of successful detections
TMS(sidex,envirx,rangex,cnfx,3) = the total dectection time (minutes)
TMS(sidex,envirx,rangex,cnfx,4) = the number of refires
TMS(sidex,envirx,rangex,cnfx,5) = the total refire time (minutes)

c. Allocation "Scoreboard." Figures D-4a and D-4b are a "cut and pasted" example from a full-scale allocation report or "scoreboard." The full-scale report fills several pages. The example shown in Figures D-4 lacks most of the columns corresponding to Red weapon types; most of the columns have been cut out with the remaining columns pasted together. A full allocation report usually corresponds to a single run of the Combat Module (on option, a report summing results over two or more runs may be produced.) A report consists of two main parts: a weapon-by-weapon section and a category-by-category summary section.

```
NUMBER OF INDIVIDUALS IN A GROUP =  10.
NUMBER OF BLUE GROUPS =    8
NUMBER OF RED GROUPS  =    5
BLUE GROUPED TYPES:
   1    2    3    4    5    6   42   44
RED GROUPED TYPES:
   1    2    3   42   44
```

ALLOCATION REPORT FOR JM84 ENV09 SEED 2

	15480 R01 RTRP DEEP	4800 R02 AKS-74	870 R03 7.60GM	426 R16 BMP2M	243 R21 T55	39 R22 T62	126 R23 T64	270 R24 T72	78 R25 T80	48 R26 ZSU-23	RED TOTAL DIR	RED TOTAL
3680 B01 BTRP DEEP	22060 6000	. .	. .	. .	. .	. .	. .	. .	. .	. .	7340	7340
540 B02 M-16	. .	1340 300	280 70	16 90	1 .	. .	1 .	2 .	. .	1 .	840	840
290 B03 7.62GM	. .	830 190	170 40	9 60	1 .	. .	. .	1 .	. .	. .	540	540
470 B05 M203	. .	1010 140	210 30	25 180	2 .	. .	1 .	2 .	. .	. .	780	780
360 B06 SAW	. .	1030 240	220 60	12 70	2 .	. .	. .	2 .	. .	. .	670	670
180 B11 DRAGON	. .	480 4	90 1	69 33	62 57	11 10	32 29	71 64	20 18	2 .	280	280
60 B12 ITV-2	. .	180 2	40 .	25 15	19 18	3 4	10 9	21 21	7 6	2 .	108	108
180 B13 VIPER	. .	590 5	120 2	82 40	78 73	13 13	40 38	89 83	26 24	2 .	358	358
260 B16 IFV	. .	1040 20	220 4	187 87	68 39	11 7	36 20	79 45	23 13	. .	487	487
114 B17 CFV	. .	470 9	100 2	83 39	31 17	6 3	17 9	36 20	11 6	. .	218	218
112 B22 M60A3	. .	80 5	20 1	50 22	60 36	10 6	31 19	69 42	20 12	. .	213	213
168 B23 M1	. .	120 9	20 2	75 34	92 55	15 9	48 29	104 63	30 18	. .	326	326
36 B26 DIVAD	. .	. .	. .	2 14	. .	. .	. .	. .	. .	. .	67	67
79 B31 STINGER	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .	158	158
24 B32 CHAP1	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .	48	48
42 B36 AH-1S	. .	130 5	30 2	6 8	12 14	2 2	7 7	13 16	4 5	83 3	79	79
24 B41 A10	. .	. .	. .	1 3	1 1	. .	. 1	1 2	. .	5 .	36	36
470 B42 LT VEH D	2820 760	. .	. .	. .	. .	. .	. .	. .	. .	. .	920	920
20 B43 HVY ARM D	1200 32	. .	. .	. .	. .	. .	. .	. .	. .	. .	38	38
760 B44 LT ARM D	4540 1240	. .	. .	. .	. .	. .	. .	. .	. .	. .	1500	1500
54 B48 OH-58	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .	. .
66 B53 M4.2S	140 26	. .	. .	. .	. .	. .	. .	. .	. .	. .	38	38
72 B56 H155S	140 28	. .	. .	. .	. .	. .	. .	. .	. .	. .	56	56
12 B57 M203S	20 4	. .	. .	. .	. .	. .	. .	. .	. .	. .	8	8
9 B58 MLRS	20 4	. .	. .	. .	. .	. .	. .	. .	. .	. .	6	6
BLUE TOTAL DIR	30940 .	7300 .	1520 .	642 .	429 .	71 .	223 .	490 .	141 .	95 .	. .	. .
BLUE TOT	30940 .	7300 .	1520 .	642 .	429 .	71 .	223 .	490 .	141 .	95 .	. .	. .

Figure D-4a. Sample Extract from Weapon-by-weapon Type Section of AFP Weapon Allocation Report

	5922 RED SMALL ARMS	SEED 2 522 RED APC	1011 RED ATGM	426 RED IFV W/ATGM	756 RED ARM VEH	48 RED FWD AA	378 RED SAM	48 RED ATK HELO	24 RED FWD ACFT	183 RED MORTARS	414 RED ARTY	18357 RED DEEP	RED TOTAL
11660 BLUE SMALL ARMS	5301 1520	241 730	19 180	62 400	15 .	1 .	. .	. .	. .	. .	. .	. .	2830
420 BLUE ATGM	1566 18	32 25	278 146	176 88	502 467	6 .	. .	8 2	. .	. .	. .	. .	746
374 BLUE IFV W/ATGM	1906 49	667 153	479 196	270 126	318 179	. .	. .	31 2	4 .	. .	. .	. .	705
280 BLUE ARM VEH	251 24	24 19	743 147	125 56	479 289	. .	. .	27 4	13 .	. .	. .	. .	539
36 BLUE FWD AA	. .	6 22	. .	2 14	. .	. .	. .	4 26	2 5	. .	. .	. .	67
103 BLUE SAM	. .	. .	. .	. .	. .	. .	. .	17 158	14 48	. .	. .	. .	206
42 BLUE ATK HELO	167 10	22 8	. .	6 8	38 44	83 3	254 6	. .	. .	. .	. .	. .	79
24 BLUE FWD ACFT	. .	2 1	2 2	1 3	2 4	5 .	496 24	. .	. .	. .	2 2	. .	36
66 BLUE MORTARS	. .	. .	. .	. .	. .	. .	. .	. .	. .	30 6	26 .	160 32	38
93 BLUE ARTY	. .	. .	. .	. .	. .	. .	. .	. .	3 .	8 4	110 22	200 44	70
77 BLUE ASSETS	. .	. .	. .	. .	. .	. .	. .	. .	2 .	. .	. .	. .	. .
4930 BLUE DEEP	. .	. .	. .	. .	. .	. .	. .	. .	. .	70 .	188 20	36314 9778	9798
BLUE TOTAL	9191 .	994 .	1521 .	642 .	1354 .	95 .	250 .	87 .	38 .	108 .	326 .	36674 .	

Figure D-4b. Sample Extract from Weapon Category-by-category Section of AFP Weapon Allocation Report

(1) The weapon-by-weapon type section (Figure D-4a) contains a separate row for each Blue weapon type and a separate column for each Red weapon type. The intersection of a row and a column includes two entries. An entry of "." represents a zero-value. Consider two entries symbolized by:

r(i,j)	the total number over 2 days of Red weapons of type j engaging Blue weapons of type i
b(i,j)	the total number over 2 days of Blue weapons of type i engaging Red weapons of type j

Any weapons that survives the first AFP "day" may be allocated again on the following day. Hence, in the standard 2-day Combat Module run, there may be up to twice as many allocations as there are weapons. Any weapon attrition on the first day will, among other things, reduce allocations on the second day.

(2) The rows and columns of the weapon-by-weapon type section of an allocation report are labelled.

(a) Columns possess labels of the form:

N(j) Rj
Name(j)

where N(j)	is the quantity of Red weapon type j in the threat starting inventory
Rj	is the AFP weapon type identification number
Name(j)	is a short nomenclature for Red weapon type j

(b) Rows of the weapon-by-weapon type section of an allocation report possess labels of the form:

N(i) Bi Name(i)

where N(i)	is the quantity of Blue weapon
Bi	is the AFP weapon type identification number
Name(i)	is a short nomenclature for Blue weapon type i

(c) The two rightmost columns of the weapon-by-weapon type section are labelled "RED TOTAL DIR" and "RED TOTAL." These labels are misleading. The allocation report applies labels first devised for the killer/victim (K/V) scoreboard, where the labels make sense. The allocation report was something of a killer/victim afterthought and was implemented in the most economical way at obvious expense in the labelling. All the Red entries are "." for zero. The lower members of entry pairs are the only ones that can be nonzero. These lower values are the total allocations of the Blue weapon type across the entire row. (Note that the row totals shown in Figures D-4 may greatly exceed the sums of values to their left because many columns were deleted in the cut and paste construction of the figure.)

(d) The two lowest rows of the weapon-by-weapon type section are labelled "BLUE TOTAL DIR" andd "BLUE TOTAL." Just as for the two rightmost columns, the two lowest row labels are misleading. The values shown are the sums of Red weapon allocations in the columns above.

(3) The upper left corner of an allocation report contains a key to a special feature of the Combat Module. The feature is usually referred to as "super-trooping." Most weapons are represented individually within the Combat Module. However, some very numerous weapons are aggregated in order to save computer time. Weapons so grouped are said to be super-trooped. In AFP practice to date, super-trooped weapons are grouped by tens. In Figure D-4a, the key specifies that a super-troop consists of 10 weapons. Eight of the Blue and five of the Red weapon types were super-trooped. Next, the AFP weapon type identifier numbers are given for the Blue and Red super-trooped weapons. Note that the entries in the main parts of an allocation report are adjusted to give the correct counts of individual (unsuper-trooped) weapons as appropriate.

(4) Just below the super-troop key in an allocation report, the report itself is identified. In the example of Figure D-4a, the report represents results for the Blue force "JM84" in combat environment "09" for the second replication (SEED 2).

(5) The weapon category-by-category section (Figure D-4b) of an allocation report is similar in format to the weapon-by-weapon type section described above. The category-by-category section gives subtotals by weapon categories. For example, tanks of all types are lumped together as "ARM VEH." Weapons of all types in the deep area are included in the special category, "DEEP."

d. Killer/Victim Scoreboard. Figures D-5a and D-5b are a "cut and pasted" example extracted from a full-scale killer/victim scoreboard report. The full-scale report fills several pages. The example shown in Figures D-5 lacks most of the columns corresponding to Red weapon types; most of the columns have been cut out of the original report, and the remaining columns have simply been pasted together. A full killer/victim scoreboard corresponds to a single run of the Combat Module. The format of a killer/victim scoreboard is similar to that of an allocation report; indeed, both reports are printed by the same output program. However, the numbers of columns and rows normally is not the same in allocation and killer/victim reports. The allocation report includes columns and rows for all weapons allocated, usually all weapons included in the starting inventories. The killer/victim scoreboard includes columns and rows only if the corresponding weapon types score at least one kill or suffer at least one loss. Any weapon whose column or row would contain all "."/s (i.e., zeros) is suppressed by the killer/victim scoreboard print program. A killer/victim scoreboard consists of two main parts: a weapon-by-weapon type section and a weapon category-by-category summary section. The formats of the two sections are similar. The killer/victim scoreboard is designed to show the losses of each weapon type to each weapon type. Such losses are also summarized by weapon category. On option, a report summarizing results over two or more Combat Module runs may be produced.

(1) The weapon-by-weapon type section (Figure D-5a) contains a separate row for each Blue weapon type and a separate column for each Red weapon type. The intersection of a row and a column corresponds to engagements between two weapon types and consists of two entries. An entry of "." represents a zero value. The appearance of two "."/s, however, is ambiguous in the sense that zero losses may have occurred because the two weapon types engaged without loss or did not engage at all. The ambiguity may be removed by reference to the corresponding elements in the allocation report. Consider two entries symbolized by:

r(i,j)	the total number over 2 days of Red weapons of type j lost to Blue weapons of type i
b(i,j)	the total number over 2 days of Blue weapons of type i lost to Red weapons of type j

Any individual weapon can be lost only once. Hence, total losses cannot exceed total inventory. In the allocation report, total allocations may exceed inventory because weapons that survive the first day may be allocated again (and counted again) on the second day.

(2) The rows and columns of the weapon-by-weapon type section are labelled.

(a) Columns possess labels of the form:

N(j) rj
Name(j)

where	N(j)	is the quantity of Red weapon type j in the starting inventory
	Rj	is the AFP weapon type identification number
	Name(j)	is a short nomenclature for Red weapon type j

NUMBER OF INDIVIDUALS IN A GROUP = 10.
NUMBER OF BLUE GROUPS = 8
NUMBER OF RED GROUPS = 5
BLUE GROUPED TYPES:
1 2 3 4 5 6 42 44
RED GROUPED TYPES:
1 2 3 42 44

KV SCOREBOARD FOR JM84 ENV 09 SEED 2

		4800 R02 AKS-74	870 R03 7.60GM	426 R16 BMP2M	243 R21 T55	39 R22 T62	126 R23 T64	270 R24 T72	78 R25 T80	48 R26 ZSU-23	RED TOTAL DIR	RED TOTAL AREA	RED TOTAL
540 B02	M-16	800 210	70 70	. 20	.	.	.	.	.	.	300	.	300
290 B03	7.62GM	830 10	110 30	. 20	.	.	.	.	.	.	60	.	60
470 B05	M203	570 140	70 30	8 .	.	.	.	.	.	.	250	.	250
360 B06	SAW	1030 10	140 50	. 20	.	.	.	.	.	.	80	.	80
180 B11	DRAGON	.	.	54 14	62 10	11 2	31 21	59 57	18 12	.	119	1	120
60 B12	ITV-2	40 .	.	21 1	5 1	.	3 1	4 3	1 .	.	10	6	16
180 B13	VIPER	.	.	.	.	.	.	.	.	.	.	1	1
260 B16	IFV	160 .	.	148 14	7 1	.	4 .	6 2	.	.	29	15	44
114 B17	CFV	.	.	52 6	1 1	.	.	2 1	.	.	11	.	11
112 B22	M60A3	.	.	13 4	5 .	.	2 1	7 1	.	.	19	.	19
168 B23	M1	.	.	7 .	8 .	.	4 1	6 5	2 1	.	15	.	15
36 B26	DIVAD	.	.	2 .	.	.	.	.	.	.	7	.	7
79 B31	STINGER	.	.	.	.	.	.	.	.	.	.	.	.
24 B32	CHAP1	.	.	.	.	.	.	.	.	.	.	.	.
42 B36	AH-1S	130 .	30 .	1 .	5 .	.	2 1	5 2	1 4	2 3	13	.	13
24 B41	A10	.	.	.	.	.	.	.	.	.	16	.	16
66 B53A	M4.2S	30 .	.	4 .	.	.	.	.	.	.	.	.	.
72 B56A	H155S	80 .	.	1 .	.	.	.	.	.	.	.	.	.
BLUE TOTAL DIR		3560 .	420 .	306 .	93 .	11 .	46 .	89 .	22 .	2 .	.	.	.
BLUE TOTAL AREA		110 .	.	5 .	.	.	.	.	.	.	.	.	.
BLUE TOTAL		3670 .	420 .	311 .	93 .	11 .	46 .	89 .	22 .	2 .	.	.	.

Figure D-5a. Sample Extract from Weapon-by-weapon Type Section of AFP Weapon Killer/Victim Scoreboard Report

(b) Rows possess labels of the form:

from:

N(i) Bi
Name(i)

where	N(i)	is the quantity of Blue weapon type i in the starting inventory
	Bi	is the AFP weapon type identification number
	Name(i)	is a short nomenclature for Blue weapon type i

(c) The three rightmost columns of the weapon-by-weapon type section are labeled "RED TOTAL DIR," "RED TOTAL AREA," and "RED TOTAL." The entries in these colulmns show the sums of losses of Blue weapons by Blue weapon type (row). Again there are two entries for each column/row intersection. The upper entry should always be "."; no Red weapons are lost to Red weapons. The column headed "RED TOTAL DIR" displays the numbers of each Blue weapon type (row) lost to Red direct fire weapons of all types. Indirect fire weapons lost to counterbattery fire are included in this "direct fire" column. The column headed "RED TOTAL AREA" displays the numbers of Blue weapons by type (row) lost to Red area fire of all types (i.e., to Red indirect fire weapons firing in the indirect or area fire role). The column headed "RED TOTAL" displays the numbers of Blue weapons by type lost to both direct and indirect (area) Red fires; values shown in this column are the sum of the values (in the same row) in the preceding two columns.

(d) The three lowest rows of the weapon-by-weapon type section are labeled "BLUE TOTAL DIR," "BLUE TOTAL AREA," and "BLUE TOTAL." The entries in these rows show the sums of losses of Red weapons by Red weapon type (column). As usual, there are two entries for each row/column intersection. The lower entry should always be "."; no Blue weapons are lost to Blue weapons. The row labeled "BLUE TOTAL DIR" displays the total numbers of Red weapons of each given type (column) lost to all Blue direct fire weapon types. Red indirect fire weapons lost to Blue counterbattery fire are included in this Blue direct fire row. The row labeled "BLUE TOTAL AREA" displays the numbers of Red weapons of each given type (column) lost to Blue area fire of all types (i.e., to Blue indirect fire weapons firing in the indirect or area fire role). The row labeled "BLUE TOTAL" displays the numbers of Red weapons by type (column) lost to the direct and indirect (area) Blue fires; values shown in this row should be the sum of the values (in the same column) in the preceding two rows.

(3) The upper left corner of a killer/victim scoreboard contains a key to a special feature of the Combat Module. This feature is usually referred to as "super-trooping." Most weapons are represented individually within the Combat Module. However, some very numerous weapons are aggregated to save computer time. Weapons so grouped are said to be super-trooped. In AFP practice to date, super-trooped weapons are grouped by tens. In Figure D-5a, the key specifies that a super-troop consists of 10 weapons. Eight of the Blue and five of the Red weapon types were super-trooped. Next the key gives the weapon type identification numbers of the Blue and Red super-trooped weapons. Entries in the main parts of a killer/victim scoreboard are always adjusted to give the correct counts of individual (unsuper-trooped) weapons as appropriate.

(4) Just below the super-troop key in a killer/victim scoreboard, the report itself is identified. In the example in Figure D-5a, the report represents results for Blue force "JM84" in combat environment "09" for the second replication (SEED 2).

(5) The weapon category-by-category section (Figure D-5b) of a killer/victim scoreboard is similar in format to the weapon-by-weapon type section described above. The category-by-category section gives subtotals by weapon categories. For example, tanks of all types on a side are lumped together as "ARM VEH."

KV SCOREBOARD FOR JM84	ENV 09 5922 RED SMALL ARMS	SEED 2 522 RED APC	1011 RED ATGM	426 RED IFV W/ATGM	756 RED ARM VEH	48 RED FWD AA	378 RED SAM	48 RED ATK HELO	24 RED FWD ACFT	183 RED MORTARS	414 RED ARTY	RED TOTAL
11660 BLUE SMALL ARMS	3800 550	27 80	18 .	8 60	. .	. .	. .	. .	. .	. .	. .	. 690
420 BLUE ATGM	45 .	12 3	10 2	75 15	194 107	. .	. .	. 2	. .	. .	. 8	. 137
374 BLUE IFV W/ATGM	160 .	24 14	52 1	200 20	20 5	. .	. .	. .	. .	. 9	. 6	. 55
280 BLUE ARM VEH	. .	1 .	665 19	20 4	34 9	. .	. .	4 2	. .	. .	. .	. 34
36 BLUE FWD AA	. .	6 2	. .	2 .	. .	. .	. .	4 5	. .	. .	. .	. 7
103 BLUE SAM	. .	. .	. .	. .	. .	. .	. .	5 .	14 .	. .	. .	. .
42 BLUE ATK HELO	166 .	2 2	. .	1 .	13 7	2 3	2 1	. .	. .	. .	. .	. 13
24 BLUE FWD ACFT	. .	. .	. .	. .	. .	. .	. 16	. .	. .	. .	. .	. 16
66 BLUE MORTARS	30 .	11 .	2 .	4 .	. .	. .	. .	. .	. .	. .	. .	. .
93 BLUE ARTY	81 .	5 .	4 .	1 .	. .	. .	. .	. .	. .	. .	. .	. .
BLUE TOTAL	4282 .	88 .	751 .	311 .	261 .	2 .	2 .	13 .	14 .	. .	. .	. .

Figure D-5b. Sample Extract from Weapon Category-by-category Section of AFP Weapon Killer/Victim Scoreboard Report

e. QA Report. Given that the Combat Module decomposes the full battlefield into as many as 60 x 60 type-on-type direct fire engagements (of ordinary or counterbattery versions), the casual analyst might suppose that any one type-on-type engagement be trivial. While it is true that any one type-on-type direct fire engaement is less complicated than the full 60 x 60 set, one engagement is far from trivial. Recall that engagement is preceded by a preference/allocation process to determine just how many of each direct fire weapon type meet in the day's engagement. The weapons so allocated are then assigned to pure duels of one or, at most, two odds classes. The duels are distributed by range. The duels at one range are collected into a conflict. Several conflicts in succession (four has been the standard maximum in work to date) may be represented provided there are survivors of the preceding conflicts. The last conflict on a day may be followed by another day starting with fresh preference/allocation of the preceding day's survivors and continuing through another duel/conflict succession. A conflict may have up to 10 types of indirect fire weapons from each side firing on the opposing direct fire weapon types. The indirect fire weapons too must be allocated to the engagement from a weapons pool. The indirect fire weapons so allocated must be distributed by range of targets. And the firing of up to all 22 weapon types must be sequenced without unacceptable sacrifice of realism. The process is complicated by the treatment of detection in the case of direct fire weapons and by the disregard of detection by indirect fire weapons. The appropriate sensing data must be used for the direct fire weapons. Correct refire rates must be applied for different weapons. Correct SSPKs and fractional kills/round must be used for the direct and indirect fire weapons, respectively. Kills must be tallied. With so much to be considered, it is not easy for an analyst to judge the correctness of results simply by inspecting final weapon and division combat potentials following rollup over 16 combat environments. The analyst's role is made a little easier by an engagement quality assurance reporter (QAREP). The basic reporter displays a useful variety of type-on-type input data and results from a single execution of the Combat Module. The report for a single pairing of direct fire types for four conflicts on each of 2 days fills four pages. Calling for the report on several hundred engagements in each of 16 combat environments generates more paper than anyone can digest. Nevertheless, the report is probably the most powerful single tool available to the AFP analyst.

(1) Figure D-6 reproduces a QAREP for a fictional example of 2 days' engagements between Blue type 23 and Red type 24 weapons. The following paragraphs describe much of the contents of Figure D-6 by way of illustrating many of the points introduced earlier in this appendix and of delving a bit more deeply into some matters.

(2) The heading of Figure D-6 identifies the direct fire weapons involved as Blue type 23 and Red type 24. At user option, short weapon nomenclatures may be included in the QAREP heading line.

(3) The second line shows that both weapon types belong to WEAPON CATEGORY 5, which is usually reserved for tanks.

```
                                        (BLUE 23) VS .          (RED  24)

WEAPON CATEGORY                              5                      5
NUMBER OF WEAPON TYPE                      196                    196
EXTERNAL LOSSES                              0                      0
NEAR/FAR RANGE                     1         6          1           3
REFIRE TIME                               .250                   .250

 DETECTION:
  SENSOR TYPE                      2         1         21           1
  RESOLVABLE CYCLES                       6.00                   6.00
  SENSING SIZE                            2.36                   .840
  LIGHT                                              600.
  CONTRAST                      .400      1.10       .200        1.00
  ATTENUATION                             .280                   .280
  MAGNIFICATION                           3.00                   3.50
  BRIGHTNESS                                         3.00

DISTRIBUTION OF INDIRECT WEAPONS          .000                   .000
EXPECTED FRACTIONAL DAMAGE                .000                   .000

                                250M      500M      1000M      1500M       2000M       2500M
BLUE PK                         .310      .230      .130    6.000-002   3.000-002   3.000-002
RED  PK                         .900      .790      .580      .000        .000        .000
RANGE DISTRIBUTION              .170      .170      .160      .160        .170        .170

ORIGINAL PREFERENCE                       .115                9.400-002
MODIFIED PREFERENCE                    9.453-002              7.626-002
PARTICIPATION                             1.00                   1.00
ENGAGEMENT DURATION                       2.01                   2.01
ENGAGEMENTS/DAY                           1.00                   1.00
# TIMES SURVIVOR IN CONFLICT              4.00                   4.00
CREW LOSSES SUFFERED PER KILL             1.00                   1.00

BLUE INDIRECT REFIRE TIMES        .5000   .5000   .5000  6.6600  6.6600   .5000  1.0000   .1600  6.6600  6.6600
BLUE FRACTION FOR AREA FIRE       .6000   .6000   .6000  1.0000  1.0000   .5000   .5000   .5000  1.0000  1.0000

BLUE INDIRECT KILLS/ROUND
                           1       2       3       4       5       6       7       8       9      10
           250M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
           500M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
          1000M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
          1500M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
          2000M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
          2500M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000

RED  INDIRECT REFIRE TIMES        .5000   .5000   .5000  6.6600  6.6600   .5000  1.0000   .1600  6.6600  6.6600
RED  FRACTION FOR AREA FIRE       .6000   .6000  1.0000  1.0000  1.0000   .5000   .5300   .5300  1.0000  1.0000

RED  INDIRECT KILLS/ROUND
                           1       2       3       4       5       6       7       8       9      10
           250M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
           500M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
          1000M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
          1500M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
          2000M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000
          2500M     .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000 .250000

DAY   1
ALLOCATIONS                                139                    133
PARTICIPATION                              139                    133
EXTERNAL LOSSES                           .000                   .000

                                250M      500M      1000M      1500M      2000M      2500M      TOTAL
  BLUE DIRECT LOSSES              24        23         22          0          0          0         69
  RED  DIRECT LOSSES               6         6          0          0          0          0         12
  BLUE DIRECT SHOTS               16        23          0          0          0          0         39
  RED  DIRECT SHOTS               28        33         28          0          0          0         89

  BLUE INDIRECT LOSSES
                          1       2       3       4       5       6       7       8       9      10    TOTAL
           500M           0       0       0       0       0       1       0       0       0       0       1
          1500M           0       0       0       0       0       4       0       0       0       0       4
          2000M           0       0       0       0       0       3       0       0       0       0       3
          2500M           0       0       0       0       0       6       0       0       0       0       6
          TOTAL           0       0       0       0       0      14       0       0       0       0      14

  RED  INDIRECT LOSSES
                          1       2       3       4       5       6       7       8       9      10    TOTAL
           500M           0       0       0       0       0       1       0       0       0       0       1
          1000M           0       0       0       0       0       2       0       0       0       0       2
          1500M           0       0       0       0       0       5       0       0       0       0       5
          2000M           0       0       0       0       0       5       0       0       0       0       5
          2500M           0       0       0       0       0       6       0       0       0       0       6
          TOTAL           0       0       0       0       0      19       0       0       0       0      19
```

Figure D-6. Example of an AFP System QAREP Report of Combat Module Input and Results for a Single Direct-fire-type on Direct-fire-type Pairing (page 1 of 5 pages)

BLUE INDIRECT SHOTS

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	10.82	.00	.00	.00	.00	10.82
500M	.00	.00	.00	.00	.00	10.82	.00	.00	.00	.00	10.82
1000M	.00	.00	.00	.00	.00	5.42	.00	.00	.00	.00	5.42
1500M	.00	.00	.00	.00	.00	21.68	.00	.00	.00	.00	21.68
2000M	.00	.00	.00	.00	.00	22.30	.00	.00	.00	.00	22.30
2500M	.00	.00	.00	.00	.00	22.00	.00	.00	.00	.00	22.00
TOTAL	.00	.00	.00	.00	.00	93.05	.00	.00	.00	.00	93.05

RED EXPECTED INDIRECT LOSSES

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	2.71	.00	.00	.00	.00	2.71
500M	.00	.00	.00	.00	.00	2.71	.00	.00	.00	.00	2.71
1000M	.00	.00	.00	.00	.00	1.35	.00	.00	.00	.00	1.35
1500M	.00	.00	.00	.00	.00	5.42	.00	.00	.00	.00	5.42
2000M	.00	.00	.00	.00	.00	5.58	.00	.00	.00	.00	5.58
2500M	.00	.00	.00	.00	.00	5.50	.00	.00	.00	.00	5.50
TOTAL	.00	.00	.00	.00	.00	23.26	.00	.00	.00	.00	23.26

RED INDIRECT SHOTS

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	11.29	.00	.00	.00	.00	11.29
500M	.00	.00	.00	.00	.00	11.29	.00	.00	.00	.00	11.29
1000M	.00	.00	.00	.00	.00	5.68	.00	.00	.00	.00	5.68
1500M	.00	.00	.00	.00	.00	22.71	.00	.00	.00	.00	22.71
2000M	.00	.00	.00	.00	.00	22.59	.00	.00	.00	.00	22.59
2500M	.00	.00	.00	.00	.00	23.00	.00	.00	.00	.00	23.00
TOTAL	.00	.00	.00	.00	.00	96.56	.00	.00	.00	.00	96.56

BLUE EXPECTED INDIRECT LOSSES

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	2.82	.00	.00	.00	.00	2.82
500M	.00	.00	.00	.00	.00	2.82	.00	.00	.00	.00	2.82
1000M	.00	.00	.00	.00	.00	1.42	.00	.00	.00	.00	1.42
1500M	.00	.00	.00	.00	.00	5.68	.00	.00	.00	.00	5.68
2000M	.00	.00	.00	.00	.00	5.65	.00	.00	.00	.00	5.65
2500M	.00	.00	.00	.00	.00	5.75	.00	.00	.00	.00	5.75
TOTAL	.00	.00	.00	.00	.00	24.14	.00	.00	.00	.00	24.14

CONFLICT 1

	DUELS	AT	DUELS	AT	BLUE DETECTIONS ATTEMPT	BLUE DETECTIONS SUCCESS	BLUE DETECTIONS AVG TM	BLUE REFIRE AVG TM	RED DETECTIONS ATTEMPT	RED DETECTIONS SUCCESS	RED DETECTIONS AVG TM	RED REFIRE AVG TM	UP BOUND ON SHOTS
250M	22	1: 1	1	2: 1	24.	24.	.055	.248	23.	23.	.064	.207	24.
500M	22	1: 1	1	2: 1	24.	24.	.082	.253	23.	23.	.066	.231	24.
1000M	20	1: 1	1	2: 1	22.	0.	.000	.000	21.	21.	.063	.212	24.
1500M	20	1: 1	1	2: 1	22.	0.	.000	.000	0.	0.	.000	.000	24.
2000M	22	1: 1	1	2: 1	24.	0.	.000	.000	0.	0.	.000	.000	24.
2500M	21	1: 1	1	2: 1	23.	0.	.000	.000	0.	0.	.000	.000	24.

BLUE INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	12.00	.00	.00	.00	.00	12.00

RED INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	12.00	.00	.00	.00	.00	12.00

CONFLICT 2

	DUELS	AT	DUELS	AT	BLUE DETECTIONS ATTEMPT	BLUE DETECTIONS SUCCESS	BLUE DETECTIONS AVG TM	BLUE REFIRE AVG TM	RED DETECTIONS ATTEMPT	RED DETECTIONS SUCCESS	RED DETECTIONS AVG TM	RED REFIRE AVG TM	UP BOUND ON SHOTS
250M	2	1: 3	3	1: 4	5.	5.	.050	.245	18.	18.	.073	.255	40.
500M	2	1: 4	2	1: 5	4.	4.	.205	.164	18.	18.	.047	.000	48.
1500M	19	1: 1	1	2: 1	21.	0.	.000	.000	0.	0.	.000	.000	24.
2000M	18	1: 1	3	2: 1	24.	0.	.000	.000	0.	0.	.000	.000	24.
2500M	20	1: 1	1	2: 1	22.	0.	.000	.000	0.	0.	.000	.000	24.

BLUE INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00

Figure D-6. Example of an AFP System QAREP Report of Combat Module Input and Results for a Single Direct-fire-type on Direct-fire-type Pairing
(page 2 of 5 pages)

```
                            TOTAL     .00     .00     .00     .00     .00   10.00     .00     .00     .00     .00   10.00

RED  INDIRECT SHOOTERS                  1       2       3       4       5       6       7       8       9      10   TOTAL
                             250M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                             500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            1500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2000M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            TOTAL     .00     .00     .00     .00     .00   10.00     .00     .00     .00     .00   10.00

CONFLICT   3
                                   ------------- BLUE ------------   ------------- RED -------------
                                   ------ DETECTIONS -----  REFIRE   ------ DETECTIONS -----  REFIRE   UP BOUND
          DUELS   AT     DUELS  AT  ATTEMPT SUCCESS AVG TM  AVG TM   ATTEMPT SUCCESS AVG TM  AVG TM   ON SHOTS
   1500M   18    1: 1      1   2: 1     20.      0.   .000    .000        0.      0.   .000    .000      24.
   2000M   17    1: 1      3   2: 1     23.      0.   .000    .000        0.      0.   .000    .000      24.
   2500M   15    1: 1      3   2: 1     21.      0.   .000    .000        0.      0.   .000    .000      24.

BLUE INDIRECT SHOOTERS                  1       2       3       4       5       6       7       8       9      10   TOTAL
                            1500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2000M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            TOTAL     .00     .00     .00     .00     .00    6.00     .00     .00     .00     .00    6.00

RED  INDIRECT SHOOTERS                  1       2       3       4       5       6       7       8       9      10   TOTAL
                            1500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2000M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            TOTAL     .00     .00     .00     .00     .00    6.00     .00     .00     .00     .00    6.00

CONFLICT   4
                                   ------------- BLUE ------------   ------------- RED -------------
                                   ------ DETECTIONS -----  REFIRE   ------ DETECTIONS -----  REFIRE   UP BOUND
          DUELS   AT     DUELS  AT  ATTEMPT SUCCESS AVG TM  AVG TM   ATTEMPT SUCCESS AVG TM  AVG TM   ON SHOTS
   1500M   17    1: 1      1   2: 1     19.      0.   .000    .000        0.      0.   .000    .000      24.
   2000M   16    1: 1      3   2: 1     22.      0.   .000    .000        0.      0.   .000    .000      24.
   2500M   13    1: 1      3   2: 1     19.      0.   .000    .000        0.      0.   .000    .000      24.

BLUE INDIRECT SHOOTERS                  1       2       3       4       5       6       7       8       9      10   TOTAL
                            1500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2000M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            TOTAL     .00     .00     .00     .00     .00    6.00     .00     .00     .00     .00    6.00

RED  INDIRECT SHOOTERS                  1       2       3       4       5       6       7       8       9      10   TOTAL
                            1500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2000M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            2500M     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00    2.00
                            TOTAL     .00     .00     .00     .00     .00    6.00     .00     .00     .00     .00    6.00

RED:BLUE EXCHANGE RATIO                      .145            2.23

DAY    2
ALLOCATIONS                                    76              90
PARTICIPATION                                  76              90
EXTERNAL LOSSES                              .000            .000

                          250M      500M      1000M      1500M      2000M      2500M      TOTAL
 BLUE DIRECT LOSSES         14        14         12          0          0          0         40
 RED  DIRECT LOSSES          1         5          0          0          0          0          6
 BLUE DIRECT SHOTS           6        13          0          0          0          0         19
 RED  DIRECT SHOTS          16        17         24          0          0          0         57

 BLUE INDIRECT LOSSES                   1       2       3       4       5       6       7       8       9      10   TOTAL
                            1500M       0       0       0       0       0       3       0       0       0       0       3
                            2000M       0       0       0       0       0       5       0       0       0       0       5
                            TOTAL       0       0       0       0       0       8       0       0       0       0       8

 RED  INDIRECT LOSSES                   1       2       3       4       5       6       7       8       9      10   TOTAL
                             500M       0       0       0       0       0       2       0       0       0       0       2
                            1000M       0       0       0       0       0       3       0       0       0       0       3
                            1500M       0       0       0       0       0       3       0       0       0       0       3
                            2000M       0       0       0       0       0       2       0       0       0       0       2
                            2500M       0       0       0       0       0       6       0       0       0       0       6
                            TOTAL       0       0       0       0       0      16       0       0       0       0      16

 BLUE INDIRECT SHOTS                    1       2       3       4       5       6       7       8       9      10   TOTAL
                             250M     .00     .00     .00     .00     .00    4.53     .00     .00     .00     .00    4.53
                             500M     .00     .00     .00     .00     .00    9.07     .00     .00     .00     .00    9.07
                            1000M     .00     .00     .00     .00     .00    4.48     .00     .00     .00     .00    4.48
                            1500M     .00     .00     .00     .00     .00   17.92     .00     .00     .00     .00   17.92
                            2000M     .00     .00     .00     .00     .00   16.59     .00     .00     .00     .00   16.59
```

Figure D-6. Example of an AFP System QAREP Report of Combat Module Input and Results for a Single Direct-fire-type on Direct-fire-type Pairing
(page 3 of 5 pages)

	1	2	3	4	5	6	7	8	9	10	TOTAL
2500M	.00	.00	.00	.00	.00	16.59	.00	.00	.00	.00	16.59
TOTAL	.00	.00	.00	.00	.00	69.19	.00	.00	.00	.00	69.19

RED EXPECTED INDIRECT LOSSES

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	1.13	.00	.00	.00	.00	1.13
500M	.00	.00	.00	.00	.00	2.27	.00	.00	.00	.00	2.27
1000M	.00	.00	.00	.00	.00	1.12	.00	.00	.00	.00	1.12
1500M	.00	.00	.00	.00	.00	4.48	.00	.00	.00	.00	4.48
2000M	.00	.00	.00	.00	.00	4.15	.00	.00	.00	.00	4.15
2500M	.00	.00	.00	.00	.00	4.15	.00	.00	.00	.00	4.15
TOTAL	.00	.00	.00	.00	.00	17.30	.00	.00	.00	.00	17.30

RED INDIRECT SHOTS

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	5.33	.00	.00	.00	.00	5.33
500M	.00	.00	.00	.00	.00	10.67	.00	.00	.00	.00	10.67
1000M	.00	.00	.00	.00	.00	5.33	.00	.00	.00	.00	5.33
1500M	.00	.00	.00	.00	.00	22.59	.00	.00	.00	.00	22.59
2000M	.00	.00	.00	.00	.00	21.33	.00	.00	.00	.00	21.33
2500M	.00	.00	.00	.00	.00	21.33	.00	.00	.00	.00	21.33
TOTAL	.00	.00	.00	.00	.00	86.59	.00	.00	.00	.00	86.59

BLUE EXPECTED INDIRECT LOSSES

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	1.33	.00	.00	.00	.00	1.33
500M	.00	.00	.00	.00	.00	2.67	.00	.00	.00	.00	2.67
1000M	.00	.00	.00	.00	.00	1.33	.00	.00	.00	.00	1.33
1500M	.00	.00	.00	.00	.00	5.65	.00	.00	.00	.00	5.65
2000M	.00	.00	.00	.00	.00	5.33	.00	.00	.00	.00	5.33
2500M	.00	.00	.00	.00	.00	5.33	.00	.00	.00	.00	5.33
TOTAL	.00	.00	.00	.00	.00	21.65	.00	.00	.00	.00	21.65

CONFLICT 1

					BLUE DETECTIONS			BLUE REFIRE	RED DETECTIONS			RED REFIRE	UP BOUND
	DUELS	AT	DUELS	AT	ATTEMPT	SUCCESS	AVG TM	AVG TM	ATTEMPT	SUCCESS	AVG TM	AVG TM	ON SHOTS
250M	11	1: 1	3	1: 2	14.	14.	.066	.285	17.	17.	.037	.248	24.
500M	11	1: 1	3	1: 2	14.	14.	.041	.198	17.	17.	.070	.236	24.
1000M	10	1: 1	2	1: 2	12.	0.	.000	.000	14.	14.	.061	.268	24.
1500M	10	1: 1	2	1: 2	12.	0.	.000	.000	0.	0.	.000	.000	24.
2000M	10	1: 1	2	1: 2	12.	0.	.000	.000	0.	0.	.000	.000	24.
2500M	10	1: 1	2	1: 2	12.	0.	.000	.000	0.	0.	.000	.000	24.

BLUE INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	12.00	.00	.00	.00	.00	12.00

RED INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
250M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	12.00	.00	.00	.00	.00	12.00

CONFLICT 2

					BLUE DETECTIONS			BLUE REFIRE	RED DETECTIONS			RED REFIRE	UP BOUND
	DUELS	AT	DUELS	AT	ATTEMPT	SUCCESS	AVG TM	AVG TM	ATTEMPT	SUCCESS	AVG TM	AVG TM	ON SHOTS
500M	3	1: 4			3.	3.	.025	.321	12.	12.	.052	.131	40.
1500M	8	1: 1	2	1: 2	10.	0.	.000	.000	0.	0.	.000	.000	24.
2000M	7	1: 1	3	1: 2	10.	0.	.000	.000	0.	0.	.000	.000	24.
2500M	11	1: 1	1	1: 2	12.	0.	.000	.000	0.	0.	.000	.000	24.

BLUE INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	8.00	.00	.00	.00	.00	8.00

RED INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	8.00	.00	.00	.00	.00	8.00

Figure D-6. Example of an AFP System QAREP Report of Combat Module Input and Results for a Single Direct-fire-type on Direct-fire-type Pairing
(page 4 of 5 pages)

CONFLICT 3

	DUELS	AT	DUELS	AT	BLUE DETECTIONS ATTEMPT	BLUE DETECTIONS SUCCESS	BLUE DETECTIONS AVG TM	BLUE REFIRE AVG TM	RED DETECTIONS ATTEMPT	RED DETECTIONS SUCCESS	RED DETECTIONS AVG TM	RED REFIRE AVG TM	UP BOUND ON SHOTS
1500M	9	1:1	1	1:2	10.	0.	.000	.000	0.	0.	.000	.000	24.
2000M	9	1:1	3	1:2	10.	0.	.000	.000	0.	0.	.000	.000	24.
2500M	10	1:1	1	2:1	12.	0.	.000	.000	0.	0.	.000	.000	24.

BLUE INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	6.00	.00	.00	.00	.00	6.00

RED INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	6.00	.00	.00	.00	.00	6.00

CONFLICT 4

	DUELS	AT	DUELS	AT	BLUE DETECTIONS ATTEMPT	BLUE DETECTIONS SUCCESS	BLUE DETECTIONS AVG TM	BLUE REFIRE AVG TM	RED DETECTIONS ATTEMPT	RED DETECTIONS SUCCESS	RED DETECTIONS AVG TM	RED REFIRE AVG TM	UP BOUND ON SHOTS
1500M	7	1:1	2	1:2	9.	0.	.000	.000	0.	0.	.000	.000	24.
2000M	4	1:1	4	1:2	8.	0.	.000	.000	0.	0.	.000	.000	24.
2500M	8	1:1	2	2:1	12.	0.	.000	.000	0.	0.	.000	.000	24.

BLUE INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	6.00	.00	.00	.00	.00	6.00

RED INDIRECT SHOOTERS

	1	2	3	4	5	6	7	8	9	10	TOTAL
1500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2000M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
2500M	.00	.00	.00	.00	.00	2.00	.00	.00	.00	.00	2.00
TOTAL	.00	.00	.00	.00	.00	6.00	.00	.00	.00	.00	6.00

RED:BLUE EXCHANGE RATIO .125 1.82

Figure D-6. Example of an AFP System QAREP Report of Combat Module Input and Results for a Single Direct-fire-type on Direct-fire-type Pairing (page 5 of 5 pages)

(4) **NUMBER OF WEAPON TYPE** displays the numbers of the two weapon types input in total to the Combat Module. In general, these numbers are larger than those allocated to this particular type-on-type engagement.

(5) **EXTERNAL LOSSES** shows how many weapons of each type are to be regarded as "lost" prior to any engagement. Here neither side suffers any external losses.

(6) **NEAR/FAR RANGE** shows the ranges within which the corresponding weapon types are to be permitted to fire. Recall the Combat Module represents six ranges from 250 to 2,500 meters, or from 250 to 2,500 meters and "deep." Figure D-6 illustrates the first case. Range 1 corresponds to the shortest range. In the example, the Blue weapon type is permitted to fire at all six ranges. The Red weapon type is permitted to fire at the first three ranges. In this case, the Red weapon was denied the opportunity to fire at greater ranges because the AFP analyst had looked at the SSPK data and noted that Red type 24 had 0.0 PK against Blue type 24 at 1,500 meters. Limiting Red firing to the first three ranges saves some computing time.

(7) **REFIRE TIME** specifies the mean times (in minutes) to refire. Refire time is assumed to be lognormally distributed with the given means and standard deviations equal to one-half the given means. Refire times do not affect first shots of the usual direct fire weapon types. However, "refire" times do affect the times (shot cycles) to the first shots of weapons in the counterbattery versions of direct fire engagements.

(8) The section **DETECTION** provides input values affecting detection: whether there is detection at all and the times to detect.

(a) **SENSOR TYPE** provides two entries per weapon type. The first entry is the index of the sensor type. The second entry specifies whether the sensing is in the optico-visual (1) or thermal (2) bands.

(b) **RESOLVABLE CYCLES** gives the minimum number of resolvable cycles required for a detection to be successful in the sense that a target is considered sufficiently well identified to be fired upon.

(c) **SENSING SIZE** gives the apparent sizes in meters. The sizes depend on weapon type and posture. In the example, all the Blue weapons are in the open, and all the Red weapons are in defilade.

(d) **LIGHT** specifies the light level in foot candles. Values greater than 100.0 are reset to 100.0 within the Combat Module.

(e) **CONTRAST** provides two entries per weapon type. The first entry is the optico-visual contrast. The second entry is the temperature difference for thermal sensing.

(f) **ATTENUATION** gives the extinction coefficients.

(g) **MAGNIFICATION** gives the magnification (in diameters) for the corresponding sensors.

(h) **BRIGHTNESS** specifies the sky over ground brightness ratio.

(9) **DISTRIBUTION OF INDIRECT WEAPONS** applies if one or both weapons are mortar, artillery, or rockets used as counterbattery weapons--i.e., in the special version of direct fire engagement. Then the entry specifies the fraction of the weapon type devoted to counterbattery actions. (There should be no engagements in which only one weapon type is in the counterbattery role.)

(10) **EXPECTED FRACTIONAL DAMAGE** applies if one or both weapons are mortar, artillery, or rockets used as counterbattery weapons. (There should be no engagements in which only one weapon type is in the counterbattery role.) Then each entry specifies the fractional damage per round for the one given weapon type opposing the other given weapon type. However, counterbattery kills in the Combat Module do not depend on these numbers but on the following Blue and Red PK entries.

(11) **BLUE PK** specifies the single shot PK for the given Blue weapon type against the given opposing Red weapon type at each of the six identified ranges. The PKs take into account the posture of shooter and target: stationary or moving, open or defilade. The same PKs are used for first and all subsequent shots.

(12) **RED PK** specifies the single shot PK for the given Red weapon type against the opposing Blue weapon type at each of the six identified ranges. Other remarks on Blue PK apply here as well.

(13) **RANGE DISTRIBUTION** specifies the intended fractional distribution of duels in each (if there are two) odds class. The entries shown are the AFP version of a "uniform" distribution. In the event of "tied" fractions, duels are inserted from shorter to longer ranges. Duels are distributed independently from each odds class. Note that fewer than six duels within an odds class inevitably leaves at least one range empty.

(14) **ORIGINAL PREFERENCE** specifies fractional preferences from the shooters' points of view without regard to targets' preferences or to possibly very different numbers of weapons of each and other types.

(15) **MODIFIED PREFERENCE** specifies fractional preferences after reconciliation of possibly different shooter and target views. The entries have been moved half-way toward the mean of the original preferences and then renormalized across weapon types. Recall that these modified preferences generally do not directly show the fractions of weapons that are to be allocated to this type-on-type engagement because the net fractions depend also on the total inventories of these and other weapon types.

(16) PARTICIPATION (first occurrence) specifies the fractions of allocated weapons to be engaged. Participation is intended to vary from 0.0 to 1.0, but more than 100 percent participation is accepted by the Combat Module. Participation is intended to provide a factor for adjusting engagement levels in accord with any of many practical realities. For example, aircraft and air defense weapons may have very high preferences for one another; but, if the speeds at which aircraft move are so great as to make aircraft and air defense engagements relatively infrequent and/or of very short duration, a participation factor less than 1.0 yields a reduced activity level. Or reduced participation may be considered necessary as a way to represent the fact that many intended engagements do not occur.

(17) ENGAGEMENT DURATION is specified in minutes. The example sets duration to 2.01 minutes.

(18) ENGAGEMENTS PER DAY should be "1". Fractional engagements have no significance.

(19) # TIMES SURVIVOR IN CONFLICT specifies the number of conflicts per day. Fewer than the specified number of conflicts may occur if there are no survivors of early conflicts.

(20) CREW LOSSES SUFFERED PER KILL specifies the number of the casualties assessed for each weapon killed of the corresponding type.

(21) BLUE INDIRECT REFIRE TIMES lists the input-specified times between rounds for the Blue indirect fire weapon types by weapon type. The times are given in minutes. The time 6.66 minutes is a default value input for nonexistent weapon types. The values 0.5 and 1.0 correspond to single round sustained refire times of ordinary tube weapons. The 0.16 minute time is artificial in the sense that it leads to the expenditure of a full 12-round load of rockets from a launcher within 2 minutes without an implied pod reload. The artifice is liable to some abuse by the integer roundup and roundoff rules used in latching indirect fire volleys to the direct fire shot cycles.

(22) BLUE FRACTION FOR AREA FIRE lists the input-specified fractions of total indirect fire weapons by type to be devoted to indirect fire roles. It is intended that counterbattery and indirect fractions sum to 1.0. However, it is possible to override this limit permitting more or less than the total number of weapons to be allocated. In the example, the fractions range from 0.5 to 1.0.

(23) BLUE INDIRECT KILLS/RND specify the "lethalities" of indirect fire rounds in order by weapon type for each direct fire range band. The values in the example are arbitrary values input for test purposes. Recall that these data are in the form of fractional kills per round not SSPKs, whose values must not exceed 1.0.

(24) **RED INDIRECT REFIRE TIMES, RED FRACTION FOR AREA FIRE, and RED INDIRECT KILLS/RND** for Red all are similar to the Blue counterparts described in (21)-(23) above.

(25) **DAY 1**

(a) **ALLOCATIONS** shows the numbers of the weapons of the given types allocated to this type-on-type engagement on the first day.

(b) **PARTICIPATION** (second occurrence) shows the numbers of weapons of the given types assigned to the type-on-type engagement after application of the participation factors given in line PARTICIPATION (first occurrence) above. In the example shown, because the participation factor is 1.0 for both weapon types, the numbers of weapons participating are exactly equal to the numbers of weapons allocated.

(c) **EXTERNAL LOSSES** shows how many losses (if any) were assessed as preengagement losses in accord with input-specified factors. In the example, Blue and Red external loss factors were both 0.0; hence, 0.0 weapons were charged as external losses. External losses are assessed each day at the input-specified fraction of allocations.

(d) **BLUE DIRECT LOSSES** lists the numbers of Blue weapons of the given type killed by the opposing Red direct fire weapon type at each of the six ranges. (The numbers of weapons assigned to each range are listed later in the QAREP.) The losses are the totals over all conflicts on the first day.

(e) **RED DIRECT LOSSES** lists the numbers of Red weapons or the given type killed by the opposing Blue direct fire weapon type at each of the six ranges. (The numbers of weapons assigned to each range are listed later in the QAREP.) The losses are the totals over all conflicts on the first day.

(f) **BLUE DIRECT SHOTS** records the number of shots (or bursts) fired by Blue direct fire weapons of the given type at each range. The entries are the totals over all conflicts between these two types on the first day.

(g) **RED DIRECT SHOTS** records the number of shots (or bursts) fired by Red direct fire weapons of the given type at each range. The entries are the totals over all conflicts between these two types on the first day.

(h) **BLUE INDIRECT LOSSES** tabulates the losses suffered by the Blue direct fire weapon type to each of 10 (columns) Red indirect fire weapon types (usually fewer than 10 types are active) by range (rows). Ranges with no associated losses are suppressed. Totals by range and by indirect fire weapon type are included. The results are totals over all conflicts on the first day. Clearly, a loss recorded in a column implies that the corresponding Red indirect fire weapon type must be assigned to the engagement. However, the absence of a loss in a column implies nothing; the corresponding indirect fire weapon type may or may not have been assigned. Fortunately, other tables in QAREP identify exactly how many weapons and rounds are involved in the losses shown.

(i) RED INDIRECT LOSSES tabulates Red indirect fire weapon losses in a fashion just like that described for BLUE INDIRECT LOSSES in (h) above.

(j) BLUE INDIRECT SHOTS tabulates the rounds fired by Blue indirect fire weapons against the corresponding Red direct fire weapon type. The rounds are recorded for each of 10 (columns) Blue weapon types and by range (rows). Total rounds by range and firing weapon type are included. The results are totals over all conflicts on the first day. The rounds fired are usually fractions of the totals that could be fired by the given numbers of tubes at the given rates and in the times available. The smaller numbers of rounds fired reflect adjustment for the fact that usually only a fraction of the targets of the given type are committed to the engagement displayed.

(k) RED EXPECTED INDIRECT LOSSES tabulates the products of Blue indirect rounds fired (from the table above) and the expected "fractional kills per round" (separately reported below). The expected losses are totals over all conflicts on the first day. Indirect fire rounds may "strike" already killed targets. No credit is given for overkill. In general, expected indirect kills may exceed the number of available targets.

(l) RED INDIRECT SHOTS and BLUE EXPECTED INDIRECT LOSSES present counterpart information to that just described for BLUE INDIRECT SHOTS and RED EXPECTED INDIRECT LOSSES above. Again, the tables cover all conflicts on the first day.

(m) CONFLICT 1 begins a section of tables providing more detailed information about the activities of the first conflict on the first day.

1. The first table under CONFLICT 1 provides information about the direct fire duels by range (rows). The two columns headed DUELS show how many duels at the corresponding odds classes (1:1 and 2:1 ratios) were allocated to the first conflict on the first day. At 250 m it should be clear that 22 duels at 1:1 and one duel at 2:1 translates to a total of 24 Blue and 23 Red direct fire weapons. From the duels/odds data, it is always possible to determine how many weapons are involved. The columns headed DETECTIONS show ATTEMPTed detections, SUCCESSful detections, and AVeraGe TiMes to detect. The number of successes should not exceed the number of weapons. However, because not all attempts are successful, the number of attempts may exceed the number of weapons. The Combat Module notes when a detection failure is the result of too few resolvable cycles; additional attempts are not made inasmuch as the same (insufficient) number of resolvable cycles would be recur. The last column in the table (UP BOUND ON SHOTS) list the number of shot cycles at each range. In the example, 24 shot cycles are permitted at each range. Recall that the number of shot cycles depends on both direct refire times and the odds classes. Because the Combat Module includes a "shoot back" feature, weapons may, outside the normal detection process, return fire after receiving an input-specified number of rounds from otherwise "undetected" opponents.

2. **BLUE INDIRECT SHOOTERS** lists the numbers of Blue indirect fire weapons by type and in total (columns) assigned to each range and in total (rows) for the first conflict on the first day. All indirect firers are considered assigned, even though less than all targets of the given type may be committed to the currently displayed engagement. (Counterbattery firers of the same weapon type are not considered "indirect firers" in the special AFP sense.) Adjustment for target numbers is made to the rounds fired--not to the tubes considered to be firing.

3. **RED INDIRECT SHOOTERS** lists counterpart information for Red similar to that for Blue described in the immediately preceding paragraph.

(n) CONFLICT 2 begins a set of three tables for the second conflict on the first day. The tables are similar to those for CONFLICT 1 and subsequent conflicts 3 and 4. However, the values in the tables may differ markedly from those of the preceding and later conflicts. Because of attrition in conflict 1, conflict 2 generally begins with fewer direct fire weapons. Different numbers of direct fire weapons will usually force a change in the odds classes of duels. Differences in odds classes in turn may change the number of shot cycles relative to the first conflict. Furthermore, even though all indirect firers survive the entire day intact, the indirect fire weapons are assigned to the conflicts only in proportion to r/R, where r is the number of targets at the beginning of a conflict and R is the number at the beginning of the day. Throughout a day the R-value remains fixed. However, the r-value is diminished by the kills in preceding conflicts. In the example, r/R is smaller for the second conflict than for the first one on the first day.

(o) CONFLICT 3 includes tables similar to those described above for conflicts 1 and 2. However, because no direct fire weapons survived conflicts 1 and 2 at ranges 250 through 1,000 meters, those ranges are "vacant" in conflict 3.

(p) CONFLICT 4 includes tables similar to those described above. As in conflict 3, the shorter range bands are vacant as the result of total attrition at those ranges in the first two conflicts.

(q) RED:BLUE EXCHANGE RATIO presents the net exchange ratios for the first day's conflicts.

(26) DAY 2 provides information about the second day's conflicts between the same two direct fire weapon types. The formats are the same as for DAY 1. The Combat Module, between the first and second days, performs the important step of pooling survivors of direct fire engagements of all types and repeating the allocation of direct and indirect fire weapons to engagements using the new availabilities. Strongly divergent attrition patterns on the first day may lead to vastly different weapons assignments on the second day, even though the preference factors remain the same on all days. The reason for the differences in allocations is that net allocations depend not only on the preferences but also on the inventories; it is the inventories that may have changed greatly from one day to the next.

The indirect firers are not attrited during a day, but the counterbattery weapons generally are attrited. Between days, the indirect and counterbattery weapons of a type are pooled before applying the original counterbattery and area fire fractions again. Hence, the numbers of indirect fire weapons allocated to indirect and counterbattery roles may both change.

D-4. RUNSTREAM. This paragraph describes the SSG program which generates runstreams to execute the preprocessors PREFGEN, RNGDSTGEN, PKSGEN, and PROJGEN, the Combat Module MAIN, and the postprocessors QA Report, Killer/Victim Scoreboard, Allocation Report, and CXPS Module. A runstream will be generated for each environment selected. Multiple executions of the preprocessor, MAIN module, and postprocessors will be in the runstream if different random number seeds are requested. Familiarity with the UNIVAC Symstream language and SSG processor is assumed.

a. INTENT. SSG generation of runstreams is intended to simplify changes to element names and parameters which are needed when creating runstreams. Execution of the AFP Combat Module requires 11 different input files, 5 output files which are saved, and several parameters per program. During production runs, there are 16 runstreams, each with 10 repetitions. Chances of missing an element name or parameter are great. A generator minimizes naming errors and would eliminate keying errors.

b. SSG Section. The SSG statements are the parameters for the SSG skeleton. Their order is not important, because the first variable on the line is a keyword. Figure D-7 is an example of the SSG statements required. Explanations of each SGS follow.

(1) RUNID - user ID on @RUN statement.

(2) USERID - user ID for permanent files which will be created during run.

(3) PFILES - permanent files where absolute programs are stored.

(4) FORCE - four-character force code for permanent file names and report titles.

(5) FORCE1CH - one-character force code for permanent file names and version names.

(6) ENV - all possible environments; this SGS does not change.

(7) ENVFIRST - first environment for which a runstream should be generated.

(8) ENVLAST - last environment for which a runstream should be generated; runstreams will be generated for the range of first through last environments.

```
SGS
RUNID 30
USERID H7
PFILES *H7AFP *98AFP
FORCE HM80
FORCE1CH H
ENV 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
ENVFIRST 1
ENVLAST 3
SUPP 01 06 11 04 01 06 11 04 01 06 11 04 01 06 11 04
PNAMES RAPD STATIC RADE BAPD RAPD STATIC RADE BAPD ;
       RAPD STATIC RADE BAPD RAPD STATIC RADE BAPD
LIGHT D D D D N N N N D D D D N N N N
BDIV 1 1 1 3 1 1 1 3 1 1 1 3 1 1 1 3
RDIV 3 1 4 1 3 1 4 1 3 1 4 1 3 1 4 1
ALPHA A B C D E F G H I J K L M N O P
NREPS 1
KVCOPIES 3
BASE *H7BASEDATA
PKS H7PKS
RNGDST *H7RNGDST
PREF *H7PREF
CSCSS *H7CSCSS
ENGAGE *H7ENGAGE RAPD
PCAS *H7PCAS H
CVALS *H7GLOBAL CVALS
FRACTS *H7GLOBAL FRACTS
GDNSTY1 T T T F T T T F T T T F T T T F
GDNSTY2 F F F T F F F T F F F T F F F T
TYPLIN 1 60 1 60
SUPERTRP ''10 8 5 ''
SUPERB ''1 2 3 4 5 6 42 44''
SUPERR ''1 2 3 42 44''
CAREP
CAPKS 0
RPTPS1 '' 1 2 3 4 5 6 7 8 9 10 ''
RPTPS1 '' 11 12 13 14 15 16 17 18 19 20 ''
RPTPS1 '' 21 22 23 24 25 26 27 28 29 30 ''
RPTPS1 '' 31 32 33 34 35 36 37 38 39 40 ''
RPTPS1 '' 41 42 43 44 45 46 47 48 49 50 ''
RPTPS1 '' 51 52 53 54 55 56 57 58 59 60 ''
RPTPS2 '' 1 2 3 4 5 6 7 8 9 10 ''
RPTPS2 '' 11 12 13 14 15 16 17 18 19 20 ''
RPTPS2 '' 21 22 23 24 25 26 27 28 29 30 ''
RPTPS2 '' 31 32 33 34 35 36 37 38 39 40 ''
RPTPS2 '' 41 42 43 44 45 46 47 48 49 50 ''
RPTPS2 '' 51 52 53 54 55 56 57 58 59 60 ''
AIR1  '' 1   41 ''
AIR2  '' 1   41   ''
OJTPD '' 80''
OKTHTP ''  F''
OJVIS 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
OJDAY 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2
OJPOS 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
IBFOR ''  2001''
IPFOR ''  1000''
ICOMBO ''  1''
TVALON TRUE
@EOF
```

Figure D-7. SGS Statements Example

(9) SUPP - environment-related suffix for CSCSS elements; four CSCSS elements are created to correspond to the four postures; all RAPD environments will use 01; STATIC environments use 06; RADE environments use 11; BAPD environments use 04.

(10) PNAMES - postures for all 16 environments; this SGS does not change.

(11) LIGHT - day/night conditions for each of 16 environments.

(12) BDIV - Blue force ratio for all 16 environments; Blue inventory is for one division, if the force to be played is larger, the force ratio is a multiplier.

(13) RDIV - Red force ratio for all 16 environments.

(14) ALPHA - 5th character of run ID.

(15) NREPS - random number seeds for replications; each runstream created will use all seeds listed here.

(16) BASE - basedata file; element names using FORCE and ENV SGS statements will be generated.

(17) PKS - SSPK file; element names using PNAMES, LIGHT, and FORCE1CH SGS statements will be generated.

(18) RNGDST - range distribution file; element names using PNAMES and FORCE1CH SGS statements will be generated.

(19) PREF - preference file; element names use PNAMES, FORCE1CH.

(20) CSCSS - CSCSS file; element names use FORCE, SUPP.

(21) ENGAGE - engagement file and element name; element name does not change by environment or posture.

(22) PCAS - PCAS file and element name; element name does not change by environment or posture.

(23) CVALS - file and element name of CVALS input to CXPS Module.

(24) FRACTS - file and element name of FRACTS input to CXPS Module.

(25) GDNSTY1 - density reductions Side 1 for all 16 environments; T = reduction occurs, F = no reduction.

(26) GSNSTY2 - density reduction Side 2 for all 16 environments.

(27) TYPLIM - types entering combat.

(a) Lower and upper limit Side 1 types.

(b) Lower and upper limit Side 2 types.

(28) SUPERTRP - super-troop parameter for allocation reports; enclose in quotes

(a) Super-troop multiplication factor

(b) Total number of Blue super-troops

(c) Total number of Red super-troops.

(29) SUPERB - AFP number of Blue super-troop items; enclose in quotes.

(30) SUPERR - AFP number of Red super-troop items; enclose in quotes.

(31) QAREP - this SGS is for the QA report; if removed or misspelled no report will be generated.

(32) QAPKS - SSPK print parameters for QA report; 0 = suppress SSPK values in report; 1 = print SSPK values which makes QA report SECRET classification.

(33) RPTPS1 - AFP number(s) of Blue type(s) to be shown in QA report; this SGS may be repeated to accommodate all AFP numbers needed; enclose in quotes.

(34) RPTPS2 - AFP number(s) of Red type(s) to be shown in QA report; this SGS may be repeated; enclose in quotes.

(35) AIR1 - Side 1 aircraft types; enclose in quotes.

(36) AIR 2 - Side 2 aircraft types; enclose in quotes.

(37) OJTPD - A year symbol to appear in output records.

(38) OKTHTR - A theater symbol to appear in output records.

(39) OJVIS - Symbols corresponding to clear (1) and degraded (2) visibility for the AFP standard 16 combat environments. A symbol appears in output records and may appear in the input CS/CSS moduli records.

(40) OJDAY - Symbols corresponding to daytime (1) and nighttime (2) for the AFP standard 16 combat environments. A symbol appears in output records and may appear in the input CS/CSS moduli records.

(41) OJPOS - Symbols corresponding to RAPD (1), STATIC (2), RADE (3), and BAPD (4) postures for the AFP standard 16 combat environments. A symbol appears in output records and may appear in input CS/CSS moduli records.

(42) IBFOR - Specifies a four- to six-digit division identifier to be included in output records. A TPSN is a logical choice for division identifier.

(43) IRFOR - Specifies a four- to six-digit threat identifier to be included in output records.

(44) ICOMBO - A case symbol appearing in CS/CSS moduli input and partial combat potential output records.

(45) TVALON - Controls the weighting of the first four components of combat potential. TRUE = apply target value weights. FALSE = apply weights of 1.0.

c. SKEL Section. The SSG Skeleton is a program which will generate ECL statements using the SGS parameters and internal looping directives. The skeleton shown in Figure D-8 will generate a standard AFP runstream which can be modified for special cases or conditions. Or, the skeleton itself can be modified to generate a special case. Points of interest about the skeleton include the following:

(1) The skeleton creates additional SGS statements which are used during runstream generation. These new SGS statements are element names which change by environment name and/or posture.

(a) COMBAT - environment for which this runstream is being generated; skeleton will loop from ENVFIRST through ENVLAST.

(b) PREFIX - USERID and FORCE are prefixed to permanent file names.

(c) CXPS - environment part of element name to which CXPS output is saved; name also uses NREP.

(d) CSELT - FORCE and SUPP used to create CSCSS element name.

(e) BASELT - FORCE and ENV used to create BASEDATA element name.

(f) PKSELT - PNAMES and LIGHT used to create PKS element name with version FORCE1CH.

(g) RNGELT - PNAMES used to create range distribution element name with version FORCE1CH; name has format PNAME - DEEP/FORCE1CH.

(h) PRFELT - PNAMES used to create preference element name with version FORCE1CH.

(i) INVNTR - file name where inventory elements are kept; FORCE used to create inventory element name with version PNAMES: if inventory does not change by posture, remove version from generated runstream or change skeleton.

```
SKEL
*INCREMENT IE FROM [ENVFIRST,1,1,1] TO [ENVLAST,1,1,1]
*CREATE SGS: COMBAT [ENV,1,IE,1]
*CREATE SGS: PREFIX [USERID,1,1,1][FORCE,1,1,1]
*CREATE SGS: CXPS [ENV,1,IE,1]
*CREATE SGS: CSELT [FORCE,1,1,1]E[SUPP,1,IE,1]
*CREATE SGS: BASELT [FORCE,1,1,1]E[ENV,1,IE,1]
*CREATE SGS: PKSELT [PNAMES,1,IE,1]-[LIGHT,1,IE,1]/[FORCE1CH,1,1,1]
*CREATE SGS: RNGELT [PNAMES,1,IE,1]-DEEP/[FORCE1CH,1,1,1]
*CREATE SGS: PRFELT [PNAMES,1,IE,1]/[FORCE1CH,1,1,1]
*CREATE SGS: INVNTR H7AFPFILES INV[FORCE,1,1,1]
*CREATE SGS: NDIV1 [BDIV,1,IE,1]
*CREATE SGS: NDIV2 [RDIV,1,IE,1]
*CREATE SGS: DNSTY1 [GDNSTY1,1,IE,1]
*CREATE SGS: DNSTY2 [GDNSTY2,1,IE,1]
*BRKPT,K H7RUNS.GO[FORCE,1,1,1]E[COMBAT,1,1,1]N[NREPS,1,1,1]
*EDIT ON
#RUN,/TPR DO[RUNID,1,1,1][ALPHA,1,IE,1][NREPS,1,1,1],E1899P2277D,&
SECRET,600,6300    . KNOX (703) 476-4923
#QUAL UNCLASSIFIED
#ASG,A H7AFPFILES/AFP/AFP.
#ASG,A H7RUNS/AFP/AFP.
*INCREMENT L TO [PFILES]
*INCREMENT M TO [PFILES,1]
#ASG,A [PFILES,L,M,1].
*LOOP
*LOOP
#HDG,S **** [FORCE,1,1,1] ENV [COMBAT,1,1,1] SEEDS [NREPS,1] ****
#PRT,S H7RUNS.GO[FORCE,1,1,1]E[COMBAT,1,1,1]N[NREPS,1,1,1]
# .  SAVE THE PROGRAMS
#ASG,T ABS-FILE.,///10000
#COPY,AC *H7AFP.870PKSGEN,ABS-FILE.870PKSGEN
#COPY,AC *H7AFP.870PROJGEN,ABS-FILE.870PROJGEN
#COPY,AC *H7AFP.870PREFGEN,ABS-FILE.870PREFGEN
#COPY,AC *H7AFP.870RNGDSTGEN,ABS-FILE.870RNGDSTGEN
#COPY,AC *H7AFP.870MAIN,ABS-FILE.870MAIN
#COPY,AC *98AFP.870GENKV,ABS-FILE.870GENKV
#COPY,AC *98AFP.870GENAL,ABS-FILE.870GENAL
#COPY,AC *98AFP.870PRTKV,ABS-FILE.870PRTKV
#COPY,AC *H7AFP.870CXPS,ABS-FILE.870CXPS
#COPY,AC *H7AFP.870REPORT,ABS-FILE.870REPORT
#COPY,AC *H7AFP.PAUSE,ABS-FILE.PAUSE
*INCREMENT L TO [PFILES]
*INCREMENT M TO [PFILES,1]
#FREE [PFILES,L,M,1].
*LOOP
*LOOP
*IF [COMBAT]
#ASG,T 20.,///4000
#ASG,T 21.,///4000
#ASG,T 22.,///4000
#ASG,T 23.,///4000
#ASG,T 24.,///5000
#ASG,T 25.,///4000
#ASG,T 26.,///4000
#ASG,T  3.,///2000
#ASG,T  4.,///2000
#ASG,T  7.,///2000
# .   UNIT 35 IS USED FOR TEMPORARY STORAGE
#ASG,T 35.,///2000
#ASG,T 34.,///25000
#ASG,T  9.,///500
#ASG,T  31.,///7000
#ASG,T  32.,///13000
#ASG,T  33.,///7000
*SET E TO 1
# .          I N P U T    F I L E S
# .
# .  LOCATE THE ELEMENTS FOR ENVIRONMENT DEPENDENT FILES
# .
#ASG,A [BASE,1,1,1].       . BASEDATA FILE
#ASG,T 15.
#ED [BASE,1,1,1].[BASELT,1,1,1],15.
#PRT,S [BASE,1,1,1].[BASELT,1,1,1]
#FREE [BASE,1,1,1].
#ASG,A [PKS,1,1,1]/XXX/XXX.      . PKS
```

Figure D-8. SSG Skeleton
(page 1 of 4 pages)

```
#ASG,T 10.
#ED [PKS,1,1,1].[PKSELT,1,1,1],10.
#FREE [PKS,1,1,1].
#ASG,A [RNGDST,1,1,1].          . RANGE DISTRIBUTION
#ASG,T 11.
#ED [RNGDST,1,1,1].[RNGELT,1,1,1],11.
#FREE [RNGDST,1,1,1].
#ASG,A [PREF,1,1,1].            . PREFERENCE FILE
#ASG,T 12.
#ED [PREF,1,1,1].[PREFELT,1,1,1],12.
#FREE [PREF,1,1,1].
#ASG,A [CSCSS,1,1,1].
#ASG,T 14.
#ED [CSCSS,1,1,1].[CSELT,1,1,1],14.
#FREE [CSCSS,1,1,1]
#ASG,A [ENGAGE,1,1,1].          . ENGAGEMENT FILE
#ASG,T 13.
#ED [ENGAGE,1,1,1].[ENGAGE,1,2,1],13.
#FREE [ENGAGE,1,1,1].
#ASG,A [PCAS,1,1,1].            . PERSONNEL CASUALTY FILE
#ASG,T 16.
#ED [PCAS,1,1,1].[PCAS,1,2,1],16.
#FREE [PCAS,1,1,1].
#ASG,A [USERID,1,1,1]KV[FORCE,1,1,1]/XXX/XXX.     . KV FILE
#ERS 20.
#ERS 21.
#ERS 25.
#ERS 26.
#XQT ABS-FILE.870PREFGEN
 1 1 1 1 1 1
#XQT ABS-FILE.870RNGDSTGEN
 1 1 1 1 1 1
#XQT ABS-FILE.870PKSGEN
 1 1 1 1 1 1
#XQT ABS-FILE.870PROJGEN
 1 1 1 1 1 1
*INCREMENT IR TO [NREPS,1]
*CREATE SGS: REPS [NREPS,1,IR,1]
*SET R TO [REPS,1,1,1]
#ERS 22.
#ERS 23.
#ERS 24.
#ERS  3.
#ERS  4.
#ERS  9.
#ERS 31.
#ERS 32.
#ERS 34.
#XQT ABS-FILE.870MAIN
 1 1 1 1 1 1
[*R]
[CNSTY1,1,E,1]  [CNSTY2,1,E,1]
[TYPLIM,1,1,1] [TYPLIM,1,2,1]   [TYPLIM,1,3,1] [TYPLIM,1,4,1]
#ASG,T SORT33.,///1000
#SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=33.
FILEOUT=SORT33.
#EOF
#USE 33.,SORT33.
#ASG,T SORT9.,///1000
#SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=9.
FILEOUT=SORT9.
#EOF
#USE 9.,SORT9.
#ASG,T SORT24.,///1000
#SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=24.
FILEOUT=SORT24.
#EOF
#USE 24.,SORT24.
```

Figure D-8. SSG Skeleton
(page 2 of 4 pages)

```
#ASG,T SORT32.,///1000
#SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=32.
FILEOUT=SORT32.
#EOF
#USE 32.,SORT32.
# .  SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
#ASG,A [PREFIX,1,1,1]/XXX/XXX.
# . COPY,I 3.,[PREFIX,1,1,1].A[COMBAT,1,E,1]R[*R]
# . COPY,I 4.,[PREFIX,1,1,1].F[COMBAT,1,E,1]R[*R]
# . COPY,I 9.,[PREFIX,1,1,1].T[COMBAT,1,E,1]R[*R]
*IF [QAREP]
#DELETE,C [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.
#XQT ABS-FILE.PAUSE
#CAT,P [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.,F///20000
#ASG,A [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.
#KEEP [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R].
#BRKPT PRINTS/[PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]
#XQT ABS-FILE.870REPORT
[CAPKS,1,1,1]
 1 1 1 1 1 1
*INCREMENT K TO [RPTPS1]
*INCREMENT L TO [RPTPS1,K]
[RPTPS1,K,L,1]
*LOOP
*LOOP
9999
*INCREMENT K TO [RPTPS2]
*INCREMENT L TO [RPTPS2,K]
[RPTPS2,K,L,1]
*LOOP
*LOOP
9999
#EOF
#BRKPT PRINTS
# . SYM,U [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R].
*END
#DELETE,C [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.
#XQT ABS-FILE.PAUSE
#CAT,P [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.,F///20000
#ASG,A [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.
#KEEP [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R].
#BRKPT PRINTS/[PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]
#XQT ABS-FILE.870GENKV
#XQT ABS-FILE.870PRTKV
KV SCOREBOARD FOR [FORCE,1,1,1]  ENV [COMBAT,1,E,1]   SEED [*R]
[SUPERTRP,1,1,1]
[SUPERB,1,1,1]
[SUPERR,1,1,1]
#ADD,E [INVNTR,1,1,1].[INVNTR,1,2,1]
#ADD,E 7.
#ED 7.,[USERID,1,1,1]KV[FORCE,1,1,1].E[COMBAT,1,E,1]R[*R]
# . COPY KV FILE FOR KV ROLLUP
#ASG,T 7.      . FOLLOWING PROGRAM WRITES TO UNIT 7
#XQT ABS-FILE.870GENAL
# . PRINT THE ALLOCATION REPORT
#XQT ABS-FILE.870PRTKV
ALLOCATION REPORT FOR [FORCE,1,1,1] ENV [COMBAT,1,E,1] SEED [*R]
[SUPERTRP,1,1,1]
[SUPERB,1,1,1]
[SUPERR,1,1,1]
#ADD,E [INVNTR,1,1,1].[INVNTR,1,2,1]
#ADD,E 7.
*IF [CXPS]
*IF NOT [COMBAT]
#ASG,T 7.,///2000
#ASG,A [BASE,1,1,1].       . BASEDATA FILE
#ASG,T 15.
#ED [BASE,1,1,1].[BASELT,1,1,1],15.
#FREE [BASE,1,1,1].
#ASG,A [CSCSS,1,1,1].
#ASG,T 14.
#ED [CSCSS,1,1,1].[CSELT,1,1,1],14.
#FREE [CSCSS,1,1,1]
#ASG,A [PCAS,1,1,1].           . PERSONNEL CASUALTY FILE
```

Figure D-8. SSG Skeleton
(page 3 of 4 pages)

```
#ASG,T 16.
#ED [PCAS,1,1,1].[PCAS,1,2,1],16.
#FREE [PCAS,1,1,1].
# .  END OF NOT COMPAT IF
*END
#HDG    ** [FORCE,1,1,1] CXPS ENV [CXPS,1,E,1]  SEED [*R] **
# .  CREATE OUTPUT FILES FOR CXPS
#ASG,T 29.
#ASG,T 30.
# .  RUN THE MERGE MODULE
# .  RUN WITH FILES DEVELOPED EARLIER
# . COPY,I [PREFIX,1,1,1].A[CXPS,1,E,1]R[*R],3.
# . COPY,I [PREFIX,1,1,1].F[CXPS,1,E,1]R[*R],4.
# . COPY,I [PREFIX,1,1,1].T[CXPS,1,E,1]R[*R],9.
#ERS 7.
#XQT ABS-FILE.870CXPS
*. IMAGE GIVING INDICES IN CBT POT OUTPUT RECORDS
*EDIT ON
[ICOMBO,1,1,1]&
[OJTPD,1,1,1]&
[OKTHTR,1,1,1]&
  [OJVIS,1,IE,1]&
  [OJDAY,1,IE,1]&
  [OJPOS,1,IE,1]&
[IBFOR,1,1,1]&
[IRFOR,1,1,1]
*. IMAGE GIVING INDICES OF INPUT CS/CSS MODULI
[TVALON,1,1,1]
1  1  1  1  1  1
[NDIV1,1,E,1]  [NDIV2,1,E,1]
[AIR1,1,1,1]
[AIR2,1,1,1]
#ADD [CVALS,1,1,1].[CVALS,1,2,1]
#ADD [FRACTS,1,1,1].[FRACTS,1,2,1]
# .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
#DATA,L 29.
#END
#COPY,I 29.,[PREFIX,1,1,1].E[CXPS,1,E,1]R[*R]
#ASG,T SORTOUT.
#SORT,S
VOLUME=SMALL
KEY=1,5,CH,A:67,10,CH,A
FILEIN=29.
FILEOUT=SORTOUT.
#ED,R SORTOUT.
LNP!
EXI
*END
#BRKPT PRINT$
#SYM,U [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R].,[KVCOPIES,1,1,1]
*REMOVE SGS REPS
*LOOP      .IR
*.  END OF CXPS IF
*.  END OF ENVIRONMENT IF
*END
#FIN
*REMOVE SGS COMBAT
*REMOVE SGS PREFIX
*REMOVE SGS CXPS
*REMOVE SGS CSELT
*REMOVE SGS BASELT
*REMOVE SGS PKSELT
*REMOVE SGS RNGELT
*REMOVE SGS PRFELT
*REMOVE SGS INVNTR
*REMOVE SGS NDIV1
*REMOVE SGS NDIV2
*REMOVE SGS DNSTY1
*REMOVE SGS DNSTY2
*LOOP      .IE
@EOF
```

Figure D-8. SSG Skeleton
(page 4 of 4 pages)

(j) **NDIV1** - Blue force ratio of environment for which this runstream is being generated; taken from SGS BDIV.

(k) **NDIV2** - Red force ratio of environment for which this runstream is being generated; taken from SGS RDIV.

(l) **DBSTT1** - Blue density reduction of environment for which this runstream is being generated; taken from SGS GDNSTY1.

(m) **DNSTY2** - Red density reduction of environment for which this runstream is being generated; taken from SGS GDNSTY2.

(2) The generated runstreams will be saved as elements in H7RUNS. The element names have the format:

GO (FORCE) E (COMBAT) N (NREPS first subfield).

For example, the SGSs FORCE HM80.

```
ENVFIRST 01
ENVLAST  02
NREPS    3  4
```

will create additional SGSs COMBAT 01 and COMBAT 02, and two runstream elements will be generated,

H7RUNS.GOHM80E01N3 and
H7RUNS.GOHM80E02N3.

(3) All program absolutes are copied to a temporary file and executed from the temporary copy during the run.

(4) If the QA report is to be executed, it is done within a breakpoint file which can be printed later with an @SYM. This is because QA reports are very large. The breakpoint file can be scanned without wasting printer time and paper. If hard copy is needed, some or all of the report can be printed.

(5) The killer/victim scoreboard and allocation reports are written to a different breakpoint file. Analysts often require multiple copies of these reports. Writing the reports to a breakpoint file allows printing several copies without reexecution of the programs.

(6) The K/V results are saved as elements in a permanent file. The format of the file name and element name is: (USERID) KV (FORCE). E (COMBAT) R (repetition from NREP)

(7) The ALLOC, FALLOC, and TTLOS files from the MAIN module can be saved, but because they are large, they generally are not. The statements on lines 164-166 in H7AFP.MAINSSG/SKELREP should be changed from @COPY to @COPY if those files should be saved.

(8) The CXPS output files are saved as elements in:

(PREFIX) . E (CXPS) R (repetition from NREP)

d. Runstream to Execute SSG. Figure D-9 is an example of the statements needed to execute the SSG skeleton, H7AFP.MAINSSG/SKELREP, using the SGS parameters, H7AFP.MAINSSG/SGS. The statements can be executed within a breakpoint because of the length of the output. Check for an "END SSG" statement in the breakpoint file to make sure there were no errors. If there are no errors, the breakpoint file can be deleted. Make sure that the file H7RUNS. is available. One of the most possible errors that could occur is, if the generated runstream cannot be copied to H7RUNS.

```
@ASG,AX H7RUNS/YYY/XXX.
@BK1 U
@SSG,ABIKE H7AFP.MAINSSG/SKELREP,H7AFP.MAINSSG/SGS
@BK2,E
L END SSG
```

Figure D-9. Runstream to Execute SSG

e. Generated Runstreams. The generated runstreams are saved as elements in H7RUNS. Two examples follow. Figure D-10 has the SGS statements, skeleton, and three elements in H7RUNS generated for environments 1 through 3, one repetition apiece, QA report, and three copies of the K/V and Allocation reports. Figure D-11 has SGS statements, skeleton, and one element in H7RUNS generated for environment 4, three repetitions, no QA report, and one copy of the K/V and allocation reports.

```
SSG   SGS STATEMENTS

SGS
RUNID 30
USERID H7
PFILES *H7AFP *98AFP
FORCE HM80
FORCE1CH H
ENV 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
ENVFIRST 1
ENVLAST 3
SUPP 01 06 11 04 01 06 11 04 01 06 11 04 01 06 11 04
PNAMES RAPD STATIC RADE BAPD RAPD STATIC RADE BAPD ;
       RAPD STATIC RADE BAPD RAPD STATIC RADE BAPD
LIGHT D D D D N N N N D D D D N N N N
BDIV 1 1 1 3 1 1 1 3 1 1 1 3 1 1 1 3
RDIV 3 1 4 1 3 1 4 1 3 1 4 1 3 1 4 1
ALPHA A B C D E F G H I J K L M N O P
NREPS 1
KVCOPIES 3
BASE *H7BASEDATA
PKS H7PKS
RNGDST *H7RNGDST
PREF *H7PREF
CSCSS *H7CSCSS
ENGAGE *H7ENGAGE RAPD
PCAS *H7PCAS H
CVALS *H7GLOBAL CVALS
FRACTS *H7GLOBAL FRACTS
GDNSTY1 T T T F T T T F T T T F T T T F
GDNSTY2 F F F T F F F T F F F T F F F T
TYPLIM  1  60  1  60
SUPERTRP ''10 8 5 ''
SUPERB ''1 2 3 4 5 6 42 44''
SUPERP ''1 2 3 42 44''
QAREP
QAPKS 0
RPTPS1 '' 1 2 3 4 5 6 7 8 9 10 ''
RPTPS1 '' 11 12 13 14 15 16 17 18 19 20 ''
RPTPS1 '' 21 22 23 24 25 26 27 28 29 30 ''
RPTPS1 '' 31 32 33 34 35 36 37 38 39 40 ''
RPTPS1 '' 41 42 43 44 45 46 47 48 49 50 ''
RPTPS1 '' 51 52 53 54 55 56 57 58 59 60 ''
RPTPS2 '' 1 2 3 4 5 6 7 8 9 10 ''
RPTPS2 '' 11 12 13 14 15 16 17 18 19 20 ''
RPTPS2 '' 21 22 23 24 25 26 27 28 29 30 ''
RPTPS2 '' 31 32 33 34 35 36 37 38 39 40 ''
RPTPS2 '' 41 42 43 44 45 46 47 48 49 50 ''
RPTPS2 '' 51 52 53 54 55 56 57 58 59 60 ''
AIR1   '' 1    41 ''
AIR2   '' 1    41   ''
OJTPD '' 80''
OKTHTR ''  E''
OJVIS 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
OJDAY 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2
OJPOS 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
IBFOR ''  2001''
IRFOR ''  1000''
ICOMBO ''  1''
TVALON TRUE
@EOF
```

Figure D-10. SSG Example 1
(page 1 of 14 pages)

SSG   REVISED SKELETON

```
0   SKEL
0   *INCREMENT IE FROM [ENVFIRST,1,1,1] TO [ENVLAST,1,1,1]
1     *CREATE SGS: COMBAT [ENV,1,IE,1]
1     *CREATE SGS: PREFIX [USERID,1,1,1][FORCE,1,1,1]
1     *CREATE SGS: CXPS [ENV,1,IE,1]
1     *CREATE SGS: CSELT [FORCE,1,1,1]E[SUPP,1,IE,1]
1     *CREATE SGS: BASELT [FORCE,1,1,1]E[ENV,1,IE,1]
1     *CREATE SGS: PKSELT [PNAMES,1,IE,1]-[LIGHT,1,IE,1]/[FORCE1CH,1,1,1]
1     *CREATE SGS: RNGELT [PNAMES,1,IE,1]-DEEP/[FORCE1CH,1,1,1]
1     *CREATE SGS: PRFELT [PNAMES,1,IE,1]/[FORCE1CH,1,1,1]
1     *CREATE SGS: INVNTR H7AFPFILES INV[FORCE,1,1,1]
1     *CREATE SGS: NDIV1 [BDIV,1,IE,1]
1     *CREATE SGS: NDIV2 [RDIV,1,IE,1]
1     *CREATE SGS: DNSTY1 [GDNSTY1,1,IE,1]
1     *CREATE SGS: DNSTY2 [GDNSTY2,1,IE,1]
1     *BRKPT,K GO[FORCE,1,1,1]E[COMBAT,1,1,1]N[NREPS,1,1,1]
1     *EDIT ON
1     #RUN,/TPR DO[RUNID,1,1,1][ALPHA,1,IE,1][NREPS,1,1,1],E1899P2277D,&
1     SECRET,600,6300    . KNOX (703) 476-4923
1     #QUAL UNCLASSIFIED
1     #ASG,A H7AFPFILES/XXX/XXX.
1     #ASG,A H7RUNS/XXX/XXX.
1     *INCREMENT L TO [PFILES]
2       *INCREMENT M TO [PFILES,1]
3         #ASG,A [PFILES,L,M,1].
2       *LOOP
1     *LOOP
1     #HDG,S **** [FORCE,1,1,1] ENV [COMBAT,1,1,1] SEEDS [NREPS,1] ****
1     #PRT,S H7RUNS.GO[FORCE,1,1,1]F[COMBAT,1,1,1]N[NREPS,1,1,1]
1     # .  SAVE THE PROGRAMS
1     #ASG,T ABS-FILE.,///10000
1     #COPY,AC *H7AFP.870PKSGEN,ABS-FILE.870PKSGEN
1     #COPY,AC *H7AFP.870PROJGEN,ABS-FILE.870PROJGEN
1     #COPY,AC *H7AFP.870PREFGEN,ABS-FILE.870PREFGEN
1     #COPY,AC *H7AFP.870RNGDSTGEN,ABS-FILE.870RNGDSTGEN
1     #COPY,AC *H7AFP.870MAIN,ABS-FILE.870MAIN
1     #COPY,AC *98AFP.870GENKV,ABS-FILE.870GENKV
1     #COPY,AC *98AFP.870GENAL,ABS-FILE.870GENAL
1     #COPY,AC *98AFP.870PRTKV,ABS-FILE.870PRTKV
1     #COPY,AC *H7AFP.870CXPS,ABS-FILE.870CXPS
1     #COPY,AC *H7AFP.870REPORT,ABS-FILE.870REPORT
1     #COPY,AC *H7AFP.PAUSE,ABS-FILE.PAUSE
1     *INCREMENT L TO [PFILES]
2       *INCREMENT M TO [PFILES,1]
3         #FREE [PFILES,L,M,1].
2       *LOOP
1     *LOOP
1     *IF [COMBAT]
2       #ASG,T 20.,///4000
2       #ASG,T 21.,///4000
2       #ASG,T 22.,///4000
2       #ASG,T 23.,///4000
2       #ASG,T 24.,///5000
2       #ASG,T 25.,///4000
2       #ASG,T 26.,///4000
2       #ASG,T  3.,///2000
2       #ASG,T  4.,///2000
2       #ASG,T  7.,///2000
2       # .    UNIT 35 IS USED FOR TEMPORARY STORAGE
2       #ASG,T 35.,///2000
2       #ASG,T 34.,///25000
2       #ASG,T  9.,///500
2       #ASG,T  31.,///7000
2       #ASG,T  32.,///13000
2       #ASG,T  33.,///2000
2       *SET E TO 1
2       # .            I N P U T    F I L E S
2       # .
2       # .  LOCATE THE ELEMENTS FOR ENVIRONMENT DEPENDENT FILES
2       # .
2       #ASG,A [BASE,1,1,1].        . BASEDATA FILE
2       #ASG,T 15.
2       #ED [BASE,1,1,1].[BASELT,1,1,1],15.
2       #PRT,S [BASE,1,1,1].[BASELT,1,1,1]
2       #FREE [BASE,1,1,1].
2       #ASG,A [PKS,1,1,1]/XXX/XXX.       . PKS
2       #ASG,T 10.
2       #ED [PKS,1,1,1].[PKSELT,1,1,1],10.
2       #FREE [PKS,1,1,1].
2       #ASG,A [RNGDST,1,1,1].          . RANGE DISTRIBUTION
2       #ASG,T 11.
2       #ED [RNGDST,1,1,1].[RNGELT,1,1,1],11.
```

SSG   REVISED SKELETON

Figure D-10. SSG Example 1
(page 2 of 14 pages)

SSG REVISED SKELETON

```
2        #FREE [RNGDST,1,1,1].
2        #ASG,A [PREF,1,1,1].           . PREFERENCE FILE
2        #ASG,T 12.
2        #ED [PREF,1,1,1].[PRFELT,1,1,1],12.
2        #FREE [PREF,1,1,1].
2        #ASG,A [CSCSS,1,1,1].
2        #ASG,T 14.
2        #ED [CSCSS,1,1,1].[CSELT,1,1,1],14.
2        #FREE [CSCSS,1,1,1]
2        #ASG,A [ENGAGE,1,1,1].         . ENGAGEMENT FILE
2        #ASG,T 13.
2        #ED [ENGAGE,1,1,1].[ENGAGE,1,2,1],13.
2        #FREE [ENGAGE,1,1,1].
2        #ASG,A [PCAS,1,1,1].           . PERSONNEL CASUALTY FILE
2        #ASG,T 16.
2        #ED [PCAS,1,1,1].[PCAS,1,2,1],16.
2        #FREE [PCAS,1,1,1].
2        #ASG,A [USERID,1,1,1]KV[FORCE,1,1,1]/XXX/XXX.    . KV FILE
2        #ERS 20.
2        #ERS 21.
2        #ERS 25.
2        #ERS 26.
2        #XQT ABS-FILE.870PREFGEN
2         1 1 1 1 1 1
2        #XQT ABS-FILE.870RNGDSTGEN
2         1 1 1 1 1 1
2        #XQT ABS-FILE.870PKSGEN
2         1 1 1 1 1 1
2        #XQT ABS-FILE.870PROJGEN
2         1 1 1 1 1 1
2        *INCREMENT IR TO [NREPS,1]
3          *CREATE SGS: PEPS [NREPS,1,IR,1]
3          *SET R TO [REPS,1,1,1]
3          #ERS 22.
3          #ERS 23.
3          #ERS 24.
3          #ERS  3.
3          #ERS  4.
3          #ERS  9.
3          #ERS 31.
3          #ERS 32.
3          #ERS 34.
3          #XQT ABS-FILE.870MAIN
3           1 1 1 1 1 1
3          [*R]
3          [DNSTY1,1,E,1]  [DNSTY2,1,E,1]
3          [TYPLIM,1,1,1] [TYPLIM,1,2,1]   [TYPLIM,1,3,1] [TYPLIM,1,4,1]
3          #ASG,T SORT33.,///1000
3          #SORT,ES
3          VOLUME=SMALL
3          KEYW=3,35,36,B,A:4,35,36,P,A:2,35,36,B,A:1,35,36,B,A
3          FILEIN=33.
3          FILEOUT=SORT33.
3          #EOF
3          #USE 33.,SORT33.
3          #ASG,T SORT9.,///1000
3          #SORT,ES
3          VOLUME=SMALL
3          KEYW=3,35,36,B,A:4,35,36,P,A:2,35,36,B,A:1,35,36,B,A
3          FILEIN=9.
3          FILEOUT=SORT9.
3          #EOF
3          #USE 9.,SORT9.
3          #ASG,T SORT24.,///1000
3          #SORT,ES
3          VOLUME=SMALL
3          KEYW=3,35,36,B,A:4,35,36,P,A:2,35,36,B,A:1,35,36,B,A
3          FILEIN=24.
3          FILEOUT=SORT24.
3          #EOF
3          #USE 24.,SORT24.
3          #ASG,T SORT32.,///1000
3          #SORT,ES
3          VOLUME=SMALL
3          KEYW=3,35,36,B,A:4,35,36,P,A:2,35,36,B,A:1,35,36,B,A
3          FILEIN=32.
3          FILEOUT=SORT32.
3          #EOF
3          #USE 32.,SORT32.
3          # .  SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
3          #ASG,A [PREFIX,1,1,1]/XXX/XXX.
3          # . COPY,I 3.,[PREFIX,1,1,1].A[COMBAT,1,E,1]R[*R]
```

SSG REVISED SKELETON

Figure D-10. SSG Example 1
(page 3 of 14 pages)

SSG REVISED SKELETON

```
3          # . COPY,I 4.,[PREFIX,1,1,1].F[COMBAT,1,E,1]R[*R]
3          # . COPY,I 9.,[PREFIX,1,1,1].T[COMBAT,1,E,1]R[*R]
3          *IF [QAPEP]
4            #DELETE,C [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.
4            #XQT ABS-FILE.PAUSE
4            #CAT,P [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.,F///20000
4            #ASG,A [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.
4            #KEEP [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R].
4            #BRKPT PRINT$/[PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]
4            #XQT ABS-FILE.870REPORT
4            [QAPKS,1,1,1]
4             1 1 1 1 1 1
4            *INCREMENT K TO [RPTPS1]
5              *INCREMENT L TO [RPTPS1,K]
6                [RPTPS1,K,L,1]
5              *LOOP
4            *LOOP
4            9999
4            *INCREMENT K TO [RPTPS2]
5              *INCREMENT L TO [RPTPS2,K]
6                [RPTPS2,K,L,1]
5              *LOOP
4            *LOOP
4            9999
4            #EOF
4            #BRKPT PRINT$
4            # . SYM,U [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R].
3          *END
3          #DELETE,C [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.
3          #XQT ABS-FILE.PAUSE
3          #CAT,P [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.,F///20000
3          #ASG,A [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.
3          #KEEP [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R].
3          #BRKPT PRINT$/[PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]
3          #XQT ABS-FILE.870GENKV
3          #XQT ABS-FILE.870PRTKV
3          KV SCOREBOARD FOR [FORCE,1,1,1] ENV [COMBAT,1,E,1]   SEED [*R]
3          [SUPERTRP,1,1,1]
3          [SUPERB,1,1,1]
3          [SUPERR,1,1,1]
3          #ADD,E [INVNTR,1,1,1].[INVNTR,1,2,1]
3          #ADD,E 7.
3          #ED 7.,[USERID,1,1,1]KV[FORCE,1,1,1].E[COMBAT,1,E,1]R[*R]
3          # . COPY KV FILE FOR KV ROLLUP
3          #ASG,T 7.     . FOLLOWING PROGRAM WRITES TO UNIT 7
3          #XQT ABS-FILE.870GENAL
3          # . PRINT THE ALLOCATION REPORT
3          #XQT ABS-FILE.870PRTKV
3          ALLOCATION REPORT FOR [FORCE,1,1,1] ENV [COMBAT,1,E,1] SEED [*R]
3          [SUPERTRP,1,1,1]
3          [SUPERB,1,1,1]
3          [SUPERR,1,1,1]
3          #ADD,E [INVNTR,1,1,1].[INVNTR,1,2,1]
3          #ADD,E 7.
3          *IF [CXPS]
4            *IF NOT [COMBAT]
5              #ASG,T 7.,///2000
5              #ASG,A [BASE,1,1,1].        . BASEDATA FILE
5              #ASG,T 15.
5              #ED [BASE,1,1,1].[BASFLT,1,1,1],15.
5              #FREE [BASE,1,1,1].
5              #ASG,A [CSCSS,1,1,1].
5              #ASG,T 14.
5              #ED [CSCSS,1,1,1].[CSELT,1,1,1],14.
5              #FREE [CSCSS,1,1,1]
5              #ASG,A [PCAS,1,1,1].          . PERSONNEL CASUALTY FILE
5              #ASG,T 16.
5              #ED [PCAS,1,1,1].[PCAS,1,2,1],16.
5              #FREE [PCAS,1,1,1].
5              # .  END OF NOT COMBAT IF
4            *END
4            #HDG    ** [FORCE,1,1,1] CXPS ENV [CXPS,1,E,1]   SEED [*R] **
4            # .  CREATE OUTPUT FILES FOR CXPS
4            #ASG,T 29.
4            #ASG,T 30.
4            # .  RUN THE MERGE MODULE
4            # .  RUN WITH FILES DEVELOPED EARLIER
4            # . COPY,I [PREFIX,1,1,1].A[CXPS,1,E,1]R[*R],3.
4            # . COPY,I [PREFIX,1,1,1].F[CXPS,1,E,1]R[*R],4.
4            # . COPY,I [PREFIX,1,1,1].T[CXPS,1,E,1]R[*R],9.
4            #ERS 7.
4            #XQT ABS-FILE.870CXPS
```

SSG REVISED SKELETON

Figure D-10. SSG Example 1
(page 4 of 14 pages)

SSG REVISED SKELETON

```
4              *. IMAGE GIVING INDICES IN CBT POT OUTPUT RECORDS
4              *EDIT ON
4              [ICOMBO,1,1,1]&
4              [OJTPC,1,1,1]&
4              [OKTHTR,1,1,1]&
4                 [OJVIS,1,IE,1]&
4                 [OJDAY,1,IE,1]&
4                 [OJPOS,1,IE,1]&
4              [IBFOR,1,1,1]&
4              [IRFOR,1,1,1]
4              *. IMAGE GIVING INDICES OF INPUT CS/CSS MODULI
4              [TVALON,1,1,1]
4              1  1  1  1  1   1
4              [NDIV1,1,E,1]  [NDIV2,1,E,1]
4              [AIR1,1,1,1]
4              [AIR2,1,1,1]
4              #ADD [CVALS,1,1,1].[CVALS,1,2,1]
4              #ADD [FRACTS,1,1,1].[FRACTS,1,2,1]
4              # .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
4              #DATA,L 29.
4              #END
4              #COPY,I 29.,[PREFIX,1,1,1].E[CXPS,1,E,1]R[*R]
4              #ASG,T SORTOUT.
4              #SORT,S
4              VOLUME=SMALL
4              KEY=1,5,CH,A:67,10,CH,A
4              FILEIN=29.
4              FILEOUT=SORTOUT.
4              #ED,R SORTOUT.
4              LNP!
4              EXI
3            *END
3            #BRKPT PRINT$
3            #SYM,U [PREFIX,1,1,1]][COMBAT,1,E,1]R[*R].,[KVCOPIES,1,1,1]
3            *REMOVE SGS REPS
2          *LOOP     .IR
2          *.  END OF CXPS IF
2          *.  END OF ENVIRONMENT IF
1        *END
1        #FIN
1        *REMOVE SGS COMBAT
1        *REMOVE SGS PREFIX
1        *REMOVE SGS CXPS
1        *REMOVE SGS CSELT
1        *REMOVE SGS BASELT
1        *REMOVE SGS PKSELT
1        *REMOVE SGS RNGELT
1        *REMOVE SGS PRFELT
1        *REMOVE SGS INVNTR
1        *REMOVE SGS NDIV1
1        *REMOVE SGS NDIV2
1        *REMOVE SGS DNSTY1
1        *REMOVE SGS DNSTY2
0      *LOOP     .IE
0      @EOF
```

SSG STREAM GENERATION RUN LOG PART 1

```
COMBAT 01
PREFIX H7HM80
CXPS 01
CSELT HM80E01
BASELT HM80E01
PKSELT RAPD-D/H
RNGELT RAPD-DEEP/H
PRFELT RAPD/H
INVNTR H7AFPFILES INVHM80
NDIV1 1
NDIV2 3
DNSTY1 T
DNSTY2 F
```

SSG GENERATED OUTPUT STREAM PART 1

```
    1.        1     SKEL
```

SSG STREAM GENERATION RUN LOG PART 1

```
REPS 1
COMBAT 02
PREFIX H7HM80
CXPS 02
CSELT HM80E06
BASELT HM80E02
PKSELT STATIC-D/H
RNGELT STATIC-DEEP/H
PRFELT STATIC/H
INVNTR H7AFPFILES INVHM80
NDIV1 1
NDIV2 1
DNSTY1 T
DNSTY2 F
```

Figure D-10. SSG Example 1
(page 5 of 14 pages)

SSG GENERATED OUTPUT STREAM PART)

```
@RUN,/TPR 0030A1,E1399P2277D,SECRET,600,6300    . KNOX (703) 476-4923
@QUAL UNCLASSIFIED
@ASG,A H7AFPFILES/XXX/XXX.
@ASG,A H7RUNS/XXX/XXX.
@ASG,A *H7AFP.
@ASG,A *98AFP.
@HDG,S **** HM80 ENV 01 SEEDS 1 ****
@PRT,S H7RUNS.GOHM80ED1N1
@ .  SAVE THE PROGRAMS
@ASG,T ABS-FILE.,///1000
@COPY,AC *H7AFP.870PKSGEN,ABS-FILE.870PKSGEN
@COPY,AC *H7AFP.870PROJGEN,ABS-FILE.870PROJGEN
@COPY,AC *H7AFP.870PREFGEN,ABS-FILE.870PREFGEN
@COPY,AC *H7AFP.870RNGDSTGEN,ABS-FILE.870RNGDSTGEN
@COPY,AC *H7AFP.870MAIN,ABS-FILE.870MAIN
@COPY,AC *98AFP.870GENKV,ABS-FILE.870GENKV
@COPY,AC *98AFP.870GENAL,ABS-FILE.870GENAL
@COPY,AC *98AFP.870PRTKV,ABS-FILE.870PRTKV
@COPY,AC *H7AFP.870CXPS,ABS-FILE.870CXPS
@COPY,AC *H7AFP.870REPORT,ABS-FILE.870REPORT
@COPY,AC *H7AFP.PAUSE,ABS-FILE.PAUSE
@FREE *H7AFP.
@FREE *98AFP.
@ASG,T 20.,///4000
@ASG,T 21.,///4000
@ASG,T 22.,///4000
@ASG,T 23.,///4000
@ASG,T 24.,///5000
@ASG,T 25.,///4000
@ASG,T 26.,///4000
@ASG,T  3.,///2000
@ASG,T  4.,///2000
@ASG,T  7.,///2000
@ .   UNIT 35 IS USED FOR TEMPORARY STORAGE
@ASG,T 35.,///2000
@ASG,T 34.,///25000
@ASG,T  9.,///500
@ASG,T  31.,///7000
@ASG,T  32.,///13000
@ASG,T  33.,///2000
@ .           I N P U T    F I L E S
@ .
@ .  LOCATE THE ELEMENTS FOR ENVIRONMENT DEPENDENT FILES
@ .
@ASG,A *H7BASEDATA.        . BASEDATA FILE
@ASG,T 15.
@ED *H7BASEDATA.HM80ED1,15.
@PRT,S *H7BASEDATA.HM80ED1
@FREE *H7BASEDATA.
@ASG,A H7PKS/XXX/XXX.       . PKS
@ASG,T 10.
@ED H7PKS.RAPD-D/H,10.
@FREE H7PKS.
@ASG,A *H7RNGDST.          . RANGE DISTRIBUTION
@ASG,T 11.
@ED *H7RNGDST.RAPD-DEEP/H,11.
@FREE *H7RNGDST.
@ASG,A *H7PREF.            . PREFERENCE FILE
@ASG,T 12.
@ED *H7PREF.RAPD/H,12.
@FREE *H7PREF.
@ASG,A *H7CSCSS.
@ASG,T 14.
@ED *H7CSCSS.HM80ED1,14.
@FREE *H7CSCSS
@ASG,A *H7ENGAGE.          . ENGAGEMENT FILE
@ASG,T 13.
@ED *H7ENGAGE.RAPD,13.
@FREE *H7ENGAGE.
@ASG,A *H7PCAS.            . PERSONNEL CASUALTY FILE
@ASG,T 16.
@ED *H7PCAS.H,16.
@FREE *H7PCAS.
@ASG,A H7KVHM80/XXX/XXX.    . KV FILE
@ERS 20.
@ERS 21.
@ERS 25.
@ERS 26.
@XQT ABS-FILE.870PREFGEN
          1          1 1 1 1 1 1
@XQT ABS-FILE.870RNGDSTGEN
          1          1 1 1 1 1 1
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-10. SSG Example 1
(page 6 of 14 pages)

```
SSG   GENERATED OUTPUT STREAM   PART  )

            @XQT ABS-FILE.870PKSGEN
      1       1 1 1 1 1 1
            @XQT ABS-FILE.870PROJGEN
      1       1 1 1 1 1 1
            @ERS 22.
            @ERS 23.
            @ERS 24.
            @ERS  3.
            @ERS  4.
            @ERS  9.
            @ERS 31.
            @ERS 32.
            @ERS 34.
            @XQT ABS-FILE.870MATN
      1       1 1 1 1 1 1
      2     1
      3     T  F
      4     1 60    1 60
            @ASG,T SORT33.,///1000
            @SORT,ES
      1     VOLUME=SMALL
      2     KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
      3     FILEIN=33.
      4     FILEOUT=SORT33.
            @EOF
            @USE 33.,SORT33.
            @ASG,T SORT9.,///1000
            @SORT,ES
      1     VOLUME=SMALL
      2     KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
      3     FILEIN=9.
      4     FILEOUT=SORT9.
            @EOF
            @USE 9.,SORT9.
            @ASG,T SORT24.,///1000
            @SORT,ES
      1     VOLUME=SMALL
      2     KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
      3     FILEIN=24.
      4     FILEOUT=SORT24.
            @EOF
            @USE 24.,SORT24.
            @ASG,T SORT32.,///1000
            @SORT,ES
      1     VOLUME=SMALL
      2     KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
      3     FILEIN=32.
      4     FILEOUT=SORT32.
            @EOF
            @USE 32.,SORT32.
            @ .  SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
            @ASG,A H7HM80/XXX/XXX.
            @ . COPY,I 3.,H7HM80.A01R1
            @ . COPY,I 4.,H7HM80.F01R1
            @ . COPY,I 9.,H7HM80.T01R1
            @DELETE,C H7HM80R01R1/XXX/XXX.
            @XQT ABS-FILE.PAUSE
            @CAT,P H7HM80R01R1/XXX/XXX.,F///20000
            @ASG,A H7HM80R01R1/XXX/XXX.
            @KEEP H7HM80R01R1.
            @BRKPT PRINT$/H7HM80R01P1
            @XQT ABS-FILE.870REPORT
      1     0
      2       1 1 1 1 1 1
      3       1 2 3 4 5 6  7  8  9 10
      4      11 12 13 14 15 16 17 18 19 20
      5      21 22 23 24 25 26 27 28 29 30
      6      31 32 33 34 35 36 37 38 39 40
      7      41 42 43 44 45 46 47 48 49 50
      8      51 52 53 54 55 56 57 58 59 60
      9     9999
     10       1 2 3 4 5 6  7  8  9 10
     11      11 12 13 14 15 16 17 18 19 20
     12      21 22 23 24 25 26 27 28 29 30
     13      31 32 33 34 35 36 37 38 39 40
     14      41 42 43 44 45 46 47 48 49 50
     15      51 52 53 54 55 56 57 58 59 60
     16     9999
            @EOF
            @BRKPT PRINT$
            @ . SYM,U H7HM80R01R1.
            @DELETE,C H7HM80D01R1/XXX/XXX.

SSG   GENERATED OUTPUT STREAM   PART  )
```

Figure D-10. SSG Example 1
(page 7 of 14 pages)

```
SSG   GENERATED OUTPUT STREAM   PART  )

                    @XQT ABS-FILE.PAUSE
                    @CAT,P H7HM80D01R1/XXX/XXX.,F///20000
                    @ASG,A H7HM80D01R1/XXX/XXX.
                    @KEEP H7HM80D01R1.
                    @BRKPT PRINT$/H7HM80D01P1
                    @XQT ABS-FILE.870GENKV
                    @XQT ABS-FILE.870PRTKV
                    KV SCOREBOARD FOR HM80  ENV 01   SEED 1
                    10 8 5
                    1 2 3 4 5 6 42 44
                    1 2 3 42 44
                    @ADD,E H7AFPFILES.INVHM80
                    @ADD,E 7.
                    @ED 7.,H7KVHM80.E01R1
                    @ . COPY KV FILE FOR KV ROLLUP
                    @ASG,T 7.     . FOLLOWING PROGRAM WRITES TO UNIT 7
                    @XQT ABS-FILE.870GENVAL
                    @ . PRINT THE ALLOCATION REPORT
                    @XQT ABS-FILE.870PRTKV
                    ALLOCATION REPORT FOR HM80 ENV 01 SEED 1
                    10 8 5
                    1 2 3 4 5 6 42 44
                    1 2 3 42 44
                    @ADD,E H7AFPFILES.INVHM80
                    @ADD,E 7.
                    @HDG    ** HM80 CXPS ENV 01  SEED 1 **
                    @ .  CREATE OUTPUT FILES FOR CXPS
                    @ASG,T 29.
                    @ASG,T 30.
                    @ .  RUN THE MERGE MODULE
                    @ .  RUN WITH FILES DEVELOPED EARLIER
                    @ . COPY,I H7HM80.A01R1,3.
                    @ . COPY,I H7HM80.F01R1,4.
                    @ . COPY,I H7HM80.T01R1,9.
                    @ERS 7.
                    @XQT ABS-FILE.870CXPS
                      1 80  E  I  1  1  2001  1000
                    TRUE
                    1  1  1  1  1  1
                    1  3
                     1    41
                     1    41
                    @ADD *H7GLOBAL.CVALS
                    @ADD *H7GLOBAL.FRACTS
                    @ .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
                    @DATA,L 29.
                    @END
                    @COPY,I 29.,H7HM80.F01R1
                    @ASG,T SORTOUT.
                    @SORT,S
                    VOLUME=SMALL
                    KEY=1,5,CH,A:67,10,CH,A
                    FILEIN=29.
                    FILEOUT=SORTOUT.
                    @ED,R SORTOUT.
                    LNP!
                    EXI
                    @BRKPT PRINT$
                    @SYM,U H7HM80D01R1.,3
                    @FIN

SSG   STREAM GENERATION RUN LOG PART  )

REPS 1
COMBAT 03
PREFIX H7HM80
CXPS 03
CSELT HM80E11
BASELT HM80E03
PKSELT RADE-O/H
RNGELT RADE-DEEP/H
PPFELT RADE/H
INVNTR H7AFPFILES INVHM80
NDIV1 1
NDIV2 4
DNSTY1 T
DNSTY2 F
```

Figure D-10. SSG Example 1
(page 8 of 14 pages)

SSG GENERATED OUTPUT STREAM PART)

```
@RUN,/TPR 0030B1,E1599P2277D,SECRET,600,6300   . KNOX (703) 476-4923
@QUAL UNCLASSIFIED
@ASG,A H7AFPFILES/XXX/XXX.
@ASG,A H7RUNS/XXX/XXX.
@ASG,A *H7AFP.
@ASG,A *98AFP.
@HDG,S **** HM80 ENV E2 SEEDS 1 ****
@PRT,S H7RUNS.GOHM80E02N1
@ .  SAVE THE PROGRAMS
@ASG,T ABS-FILE.,///10000
@COPY,AC *H7AFP.870PKSGEN,ABS-FILE.870PKSGEN
@COPY,AC *H7AFP.870PROJGEN,ABS-FILE.870PROJGEN
@COPY,AC *H7AFP.870PREFGEN,ABS-FILE.870PREFGEN
@COPY,AC *H7AFP.870RNGDSTGEN,ABS-FILE.870RNGDSTGEN
@COPY,AC *H7AFP.870MAIN,ABS-FILE.870MAIN
@COPY,AC *98AFP.870GENKV,ABS-FILE.870GENKV
@COPY,AC *98AFP.870GENAL,ABS-FILE.870GENAL
@COPY,AC *98AFP.870PRTKV,ABS-FILE.870PRTKV
@COPY,AC *H7AFP.870CXPS,ABS-FILE.870CXPS
@COPY,AC *H7AFP.870REPORT,ABS-FILE.870REPORT
@COPY,AC *H7AFP.PAUSE,ABS-FILE.PAUSE
@FREE *H7AFP.
@FREE *98AFP.
@ASG,T 20.,///4000
@ASG,T 21.,///4000
@ASG,T 22.,///4000
@ASG,T 23.,///4000
@ASG,T 24.,///5000
@ASG,T 25.,///4000
@ASG,T 26.,///4000
@ASG,T  3.,///2000
@ASG,T  4.,///2000
@ASG,T  7.,///2000
@ .  UNIT 35 IS USED FOR TEMPORARY STORAGE
@ASG,T 35.,///2000
@ASG,T 34.,///25000
@ASG,T  9.,///500
@ASG,T  31.,///7000
@ASG,T  32.,///13000
@ASG,T  33.,///2000
@ .          I N P U T    F I L E S
@ .
@ .  LOCATE THE ELEMENTS FOR ENVIRONMENT DEPENDENT FILES
@ .
@ASG,A *H7BASEDATA.        . BASEDATA FILE
@ASG,T 15.
@ED *H7BASEDATA.HM80E02,15.
@PRT,S *H7BASEDATA.HM80E02
@FREE *H7BASEDATA.
@ASG,A H7PKS/XXX/XXX.      . PKS
@ASG,T 10.
@ED H7PKS.STATIC-D/W,10.
@FREE H7PKS.
@ASG,A *H7RNGDST.          . RANGE DISTRIBUTION
@ASG,T 11.
@ED *H7RNGDST.STATIC-DEFP/H,11.
@FREE *H7RNGDST.
@ASG,A *H7PREF.            . PREFERENCE FILE
@ASG,T 12.
@ED *H7PREF.STATIC/H,12.
@FREE *H7PREF.
@ASG,A *H7CSCSS.
@ASG,T 14.
@ED *H7CSCSS.HM80E05,14.
@FREE *H7CSCSS
@ASG,A *H7ENGAGE.          . ENGAGEMENT FILE
@ASG,T 13.
@ED *H7ENGAGE.RAPD,13.
@FREE *H7ENGAGE.
@ASG,A *H7PCAS.            . PERSONNEL CASUALTY FILE
@ASG,T 16.
@ED *H7PCAS.H,16.
@FREE *H7PCAS.
@ASG,A H7KVHM80/XXX/XXX.   . KV FILE
@ERS 20.
@ERS 21.
@ERS 25.
@ERS 26.
@XQT ABS-FILE.870PREFGEN
         1        1 1 1 1 1 1
@XQT ABS-FILE.870RNGDSTGEN
         1        1 1 1 1 1 1
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-10. SSG Example 1
(page 9 of 14 pages)

SSG GENERATED OUTPUT STREAM PART)

```
             @XQT ABS-FILE.870PKSGEN
        1     1 1 1 1 1 1
             @XQT ABS-FILE.870PROJGEN
        1     1 1 1 1 1 1
             @ERS 22.
             @ERS 23.
             @ERS 24.
             @ERS  3.
             @ERS  4.
             @ERS  9.
             @ERS 31.
             @ERS 32.
             @ERS 34.
             @XQT ABS-FILE.870MAIN
        1     1 1 1 1 1 1
        2    1
        3    T  F
        4    1 60   1 60
             @ASG,T SORT33.,///1000
             @SORT,ES
        1    VOLUME=SMALL
        2    KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
        3    FILEIN=33.
        4    FILEOUT=SORT33.
             @EOF
             @USE 33.,SORT33.
             @ASG,T SORT9.,///1000
             @SORT,ES
        1    VOLUME=SMALL
        2    KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
        3    FILEIN=9.
        4    FILEOUT=SORT9.
             @EOF
             @USE 9.,SORT9.
             @ASG,T SORT24.,///1000
             @SORT,ES
        1    VOLUME=SMALL
        2    KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
        3    FILEIN=24.
        4    FILEOUT=SORT24.
             @EOF
             @USE 24.,SORT24.
             @ASG,T SORT32.,///1000
             @SORT,ES
        1    VOLUME=SMALL
        2    KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
        3    FILEIN=32.
        4    FILEOUT=SORT32.
             @EOF
             @USE 32.,SORT32.
             @ .  SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
             @ASG,A H7HM80/XXX/XXX.
             @ . COPY,I 3.,H7HM80.A02R1
             @ . COPY,I 4.,H7HM80.F02R1
             @ . COPY,I 9.,H7HM80.T02R1
             @DELETE,C H7HM80R02R1/XXX/XXX.
             @XQT ABS-FILE.PAUSE
             @CAT,P H7HM80R02R1/XXX/XXX.,F///20000
             @ASG,A H7HM80R02R1/XXX/XXX.
             @KEEP H7HM80R02R1.
             @BRKPT PRINT$/H7HM80P02R1
             @XQT ABS-FILE.870REPORT
        1    0
        2     1  1  1  1  1  1
        3     1  2  3  4  5  6  7  8  9 10
        4    11 12 13 14 15 16 17 18 19 20
        5    21 22 23 24 25 26 27 28 29 30
        6    31 32 33 34 35 36 37 38 39 40
        7    41 42 43 44 45 46 47 48 49 50
        8    51 52 53 54 55 56 57 58 59 60
        9    9999
       10     1  2  3  4  5  6  7  8  9 10
       11    11 12 13 14 15 16 17 18 19 20
       12    21 22 23 24 25 26 27 28 29 30
       13    31 32 33 34 35 36 37 38 39 40
       14    41 42 43 44 45 46 47 48 49 50
       15    51 52 53 54 55 56 57 58 59 60
       16    9999
             @EOF
             @BRKPT PRINT$
             @ . SYM,U H7HM80R02R1.
             @DELETE,C H7HM80D02R1/XXX/XXX.
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-10. SSG Example 1
(page 10 of 14 pages)

```
SSG   GENERATED OUTPUT STREAM   PART  )

                @XQT ABS-FILE.PAUSE
                @CAT,P H7HM80D02R1/XXX/XXX.,F///20000
                @ASG,A H7HM80D02R1/XXX/XXX.
                @KEEP H7HM80D02R1.
                @BRKPT PRINT$/H7HM8JD02R1
                @XQT ABS-FILE.870GE"KV
                @XQT ABS-FILE.870PRTKV
          1     KV SCOREBOARD FOR HM80  ENV 02   SEED 1
          2     10 8 5
          3     1 2 3 4 5 6 42 44
          4     1 2 3 42 44
                @ADD,E H7AFPFILES.INVHM80
                @ADD,E 7.
                @ED 7.,H7KVHM80.E02?1
                @ . COPY KV FILE FO? KV ROLLUP
                @ASG,T 7.      . FOLLOWINF PROGRAM WRITES TO UNIT 7
                @XQT ABS-FILE.870GENAL
                @ . PRINT THE ALLOCATION REPORT
                @XQT ABS-FILE.870PRTKV
          1     ALLOCATION REPORT FOR HM80 ENV 02 SEED 1
          2     10 8 5
          3     1 2 3 4 5 6 42 44
          4     1 2 3 42 44
                @ADD,E H7AFPFILES.INVHM80
                @ADD,E 7.
                @HDG    ** HM80 CXPS ENV 02  SEED 1 **
                @ .  CREATE OUTPUT FILES FOR CXPS
                @ASG,T 29.
                @ASG,T 30.
                @ .  RUN THE MERGE MODULE
                @ .  RUN WITH FILES DEVELOPED EARLIER
                @ . COPY,I H7HM80.A02R1,3.
                @ . COPY,I H7HM80.F02R1,4.
                @ . COPY,I H7HM80.T?2R1,9.
                @ERS 7.
                @XQT ABS-FILE.870CXPS
          1       1 80  E  1  1  2  2001  1000
          2     TRUE
          3     1  1  1  1  1  1
          4     1  1
          5      1   41
          6      1   41
                @ADD *H7GLOBAL.CVALS
                @ADD *H7GLOBAL.FRACTS
                @ .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
                @DATA,L 29.
                @END
                @COPY,I 29.,H7HM80.E02RI
                @ASG,T SORTOUT.
                @SORT,S
          1     VOLUME=SMALL
          2     KEY=1,5,CH,A:67,10,CH,A
          3     FILEIN=29.
          4     FILEOUT=SORTOUT.
                @ED,R SORTOUT.
          1     LNP!
          2     EXI
                @BRKPT PRINT$
                @SYM,U H7HM80D02R1.,3
                @FIN

SSG   STREAM GENERATION RUN LOG PART  )

REPS 1
```

Figure D-10. SSG Example 1
(page 11 of 14 pages)

SSG GENERATED OUTPUT STREAM PART)

```
                  @RUN,/TPR D030C1,E1*99P2277D,SECRET,600,6300    . KNOX (703) 476-4923
                  @QUAL UNCLASSIFIED
                  @ASG,A H7AFPFILES/XXX/XXX.
                  @ASG,A H7RUNS/XXX/XXX.
                  @ASG,A *H7AFP.
                  @ASG,A *98AFP.
                  @HDG,S **** HM80 ENV 03 SEEDS 1 ****
                  @PRT,S H7RUNS.G0HM80E03N1
                  @ .  SAVE THE PROGRAMS    .
                  @ASG,T ABS-FILE.,///10000
                  @COPY,AC *H7AFP.870PKSGEN,ABS-FILE.870PKSGEN
                  @COPY,AC *H7AFP.870PROJGEN,ABS-FILE.870PROJGEN
                  @COPY,AC *H7AFP.870PREFGEN,ABS-FILE.870PREFGEN
                  @COPY,AC *H7AFP.870RNGDSTGEN,ABS-FILE.870RNGDSTGEN
                  @COPY,AC *H7AFP.870MAIN,ABS-FILE.870MAIN
                  @COPY,AC *98AFP.870GENKV,ABS-FILE.870GENKV
                  @COPY,AC *98AFP.870GENAL,ABS-FILE.870GENAL
                  @COPY,AC *98AFP.870PRTKV,ABS-FILE.870PRTKV
                  @COPY,AC *H7AFP.870CXPS,ABS-FILE.870CXPS
                  @COPY,AC *H7AFP.870REPORT,ABS-FILE.870REPORT
                  @COPY,AC *H7AFP.PAUSE,ABS-FILE.PAUSE
                  @FREE *H7AFP.
                  @FREE *98AFP.
                  @ASG,T 20.,///4000
                  @ASG,T 21.,///4000
                  @ASG,T 22.,///4000
                  @ASG,T 23.,///4000
                  @ASG,T 24.,///5000
                  @ASG,T 25.,///4000
                  @ASG,T 26.,///4000
                  @ASG,T  3.,///2000
                  @ASG,T  4.,///2000
                  @ASG,T  7.,///2000
                  @ .   UNIT 35 IS USED FOR TEMPORARY STORAGE
                  @ASG,T 35.,///2000
                  @ASG,T 34.,///25000
                  @ASG,T  9.,///500
                  @ASG,T  31.,///7000
                  @ASG,T  32.,///13000
                  @ASG,T  33.,///2000
                  @ .          I N P U T    F I L E S
                  @ .
                  @ .  LOCATE THE ELEMENTS FOR ENVIRONMENT DEPENDENT FILES
                  @ .
                  @ASG,A *H7BASEDATA.        . BASEDATA FILE
                  @ASG,T 15.
                  @ED *H7BASEDATA.HM80E03,15.
                  @PRT,S *H7BASEDATA.HM80E03
                  @FREE *H7BASEDATA.
                  @ASG,A H7PKS/XXX/XXX.       . PKS
                  @ASG,T 10.
                  @ED H7PKS.RADE-D/H,10.
                  @FREE H7PKS.
                  @ASG,A *H7RNGDST.          . RANGE DISTRIBUTION
                  @ASG,T 11.
                  @ED *H7RNGDST.RADE-DEEP/H,11.
                  @FREE *H7RNGDST.
                  @ASG,A *H7PREF.            . PREFERENCE FILE
                  @ASG,T 12.
                  @ED *H7PREF.RADE/H,12.
                  @FREE *H7PREF.
                  @ASG,A *H7CSCSS.
                  @ASG,T 14.
                  @ED *H7CSCSS.HM80E11,14.
                  @FREE *H7CSCSS
                  @ASG,A *H7ENGAGE.          . ENGAGEMENT FILE
                  @ASG,T 13.
                  @ED *H7ENGAGE.RAPD,13.
                  @FREE *H7ENGAGE.
                  @ASG,A *H7PCAS.            . PERSONNEL CASUALTY FILE
                  @ASG,T 16.
                  @ED *H7PCAS.H,16.
                  @FREE *H7PCAS.
                  @ASG,A H7KVHM80/XXX/XXX.     . KV FILE
                  @ERS 20.
                  @ERS 21.
                  @ERS 25.
                  @ERS 26.
                  @XQT ABS-FILE.870PREFGEN
         1         1 1 1 1 1 1
                  @XQT ABS-FILE.870RNGDSTGEN
         1         1 1 1 1 1 1
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-10. SSG Example 1
(page 12 of 14 pages)

```
SSG    GENERATED OUTPUT STREAM    PART   )
                   @XQT ABS-FILE.870PKSGEN
           1         1 1 1 1 1 1
                   @XQT ABS-FILE.870PROJGEN
           1         1 1 1 1 1 1
                   @ERS 22.
                   @ERS 23.
                   @ERS 24.
                   @ERS  3.
                   @ERS  4.
                   @ERS  9.
                   @ERS 31.
                   @ERS 32.
                   @ERS 34.
                   @XQT ABS-FILE.870MAIN
           1         1 1 1 1 1 1
           2       1
           3       T  F
           4       1 60   1 60
                   @ASG,T SORT33.,///1000
                   @SORT,ES
           1       VOLUME=SMALL
           2       KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
           3       FILEIN=33.
           4       FILEOUT=SORT33.
                   @EOF
                   @USE 33.,SORT33.
                   @ASG,T SORT9.,///1000
                   @SORT,ES
           1       VOLUME=SMALL
           2       KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
           3       FILEIN=9.
           4       FILEOUT=SORT9.
                   @EOF
                   @USE 9.,SORT9.
                   @ASG,T SORT24.,///1000
                   @SORT,ES
           1       VOLUME=SMALL
           2       KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
           3       FILEIN=24.
           4       FILEOUT=SORT24.
                   @EOF
                   @USE 24.,SORT24.
                   @ASG,T SORT32.,///1000
                   @SORT,ES
           1       VOLUME=SMALL
           2       KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
           3       FILEIN=32.
           4       FILEOUT=SORT32.
                   @EOF
                   @USE 32.,SORT32.
                   @ .  SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
                   @ASG,A H7HM80/XXX/XXX.
                   @ . COPY,I 3.,H7HM80.A03R1
                   @ . COPY,I 4.,H7HM80.F03R1
                   @ . COPY,I 9.,H7HM80.T03R1
                   @DELETE,C H7HM80R03R1/XXX/XXX.
                   @XQT ABS-FILE.PAUSE
                   @CAT,P H7HM80R03R1/XXX/XXX.,F///20000
                   @ASG,A H7HM80R03R1/XXX/XXX.
                   @KEEP H7HM80R03R1.
                   @BRKPT PRINT$/H7HM80R03R1
                   @XQT ABS-FILE.870REPORT
           1       0
           2         1 1 1 1 1 1
           3         1  2  3  4  5  6  7  8  9 10
           4        11 12 13 14 15 16 17 18 19 20
           5        21 22 23 24 25 26 27 28 29 30
           6        31 32 33 34 35 36 37 38 39 40
           7        41 42 43 44 45 46 47 48 49 50
           8        51 52 53 54 55 56 57 58 59 60
           9       9999
          10         1  2  3  4  5  6  7  8  9 10
          11        11 12 13 14 15 16 17 18 19 20
          12        21 22 23 24 25 26 27 28 29 30
          13        31 32 33 34 35 36 37 38 39 40
          14        41 42 43 44 45 46 47 48 49 50
          15        51 52 53 54 55 56 57 58 59 60
          16       9999
                   @EOF
                   @BRKPT PRINT$
                   @ . SYM,U H7HM80R03R1.
                   @DELETE,C H7HM80D03R1/XXX/XXX.
SSG    GENERATED OUTPUT STREAM    PART   )
```

Figure D-10. SSG Example 1
(page 13 of 14 pages)

SSG GENERATED OUTPUT STREAM PART)

```
           @XQT ABS-FILE.PAUSE
           @CAT,P H7HM80D03R1/XXX/XXX.,F///20000
           @ASG,A H7HM80D03R1/XXX/XXX.
           @KEEP H7HM80D03R1.
           @BRKPT PRINT$/H7HM80D03R1
           @XQT ABS-FILE.870GENKV
           @XQT ABS-FILE.870PRTKV
     1     KV SCOREBOARD FOR HM80  ENV 03   SEED 1
     2     10 8 5
     3     1 2 3 4 5 6 42 44
     4     1 2 3 42 44
           @ADD,E H7AFPFILES.INVHM80
           @ADD,E 7.
           @ED 7.,H7KVHM80.E03R1
           @ . COPY KV FILE FOR KV ROLLUP
           @ASG,T 7.     . FOLLOWING PROGRAM WRITES TO UNIT 7
           @XQT ABS-FILE.870GEVAL
           @ . PRINT THE ALLOCATION REPORT
           @XQT ABS-FILE.870PRTKV
     1     ALLOCATION REPORT FOR HM80 ENV 03 SEED 1
     2     10 8 5
     3     1 2 3 4 5 6 42 44
     4     1 2 3 42 44
           @ADD,E H7AFPFILES.INVHM80
           @ADD,E 7.
           @HDG    ** HM80 CXPS ENV 03  SEED 1 **
           @ .  CREATE OUTPUT FILES FOR CXPS
           @ASG,T 29.
           @ASG,T 30.
           @ .  RUN THE MERGE MODULE
           @ .  RUN WITH FILES DEVELOPED EARLIER
           @ . COPY,I H7HM80.A03R1,3.
           @ . COPY,I H7HM80.F03R1,4.
           @ . COPY,I H7HM80.T03R1,9.
           @ERS 7.
           @XQT ABS-FILE.870CXPS
     1       1 80  E  1  1  3  2001  1000
     2     TRUE
     3     1  1  1  1  1  1
     4     1  4
     5      1   41
     6      1   41
           @ADD *H7GLOBAL.CVALS
           @ADD *H7GLOBAL.FRACTS
           @ .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
           @DATA,L 29.
           @END
           @COPY,I 29.,H7HM80.F03RI
           @ASG,T SORTOUT.
           @SORT,S
     1     VOLUME=SMALL
     2     KEY=1,5,CH,A:67,10,CH,A
     3     FILEIN=29.
     4     FILEOUT=SORTOUT.
           @ED,R SORTOUT.
     1     LNP!
     2     EXI
           @BRKPT PRINT$
           @SYM,U H7HM80D03R1.,3
           @FIN
           @EOF
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-10. SSG Example 1
(page 14 of 14 pages)

```
SSG    SGS STATEMENTS

SGS
RUNID 30
USERID H7
PFILES *H7AFP *98AFP
FORCE HM80
FORCEICH H
ENV 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
ENVFIRST 4
ENVLAST 4
SUPP 01 06 11 04 01 06 11 04 01 06 11 04 01 06 11 04
PNAMES RAPD STATIC RADE BAPD RAPD STATIC RADE BAPD ;
       RAPD STATIC RADE BAPD RAPD STATIC RADE BAPD
LIGHT D D D D N N N N D D D D N N N N
BDIV 1 1 1 3 1 1 1 3 1 1 1 3 1 1 1 3
RDIV 3 1 4 1 3 1 4 1 3 1 4 1 3 1 4 1
ALPHA A B C D E F G H I J K L M N O P
NREPS 1 2 3
KVCOPIES 1
BASE *H7BASEDATA
PKS H7PKS
RNGDST *H7RNGDST
PREF *H7PREF
CSCSS *H7CSCSS
ENGAGE *H7ENGAGE RAPD
PCAS *H7PCAS H
CVALS *H7GLOBAL CVALS
FRACTS *H7GLOBAL FRACTS
GDNSTY1 T T T F T T T F T T T F T T T F
GDNSTY2 F F F T F F F T F F F T F F F T
TYPLIM  1  60  1  60
SUPERTRP ""10 8 5 ""
SUPERB ""1 2 3 4 5 6 42 44""
SUPERF ""1 2 3 42 44""
QAREQ
QAPKS 0
RPTPS1 ""  1  2  3  4  5  6  7  8  9  10  ""
RPTPS1 ""  11 12 13 14 15 16 17 18 19 20  ""
RPTPS1 ""  21 22 23 24 25 26 27 28 29 30  ""
RPTPS1 ""  31 32 33 34 35 36 37 38 39 40  ""
RPTPS1 ""  41 42 43 44 45 46 47 48 49 50  ""
RPTPS1 ""  51 52 53 54 55 56 57 58 59 60  ""
RPTPS2 ""  1  2  3  4  5  6  7  8  9  10  ""
RPTPS2 ""  11 12 13 14 15 16 17 18 19 20  ""
RPTPS2 ""  21 22 23 24 25 26 27 28 29 30  ""
RPTPS2 ""  31 32 33 34 35 36 37 38 39 40  ""
RPTPS2 ""  41 42 43 44 45 46 47 48 49 50  ""
RPTPS2 ""  51 52 53 54 55 56 57 58 59 60  ""
AIR1   "" 1    41 ""
AIR2   "" 1    41   ""
OJTPD  "" 80""
OKTHTR ""  E""
OJVIS 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
OJDAY 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2
OJPOS 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
IBFOR ""  2001""
IRFOR ""  1000""
ICOMBO ""  1""
TVALON TRUE
@EOF



SSG    SGS STATEMENTS
```

Figure D-11. SSG Example 2
(page 1 of 10 pages)

SSG REVISED SKELETON

```
0   SKEL
0   *INCREMENT IE FROM [ENVFIRST,1,1,1] TO [ENVLAST,1,1,1]
1     *CREATE SGS: COMBAT [ENV,1,IE,1]
1     *CREATE SGS: PREFIX [USERID,1,1,1][FORCE,1,1,1]
1     *CREATE SGS: CXPS [ENV,1,IE,1]
1     *CREATE SGS: CSELT [FORCE,1,1,1][[SUPP,1,IE,1]
1     *CREATE SGS: BASELT [FORCE,1,1,1][[ENV,1,IE,1]
1     *CREATE SGS: PKSELT [PNAMES,1,IE,1]-[LIGHT,1,IE,1]/[FORCE1CH,1,1,1]
1     *CREATE SGS: RNGELT [PNAMES,1,IE,1]-DEEP/[FORCE1CH,1,1,1]
1     *CREATE SGS: PRFELT [PNAMES,1,IE,1]/[FORCE1CH,1,1,1]
1     *CREATE SGS: INVNTR H7AFPFILES INV[FORCE,1,1,1]
1     *CREATE SGS: NDIV1 [BDIV,1,IE,1]
1     *CREATE SGS: NDIV2 [RDIV,1,IE,1]
1     *CREATE SGS: DNSTY1 [GDNSTY1,1,IE,1]
1     *CREATE SGS: DNSTY2 [GDNSTY2,1,IE,1]
1     *BRKPT,K GO[FORCE,1,1,1][[COMBAT,1,1,1]N[NREPS,1,1,1]
1     *EDIT ON
1     #RUN,/TPR DO[RUNID,1,1,1][ALPHA,1,IE,1][NREPS,1,1,1],E1899P22770,&
1     SECPET,600,6300     . KNOX (703) 476-4923
1     #QUAL UNCLASSIFIED
1     #ASG,A H7AFPFILES/XXX/XXX.
1     #ASG,A H7RUNS/XXX/XXX.
1     *INCREMENT L TO [PFILES]
2       *INCREMENT M TO [PFILES,1]
3         #ASG,A [PFILES,L,M,1].
2       *LOOP
1     *LOOP
1     #HDG,S **** [FORCE,1,1,1] ENV [COMBAT,1,1,1] SEEDS [NREPS,1] ****
1     #PRT,S H7RUNS.GO[FORCE,1,1,1][[COMBAT,1,1,1]N[NREPS,1,1,1]
1     # . SAVE THE PROGRAMS
1     #ASG,T ABS-FILE.,///10000
1     #COPY,AC *H7AFP.870PKSGEN,ABS-FILE.870PKSGEN
1     #COPY,AC *H7AFP.870PROJGEN,ABS-FILE.870PROJGEN
1     #COPY,AC *H7AFP.870PREFGEN,ABS-FILE.870PREFGEN
1     #COPY,AC *H7AFP.870RNGDSTGEN,ABS-FILE.870RNGDSTGEN
1     #COPY,AC *H7AFP.870MAIN,ABS-FILE.870MAIN
1     #COPY,AC *98AFP.870GENKV,ABS-FILE.870GENKV
1     #COPY,AC *98AFP.870GENAL,ABS-FILE.870GENAL
1     #COPY,AC *98AFP.870PRTKV,ABS-FILE.870PRTKV
1     #COPY,AC *H7AFP.870CXPS,ABS-FILE.870CXPS
1     #COPY,AC *H7AFP.870REPORT,ABS-FILE.870REPORT
1     #COPY,AC *H7AFP.PAUSE,ABS-FILE.PAUSE
1     *INCREMENT L TO [PFILES]
2       *INCREMENT M TO [PFILES,1]
3         #FREE [PFILES,L,M,1].
2       *LOOP
1     *LOOP
1     *IF [COMBAT]
2       #ASG,T 20.,///4000
2       #ASG,T 21.,///4000
2       #ASG,T 22.,///4000
2       #ASG,T 23.,///4000
2       #ASG,T 24.,///5000
2       #ASG,T 25.,///4000
2       #ASG,T 26.,///4000
2       #ASG,T  3.,///2000
2       #ASG,T  4.,///2000
2       #ASG,T  7.,///2000
2       # .   UNIT 35 IS USED FOR TEMPORARY STORAGE
2       #ASG,T 35.,///2000
2       #ASG,T 34.,///25000
2       #ASG,T  9.,///500
2       #ASG,T  31.,///7000
2       #ASG,T  32.,///13000
2       #ASG,T  33.,///2000
2       *SET E TO 1
2       # .           I N P U T     F I L E S
2       # .
2       # .   LOCATE THE ELEMENTS FOR ENVIRONMENT DEPENDENT FILES
2       # .
2       #ASG,A [BASE,1,1,1].         . BASEDATA FILE
2       #ASG,T 15.
2       #ED [BASE,1,1,1].[BASELT,1,1,1],15.
2       #PRT,S [BASE,1,1,1].[BASELT,1,1,1]
2       #FREE [BASE,1,1,1].
2       #ASG,A [PKS,1,1,1]/XXX/XXX.       . PKS
2       #ASG,T 10.
2       #ED [PKS,1,1,1].[PKSELT,1,1,1],10.
2       #FREE [PKS,1,1,1].
2       #ASG,A [RNGDST,1,1,1].           . RANGE DISTRIBUTION
2       #ASG,T 11.
2       #ED [RNGDST,1,1,1].[RNGELT,1,1,1],11.
```

SSG REVISED SKELETON

Figure D-11. SSG Example 2
(page 2 of 10 pages)

SSG   REVISED SKELETON

```
  2        #FREE [RNGDST,1,1,1].
  2        #ASG,A [PREF,1,1,1].               . PREFERENCE FILE
  2        #ASG,T 12.
  2        #ED [PREF,1,1,1].[PRFELT,1,1,1],12.
  2        #FREE [PREF,1,1,1].
  2        #ASG,A [CSCSS,1,1,1].
  2        #ASG,T 14.
  2        #ED [CSCSS,1,1,1].[CSELT,1,1,1],14.
  2        #FREE [CSCSS,1,1,1]
  2        #ASG,A [ENGAGE,1,1,1].             . ENGAGEMENT FILE
  2        #ASG,T 13.
  2        #ED [ENGAGE,1,1,1].[ENGAGE,1,2,1],13.
  2        #FREE [ENGAGE,1,1,1].
  2        #ASG,A [PCAS,1,1,1].               . PERSONNEL CASUALTY FILE
  2        #ASG,T 16.
  2        #ED [PCAS,1,1,1].[PCAS,1,2,1],16.
  2        #FREE [PCAS,1,1,1].
  2        #ASG,A [USERID,1,1,1]KV[FORCE,1,1,1]/XXX/XXX.    . KV FILE
  2        #ERS 20.
  2        #ERS 21.
  2        #ERS 25.
  2        #ERS 26.
  2        #XQT ABS-FILE.870PREFGEN
  2         1 1 1 1 1 1
  2        #XQT ABS-FILE.870RNGDSTGEN
  2         1 1 1 1 1 1
  2        #XQT ABS-FILE.870PKSGEN
  2         1 1 1 1 1 1
  2        #XQT ABS-FILE.870PROJGEN
  2         1 1 1 1 1 1
  2        *INCREMENT IR TO [NREPS,1]
  3          *CREATE SGS: REPS [NREPS,1,IR,1]
  3          *SET R TO [REPS,1,1,1]
  3          #ERS 22.
  3          #ERS 23.
  3          #ERS 24.
  3          #ERS  3.
  3          #ERS  4.
  3          #ERS  9.
  3          #ERS 31.
  3          #ERS 32.
  3          #ERS 34.
  3          #XQT ABS-FILE.870MAIN
  3           1 1 1 1 1 1
  3          [*R]
  3          [DNSTY1,1,E,1]  [DNSTY2,1,E,1]
  3          [TYPLIM,1,1,1] [TYPLIM,1,2,1]   [TYPLIM,1,3,1] [TYPLIM,1,4,1]
  3          #ASG,T SORT33.,///1000
  3          #SORT,ES
  3          VOLUME=SMALL
  3          KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
  3          FILEIN=33.
  3          FILEOUT=SORT33.
  3          #EOF
  3          #USE 33.,SORT33.
  3          #ASG,T SORT9.,///1000
  3          #SORT,ES
  3          VOLUME=SMALL
  3          KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
  3          FILEIN=9.
  3          FILEOUT=SORT9.
  3          #EOF
  3          #USE 9.,SORT9.
  3          #ASG,T SORT24.,///1000
  3          #SORT,ES
  3          VOLUME=SMALL
  3          KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
  3          FILEIN=24.
  3          FILEOUT=SORT24.
  3          #EOF
  3          #USE 24.,SORT24.
  3          #ASG,T SORT32.,///1000
  3          #SORT,ES
  3          VOLUME=SMALL
  3          KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
  3          FILEIN=32.
  3          FILEOUT=SORT32.
  3          #EOF
  3          #USE 32.,SORT32.
  3          # .  SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
  3          #ASG,A [PREFIX,1,1,1]/XXX/XXX.
  3          # . COPY,I 3.,[PREFIX,1,1,1].A[COMBAT,1,E,1]R[*R]
```

SSG   REVISED SKELETON

Figure D-11. SSG Example 2
(page 3 of 10 pages)

```
SSG    REVISED SKELETON

3          # . COPY,T 4.,[PREFIX,1,1,1].F[COMBAT,1,E,1]R[*R]
3          # . COPY,T 9.,[PREFIX,1,1,1].T[COMBAT,1,E,1]R[*R]
3          *IF [QAREP]
4            #DELETE,C [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.
4            #XQT ABS-FILE.PAUSE
4            #CAT,P [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.,F///20000
4            #ASG,A [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]/XXX/XXX.
4            #KEEP [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R].
4            #BRKPT PRINT$/[PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R]
4            #XQT ABS-FILE.870REPORT
4            [QAPKS,1,1,1]
4             1 1 1 1 1 1
4            *INCREMENT K TO [RPTPS1]
5              *INCREMENT L TO [RPTPS1,K]
6                [RPTPS1,K,L,1]
5              *LOOP
4            *LOOP
4            9999
4            *INCREMENT K TO [RPTPS2]
5              *INCREMENT L TO [RPTPS2,K]
6                [RPTPS2,K,L,1]
5              *LOOP
4            *LOOP
4            9999
4            #EOF
4            #BRKPT PRINT$
4            # . SYM,U [PREFIX,1,1,1]R[COMBAT,1,E,1]R[*R].
3          *END
3          #DELETE,C [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.
3          #XQT ABS-FILE.PAUSE
3          #CAT,P [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.,F///20000
3          #ASG,A [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]/XXX/XXX.
3          #KEEP [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R].
3          #BRKPT PRINT$/[PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R]
3          #XQT ABS-FILE.870GENKV
3          #XQT ABS-FILE.870PRTKV
3          KV SCOREBOARD FOR [FORCE,1,1,1]  ENV [COMBAT,1,E,1]   SEED [*R]
3          [SUPERTRP,1,1,1]
3          [SUPERB,1,1,1]
3          [SUPERR,1,1,1]
3          #ADD,E [INVNTR,1,1,1].[INVNTR,1,2,1]
3          #ADD,E 7.
3          #ED 7.,[USERID,1,1,1]KV[FORCE,1,1,1].E[COMBAT,1,E,1]R[*R]
3          # . COPY KV FILE FOR KV ROLLUP
3          #ASG,T 7.     . FOLLOWING PROGRAM WRITES TO UNIT 7
3          #XQT ABS-FILE.870GENAL
3          # . PRINT THE ALLOCATION REPORT
3          #XQT ABS-FILE.870PRTKV
3          ALLOCATION REPORT FOR [FORCE,1,1,1] ENV [COMBAT,1,E,1] SEED [*R]
3          [SUPERTRP,1,1,1]
3          [SUPERB,1,1,1]
3          [SUPERR,1,1,1]
3          #ADD,E [INVNTR,1,1,1].[INVNTR,1,2,1]
3          #ADD,E 7.
3          *IF [CXPS]
4            *IF NOT [COMBAT]
5              #ASG,T 7.,///2000
5              #ASG,A [BASE,1,1,1].        . BASEDATA FILE
5              #ASG,T 15.
5              #ED [BASE,1,1,1].[BASELT,1,1,1],15.
5              #FREE [BASE,1,1,1].
5              #ASG,A [CSCSS,1,1,1].
5              #ASG,T 14.
5              #ED [CSCSS,1,1,1].[CSELT,1,1,1],14.
5              #FREE [CSCSS,1,1,1]
5              #ASG,A [PCAS,1,1,1].             . PERSONNEL CASUALTY FILE
5              #ASG,T 16.
5              #ED [PCAS,1,1,1].[PCAS,1,2,1],16.
5              #FREE [PCAS,1,1,1].
5              # .  END OF NOT COMBAT IF
4            *END
4            #HDG    ** [FORCE,1,1,1] CXPS ENV [CXPS,1,E,1]  SEED [*R] **
4            # .  CREATE OUTPUT FILES FOR CXPS
4            #ASG,T 29.
4            #ASG,T 30.
4            # .  RUN THE MERGE MODULE
4            # .  RUN WITH FILES DEVELOPED EARLIER
4            # . COPY,I [PREFIX,1,1,1].A[CXPS,1,E,1]R[*R],3.
4            # . COPY,I [PREFIX,1,1,1].F[CXPS,1,E,1]R[*R],4.
4            # . COPY,I [PREFIX,1,1,1].T[CXPS,1,E,1]R[*R],9.
4            #ERS 7.
4            #XQT ABS-FILE.870CXPS

SSG    REVISED SKELETON
```

Figure D-11. SSG Example 2
(page 4 of 10 pages)

```
SSG   REVISED SKELETON

4              *. IMAGE GIVING INDICES IN CBT POT OUTPUT RECORDS
4              *EDIT ON
4              [ICOMBO,1,1,1]&
4              [OJTPD,1,1,1]&
4              [OKTHTR,1,1,1]&
4                [OJVIS,1,IE,1]&
4                [OJDAY,1,IE,1]&
4                [OJPOS,1,IE,1]&
4              [TBFOR,1,1,1]&
4              [IRFOR,1,1,1]
4              *. IMAGF GIVING INDICES OF INPUT CS/CSS MODULI
4              [TVALON,1,1,1]
4              1  1  1  1  1  1
4              [NDIV1,1,E,1]  [NDIV2,1,E,1]
4              [AIR1,1,1,1]
4              [AIR2,1,1,1]
4              #ADD [CVALS,1,1,1].[CVALS,1,2,1]
4              #ADD [FRACTS,1,1,1].[FRACTS,1,2,1]
4              # .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
4              #DATA,L 29.
4              #END
4              #COPY,I 29.,[PREFIX,1,1,1].E[CXPS,1,E,1]R[*R]
4              #ASG,T SORTOUT.
4              #SORT,S
4              VOLUME=SMALL
4              KEY=1,5,CH,A:67,10,CH,A
4              FILEIN=29.
4              FILEOUT=SORTOUT.
4              #ED,R SORTOUT.
4              LNP!
4              EXI
3            *END
3            #BRKPT PRINT$
3            #SYM,U [PREFIX,1,1,1]D[COMBAT,1,E,1]R[*R].,[KVCOPIES,1,1,1]
3            *REMOVE SGS REPS
2          *LOOP     .IP
2          *.  END OF CXPS IF
2          *.  END OF ENVIRONMENT IF
1        *END
1        #FIN
1        *REMOVE SGS COMBAT
1        *REMOVE SGS PREFIX
1        *REMOVE SGS CXPS
1        *REMOVE SGS CSELT
1        *REMOVE SGS BASELT
1        *REMOVE SGS PKSELT
1        *REMOVE SGS RNGELT
1        *REMOVE SGS PRFELT
1        *REMOVE SGS INVNTR
1        *REMOVE SGS NDIV1
1        *REMOVE SGS NDIV2
1        *REMOVE SGS DNSTY1
1        *REMOVE SGS DNSTY2
0      *LOOP     .IE
0      @EOF

SSG   STREAM GENERATION RUN LOG PART 1

COMBAT 04
PREFIX H7HM8C
CXPS 04
CSELT HM80E04
BASELT HM80E04
PKSELT BAPD-D/H
RNGELT BAPD-DEEP/H
PRFELT BAPD/H
INVNTR H7AFPFILES INVHM80
NDIV1 3
NDIV2 1
DNSTY1 F
DNSTY2 T

SSG   GENERATED OUTPUT STREAM   PART 1

     1.        1     SKEL

SSG   STREAM GENERATION RUN LOG PART 1

REPS 1
REPS 2
REPS 3
```

Figure D-11. SSG Example 2
(page 5 of 10 pages)

SSG GENERATED OUTPUT STREAM PART)

```
          @RUN,/TPR 003001,F1899P2277D,SECRET,600,6300    . KNOX (703) 476-4923
          @QUAL UNCLASSIFIED
          @ASG,A H7AFPFILES/XXX/XXX.
          @ASG,A H7RUNS/XXX/XXX.
          @ASG,A *H7AFP.
          @ASG,A *98AFP.
          @HDG,S **** HM80 ENV C4 SEEDS 3 ****
          @PRT,S H7RUNS.GOHM80E04N1
          @ .  SAVE THE PROGRAMS
          @ASG,T ABS-FILE.,///1000U
          @COPY,AC *H7AFP.870PKSGEN,ABS-FILE.870PKSGEN
          @COPY,AC *H7AFP.870PROJGEN,ABS-FILE.870PROJGEN
          @COPY,AC *H7AFP.870PREFGEN,ABS-FILE.870PREFGEN
          @COPY,AC *H7AFP.870RNGDSTGEN,ABS-FILE.870RNGDSTGEN
          @COPY,AC *H7AFP.870MAIN,ABS-FILE.870MAIN
          @COPY,AC *98AFP.870GENKV,ABS-FILE.870GENKV
          @COPY,AC *98AFP.870GENAL,ABS-FILE.870GENAL
          @COPY,AC *98AFP.870PRTKV,ABS-FILE.870PRTKV
          @COPY,AC *H7AFP.870CXPS,ABS-FILE.870CXPS
          @COPY,AC *H7AFP.870REPORT,ABS-FILE.870REPORT
          @COPY,AC *H7AFP.PAUSE,ABS-FILE.PAUSE
          @FREE *H7AFP.
          @FREE *98AFP.
          @ASG,T 20.,///4000
          @ASG,T 21.,///4000
          @ASG,T 22.,///4000
          @ASG,T 23.,///4000
          @ASG,T 24.,///5000
          @ASG,T 25.,///4000
          @ASG,T 26.,///4000
          @ASG,T  3.,///2000
          @ASG,T  4.,///2000
          @ASG,T  7.,///2000
          @ .   UNIT 35 IS USED FOR TEMPORARY STORAGE
          @ASG,T 35.,///2000
          @ASG,T 34.,///25000
          @ASG,T  9.,///500
          @ASG,T  31.,///7000
          @ASG,T  32.,///13000
          @ASG,T  33.,///2000
          @ .            I N P U T    F I L E S
          @ .
          @ .  LOCATE THE ELEMENTS FOR ENVIRONMENT DEPENDENT FILES
          @ .
          @ASG,A *H7BASEDATA.        . BASEDATA FILE
          @ASG,T 15.
          @ED *H7BASEDATA.HM80E04,15.
          @PRT,S *H7BASEDATA.HM80E04
          @FREE *H7BASEDATA.
          @ASG,A H7PKS/XXX/XXX.       . PKS
          @ASG,T 10.
          @ED H7PKS.BAPD-D/H,10.
          @FREE H7PKS.
          @ASG,A *H7RNGDST.          . RANGE DISTRIBUTION
          @ASG,T 11.
          @ED *H7RNGDST.BAPD-DEEP/H,11.
          @FREE *H7RNGDST.
          @ASG,A *H7PREF.            . PREFERENCE FILE
          @ASG,T 12.
          @ED *H7PREF.BAPD/H,12.
          @FREE *H7PREF.
          @ASG,A *H7CSCSS.
          @ASG,T 14.
          @ED *H7CSCSS.HM80E04,14.
          @FREE *H7CSCSS
          @ASG,A *H7ENGAGE.          . ENGAGEMENT FILE
          @ASG,T 13.
          @ED *H7ENGAGE.RAPD,13.
          @FREE *H7ENGAGE.
          @ASG,A *H7PCAS.            . PERSONNEL CASUALTY FILE
          @ASG,T 16.
          @ED *H7PCAS.H,16.
          @FREE *H7PCAS.
          @ASG,A H7KVHM80/XXX/XXX.    . KV FILE
          @ERS 20.
          @ERS 21.
          @ERS 25.
          @ERS 26.
          @XQT ABS-FILE.870PREFGEN
1           1 1 1 1 1 1
          @XQT ABS-FILE.870RNGDSTGEN
1           1 1 1 1 1 1
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-11. SSG Example 2
(page 6 of 10 pages)

```
SSG   GENERATED OUTPUT STREAM   PART  )
                @XQT ABS-FILE.87OPKSGEN
       1         1 1 1 1 1 1
                @XQT ABS-FILE.87OPROJGEN
       1         1 1 1 1 1 1
                @ERS 22.
                @ERS 23.
                @ERS 24.
                @ERS  3.
                @ERS  4.
                @ERS  9.
                @ERS 31.
                @ERS 32.
                @ERS 34.
                @XQT ABS-FILE.87OMAIN
       1         1 1 1 1 1 1
       2        1
       3        F  T
       4        1 60   1 60
                @ASG,T SORT33.,///1000
                @SORT,ES
       1        VOLUME=SMALL
       2        KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
       3        FILEIN=33.
       4        FILEOUT=SORT33.
                @EOF
                @USE 33.,SORT33.
                @ASG,T SORT9.,///1000
                @SORT,ES
       1        VOLUME=SMALL
       2        KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
       3        FILEIN=9.
       4        FILEOUT=SORT9.
                @EOF
                @USE 9.,SORT9.
                @ASG,T SORT24.,///1000
                @SORT,ES
       1        VOLUME=SMALL
       2        KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
       3        FILEIN=24.
       4        FILEOUT=SORT24.
                @EOF
                @USE 24.,SORT24.
                @ASG,T SORT32.,///1000
                @SORT,ES
       1        VOLUME=SMALL
       2        KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
       3        FILEIN=32.
       4        FILEOUT=SORT32.
                @EOF
                @USE 32.,SORT32.
                @ . SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
                @ASG,A H7HM80/XXX/XXX.
                @ . COPY,I 3.,H7HM80.AO4R1
                @ . COPY,I 4.,H7HM80.FO4R1
                @ . COPY,I 9.,H7HM80.TO4R1
                @DELETE,C H7HM80DO4R1/XXX/XXX.
                @XQT ABS-FILE.PAUSE
                @CAT,P H7HM80DO4R1/XXX/XXX.,F///20000
                @ASG,A H7HM80DO4R1/XXX/XXX.
                @KEEP H7HM80DO4R1.
                @BRKPT PRINT$/H7HM80DO4P1
                @XQT ABS-FILE.87OGENKV
                @XQT ABS-FILE.87OPRTKV
       1        KV SCOREBOARD FOR HM80  ENV 04   SEED 1
       2        10 8 5
       3        1 2 3 4 5 6 42 44
       4        1 2 3 42 44
                @ADD,E H7AFPFILES.INVHM80
                @ADD,E 7.
                @ED 7.,H7KVHM80.EO4R1
                @ . COPY KV FILE FOR KV ROLLUP
                @ASG,T 7.     . FOLLOWING PROGRAM WRITES TO UNIT 7
                @XQT ABS-FILE.87OGENAL
                @ . PRINT THE ALLOCATION REPORT
                @XQT ABS-FILE.87OPRTKV
       1        ALLOCATION REPORT FOR HM80 ENV 04 SEED 1
       2        10 8 5
       3        1 2 3 4 5 6 42 44
       4        1 2 3 42 44
                @ADD,E H7AFPFILES.INVHM80
                @ADD,E 7.
                @HDG    ** HM80 CXPS ENV 04   SEED 1 **
SSG   GENERATED OUTPUT STREAM   PART  )
```

Figure D-11. SSG Example 2
(page 7 of 10 pages)

SSG GENERATED OUTPUT STREAM PART)

```
@ .  CREATE OUTPUT FILES FOR CXPS
@ASG,T 29.
@ASG,T 30.
@ .  RUN THE MERGE MODULE
@ .  RUN WITH FILES DEVELOPED EARLIER
@ . COPY,I H7HM80.A04R1,3.
@ . COPY,I H7HM80.F04R1,4.
@ . COPY,I H7HM80.T04R1,9.
@ERS 7.
@XQT ABS-FILE.870CXPS
  1 80  E  1  1  4  2001  1000
TRUE
1  1  1  1  1  1
3  1
 1   41
 1   41
@ADD *H7GLOBAL.CVALS
@ADD *H7GLOBAL.FRACTS
@ .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
@DATA,L 29.
@END
@COPY,I 29.,H7HM80.E04R1
@ASG,T SORTOUT.
@SORT,S
VOLUME=SMALL
KEY=1,5,CH,A:67,10,CH,A
FILEIN=29.
FILEOUT=SORTOUT.
@ED,R SORTOUT.
LNP!
EXI
@BRKPT PRINTS
@SYM,U H7HM80D04R1.,1
@ERS 22.
@ERS 23.
@ERS 24.
@ERS  3.
@ERS  4.
@ERS  9.
@ERS 31.
@ERS 32.
@ERS 34.
@XQT ABS-FILE.870MAIN
 1 1 1 1 1 1
2
F  T
1 60   1 60
@ASG,T SORT33.,///1000
@SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=33.
FILEOUT=SORT33.
@EOF
@USE 33.,SORT33.
@ASG,T SORT9.,///1000
@SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=9.
FILEOUT=SORT9.
@EOF
@USE 9.,SORT9.
@ASG,T SORT24.,///1000
@SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=24.
FILEOUT=SORT24.
@EOF
@USE 24.,SORT24.
@ASG,T SORT32.,///1000
@SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=32.
FILEOUT=SORT32.
@EOF
@USE 32.,SORT32.
@ .  SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
@ASG,A H7HM80/XXX/XXX.
@ . COPY,I 3.,H7HM80.A04R2
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-11. SSG Example 2
(page 8 of 10 pages)

SSG GENERATED OUTPUT STREAM PART)

```
            @ . COPY,I 4.,H7HM80.FC4R2
            @ . COPY,I 9.,H7HM80.TO4R2
            @DELETE,C H7HM80DO4R2/XXX/XXX.
            @XQT ABS-FILE.PAUSE
            @CAT,P H7HM80DO4R2/XXX/XXX.,F///20000
            @ASG,A H7HM80DO4R2/XXX/XXX.
            @KEEP H7HM80DO4R2.
            @BRKPT PRINT$/H7HM80DO4R2
            @XQT ABS-FILE.870GENKV
            @XQT ABS-FILE.870PRTKV
        1   KV SCOREBOARD FOR HM80  ENV 04   SEED 2
        2   10 8 5
        3   1 2 3 4 5 6 42 44
        4   1 2 3 42 44
            @ADD,E H7AFPFILES.INVHM80
            @ADD,E 7.
            @ED 7.,H7KVHM80.E04R2
            @ . COPY KV FILE FOR KV ROLLUP
            @ASG,T 7.      . FOLLOWING PROGRAM WRITES TO UNIT 7
            @XQT ABS-FILE.870GENAL
            @ . PRINT THE ALLOCATION REPORT
            @XQT ABS-FILE.870PRTKV
        1   ALLOCATION REPORT FOR HM80 ENV 04 SEED 2
        2   10 8 5
        3   1 2 3 4 5 6 42 44
        4   1 2 3 42 44
            @ADD,E H7AFPFILES.INVHM80
            @ADD,E 7.
            @HDG    ** HM80 CXPS ENV 04  SEED 2 **
            @ .  CREATE OUTPUT FILES FOR CXPS
            @ASG,T 29.
            @ASG,T 30.
            @ .  RUN THE MERGE MODULE
            @ .  RUN WITH FILES DEVELOPED EARLIER
            @ . COPY,I H7HM80.A04R2,3.
            @ . COPY,I H7HM80.F04R2,4.
            @ . COPY,I H7HM80.T04R2,9.
            @ERS 7.
            @XQT ABS-FILE.870CXPS
        1     1 80  E   1  1  4   2001  1000
        2   TRUE
        3   1  1   1   1   1   1
        4   3  1
        5    1    41
        6    1    41
            @ADD *H7GLOBAL.CVALS
            @ADD *H7GLOBAL.FRACTS
            @ .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
            @DATA,L 29.
            @END
            @COPY,I 29.,H7HM80.EC4R2
            @ASG,T SORTOUT.
            @SORT,S
        1   VOLUME=SMALL
        2   KEY=1,5,CH,A:67,10,CH,A
        3   FILEIN=29.
        4   FILEOUT=SORTOUT.
            @ED,R SORTOUT.
        1   LNP!
        2   EXI
            @BRKPT PRINTS
            @SYM,U H7HM80DO4R2.,1
            @ERS 22.
            @ERS 23.
            @ERS 24.
            @ERS  3.
            @ERS  4.
            @ERS  9.
            @ERS 31.
            @ERS 32.
            @ERS 34.
            @XQT ABS-FILE.870MAIN
        1    1 1 1 1 1 1
        2   3
        3   F   T
        4   1 60    1 60
            @ASG,T SORT33.,///1000
            @SORT,ES
        1   VOLUME=SMALL
        2   KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
        3   FILEIN=33.
        4   FILEOUT=SORT33.
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-11. SSG Example 2
(page 9 of 10 pages)

SSG GENERATED OUTPUT STREAM PART)

```
          @EOF
          @USE 33.,SORT33.
          @ASG,T SORT9.,///1000
          @SORT,ES
    1     VOLUME=SMALL
    2     KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
    3     FILEIN=9.
    4     FILEOUT=SORT9.
          @EOF
          @USE 9.,SORT9.
          @ASG,T SORT24.,///1000
          @SORT,ES
    1     VOLUME=SMALL
    2     KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
    3     FILEIN=24.
    4     FILEOUT=SORT24.
          @EOF
          @USE 24.,SORT24.
          @ASG,T SORT32.,///1000
          @SORT,ES
    1     VOLUME=SMALL
    2     KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
    3     FILEIN=32.
    4     FILEOUT=SORT32.
          @EOF
          @USE 32.,SORT32.
          @ .  SAVE DIRECT ACCESS FILES ALLOC, FALLOC, AND TTLOS IN ELEMENTS
          @ASG,A H7HM8C/XXX/XXX.
          @ . COPY,I 3.,H7HM8C.A04R3
          @ . COPY,I 4.,H7HM8C.FC4R3
          @ . COPY,I 9.,H7HM80.T04R3
          @DELETE,C H7HM8CD04R3/XXX/XXX.
          @XQT ABS-FILE.PAUSE
          @CAT,P H7HM8CD04R3/XXX/XXX.,F///20000
          @ASG,A H7HM8CD04R3/XXX/XXX.
          @KEEP H7HM8CD04R3.
          @PRKPT PRINT$/H7HM8CD04P3
          @XQT ABS-FILE.870GENKV
          @XQT ABS-FILE.870PRTKV
    1     KV SCOREBOARD FOR HM8C  ENV C4   SEED 3
    2     10 8 5
    3     1 2 3 4 5 6 42 44
    4     1 2 3 42 44
          @ADD,E H7AFPFILES.INVHM80
          @ADD,E 7.
          @ED 7.,H7KVHM80.E04R3
          @ . COPY KV FILE FOR KV ROLLUP
          @ASG,T 7.     . FOLLOWING PROGRAM WRITES TO UNIT 7
          @XQT ABS-FILE.870GENAL
          @ . PRINT THE ALLOCATION REPORT
          @XQT ABS-FILE.870PRTKV
    1     ALLOCATION REPORT FOR HM8C ENV 04 SEED 3
    2     10 8 5
    3     1 2 3 4 5 6 42 44
    4     1 2 3 42 44
          @ADD,E H7AFPFILES.INVHM80
          @ADD,E 7.
          @HDG    ** HM80 CXPS ENV 04  SEED 3 **
          @ .  CREATE OUTPUT FILES FOR CXPS
          @ASG,T 29.
          @ASG,T 30.
          @ .  RUN THE MERGE MODULE
          @ .  RUN WITH FILES DEVELOPED EARLIER
          @ . COPY,I H7HM80.A04R3,3.
          @ . COPY,I H7HM8C.FC4R3,4.
          @ . COPY,I H7HM80.T04R3,9.
          @ERS 7.
          @XQT ABS-FILE.870CXPS
    1       1 80  E  1  1  4  2001  1000
    2     TRUE
    3     1  1  1  1  1  1
    4     3  1
    5      1   41
    6      1   41
          @ADD *H7GLOBAL.CVALS
          @ADD *H7GLOBAL.FRACTS
          @ .  COPY CXPS OUTPUT FOR INPUTTING TO ROLLUP
          @DATA,L 29.
          @END
          @COPY,I 29.,H7HM80.E04R3
          @ASG,T SORTOUT.
          @SORT,S
    1     VOLUME=SMALL
    2     KEY=1,5,CH,A:67,10,CH,A
    3     FILEIN=29.
    4     FILEOUT=SORTOUT.
          @ED,R SORTOUT.
    1     LNP!
    2     EXI
          @PRKPT PRINT$
          @SYM,U H7HM8CD04R3.,1
          @FIN
          @EOF
```

SSG GENERATED OUTPUT STREAM PART)

Figure D-11. SSG Example 2
(page 10 of 10 pages)

D-5. MAIN PROGRAM OF FIREPOWER AND COUNTERFIREPOWER MODULE. This paragraph presents the source programs and general logic of the main program of the AFP Combat Module. It highlights relations among variables and arrays, subroutines, input and output files, and some preprocessors. Considerable referencing to other paragraphs and this appendix's annexes may be required. Most routines contain enough comments for someone to track data and program flow. On the other hand, it is not easy to follow all interactions within the Combat Module. Subroutine arguments may be defined in calling or called routines.

a. General

(1) Figure D-12 provides a schematic diagram of the relation among many input and output files and corresponding routines (MAIN, CNFAOC, INPUT, FRCALC, DIRIO, and OUTPT) of the Main Program of the AFP Combat Module. Figure D-12 implies the relation of many of the following paragraphs of this section to other sections (e.g., input and output) of this appendix and to the appendix's annexes.

(2) The principal purpose of the Main Program of the AFP Combat Module is to provide estimates of the kills and losses by weapon type for two opposing sides in a specific combat environment. In brief, the purpose is to generate K/V scoreboards. Kills and losses are subject to probabilities. Although many replications and different environments can, in principle, be examined in a single execution of the Combat Module, the preferred, standard approach is to execute the Main Program separately for each combination of random number seed and combat environment. The purpose of AFP in total extends beyond the generation of K/V scoreboards to the transformation of many replications of K/V scoreboards across many combat environments into measures of combat potential of equipment and units.

(3) The Main Program of the Combat Module may be divided logically into two parts.

(a) The first part processes surviving inventories, type-on-type preference and participation factors, and prescribed range distributions in order to generate the specific matchups of opposing weapon types. The first part is described in total as the "allocation processor."

(b) The second part is primarily devoted to detection, shooting, and kill/survivor assessment among the previously allocated weapons. The second part does perform some additional allocation of area fire weapons. However, the second part is described in total simply as the "attrition processor."

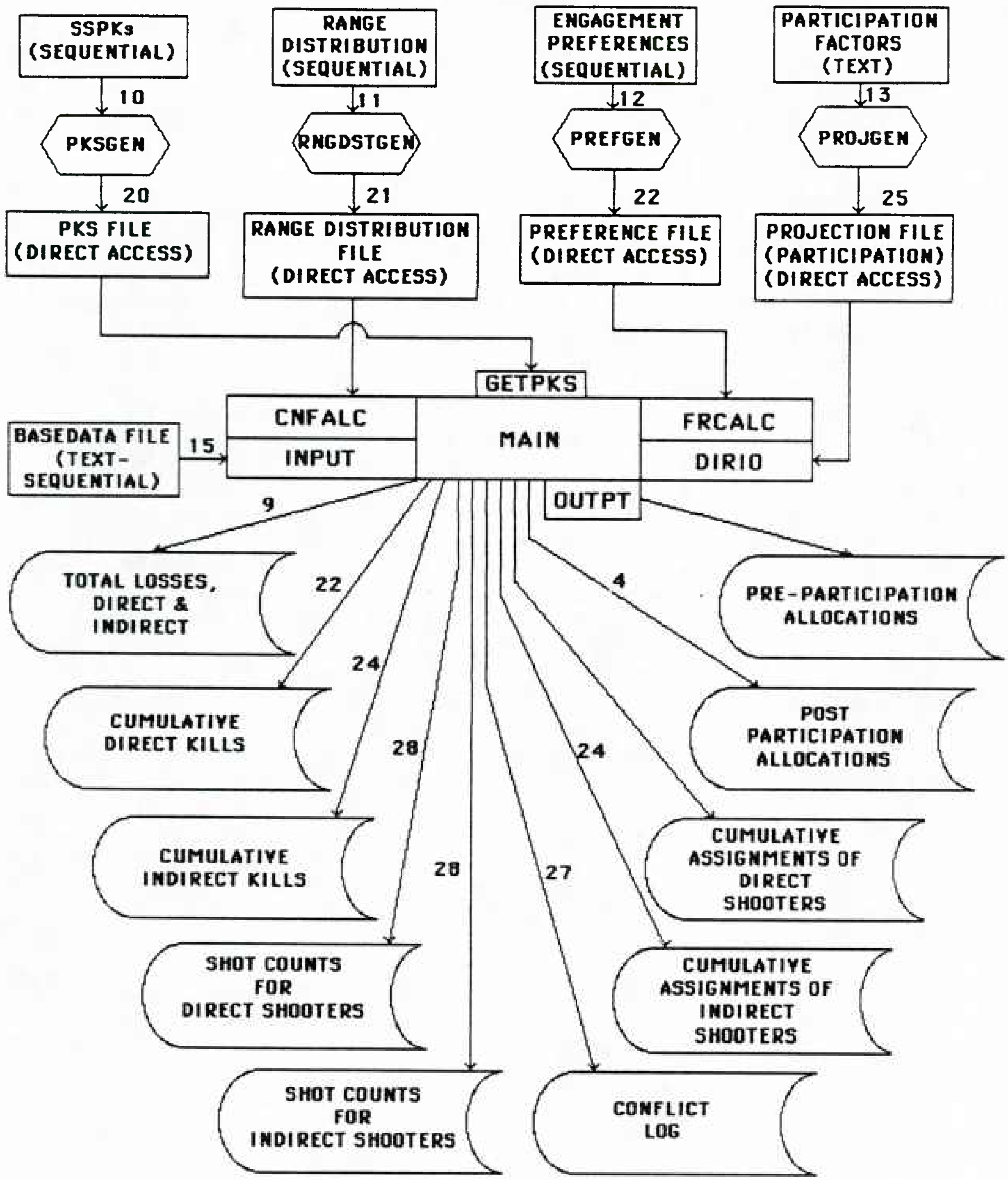


Figure D-12. Schematic Representation of Relation among AFP Input/Output Files and the Combat Module Routines that Create or Use Files

(c) Several subroutines of the Main Program of the Combat Module may be clearly assigned to one of the two principal processes--allocation or attrition. These assignments are shown in Figure D-13. Figure D-13 also briefly identifies the tasks performed by the included subroutines. These subroutines, along with all others, are described in greater detail in paragraph D-5c. The allocation and attrition processes and their individual subroutines are embedded in a somewhat complex looping structure addressed later. Figure D-13 suggests a very general flow in the sense of what subprocesses must precede others, but does not convey anything of the module's real looping/nesting structure.

(d) Figure D-14 outlines the nesting structure of the main program of the Combat Module. Each block in Figure D-14 represents a logical loop. For example, the block referred to as "REPS" represents the loop over replications. The next block, called "DAYS," is a loop over the days. The fact that the DAYS block is contained within the REPS block indicates that the DAYS loop is nested within the REPS loop. As noted above, standard AFP practice is to limit program execution to a single replication and single environment within a single run of the module. Hence, the REPS and ENVIRONMENT loop are degenerate.

(e) Figure D-15 is an expansion of the preceding Figure D-14. Only the top and bottom lines of each block from Figure D-14 have been retained. Each block in Figure D-15 identifies the tasks performed within that block. Note that specific subroutines are shown in relation to their corresponding nested blocks (and loops). Looping is controlled by the MAIN subprogram of the module.

b. FORTRAN PROCS, Common Blocks, Arrays, Variables, Parameters, and Files

(1) Figure D-16 provides a listing of the symbolic element PROCS. FORTRAN common blocks, used throughout the program, are identified. The routines in which the blocks are referenced are shown. The common blocks defined are FORTRAN procs in accord with the UNIVAC PDP processor's FORTRAN conventions. Many of the arrays defined in the common blocks have dimensions defined as FORTRAN parameters. The parameter values applied in the current version of the Combat Module are set as shown beginning at line 188 in the FORTRAN proc named PARM. Some of the parameters are named mnemonically. Many of the Combat Module's file usages are identified beginning at line 244 in Figure D-16.

(2) Table D-2 provides brief notes about many of the common blocks, arrays, and variables listed in Figure D-16.

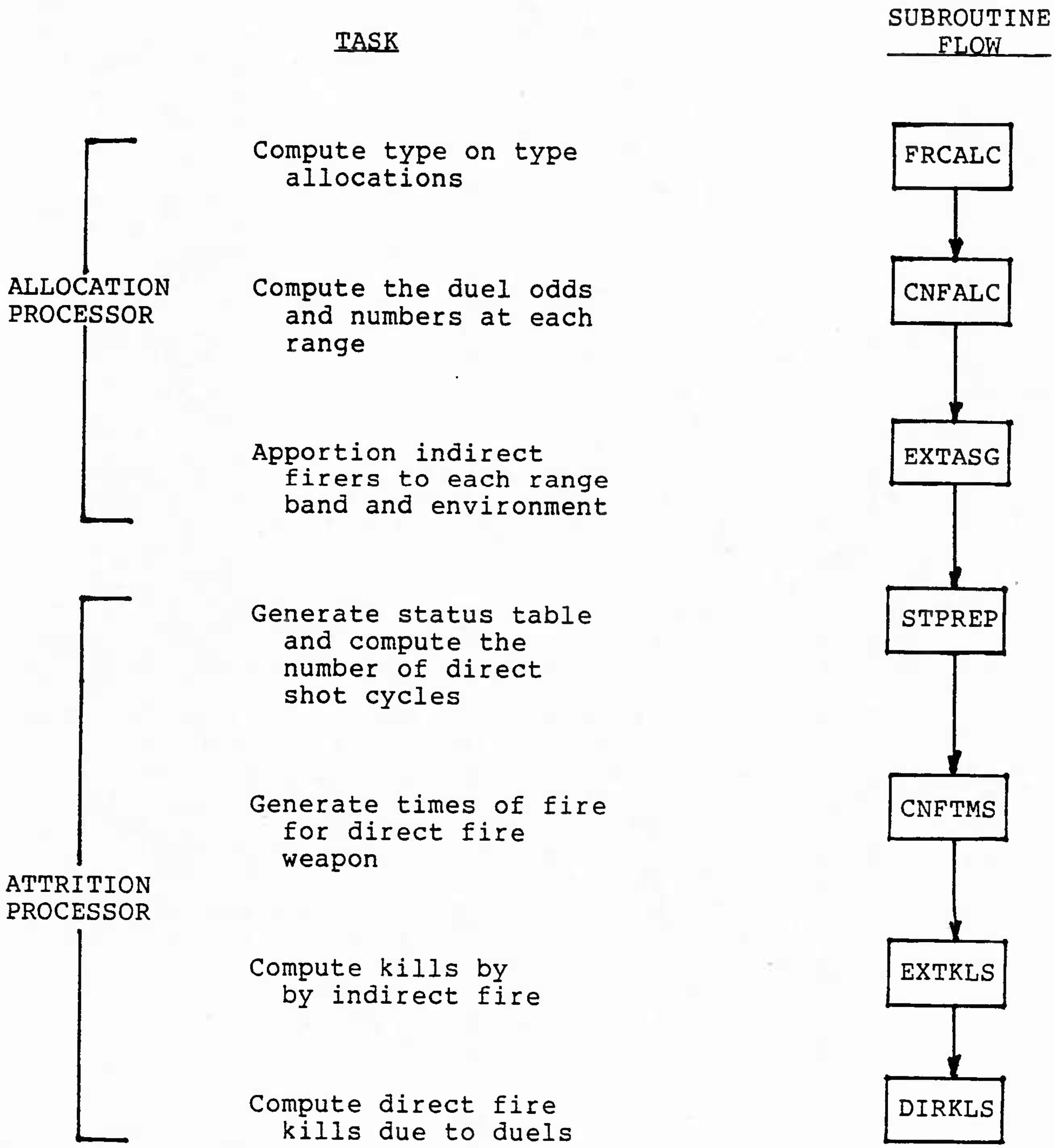


Figure D-13. Principal Subroutines of the Weapon Allocation and Attrition Processes within the Main Program of the AFP Combat Module

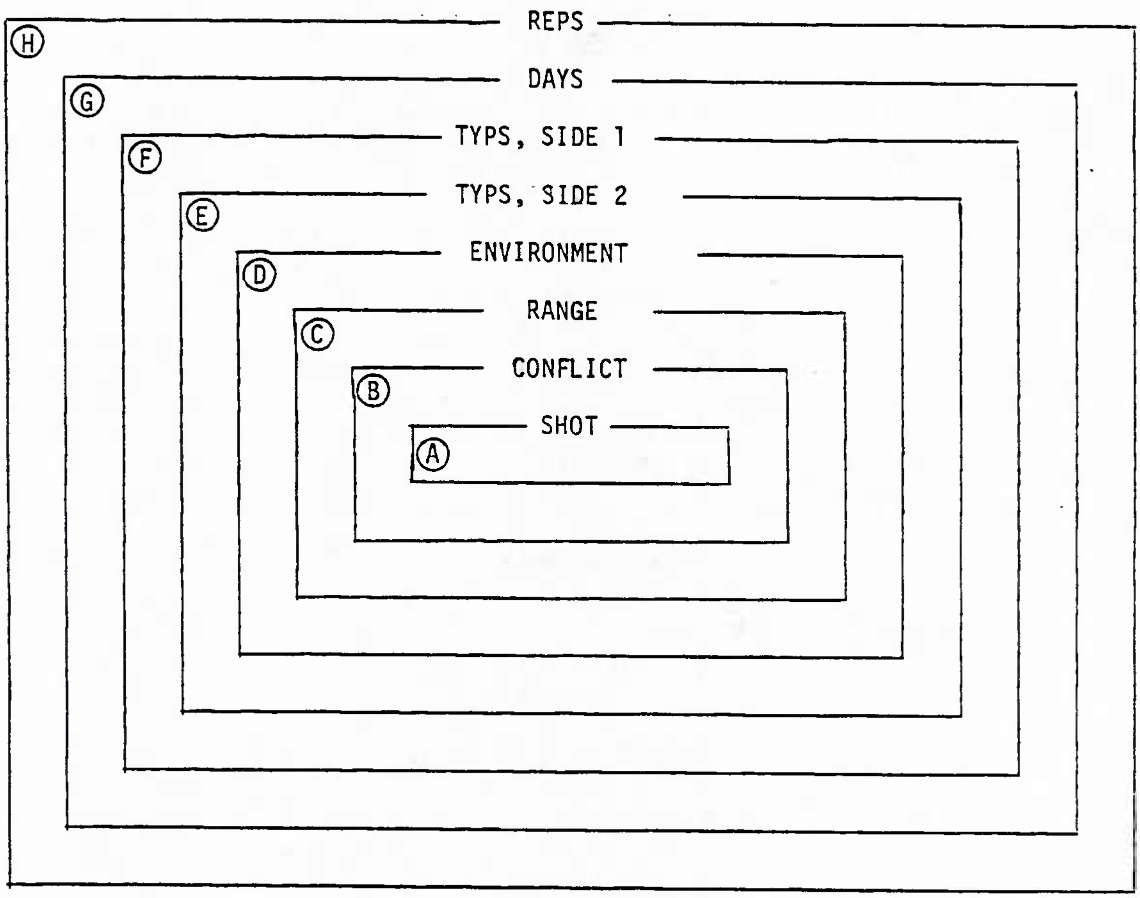


Figure D-14. Logical Nesting Structure of the Main Program of the AFP Combat Module

REP

DAYS

1. Call FRCALC - Compute the overall initial allocations of each pair of opposing types to each other.

2. Call OUTPT - Write out these allocations.

TYPES, SIDE 1

1. Call DIRIO - Read in the cumulative kills to date, the number of instances direct fire weapons participated in duels, and the appropriate entries in the projections file.

2. Write out the post participation allocations.

TYPES, SIDE 2

1. Call CNFALC - Compute the number of opposing types at each range as well as the number of duels at each range for each odds.

2. Call EXTASG - Compute the number of indirect weapons allocated against each of the two opposing types at each range band.

ENVIRONMENTS

Read in the appropriate pks.

RANGES

CONFLICTS

1. Call CNFLC1 - Compute the number of opposing types and the number of duels for the given conflict at the given range and environment.

2. Call STPREP - Set up a table with one entry for each duel in the conflict, for the given types, range and environment.

Figure D-15. Expanded View of the Logical Nesting (and Looping) Structure of the Main Program of the AFP Combat Module (page 1 of 2 pages)

SHOTS

Call CNFTMS - Generate time to fire for each weapon requiring one. Initially, all weapons require a time. Thereafter, a time must be generated only for surviving weapons that have fired in the last round. The time is generated either by time to detect logic or by time to refire.

Call EXTKLS - Compute kills due to interference in the duels by indirect fire.

Call DIRKLS - Compute kills resulting from the duels.

END OF SHOT BLOCK

END OF CONFLICTS BLOCK

END OF RANGE BLOCK

END OF ENVIRONMENT BLOCK

Write out the external shot counts

END OF TYPES, SIDE 2 BLOCK

Write out the cumulative kills

Write out the number of duel participations

Write out the type by type loss table

END OF TYPES, SIDE 1 BLOCK

Write out the indirect kills

Write out the number of indirect firer involvements in duels

Write out the shot data for duels and the ammo type table

Write out the maximum number of duels and the maximum odds

Figure D-15. Expanded View of the Logical Nesting (and Looping) Structure of the Main Program of the AFP Combat Module (page 2 of 2 pages)

```
C   CODES USED IN NAMING THE COMMON BLOCKS
C       I                   INPUT
C       M                   MAIN
C       1                   FRCALC
C       2                   DIRIO
C       3                   CNFALC
C       4                   STPREP
C       5                   CNFTMS
C       6                   EXTKLS
C       7                   DIRKLS
C       8                   CNFLC1
BK1     PROC
C  USED IN INPUT, MAIN, FRCALC, DIRIO, CNFALC, STPREP, EXTKLS, DIRKLS
C
        COMMON/C1TO57 /NTYPS(2),CASE,PREF1,PREF2,PREF3,PREF4,INSEED,DSEED
        INTEGER CASE, PREF1, PREF2, PREF3, PREF4
        DOUBLE PRECISION DSEED
  END
BK2     PROC
C   USED IN MAIN, CNFALC, STPREP, CNFTMS, EXTKLS, DIRKLS
C
        COMMON/C34567/TYPS(2), SMALER, BIGGER, STATUS(ICONFL,ILEN),CNFLCT,
      * CSHOTS(2,ICONFL)
        INTEGER TYPS, SMALER, BIGGER, CNFLCT, CSHOTS
  END
BK3     PROC
C   USED IN INPUT, MAIN, STPREP, EXTKLS
C
        COMMON/CIM46 /IOUNIT, OUTFIL, NSTAGE, NSTAGO, NDAYS
        INTEGER OUTFIL
  END
BK4     PROC
C   USED IN CNFALC, STPREP
C
        COMMON/C34    /NODDS, ODDS(2), CONFLO(2,IRANGE,IENVIR)
        INTEGER ODDS, CONFLO
  END
BK5     PROC
C   USED IN INPUT, CNFTMS
C
        COMMON/CI5    /LIMITS(2,ITYPS,2),
      * SENSOR(2,ITYPS,IENVIR,2), SIZE(2,ITYPS,IENVIR),
      * CNTRST(2,ITYPS,IENVIR,ISGNTR), LIGHT(IENVIR), MAG(ISENSR),
      * ATTN(ISENSR,IENVIR), BRIGHT(IENVIR), RCDET(2,ITYPS), NDTCTN,
      * NS2DCT(2,ITYPS)
        INTEGER SENSOR
        REAL LIGHT,MAG
  END
BK6     PROC
C   USED IN INPUT, MAIN, EXTKLS, DIRKLS
C
        COMMON/CIM67  /NOEXT(2),EXT(2,IEXT,ITYPS,IENVIR,IRANGE)
      * ,EXTN(2,IEXT,IENVIR,IRANGE,ICDPTH)
      * ,EXTK(2,IEXT,IENVIR,IRANGE,ICDPTH), EXTPER(2,IEXT)
      * ,EXTN1(2,IEXT,IENVIR,IRANGE), SHOTS(2,IENVIR,IRANGE)
      * ,AMOTYP(2,ITYPS,ITYPS), ESHOTS(2,IEXT,IENVIR,IRANGE)
      * ,BWPNS(2,ITYPS,IENVIR,IRANGE),INDDST(2,IEXT,IRANGE),LESDNS(2)
      * ,INDPAR(2,IEXT,ITYPS)
        INTEGER EXTK, EXTN1, SHOTS, AMOTYP, BWPNS
        REAL INDDST, INDPAR
        LOGICAL LESDNS
  END
BK7     PROC
C   USED IN MAIN, FRCALC, EXTKLS
C
        COMMON/CM16   /IOPTR3
  END
BK8     PROC
C   USED IN INPUT, MAIN, FRCALC
C
        COMMON/CIM1   /INSERT(IPHASE,2,ITYPS), EXTLOS(2,ITYPS),
      * PHASE(IDAYS)
        INTEGER PHASE
  END
BK9     PROC
C   USED IN INPUT, CNFALC, DIRKLS
C
        COMMON/CI37   /ENVDST(IENVIR)
  END
BK10    PROC
C   USED IN MAIN, DIRIO, DIRKLS
```

Figure D-16. Listing of FORTRAN Procs Identifying Common Blocks, Variables, Arrays, and Parameters of the Main Program of the AFP Combat Module
(page 1 of 4 pages)
(See Table D-2 for definitions of many of the variables and arrays)

```
C
      COMMON/CM27  /IOPTR4, CKILLS(2,ITYPS2,7)
      INTEGER CKILLS
 END
BK11  PROC
C  USED IN MAIN, STPREP, DIRKLS
C
      COMMON/CM47  /PKS(2,ITYPS,IPKS,IRANGE), MAXCON,
     *     IU1S, IU1T1, IU1T2, KEYU1, PK1(IRANGE)
 END
BK12  PROC
C  USED IN MAIN, CNFALC
C
      COMMON/CM3   /MAXODS
 END
BK13  PROC
C  USED IN MAIN, FRCALC, DIRIO, EXTKLS
C
      COMMON/CM126  /FORCES(2,ITYPS,ITYPS)
      INTEGER FORCES
 END
BK14  PROC
C  USED IN CNFALC,EXTKLS
C
      COMMON/C36   /CTR(2),DAYST(2)
     * ,SAVIT(4)
      INTEGER CTR , DAYST, SAVIT
 END
BK15  PROC
C  USED IN EXTKLS, DIRKLS
C
      COMMON/C67   /ATEMP(3,IPKS)
 END
BK16  PROC
C  USED IN MAIN, CNFALC, DIRIO, STPREP
      COMMON /CM234  / PROJCT(ITYPS,5)
      REAL PROJCT
 END
BK17  PROC
C  USED IN MAIN, FRCALC, EXTKLS, DIRKLS
      COMMON /CM167/ WPNS(2,ITYPS),WPNS1(2,ITYPS)
     * ,DUELS(IENVIR,IRANGE,ICDPTH,5)
     * ,TMS(2,IENVIR,IRANGE,ICDPTH,5), WPNS11(2,ITYPS)
      INTEGER WPNS, WPNS1, DUELS, WPNS11
 END
BK18  PROC
C  USED IN CNFTMS, DIRKLS
      PARAMETER RMAXT=2.0**30
 END
BK19  PROC
C  USED IN MAIN, CNFALC, EXTKLS, DIRKLS
      COMMON /CM367/ FORCS1(2)
      INTEGER FORCS1
 END
BK20  PROC
C  USED IN MAIN, INPUT, CNFALC
      COMMON /CIM3/ NENVIR
 END
BK21  PROC
C  USED IN MAIN, DIPIO, CNFALC, DIRKLS
      COMMON /CM237/ NUMWPN(ITYPS,2)
 END
BK22  PROC
C  USED IN INPUT, STPREP, CNFTMS
      COMMON /CI45  / REFIRE(2,ITYPS,ITYPS,IENVIR)
 END
BK23  PROC
C  USED IN MAIN, CNFALC, DIRKLS
      COMMON /CM37  /FORCS2(2),NOTPAR(2,IENVIR,IRANGE)
      INTEGER FORCS2,NOTPAR
 END
BK24  PROC
C  USED IN CNFALC, DIRKLS
      COMMON /C37  / RNGDST(IRANGE)
 END
BK25  PROC
C   USED IN MAIN, INPUT, CNFLC1, EXTKLS, DIRKLS
      COMMON/CIM678/ LOG(ICDPTH,2,ITYPS,9),LOGIT,STATE
      LOGICAL LOGIT,STATE
 END
BK27  PROC
C   USED FOR OUTPUT FILES
```

Figure D-16. Listing of FORTRAN Procs Identifying Common Blocks, Variables, Arrays, and Parameters of the Main Program of the AFP Combat Module (page 2 of 4 pages)

```
             COMMON /CM67/  TTLOS(2,ITLOS,IENVIR,IRANGE)
             INTEGER TTLOS
       END
      BK28   PROC
      C   USED IN CIPS, COPS, AND AFP FILE GENERATION
      C
             PARAMETER ITTOVC=4, ITTOTC=12, IINF=1
      C
             COMMON /0123/ TTOVC(2,ITYPS,ITTOVC), CTOVC(2,ITTOTC,ITTOVC)
           & , STOVC(2,ITTOVC), PERK(2,ITYPS), IVCM(2), ICM(2)
           & , CKILS1(ITYPS1,ITYPS2,2), PLOSS(2,ITYPS,ITYPS,3)
           & , TPTOVC(2,ITYPS), TPTOTC(2,ITYPS), NOS(2,ITYPS)
           & , CIPS1(2,ITYPS,ITTOVC),JTTOVC(2),JTTOTC(2)
           & , LOSSES(2,ITYPS), FAC(2,ITYPS)
      C
             INTEGER TPTOVC, TPTOTC
      C
             REAL PLOSS, CKILS1, PERK, TTOVC, CTOVC, STOVC, NOS, CIPS1, LOSSES
      C
       END
      BK29   PROC
      C   USED IN INDLOS,TPXTP
                COMMON/COM1/INLOSS(2,ITLOS)
       END
      PARM   PROC
      C   USED IN INPUT, MAIN, FRCALC, DIRIO, CNFALC, STPREP,
      C             CNFTMS, EXTKLS, DIRKLS
      C
      C GENERAL USAGE PARAMETERS
             PARAMETER SMALL1=.00001,SMALL2=.000001
             PARAMETER IPHASE=2,IDAYS=2,ICDPTH=5
             PARAMETER IODDS=50,IENVIR=1,IRANGE=6,ISGNTR=2
             PARAMETER ICONFL=1500,ISTAGE=10000000,IPKS=4
             PARAMETER ITYPS1=60,ITYPS2=60,ITYPS=MAX(ITYPS1,ITYPS2),IREPS=1
             PARAMETER LNCNT=30
             PARAMETER IPOS=4, IDAY=2, IVIS=2, ICIPTP=4
             PARAMETER ICASE=IPOS*IVIS*IDAY , ITHTR=1, ITPD=1, ICOMBO=1
             PARAMETER IP1= ICASE*ITHTR , IP3=1
             PARAMETER IP4=ITHTR*ICASE, IP2=ITHTR*ICASE*ITPD
             PARAMETER IP5=ICOMBO*ITHTR*ICASE*ITPD
      C  PARAMETERS FOR THE EXTERNAL KILLS
             PARAMETER  IEXT1=10,IEXT2=10,IEXT=MAX(IEXT1,IEXT2)
             PARAMETER  ILEN=IODDS+3, ITLOS=IEXT+1
      C PARAMETERS FOR DIRECT I/O
             PARAMETER N11=ITYPS*IENVIR,N12=IENVIR,IU1=20
              PARAMETER IRSZ1=3+IRANGE
             PARAMETER N21=ITYPS2,IU2=21
             PARAMETER NR2=IP1*ITYPS1*N21,IRSZ2=IRANGE*4+10
             PARAMETER NR6=IP1*ITYPS1,IRSZ6=5*ITYPS*6+10,IU6=25
             PARAMETER NR7=IP1*2*ITYPS,IRSZ7=2*ITYPS*8+10,IU7=26
             PARAMETER N84=IRANGE,N83=IENVIR*N84,N82=ITYPS*N83
           &    ,N81=IDAYS*N82,N80=IREPS*N81
             PARAMETER NR8=IP3*2*N80
             PARAMETER IRSZ8=ICDPTH*ITYPS*9*6+10,IU8=27
      C PARAMETERS FOR DETECT INTERFACES
             PARAMETER ISENSR=50
      C PARAMETERS FOR CKILLS DIRECT ACCESS I/O
             PARAMETER N31=ITYPS1,IU3=22
             PARAMETER NR3=IP5*IREPS*N31,IRSZ3=2*ITYPS2*7*6+10
             PARAMETER N41=ITYPS1,IU4=23
             PARAMETER NR4=IP3*IREPS*N41,IPSZ4=2*ITYPS*6+10
             PARAMETER IU5=24
             PARAMETER IRSZ5=4 + 2*2*IEXT*IENVIR*IRANGE*ICDPTH
             PARAMETER N90=IREPS, IU9=3
             PARAMETER NR9=IP5*IREPS*IDAYS,IRSZ9=2*ITYPS*ITYPS*6+10
             PARAMETER N100=IREPS, N101=IREPS*IDAYS, IU10=4
             PARAMETER NR10=IP5*N101*ITYPS1,IRSZ10=2*ITYPS2*6+10
             PARAMETER NR11=IP5*ICIPTP,IU11=7,IRSZ11=2*ITYPS*ICIPTP*10+10
             PARAMETER NR12=IP5*ICIPTP,IU12=8,IRSZ12=2*ITYPS*ICIPTP*10+10
             PARAMETER NR14=IP3*IREPS*IDAYS*ITYPS1*ITYPS2, IU14=31
           &     ,IRSZ14=(3+IENVIR*IRANGE*ICDPTH*5)+10
             PARAMETER IU15=32,IRSZ15=5+2*IENVIR*IRANGE*ICDPTH*5
      C PARAMETERS FOR NON-DIRECT ACCESS FILES
             PARAMETER N132=IREPS, N131=IDAYS*N132, IU13=9
             PARAMETER IRSZ13=2*ITLOS*IENVIR*IRANGE+4
             PARAMETER IUIN1=10, IUIN2=11, IUIN3=12, IUIN4=13, IUIN6=15
             PARAMETER IUIN7=16, IUIN8=14, IAFPIN=17, IUOUT1=18
             PARAMETER IUOUT3=33, IRSZU3=4 + IRANGE*IENVIR*IEXT*2 +
           &    IRANGE*IENVIR*2
      C
      C      INPUT FILES            NUMBER          UNIT          FORMAT
      C
```

Figure D-16. Listing of FORTRAN Procs Identifying Common Blocks, Variables, Arrays, and Parameters of the Main Program of the AFP Combat Module (page 3 of 4 pages)

```
C        03PKS3A                 10          IUIN1        24F3.3
C        03RNGDST3A              11          IUIN2        24I3
C        03PREF3A                12          IUIN3        FREE
C        03PROJ3A                13          IUIN4        FREE
C        03KILLSD3               15          IUIN6        FREE
C        03CIPSD3                16          IUIN7        FREE
C        03CSCSS                 14          IUIN8        FREE
C        03AFPIN                 17          IUIN9        IN SPECS
C
C      FILE TO UNIT ASSOCIATIONS
C
C        FILE          UNIT     NUMBER     FORMAT
C
C        PKS           IU1        20       UNFORMATTED (BINARY)
C        RNGDST        IU2        21       F4.2
C        CKILLS        IU3        22       I6
C        NUMWPN        IU4        23       I6
C        EXT           IU5        24       UNFORMATTED (BINARY)
C        PROJCT        IU6        25       F6.2
C        PREF          IU7        26       F8.6
C        LOG           IU8        27       I6
C        ALLOCATE      IU9         3       I6
C        FALLOC        IU10        4       I6
C        CIPS          IU11        7       F10.3
C        COPS          IU12        8       F10.3
C        TTLOS         IU13        9       UNFORMATTED (BINARY)
C        AFPIN         IAFPIN     17       IN  SPECS
C        AFPOUT        IUOUT1     18       IN  SPECS
C        ESHOTS,SHOTS  IUOUT3     33       UNFORMATTED (BINARY)
C        DUELS         IU14       31       I5
C        TIMES         IU15       32       UNFORMATTED (BINARY)
   END
```

Figure D-16. Listing of FORTRAN Procs Identifying Common Blocks, Variables, Arrays, and Parameters of the Main Program of the AFP Combat Module (page 4 of 4 pages)

Table D-2. **Notes on AFP Combat Module Common Blocks, Variables, and Arrays**
(page 1 of 5 pages)

Blocks/Variable name			Description
BK1			
NTYPS	I[a]	I[b]	Number of types on each side
CASE	I	C	A coded integer encoding the visibility, day/night and posture
INSEED	I	I	Pointer to a seed ID for FIMSLU
DSEED	D	C	Contains seed for DETECT
BK2			
TYPS	I	C	The IDs of the weapon type we're examining, one for each side
SMALER	I	C	The side having less weapons in the conflict at hand
BIGGER	I	C	The side having more weapons in the conflict at hand
CNFLCT	I	C	The number of duels in the conflict at hand
BK3			
IOUNIT	I	C	Input unit for symbolic file
OUTFIL	I	C	Unit number for printed output
NSTAGE	I	C	The expected number of shots in each duel of the conflict
NSTAGEO	I	I	The maximum number of shots allowed in any duel in any conflict
NDAYS	I	I	The number of days in the battle

[a]In the first column, I denotes integer variable type; R denotes real variable type; D denotes double precision variable type; and L denotes logical variable type.

[b]In the second column, I denotes input and C denotes computed.

Table D-2. Notes on AFP Combat Module Common Blocks, Variables, and Arrays (page 2 of 5 pages)

Blocks/Variable name			Description
BK4			
NODDS	I	C	Number of odds in the conflict (maximum of two, initially)
ODDS	I	C	The initial odds at which duels start out
CONFLO	I	C	The number of duels by range and environment
BK5			
LIMITS	I	I	The ranges (encoded 1-6) between which the weapons are effective
SENSOR	I	I	The sensor ID and the ID of the medium sensed (e.g., light, heat) for each weapon type
SIZE	R	I	Size of targets (minimum dimension in meters/2)[2]
CNTRST	R	I	Contrast thermal measured in degrees centigrade Contrast optical measured in foot-candles
LIGHT	R	I	Light level measured in foot-candles
MAG	R	I	Magnification
ATTN	R	I	Attenuation fractional reduction in intensity are kilometer traveled
BRIGHT	R	I	Brightness level sky to ground brightness ratio (no units)
BK6			
NOEXT	I	I	Number of types capable of indirect fire on each side; by convention, these must be the last types on each side
EXT	I	C	Combined preference and participation factors against each target type for each indirect shooter, and the kills/round for that shooter-target combination
EXTN	I	C	Number of indirect assignments for each indirect shooter against each target type

Table D-2. Notes on AFP Combat Module Common Blocks, Variables, and Arrays
(page 3 of 5 pages)

Blocks/Variable name	Description
EXTK I C	Kills by indirect shooter by target type
EXTN1 I C	Initial daily allocation of indirect shooter against targets
SHOTS I C	Direct fire shots by shooter and target type
AMOTYP I I	Coded table giving ammo type used for each shooter/target combination
ESHOTS I C	External shot counts, recomputed for each type or type combination
BK8	
INSERT I I	Initial weapons entering battle at start of day and reinforcements
PHASE I C	Grouping of days; currently day 1 is in Phase 1 and days 2 through 180 are in Phase 2
EXTLOS R I	Losses due to external (extraneous) causes
BK9	
ENVDST I C	Environmental distribution
BK10	
CKILLS I C	Cumulative direct fire kills by kill category
BK11	
PKS R I	Direct fire PKS
MAXCON I C	Maximum number of duels encountered
BK12	
MAXODS I C	The largest offered encountered in the model

Table D-2. Notes on AFP Combat Module Common Blocks, Variables, and Arrays (page 4 of 5 pages)

Blocks/Variable name	Description
BK13	
FORCES I C	The initial daily type or type preparticipation force allocations for direct fire weapons
BK16	
PROJCT R I	Contains participation rates, duration of conflict in minutes, number of sites (only one conflict site currently), and consecutive conflict/day for each type or type weapon combination
BK17	
WPNS R I	The daily weapon allocation for each type on each side, reduced as losses occur
WPNSI I C	The initial daily weapon allocations for each type on each side, unreduced
BK18	
RMAXT R C	2^{30}, used to indicate opposing weapons out of range/zero PK for direct fire weapons (2^{29} indicates nondetection)
BK19	
FORCSI I C	The currect total number of surviving weapons for the weapon types being processed; the total is across all conflicts
BK20	
NENVIR I I	The number of environments in which conflicts occur. Currently = 1
BK21	
NUMWPN I C	The number of direct fire weapons assigned; cumulative during battle

Table D-2. Notes on AFP Combat Module Common Blocks, Variables, and Arrays (page 5 of 5 pages)

Blocks/Variable name	Description
BK22	
REFIRE R C	Refire time in minutes for each shooter and target type
BK23	
FORCS2 I C	The current number of weapons for the two types in a particular conflict, decremented as losses occur.
NOTPAR I C	Nonparticipating weapons, only applies if participation rates less than 1.0 are used
BK24	
RNGDST R I	Range distribution
BK25	
LOG I C	Used in logging conflict
LOGIT L I	Writes out to LOG file if .TRUE.
STATE L I	Prints out other debug information if .TRUE.
BK27	
TTLOS I C	Total losses; direct and indirect by indirect shooter for each side

c. Program Routines

(1) **MAIN**

(a) Purpose. To control execution flow of the main program of the AFP Combat Module. Figure D-17 is a source listing of the main routine MAIN.

(b) Subtasks

1. Initializes many reference and working variables and arrays.

2. Directly or by calls to special subroutines, inputs needed data.

3. Controls the nested, looped calls to routines in accord with the scheme illustrated in Figure D-15.

4. Directly or by calls to special subroutines, outputs data needed by postprocessors and other AFP modules.

(2) **CNFALC**

(a) Purpose. To compute the initial odds and the composition of each conflict, i.e., compute the numbers of duels at each odds in each conflict for the given weapon types at a particular range and environment. Figure D-18 is a source listing of subroutine CNFALC.

(b) When called. Every time a new pairing of opposing weapon types occurs, called from MAIN.

(c) Subtasks

1. Computes the nonparticipating forces for each range and environment and subtract them from the available forces.

2. Tallies the direct fire assignments.

3. Computes the odds and the number of duels at odds as follows:

Let there m weapons of one type and n of the other.

Let max(m,n) = q min(m,n) + r, where all symbols are integers; q is the quotient, r is the remainder.

```
C  !!! NOTE THAT EXTERNAL WEAPON TYPES MUST HAVE THE LAST SEQUENCE !!!
C  !!! NUMBERS ON EACH SIDE I.E. MUST BE THE LAST !!!
C INCLUDE DECLARATIONS
       INCLUDE PARM
       INCLUDE BK1
       INCLUDE BK2
       INCLUDE BK3
       INCLUDE BK4
       INCLUDE BK6
       INCLUDE BK7
       INCLUDE BK8
       INCLUDE BK10
       INCLUDE BK11
       INCLUDE BK12
       INCLUDE BK13
       INCLUDE BK14
       INCLUDE BK16
       INCLUDE BK17
       INCLUDE BK19
       INCLUDE BK20
       INCLUDE BK21
       INCLUDE BK23
       INCLUDE BK25
       INCLUDE BK27
C DEFINE THE DIRECT ACCESS FILES
       DEFINE FILE IU2(NR2,IRSZ2,F,NEXT2)
       DEFINE FILE IU3(NR3,IRSZ3,F,NEXT3)
       DEFINE FILE IU4(NR4,IRSZ4,F,NEXT4)
       DEFINE FILE IU6(NR6,IRSZ6,F,NEXT6)
       DEFINE FILE IU7(NR7,IRSZ7,F,NEXT7)
       DEFINE FILE IU9(NR9,IRSZ9,F,NEXT9)
       DEFINE FILE IU10(NR10,IRSZ10,F,NEXT10)
       DEFINE FILE IU14(NR14,IRSZ14,V,NEXT14)
C SEQUENTIAL FILE DEFINITION:
       DEFINE FILE IU1(SDF,,IRSZ1,,IRSZ1)
       DEFINE FILE IU5(SDF,,IRSZ5,,IRSZ5)
       DEFINE FILE IU13(SDF,,IRSZ13,,IRSZ13)
       DEFINE FILE IU15(SDF,,IRSZ15,,IRSZ15)
       DEFINE FILE IUOUT3 (SDF,,IRSZU3,,IRSZU3)
C DECLARATIONS PARTICULAR TO KILLS
C
       INTEGER REPSX,DAYSX,TYPS1X,TYPS2X,ENVIRX,RANGEX,STAGEX,SIDEX,CNFX
       INTEGER ODDSX
       INTEGER COMBO,THTR,TPD,NVIS,NDAY,NPOS
       INTEGER LOTWO,LOONE, HITWO, HIONE
       LOGICAL ALLZ, IALLZ
C
       DIMENSION ITEMP6(2)
C
       DOUBLE PRECISION   SEED(100)
C
       DATA PHASE/1,2/,MAXCON/0/,MAXODS/0/
C
       DATA              (SEED(I),I=1,52)/
     *        123457.D0,  848661055.D0,  331868080.D0,  795694002.D0,
     *     1133506416.D0,  179392340.D0,  880089848.D0,  373453165.D0,
     *      330310551.D0, 1864683550.D0,  957584520.D0,  611600725.D0,
     *     1514665799.D0,  609430781.D0, 2126485978.D0,  230140735.D0,
     *     1460567237.D0, 1903540707.D0,  773122865.D0, 1753901423.D0,
     *      274992647.D0, 1588509478.D0, 1759740400.D0, 1941135750.D0,
     *       83729995.D0,  307509594.D0, 1235568876.D0, 1540164579.D0,
     *     1567253092.D0,   34485768.D0,  307246871.D0, 1222711937.D0,
     *      352414837.D0, 1812009750.D0,  892733067.D0, 1257585462.D0,
     *     1990506775.D0,   34124592.D0,  134177024.D0, 1176990974.D0,
     *     1265016020.D0, 1110128581.D0,   79706148.D0, 1970893844.D0,
     *      235610717.D0,  716522888.D0,   65884601.D0,   26530237.D0,
     *     1946768467.D0, 1915124666.D0,  605204808.D0, 1250036153.D0/
       DATA              (SEED(I),I=53,100)/
     *     1241237486.D0, 1826480319.D0, 1761451800.D0,  648974769.D0,
     *     1328095926.D0,  735026667.D0, 1922904882.D0,  112740804.D0,
```

Figure D-17. Source Listing of the Main Routine of the Main Program of the AFP Combat Module (page 1 of 6 pages)

```
          *      1690100682.D0,  673720340.D0, 1103976199.D0,  200368466.D0,
          *      1409578632.D0,   12672413.D0, 2082847903.D0, 2102585575.D0,
          *       585411451.D0,  560061851.D0, 1357483287.D0, 2035203198.D0,
          *      1995361080.D0, 2009993142.D0,  270796833.D0, 1364574012.D0,
          *      2037737285.D0,   37500927.D0, 2052071248.D0, 1860954082.D0,
          *      1475434694.D0, 2074851491.D0,  330340421.D0, 2037642869.D0,
          *       133673398.D0,  851290297.D0, 1651586143.D0,  301620020.D0,
          *      1510133733.D0, 1940663167.D0, 1877417917.D0,  617723160.D0,
          *      1093293442.D0, 1138218931.D0,  950324013.D0,  199964875.D0,
          *      1721714736.D0, 1039836864.D0, 1390471833.D0,  403248081.D0/
C
2         CONTINUE
          READ(5,10) COMBO,TPD,THTR,NVIS,NDAY,NPOS,INSEED
10        FORMAT()
          CASE=(NVIS-1)*IDAY*IPOS+(NDAY-1)*IPOS+(NPOS-1)
          PREF1=0
          PREF2=0
          PREF3=(THTR-1)*ICASE+CASE
          PREF4=(COMBO-1)*ITHTR*ITPD*ICASE+(TPD-1)*ITHTR*ICASE+PREF3
C       READ IN THE DENSITY REDUCTION INDICATORS
          READ(5,*) LESDNS(1),LESDNS(2)
C       READ IN THE TYPE LOOP LIMITS
          READ(5,10) LOONE,HIONE,LOTWO,HITWO
          IOUNIT=IUIN6
          OUTFIL=6
C READ IN ALL OF THE INPUT EXCEPT PKS AND RNGDST
          CALL INPUT
C         OPEN LOG FILE, IF APPROPRIATE
          IF (LOGIT) OPEN (UNIT=IU8, STATUS='UNKNOWN', ACCESS='DIRECT',
         &     RECL=IRSZ8, BLANK='ZERO', RFORM='F', RCDS=NR8,
         &     ASSOC=NEXT8)
          DSEED=SEED(INSEED)
C REPLICATION LOOP BEGINS HERE
          DO 50000 REPSX=1,IREPS
            LCOUNT=0
            WRITE(OUTFIL,25) REPSX
25          FORMAT(1H1,T51,' STARTING REPETITION ',I5,///)
          CALL IZERO (WPNS,2*ITYPS)
C DAY LOOP BEGINS HERE
            DO 40000 DAYSX=1,NDAYS
C             PREPARE TO (RE-) READ PKS
              REWIND IU1
              IU1T1 = 0  @ TO FORCE INITIAL READ
              KEYU1 = 0
              CALL FRCALC(DAYSX)
              CALL OUTPT(REPSX,DAYSX)
C       INITIALIZE THE DAILY BEGINNING WEAPON COUNTS
          CALL IZERO (BWPNS,2*IRANGE*IENVIR*ITYPS)
C       COMPUTE INITIAL DAILY WEAPON COUNTS FOR EXTKLS
                DO 86 TYPS1X=1,NTYPS(1)
                  DO 80 TYPS2X=1,NTYPS(2)
                    TYPS(1)=TYPS1X
                    TYPS(2)=TYPS2X
                    FORCS1(1)=FORCES(1,TYPS1X,TYPS2X)
                    FORCS1(2)=FORCES(2,TYPS2X,TYPS1X)
                    CALL CNFALC($80,TYPS1X,TYPS2X,0)
                    DO 78 ENVIRX=1,NENVIR
                    DO 78 RANGEX=1,IRANGE
                      ITEMP6(1)=0
                      ITEMP6(2)=0
                      DO 77 ODDSX=1,2
                        ITEMP6(SMALER)=ITEMP6(SMALER)
         &                            + CONFLO(ODDSX,RANGEX,ENVIRX)
                        ITEMP6(BIGGER)=ITEMP6(BIGGER)
         &                  + ODDS(ODDSX) * CONFLO(ODDSX,RANGEX,ENVIRX)
77                    CONTINUE
                      BWPNS(SMALER,TYPS(SMALER),ENVIRX,RANGEX)=
         &             BWPNS(SMALER,TYPS(SMALER),ENVIRX,RANGEX)
         &              + ITEMP6(SMALER)
                      BWPNS(BIGGER,TYPS(BIGGER),ENVIRX,RANGEX)=
```

Figure D-17. Source Listing of the Main Routine of the Main Program of the AFP Combat Module
(page 2 of 6 pages)

```
          &              BWPNS(BIGGER,TYPS(BIGGER),ENVIRX,RANGEX)
          &              + ITEMP6(BIGGER)
       78             CONTINUE
       80           CONTINUE
       86         CONTINUE
       C    WRITE THE TYPE LOOP BOUNDS IN THE SHOT FILE
       C BEGIN TYPE LOOPS
                DO 31000 TYPS1X=LOONE,HIONE
                  CALL DIRIO(TYPS1X,REPSX,DAYSX)
                  CALL GETPKS(TYPS1X)
       C  WRITE OUT THE POST PARTICIPATION ALLOCATIONS
                  IOPTRA=REPSX+(DAYSX-1)*N100+(TYPS1X-1)*N101
                  IOPTRA=IOPTRA+PREF4*IREPS*IDAYS*ITYPS1
                  WRITE(IU10'IOPTRA,1030,ERR=130)
          &         (IFIX(FORCES(1,TYPS1X,J)*PROJCT(J,1)+0.5),J=1,NTYPS(2))
          &         ,(IFIX(FORCES(2,J,TYPS1X)*PROJCT(J,2)+0.5),J=1,NTYPS(2))
                  GOTO 200
       130        WRITE(OUTFIL,140) REPSX,DAYSX
       140        FORMAT(1X,30(1H*),5X,'REP= ',I3,3X,'DAY= ',I3,3X
          &       ,'FALLOC DIRECT WRITE ERROR',5X,30(1H*))
                  STOP
       200        CONTINUE
                  DO 30000 TYPS2X=LOTWO,HITWO
                    TYPS(1)=TYPS1X
                    TYPS(2)=TYPS2X
       C    INITIALIZE THE TOTAL LOSSES TABLE
                    CALL IZERO (TTLOS, 2*ITLOS*IENVIR*IRANGE)
       C    INITIALIZE THE EXTERNAL SHOT COUNTER FOR EACH TYPE ON TYPE
       C       COMBINATION
                    CALL IZERO (SHOTS, 2*IENVIR*IRANGE)
                    CALL ZERO (ESHOTS, 2*IEXT*IENVIR*IRANGE)
                    CALL ZERO (EXTN, 2*IEXT*IENVIR*IRANGE*ICDPTH)
                    CALL IZERO (EXTK, 2*IEXT*IENVIR*IRANGE*ICDPTH)
       C    INITIALIZE THE DUELS AND TIMES TABLES
                    CALL IZERO (DUELS, IENVIR*IRANGE*ICDPTH*5)
                    CALL ZERO (TMS, 2*IENVIR*IRANGE*ICDPTH*5)
       C    PLACE THE INITIAL FORCES INTO FORCS1
                    FORCS1(1)=
          &           FORCES(1,TYPS(1),TYPS(2))
                    FORCS1(2)=
          &           FORCES(2,TYPS(2),TYPS(1))
                    IF ( FORCS1(1)*FORCS1(2) .EQ. 0 ) GOTO 30000
                    CALL CNFALC($30000,TYPS1X,TYPS2X,1)
                    NOCONF=PROJCT(TYPS2X,5)
                    IF ( NOCONF .GT. ICDPTH ) THEN
                      WRITE(OUTFIL,1000) NOCONF
       1000           FORMAT(1X,30(1H*),5X,'NUMBER OF CONFLICTS EXCESSIVE',
          &           2X,I5,5X,30(1H*))
                      STOP
                    ENDIF
       C    COMPUTE THE EXTERNAL WEAPON ASSIGNMENTS
                    CALL EXTASG
       C THE ENVIRONMENT LOOP
                    DO 21000 ENVIRX=1,NENVIR
       C RANGE LOOP
                      DO 20000 RANGEX=1,IRANGE
                        IF ( LOGIT ) THEN
       C    INITIALIZE THE LOG TABLE
                         DO 1650 SIDEX=1,2
       C    FOR LOG, SIDEX IS THE VICTIM TYPE
       C    READ IN THE LOG TABLE, AND INITIALIZE IF NOT THERE
                         IOPTR8=(SIDEX-1)*N80+(REPSX-1)*N81+(DAYSX-1)*N82
          &                    +(TYPS(SIDEX)-1)*N83+(ENVIRX-1)*N84+RANGEX
                         IOPTR8=IOPTR8+PREF2*2*IREPS*IDAYS*ITYPS*IENVIR
          &                     *IRANGE
                         READ(IU8'IOPTR8,16000,ERR=1500) NOCONF,
          &                (((LOG(I,SIDEX,K,L),L=1,9),K=1,NTYPS(3-SIDEX)),
          &                  I=1,NOCONF)
                         GOTO 1650
       1500              CONTINUE
```

Figure D-17. Source Listing of the Main Routine of the Main Program of the AFP Combat Module (page 3 of 6 pages)

```
                      CALL IZERO (LOG, ICDPTH*2*ITYPS*9)
1650                  CONTINUE
                     ENDIF
                    DO 15000 CNFX=1,NOCONF
                     CALL CNFLC1($15500,TYPS2X,ENVIRX,RANGEX,CNFX)
CCCCCCCC FOLLOWING TRACE ADDED TO ASSIST TESTING, 1 FEB 84:
CC       WRITE (6,66) TYPS1X, TYPS2X, DAYSX, CNFX, RANGEX,
CC     &        FORCS2(1), FORCS2(2)
CC66     FORMAT (' *** BLUE TYPE ', I2, ' VS RED TYPE ', I2,
CC     &     ' DAY ', I2, ' CONFLICT ', I2, ' RANGE ', I2, /,
CC     &     ' *** BLUE FORCE ', I6, ' RED FORCE ', I6)
CCCCCCCC END OF TEST TRACE
C    STORE THE ASSIGNMENTS
                     DO 1700 SIDEX=1,2
                     DO 1700 J=1,2
                       LOG(CNFX,SIDEX,TYPS(3-SIDEX),J)=
     &                   LOG(CNFX,SIDEX,TYPS(3-SIDEX),J)+
     &                   FORCS2(J)*PROJCT(TYPS2X,4)
1700                 CONTINUE
                     CALL STPREP($15500,RANGEX,ENVIRX,TYPS1X,TYPS2X)
                     DO 10000 STAGEX=1,NSTAGE
                       CALL CNFTMS(ENVIRX,RANGEX,CNFX,STAGEX)
                       CALL EXTKLS(ENVIRX,TYPS2X,CNFX,RANGEX,STAGEX)
                       CALL DIRKLS(TYPS2X,ENVIRX,RANGEX,CNFX,STAGEX)
CC                       IF ( STATE ) THEN
CC                         IF ( MOD(LCOUNT,LNCNT) .EQ. 0 ) THEN
CC                           WRITE(OUTFIL,1725)
CC1725                       FORMAT(/,1X,I2,'  REPS  ',T10,'  DAYS  ',T18
CC     &                       ,' TYPS 1 ',T26,' TYPS 2 ',T34,' ENVIRX '
CC     &                       ,T42,' RANGEX ',T50,'  CNFX  ',T58
CC     &                       ,' STAGEX ',T68,' CONFLO ',T86
CC     &                       ,'  ODDS  ',T104,' FORCS2 ',T122
CC     &                       ,' CNFLCT ',/,1X,10('-'),T15,'CKILLS',/
CC     &                       ,1X,15('-'))
CC                         ENDIF
CC                         LCOUNT=LCOUNT+1
CC                         WRITE(OUTFIL,1750) REPSX,DAYSX,TYPS1X,TYPS2X
CC     &                     ,ENVIRX,RANGEX,CNFX,STAGEX
CC     &                     ,(CONFLO(L,RANGEX,ENVIRX),L=1,2)
CC     &                     ,(ODDS(L),L=1,2),(FORCS2(L),L=1,2),CNFLCT
CC     &                     ,((CKILLS(L,TYPS(2),K),K=1,7),L=1,2)
CC1750                     FORMAT(1X,T2,I6,T10,I5,T18
CC     &                       ,I6,T26,I6,T34,I5
CC     &                       ,T42,I5,T50,I6,T58
CC     &                       ,I6,T68,2(I8),T86
CC     &                       ,2I8,T104,2I8,T122
CC     &                       ,I5,/,1X,10('-'),T15,2(7(I6,1X),3X),/
CC     &                       ,1X,15('-'))
CC                       ENDIF
C    SKIP IF NO ENTRIES LEFT
C    END OF STAGE LOOP
10000                CONTINUE
C    END OF CONFLICTS LOOP
15000               CONTINUE
C    IF UNUSUAL RETURN, JUMP HERE TO SAVE THE LOG
15500               CONTINUE
                    IF ( LOGIT ) THEN
C    WRITE OUT THE LOG TABLE
C         SIDEX POINTS TO THE SIDE OF THE VICTIM TYPE
                     DO 17000 SIDEX=1,2
                      IOPTR8=(SIDEX-1)*N80+(REPSX-1)*N81+(DAYSX-1)*N82
     &                   +(TYPS(SIDEX)-1)*N83+(ENVIRX-1)*N84+RANGEX
                      IOPTR8=IOPTR8+PREF2*2*IREPS*IDAYS*ITYPS*IENVIR
     &                   *IRANGE
                      WRITE(IU8'IOPTR8,16000,ERR=16500) NOCONF,
     &                  (((LOG(I,SIDEX,K,L),L=1,9),K=1,NTYPS(3-SIDEX)),
     &                   I=1,NOCONF)
16000                 FORMAT(500(500I6))
                      GOTO 17000
16500                 WRITE(OUTFIL,16600) REPSX,DAYSX,TYPS(3-SIDEX),
```

Figure D-17. Source Listing of the Main Routine of the Main Program of the AFP Combat Module (page 4 of 6 pages)

```
      &                  ENVIRX,RANGEX
16600                  FORMAT(1X,30(1H*),5X,'LOG WRITE ERROR',5X,'INDICES=',
      &                  5(3X,I5),5X,30(1H*))
                       STOP
17000                 CONTINUE
                     ENDIF
C    END OF RANGE LOOP
20000               CONTINUE
C    END OF ENVIRONMENT LOOP
21000             CONTINUE
C    WRITE OUT THE SHOT COUNTS (ESHOTS, SHOTS)
        IF (.NOT. IALLZ (SHOTS, 2*IENVIR*IRANGE) .OR.
      &     .NOT. ALLZ (ESHOTS, 2*IEXT*IENVIR*IRANGE))
      &  WRITE(IUOUT3) REPSX, DAYSX, TYPS1X, TYPS2X,
      &           ESHOTS, SHOTS
CC                  DO 21300 I=1,2
CC                   DO 21300 RANGEX = 1,IRANGE
CC                    WRITE(OUTFIL,21200) TYPS1X,TYPS2X
CC       &             ,I,RANGEX,(ESHOTS(I,J,1,RANGEX),J=1,NOEXT(I))
CC21200               FORMAT(/,1X,'EXTERNAL SHOTS FOR TYPES ',2I7,/
CC       &             ,1X,'SHOOTER SIDE= ',I2,' RANGE=',I2
CC       &             ,/,' SHOT COUNTS ',(T15,10(F7.2,1X)) )
CC21300             CONTINUE
C    WRITE OUT THE DUEL DATA
                 IOPTRE=REPSX+(DAYSX-1)*IREPS+(TYPS1X-1)*IDAYS*IREPS
      &             +(TYPS2X-1)*ITYPS1*IDAYS*IREPS
                 IOPTRE=IOPTRE+PREF2*IREPS*IDAYS*ITYPS1*ITYPS2
                 WRITE(IU14'IOPTRE,ERR=22200) NENVIR,IRANGE,NOCONF
      &             ,((((DUELS(I,J,K,L),L=1,5),K=1,NOCONF),J=1,IRANGE)
      &             ,I=1,NENVIR)
                 GOTO 22400
22200            WRITE(OUTFIL,22300)
22300            FORMAT(1X,30(1H*),5X,'DUELS WRITE ERROR',5X,30(1H*))
                 STOP
22400            CONTINUE
CC                  WRITE(OUTFIL,22500) NENVIR,IRANGE,NOCONF
CC22500             FORMAT(/,1X,'DUELS FOR ',I3,' ENVIRONMENTS, '
CC       &              ,I3,' RANGES, AND ',I2,' CONFLICTS AT MOST')
CC                  DO 22700 I=1,NENVIR
CC                  DO 22700 J=1,IRANGE
CC                     WRITE(OUTFIL,22600) I,J,((DUELS(I,J,K,L),L=1,5)
CC       &               ,K=1,NOCONF)
CC22600                FORMAT( 1X,'ENVIRONMENT ',I2,3X,'RANGE ',I2,3X
CC       &               ,500(T32,2(5(I5,2X),10X),/) )
CC22700             CONTINUE
C    WRITE OUT THE TIME DATA
                 IF (.NOT. ALLZ (TMS, 2*IENVIR*IRANGE*ICDPTH*5))
      &                WRITE (IU15) REPSX, DAYSX, TYPS1X, TYPS2X,
      &                NOCONF, TMS
CC                  WRITE(OUTFIL,23500) NENVIR,IRANGE,NOCONF
CC23500             FORMAT(/, 1X,'TIMES FOR ',I3,' ENVIRONMENTS, '
CC       &              ,I3,' RANGES, AND ',I2,' CONFLICTS AT MOST')
CC                  DO 23700 I=1,NENVIR
CC                  DO 23700 J=1,IRANGE
CC                     WRITE(OUTFIL,23600) I,J,((M,(TMS(M,I,J,K,L),L=1,5)
CC       &               ,M=1,2),K=1,NOCONF)
CC23600                FORMAT( 1X,'ENVIRONMENT ',I2,3X,'RANGE ',I2
CC       &               ,500(:T32,' SIDE ',I1,2X,5(F12.5,2X),/) )
CC23700             CONTINUE
C                WRITE OUT EXTERNAL ASSIGNMENTS, KILLS
                 IF (.NOT. ALLZ (EXTN, 2*IEXT*IENVIR*IRANGE*ICDPTH) .OR.
      &                .NOT. IALLZ (EXTK, 2*IEXT*IENVIR*IRANGE*ICDPTH))
      &                WRITE (IU5) REPSX, DAYSX, TYPS1X, TYPS2X,
      &                EXTN, EXTK
C    WRITE OUT THE TOTAL LOSSES TABLE, IF THERE WERE ANY LOSSES
                 IF (.NOT. IALLZ (TTLOS, 2*ITLOS*IENVIR*IRANGE))
      &                WRITE (IU13) REPSX, DAYSX, TYPS1X,
      &             TYPS2X, TTLOS
C    END OF SIDE 2 TYPES LOOP
30000           CONTINUE
```

Figure D-17. Source Listing of the Main Routine of the Main Program of the AFP Combat Module (page 5 of 6 pages)

```
        DO 30150 I=1,2
        DO 30150 J=1,NTYPS(2)
          ISUM=0
          DO 30050 K=1,7
            ISUM=ISUM+CKILLS(I,J,K)
30050     CONTINUE
CC          IF ( ISUM .LE. 0 ) GOTO 30150
CC          WRITE(OUTFIL,30100) I,TYPS(1),J,(CKILLS(I,J,K),K=1,7)
CC30100     FORMAT(/,9X,'CKILLS',/,1X,3(I5,2X),100(T30,10(I5,2X),/))
30150 CONTINUE
              IOPTR3=(REPSX-1)*N31+TYPS1X
              IOPTR3=IOPTR3+PREF4*IREPS*ITYPS1
              WRITE(IU3'IOPTR3,1030,ERR=30200)
     &          (((CKILLS(I,J,K),K=1,7),J=1,NTYPS(2)),I=1,2)
1030          FORMAT(500(500I6))
              GOTO 30400
30200         WRITE(OUTFIL,30300)
30300         FORMAT(1X,30(1H*),5X,'CKILLS WRITE ERROR',
     &               5X,30(1H*))
              STOP
30400         CONTINUE
              IOPTR4=(REPSX-1)*N41+TYPS1X
              IOPTR4=IOPTR4+PREF2*IREPS*ITYPS1
              WRITE(IU4'IOPTR4,1030,ERR=30500)
     &          ((NUMWPN(I,J),J=1,2),I=1,NTYPS(2))
              GOTO 30700
30500         WRITE(OUTFIL,30600)
30600         FORMAT(1X,30(1H*),5X,'NUMWPN WRITE ERROR',5X,30(1H*))
              STOP
30700         CONTINUE
CCC     WRITE OUT THE NUMBER OF DIRECT ENCOUNTERS
CC        DO 30900 J=1,2
CC          ISUM=0
CC          DO 30750 I=1,NTYPS(2)
CC            ISUM=ISUM+NUMWPN(I,J)
CC30750     CONTINUE
CC          IF ( ISUM .LE. 0 ) GOTO 30900
CC          WRITE(OUTFIL,30800) J,TYPS1X,(NUMWPN(I,J),I=1,NTYPS(2))
CC30800     FORMAT(//,1X,'ENCOUNTERS',5X,'SIDE ',I1,5X,'SIDE 1 TYPE ',I3,3X
CC     &    ,500(T50,10(I6,2X),/))
CC30900 CONTINUE
C    END OF SIDE 1 TYPES LOOP
31000       CONTINUE
C    END OF DAY LOOP
40000     CONTINUE
C   END OF REPS LOOP
50000 CONTINUE
        WRITE(OUTFIL,50100) MAXCON
50100 FORMAT(T41,'THE MAXIMUM NUMBER OF CONFLICTS WAS ',
     &  2X,I6)
        WRITE(OUTFIL,50200) MAXODS
50200 FORMAT(//,T41,'THE MAXIMUM ODDS WERE ',
     &  2X,I6)
        STOP
        END
```

Figure D-17. Source Listing of the Main Routine of the Main Program of the AFP Combat Module (page 6 of 6 pages)

```
      SUBROUTINE CNFALC(*,TYPS1X,TYPS2X,CASE1)
C
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK2
      INCLUDE BK4
      INCLUDE BK9
      INCLUDE BK12
      INCLUDE BK13
      INCLUDE BK14
      INCLUDE BK16
      INCLUDE BK19
      INCLUDE BK20
      INCLUDE BK21
      INCLUDE BK23
      INCLUDE BK24
C
      DIMENSION CONFL1(2),CLASS(2,IRANGE,IENVIR)
C
      LOGICAL LOG
C
      INTEGER TYPS2X,TYPS1X,CONFL1,ODDSX,POINT,POINT1,COUNT,CASE1,TPS1X
      INTEGER OUTFIL
C
      SAVE TPS1X
C
2     CONTINUE
      OUTFIL=6
C   READ IN THE RANGE DISTRIBUTION
                  IOPTR2=(TYPS1X-1)*N21+TYPS2X
                  IOPTR2=IOPTR2+PREF3*ITYPS1*ITYPS2
                  READ(IU2'IOPTR2,1075,ERR=1085)(RNGDST(I),I=1,IRANGE)
1075              FORMAT(500(500F4.2))
                  GOTO 1095
1085              WRITE(6,1090)
1090              FORMAT(1X,30(1H*),5X,'RNGDST LOAD ERROR',
     &                5X,30(1H*))
                  STOP
1095              CONTINUE
C   INITIALIZE
                  ODDS(1)=0
                  ODDS(2)=0
                  DO 90 ODDSX=1,2
                  DO 90 ENVIRX=1,NENVIR
                  DO 90 RANGEX=1,IRANGE
                    CONFLO(ODDSX,RANGEX,ENVIRX)=0
90                CONTINUE
C   READ IN PROJCT FILE IF INITIAL CALL TO CNFALC
C   CHECK TO SEE IF THIS ISTHE FIRST CALL
                  LOG=( TYPS1X .EQ. 1 ) .AND. (  TYPS2X .EQ. 1 )
C   CHECK TO SEE IF THIS IS A CASE 0 CALL
                  LOG=LOG .OR. ( CASE1 .EQ. 0 )
                  IF ( LOG ) THEN
                    TPS1X=TYPS1X
                    IOPTR6=TYPS1X
                    READ(IU6'IOPTR6,110,ERR=200)
     &                (( PROJCT(I,J),J=1,5),I=1,NTYPS(2))
110                 FORMAT(500(500F6.2))
                    GOTO 300
200                 WRITE(OUTFIL,210)
210                 FORMAT(1X,30(1H*),' PROJCT READ ERROR, CNFALC ',30(1H*))
                    STOP
300                 CONTINUE
                  ENDIF
C COMPUTE THE FORCES ACTUALLY ENGAGING BY MULTIPLYING BY THE
C   PROBABILITY OF ENGAGEMENT
                    FORCS2(1)=FORCS1(1)
                    FORCS2(2)=FORCS1(2)
                    DO 100 ENVIRX=1,NENVIR
                    DO 100 RANGEX=1,IRANGE
                    DO 100 I=1,2
C   COMPUTE NON-PARTICIPANTS
                      NOTPAR(I,ENVIRX,RANGEX)=FORCS2(I)*RNGDST(RANGEX)
     &                                         *ENVDST(ENVIRX)
     &                                         *(1.0-PROJCT(TYPS2X,I))
C SUBTRACT OFF NON-PARTICIPANTS
                      FORCS2(I)=FORCS2(I)-NOTPAR(I,ENVIRX,RANGEX)
100                 CONTINUE
                      IF ( FORCS2(1)*FORCS2(2) .LE. 0 ) RETURN 1
C SET UP POINTER TO SIDE WITH MORE WEAPONS ( OR SIDE 1 IF
C   BOTH SIDES HAVE AN EQUAL NUMBER OF WEAPONS )
```

Figure D-18. Source Listing of the CNFALC Subroutine of the Main Program of the AFP Combat Module (page 1 of 2 pages)

```
                    BIGGER=1
                    IF ( FORCS2(1) .LT.
     &                   FORCS2(2) ) BIGGER=2
                    SMALER=3-BIGGER
C NODDS WILL CONTAIN THE NUMBER OF ODDS CLASSES ( 1 OR 2 )
                    NODDS=1
C PLACE THE NUMBER OF CONFLICTS INTO COUNT
                    COUNT=MIN( FORCS2(1),
     &                   FORCS2(2) )
C PLACE THE LOWER ODDS INTO ODDS(1)
                    ODDS(1)=FORCS2(BIGGER)/
     &                   FORCS2(SMALER)
                    ODDS(2)=0
C FIND THE REMAINDER
                    ITEMP=MOD( FORCS2(BIGGER),
     &                   FORCS2(SMALER) )
C STORE THE NUMBER OF CONFLICTS AT LOWER ODDS
                    CONFL1(1)=COUNT-ITEMP
C STORE THE NUMBER OF CONFLICTS AT HIGHER ODDS
                    CONFL1(2)=ITEMP
                    IF ( (CONFL1(1)+CONFL1(2)) .LE. 0 ) RETURN 1
                    IF ( ITEMP .GT. 0 ) THEN
                      NODDS=2
                      ODDS(2)=ODDS(1)+1
                    ENDIF
C CHECK FOR OUT OF BOUNDS ODDS
                    IF ( ODDS(NODDS) .GT. IODDS ) THEN
                      WRITE(6,1050) NODDS,ODDS(NODDS),TYPS1X,TYPS2X
1050                  FORMAT(10(1H*),5X,'CNFALC- ODDS OUT OF BOUNDS',
     &                     5X,4(I10,2X),5X,10(1H*))
                    ENDIF
C  TO RECORD THE HIGHEST ODDS ENCOUNTERED
                    IF ( ODDS(NODDS) .GT. MAXODS ) MAXODS=ODDS(NODDS)
C DISTRIBUTE THE CONFLICTS
                    CTR(1)=0
                    IF ( NODDS .GT. 1 ) CTR(2)=0
                    DO 1100 RANGEX=1,IRANGE
                    DO 1100 ENVIRX=1,NENVIR
                    DO 1100 ODDSX=1,NODDS
                      TEMP=CONFL1(ODDSX)*RNGDST(RANGEX)*ENVDST(ENVIRX)
                      ITEMP=TEMP
C MAKING THE INTEGRAL ALLOCATIONS
                      CONFLO(ODDSX,RANGEX,ENVIRX)=ITEMP
C SAVING THE FRACTIONAL PARTS
                      CLASS(ODDSX,RANGEX,ENVIRX)=TEMP-ITEMP
C ADD UP THE ALLOCATED WEAPONS
                      CTR(ODDSX)=CTR(ODDSX)+ITEMP
1100                CONTINUE
C FIND THE NUMBER OF UNALLOCATED CONFLICTS AT EACH ODDS
                    DO 1150 ODDSX=1,NODDS
                      CTR(ODDSX)=CONFL1(ODDSX)-CTR(ODDSX)
1150                CONTINUE
C ROUND CONFLO, THE CONFLICTS BY ODDS, RANGE, ENVIRONMENT
C LOOP ON THE ODDS CLASS
                    DO 1400 ODDSX=1,NODDS
                      COUNT=CTR(ODDSX)
                      IF ( COUNT .EQ. 0 ) GOTO 1400
1200                  CONTINUE
C INITIALIZE POINTERS TO RANGE AND ENVIRONMENT INDICES
                      POINT=1
                      POINT1=1
C FIND THE LARGEST FRACTIONAL PART
                      DO 1300 RANGEX=1,IRANGE
                      DO 1300 ENVIRX=1,NENVIR
                        IF ( CLASS(ODDSX,RANGEX,ENVIRX) .GT.
     &                       CLASS(ODDSX,POINT,POINT1) ) THEN
                          POINT=RANGEX
                          POINT1=ENVIRX
                        ENDIF
1300                  CONTINUE
C ALLOCATE TO THE LARGEST REMAINDER
                      CONFLO(ODDSX,POINT,POINT1)=
     &                    CONFLO(ODDSX,POINT,POINT1)+1
C INSURE NO FURTHER ALLOCATION TAKE PLACE
                      CLASS(ODDSX,POINT,POINT1)=-1.0
                      COUNT=COUNT-1
                      IF ( COUNT .GT. 0 ) GOTO 1200
1400                CONTINUE
C    SAVE THE STATUS FOR CNFLC1
                    SAVIT(1)=SMALER
                    SAVIT(2)=BIGGER
                    SAVIT(3)=ODDS(1)
                    SAVIT(4)=ODDS(2)
                  RETURN
                  END
```

Figure D-18. Source Listing of the CNFALC Subroutine of the Main Program of the AFP Combat Module (page 2 of 2 pages)

The odds are q + 1:1 and q:1, where the q weapons belong to the more numerous side.

The number of duels is min(m,n).

The number of duels at q + 1:1 is r.

The number of duels at q:1 is min(m,n) - r.

4. Allocates the conflicts to each range and environment using the range distribution and applying the same treatment of integers and fractions as does FRCALC.

(3) CNFLC1

(a) Purpose. To compute the number of weapons for each side in the conflict. When the conflict is between survivors, to compute the odds and the number of duels at each odd (i.e., conflict composition) for every conflict. Figure D-19 is a source listing of subroutine CNFLC1.

(b) When Called. Every time another conflict is scheduled, for every pair of opposing types, environment and range. This routine is called from MAIN.

(c) Subtasks. Note that conflicts for a given pair of opposing types, at a given environment and a given range, are processed in succession in a given day; i.e., when one conflict ends, the survivors from another conflict which is then processed.

1. For first conflicts at a given environment and range.

a. In the first conflict of every day and for each pair of opposing types, status information developed by CNFALC is saved to be refused by other first conflicts at different environments and ranges for the same opposing types. This information includes the odds, which side is smaller, and which side is larger.

b. In first conflicts for other ranges/environments, the odds, smaller and bigger indicators are reset.

c. In any first conflict, the number of weapons of each type participating in the conflict at hand is computed.

2. For subsequent conflicts.

a. Computes the participating forces at the given range and environment.

```
UNCLASSIFIED*H7AFP(1).CNFLC1(26)
      SUBROUTINE CNFLC1(*,TYPS2X,ENVIRX,RANGEX,CNFX)
C
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK2
      INCLUDE BK4
      INCLUDE BK9
      INCLUDE BK12
      INCLUDE BK13
      INCLUDE BK14
      INCLUDE BK16
      INCLUDE BK17
      INCLUDE BK19
      INCLUDE BK20
      INCLUDE BK21
      INCLUDE BK23
      INCLUDE BK24
C
      DIMENSION NOTIN(2)
C
      INTEGER TYPS2X,COUNT,RANGEX,ENVIRX,CNFX,NOTIN,ODDSX
C
      SAVE NOTIN
C
2     CONTINUE
C
      IF ( CNFX .GT. 1 ) THEN
C    USE THE REMAINING ENTRIES IN THE STATUS TABLE TO OBTAIN SURVIVORS
C    SUM THE SURVIVORS AND THE NON-PARTICIPANTS
          FORCS2(1)=FORCS2(1)+NOTIN(1)
          FORCS2(2)=FORCS2(2)+NOTIN(2)
C    COMPUTE THE NON-PARTICIPANTS
          NOTIN(1)=FORCS2(1)*(1.0-PROJCT(TYPS2X,1))
          NOTIN(2)=FORCS2(2)*(1.0-PROJCT(TYPS2X,2))
C    SUBTRACT OFF THE NON PARTICIPANTS
          FORCS2(1)=FORCS2(1)-NOTIN(1)
          FORCS2(2)=FORCS2(2)-NOTIN(2)
          IF ( FORCS2(1)*FORCS2(2) .LE. 0 ) RETURN 1
C SET UP POINTER TO SIDE WITH MORE WEAPONS ( OR SIDE 1 IF
C    BOTH SIDES HAVE AN EQUAL NUMBER OF WEAPONS )
              BIGGER=1
              IF ( FORCS2(1) .LT.
     &             FORCS2(2) ) BIGGER=2
              SMALER=3-BIGGER
C NODDS WILL CONTAIN THE NUMBER OF ODDS CLASSES ( 1 OR 2 )
              NODDS=1
C PLACE THE NUMBER OF CONFLICTS INTO COUNT
              COUNT=MIN( FORCS2(1),
     &             FORCS2(2) )
C PLACE THE LOWER ODDS INTO ODDS(1)
              ODDS(1)=FORCS2(BIGGER)/
     &             FORCS2(SMALER)
              ODDS(2)=0
C FIND THE REMAINDER
              ITEMP=MOD( FORCS2(BIGGER),
     &             FORCS2(SMALER) )
              IF ( ITEMP .GT. 0 ) THEN
                NODDS=2
                ODDS(2)=ODDS(1)+1
              ENDIF
C CHECK FOR OUT OF BOUNDS ODDS
              IF ( ODDS(NODDS) .GT. IODDS ) THEN
                WRITE(6,1050) NODDS,ODDS(NODDS),TYPS(1),TYPS(2)
1050            FORMAT(10(1H*),5X,'CNFLC1- ODDS OUT OF BOUNDS',
     &                 5X,4(I10,2X),5X,10(1H*))
              ENDIF
C STORE THE NUMBER OF CONFLICTS AT LOWER ODDS
              CONFLO(1,RANGEX,ENVIRX)=COUNT-ITEMP
C STORE THE NUMBER OF CONFLICTS AT HIGHER ODDS
              CONFLO(2,RANGEX,ENVIRX)=ITEMP
              ITEMP1=CONFLO(1,RANGEX,ENVIRX)
              IF ( NODDS .GT. 1 ) ITEMP1=ITEMP1
     &          + CONFLO(2,RANGEX,ENVIRX)
CC *****
CC        WRITE(6,91000) RANGEX,CNFX,(FORCS1(I),FORCS2(I),NOTIN(I)
CC     &    ,ODDS(I),CONFLO(I,RANGEX,ENVIRX),I=1,2)
CC     &    ,SMALER,BIGGER,NOTPAR(1,ENVIRX,RANGEX)
CC     &    ,NOTPAR(2,ENVIRX,RANGEX)
CC 91000 FORMAT(1X,'2 CNFLC1',' FOR RANGE,CNFX= ',2(I2,1X),/,2(' FORCS1'
CC     &',FORCS2,NOTIN,ODDS,CONFLO= ',5(I8,2X),/),' SMALER= '
CC     & ,I1,1X,'BIGGER= ',I1,1X,'NOTPAR= ',2(I6,1X))
CC *****
              IF ( ITEMP1 .LE. 0 ) RETURN 1
C   TO RECORD THE HIGHEST ODDS ENCOUNTERED
              IF ( ODDS(NODDS) .GT. MAXODS ) MAXODS=ODDS(NODDS)
      ELSE
C    FIRST CONFLICT IN CHAIN
          NOTIN(1)=NOTPAR(1,ENVIRX,RANGEX)
          NOTIN(2)=NOTPAR(2,ENVIRX,RANGEX)
C      RESTORE PREVIOUSLY SAVED CONDITIONS
          SMALER=SAVIT(1)
          BIGGER=SAVIT(2)
          ODDS(1)=SAVIT(3)
          ODDS(2)=SAVIT(4)
C    SET THE NUMBER OF ODDS CLASSES
          NODDS=2
          IF ( CONFLO(2,RANGEX,ENVIRX) .EQ. 0 ) NODDS=1
          FORCS2(SMALER)=CONFLO(1,RANGEX,ENVIRX)
          IF ( NODDS .GT. 1 ) FORCS2(SMALER)=FORCS2(SMALER)+
     &        CONFLO(2,RANGEX,ENVIRX)
          FORCS2(BIGGER)=CONFLO(1,RANGEX,ENVIRX)*ODDS(1)
          IF ( NODDS .GT. 1 ) FORCS2(BIGGER)=FORCS2(BIGGER)+
     &        CONFLO(2,RANGEX,ENVIRX)*ODDS(2)
CC *****
CC        WRITE(6,90000) RANGEX,CNFX,(FORCS1(I),FORCS2(I),NOTIN(I)
CC     &    ,ODDS(I),CONFLO(I,RANGEX,ENVIRX),I=1,2)
CC     &    ,SMALER,BIGGER,NOTPAR(1,ENVIRX,RANGEX)
CC     &    ,NOTPAR(2,ENVIRX,RANGEX)
CC 90000 FORMAT(1X,'1 CNFLC1',' FOR RANGE,CNFX= ',2(I2,1X),/,2(' FORCS1'
CC     &',FORCS2,NOTIN,ODDS,CONFLO= ',5(I8,2X),/),' SMALER= '
CC     & ,I1,1X,'BIGGER= ',I1,1X,'NOTPAR= ',2(I6,1X))
CC *****
C     SAVE THE NUMBER OF WEAPONS AT THE START OF THE DAY
          DAYST(1)=FORCS2(1)
          DAYST(2)=FORCS2(2)
          IF ( FORCS2(1)*FORCS2(2) .LE. 0 ) RETURN 1
      ENDIF
      NUMWPN(TYPS2X,1)=NUMWPN(TYPS2X,1)+
     &   FORCS2(1)*PROJCT(TYPS2X,4)
      NUMWPN(TYPS2X,2)=NUMWPN(TYPS2X,2)+
     &   FORCS2(2)*PROJCT(TYPS2X,4)
            DO 2000 I=1,NODDS
              DUELS(ENVIRX,RANGEX,CNFX,2*(I-1)+1)=ODDS(I)
              DUELS(ENVIRX,RANGEX,CNFX,2*I)=CONFLO(I,RANGEX,ENVIRX)
2000        CONTINUE
            DUELS(ENVIRX,RANGEX,CNFX,5)=BIGGER
C
C     REDUCE ODDS TO MAX ALLOWED IF NECESSARY
C
            DO 3000 ODDSX=1,NODDS
              ODDS(ODDSX)=MIN(IODDS,ODDS(ODDSX))
3000        CONTINUE
            RETURN
            END
```

Figure D-19. Source Listing of the CNFLC1 Subroutine of the Main Program of the AFP Combat Module

b. Updates the direct firer assignment counts. For each combination of side 1 and side 2 types, this is a cumulative tally of the number of weapons participating in conflicts; note that a weapon killed in its third conflict is counted three times.

c. Computes the odds and the number of duels at each odds.

(4) CNFTMS

(a) Purpose. To provide a time to fire to every weapon initially, and to provide a time to fire to every surviving weapon that has fired in the previous shot cycle. Figure D-20 is a source listing of subroutine CNFTMS.

(b) When Called. During every shot cycle, from MAIN.

(c) Subtasks

1. Computes an inflated detection size for multiple weapons.

2. Sets a flag to prevent generating a firing time for shooters with a zero PK.

3. Sets a flag to prevent generating a firing time for shooters who connot fire at the given range.

4. For direct fire weapons that have not fired previously in this conflict and have targets in range, uses a detection routine to compute a time to fire; if nondetection occurs, sets a flag. The detect routine is called in subsequent shot cycles for a flagged weapon until detection occurs, the weapon is killed, or the conflict ends.

5. For indirect fire weapons or for direct fire weapons that have just fired (and therefore have adequate range and have detected previously), a log normal distribution is used to generate times to fire. The mean of this distribution is given as input, and the standard deviation is assumed to be half the mean.

6. For direct fire duels where no detections occurred previously, when weapons on both sides achieve a first detection in the same shot cycle, the physically smaller weapon gets to fire first.

```
      SUBROUTINE CNFTMS(ENVIRX,RANGEX,CNFX,STAGEX)
C
C +++++ DENOTES CODE TO FORCE NUMERICALLY SMALLER FORCE TO
C         TO FIRE FIRST
C
C
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK2
      INCLUDE BK5
      INCLUDE BK6
      INCLUDE BK11
      INCLUDE BK17
      INCLUDE BK18
      INCLUDE BK22
C
      DIMENSION OUTPUT(2), BAND(IRANGE)
C
      INTEGER STAGEX,RANGEX,ENVIRX,CONFLX,POINT1,CNFX
C
      REAL MEAN,LIGHT1,SIZE1,MEAN1,SIGMA1
C
      LOGICAL SIGNAL,IFOUT,ISEXT
C
      DATA NUM/1/
C     RANGE BANDS (IN KILOMETERS):
      DATA BAND /.250, .500, 1.000, 1.500, 2.000, 2.500/
C
2     CONTINUE
      MODE=1
      NNRP=INSEED
C TO COMPUTE THE INITIAL/REFIRE TIMES
      DO 2500 CONFLX=1,CNFLCT
C LOOP THROUGH THE STATUS TABLE ENTRIES
C   SKIP IF COMBAT HAS CEASED FOR THIS DUEL
        IF ( STATUS(CONFLX,1) .LT. -SMALL2 ) GOTO 2500
        POINT1=IFIX(STATUS(CONFLX,2))
C   LOOP THROUGH WEAPONS IN THE CONFLICT
        DO 2000 I=0,POINT1
          IF ( I .EQ. 0 ) THEN
            J=SMALER
            K=TYPS(SMALER)
            NTGTS = POINT1
          ELSE IF (I .EQ. 1) THEN
              J=BIGGER
              K=TYPS(BIGGER)
              NTGTS = 1
          ENDIF
C         AT THIS POINT,
C              J = THE SHOOTING SIDE
C              K = THE SHOOTING WEAPON TYPE
C              NTGTS = THE NUMBER OF TARGETS
C         IS THE SHOOTER INDIRECT FIRE ?
          ISEXT = (TYPS(J) .GT. NTYPS(J)-NOEXT(J))
C   SKIP IF THE ENTRY IS ALREADY ELIMINATED
          IF ( STATUS(CONFLX,3+I) .LT. -SMALL2 )
     &                         GOTO 2000
C   IF PK IS ZERO, TREAT AS IF OUT OF RANGE
          IF ( .NOT. ISEXT .AND.
     &                         (PKS(J,TYPS(2),4,RANGEX) .LT. SMALL1) )
     &                          THEN
          STATUS(CONFLX,3+I)=RMAXT
CC *****
CC          WRITE(6,91800) RANGEX,STAGEX,PKS(J,TYPS(3-J),4,RANGEX),ISEXT
CC     & ,STATUS(CONFLX,3+I)
CC91800     FORMAT(1X,'CNFTMS 1 ','RANGEX ',I2,1X,'STAGEX= ',I2,1X
CC     &    ,' PKS= ',F8.6,1X,'ISEXT= ',L5,1X,'STATUS= ',F15.5)
CC *****
          GOTO 2000
        ENDIF
C   CHECK FOR FIRST STAGE OR PREVIOUS NON-DETECTION
        SIGNAL=STATUS(CONFLX,3+I) .GT. RMAXT/4.1
        SIGNAL=SIGNAL .AND.
     &                     ( STATUS(CONFLX,3+I) .LT. RMAXT/1.2 )
        SIGNAL=SIGNAL .OR. ( STAGEX .EQ. 1 )
        IF ( SIGNAL ) THEN
          IFOUT=(RANGEX.GT.LIMITS(J,K,2)) .OR.
     &                         (RANGEX.LT.LIMITS(J,K,1))
          IFOUT = IFOUT .AND. ( .NOT. ISEXT )
          IF ( IFOUT ) THEN
C OUT OF RANGE
```

Figure D-20. Source Listing of the CNFTMS Subroutine of the Main Program of the AFP Combat Module (page 1 of 3 pages)

```
                  STATUS(CONFLX,3+I)=RMAXT
                ELSE
CC  *****
CC                WRITE(6,91900) RANGEX,STAGEX,IFOUT,STATUS(CONFLX,3+I)
CC        &          ,SIGNAL
CC91900           FORMAT(1X,'CNFTMS 2',1X,'RANGEX= ',I2,1X,'STAGEX= ',I2
CC        & ,1X,'IFOUT= ',L5,1X,'STATUS= ',F15.5
CC        &   ,1X,'SIGNAL= ',L5)
CC  *****
C    CHECK TO SEE IF DETECTION SHOULD BE SKIPPED SINCE THE
C       WEAPON IN QUESTION IS EXTERNAL
                  IF ( .NOT. ISEXT ) THEN
C                   FORCE DETECTION IF FIRED AT OFTEN ENOUGH
                    IF (CSHOTS(3-J,CONFLX) .GT. NS2DCT(3-J,TYPS(3-J))) THEN
                      STATUS(CONFLX,3+I) = 0.01
                      GOTO 2000
                    ENDIF
C                   DON'T ATTEMPT DETECTION IF IT WOULD BE FRUITLESS
                    IF (STATUS(CONFLX,3+I) .EQ. RMAXT/4.0) GOTO 2000
C    L1 POINTS TO SENSOR USED BY SHOOTER, L2 TO THE SIGNATURE TYPE
C        FOR THAT SENSOR
                    L1=SENSOR(J,K,ENVIRX,1)
                    L2=SENSOR(J,K,ENVIRX,2)
                    SIZE1 = SIZE(3-J, TYPS(3-J), ENVIRX)
                    TBAR = RMAXT/2.0  @ SIGNIFIES NO DETECTION
                    DO 3000 IDTCT = 1,MIN(NDTCTN,NTGTS)
                      RANDOM=FIMSLU(INSEED)
                      RTM=BAND(RANGEX)
C !!!! OPTICS DESTROYS LIGHT1 EACH CALL !!!!
                      LIGHT1=LIGHT(ENVIRX)
                      CALL DETECT(RANDOM,L1,RTM
     &                             ,SIZE1
     &                             ,CNTRST(3-J,TYPS(3-J),ENVIRX,L2)
     &                             ,ATTN(L1,ENVIRX),MAG(L1)
     &                             ,BRIGHT(ENVIRX),LIGHT1,MODE
     &                             ,T,RC,NNRP,IDETCT)
                      IF (RC .LT. RCDET(J,K)) THEN
C                       CAN'T EVER DETECT (UNLESS SHOT AT ENOUGH)
                        TBAR = RMAXT/4.0
                        GOTO 3099
                      ENDIF
                      IF (IDETCT .EQ. 1) THEN
C                       SUCCESSFUL DETECTION
C                       CONVERT TIME TO DETECT TO MINUTES
                        T=T/60.
                        TBAR = MIN(TBAR,T)
                      ENDIF
3000                CONTINUE
3099                CONTINUE
C    UPDATE THE DETECT TRIES COUNT
                    TMS(J,ENVIRX,RANGEX,CNFX,1) =
     &                TMS(J,ENVIRX,RANGEX,CNFX,1) + 1.0
                    IF (TBAR .LT. RMAXT/4.1) THEN
C                       UPDATE THE DETECTION SUCCESS COUNT
                        TMS(J,ENVIRX,RANGEX,CNFX,2)=
     &                      TMS(J,ENVIRX,RANGEX,CNFX,2)+1.0
C                       INDICATE THE TIME IS DUE TO DETECT
                        TMS(J,ENVIRX,RANGEX,CNFX,3) = T +
     &                      TMS(J,ENVIRX,RANGEX,CNFX,3)
                    ENDIF
                    STATUS(CONFLX,3+I) = TBAR
CC  *****
CC                  WRITE(6,92000) RANGEX,STAGEX,L1,L2,RANDOM,RTM,LIGHT1
CC        &            ,SIZE1
CC        &      ,TBAR,STATUS(CONFLX,3+I),SIGNAL,ISEXT
CC92000             FORMAT(1X,'CNFTMS 3',1X,'RANGEX= ',I2,1X,'STAGEX= ',I2
CC        &   ,'  L1,L2,RANDOM,RTM,LIGHT1,SIZE1= ',2I5
CC        &     ,1X,4F10.3,/,1X,'TBAR= ',F10.3,1X,' STATUS= ',F15.3
CC        &   ,1X,'SIGNAL,ISEXT= ',2L5)
CC  *****
                  ELSE
C    USE REFIRE TIME FOR INDIRECT WEAPONS
                    MEAN1=REFIRE(J,K,TYPS(3-J),ENVIRX)
                    SIGMA1=MEAN1/2.0
                    MEAN=ALOG(.8*MEAN1*MEAN1)/2.0
C    USE 0.4723807 SINCE THAT IS THE SQRT(LN(1.25))
                    SIGMA=0.4723807
                    CALL GGNLG(DSEED,NUM,MEAN,SIGMA,OUTPUT)
                    STATUS(CONFLX,3+I)=OUTPUT(1)
C    UPDATE THE REFIRE TIME COUNT
                    TMS(J,ENVIRX,RANGEX,CNFX,4) =
     &                TMS(J,ENVIRX,RANGEX,CNFX,4) + 1.0
```

Figure D-20. Source Listing of the CNFTMS Subroutine of the Main Program of the AFP Combat Module (page 2 of 3 pages)

```
C    INDICATE THE TIME IS DUE TO REFIRE
                   TMS(J,ENVIRX,RANGEX,CNFX,5) = STATUS(CONFLX,3+I) +
      &                TMS(J,ENVIRX,RANGEX,CNFX,5)
CC *****
CC                   WRITE(6,93000) RANGEX,STAGEX,STATUS(CONFLX,3+I)
CC93000              FORMAT(1X,'CNFTMS 4',1X,'RANGE= ',I2,1X,'STAGE= ',I2,1X
CC        &   ,'STATUS= ',F15.5)
CC *****
C ENDIF FOR DIRECT/INDIRECT WEAPON
                  ENDIF
C    ENDIF FOR IN RANGE OR INDIRECT/OUT OF RANGE IF
              ENDIF
C    ELSE FOR " IF ( SIGNAL) "
           ELSE
C    GENERATE REFIRE TIMES IF IN RANGE AND FIRED LAST ROUND
              IF ( STATUS(CONFLX,3+I) .GT. RMAXT/1.2)
      &                          GOTO 2000
C USE THE REFIRE TIMES
              IF ( STATUS(CONFLX,3+I) .LT. SMALL1 ) THEN
C ENTRY HAD FIRED IN THE LAST ROUND, RECOMPUTE REFIRE TIMES
                  MEAN1=REFIRE(J,K,TYPS(3-J),ENVIRX)
                  SIGMA1=MEAN1/2.0
                  MEAN=ALOG(.8*MEAN1*MEAN1)/2.0
C    USE 0.4723807 SINCE THAT IS THE SQRT(LN(1.25))
                  SIGMA=0.4723807
                  CALL GGNLG(DSEED,NUM,MEAN,SIGMA,OUTPUT)
                  STATUS(CONFLX,3+I)=OUTPUT(1)
C    UPDATE THE REFIRE COUNT
                  TMS(J,ENVIRX,RANGEX,CNFX,4) =
      &              TMS(J,ENVIRX,RANGEX,CNFX,4) + 1.0
C    INDICATE THE TIME IS DUE TO REFIRE
                  TMS(J,ENVIRX,RANGEX,CNFX,5) = STATUS(CONFLX,3+I) +
      &                TMS(J,ENVIRX,RANGEX,CNFX,5)
              ENDIF
C    END OF IF ON SIGNAL
           ENDIF
2000   CONTINUE
C END OF CONFLX LOOP TO PROVIDE TIMES
2500   CONTINUE
CC *****
CC         IF ( CNFLCT .GT. 0 ) THEN
CC         WRITE(6,90000) RANGEX,STAGEX,CNFLCT
CC       & ,REFIRE(1,TYPS(1),TYPS(2),ENVIRX),REFIRE(2,TYPS(2),TYPS(1)
CC       & ,ENVIRX),((STATUS(IO,JO),JO=1,20),IO=1,CNFLCT)
CC90000 FORMAT(1X,'CNFTMS 5',1X,'RANGEX= ',I2,1X,'STAGEX= ',I2,1X
CC       & ,'CNFLCT= ',I3,1X,'REFIRE= ',2(F8.4,2X)
CC       &    ,'STATUS= ',500(/,' *',10(F10.4,2X),/,4X,10(F10.4,2X),/) )
CC         ENDIF
CC *****
       RETURN
       END
```

Figure D-20. Source Listing of the CNFTMS Subroutine of the Main Program of the AFP Combat Module
(page 3 of 3 pages)

(5) DETECT

(a) **Purpose.** When called, to return whether a weapon has detected a target and, if so, the elapsed time increment. Figure D-21 is a source listing of subroutine DETECT.

(b) **Source.** DETECT is an implementation of logic developed at the Night Vision Laboratory.

(c) **When Called.** During every shot cycle for surviving weapons that have not previously detected a target or otherwise begun to fire.

Note: As implemented here, detection is deterministic but dependent on the routine's call arguments. Only elapsed time is a stochastic variable.

(6) DIRIO

(a) **Purpose.** To initialze/read in the cumulative kills and cumulative direct fire assignments, and to read in the relevant data from the projection file (participation rates, conflict durations, number of conflicts/day, number of sites at which simultaneous identical conflicts occur). Figure D-22 is a source listing of subroutine DIRIO.

(b) **When Called.** Every time a conflict is being processed which includes a side 1 type that has not been processed earlier.

(7) DIRKLS

(a) **Purpose.** To compute the direct fire kills in each conflict. Figure D-23 is a source listing of subroutine DIRKLS.

(b) **When Called.** Called once during each shot cycle, by MAIN.

(c) **Process.** DIRKLS examines every duel in each shot cycle. In each duel, the weapon(s) with the shortest time to fire (if any are eligble to fire) is made to fire, and the kill(s) are assesed. Note that all duels are processed simultaneously in every call to this routine. Duels whose expended time exceeds the conflict's duration are prevented from firing again. The conflict ends when the estimated number of shot cycles is processed.

(d) **Subtasks**

1. Locates the shortest time to fire among eligible shooters in each duel, and subtracts this time from every eligible shooter's time to fire.

2. Updates the duel's elapsed time counter, and sets flag to indicate that no further firing should occur in the duel if the elapsed time exceeds the conflict's maximum permitted duration (an input).

```
      SUBROUTINE DETECT (RN,LSCC,RTM,CD,ACON,ATTN,AMAG,SOG,AL,MODE,TBAR,
     + RC,NNRP,IDETCT)
C*****                    CALLED BY -TGTACQ-
C*****INPUTS VARIABLES  FOR NVL SENSOR LOGIC
C*****
C       LSCC = SENSOR DEVICE
C       RTM  = RANGE TO TARGET IN KM
C       CD= TARGET MINIMUM DIMENSION IN METERS
C       AL   = LIGHT LEVEL IN FT CDLS
C       ACON = OPTICAL IS CONTRAST, THERMO IS TEMPERATURE DIFFERENCE
C       ATTN = EXTINCTION COEFFICIENT ARRAY
C       AMAG = MAGNIFICATION ARRAY
C       FOV  = FIELD OF VIEW ARRAY
C       SOG = SKY OVER GROUND BRIGHTNESS RATIO
C*****
C-    TBAR  = TIME TO ACHIEVE DETECTION
C-    RC = RESOLVABLE CYCLES
C-    IDETCT = 1 ON SUCCESSFUL DETECTION, 0 OTHERWISE
C
      DOUBLE PRECISION  E1
C                                                CHECK SENSOR DEVICE
C   1 = OPTICAL  - EYES
C   2 = OPTICAL  - BINOCULARS
C   3 = IMAGE INTENSFIER - STARLIGHT
C   4 =                  - RED SIGHT
C   5 =                  - CREW SIGHT
C   6 = THERMO
C   7 = THERMO
C   8 = THERMO
C   9 = THERMO - AIRBORNE FLIR
C  10 = OPTICAL- TV
C  11 = THERMO - NIGHT
C  12 = OPTICAL - TV
C  13 = RADAR
C  14 = THERMO
C  15 = OPTICAL
C  16 = OPTICAL
C*****
2     CONTINUE
      IDETCT = 0
      RC=0.0
      S=0.0
      TBAR=0.
      PINF=0.0
      AL1=AL
C
      IF (AMAG.LE.0.) AMAG=1.0
C
      IF (ATTN.LE.0.0) GO TO 190
C
C-    OPTICAL VISIBLE BANDS (1,2,10,12,15,16,19-50)
C
C-    IMAGE INTENSIFIERS (3,4,5)
C
C-      THERMO DEVICES (6,7,8,9,11,14,17,18)
C
      IF (LSCC.LE.0.OR.LSCC.GT.50) GO TO 190
C
      GO TO (100,100,110,110,110,120,120,120,120,100,120,100,130,120,100
     +,100,120,120,100,100,100,100,100,100,100
     +,100,100,100,100,100,100,100,100,100,100,100,100,100
     +,100,100,100,100,100,100,100,100,100,100,100,100),LSCC
C
C                 -                               VISIBLE BAND DEVISES
  100 CALL OPTICS (RC,ACON,ATTN,RTM,LSCC,SOG,AL1,MODE,AMAG)
      GO TO 140
C                 -                               IMAGE INTENSIFIERS
  110 CALL IMAGES (RC,ACON,ATTN,RTM,LSCC,SOG,AL1)
      GO TO 140
C                 -                                THERMO DEVICES
  120 CALL THERMO (RC,ACON,ATTN,RTM,LSCC,MODE)
      GO TO 140
C                 -                                RADAR DEVICES
  130 CONTINUE
      RC=0.
      IF(FIMSLU(NNRP) .LT. .80) THEN
        RC = 1.0
        IDETCT = 1
      ENDIF
C
      TBAR = 3.0 + FIMSLU(NNRP) * 12.0
```

Figure D-21. Source Listing of the DETECT Subroutine of the Main Program of the AFP Combat Module
(page 1 of 2 pages)

```
          IF(RC .LE. 0.) PRINT 333
  333     FORMAT(55X,'RADAR LESS THAN 80%')
C
      GO TO 190
C     -
  140 CONTINUE                        CONVERSION FROM ... TO RESOLVABLE
        IF(RC .LE. 0.) GO TO 190
C
C     -       CRTITICAL DIMENSION/RANGE PER THOUSAND METERS
      S=CD/RTM
C     -       RESOLVABLE CYCLES ACROSS TARGET
      RC=RC*S
C     -       IF LESS THAN A TENTH OF A CYCLE - FORGET IT
      IF (RC.LT.0.10) GO TO 190
C
C     -       FIND PROBABILITIES OF DETECTION AND DESIRED LEVEL OF
      AA=1.0
      RC1 = RC / AA
C
C
      IF ( RC1 .LE. 4) GO TO 150
C
      PINF = 1.0
      GO TO 160
C
C           PINF = (RC1)**E / ( 1 + (RC1)**E )
C
  150 E1 = 2.7 + 0.7 * RC1
C
      EX= RC1 ** E1
C
      PINF = EX / (1 + EX)
C
  160 CONTINUE
C
C                                     TGT NOT DETECTED
      IF (RN.GE.PINF) GO TO 190
C
      TBAR = (-3.6 / PINF) * LOG(1.0 - RN/PINF)
C
      IF (TBAR .LT. 30) IDETCT = 1
C
  190 CONTINUE
      RETURN
      END
```

Figure D-21. Source Listing of the DETECT Subroutine of the Main Program of the AFP Combat Module (page 2 of 2 pages)

```
      SUBROUTINE DIRIO(TYPS1X,REPSX,DAYSX)
C
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK10
      INCLUDE BK13
      INCLUDE BK16
      INCLUDE BK21
C
      INTEGER TYPS1X,REPSX,DAYSX,SIDEX
C
2     CONTINUE
            IOPTR3=(REPSX-1)*N31+TYPS1X
            IOPTR3=IOPTR3+PREF4*IREPS*ITYPS1
            IOPTR4=(REPSX-1)*N41+TYPS1X
            IOPTR4=IOPTR4+PREF2*IREPS*ITYPS1
            IOPTR6=TYPS1X
            IOPTR6=IOPTR6+PREF3*ITYPS1
C  TO REINITIALIZE AT THE BEGINNING OF EACH DAY ONLY
            IF ( DAYSX .EQ. 1 ) THEN
C  INITIALIZE CKILLS
              DO 1025 SIDEX=1,2
              DO 1025 I=1,NTYPS(2)
                NUMWPN(I,SIDEX)=0
              DO 1025 J=1,7
                CKILLS(SIDEX,I,J)=0
1025          CONTINUE
            ELSE
C  READ IN THE CUMULATIVE KILLS TO DATE
              READ(IU3'IOPTR3,1030,ERR=1035)
     &          (((CKILLS(I,J,K),K=1,7),J=1,NTYPS(2)),I=1,2)
1030          FORMAT(500(500I6))
              GOTO 1040
1035          WRITE(6,1037)
1037          FORMAT(1X,30(1H*),5X,'CKILLS READ ERROR',5X,
     &          30(1H*))
              STOP
1040          CONTINUE
              READ(IU4'IOPTR4,1030,ERR=1045)
     &          ((NUMWPN(I,J),J=1,2),I=1,NTYPS(2))
              GOTO 1047
1045          WRITE(6,1046)
1046          FORMAT(1X,30(1H*),5X,'NUMWPN READ ERROR',5X,30(1H*))
              STOP
1047          CONTINUE
            ENDIF
C    READ IN THE DATA USED TO PROJECT LOSSES
            READ(IU6'IOPTR6,1048,ERR=1043) ((PROJCT(I,J),J=1,5),
     &        I=1,NTYPS(2))
1048        FORMAT(500(500F6.2))
            GOTO 1049
1043        WRITE(6,1044)
1044        FORMAT(1X,30(1H*),5X,'PROJCT READ ERROR',5X,30(1H*))
            STOP
1049        CONTINUE
          RETURN
          END
```

Figure D-22. Source Listing of the DIRIO Subroutine of the Main Program of the AFP Combat Module

```
          SUBROUTINE DIRKLS(TYPS2X,ENVIRX,RANGEX,CNFX,STAGEX)
C
          INCLUDE PARM
          INCLUDE BK1
          INCLUDE BK2
          INCLUDE BK6
          INCLUDE BK9
          INCLUDE BK10
          INCLUDE BK11
          INCLUDE BK15
          INCLUDE BK17
          INCLUDE BK16
          INCLUDE BK18
          INCLUDE BK19
          INCLUDE BK21
          INCLUDE BK23
          INCLUDE BK24
          INCLUDE BK25
          INCLUDE BK27
C
          DIMENSION DONE(3),KILLS(IODDS+1),DURATN(ICONFL)
C
          LOGICAL DONE,SIGNAL,DWPN,COND1,COND2
C
          INTEGER RANGEX,TYPS2X,POINT1,CONFLX,SIDE,OTHER,PEXT,CNFX,STAGEX,
         &     ENVIRX
C
2         CONTINUE
CC *****
CD           IF ( CNFLCT .GT. 0 ) THEN
CD           WRITE(6,90000) TYPS2X,RANGEX,CNFX,CNFLCT
CD        &    ,((STATUS(IO,JO),JO=1,20),IO=1,CNFLCT)
CD90000 FORMAT(1X,'DIRKLS 1 TYPS2X,RANGEX,CNFX,CNFLCT= ',4(I3,2X)
CD        &    ,' STATUS= ',500(/,' *',10(F10.4,2X),/,4X,10(F10.4,2X),/) )
CD           ENDIF
CC *****
C    INITIALIZE THE DUEL DURATION TABLE
                        IF ( STAGEX .EQ. 1 ) THEN
                          DO 2000 CONFLX=1,CNFLCT
                            DURATN(CONFLX)=0.0
2000                      CONTINUE
                        ENDIF
C LOOP TO COMPUTE KILLS DUE TO DIRECT FIRE
                        DO 5000 CONFLX=1,CNFLCT
C    SKIP IF EXTERNAL SHOOTERS ELIMINATED THE DUEL
                          IF ( STATUS(CONFLX,1) .LT. 0.0 ) GOTO 5000
                          POINT1=IFIX(STATUS(CONFLX,2))
C    CHECK THAT NEITHER SIDE IN DUEL IS DEPLETED
                          COND1=STATUS(CONFLX,3) .GT. -SMALL2
                          DO 2100 J=1,POINT1
                            COND2=COND1 .AND.
         &                            ( STATUS(CONFLX,3+J) .GT. -SMALL2 )
                            IF ( COND2 ) GOTO 2200
2100                      CONTINUE
                          STATUS(CONFLX,1)=-1.0
                          GOTO 5000
2200                      CONTINUE
                          ITEMP=-1
                          TEMP=RMAXT/16.0
C SEARCH FOR THE SHORTEST TIME TO FIRE
                          DO 2700 I=0,POINT1
                            IF ( ( STATUS(CONFLX,3+I).LT.TEMP ) .AND.
         &                       ( STATUS(CONFLX,3+I) .GE. 0.0 ) ) THEN
C FOUND A NEW CANDIDATE
                              ITEMP=I
                              TEMP=STATUS(CONFLX,3+I)
                            ENDIF
2700                      CONTINUE
C CHECK FOR NO SHOOTERS IN RANGE
                          IF ( ITEMP .EQ. -1 ) THEN
                            GOTO 5000
                          ENDIF
C SUBTRACT OFF THE SHORTEST TIME TO FIRE
                          DO 2800 I=0,POINT1
                            IF ( (STATUS(CONFLX,3+I).LT. RMAXT/16.0 ) .AND.
         &                       ( STATUS(CONFLX,3+I) .GT. 0.0 ) ) THEN
C SUBTRACT OFF THE TIME THE NEXT FIRING OCCURS AT
                              STATUS(CONFLX,3+I)=STATUS(CONFLX,3+I)-TEMP
C SET ENTRIES CLOSE TO 0.0 TO 0.0
                              IF ( ABS(STATUS(CONFLX,3+I)).LT.SMALL1 )
         &                      STATUS(CONFLX,3+I)=0.0
```

Figure D-23. Source Listing of the DIRKLS Subroutine of the Main Program of the AFP Combat Module (page 1 of 4 pages)

```
                           ENDIF
 2800                   CONTINUE
 C    UPDATE THE DUEL'S DURATION
                        DURATN(CONFLX)=DURATN(CONFLX)+TEMP
                        IF ( DURATN(CONFLX) .GT. PROJCT(TYPS2X,3) ) THEN
 C    TIME IN DUAL AT MAX, SET UP TO DENY FURTHER COMBAT
                          STATUS(CONFLX,1)=-1.0
 C    GO TO THE NEXT DUEL
                          GOTO 5000
                        ENDIF
                        ITEMP=0
                        DONE(1)=.FALSE.
                        DO 2850 I=0,POINT1
 C    INDICATE WHICH SIDE SHOOTS FIRST/BOTH
                          IF ( ABS(STATUS(CONFLX,3+I)) .LT. SMALL1 ) THEN
                            IF ( I .EQ. 0 ) THEN
                              ITEMP=SMALER
                            ELSE
                              IF ( .NOT. DONE(1) ) ITEMP=ITEMP+BIGGER
                              DONE(1)=.TRUE.
                            ENDIF
                          ENDIF
 2850                   CONTINUE
 C  1 IF SIDE 1 SHOOTS FIRST, 2 IF SIDE 2, 3 IF BOTH
                        STATUS(CONFLX,1)=ITEMP+.01
 C    DETERMINE THE KILLS
                        SIGNAL=.TRUE.
                        DO 3000 J=0,POINT1
                          KILLS(J+1)=0
                          IF ( ABS(STATUS(CONFLX,3+J)) .LT. SMALL1 ) THEN
 C    WE HAVE A SHOOTER
 C    RECORD THE SHOTS
                            ITEMP1=BIGGER
                            IF ( J .EQ. 0 ) ITEMP1=SMALER
                SHOTS(ITEMP1,ENVIRX,RANGEX)=
      &         SHOTS(ITEMP1,ENVIRX,RANGEX)
      &           + PROJCT(TYPS2X,4) + SMALL2
                CSHOTS(ITEMP1,CONFLX)=
      &         CSHOTS(ITEMP1,CONFLX)
      &           + PROJCT(TYPS2X,4) + SMALL2
 C     OMIT REPETITIVE PROCESSING
                            IF ( SIGNAL ) THEN
                              IF ( J .GE. 1 ) SIGNAL=.FALSE.
                              IF ( J .EQ. 0 ) THEN
                                SIDE=SMALER
                              ELSE
                                SIDE=BIGGER
                              ENDIF
                              OTHER=3-SIDE
 C    CHECKING FOR DIRECT FIRE SHOOTER
                              DWPN=TYPS(SIDE).LE.(NTYPS(SIDE)-NOEXT(SIDE))
                              IF ( DWPN ) THEN
                                DO 2875 I=1,IPKS
 C    LOOP THROUGH M, F, K, M OR F KILLS
                                  ATEMP(1,I)=
      &                             PKS(SIDE,TYPS(2),I,RANGEX)
 2875                           CONTINUE
 C    COMPUTE BREAKDOWN OF [0,1] INTO DISTINCT INTERVALS
 C     PROPORTIONAL TO THE M,F,K PROBABILITIES
 C    PROBABILITY OF NO KILL
                                ATEMP(2,1)=1.0-ATEMP(1,4)
 C OBVIATE REST OF ATEMP
                                ATEMP(2,2)=2.0
                                ATEMP(2,3)=2.0
                                ATEMP(2,4)=1.0
 C    NORMALIZE
                                ATEMP(3,1)=ATEMP(2,1)/ATEMP(2,4)
                                ATEMP(3,2)=ATEMP(2,2)/ATEMP(2,4)
                                ATEMP(3,3)=ATEMP(2,3)/ATEMP(2,4)
                              ELSE
 C  AN EXTERNAL WEAPON IS SHOOTING IN A DIRECT FIRE DUEL
 C  DERIVE THE EXTERNAL WEAPON NUMBER
                                PEXT=TYPS(SIDE)-NTYPS(SIDE)+NOEXT(SIDE)
 C  COMPUTE THE EXPECTED KILLS
                                ATEMP(3,3)=
      &                           EXT(SIDE,PEXT,TYPS(OTHER),ENVIRX,RANGEX)
                                ATEMP(3,3)=AMIN1(1.0,ATEMP(3,3))
                                ATEMP(3,3)=1.0-ATEMP(3,3)
                                ATEMP(3,1)=2.0
                                ATEMP(3,2)=2.0
                              ENDIF
                            ENDIF
```

Figure D-23. Source Listing of the DIRKLS Subroutine of the Main Program of the AFP Combat Module (page 2 of 4 pages)

```
C    GENERATE A RANDOM NUMBER TO FIND THE TYPE OF KILL
                                 TEMP=FIMSLU(INSEED)
                                 IF ( TEMP .GE. ATEMP(3,3) ) THEN
                                   KILLS(J+1)=3
                                 ELSE
                                   IF ( TEMP .GE. ATEMP(3,2) ) THEN
                                     KILLS(J+1)=2
                                   ELSE
                                     IF ( TEMP .GE. ATEMP(3,1) ) KILLS(J+1)=1
                                   ENDIF
                                 ENDIF
C    END OF SHOOTER IF
                               ENDIF
C   END OF LOOP THROUGH COORDINATES OF STATUS ENTRY
3000                         CONTINUE
C    FOR THE SIDE WITH MULTIPLE WEAPONS, DETERMINE THE TOTAL
C     DAMAGE DONE TO THE SINGLE WEAPON
                             KILLS1=0
                             DO 3200 I=1,3
                               DONE(I)=.FALSE.
C    LOOPING THROUGH THE DAMAGE CATEGORIES
                               DO 3100 J=1,POINT1
C    CHECK TO SEE IF THIS DAMAGE CATEGORY HAS ALREADY OCCURRED
                                 IF ( DONE(I) ) GOTO 3200
                                 IF ( KILLS(J+1) .EQ. I ) THEN
C  MAKE SURE THIS CATEGORY IS NOT EXAMINED AGAIN
                                   DONE(I)=.TRUE.
C    ACHIEVED A KILL OF TYPE TESTED FOR IN THIS PASS
                                   KILLS1=KILLS1+2**(I-1)
C    CHECK THE NEXT DAMAGE CATEGORY, SINCE THIS ONE IS DONE
                                   GOTO 3200
C    NOW KILLS 1 CONTAINS THE APPROPRIATE CODE FROM 1 TO 7
                                 ENDIF
3100                           CONTINUE
3200                         CONTINUE
C    RECORD THE KILLS
                             IF ( KILLS1 .GT. 0 ) THEN
C    THE WEAPON ON THE SMALLER SIDE IS M,F,K KILLED
C      ( OR SOME COMBINATION THEREOF )
C    MARK STATUS ENTRY FOR DELETION
                               STATUS(CONFLX,1)=-1.0
CC *****
CD        IF ( CNFLCT .GT. 0 ) THEN
CD        WRITE(6,90100) CONFLX
CD90100  FORMAT(' DIRKLS 1.5',3X,'HIT SMALLER SIDE, DUEL '
CD     &      ,I5)
CD        ENDIF
CC *****
C    SKIP PROCESSING IF ALREADY DEAD
                               IF ( STATUS(CONFLX,3) .LT. 0.0 ) GOTO 3250
C    SET ENTRY TO SHOW KILL
                               STATUS(CONFLX,3)=-10.0*ITYPS
C    RECORD THE KILL
                               CKILLS(BIGGER,TYPS2X,KILLS1)=
     &                          CKILLS(BIGGER,TYPS2X,KILLS1)
     &                           +PROJCT(TYPS2X,4)
CC *****
CD        IF ( CNFLCT .GT. 0 ) THEN
CD        WRITE(6,94000) TYPS2X,RANGEX,CNFX,BIGGER
CD     &   ,CKILLS(BIGGER,TYPS2X,KILLS1),TEMP,ATEMP(3,1),ATEMP(3,3)
CD     &   ,INSEED,CONFLX
CD94000  FORMAT(1X,'*DIRKLS 2 TYPS2X,RANGEX,CNFX,BIGGER= ',4I5,1X
CD     & ,'CKILLS= ',I8,/,' RANDOM= ',F8.6,3X,'1-PKS= ',2(F8.6,3X)
CD     &  ,'INSEED= ',I6,3X,'DUEL ',I5,' ENTRY 3')
CD        ENDIF
CC *****
                               TTLOS(SMALER,1,ENVIRX,RANGEX) =
     &                           TTLOS(SMALER,1,ENVIRX,RANGEX)
     &                           + PROJCT(TYPS2X,4)
C    STORE THE KILLS IN THE LOG TABLE
                               LOG(CNFX,SMALER,TYPS(BIGGER),2+KILLS1)=
     &                           LOG(CNFX,SMALER,TYPS(BIGGER),2+KILLS1)+
     &                           PROJCT(TYPS2X,4)
                               FORCS1(SMALER)=MAX(
     &                          0,FORCS1(SMALER)-1)
                               FORCS2(SMALER)=MAX(
     &                          0,FORCS2(SMALER)-1)
                               WPNS(SMALER,TYPS(SMALER))=MAX(0,WPNS(SMALER,
     &                           TYPS(SMALER))-1)
                             ENDIF
C    SKIP HERE IF TARGET IS ALREADY DEAD
3250                         CONTINUE
```

Figure D-23. Source Listing of the DIRKLS Subroutine of the Main Program of the AFP Combat Module (page 3 of 4 pages)

```
C    CHECK TO SEE IF SMALLER SIDE KILLED SOMETHING
                               IF ( KILLS(1) .GT. 0 ) THEN
C    SMALLER SIDE KILLED SOMEONE. RECORD THIS KILL IF NOT ALREADY DEAD
                                 IF ( POINT1 .EQ. 1 ) THEN
C    END OF THIS DUEL, KILLED SOLE WEAPON ON A SIDE
C      MARK STATUS ENTRY FOR DELETION
                                  STATUS(CONFLX,1)=-1.0
C    SET ITEMP FOR DIRKLS 4 DEBUG PRINTOUT
                                  ITEMP=1
CC *****
CD       IF ( CNFLCT .GT. 0 ) THEN
CD       WRITE(6,94010) CONFLX
CD94010 FORMAT(' DIRKLS 2.2',3X,'HIT BIGGER SIDE ,DUEL '
CD     &       ,I5,3X,'ENTRY 4')
CD       ENDIF
CC *****
C    SKIP IF TARGET IS ALREADY DEAD
                                  IF ( STATUS(CONFLX,4) .LT. 0.0 )
     &                                GOTO 3300
C    SET STATUS ENTRY NEGATIVE TO DENOTE KILL
                                  STATUS(CONFLX,4)=-10.0*ITYPS
C    TALLY THIS WIN
                                 ELSE
C    SIDE HAS MORE WEAPONS, DETERMINE WHICH WAS KILLED
                                  DO 1000 L = 1,POINT1
                                    IF (STATUS(CONFLX,3+L) .GE. 0.0) THEN
                                      ITEMP = L
                                      GOTO 1099
                                    ENDIF
1000                              CONTINUE
1099                              CONTINUE
CC *****
CD       IF ( CNFLCT .GT. 0 ) THEN
CD       WRITE(6,94020) CONFLX,3+ITEMP
CD94020 FORMAT(' DIRKLS 2.6',3X,'HIT BIGGER SIDE, DUEL '
CD     &       ,I5,3X,'ENTRY ',I3)
CD       ENDIF
CC *****
C    SKIP IF TARGET IS ALREADY DEAD
                                  IF ( STATUS(CONFLX,3+ITEMP) .LT. 0.0 )
     &                                GOTO 3300
C    ITEMP CONTAINS POINTER TO THE KILLED ENTRY
C    SET KILLED ENTRIES VERY NEGATIVE
                                  STATUS(CONFLX,3+ITEMP)=-10.0*ITYPS
C    END OF IF ( POINT1 .EQ. 1 )
                                 ENDIF
                                 ITEMP1=2**(KILLS(1)-1)
                                 CKILLS(SMALER,TYPS2X,ITEMP1)=
     &                                CKILLS(SMALER,TYPS2X,ITEMP1)+
     &                                PROJCT(TYPS2X,4)
CC *****
CD       IF ( CNFLCT .GT. 0 ) THEN
CD       WRITE(6,94100) TYPS2X,RANGEX,CNFX,SMALER
CD     &   ,CKILLS(SMALER,TYPS2X,ITEMP1),TEMP,ATEMP(3,1),ATEMP(3,3)
CD     &     ,INSEED,CONFLX,3+ITEMP
CD94100 FORMAT(1X,'*DIRKLS 3 TYPS2X,RANGEX,CNFX,SMALER= ',4I5,1X
CD     & ,'CKILLS= ',I8,/,' RANDOM= ',F8.6,3X,'1-PKS= ',2(F8.6,3X)
CD     &   ,'INSEED= ',I6,3X,'DUEL ',I5,' ENTRY ',I3)
CD       ENDIF
CC *****
                                 TTLOS(BIGGER,1,ENVIRX,RANGEX) =
     &                              TTLOS(BIGGER,1,ENVIRX,RANGEX)
     &                              + PROJCT(TYPS2X,4)
C    RECORD THE KILLS
                                 LOG(CNFX,BIGGER,TYPS(SMALER),2+ITEMP1)=
     &                             LOG(CNFX,BIGGER,TYPS(SMALER),2+ITEMP1)+
     &                             PROJCT(TYPS2X,4)
                                 FORCS1(BIGGER)=MAX(
     &                             FORCS1(BIGGER)-1,0)
                                 FORCS2(BIGGER)=MAX(
     &                             FORCS2(BIGGER)-1,0)
                                 WPNS(BIGGER,TYPS(BIGGER))=MAX(0,WPNS(BIGGER,
     &                                TYPS(BIGGER))-1)
C    END OF IF ( KILLS(1) .GT. 0 )
                                ENDIF
C    SKIP HERE IF TARGET ALREADY DEAD
3300                            CONTINUE
C    CHECK THAT BOTH SIDES SURVIVE I.E. THAT FURTHER COMBAT IS
C       POSSIBLE
                                COND1=STATUS(CONFLX,3) .GT. -SMALL2
                                DO 3400 I=1,POINT1
                                  COND2=COND1 .AND.
     &                                  (STATUS(CONFLX,3+I) .GT. -SMALL2 )
                                  IF ( COND2 ) GOTO 3500
3400                            CONTINUE
                                STATUS(CONFLX,1)=-1.0
3500                            CONTINUE
C    END OF CONFLX LOOP FOR LOSSES
5000                          CONTINUE
CC *****
CD       IF ( CNFLCT .GT. 0 ) THEN
CD       WRITE(6,91000) CNFLCT,((STATUS(IO,JO),JO=1,20),IO=1,CNFLCT)
CD91000 FORMAT(1X,'DIRKLS 4',1X,'CNFLCT= ',I3,1X,'STATUS= '
CD     &       ,500(/,' *',10(F10.4,2X),/,4X,10(F10.4,2X),/) )
CD       ENDIF
CC *****
                        RETURN
                        END
```

Figure D-23. Source Listing of the DIRKLS Subroutine of the Main Program of the AFP Combat Module (page 4 of 4 pages)

3. Records the side firing in each duel.

4. For each duel, determines whether a kill occurs by comparing the PK to a random number, and computes the type of kill (currently, only one kill type is used).

5. Record the kills.

(8) EXTASG

(a) Purpose. To compute the number of indirect firers assigned to each range and environment. Figure D-24 is a source listing of subroutine EXTASG.

(b) When Called. Every time a new pair of opposing direct fire types is to be processed, called by MAIN.

(c) Subtasks

1. The fraction of each indirect firer type allocated to each range and environment is the product of the following factors:

a. The input datum that is the product of the preference and participation factors for the indirect shooter.

b. The fraction of indirect shooters used for area fire.

c. The fraction of direct fire conflicts occurring at the given range.

d. The fraction of direct fire conflicts occurring at the given environment.

2. This fraction is multiplied by the initial daily inventory of the indirect firer to obtain a preliminary integer assignment.

3. The unassigned fractional weapons are allocated by a method similar to the one used in FRCALC.

```
      SUBROUTINE EXTASG
C
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK2
      INCLUDE BK6
      INCLUDE BK9
      INCLUDE BK13
      INCLUDE BK16
      INCLUDE BK17
      INCLUDE BK20
      INCLUDE BK22
      INCLUDE BK24
C
      DIMENSION REMS(IENVIR,IRANGE)
C
      INTEGER SIDEX,EXTX,RANGEX,ENVIRX
      INTEGER COUNT
C
2     CONTINUE
      DO 4000 SIDEX=1,2
        IOTHER=3-SIDEX
C   LOOP OVER THE EXTERNAL WEAPON TYPES (ASSUMED TO BE THE
C        LAST TYPES )!!!!
C   SKIP IF NO EXTERNAL WEAPONS PRESENT
        IF ( (SIDEX .EQ. 1) .AND. (IEXT1 .EQ. 0) )
     &     GOTO 4000
        IF ( (SIDEX .EQ. 2) .AND. (IEXT2 .EQ. 0) )
     &     GOTO 4000
        DO 4000 EXTX=1,NOEXT(SIDEX)
         COUNT=0
         IPTR=NTYPS(SIDEX)-NOEXT(SIDEX)+EXTX
         DO 2525 RANGEX=1,IRANGE
         DO 2525 ENVIRX=1,NENVIR
C   CREATE POINTER TO THE TYPE NUMBER OF THE EXTERNAL WEAPON
C   COMPUTE THE NUMBER OF EXTERNAL SHOOTERS
            TEMP=WPNS1(SIDEX,IPTR)*INDDST(SIDEX,EXTX,RANGEX)
     &              *EXTPER(SIDEX,EXTX)*ENVDST(ENVIRX)
C   APPEND TO THE EXTERNAL ASSIGNMENTS ARRAY
            EXTN1(SIDEX,EXTX,ENVIRX,RANGEX)=TEMP
            REMS(ENVIRX,RANGEX)=TEMP-EXTN1(SIDEX,EXTX,ENVIRX,RANGEX)
            COUNT=COUNT+
     &          EXTN1(SIDEX,EXTX,ENVIRX,RANGEX)
2525     CONTINUE
         COUNT=WPNS1(SIDEX,IPTR)
     &           *EXTPER(SIDEX,EXTX)-COUNT
         DO 3000 I=1,COUNT
            IPTRR=1
            IPTRE=1
            DO 3500 ENVIRX=1,NENVIR
            DO 3500 RANGEX=1,IRANGE
              IF ( REMS(ENVIRX,RANGEX) .GT.
     &              REMS(IPTRE,IPTRR) ) THEN
                 IPTRE=ENVIRX
                 IPTRR=RANGEX
              ENDIF
3500        CONTINUE
         EXTN1(SIDEX,EXTX,IPTRE,IPTRR)=
     &      EXTN1(SIDEX,EXTX,IPTRE,IPTRR)+1
         REMS(IPTRE,IPTRR)=-1.0
3000     CONTINUE
4000  CONTINUE
      RETURN
      END
```

Figure D-24. Source Listing of the EXTASG Subroutine of the Main Program of the AFP Combat Module

(9) **EXTKLS**

(a) **Purpose.** To assess the kills due to indirect fire during a conflict. Figure D-25 is a source listing of subroutine EXTKLS.

(b) **When Called.** During every shot cycle, from MAIN.

(c) **Process.** The number of indirect shots during this shot cycle is computed and the number of expected kills derived. For each expected kill, a pointer is generated to an opponent's particular weapon in a particular duel in the conflict. If the weapon has been destroyed previously, no credit is given. If the weapon was intact, a kill is recorded, and the weapon ceases to participate in the duel. Note that when the direct fire duel terminates before the estimated number of shots occur, the artillery may cause casualties after the duel's termination.

(d) **Subtasks.** For a given indirect shooter type:

1. The number of shooters allocated to the given opponent type, range, and environment is obtained from the table computed by EXTASG.

2. The number of tubes derived above is multiplied by the number of indirect fire salvos expected during this shot cycle to obtain the number of rounds. The number of indirect salvos during this shot cycle is computed by the routine.

3. Multiplication of the number derived in step 3 by the fractional kill per round per target gives the expected number of kills. This number is reduced by multiplication by the fraction of the initial daily inventory of the target type present at the conflict. Another adjustment may be applied to account for changes in target density associated with whether an attrited force defends or attacks over the original or reduced area.

4. For each expected kill, a potential victim is picked randomly from the conflict. If the victim is not previously destroyed, a kill is credited and the weapon is removed from the duel. If the victim was previously destroyed, no credit is given. As usual, there is some special logic to determine whether fractional results are to be rounded "up" or "off."

5. During the process, the number of indirect shots is tallied.

```
      SUBROUTINE EXTKLS(ENVIRX,TYPS2X,CNFX,RANGEX,STAGEX)
C
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK2
      INCLUDE BK3
      INCLUDE BK4
      INCLUDE BK6
      INCLUDE BK7
      INCLUDE BK9
      INCLUDE BK13
      INCLUDE BK14
      INCLUDE BK15
      INCLUDE BK16
      INCLUDE BK17
      INCLUDE BK19
      INCLUDE BK22
      INCLUDE BK23
      INCLUDE BK24
      INCLUDE BK25
      INCLUDE BK27
C
      DIMENSION CNFST(2)
C
      REAL INTERV,STRCNT
C
      INTEGER EXTX,POINT1,POINT2,CONPTR,CONFLX,SIDEX,STAGEX,ODDSX
      INTEGER ENVIRX,TYPS2X,CNFX,RANGEX,COUNT,WPNPTR,VPTR,CNFST
C
      LOGICAL COND1,COND2
C
      SAVE CNFST,COUNT
C
2     CONTINUE
C   LOOP THROUGH CONFLICTS RECORDING KILLS BY EXTERNAL WEAPONS
                    IF ( CNFLCT .LE. 0 ) RETURN
                    KILLS1=0
                    IF ( STAGEX .EQ. 1 )  THEN
C   SAVE THE STRENGTH THE CONFLICT STARTS WITH
                      CNFST(1)=FORCS2(1)
                      CNFST(2)=FORCS2(2)
                      COUNT=0
                      DO 10 ODDSX=1,NODDS
                        COUNT=COUNT+CONFLO(ODDSX,RANGEX,ENVIRX)
     &                              * ODDS(ODDSX)
10                    CONTINUE
                    ENDIF
C   LOOP OVER THE SIDES
                    DO 2527 SIDEX=1,2
C     LOOP OVER THE EXTERNAL WEAPON TYPES (ASSUMED TO BE THE
C       LAST TYPES )!!!!
C   SKIP IF NO EXTERNAL WEAPONS PRESENT
                    IF ( (SIDEX .EQ. 1) .AND. (IEXT1 .EQ. 0) )
     &                GOTO 2527
                    IF ( (SIDEX .EQ. 2) .AND. (IEXT2 .EQ. 0) )
     &                GOTO 2527
                    DO 2525 EXTX=1,NOEXT(SIDEX)
C   CREATE POINTER TO THE TYPE NUMBER OF THE EXTERNAL WEAPON
                      ITEMP=NTYPS(SIDEX)-NOEXT(SIDEX)+EXTX
                      IF ( REFIRE(SIDEX,ITEMP,TYPS(3-SIDEX),ENVIRX)
     &                    .LT. SMALL2 ) GOTO 2525
C   COMPUTE THE NUMBER OF EXTERNAL SHOOTERS, BY REDUCING BY THE
C     PARTICIPATION RATE
                      STRCNT=EXTN1(SIDEX,EXTX,ENVIRX,RANGEX)
     &                          * INDPAR(SIDEX,EXTX,TYPS(3-SIDEX))
                      IF ( STAGEX .EQ. 1 ) THEN
C   APPEND TO THE EXTERNAL ASSIGNMENTS ARRAY
                        EXTN(SIDEX,EXTX,ENVIRX,RANGEX,CNFX) =
     &                    EXTN(SIDEX,EXTX,ENVIRX,RANGEX,CNFX)
     &                    +  STRCNT*PROJCT(TYPS2X,4) + SMALL2
CC  ******
CD          WRITE (6,97) STRCNT, PROJCT(TYPS2X,4),
CD     &    STRCNT*PROJCT(TYPS2X,4)+SMALL2,
CD     &    EXTN(SIDEX,EXTX,ENVIRX,RANGEX,CNFX)
CD97        FORMAT (' EXTN .5 ', 4G10.4)
CC  ******
C   RECORD THE ASSIGNMENTS
                        LOG(CNFX,3-SIDEX,ITEMP,SIDEX)=
     &                    LOG(CNFX,3-SIDEX,ITEMP,SIDEX)+STRCNT*
     &                    PROJCT(TYPS2X,4)
                      ENDIF
```

Figure D-25. Source Listing of the EXTKLS Subroutine of the Main Program of the AFP Combat Module (page 1 of 4 pages)

```
                        IF ( STRCNT .EQ. 0 ) GOTO 2525
C     COMPUTE THE TIME PER SHOT
                        RATIO=PROJCT(TYPS2X,3)/NSTAGE
                        RATIO=RATIO/REFIRE(SIDEX,ITEMP,TYPS(3-SIDEX)
     &                                          ,ENVIRX)
C     SEE IF THE ELAPSED TIME ALLOWS ANOTHER SALVO
                        ITEMP4=IFIX(RATIO*STAGEX)-IFIX(RATIO*(STAGEX-1))
CC *****
CD          IF (CNFLCT .GT. 0 ) THEN
CD          WRITE(6,99) (PROJCT(TYPS2X,M),M=3,4),NSTAGE,RATIO
CD       &      ,REFIRE(SIDEX,ITEMP,TYPS(3-SIDEX),ENVIRX),ITEMP
CD       &      ,STAGEX,RATIO*STAGEX,RATIO*(STAGEX-1),ITEMP4
CD       &      ,STRCNT*ITEMP4*EXT(SIDEX,EXTX,TYPS(3-SIDEX),ENVIRX,RANGEX)
CD       &      ,EXT(SIDEX,EXTX,TYPS(3-SIDEX),ENVIRX,RANGEX),STRCNT
CD       &  ,TYPS(3-SIDEX),WPNS1(3-SIDEX,TYPS(3-SIDEX)),FORCS2(3-SIDEX)
CD99        FORMAT(1X,'EXTKLS 1  PROJCT= ',2(3X,F5.2),3X,'NSTAGE= ',I5
CD       &      ,3X,'RATIO= ',F10.5,3X,'REFIRE= ',F10.5,3X/
CD       &      ,1X,'ITEMP= ',I5,3X,'STAGEX= ',I5,3X
CD       &      ,'RAT*STG= ',F10.5,3X,'RAT*STG-1= ',F10.5
CD       &    ,3X,'ITEMP4= ',I5,3X,/,1X,'KILLS= ',F10.3
CD       &    ,3X,'KPR= ',F10.5,3X,'STRCNT= ',F10.3,/1X,'TARGET TYPE= ',I3
CD       &    ,1X,'TARGET INVENTORY= ',I10,3X,'LIVE TARGETS= ',I10)
CD          ENDIF
CC *****
                        IF ( ITEMP4 .EQ. 0 ) GOTO 2525
C     OBTAIN THE EXPECTED KILLS PER ROUND
                        TEMP=EXT(SIDEX,EXTX,TYPS(3-SIDEX),ENVIRX,RANGEX)
C       TO OBTAIN THE KILLS FOR THIS SHOT CYCLE, BY MULTIPLYING THE
C         KILLS PER ROUND PER TARGET BY THE NUMBER OF SHOOTERS AND
C         BY THE NUMBER OF SALVOS PER SHOT CYCLE
                        TEMP=TEMP*STRCNT*ITEMP4
C     DEGRADE KILLS BY FRACTION OF INVENTORY PRESENT
                        TEMP5=FLOAT(DAYST(3-SIDEX))/
     &                      BWPNS(3-SIDEX,TYPS(3-SIDEX),ENVIRX,RANGEX)
                        TEMP=TEMP*TEMP5
C     DECREASE KILLS DUE TO REDUCED DENSITY IF REQUIRED
                        IF ( LESDNS(3-SIDEX) ) THEN
                          TEMP6=FLOAT(WPNS1(3-SIDEX,TYPS(3-SIDEX)))
     &                      / FLOAT(WPNS11(3-SIDEX,TYPS(3-SIDEX)))
                        ELSE
                          TEMP6 = 1.0
                        ENDIF
                        TEMP = TEMP * TEMP6
CC*****
C         IF ( CNFLCT .GT. 0 ) THEN
C         WRITE (6,101) TYPS(1), TYPS(2), CNFX,
C       & TEMP,DAYST(3-SIDEX),BWPNS(3-SIDEX,TYPS(3-SIDEX),
C       & ENVIRX,RANGEX),TEMP5,WPNS1(3-SIDEX,TYPS(3-SIDEX)),
C       & WPNS11(3-SIDEX,TYPS(3-SIDEX)),LESDNS(3-SIDEX),TEMP6
C101      FORMAT(1X,'EXTKLS 2 ', 3I3,' KILLS= ',F10.3,3X,'DAYST= ',I5,3X
C       &   ,'BWPNS= ',I7,3X,'FRACTION PRESENT= ',F6.4,3X,'WPNS1= '
C       &   ,I5,/,4X,'WPNS11= ',I5,3X,'DENSITY? ',L5,3X
C       &   ,'DENSITY FACTOR= ',F6.4)
C         ENDIF
CC*****
C     ADD THE NUMBER OF EXTERNAL SHOTS
                        ESHOTS(SIDEX,EXTX,ENVIRX,RANGEX) =
     &                    ESHOTS(SIDEX,EXTX,ENVIRX,RANGEX)
     &                    + STRCNT*ITEMP4*PROJCT(TYPS2X,4)*TEMP5
CC    *****
CD              WRITE(6,98) EXTN(SIDEX,EXTX,ENVIRX,RANGEX,CNFX)
CD       &        ,ESHOTS(SIDEX,EXTX,ENVIRX,RANGEX),STRCNT,ITEMP4
CD       &        ,TEMP5,LESDNS(3-SIDEX),TEMP6
CD       &        ,EXT(SIDEX,EXTX,TYPS(3-SIDEX),ENVIRX,RANGEX)
CD98            FORMAT(1X,'EXTKLS 2.25 EXTN= ',G10.4,3X,'SHOTS= ',F12.5
CD       &        ,3X,'TUBES= ',G10.4,3X,/,' SALVOS= ',I5,3X
CD       &        ,'FRACTION PRESENT= ',F6.4,3X,'REDUCE DENSITY ',L3
CD       &        ,/,' REDUCTION= ',F6.4,3X,'KILLS PER ROUND= ',G10.4)
CC    *****
                        KILLS=IFIX(TEMP)
                        REMN=TEMP-KILLS
C     ADD 1 IN ORDER TO ALLOW ASSESSMENT OF FRACTIONAL KILL
                        KILLS=KILLS+1
C     LOOP OVER THE NUMBER OF CONFLICTS IN WHICH EXTERNAL WEAPONS
C         WILL INTERVENE
                        DO 2520 I=1,KILLS
C     DETERMINE WHETHER A FRACTIONAL KILL IS TO BE ASSESSED
                          IF ( I .EQ. KILLS ) THEN
                            TEMP2=FIMSLU(INSEED)
C     SKIP IF RANDOM NUMBER BIGGER THAN FRACTION
                            IF ( TEMP2 .GT. REMN ) GOTO 2520
                          ENDIF
```

Figure D-25. Source Listing of the EXTKLS Subroutine of the Main Program of the AFP Combat Module
(page 2 of 4 pages)

```
                              IF ( SIDEX .EQ. BIGGER ) THEN
                                CONPTR= ( CNFLCT-1 ) * FIMSLU(INSEED) + 1
                              ELSE
C     COMPUTE THE INTERVAL SIZE PER WEAPON ON LARGER SIDE
                                INTERV=1.0/COUNT
C     COMPUTE A WEAPON NUMBER FOR THE WEAPON. IDS RUN FROM 0
C          TO COUNT-1
                                WPNPTR=FIMSLU(INSEED)/INTERV
C     REDUCE THE WEAPON NUMBER IF NECESSARY
                                IF ( WPNPTR .EQ. COUNT ) WPNPTR=COUNT-1
C
C     CHECK TO SEE IF WEAPON IS IN THE LARGER ODDS DUELS
                                IF ( WPNPTR .GT.
     &                           ODDS(1)*CONFLO(1,RANGEX,ENVIRX)-1 ) THEN
C     REORIGIN THE WEAPON NUMBER
                                  WPNPTR=WPNPTR
     &                                     -ODDS(1)*CONFLO(1,RANGEX,ENVIRX)
C     SET THE CONFLICT POINTER
                                  CONPTR=WPNPTR/ODDS(2)
     &                                     +CONFLO(1,RANGEX,ENVIRX)+1
C     COMPUTE THE POINTER TO THE WEAPON IN THE DUEL
                                  VPTR=MOD(WPNPTR,ODDS(2))+1
                                ELSE
C     WEAPON IN THE LOWER ODDS DUELS
                                  CONPTR=(WPNPTR/ODDS(1))+1
                                  VPTR=MOD(WPNPTR,ODDS(1))+1
                                ENDIF
                              ENDIF
C     POINTER(S) TO THE STATUS ENTRY FOR THE IMPACTED WEAPON
C     DETERMINE THE CONFLICT IN WHICH THE EXTERNAL WEAPONS INTERVENE
C     TO DETERMINE THE KILLS, IF ANY
                                IF ( BIGGER .EQ. SIDEX ) THEN
CC *****
CD         IF ( CNFLCT .GT. 0 ) THEN
CD           WRITE(6,102) CONPTR
CD102        FORMAT(1X,'EXTKLS 2.5',3X,'HIT DUEL= ',I5,3X,'ENTRY 3')
CD         ENDIF
CC *****
C     RECORD KILLING WEAPON IF WEAPON HASN'T BEEN KILLED EARLIER
                                  IF ( STATUS(CONPTR,3) .LT. 0.0 )
     &                              GOTO 2520
                                  STATUS(CONPTR,3)=-EXTX-.1
                                  KILLS1=KILLS1+1
CC *****
CD         IF ( CNFLCT .GT. 0 ) THEN
CD           WRITE(6,104) KILLS,CONPTR,EXTX
CD104        FORMAT(1X,'*EXTKLS 3, KILLS= ',I5,3X,'DUEL= ',I5
CD       &           ,3X,'EXTERNAL WPN ID= ',I5)
CD         ENDIF
CC *****
                                  GOTO 2520
                                ELSE
C     SMALLER SIDE DID KILLING. FIND WHICH WEAPON WAS KILLED
CC *****
CD         IF ( CNFLCT .GT. 0 ) THEN
CD           WRITE(6,106) CONPTR,3+VPTR,COUNT,WPNPTR,INTERV
CD106        FORMAT(1X,'EXTKLS 3.5',3X,'HIT DUEL= ',I5,3X,'ENTRY= ',I5
CD       &    ,/,' COUNT= ',I5,3X,'WPNPTR= ',I5,3X,'INTERVAL= ',F8.6)
CD         ENDIF
CC *****
C     RECORD KILLING WEAPON IF WEAPON HASN'T BEEN KILLED EARLIER
                                  IF ( STATUS(CONPTR,3+VPTR) .LT. 0.0 )
     &                              GOTO 2520
                                  STATUS(CONPTR,3+VPTR)=-EXTX-.1
                                  KILLS1=KILLS1+1
CC *****
CD         IF ( CNFLCT .GT. 0 ) THEN
CD           WRITE(6,105) KILLS,CONPTR,3+VPTR,EXTX
CD105        FORMAT(1X,'*EXTKLS 4, KILLS= ',I5,3X,'DUEL= ',I5,3X,'ENTRY= '
CD       &           ,I5,3X,'EXTERNAL WPN ID= ',I5)
CD         ENDIF
CC *****
                                ENDIF
2520                        CONTINUE
2525                      CONTINUE
2527                      CONTINUE
C     NOW TO RECORD EXTERNAL KILLS
                          IF ( KILLS1 .NE. 0 ) THEN
C     EXTERNAL WEAPONS HAVE KILLED SOMETHING
                            DO 2580 CONFLX=1,CNFLCT
C     SIDEX INDEXES THE SHOOTERS
                              DO 2560 SIDEX=1,2
```

Figure D-25. Source Listing of the EXTKLS Subroutine of the Main Program of the AFP Combat Module (page 3 of 4 pages)

```
                          IF ( SIDEX .EQ. BIGGER ) THEN
                            POINT1=0
                            POINT2=0
                          ELSE
                            POINT1=1
                            POINT2=IFIX(STATUS(CONFLX,2))
                          ENDIF
C    LOOP THROUGH LOOKING FOR VICTIMS, J INDEXES THE POTENTIAL VICTIMS
                          DO 2540 J=POINT1,POINT2
                            ITEMP2=STATUS(CONFLX,3+J)
C    SKIP TO NEXT POTENTIAL VICTIM IF EXTERNAL TYPE MADE NO KILLS
                            IF ( ITEMP2 .GE. 0 ) GOTO 2540
C    SKIP ENTRIES ALREADY DEAD
                            IF ( ITEMP2 .LT. -IEXT ) GOTO 2540
C    PLACE THE ID OF THE EXTERNAL KILLER IN ITEMP2
                            ITEMP2=-ITEMP2
                            ITEMP3=NTYPS(SIDEX)-NOEXT(SIDEX)+ITEMP2
                            IF ( SIDEX .EQ. BIGGER ) THEN
C    THE BIGGER SIDE IS DOING THE SHOOTING, THE SMALLER SIDE THEN
C     FURNISHES THE POTENTIAL VICTIMS
C     MARK THE ENTRY FOR DELETION
                              STATUS(CONFLX,1)=-1.0
C    POINT TO THE TYPE OF THE KILL
                              EXTK(SIDEX,ITEMP2,ENVIRX,RANGEX,CNFX) =
     &                            EXTK(SIDEX,ITEMP2,ENVIRX,RANGEX,CNFX)
     &                            +PROJCT(TYPS2X,4)
C    STORE THE KILLS
                              LOG(CNFX,3-SIDEX,ITEMP3,6)=
     &                            LOG(CNFX,3-SIDEX,ITEMP3,6)+
     &                            PROJCT(TYPS2X,4)
                              WPNS(3-SIDEX,TYPS(3-SIDEX))=MAX(0,
     &                            WPNS(3-SIDEX,TYPS(3-SIDEX))-1)
                        TTLOS(SMALER,ITEMP2+1,ENVIRX,RANGEX)=
     &                  TTLOS(SMALER,ITEMP2+1,ENVIRX,RANGEX)+
     &                             PROJCT(TYPS2X,4)
C    SUBTRACT LOSS FROM FORCES AVAILABLE FOR THE NEXT ENGAGEMENT
                              FORCS1(SMALER)
     &                            =MAX(FORCS1(SMALER)-1,0)
                              FORCS2(SMALER)
     &                            =MAX(FORCS2(SMALER)-1,0)
C    PROCESS OTHER SIDE, ONLY WEAPON POSSIBLE HAS BEEN KILLED
                            ELSE
C    THE SMALLER SIDE IS DOING THE SHOOTING
C    PICK UP THE NUMBER OF WEAPONS ON THE LARGER SIDE
                              ITEMP=IFIX(STATUS(CONFLX,2))
C    RECORD THE LOSSES BY EXTERNAL WEAPON SHOOTER TYPE
                              EXTK(SIDEX,ITEMP2,ENVIRX,RANGEX,CNFX) =
     &                          EXTK(SIDEX,ITEMP2,ENVIRX, RANGEX,CNFX)
     &                        + MIN(ITEMP,1)*PROJCT(TYPS2X,4)
C    STORE THE KILLS
                              LOG(CNFX,3-SIDEX,ITEMP3,6)=
     &                            LOG(CNFX,3-SIDEX,ITEMP3,6)+
     &                            MIN(ITEMP,1)*PROJCT(TYPS2X,4)
                              WPNS(3-SIDEX,TYPS(3-SIDEX))=MAX(0,
     &                            WPNS(3-SIDEX,TYPS(3-SIDEX))-
     &                            MIN(ITEMP,1))
                        TTLOS(BIGGER,ITEMP2+1,ENVIRX,RANGEX) =
     &                  TTLOS(BIGGER,ITEMP2+1,ENVIRX,RANGEX) +
     &                             MIN(ITEMP,1)*PROJCT(TYPS2X,4)
C    SUBTRACT LOSS FROM FORCES AVAILABLE FOR THE NEXT ENGAGEMENT
                              FORCS1(BIGGER)
     &                            = MAX(FORCS1(BIGGER)
     &                            -MIN(ITEMP,1),0)
                              FORCS2(BIGGER)
     &                            = MAX(FORCS2(BIGGER)
     &                            -MIN(ITEMP,1),0)
C   ENDIF FOR IF ( SIDEX .EQ. BIGGER ) THEN
                            ENDIF
C    SET KILLED ENTRIES VERY NEGATIVE
                            IF ( STATUS(CONFLX,3+J) .LT. -SMALL1 )
     &                        STATUS(CONFLX,3+J)=-10.0*ITYPS
C END OF LOOP ON VICTIMS
2540                      CONTINUE
C    END OF LOOP ON SIDES
2560                    CONTINUE
C    CHECK FOR SURVIVALS INSUFFICIENT FOR FURTHER COMBAT
                        COND1=STATUS(CONFLX,3) .GT. -SMALL2
                        DO 2570 K=1,IFIX(STATUS(CONFLX,2))
                          COND2= COND1 .AND.
     &                      ( STATUS(CONFLX,3+K) .GT. -SMALL2 )
                          IF ( COND2 ) GOTO 2575
2570                    CONTINUE
                        STATUS(CONFLX,1)=-1.0
2575                    CONTINUE
2580                  CONTINUE
C    ENDIF FOR IF ( KILLS1 .NE. 0 )
                    ENDIF
                  RETURN
C ***
C      DEBUG SUBCHK,INIT(EXTK,RATIO,ITEMP,ITEMP4,NSTAGE,STRCNT)
C      AT 2
C      TRACE ON
C ***
                  END
```

Figure D-25. Source Listing of the EXTKLS Subroutine of the Main Program of the AFP Combat Module (page 4 of 4 pages)

(10) FIMSLU

(a) Purpose. To return a pseudoramdom real number in accord with seed designated on first call with Combat Module run. Figure D-26 is a source listing of function subroutine FIMSLU.

(b) Source. FIMSLU is an adaptation of a routine from the International Mathematics and Statistics Library.

(11) FRCALC

(a) Purpose. This routine computes the type on type allocations. More specifically, for each pair of opposing types, this routine computes the number of weapons of each type slated to meet the other type, exclusive of participation rates. Figure D-27 is a source listing of subroutine FRCALC.

(b) When Called. At the beginning of each day, by MAIN.

(c) Subtasks

1. Obtains the available weapons by adding reinforcements/initial allocations to survivors.

2. Removes external losses, i.e., losses before combat, if any.

3. Saves the initial inventory for the day.

4. Reads in the appropriate modified preferences.

5. Computes the preferred target, i.e., the product of the preferences and the initial daily target inventory.

6. Computes the unnormalized fractional allocation and normalizes it to sum to the sum of the preferences.

7. Allocates opponents using the normalized allocations. The first phase allocates integral numbers of weapons. The second phase allocates the fractional weapons unallocated in the first phase.

```
      REAL FUNCTION FIMSLU (IREP)                                       GGUT0390
C                                 SPECIFICATIONS FOR ARGUMENTS          GGUT0400
      INTEGER            IREP
C                                 SPECIFICATIONS FOR LOCAL VARIABLES    GGUT0440
      INTEGER            IREPM,LREP,MREP
      DOUBLE PRECISION   DSEED,DSEED1,DSEED2                            GGUT0430
      DOUBLE PRECISION   D2P31M,D2P31                                   GGUT0460
C                                 D2P31M=(2**31) - 1                    GGUT0470
C                                 D2P31 =(2**31)(OR AN ADJUSTED VALUE)  GGUT0480
      DOUBLE PRECISION   SEED(100)
C                                 SEEDS SPACED 100,000 RAND NOS APART
      DATA               D2P31M/2147483647.D0/                          GGUT0490
      DATA               D2P31/2147483648.D0/                           GGUT0500
      DATA               (SEED(I),I=1,52)/
     *        123457.D0,  848661055.D0,  331868080.D0,  795694002.D0,
     *     1133506416.D0,  179392340.D0,  880089848.D0,  373453165.D0,
     *      330310551.D0, 1864683550.D0,  957584520.D0,  611600725.D0,
     *     1514685799.D0,  609430781.D0, 2126485978.D0,  230140735.D0,
     *     1460567237.D0, 1903540707.D0,  773122865.D0, 1753901423.D0,
     *      274992647.D0, 1588509478.D0, 1759740400.D0, 1941135750.D0,
     *       83729995.D0,  307509594.D0, 1235568876.D0, 1540164579.D0,
     *     1567253092.D0,   34485768.D0,  307246871.D0, 1222711937.D0,
     *      352414837.D0, 1812009750.D0,  892733067.D0, 1257585462.D0,
     *     1990506775.D0,   34124592.D0,  134177024.D0, 1176990974.D0,
     *     1265016020.D0, 1110128581.D0,   79706148.D0, 1970893844.D0,
     *      235610717.D0,  716522888.D0,   65884601.D0,   26530237.D0,
     *     1946788467.D0, 1915124666.D0,  605204808.D0, 1250036153.D0/
      DATA               (SEED(I),I=53,100)/
     *     1241237486.D0, 1826480319.D0, 1761451800.D0,  648974769.D0,
     *     1328095926.D0,  735026667.D0, 1922904882.D0,  112740804.D0,
     *     1690100682.D0,  673720340.D0, 1103976199.D0,  200368466.D0,
     *     1409578632.D0,   12672413.D0, 2082847903.D0, 2102585575.D0,
     *      585411451.D0,  560061851.D0, 1357483287.D0, 2035203198.D0,
     *     1995361080.D0, 2009993142.D0,  270796833.D0, 1364574012.D0,
     *     2037737285.D0,   37500927.D0, 2052071248.D0, 1860954082.D0,
     *     1475434694.D0, 2074851491.D0,  330340421.D0, 2037642869.D0,
     *      133673398.D0,  851290297.D0, 1651586143.D0,  301620020.D0,
     *     1510133733.D0, 1940663167.D0, 1877417917.D0,  617723160.D0,
     *     1093293442.D0, 1138218931.D0,  950324013.D0,  199964875.D0,
     *     1721714736.D0, 1039836864.D0, 1390471833.D0,  403248081.D0/
C                                 FIRST EXECUTABLE STATEMENT            GGUT0510
      IF (IREP.GT.0) GO TO 100
      IF (DSEED.EQ.0.0) DSEED=SEED(1)
      GO TO 200
C
  100 IF (IREP.EQ.LREP) GO TO 200
      IREPM  = IREP-1
      MREP   = MOD(IREPM,100)+1
      DSEED  = SEED(MREP)
C
  200 DSEED1 = DMOD( 6060.D0*DSEED,D2P31M)                              GGUT0530
      DSEED2 = DMOD(55934.D0*DSEED,D2P31M)                              GGUT0540
      DSEED  = DMOD(65536.D0*DSEED1+DSEED2,D2P31M)                      GGUT0550
      FIMSLU = DSEED / D2P31                                            GGUT0560
      LREP   =IREP
      RETURN                                                            GGUT0580
      END                                                               GGUT0590
```

Figure D-26. Source Listing of the FIMSLU Subroutine of the Main Program of the AFP Combat Module

```
      SUBROUTINE FRCALC(DAYSX)
C
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK7
      INCLUDE BK8
      INCLUDE BK13
      INCLUDE BK17
C
      DIMENSION REMS(2,ITYPS,ITYPS), ALLOC(2,ITYPS,ITYPS),PREF(2,ITYPS)
C
      INTEGER DAYSX,TYPSX,TYPESX,COUNT,POINT,SIDEX
C
      EQUIVALENCE (ALLOC,REMS)
C
2     CONTINUE
C     DEVELOP THE DAY'S INITIAL FORCES
          DO 200 SIDEX=1,2
            DO 100 TYPSX=1,NTYPS(SIDEX)
C INCLUDE ADDED WEAPONS
              WPNS(SIDEX,TYPSX)=INSERT(PHASE(DAYSX),SIDEX,TYPSX)+
     &          WPNS(SIDEX,TYPSX)
C REMOVE LOSSES EXTERNAL TO THE MODEL
              WPNS(SIDEX,TYPSX)=WPNS(SIDEX,TYPSX)*
     &          (1.0-EXTLOS(SIDEX,TYPSX))
              WPNS1(SIDEX,TYPSX)=WPNS(SIDEX,TYPSX)
              IF (DAYSX .EQ. 1) WPNS11(SIDEX,TYPSX) = WPNS(SIDEX,TYPSX)
100         CONTINUE
200       CONTINUE
C   COMPUTE THE WEAPON ALLOCATIONS
          DO 240 SIDEX=1,2
          DO 240 TYPSX=1,NTYPS(SIDEX)
C   READ IN THE APPROPRIATE PREFERENCE FILES
            IOPTR7=(SIDEX-1)*ITYPS+TYPSX
            IOPTR7=IOPTR7+PREF3*2*ITYPS
            READ(IU7'IOPTR7,2000,ERR=2100)
     &        (PREF(SIDEX,J),J=1,NTYPS(3-SIDEX))
2000        FORMAT(500(500F8.6))
            GOTO 2300
2100        WRITE(6,2200) SIDEX
2200        FORMAT(1X,30(1H*),5X,'PREF ',I1,' READ ERROR',5X,30(1H*))
            STOP
2300        CONTINUE
            TEMP=0.0
            TEMP1=0.0
            DO 220 TYPESX=1,NTYPS(3-SIDEX)
C   COMPUTE THE WEIGHTED ALLOCATION
              ALLOC(SIDEX,TYPSX,TYPESX)=PREF(SIDEX,TYPESX)*
     &          WPNS(3-SIDEX,TYPESX)
C   COMPUTE THE WEIGHTED ALLOCATION TOTALS
              TEMP=TEMP+ALLOC(SIDEX,TYPSX,TYPESX)
              TEMP1=TEMP1+PREF(SIDEX,TYPESX)
220         CONTINUE
            TEMP1=AMIN1(1.0,TEMP1)
C   NORMALIZE BY THE WEIGHTED ALLOCATION TOTALS
            DO 230 TYPESX=1,NTYPS(3-SIDEX)
              IF ( TEMP .GE. SMALL2 ) THEN
                ALLOC(SIDEX,TYPSX,TYPESX)=(ALLOC(SIDEX,TYPSX,TYPESX)/
     &            TEMP)*TEMP1
              ELSE
```

Figure D-27. Source Listing or the FRCALC Subroutine of the Main Program of the AFP Combat Module (page 1 of 2 pages)

```
                    ALLOC(SIDEX,TYPSX,TYPESX)=0.0
                  ENDIF
230             CONTINUE
240           CONTINUE
C ALLOCATE OPPONENTS
              DO 1000 SIDEX=1,2
              DO 1000 TYPSX=1,NTYPS(SIDEX)
                COUNT=0
                TEMP1=0.0
                DO 700 TYPESX=1,NTYPS(3-SIDEX)
                  TEMP=WPNS(SIDEX,TYPSX)*ALLOC(SIDEX,TYPSX,TYPESX)
                  TEMP1=TEMP1+ALLOC(SIDEX,TYPSX,TYPESX)
C ALLOCATE THE INTEGER PART
                  FORCES(SIDEX,TYPSX,TYPESX)=TEMP
                  REMS(SIDEX,TYPSX,TYPESX)=TEMP-FORCES(SIDEX,TYPSX,TYPESX)
C TALLY THE ALLOCATED WEAPONS
                  COUNT=COUNT+FORCES(SIDEX,TYPSX,TYPESX)
700             CONTINUE
                TEMP1=AMIN1(1.0,TEMP1)
C RETAIN THE UNALLOCATED WEAPON COUNT
                COUNT=WPNS(SIDEX,TYPSX)*TEMP1-COUNT+SMALLZ
                IF ( COUNT .LE. 0 ) GOTO 1000
C TO ALLOCATE THE REMAINING WEAPONS USING THE LARGEST
C     FRACTIONAL PARTS
800             CONTINUE
                POINT=1
                DO 900 TYPESX=2,NTYPS(3-SIDEX)
C TEST FOR A LARGER REMAINDER
                  IF ( REMS(SIDEX,TYPSX,TYPESX) .GT.
     &                REMS(SIDEX,TYPSX,POINT) ) THEN
                    POINT=TYPESX
                  ENDIF
900             CONTINUE
C MAKE SURE NO FURTHER ALLOCATIONS OCCUR TO MAX JUST FOUND
                REMS(SIDEX,TYPSX,POINT)=-1.0
C ALLOCATE A WEAPON
                FORCES(SIDEX,TYPSX,POINT)=FORCES(SIDEX,TYPSX,POINT)+1
C DECREMENT THE COUNT
                COUNT=COUNT-1
                IF ( COUNT .GT. 0 ) GOTO 800
1000          CONTINUE
            RETURN
            END
```

Figure D-27. Source Listing or the FRCALC Subroutine of the Main Program of the AFP Combat Module (page 2 of 2 pages)

(d) **Example:** Let's say there are 10 weapons of a given type to be allocated against three opposing types. Say also that applying the rates yields the following allocations

1.8

2.5

5.7

against opposing types. We rewrite this as:

Allocation	Remainders
1	.8
2	.5
5	.7

The first phase allocates, 1, 2, and 5 weapons; note that two weapons remain unallocated.

The second phase uses the remainders, which are:

.8
.5
.7

The phase allocates weapons based on the size of the remainders. The first weapon is allocated against the target associated with the largest remainder (.8). The next weapon is allocated against the target type associated with the next largest remainder (.7). Since no unallocated weapons remain, the third target is not allocated additional weapons in this day.

(12) **F2FRT.** The alternate file reference table element F2FRT is included in order to extend the number of write files available to the Main Program of the AFP Combat Module beyond the standard number defined within the installation FORTRAN system library. The version of F2FRT included for the combat modules adds units 91-99 as alternate print symbionts. Figure D-28 provides the source listing of the alternate file reference table element.

```
              F2FRT        99
              PR            6
              PU           1
              CR            5
             APR           91,92,93,94,95,96,97,98,99
              RR            0
       . 1 PRINT FILES = I/O UNIT 6
       . 1 PUNCH FILE = I/O UNIT 1
       . 1 CARD READ FILE = I/O UNIT 5
       . 1 RE-READ FILE (NOT USED) = I/O UNIT 0
       . 9 ALTERNATE PRINT SYMBIONTS = I/O UNITS 91-99
       .
       .      *** FILE DEFINITIONS ***
       .
       .
                 END
```

Figure D-28. Source Listing of the F2FRT Element of the Main Program of the AFP Combat Module

(13) **IMAGES**

(a) **Purpose.** For specified image intensifier and conditions, to return the number of resolvable cycles. Figure D-29 is a source listing of subroutine IMAGES.

(b) **Source.** IMAGES is an implementation of logic developed at the Night Vision Laboratory.

(c) **When Called.** From DETECT if the weapon platform in question depends on an image intensifier.

```
      SUBROUTINE IMAGES(RC,ACON,ATTN,RNG,LSC,SOG,AL)
C
C                                         IMAGE INTENSIFIERS
      DIMENSION AMRCS(6,2),AMRCC(6,2),AMRCI(6,2)
C
C                                                   LIGHT LEVELS,COEFF CURVE
      DATA (AMRCS(I,1),I=1,6)/
     1     0.01988,0.04195,0.11580,0.35398,1.11886,3.38584/
C
      DATA (AMRCS(I,2),I=1,6)/
     1     0.33982,0.41019,0.47368,0.48725,0.29944,-0.20059/
C
      DATA (AMRCC(I,1),I=1,6)/
     1     C.01285,0.02657,0.07682,0.22171,0.71499,2.09944/
C
      DATA (AMRCC(I,2),I=1,6)/
     1     0.19153,0.23353,0.24174,0.29012,0.10917,-0.12972/
C
      DATA (AMRCI(I,1),I=1,6)/
     1     0.01464,0.02648,0.06138,0.14226,0.54625,1.64788/
      DATA (AMRCI(I,2),I=1,6)/
     1     0.35622,0.36992,0.40250,0.61333,0.36543,0.28749/
C                                        IMAGE INTENSIFIER DEVICES (3,4,5)
      RC=0.
      RC1=0.
      RC2=0.
      C = ACON / (1.+SOG*(EXP(ATTN*RNG) -1.))
C                                              CHECK IF TOO DARK
      IF (AL.LT.0.000001) GO TO 270
C
      ACK=0.10
      IF (AL.GT.0.1) GO TO 270
C
      DO 200 II=1,5
      IL1=II
      IF (AL.LE.ACK.AND.AL.GE.ACK/10.) GO TO 210
C
      ACK=ACK/10.
C
  200 CONTINUE
  210 IL2=IL1+1
C
      IF (LSC.EQ.5) GO TO 220
      IF (LSC.EQ.4) GO TO 230
C                                         IMAGE INTENSIFIER 3  STARLIGHT
      RC1=C/(AMRCS(IL1,1)+AMRCS(IL1,2)*C)
      RC2=C/(AMRCS(IL2,1)+AMRCS(IL2,2)*C)
      GO TO 240
C                                        IMAGE INTENSIFIER 5  RED SIGHT
  220 RC1=C/(AMRCI(IL1,1)+AMRCI(IL1,2)*C)
      RC2=C/(AMRCI(IL2,1)+AMRCI(IL2,2)*C)
      GO TO 240
C                                          IMAGE INTENSIFIER 4  CREW WPN S
  230 RC1=C/(AMRCC(IL1,1)+AMRCC(IL1,2)*C)
      RC2=C/(AMRCC(IL2,1)+AMRCC(IL2,2)*C)
C
  240 IF (C.LT.0.02.OR.(C.LT.0.04.AND.IL1.EQ.5)) GO TO 270
C
      IF (LSC.EQ.3) GO TO 250
C
      IF (C.LT.0.07.AND.IL1.EQ.5) GO TO 270
      IF (C.LT.0.33.AND.IL1.EQ.5) RC2=0.
      GO TO 260
C
  250 IF (C.LT..33.AND.IL1.EQ.5) GO TO 270
      IF (C.LT..57.AND.IL1.EQ.5) RC2=0.
C                                                         INTERPOLATE
  260 RC=RC2+(RC1-RC2)*((AL-ACK/10.)/(ACK-ACK/10.))
C
  270 RETURN
      END
```

Figure D-29. Source Listing of the IMAGES Subroutine of the Main Program of the AFP Combat Module

(14) INPUT

(a) **Purpose.** To read input data from the run-specific file BASEDATA. Figure D-30 is a source listing of subroutine INPUT.

(b) **When Called.** Near beginning of MAIN.

Note: See Annex I to Appendix D for description of BASEDATA file.

(15) OPTICS

(a) **Purpose.** For specified optical band sensor and conditions, to return the number of resolvable cycles. Figure D-31 is a source listing of subroutine OPTICS.

(b) **Source.** OPTICS is an implementation of logic developed at the Night Vision Laboratory.

(c) **When Called.** From DETECT if the weapon platform in question depends on an optical band sensor.

(16) OUTPT

(a) **Purpose.** To write out the preparticipation reduced force allocations. Figure D-32 is a source listing of subroutine OUTPT.

(b) **When Called.** At the beginning of each day, by MAIN.

(c) **Format.** 16

```
      SUBROUTINE INPUT
C
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK3
      INCLUDE BK5
      INCLUDE BK6
      INCLUDE BK8
      INCLUDE BK9
      INCLUDE BK20
      INCLUDE BK22
      INCLUDE BK25
C
      PARAMETER MAXWPN=100000,REFIRM=30.,SIZEL=-.01,SIZEU=5.0
      PARAMETER RLITEM=650.
      PARAMETER CNTRSL=.00,CNTRSU=25.00,ATTNM=15.0,RMAGM=125.
      PARAMETER UBRUND=100., UBPART=1.0
      PARAMETER BRIGTM=7.00
C
      CHARACTER*80 JUNK
C
      DATA IER/0/
C
2     CONTINUE
C   DEFINE THE OUTPUT FILE
      OUTFIL=6
C   STORE THE ID OF THE LAST PHASE, ASSUMED EQUAL TO THE NUMBER
C   OF PHASES
      NPHASE=PHASE(IDAYS)
C
C   !!!! NSTAGO IS USED TO REDUCE THE NUMBER OF STAGES, SO IN !!!!!
C   !!!!    NON TEST RUNS, MAKE INPUT VERY LARGE              !!!!!
C
C READ IN THE NUMBER OF DAYS, NUMBER OF WEAPONS FOR SIDES
C   1 AND 2, NUMBER OF ENVIRONMENTS, THE NUMBER OF STAGES,
C   AND THE NUMBER OF EXTERNAL WEAPONS FOR SIDES 1 AND 2
C
      READ(IOUNIT,100) NDAYS,NTYPS(1),NTYPS(2),NENVIR,NSTAGO,
     &    NOEXT(1),NOEXT(2), NDTCTN
100   FORMAT()
C
      IF ( (NDAYS .GT. IDAYS) .OR. (NDAYS .LT. 0) ) THEN
        WRITE(OUTFIL,200) NDAYS
200     FORMAT(1X,30(1H*),5X,'NDAYS= ',I5,5X,30(1H*))
        IER=1
      ENDIF
C
      IF ( (NTYPS(1).GT.ITYPS1) .OR. (NTYPS(1).LE.0) ) THEN
        WRITE(OUTFIL,300) NTYPS(1)
300     FORMAT(1X,30(1H*),5X,'SIDE1 TYPES ',I5,5X,30(1H*))
        IER=1
      ENDIF
      IF ( (NTYPS(2).GT.ITYPS2) .OR. (NTYPS(2).LE.0) ) THEN
        WRITE(OUTFIL,400) NTYPS(2)
400     FORMAT(1X,30(1H*),5X,'SIDE2 TYPES ',I5,5X,30(1H*))
        IER=1
      ENDIF
C
      IF ( (NENVIR.GT.IENVIR) .OR. (NENVIR.LE.0) ) THEN
        WRITE(OUTFIL,500) NENVIR
500     FORMAT(1X,30(1H*),5X,'NENVIR=',I5,5X,30(1H*))
        IER=1
      ENDIF
C
      IF ( (NSTAGO .GT.ISTAGE) .OR. (NSTAGO.LE.0) ) THEN
          WRITE(OUTFIL,550) NSTAGE
550     FORMAT(1X,30(1H*),5X,'NSTAGE= ',I5,5X,30(1H*))
        IER=1
      ENDIF
C
      IF ( (NPHASE.GT.IPHASE) .OR. (NPHASE.LE.0) ) THEN
        WRITE(OUTFIL,600) NPHASE
600     FORMAT(1X,30(1H*),5X,'NPHASE=',I5,5X,30(1H*))
        IER=1
      ENDIF
      IF ( (NOEXT(1) .LT. 0) .OR. (NOEXT(1).GT.IEXT1) ) THEN
        WRITE(6,3220) NOEXT(1)
3220    FORMAT(1X,30(1H*),5X,'NOEXT(1)=',1X,I10,5X,30(1H*))
        IER=1
      ENDIF
      IF ( (NOEXT(2) .LT. 0) .OR. (NOEXT(2).GT.IEXT2) ) THEN
```

Figure D-30. Source Listing of the INPUT Subroutine of the Main Program of the AFP Combat Module (page 1 of 4 pages)

```
            WRITE(6,3240) NOEXT(2)
 3240       FORMAT(1X,30(1H*),5X,'NOEXT(2)=',1X,I10,5X,30(1H*))
            IER=1
          ENDIF
          IF (NDTCTN .LT. 1) THEN
            WRITE (6,3241) NDTCTN
 3241       FORMAT (1X, 30('*'), 5X, 'NDTCTN=', I10, 5X, 30('*'))
            IER = 1
          ENDIF
 C
          READ (IOUNIT,100) ((NS2DCT(I,J), J=1,NTYPS(I)), I=1,2)
          DO 650 I = 1,2
            DO 650 J = 1,NTYPS(I)
              IF (NS2DCT(I,J) .LT. 0) THEN
                WRITE (6,3242) I, J, NS2DCT(I,J)
 3242           FORMAT (1X, 30('*'), ' INDICES=', 2I3,
         &          ' NS2DCT=', I10, 5X, 30('*'))
                IER = 1
              ENDIF
 650      CONTINUE
 C
          READ(IOUNIT,100) (((INSERT(I,J,K),I=1,NPHASE),K=1,NTYPS(J)),
         &  J=1,2)
          DO 800 I=1,NPHASE
          DO 800 J=1,2
          DO 800 K=1,NTYPS(I)
            IF ( (INSERT(I,J,K) .GT. MAXWPN) .OR.
         &      (INSERT(I,J,K) .LT. 0) ) THEN
              WRITE(OUTFIL,700)I,J,K,INSERT(I,J,K)
 700          FORMAT(1X,30(1H*),5X,'INDICES=',3(I4,2X),'INSERT=',
         &       I7,5X,30(1H*))
              IER=1
            ENDIF
 800      CONTINUE
 C
          READ(IOUNIT,100) ((EXTLOS(I,J),J=1,NTYPS(I)),I=1,2)
          DO 1000 I=1,2
          DO 1000 J=1,NTYPS(I)
            IF ( (EXTLOS(I,J).GT.1.0) .OR. (EXTLOS(I,J).LT.0.0) )
         &      THEN
              WRITE(OUTFIL,900) I,J,EXTLOS(I,J)
 900          FORMAT(1X,30(1H*),5X,'INDICES=',2(I4,2X),I8,5X,30(1H*))
              IER=1
            ENDIF
 1000     CONTINUE
 C
          READ(IOUNIT,100) (((LIMITS(I,J,K),K=1,2),J=1,NTYPS(I)),I=1,2)
          DO 1400 I=1,2
          DO 1400 J=1,NTYPS(I)-NOEXT(I)
            IF ( (LIMITS(I,J,2).GT.IRANGE) .OR. (LIMITS(I,J,1).LT.1) )
         &      THEN
              WRITE(OUTFIL,1300) I,J,LIMITS(I,J,1),LIMITS(I,J,2)
 1300         FORMAT(1X,30(1H*),5X,'INDICES=',2(I5,5X),
         &          'LOW LIMITS= ',I3,5X,'HIGH LIMITS= ',I3,
         &        5X,30(1H*))
              IER=1
            ENDIF
 1400     CONTINUE
 C
          READ(IOUNIT,100) (ENVDST(I),I=1,NENVIR)
          DO 1600 I=1,NENVIR
            IF ( (ENVDST(I).GT.1.0) .OR. (ENVDST(I).LT.0.0) ) THEN
              WRITE(OUTFIL,1500) I,ENVDST(I)
 1500         FORMAT(1X,30(1H*),5X,'INDEX=',I5,5X,'ENVDST=',F8.2,
         &        5X,30(1H*))
              IER=1
            ENDIF
 1600     CONTINUE
 C
 C    READ IN THE INDIRECT FIRE DISTRIBUTION
          READ(IOUNIT,*) (((INDDST(I,J,K),K=1,IRANGE),J=1,NOEXT(I))
         &      ,I=1,2)
          DO 1660 I=1,2
          DO 1660 J=1,NOEXT(I)
          DO 1660 K=1,IRANGE
            IF( (INDDST(I,J,K) .LT. 0.0) .OR. (INDDST(I,J,K) .GT. 1.0) )
         &  THEN
              WRITE(OUTFIL,1620) I,J,K,INDDST(I,J,K)
 1620         FORMAT(1X,30(1H*),5X,'INDICES= ',3(I5,3X)
         &    ,'INDIRECT DISTRIBUTION= ',F8.4,5X,30(1H*))
              IER=1
            ENDIF
```

Figure D-30. Source Listing of the INPUT Subroutine of the Main Program of the AFP Combat Module (page 2 of 4 pages)

```
 1660  CONTINUE
       READ(IOUNIT,100) ((((REFIRE(I,J,K,L),L=1,NENVIR),K=1,NTYPS(3-I))
      &   ,J=1,NTYPS(I)),I=1,2)
       DO 1800 I=1,2
       DO 1800 J=1,NTYPS(I)
       DO 1800 K=1,NTYPS(3-I)
       DO 1800 L=1,NENVIR
         IF ( (REFIRE(I,J,K,L).GT.REFIPM) .OR. (REFIRE(I,J,K,L).LT.0.0) )
      &     THEN
           WRITE(OUTFIL,1700) I,J,K,L,REFIRE(I,J,K,L)
 1700      FORMAT(1X,30(1H*),5X,'INDICES= ',4(I5,3X),'REFIRE= ',
      &       F8.2,5X,30(1H*))
           IER=1
         ENDIF
 1800  CONTINUE
C
       READ(IOUNIT,100) ((((SENSOR(I,J,K,L),L=1,2),K=1,NENVIR),
      &    RCDET(I,J),J=1,NTYPS(I)),I=1,2)
       DO 2000 I=1,2
       DO 2000 J=1,NTYPS(I)
       DO 2000 K=1,NENVIR
       DO 2000 L=1,2
         IF ( (SENSOR(I,J,K,L).GT.ISENSR) .OR. (SENSOR(I,J,K,L).LT.0) )
      &     THEN
           WRITE(OUTFIL,1900) I,J,K,L,SENSOR(I,J,K,L)
 1900      FORMAT(1X,30(1H*),5X,'INDICES= ',4(I5,3X),'SENSOR= ',
      &       I5,5X,30(1H*))
           IER=1
         ENDIF
 2000  CONTINUE
       DO 2010 I = 1,2
         DO 2020 J = 1,NTYPS(I)
           IF (RCDET(I,J) .LE. 0.0) THEN
             WRITE (6,2021) I, J, RCDET(I,J)
 2021        FORMAT (1X, 30('*'), 5X, 'INDICES =', 2I4, ' RCDET =',
      &        1PG11.4)
             IER = 1
           ENDIF
 2020    CONTINUE
 2010  CONTINUE
C
       READ(IOUNIT,100) (((SIZE(I,J,K),K=1,NENVIR),J=1,NTYPS(I)),
      &    I=1,2)
       DO 2200 I=1,2
       DO 2200 J=1,NTYPS(I)
       DO 2200 K=1,NENVIR
         IF ( (SIZE(I,J,K).GT.SIZEU) .OR. (SIZE(I,J,K).LT.SIZEL) )
      &     THEN
           WRITE(OUTFIL,2100) I,J,K,SIZE(I,J,K)
 2100      FORMAT(1X,30(1H*),5X,'INDICES= ',3(I5,5X),'SIZE= ',
      &       F8.2,5X,30(1H*))
           IER=1
         ENDIF
 2200  CONTINUE
C
       READ(IOUNIT,100) (LIGHT(I),I=1,NENVIR)
       DO 2400 I=1,NENVIR
         IF ( (LIGHT(I).GT.RLITEM) .OR. (LIGHT(I).LT.0.0) ) THEN
           WRITE(OUTFIL,2300) I,LIGHT(I)
 2300      FORMAT(1X,30(1H*),5X,'INDEX= ',I5,5X,'LIGHT= ',F8.2,
      &       5X,30(1H*))
           IER=1
         ENDIF
 2400  CONTINUE
C
       READ(IOUNIT,100) ((((CNTRST(I,J,K,L),L=1,ISGNTR),K=1,NENVIR)
      &   ,J=1,NTYPS(I)),I=1,2)
       DO 2600 I=1,2
       DO 2600 J=1,NTYPS(I)
       DO 2600 K=1,NENVIR
       DO 2600 L=1,ISGNTR
         IF ( (CNTRST(I,J,K,L).GT.CNTRSU) .OR. (CNTRST(I,J,K,L).LT.
      &     CNTRSL) ) THEN
           WRITE(OUTFIL,2500) I,J,K,L,CNTRST(I,J,K,L)
 2500      FORMAT(1X,30(1H*),5X,'INDICES= ',4(I5,5X),'CNTRST= ',
      &       F8.2,5X,30(1H*))
           IER=1
         ENDIF
 2600  CONTINUE
C
       READ(IOUNIT,100) ((ATTN(I,J),J=1,NENVIR),I=1,ISENSR)
       DO 2800 I=1,ISENSR
```

Figure D-30. Source Listing of the INPUT Subroutine of the Main Program of the AFP Combat Module (page 3 of 4 pages)

```
            DO 2800 J=1,NENVIR
              IF ( (ATTN(I,J).GT.ATTNM) .OR. (ATTN(I,J).LT.0.0) )
     &          THEN
                WRITE(OUTFIL,2700) I,J,ATTN(I,J)
 2700           FORMAT(1X,30(1H*),5X,'INDICES= ',2(I5,3X),'ATTN= ',
     &              F8.2,5X,30(1H*))
                IER=1
              ENDIF
 2800       CONTINUE
C
            READ(IOUNIT,100) (MAG(I),I=1,ISENSR)
            DO 3000 I=1,ISENSR
              IF ( (MAG(I).GT.RMAGM) .OR. (MAG(I).LT.0.0) ) THEN
                WRITE(OUTFIL,2900) I,MAG(I)
 2900           FORMAT(1X,30(1H*),5X,'INDEX= ',I5,5X,'MAG= ',F8.2,
     &              5X,30(1H*))
                IER=1
              ENDIF
 3000       CONTINUE
C
            READ(IOUNIT,100) (BRIGHT(I),I=1,NENVIR)
            DO 3200 I=1,NENVIR
              IF ( (BRIGHT(I).GT.BRIGTM) .OR. (BRIGHT(I).LT.1.0) )
     &            THEN
                WRITE(OUTFIL,3100) I,BRIGHT(I)
 3100           FORMAT(1X,30(1H*),5X,'INDEX= ',I5,5X,'BRIGHT= ',F8.2,
     &          5X,30(1H*))
                IER=1
              ENDIF
 3200       CONTINUE
            IF ( (NOEXT(1)+NOEXT(2)) .EQ. 0 ) GOTO 3500
C       READ IN THE PARTICIPATION RATES
            READ(IOUNIT,*) (((INDPAR(I,J,K),K=1,NTYPS(I)),J=1,NOEXT(I))
     &        ,I=1,2)
            DO 3250 I=1,2
            DO 3250 J=1,NOEXT(I)
            DO 3250 K=1,NTYPS(I)
              IF ( ( INDPAR(I,J,K) .LT. 0.0 ) .OR. ( INDPAR(I,J,K) .GT.
     &          UPRUND ) ) THEN
                WRITE(OUTFIL,3225) I,J,K,INDPAR(I,J,K)
 3225           FORMAT(1X,30(1H*),5X,'INDICES= ',3(I5,3X)
     &            ,'PARTICIPATION= ',F8.4,5X,30(1H*))
                IER=1
              ENDIF
 3250       CONTINUE
            READ(IOUNIT,100)(((((EXT(I,J,K,L,M),M=1,IRANGE),L=1,NENVIR)
     &        ,K=1,NTYPS(3-I)),J=1,NOEXT(I)),I=1,2)
            DO 3400 I=1,2
            DO 3400 J=1,NOEXT(I)
            DO 3400 K=1,NTYPS(3-I)
            DO 3400 L=1,NENVIR
            DO 3400 M=1,IRANGE
              IF ( ( EXT(I,J,K,L,M) .LT. 0.0 ) .OR.
     &           ( EXT(I,J,K,L,M) .GT. UPRUND ) ) THEN
                WRITE(6,3300) I,J,K,L,M,EXT(I,J,K,L,M)
 3300           FORMAT(1X,30(1H*),5X,'INDICES= ',5I5,5X,'EXT= ',
     &              F8.2,5X,30(1H*))
                IER=1
              ENDIF
 3400       CONTINUE
            READ(IOUNIT,100) ((EXTPER(I,J),J=1,NOEXT(I)),I=1,2)
            DO 3430 I=1,2
            DO 3430 J=1,NOEXT(I)
              IF ( (EXTPER(I,J) .LT. 0.0 ) .OR. (EXTPER(I,J) .GT. 1.0 ) ) THEN
                WRITE(6,3420) I,J,EXTPER(I,J)
 3420           FORMAT(1X,30(1H*),5X,'INDICES= ',2I5,5X,'EXTPER= ',F8.2
     &          ,5X,30(1H*))
                IER=1
              ENDIF
 3430       CONTINUE
 3500       CONTINUE
            READ(IOUNIT,100) (((AMOTYP(I,J,K),K=1,NTYPS(3-I)),J=1,NTYPS(I))
     &        ,I=1,2)
            READ(IOUNIT,100) LOGIT,STATE
C CHECK THAT NO DATA IS LEFT
            READ(IOUNIT,10000,END=10100) JUNK
10000       FORMAT(A1)
            WRITE(OUTFIL,10050) JUNK
10050       FORMAT(1X,10(1H*),5X,'LEFTOVER DATA ',A80,5X,10(1H*))
            IER=1
10100       IF ( IER .EQ. 1 ) STOP
            RETURN
            END
```

Figure D-30. Source Listing of the INPUT Subroutine of the Main Program of the AFP Combat Module (page 4 of 4 pages)

```
      SUBROUTINE OPTICS(RC,ACON,ATTN,RNG,LSC,SOG,ALEV,MODE,AMAG)
C
      DIMENSION RCDAT(7),RC3(2)
C
      DATA RCDAT/2.74,2.74,2.29,1.80,1.32,0.47,0.14/
C                                             VISIBLE BAND DEVICES
      RC=0.
      C = ACON / (1.+SOG * (EXP(ATTN * RNG)-1.0))
C
      IF (LSC.EQ.10) GO TO 250
      IF (LSC.EQ.12) GO TO 250
      IF (LSC.EQ.15) GO TO 250
C                                  30% LOST BY OPTICAL
      IF (LSC.EQ.2) ALEV=ALEV*0.7
C
      ACK=100.
C
      IF (ALEV.LT.100.) GO TO 100
C                                                    OVERCAST DAYLIGHT
      IL1=1
      IL2=1
      ALEV=100.
      GO TO 130
C                                                CHECK IF TOO DARK
  100 IF (ALEV.LT.0.0001) GO TO 320
C
      DO 110 II=1,6
      IL1=II
      IF (ALEV.LE.ACK.AND.ALEV.GE.ACK/10.) GO TO 120
      ACK=ACK/10.
  110 CONTINUE
C
  120 IL2=IL1+1
C                                          DETERMINE LIGHT LEVEL RANGE
  130 RC3(1)=0.
      RC3(2)=0.
      DO 240 I=1,2
      IF (I.EQ.1) LIGHT=IL1
      IF (I.EQ.2) LIGHT=IL2
C
      RC3(I)=RCDAT(LIGHT)
C
      IF (C.GT.0.8) GO TO 240
C
      GO TO (140,160,180,190,200,210,220),LIGHT
C                                   (OVERCAST DAY   100 FT CDLS)
  140 IF (C.LE.0.25) GO TO 150
C
      RC3(I)=2.8839221*(C**0.2291015)
C
      GO TO 240
C
  150 IF (C.LT.0.025) GO TO 230
C
      RC3(I)=2.3365166-0.0509533/C
      GO TO 240
C                                   (HEAVILY OVERCAST DAY  10 FT CDL
  160 IF (C.LT.0.35) GO TO 170
C
      RC3(I)=2.8667792*(C**0.2251284)
      GO TO 240
C
  170 IF (C.LT.0.025) GO TO 230
C
      RC3(I)=2.3262999-0.0514585/C
      GO TO 240
C                                  (SUNSET OVERCAST DAY  1.0 FT CDLS
  180 IF (C.GT.0.7) GO TO 240
C
      IF (C.LT.0.03) GO TO 230
C
      RC3(I)=C/(0.0497605+0.3579996*C)
      GO TO 240
C                                 (1/4 HR AFTER SUNSET  0.1 FT CDLS)
  190 IF (C.GT.0.7) GO TO 240
      IF (C.LT.0.05) GO TO 230
C
      RC3(I)=1.6149345-0.0651033/C
C
      GO TO 240
C                                 (1/2 HR AFTER SUNSET 0.01 FT CDLS)
```

Figure D-31. Source Listing of the OPTICS Subroutine of the Main Program of the AFP Combat Module (page 1 of 2 pages)

```
  200 IF (C.LT.0.084) GO TO 230
C
      RC3(I)=1.336422-0.1123916/C
      GO TO 240
C                                                  (MOONLIGHT CLEAR 0.001 FT CDL
  210 IF (C.LT.0.18) GO TO 230
C
      RC3(I)=0.5686696-0.0968436/C
C
      GO TO 240
C                                                  (MOONLIGHT CLEAR NIGHT 0.0001
  220 IF (C.LT.0.5) GO TO 230
C
      RC3(I)=0.29-0.12/C
C
      GO TO 240
C                                             TOO DARK
  230 RC3(I)=0.0
C
  240 CONTINUE
C
      RC1=RC3(1)
      RC2=RC3(2)
C                                                       INTERPOLATE
      RC=RC2+(RC1-RC2)*((ALEV-ACK/10.)/(ACK-ACK/10.))
C
      RC=RC*AMAG
C
      GO TO 320
C
C                                                       SILICON  TELEVISIONS
  250 IF (ALEV .LT. 10. .AND. LSC .NE. 15) GO TO 320
C                                          CHECK  LIMINAL CONTRAST
      C=C*1.63
      IF (C.LT.0.01) GO TO 320
      IF (LSC.EQ.12) GO TO 260
      IF (LSC.EQ.15) GO TO 270
C
      RC=13.43*C/(0.0303+0.1473*C)
C
      IF (MODE.EQ.1) RC=RC/8.
      GO TO 320
  260 RC=8.06*C/(.0206+.2559*C)
      IF (MODE.EQ.2) RC=RC*4.0
      GO TO 320
  270 IF (ALEV.LT..01) GO TO 280
C
      IF (C.LT.0.035) GO TO 320
      IF (C.GE.0.035) RC=-1.25+63.66*C
      IF (C.GE.0.115) RC=3.48+23.39*C
      IF (C.GT.0.24) RC=8.38+3.41*C
      GO TO 310
C
  280 IF (ALEV.LT.0.001) GO TO 290
C
      IF (C.LT.0.05) GO TO 320
      IF (C.GE.0.05) RC=-2.47+57.59*C
      IF (C.GE.0.15) RC=4.13+14.24*C
      IF (C.GT.0.35) RC=8.01+3.17*C
      GO TO 310
C
  290 IF (ALEV.LT.0.0001) GO TO 300
C
      IF (C.LT.0.082) GO TO 320
      IF (C.GE.0.082) RC=-2.12+29.62*C
      IF (C.GT.0.23) RC=3.70+5.42*C
      GO TO 310
C
  300 IF (ALEV.LT.0.00001) GO TO 320
C
      IF (C.LT.0.15) GO TO 320
      IF (C.LE.0.34) RC=-2.14+15.39*C
      IF (C.GT.0.34) RC=1.91+3.66*C
C
  310 IF (MODE.EQ.1) RC=RC/3.0
C
  320 RETURN
      END
```

Figure D-31. Source Listing of the OPTICS Subroutine of the Main Program of the AFP Combat Module
(page 2 of 2 pages)

```
            SUBROUTINE OUTPT(REPSX,DAYSX)
      C
            INCLUDE PAR*
            INCLUDE BK1
            INCLUDE BK2
            INCLUDE BK6
            INCLUDE BK13
      C
            INTEGER SIDEX,TYPSX,TYPESX,OUTFIL,REPSX,DAYSX
      C
      2     CONTINUE
      C
            OUTFIL=6
      C
            IOPTR9=REPSX+(DAYSX-1)*N90
            IOPTR9=IOPTR9+PREF4*IREPS*IDAYS
      CC      WRITE(OUTFIL,100)
      CC100   FORMAT(//,T51,'DIRECT ASSIGNMENTS',/)
      CC      DO 300 SIDEX=1,2
      CC      DO 300 TYPSX=1,NTYPS(SIDEX)
      CC        ISUM=0
      CC        DO 150 TYPESX=1,NTYPS(3-SIDEX)
      CC          ISUM=ISUM+FORCES(SIDEX,TYPSX,TYPESX)
      CC150     CONTINUE
      CC        IF ( ISUM .LE. 0 ) GOTO 300
      CC        WRITE(OUTFIL,200) SIDEX,TYPSX,(FORCES(SIDEX,TYPSX,TYPESX)
      CC     &     ,TYPESX=1,NTYPS(3-SIDEX))
      CC200     FORMAT(1X,I2,3X,I3,/,500(1X,10(I8,2X),/))
      CC300   CONTINUE
      C   WRITE OUT THE FORCE ALLOCATIONS
            WRITE(IU9'IOPTR9,225,ERR=250)
           &     (((FORCES(I,J,K),K=1,NTYPS(3-I)),J=1,NTYPS(I)),I=1,2)
      225   FORMAT(500(500I6))
            GOTO 275
      250   CONTINUE
            WRITE(OUTFIL,260) REPSX,DAYSX
      260   FORMAT(1X,30(1H*),5X,'REP= ',I3,3X,'DAY= ',I3,3X
           &     ,'ALLOCATION WRITEOUT DIRECT ERROR',5X,30(1H*))
            STOP
      275   CONTINUE
            RETURN
            END
```

Figure D-32. Source Listing of the OUTPT Subroutine of the Main Program of the AFP Combat Module

(17) STPREP

(a) Purpose. To prepare the status table and to determine the maximum number of shots in each duel in a conflict. Figure D-33 is a source listing of subroutine STPREP.

(b) When Called. Every time another conflict is scheduled, for every pair of opposing types, environment and range. This routine is called by MAIN.

(c) Subtasks

1. Computes the number of duels in the conflict.

2. Sets the entries for each duel to zero.

3. Stores the odds in each entry.

4. Bounds the number of shots in the conflict by the sum of:

a. Conflict duration/refire time for smaller side.

b. (Conflict duration/refire time) x largest odds for the larger side.

```
          SUBROUTINE STPREP(*,RANGEX,ENVIRX,TYPS1X,TYPS2X)
C
C    !!!!! NSTAGE WILL TRUNCATE THE NUMBER OF STAGES UNLESS !!!!!
C    !!!!!    IT IS SET TO A LARGE NUMBER WHEN RUNNING      !!!!!
C
          INCLUDE PARM
          INCLUDE BK1
          INCLUDE BK2
          INCLUDE BK3
          INCLUDE BK4
          INCLUDE BK11
          INCLUDE BK16
          INCLUDE BK22
C
          DIMENSION TIMES(2)
C
          INTEGER RANGEX,ENVIRX,CONFLX,TYPS1X,TYPS2X,ODDSX
C
2         CONTINUE
C CREATE THE STATUS TABLE ENTRIES
                    CNFLCT=0
                    IF ( REFIRE(1,TYPS1X,TYPS2X,ENVIRX) *
     &                 REFIRE(2,TYPS2X,TYPS1X,ENVIRX) .LT. SMALL2 ) THEN
                      NSTAGE=0
                      GOTO 999
                    ENDIF
                    CONFLX=0
C LOOP THROUGH THE ODDS
                    DO 1700 ODDSX=1,NODDS
C SKIP IF NO CONFLICTS
                      IF ( ODDS(ODDSX) .EQ. 0 ) GOTO 1700
                      IF ( CONFLO(ODDSX,RANGEX,ENVIPX) .LE. 0 )
     &                   GOTO 1700
                      CNFLCT=CONFLO(ODDSX,RANGEX,ENVIRX)+CNFLCT
                      IF ( CNFLCT .EQ. C ) GOTO 1700
                      IF ( CNFLCT .GT. ICONFL ) THEN
                        WRITE(6,1475) CNFLCT,ODDSX,RANGEX,ENVIRX
1475                    FORMAT(1X,30(1H*),4(I5,2X),
     &                    'STATUS TABLE TOO LARGE',5X,30(1H*))
                        STOP
                      ENDIF
1500                  CONTINUE
                      CONFLX=CONFLX+1
C INITIALIZE THE STATUS TABLE ENTRIES
                      DO 1600 I=1,ILEN
                        STATUS(CONFLX,I)=0.0
1600                  CONTINUE
                      STATUS(CONFLX,2)=MIN(IODDS,ODDS(ODDSX))+.1
C                     INITIALZE SHOT COUNTS
                      CSHOTS(1,CONFLX) = 0
                      CSHOTS(2,CONFLX) = 0
C IF NOT FINISHED, GO BACK TO CREATE ANOTHER ENTRY
                      IF ( CONFLX .LT. CNFLCT ) GOTO 1500
1700                CONTINUE
C CHECK FOR NO CONFLICTS LEFT
                    IF ( CNFLCT .EQ. 0 ) GOTO 999
C SEEK MAX NUMBER OF CONFLICTS
                    IF ( CNFLCT.GT.MAXCON )
     &                MAXCON=CNFLCT
C STAGE LOOP
C    CALCULATE THE NUMBER OF SHOTS IN THE TIME ALLOTTED
                    TIMES(1)=PROJCT(TYPS2X,3)/REFIRE(1,TYPS1X,TYPS2X
     &                ,ENVIRX)
                    TIMES(2)=PROJCT(TYPS2X,3)/REFIRE(2,TYPS2X,TYPS1X
     &                ,ENVIRX)
                    TEMP=TIMES(SMALER)+(MINO(IODDS,ODDS(NODDS))
     &                                  *TIMES(BIGGER))
                    NSTAGE=MIN( NSTAGO,IFIX(TEMP + 0.5) )
999               CONTINUE
CCCC   FOLLOWING TRACE ADDED FOR TEST PURPOSES, 18 JAN 84:
CC                  WRITE (6,66) TYPS1X, TYPS2X, RANGEX, NSTAGE
CC66                FORMAT (' ** STPREP: TYPS1X =', I5,
CC   &                ' TYPS2X =', I5, ' RANGEX =', I5,
CC   &                ' NSTAGE =', I5)
CCCC   END OF TEST TRACE
                  IF ((NSTAGE .LE. 0) .OR. (CNFLCT .EQ. 0)) RETURN 1
                  RETURN
                  END
```

Figure D-33. Source Listing of the STPREP Subroutine of the Main Program of the AFP Combat Module

(18) THERMO

(a) Purpose. For specified infrared (thermal) band sensor and conditions, to return the number of resolvable cycles. Figure D-34 is a source listing of subroutine THERMO.

(b) Source. THERMO is an implementation of logic developed at the Night Vision Laboratory.

(c) When Called. From DETECT if the weapon platform in question depends on a thermal band sensor.

```
      SUBROUTINE THERMO(RC,ACON,ATTN,RNG,LSC,MODE)
C
C                                        THERMO DEVICES (6,7,8,9,11,14)
      RC=0.
      C = ACON * EXP(-ATTN * RNG)
C
      IF (LSC.EQ.8) GO TO 330
C                                        TEMPERATURE TOO LOW
      IF (C.LE.0.0112) GO TO 350
C
      IF (LSC.EQ.9) GO TO 340
C
      IF (LSC.EQ.11) GO TO 300
C
      IF (LSC.EQ.14) GO TO 310
      GO TO 320
C                                        NIGHT SIGHTS
  300 RC=C/(0.0207+0.1291*C)
      IF (MODE.EQ.1) RC=RC/3.0
      GO TO 350
C
  310 IF (C.LT.0.037) GO TO 350
      RC=(0.91287*C)/(0.0297567+0.1991449*C)
      IF (MODE.EQ.1) RC=RC/3.0
      GO TO 350
C
  320 IF (C.LE.0.133) RC=C/(0.09772+0.34779*C)
C
      IF (C.GT.0.133.AND.C.LE.8.380) RC=C/(0.08162+0.54786*
     1     C)
C
      IF (C.GT.8.380) RC=2.0
C                                        THERMO DEVICE 6 NARROW FOV
      IF (LSC.EQ.6.AND.MODE.EQ.2) RC=RC*3.0
C
      GO TO 350
C                                        THERMO DEVICE 8
  330 IF (C.LT.0.037) GO TO 350
C
      RC=C/(0.0297567+.1991449*C)
C
      IF (C.GT.2.18) RC=5.1
      IF (MODE.EQ.1) RC=RC/3.0
      GO TO 350
C
C                                        AIRBORNE FLIR
  340 RC=C/(.0289+.1092*C)
C
      IF (MODE.EQ.1) RC=RC/4.0
  350 RETURN
      END
```

Figure D-34. Source Listing of the THERMO Subroutine of the Main Program of the AFP Combat Module

(19) GETPKS

(a) Purpose. To input a set of SSPKs for a single Blue weapon type (TYPE1) versus all Red weapon types. Figure D-35 is a source listing of subroutine GETPKS.

(b) When Called. From MAIN whenever the index of the Side 1 (Blue) direct fire weapon type is incremented.

(c) Task. Retrieve SSPK data as needed from the previously prepared PKS file described in Annex VI of Appendix D and in paragraph D-VII-4 of Annex VII to Appendix D.

```
           SUBROUTINE GETPKS (TYPE1)
      C    ****************************************************************
      C    *                                                              *
      C    * GETPKS: GET PKS FOR A BLUE TYPE VS ALL RED TYPES             *
      C    *                                                              *
      C    ****************************************************************
      C
      C    INPUT ARGUMENT (INTEGER):
      C      TYPE1 = BLUE WEAPON TYPE
      C
      C    NOTE: THIS ROUTINE ASSUMES THAT THE REQUESTS FOR PK'S, AS WELL
      C        AS THE INPUT FILE, ARE SORTED IN ORDER OF INCREASING BLUE TYPE
      C
           INCLUDE PARM,LIST
           INCLUDE BK11,LIST
           INTEGER TYPE1
           IF (KEYU1 .NE. TYPE1) THEN
      C      ZERO PKS
             CALL ZERO (PKS, 2*ITYPS*IPKS*IRANGE)
      C      FLUSH ANY IRRELEVANT INPUT RECORDS
      100    IF (IU1T1 .GE. TYPE1) GOTO 199
               READ (IU1,END=999) IU1S, IU1T1, IU1T2, PK1
             GOTO 100
      199    CONTINUE
      C      PROCESS RECORDS FOR TYPE1, TRANSFERRING DATA TO PKS ARRAY
      200    IF (IU1T1 .NE. TYPE1) GOTO 299
               DO 300 I = 1,IRANGE
                 PKS(IU1S,IU1T2,4,I) = PK1(I)
      300      CONTINUE
      C        GET THE NEXT RECORD
               READ (IU1,END=999) IU1S, IU1T1, IU1T2, PK1
             GOTO 200
      299    CONTINUE
             KEYU1 = TYPE1
           ENDIF
           RETURN
      999  CONTINUE  @ END OF FILE
      C    FORCE INPUT RECORD TYPE1 HIGH
           IU1T1 = 999999
           RETURN
           END
```

Figure D-35. Source Listing of the GETPKS Subroutine of the Main Program of the AFP Combat Module

(20) ZERO. Subroutine ZERO sets all the elements of a real array of length N to 0.0. Figure D-36 is a source listing of subroutine ZERO.

```
          SUBROUTINE ZERO (A,N)
C         *****************************************************************
C         *                                                               *
C         * ZERO: ZERO A REAL MATRIX                                      *
C         *                                                               *
C         *****************************************************************
C
          REAL A(N)
          DO 100 I = 1,N
            A(I) = 0.0
  100     CONTINUE
          RETURN
          END
```

Figure D-36. Source Listing of the ZERO Subroutine of the Main Program of the AFP Combat Module

(21) IZERO. Subroutine IZERO sets all the elements of an integer array of length N to 0. Figure D-37 is a source listing of subroutine IZERO.

```
          SUBROUTINE IZERO (IA,N)
C         *****************************************************************
C         *                                                               *
C         * ZERO: ZERO AN INTEGER MATRIX                                  *
C         *                                                               *
C         *****************************************************************
C
          INTEGER IA(N)
          DO 100 I = 1,N
            IA(I) = 0
  100     CONTINUE
          RETURN
          END
```

Figure D-37. Source Listing of the IZERO Subroutine of the Main Program of the AFP Combat Module

(22) **ALLZ.** Logical subroutine ALLZ tests the elements of a real array of length N for zero value. If all elements are zero, the function returns the value .TRUE. The first nonzero value in the array causes the function to return a value of .FALSE. Figure D-38 is a source listing of logical function ALLZ.

```
      LOGICAL FUNCTION ALLZ (A,N)
C     ****************************************************************
C     *                                                              *
C     * ALLZ: TEST A REAL MATRIX FOR ALL ZERO ELEMENTS               *
C     *                                                              *
C     ****************************************************************
C
      REAL A(N)
      ALLZ = .TRUE.
      DO 100 I = 1,N
        IF (A(I) .NE. 0.0) THEN
          ALLZ = .FALSE.
          GOTO 199
        ENDIF
100   CONTINUE
199   CONTINUE
      RETURN
      END
```

Figure D-38. Source Listing of the ALLZ Subroutine of the Main Program of the AFP Combat Module

(23) **IALLZ.** Logical function subroutine IALLZ tests the elements of an integer array of length N for zero value. If all elements are zero, the function returns the value .TRUE. The first nonzero value in the array causes the function to return a value of .FALSE. Figure D-39 is a source listing of logical function IALLZ.

```
      LOGICAL FUNCTION IALLZ (IA,N)
C     ****************************************************************
C     *                                                              *
C     * ALLZ: TEST AN INTEGER MATRIX FOR ALL ZERO ELEMENTS           *
C     *                                                              *
C     ****************************************************************
C
      INTEGER IA(N)
      IALLZ = .TRUE.
      DO 100 I = 1,N
        IF (IA(I) .NE. 0) THEN
          IALLZ = .FALSE.
          GOTO 199
        ENDIF
100   CONTINUE
199   CONTINUE
      RETURN
      END
```

Figure D-39. Source Listing of the IALLZ Subroutine of the Main Program of the AFP Combat Module

d. MAP Element. Figure D-40 provides a listing of the MAP element for collection of the program elements of the Main Program within the AFP Combat Module.

```
@MAP,E ,H7AFP.870MAIN
IN H7AFP.CNFALC
IN H7AFP.CNFLC1
IN H7AFP.CNFTMS
IN H7AFP.DETECT
IN H7AFP.DIRIO
IN H7AFP.DIRKLS
IN H7AFP.EXTASG
IN H7AFP.EXTKLS
IN H7AFP.FIMSLU
IN H7AFP.FRCALC
IN H7AFP.F2FRT
IN H7AFP.IMAGES
IN H7AFP.INPUT
IN H7AFP.MAIN
IN H7AFP.OPTICS
IN H7AFP.OUTPT
IN H7AFP.STPREP
IN H7AFP.THERMO
IN H7AFP.GETPKS
IN H7AFP.ZERO
IN H7AFP.IZERO
IN H7AFP.ALLZ
IN H7AFP.IALLZ
LIB LIB$*IMSL.
IN C1T057,C34567,CIM46,C34,CI5,CIM67,CM16,CIM1,CI37,CM27,CM47, ;
   CM3,CM126,C36,C67,CM167,CM367,CIM3,CM237,CI45,CM37,C37,CIM678, ;
   CM234,CM67
END
```

Figure D-40. Listing of the MAP Element for Collection of the Program Elements of the Main Program of the AFP Combat Module

ANNEX I TO APPENDIX D

AFP BASEDATA FILE

Section I. OVERVIEW

D-I-1. This annex describes the input file, H7BASEDATA, which contains: the force inventories; data required for the Combat Module detection routine; weapon refire times; indirect fire expected functional damage values; distribution of artillery tubes to range bands; and external or noncombat losses. These data vary by posture and environment, so there are 16 elements within the H7BASEDATA file for every force evaluated. The data are entered into labeled workfiles which are read by programs which construct the 16 environmental elements.

D-I-2. Figure D-I-1 illustrates the process by which the H7BASEDATA file is generated. Source worksheets are developed using references and judgment as appropriate. These data are then keyed into the workfile elements which are clearly labeled to facilitate their use. Eleven programs and system EDITOR routines are used to construct the 16 file elements in the format required by the Combat Module.

H7BASEDATA Organization

	Lines
Miscellaneous data	1-1
Detection based on rounds fired	2-13
Force inventories	14-133
External losses	134-253
Minimum and maximum weapon ranges	254-373
Number of environmental sites	374-374
Indirect shooter range distribution	375-394
Weapon refire times	395-1114
Sensor types, signature sought, resolvable cycles	1115-1234
Sensing size	1235-1354
Light level	1355-1355
Signature emitted	1356-1475
Atmospheric attenuation	1476-1479
Magnification	1480-1483
Sky over ground-brightness	1484-1484
Indirect fire participation	1485-1604
Indirect fire expected fractional damage	1605-2804
Percentage of indirect fire on duels	2805-2806
Ammunition type	2807-3166
Print switches	3167-3167

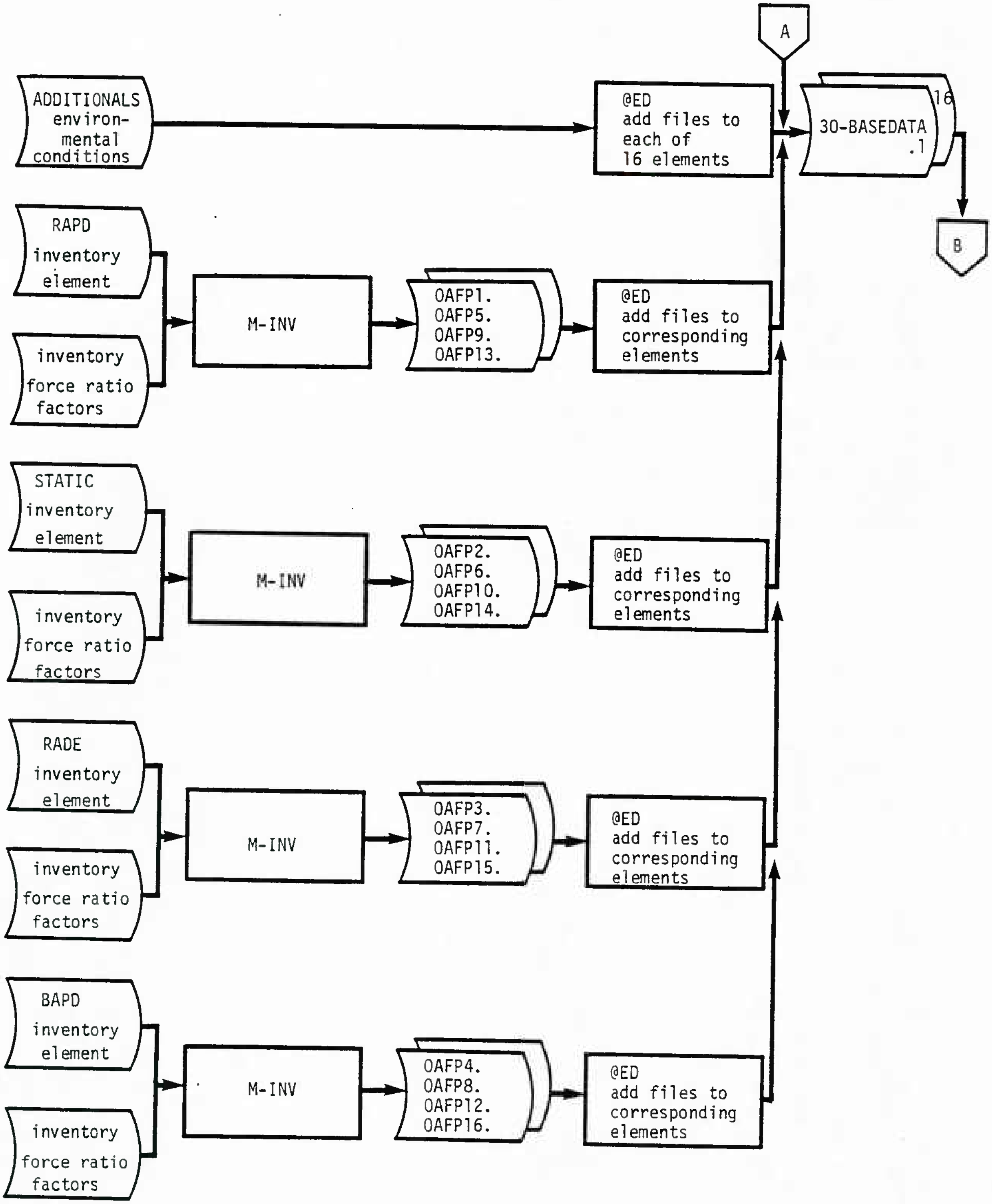


Figure D-I-1. BASEDATA Element Generation
(page 1 of 4 pages)

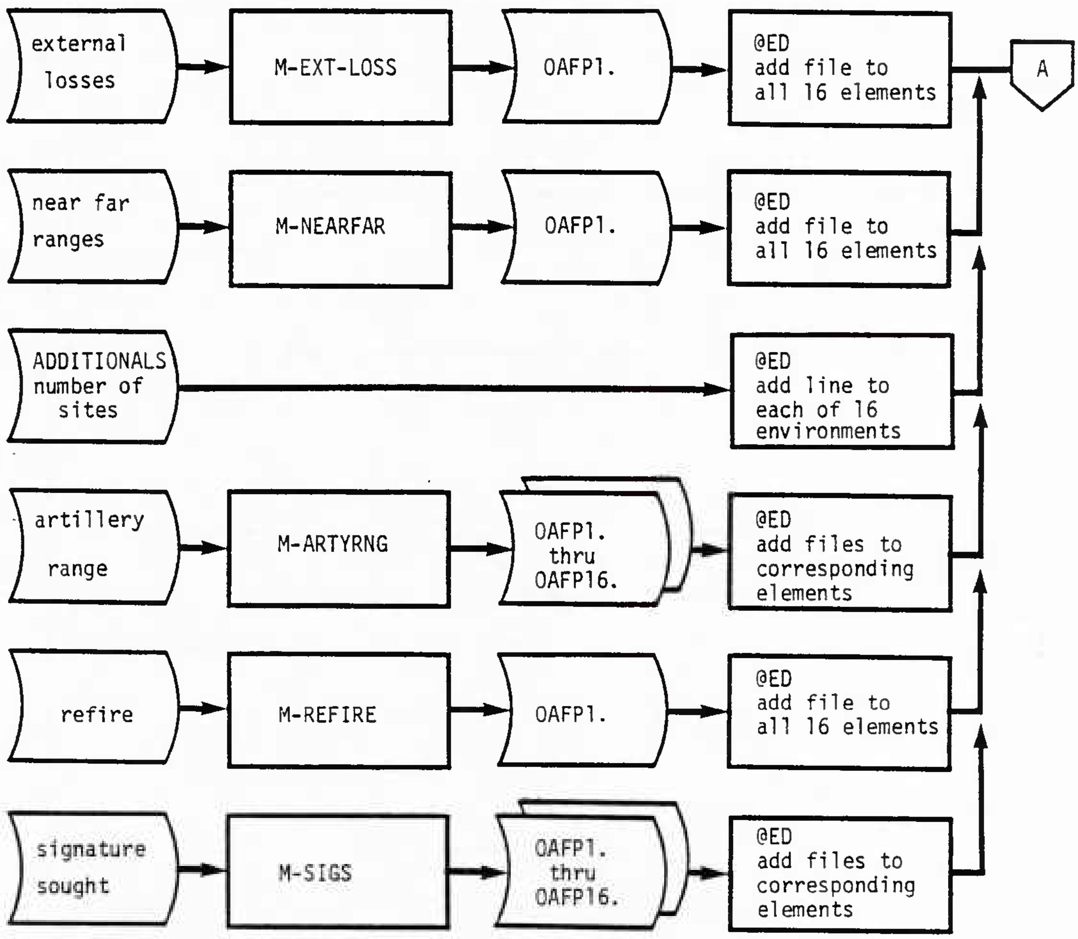


Figure D-I-1. BASEDATA Element Generation
(page 2 of 4 pages)

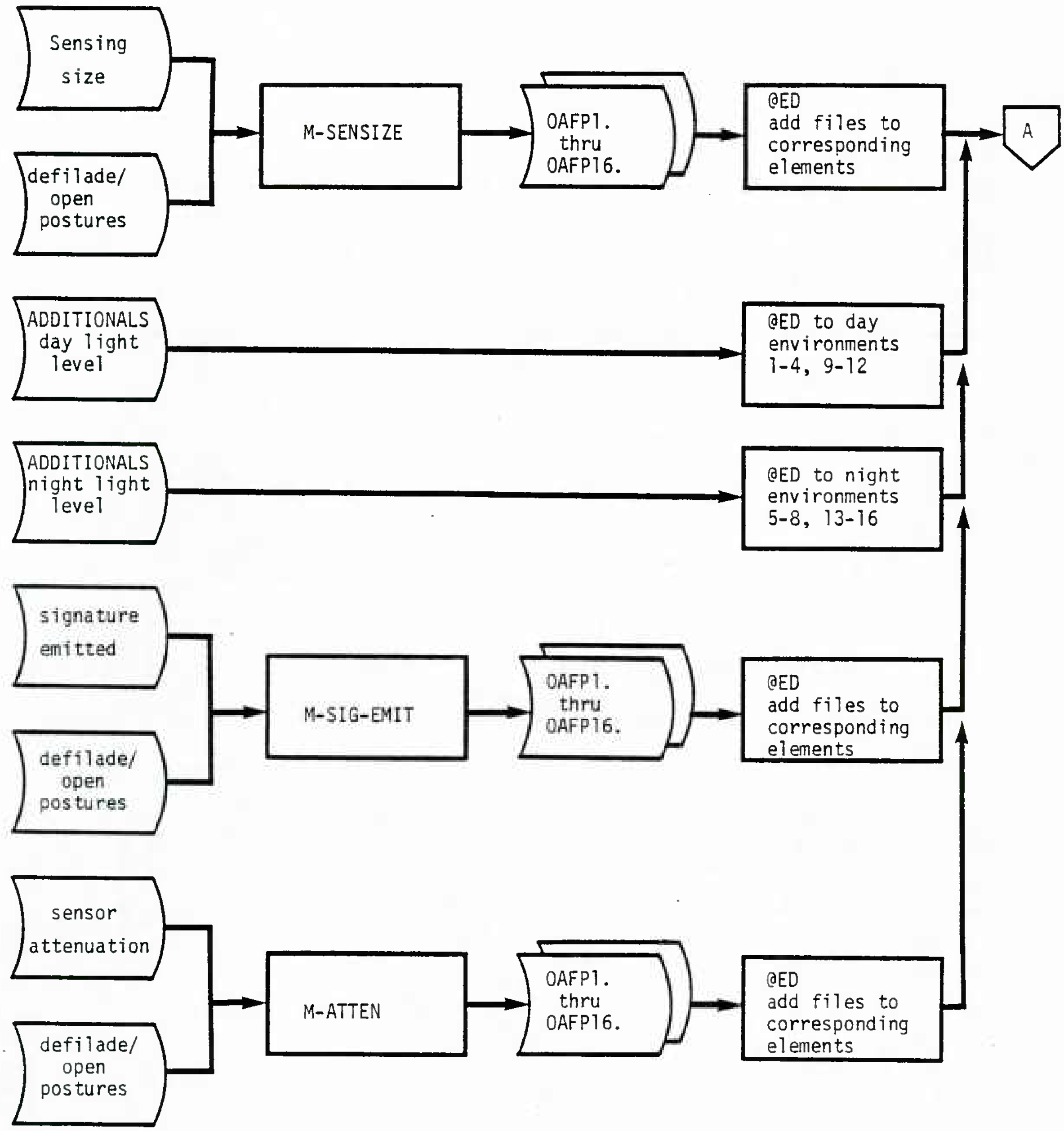


Figure D-I-1. BASEDATA Element Generation
(page 3 of 4 pages)

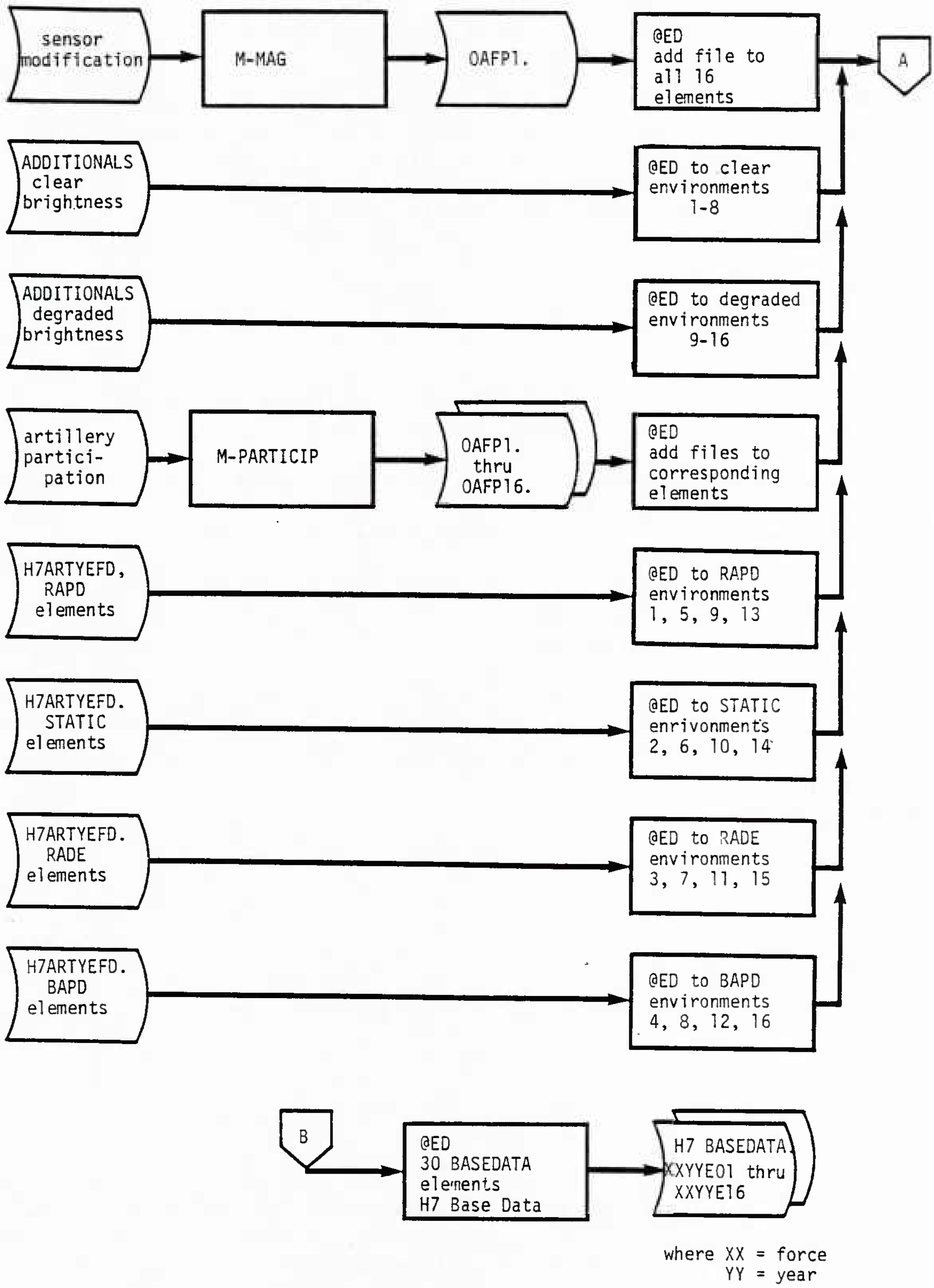


Figure D-I-1. BASEDATA Element Generation
(page 4 of 4 pages)

Table D-I-1 shows the relationship between the H7BASEDATA file (combat module format) and the H7AFPFILES (labeled format) workfiles. One section, indirect fire expected fractional damage, comes from a different file, H7ARTYEFD. The workfile elements contain the data required for all 16 elements of H7BASEDATA. It should be noted that some of the data, such as weapon refire times, are the same for all 16 environments while some data vary by combat posture and other data vary by environmental condition, i.e., day or night and clear and degraded conditions. The programs which build H7BASEDATA select the appropriate data from the workfiles for each AFP environment. Also, there are two workfile elements which define the force ratio for each environment, and the open or defilade condition of each side for all environments.

Table D-I-1. Combat Module Format Basedata With Related Labeled Elements

H7BASEDATA element	H7AFPFILES
Section	Element
Miscellaneous data	ADDITIONALS
Detection based on rounds fired	ADDITIONALS
Force inventories	INVENTORY INV-FACTORS
External losses	EXT-LOSSES
Minimum and maximum weapon ranges	NEAR-FAR
Number of environmental sites	ADDITIONALS
Indirect shooter range distribution	ARTY-RANGE
Weapon refire times	REFIRE
Sensor types, signature sought, resolvable cycles	SIG-SOUGHT
Sensing size	SENSING-SIZE
Light level	ADDITIONALS
Signature emitted	SIG-EMIT
Atmospheric attenuation	SENSOR-ATTEN
Magnification	SENSOR-MAG
Brightness	ADDITIONALS
Indirect fire participation	PARTICIPATE
Indirect fire expected fractional damage	H7ARTYEFD. elements
Percentage of indirect fire on duels	ADDITIONALS
Ammunition type	ADDITIONALS
Print switches	ADDITIONALS

The subsequent sections of this annex describe the workfiles, the program which generates the elements for the 16 environments, runstreams used in the process, and the file elements in the format requisite for input to the Combat Module. Indirect fire expected fractional damage will be explained in Annex D-VIII. The EFDs are generated using a different preprocessor which is documented in that annex.

Section II. INPUT WORKFILE ELEMENTS

D-I-3. ADDITIONALS. The data in H7AFPFILES.Additionals are the miscellaneous values which appear as less than 10 lines in the H7BASEDATA file. Figure D-I-2 is an example of an Additionals element. The first line contains the data which appear as the first line of each H7BASEDATA element. The first entry on line 1 reading from left to right depicts the number of days of AFP combat to be evaluated (2 days in this example). The next two entries (60 60) indicate the number of AFP types on Side 1 and Side 2, respectively. The fourth entry (1) specifies the number of environments followed by the maximum number of stages (4,000). Positions 6 and 7 (10 10) are the number of indirect shooters on Side 1 and Side 2, and the last position (5) indicates the maximum number of unsuccessful detection attempts per shot cycle. Lines 2 through 13 of H7AFPFILES. Additionals, which are also lines 2 through 13 of each element in H7BASEDATA, relate to the number of rounds that a shooter fires before his opponent shoots back--given that the opponent failed to successfully detect via the NVL detection routine. This feature is intended to allow those who fail to detect to return fire based on the perception that they are being fired upon. Lines 2-7 contain the values for Side 1 shooters by type (B01-B60) while lines 8 through 13 contain the values for Side 2 shooters by type (R01-R60). Each entry represents the number of rounds that the shooter (who has detected) will fire before the opponent (who has not detected) will begin firing. For example, the first entry on line 2, (20), means that Side 1, type 1, (or B01), will fire 20 rounds at any nondetecting opponent before the opponent begins returning fire. Line 14 of H7AFPFILES. Additionals appears as line 374 of each H7BASEDATA element. The value, which is a constant 1.00, indicates the number of environmental sites. Since each of the 16 AFP environments have been established as independent runs, the environmental site value has always been 1.00. The next two lines of H7AFPFILES.Additionals (lines 15 and 16) are light level inputs to the detection routine. The value of 600.00 (line 15) is used for the eight daytime environments, and the .0001 value (line 16) is used for the eight nighttime environments. Line 1355 of H7BASEDATA contains the light level input value. Lines 17 and 18 of H7AFPFILES.Additionals contain the sky over ground or brightness values for clear and degraded environmental conditions. The value of 3.00 (line 17) is used for clear conditions, day or night, and 1.00 (line 18) is used for degraded conditions. Line 1484 of each H7BASEDATA element contains the appropriate brightness value. The next two lines of the H7AFPFILES.Additionals element pertain to the percentage of each indirect shooter's fire allocated to the direct fire duels. Line 19 contains the values for the 10 Side 1 indirect shooters while line 20 contains the 10 Side 2 values. The 10 shooters for each side correspond to AFP types B51 through B60 and R51 through R60. The entry of .7 in the first position on line 19 means that 70 percent of type B51 fire would be allocated to direct fire duels and that the remaining 30 percent would be involved in counterbattery or countermortar fire missions. The values for percentage of indirect fire allocation are contained in lines 2805 and 2806 of each H7BASEDATA element. Line 2805 contains the values for Side 1, types 51-60, and line 2806 represents Side 2 input for types 51-60. Lines 21 through 380 contain the ammunition type which are currently set to a default value of 1. The last line, line 381, has two logical variables used as print switches. If the first is TRUE, the conflicts log file is written to output. If the second is TRUE, other debug information is written.

Table D-I-2 summarizes the ADDITIONALS sections.

Table D-I-2. ADDITIONALS Element Sections

Section	Lines
Number of days of combat	1
Number of Side 1 AFP types	1
Number of Side 2 AFP types	1
Number of environmental sites	1
Maximum number of stages	1
Number of Side 1 indirect shooters	1
Number of Side 2 indirect shooters	1
Maximum number of unsuccessful detection attempts	1
Detection based on rounds fired	2-13
Number of environmental sites	14
Daytime light level	15
Nighttime light level	16
Brightness for clear conditions	17
Brightness for degraded conditions	18
Percentage indirect shooters fire allocated to direct fire (Side 1)	19
Percentage indirect shooters fire allocated to direct fire (Side 2)	20
Ammunition type	21-380
Print switches	381

```
  1         2   60   60    1       4000   10   10    5  (MISC DATA)
  2     20 20 5 5 5 1 1 1 1 1                           (DETECTION)
  3     1 1 1 1 1 1 1 1 1 1
  4     1 1 1 1 1 1 1 1 1 1
  5     1 1 1 1 1 1 1 1 1 1
  6     1 1 1 1 1 1 1 1 1 1
  7     1 1 1 1 1 1 1 1 1 1
  8     20 20 5 5 5 1 1 1 1 1
  9     1 1 1 1 1 1 1 1 1 1
 10     1 1 1 1 1 1 1 1 1 1
 11     1 1 1 1 1 1 1 1 1 1
 12     1 1 1 1 1 1 1 1 1 1
 13     1 1 1 1 1 1 1 1 1 1
 14      1.00                                           (# SITES)
 15         600.00                                      (LIGHT DAY)
 16         0.001                                       (LIGHT NIGHT)
 17           3.00                                      (CLEAR)
 18           1.00                                      (DEGRADE)
 19      .7  .7  .7 1. 1. 1. .6 .1 1. 1.                (% DIRECT FIRE)
 20      .7  .7  .7 1. 1. 1. .6 .1 1. 1.
 21       1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1(AMMO)
LINES 22 - 379 HAVE BEEN OMITTED TO SHORTEN THIS FIGURE
THE VALUES ARE IDENTICAL TO LINE 21

380       1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
381     .FALSE. .FALSE.                                 (PRT SWITCHES)
```

Figure D-I-2. ADDITIONALS

D-I-4. INVENTORIES. Figure D-I-3 illustrates the labeled version of the force inventories which are elements in the file H7AFPFILES. Each inventory element contains the number of Side 1 and Side 2 types to be evaluated in the Combat Module. The elements are presently structured for 60 types per side. B01 is the AFP designation for Side 1, type 1, while R01 represents Side 2, type 1. As shown at Figure D-I-3, the AFP number is in columns 1 through 3. If the item is "super trooped," there is an asterisk (*) immediately following the AFP number. See paragraph D-I-5 for an explanation of "super troops." Following the AFP number is a 10-character name of the type equipment being input. The next column depicts the number of items to be input to the Combat Module for one division. The number of divisions on Side 1 and Side 2 vary with the combat postures. A separate element in H7AFPFILES called INV-FACTORS is used to multiply the values contained in the inventory element to produce the desired inventory values by environment. Figure D-I-4 shows the multiplication factors for the Blue (Side 1) and Red (Side 2) divisions by environment. As an example, for environment 1 which is Red Attack-Blue Prepared Defense, each item in the Red inventory would be multiplied by 3 to produce the desired force ratio of three Red divisions to one Blue division. Returning to Figure D-I-3, the final column is used to flag inventory values not to be increased by the multiplication factors. An entry of "N" in this column will result in an environmental element where the value used in the #Items/Div column is used for the type flagged while all other types are multiplied by the number of divisions as appropriate.

D-I-5. SUPER TROOPS. Certain inventory values are divided by 10 to expedite processing time by limiting the number of identical type-on-type duels and to facilitate use of the allocation routine which is greatly influenced by inventory density. The AFP types divided by 10 are referred to as "super troops." In Figure D-I-3, the number of effective M-16s is shown as 107. Since B02 is one of the "super-trooped" types as indicated by the asterisk in the AFP # column, the real value is 1,070 rather than 107. Lines 14 through 133 of each H7BASEDATA element contain the number of each type to be input to the Combat Module. The values are for Side 1, Types 1-60, followed by Side 2, Types 1-60.

D-I-6. EXTERNAL LOSSES. Figure D-I-5 illustrates the EXT-LOSSES element of H7AFPFILES. This element contains, for each type on each side, the number of noncombat losses. These losses are deducted from the inventory values for each AFP type prior to the allocation process and are intended to model effects such as broken tracks which preclude combat engagement. To date, this feature has not been used. The external loss input values appear on lines 134 through 253 of H7BASEDATA and are sequenced as Side 1, types 1-60, followed by Side 2, types 1-60.

D-I-7. NEAR-FAR. Figure D-I-6 is an example of the NEAR-FAR element of H7AFPFILES. The element contains the AFP number, description, and weapon minimum and maximum range band value for each type on Side 1 followed by Side 2. The file is used by the AFP Combat Module to prevent unnecessary subroutine calls. The near and far values for B02 in the example mean that an M16 has SSPKs in range bands 1 (250 meters) and 2 (500 meters) but not in the remaining four bands. The far-band value must be at least equal to the greatest range at which the type weapon has an SSPK value. An input value for a range band which exceeds the SSPK range will have no effect of the Combat Module results but will cause larger processing time than is necessary. The NEAR-FAR input values are lines 254 through 373 of each H7BASEDATA element. The values are for Side 1, types 1 through 60, followed by Side 2, types 1 through 60.

D-I-8. INDIRECT SHOOTER RANGE DISTRIBUTION. Figure D-I-7 illustrates the indirect shooter range distribution element, H7AFPFILES.ARTY-RANGE. Each indirect shooter, B51 through B60 and R51 through R60, directs a percentage of its fire at a certain range. Those percentages, which vary by posture and day/night conditions, are shown in tables. Dummy shooters have an even distribution across all ranges. The correspondence between the conditions (posture, day/night) and the environments is hardwired in the labeled to Combat Module conversion program. There is no program to convert module format to labeled.

D-I-9. REFIRE TIME. Figure D-I-8 illustrates the REFIRE element of H7AFPFILES. The values in this element represent the time between rounds expressed in minutes. The element shows for each of the Side 1 and Side 2 shooters the weapon used against each target and the refire time for the weapon. Each AFP type shooter may fire one or more weapons but may fire only one weapon per target. In the example shown, the shooter is B16 or the IFV. The Bushmaster is used against personnel targets while the ATGM is used against light and heavy armored vehicles. The times between rounds for the Bushmaster and TOW are .05 minutes and 1 minute, respectively. The default value of 6.66 is used as the refire time for opposing types which the shooter does not engage. The refire times are contained in lines 395 through 1114 of each H7BASEDATA element.

D-I-10. SIGNATURE SOUGHT. Figure D-I-9 illustrates H7AFPFILES.SIG-SOUGHT, the element which specifies the type of sensor to be used for each AFP type as well as the type of signature sought by the sensor and the number of resolvable cycles required for successful detection. The sensor numbers (SEN # column) correspond to the sensors listed in the sensor attenuation element illustrated in Figure D-I-13. The SIG column is the type signature sought; 1=light contrast, 2=delta temperature, and 3=silicon. The RC column is the resolvable cycles required for detection. The sensors used vary by visibility conditions, clear day, clear night, degraded day, and degraded night. Only one sensor may be used at one time. However, sensors may be changed when conditions change. Night sights are used for night environments. This element is classified SECRET. However, the values in Figure D-I-9 have been changed to keep this documentation UNCLASSIFIED.

1	AFP #	EQUIPMENT	# ITEMS/DIV	MULTIP?
2				
3	B01 *	BTRP DEEP	389	
4	B02 *	M-16	107	
5	B03 *	7.62GM	39	
6	B04 *	MG50GM	0	
7	B05 *	M203	75	
8	B06 *	SAW	0	
9	B07	M113	460	
10	B08	90MMRR	0	
11	B09	106MRR	0	
12	B10	DUMMY	0	
13	B11	DRAGON	278	
14	B12	ITV-2	0	
15	B13	VIPER	0	
16	B14	LAW	135	
17	B15	ITV-1	146	
18	B16	IFV	0	
19	B17	CFV	0	
20	B18	1/4T TOW I	0	
21	B19	M48A5	0	
22	B20	M1E1	0	
23	B21	M551	0	
24	B22	M60A3	0	
25	B23	M1	0	
26	B24	M60A1	291	
27	B25	M60A2	0	
28	B26	DIVAD	0	
29	B27	VULC1	24	
30	B28	1/4T TOWII	0	
31	B29	HMV W TOWII	0	
32	B30	FAV TOW I	0	
33	B31	STINGER	0	
34	B32	CHAP1	24	
35	B33	REDEYE	83	
36	B34	FAV TOWII	0	
37	B35	LAV25 TOWII	0	
38	B36	AH-1S	42	
39	B37	AH-64	0	
40	B38	AH-1G	0	
41	B39	M113 W M19	0	
42	B40	HMV W M19	0	
43	B41	A10	24	N
44	B42 *	LT VEH D	44	
45	B43	HVY ARM D	22	
46	B44 *	LT ARM D	18	
47	B45	UH-60A	0	
48	B46	CEV	8	
49	B47	UH-1H	25	
50	B48	OH-58	52	
51	B49	UC3V	0	
52	B50	M081M	0	
53	B51	M081S	54	
54	B52	M4.2M	0	
55	B53	M4.2S	53	
56	B54	M060	0	
57	B55	H-155T	0	
58	B56	H155S	54	
59	B57	H203S	12	
60	B58	MLRS	0	
61	B59	H-105T	0	
62	B60	H-203T	0	
63	R01 *	RTPP DEEP	509	
64	R02 *	AKS-74	160	
65	R03 *	7.60GM	51	
66	R04	RSNIPE	63	
67	R05	GREN30	21	

Figure D-I-3. Labeled Inventory Element
(page 1 of 2 pages)

```
R06        BRDM-2              31
R07        BTR-60             137
R08        BMP-R                6
R09        DUMMY                0
R10        DUMMY                0
R11        SPG-9                0
R12        RPG-16             140
R13        AT-5                 9
R14        AT-7                28
R15        RPG 7              160
R16        BMP2M              142
R17        DUMMY                0
R18        DUMMY                0
R19        DUMMY                0
R20        DUMMY                0
R21        T55                 81
R22        T62                 13
R23        T64                 42
R24        T72                 90
R25        T80                 26
R26        ZSU-23              16
R27        DUMMY                0
R28        DUMMY                0
R29        DUMMY                0
R30        SA-14               39
R31        SA-6                 7
R32        SA-7                64
R33        SA-8                 8
R34        SA-9                 8
R35        SA-11                0
R36        HOPLITE              2
R37        HIP-E                2
R38        HIND-D              12
R39        DUMMY                0
R40        DUMMY                0
R41        MIG-21              24           N
R42 *      LT VEH D            88
R43        HVY ARM D           18
R44 *      LT ARM D             4
R45        DUMMY                0
R46        ACRV-2 C&C           0
R47        DUMMY                0
R48        DUMMY                0
R49        DUMMY                0
R50        DUMMY                0
R51        M082M                4
R52        M120T               57
R53        DUMMY                0
R54        DUMMY                0
R55        DUMMY                0
R56        H122T/S            100
R57        H152T/S             20
R58        L122T+              18
R59        DUMMY                0
R60        DUMMY                0

NOTES FOR INVENTORY FILE:
(1) THE # ITEMS/DIV IS READ WITH A FORMATTED READ
    AND IS THEREFORE COLUMN DEPENDENT.  MAKE SURE
    THE NUMBER ENDS IN COL 28.
(2) MULTIP? COLUMN IS FOR ITEMS THAT SHOULD NOT BE
    MULTIPLIED BY THE FORCE RATIO
    AN 'N' IN COLUMN 40 WILL CAUSE # ITEMS/DIV TO BE
    MOVED DIRECTLY TO THE BASEDATA FILE
(3) AFP # FOLLOWED BY * IN COL 5 INDICATES THE ITEM IS
    A SUPERTROOP.  THE # ITEMS/DIV HAS BEEN DIVIDED BY 10.
```

Figure D-I-3. Labeled Inventory Element
(page 2 of 2 pages)

ENVIRON	# BLUE DIVS	# RED DIVS	POSTURE
1 DAY CLEAR	1	3	RAPD
2 DAY CLEAR	1	1	STATIC
3 DAY CLEAR	1	4	RADE
4 DAY CLEAR	3	1	BAPD
5 NIGHT CLEAR	1	3	RAPD
6 NIGHT CLEAR	1	1	STATIC
7 NIGHT CLEAR	1	4	RADE
8 NIGHT CLEAR	3	1	BAPD
9 DAY DEGRADE	1	3	RAPD
10 DAY DEGRADE	1	1	STATIC
11 DAY DEGRADE	1	4	RADE
12 DAY DEGRADE	3	1	BAPD
13 NIGHT DEGRADE	1	3	RAPD
14 NIGHT DEGRADE	1	1	STATIC
15 NIGHT DEGRADE	1	4	RADE
16 NIGHT DEGRADE	3	1	BAPD

Figure D-I-4. Force Multiplication Factors

EXTERNAL LOSSES FILES

AFP #	EQUIPMENT	LOSSES
B01	BTRP	0.0
B02	M-16	0.0
B03	7.62GM	0.0
B04	MG50GM	0.0
B05	M203	0.0
B06	SAW	0.0
B07	M113	0.0
B08	DUMMY	0.0
B09	DUMMY	0.0
B10	M113 w TOW	0.0
B11	DRAGON	0.0
B12	ITV	0.0
B13	VIPER	0.0
B14	LAW	0.0
B15	DUMMY	0.0
B16	IFV	0.0
B17	CFV	0.0
B18	DUMMY	0.0
B19	DUMMY	0.0
B20	M1E1	0.0
B21	M551	0.0
B22	M60A3	0.0
B23	M1	0.0
B24	M60A1	0.0
B25	M60A2	0.0
B26	DIVAD	0.0
B27	VULC1	0.0
B28	DUMMY	0.0
B29	DUMMY	0.0
B30	DUMMY	0.0
B31	STINGER	0.0
B32	CHAP1	0.0
B33	REDEYE	0.0
B34	DUMMY	0.0
B35	DUMMY	0.0
B36	AH-1S	0.0
B37	AH-64	0.0
B38	DUMMY	0.0
B39	DUMMY	0.0
B40	DUMMY	0.0
B41	A10	0.0
B42	DUMMY	0.0
B43	DUMMY	0.0
B44	DUMMY	0.0
B45	UH-60A	0.0
B46	CEV	0.0
B47	UH-1H	0.0
B48	OH-58	0.0
B49	UC3V	0.0
B50	M081M	0.0
B51	M081S	0.0
B52	M4.2M	0.0
B53	M4.2S	0.0
B54	DUMMY	0.0
B55	DUMMY	0.0
B56	H155S	0.0
B57	H203S	0.0
B58	MLRS	0.0
B59	DUMMY	0.0
B60	DUMMY	0.0

Figure D-I-5. Labeled External Loss Element
(page 1 of 2 pages)

R01	RTRP	0.0
R02	AKS-74	0.0
R03	7.60GM	0.0
R04	RSNIPE	0.0
R05	GREN30	0.0
R06	BRDM-2	0.0
R07	BTR-60	0.0
R08	BMP-R	0.0
R09	DUMMY	0.0
R10	DUMMY	0.0
R11	SPG-9	0.0
R12	RPG-16	0.0
R13	AT-5	0.0
R14	AT-7	0.0
R15	RPG 7	0.0
R16	BMP2M	0.0
R17	DUMMY	0.0
R18	DUMMY	0.0
R19	DUMMY	0.0
R20	DUMMY	0.0
R21	T55	0.0
R22	T62	0.0
R23	T64	0.0
R24	T72	0.0
R25	T80	0.0
R26	ZSU-23	0.0
R27	DUMMY	0.0
R28	DUMMY	0.0
R29	DUMMY	0.0
R30	SA-14	0.0
R31	SA-6	0.0
R32	SA-7	0.0
R33	SA-8	0.0
R34	SA-9	0.0
R35	SA-11	0.0
R36	HOPLITE	0.0
R37	HIP-E	0.0
R38	HIND-D	0.0
R39	DUMMY	0.0
R40	DUMMY	0.0
R41	MIG-21	0.0
R42	DUMMY	0.0
R43	DUMMY	0.0
R44	DUMMY	0.0
R45	DUMMY	0.0
R46	ACRV-2 C&C	0.0
R47	DUMMY	0.0
R48	DUMMY	0.0
R49	DUMMY	0.0
R50	DUMMY	0.0
R51	M082M	0.0
R52	M120T	0.0
R53	DUMMY	0.0
R54	DUMMY	0.0
R55	DUMMY	0.0
R56	H122T/S	0.0
R57	H152T/S	0.0
R58	L122T+	0.0
R59	DUMMY	0.0
R60	DUMMY	0.0

Figure D-I-5. Labeled External Loss Element
(page 2 of 2 pages)

NEAR FAR RANGE BANDS FILE

AFP #	EQUIPMENT	RANGE BANDS: NEAR	FAR
B01	STRP	6	6
B02	M-16	1	2
B03	7.62GM	1	3
B04	MG50GM	1	4
B05	M203	1	2
B06	SAW	1	2
B07	M113	1	4
B08	90MMRR	1	2
B09	DUMMY	1	5
B10	DUMMY	1	5
B11	DRAGON	1	3
B12	ITV-2	1	5
B13	VIPER	1	2
B14	LAW	1	2
B15	ITV-1	1	5
B16	IFV	1	5
B17	CFV	1	5
B18	1/4T TOWI	1	5
B19	M48A5	1	5
B20	M1E1	1	5
B21	M551	1	5
B22	M60A3	1	5
B23	M1	1	5
B24	M60A1	1	5
B25	M60A2	1	5
B26	DIVAD	1	5
B27	VULC1	1	5
B28	1/4T TOWII	1	5
B29	DUMMY	1	5
B30	DUMMY	1	5
B31	STINGER	1	5
B32	CHAP 1	1	5
B33	REDEYE	1	5
B34	DUMMY	1	5
B35	DUMMY	1	5
B36	AH-1S	1	5
B37	AH-64	1	5
B38	DUMMY	1	5
B39	DUMMY	1	5
B40	DUMMY	1	5
B41	A10	1	5
B42	LT VEH DEEP	6	6
B43	HVY ARM DEEP	6	6
B44	LT ARM DEEP	6	6
B45	UH-60A	1	5
B46	CEV	1	5
B47	UH-1H	1	5
B48	OH-58	1	5
B49	UC3V	1	5
B50	M081M	1	6
B51	M081S	1	5
B52	M4.2M	1	5
B53	M4.2S	1	5
B54	DUMMY	1	6
B55	DUMMY	1	6
B56	H155S	1	6
B57	H203S	1	6
B58	MLRS	1	6
B59	DUMMY	1	6

Figure D-I-6. Labeled NEAR-FAR Element
(page 1 of 2 pages)

B60	DUMMY	1	6
R01	RTRP	6	6
R02	AKS-74	1	2
R03	7.60GM	1	3
R04	RSNIPE	1	3
R05	GREN30	1	2
R06	BRDM-2	1	4
R07	BTR-60	1	4
R08	BMP-R	1	4
R09	DUMMY	1	5
R10	DUMMY	1	5
R11	SPG-9	1	3
R12	RPG-16	1	2
R13	AT-5	1	5
R14	AT-7	1	3
R15	RPG 7	1	1
R16	BMP2M	1	5
R17	DUMMY	1	5
R18	DUMMY	1	5
R19	DUMMY	1	5
R20	DUMMY	1	5
R21	T55	1	5
R22	T62	1	5
R23	T64	1	5
R24	T72	1	5
R25	T80	1	5
R26	ZSU-23	1	5
R27	DUMMY	1	5
R28	DUMMY	1	5
R29	DUMMY	1	5
R30	SA-14	1	5
R31	SA-6	1	5
R32	SA-7	1	5
R33	SA-8	1	5
R34	SA-9	1	5
R35	SA-11	1	5
R36	HOPLITE	1	5
R37	HIP-E	1	5
R38	HIND-D	1	5
R39	DUMMY	1	5
R40	DUMMY	1	5
R41	MIG-21	1	5
R42	LT VEH DEEP	6	6
R43	HVY ARM DEEP	6	6
R44	LT ARM DEEP	6	6
R45	DUMMY	1	5
R46	ACRV-2 C&C	1	5
R47	DUMMY	1	5
R48	DUMMY	1	5
R49	DUMMY	1	5
R50	DUMMY	1	5
R51	M082M	1	5
R52	M120T	1	5
R53	DUMMY	1	5
R54	DUMMY	1	6
R55	DUMMY	1	6
R56	H122T/S	1	6
R57	H152T/S	1	6
R58	L122T+	1	6
R59	DUMMY	1	6
R60	DUMMY	1	6

END OF FILE

Figure D-I-6. Labeled NEAR-FAR Element
(page 2 of 2 pages)

ARTY RANGE DISTRIBUTION FILES

AFP #	TYPE	RAPD DAY (ENVIRON 1,9) 250	500	1000	1500	2500	DEEP	RAPD NIGHT (ENVIRON 5,13) 250	500	1000	1500	2500	DEEP
B51	M081S	.70	.20	.10	.0	.0	.0	.70	.20	.10	.0	.0	.0
B52	M4.2M	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
B53	M4.2S	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
B54	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B55	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B56	H155S	.25	.25	.15	.20	.10	.05	.25	.25	.15	.20	.10	.05
B57	H203S	.15	.15	.00	.00	.00	.70	.15	.15	.00	.00	.00	.70
B58	MLRS	.0	.0	.0	.0	.10	.90	.0	.0	.0	.0	.10	.90
B59	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B60	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R51	M082M	.70	.20	.10	.0	.0	.0	.70	.20	.10	.0	.0	.0
R52	M120T	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
R53	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R54	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R55	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R56	H122T/S	.25	.25	.15	.20	.10	.05	.25	.25	.15	.20	.10	.05
R57	H152T/S	.13	.13	.12	.16	.06	.40	.13	.13	.12	.16	.06	.40
R58	L122T+	.20	.25	.30	.10	.10	.05	.20	.25	.30	.10	.10	.05
R59	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R60	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17

AFP #	TYPE	STATIC DAY (ENVIRON 2,10) 250	500	1000	1500	2500	DEEP	STATIC NIGHT (ENVIRON 6,14) 250	500	1000	1500	2500	DEEP
B51	M081S	.70	.20	.10	.0	.0	.0	.70	.20	.10	.0	.0	.0
B52	M4.2M	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
B53	M4.2S	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
B54	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B55	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B56	H155S	.25	.25	.15	.20	.10	.05	.25	.25	.15	.20	.10	.05
B57	H203S	.15	.15	.00	.00	.00	.70	.15	.15	.00	.00	.00	.70
B58	MLRS	.0	.0	.0	.0	.10	.90	.0	.0	.0	.0	.10	.90
B59	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B60	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R51	M082M	.70	.20	.10	.0	.0	.0	.70	.20	.10	.0	.0	.0
R52	M120T	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
R53	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R54	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R55	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R56	H122T/S	.25	.25	.15	.20	.10	.05	.25	.25	.15	.20	.10	.05
R57	H152T/S	.13	.13	.12	.16	.06	.40	.13	.13	.12	.16	.06	.40
R58	L122T+	.20	.25	.30	.10	.10	.05	.20	.25	.30	.10	.10	.05
R59	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R60	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17

AFP #	TYPE	RADE DAY (ENVIRON 3,11) 250	500	1000	1500	2500	DEEP	RADE NIGHT (ENVIRON 7,15) 250	500	1000	1500	2500	DEEP
B51	M081S	.70	.20	.10	.0	.0	.0	.70	.20	.10	.0	.0	.0
B52	M4.2M	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
B53	M4.2S	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
B54	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B55	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B56	H155S	.25	.25	.15	.20	.10	.05	.25	.25	.15	.20	.10	.05
B57	H203S	.15	.15	.00	.00	.00	.70	.15	.15	.00	.00	.00	.70
B58	MLRS	.0	.0	.0	.0	.10	.90	.0	.0	.0	.0	.10	.90
B59	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B60	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R51	M082M	.70	.20	.10	.0	.0	.0	.70	.20	.10	.0	.0	.0
R52	M120T	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
R53	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R54	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R55	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R56	H122T/S	.25	.25	.15	.20	.10	.05	.25	.25	.15	.20	.10	.05
R57	H152T/S	.13	.13	.12	.16	.06	.40	.13	.13	.12	.16	.06	.40
R58	L122T+	.20	.25	.30	.10	.10	.05	.20	.25	.30	.10	.10	.05
R59	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R60	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17

AFP #	TYPE	BAPD DAY (ENVIRON 4,12) 250	500	1000	1500	2500	DEEP	BAPD NIGHT (ENVIRON 8,16) 250	500	1000	1500	2500	DEEP
B51	M081S	.70	.20	.10	.0	.0	.0	.70	.20	.10	.0	.0	.0
B52	M4.2M	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
B53	M4.2S	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
B54	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B55	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B56	H155S	.25	.25	.15	.20	.10	.05	.25	.25	.15	.20	.10	.05
B57	H203S	.15	.15	.00	.00	.00	.70	.15	.15	.00	.00	.00	.70
B58	MLRS	.0	.0	.0	.0	.10	.90	.0	.0	.0	.0	.10	.90
B59	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
B60	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R51	M082M	.70	.20	.10	.0	.0	.0	.70	.20	.10	.0	.0	.0
R52	M120T	.60	.20	.10	.10	.0	.0	.60	.20	.10	.10	.0	.0
R53	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R54	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R55	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R56	H122T/S	.25	.25	.15	.20	.10	.05	.25	.25	.15	.20	.10	.05
R57	H152T/S	.13	.13	.12	.16	.06	.40	.13	.13	.12	.16	.06	.40
R58	L122T+	.20	.25	.30	.10	.10	.05	.20	.25	.30	.10	.10	.05
R59	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17
R60	DUMMY	.17	.17	.16	.16	.17	.17	.17	.17	.16	.16	.17	.17

Figure D-I-7. Labeled Indirect Shooter Range Distribution Element

B16	IFV	BUSH25	RTRP DEEP	0.05
		BUSH25	AKS-74	0.05
		BUSH25	7.62GM	0.05
		BUSH25	RSNIPE	0.05
		BUSH25	GREN30	0.05
		BUSH25	BRDM-2	1.00
		BUSH25	BTR-60	1.00
		BUSH25	BMP-R	1.00
			DUMMY	6.66
			DUMMY	6.66
		BUSH25	SPG-9	.05
		BUSH25	RPG-16	.05
		BUSH25	AT-5	1.00
		BUSH25	AT-7	0.05
		BUSH25	RPG 7	0.05
		BUSH25	BMP2M	1.00
			DUMMY	6.66
			DUMMY	6.66
			DUMMY	6.66
			DUMMY	6.66
		ATGM	T55	1.00
		ATGM	T62	1.00
		ATGM	T64	1.00
		ATGM	T72	1.00
		ATGM	T80	1.00
		ATGM	ZSU-23	1.00
			DUMMY	6.66
			DUMMY	6.66
			DUMMY	6.66
		BUSH25	SA-14	0.05
			SA-6	6.66
		BUSH25	SA-7	0.05
			SA-8	6.66
			SA-9	6.66
			SA-11	6.66
			HOPLITE	6.66
			HIP-E	6.66
			HIND-D	6.66
			DUMMY	6.66
			DUMMY	6.66
			MIG-21	6.66
			DUMMY	6.66
			HVY VEH DEEP	6.66
			LT VEH DEEP	6.66
			DUMMY	6.66
		ATGM	ACRV-2 C&C	1.00
			DUMMY	6.66
			DUMMY	6.66
			DUMMY	6.66
			DUMMY	6.66
			MO82M	6.66
			M120T	6.66
			DUMMY	6.66
			DUMMY	6.66
			DUMMY	6.66
		ATGM	H122T/S	0.05
		ATGM	H152T/S	0.05
			L122T+	6.66
			DUMMY	6.66
			DUMMY	6.66

Figure D-I-8. Labeled Refire Element

TYPE SIGNATURE SOUGHT FILE

SEN# - SENSOR NUMBER; SEE SENSOR ATTENUATION ELEMENT FOR SENSOR NAMES

SIG - TYPE-SIGNATURE: 1 = LIGHT CONTRAST
2 = DELTA TEMPERATURE
3 = SILICON

RC - RESOLVABLE CYCLES REQ FOR DETECTION

AFP #	TYPE SYSTEM	CLEAR DAY ENV1-4 SEN#	SIG	RC	CLEAR NIGHT ENV5-8 SEN#	SIG	RC	DEGRADE DAY ENV9-12 SEN#	SIG	RC	DEGRADE NIGHT ENV13-16 SEN#	SIG	RC
B01	BTRP DEEP	1	1	1	1	1	1	1	1	1	1	1	1
B02	M-16	1	1	1	2	1	1	1	1	1	2	1	1
B03	M60 MG	1	1	1	4	1	1	1	1	1	4	1	1
B04	DUMMY	1	1	1	4	1	1	1	1	1	4	1	1
B05	M203GL	1	1	1	1	1	1	1	1	1	1	1	1
B06	SAW	1	1	1	3	1	1	1	1	1	3	1	1
B07	M113	15	1	1	15	1	1	15	1	1	15	1	1
B08	90RR	1	1	1	3	1	1	1	1	1	3	1	1
B09	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
B10	SNIPERM14	17	1	1	2	2	1	2	2	1	2	2	1
B11	DRAGON	10	1	1	5	2	1	10	1	1	5	2	1
B12	DUMMY	18	1	1	6	2	1	18	1	1	6	2	1
B13	VIPER	1	1	1	1	1	1	1	1	1	1	1	1
B14	LAW	1	1	1	1	1	1	1	1	1	1	1	1
B15	TOW,ITV	18	1	1	3	2	1	18	1	1	3	2	1
B16	DUMMY	0	1	1	0	2	1	0	1	1	0	2	1
B17	DUMMY	0	1	1	0	2	1	20	1	1	0	2	1
B18	TOW,1/4TON	1	1	1	0	2	1	1	1	1	6	2	1
B19	DUMMY	19	1	1	4	1	1	19	1	1	4	1	1
B20	DUMMY	15	1	1	4	2	1	15	1	1	4	2	1
B21	M551	26	1	1	8	2	1	26	1	1	8	2	1
B22	M60A3	2	1	1	9	2	1	2	1	1	9	2	1
B23	DUMMY	8	1	1	8	2	1	8	1	1	8	2	1
B24	M60A1	4	1	1	4	1	1	4	1	1	4	1	1
B25	DUMMY	0	1	1	0	1	1	0	1	1	0	1	1
B26	40MMDUSTER	2	1	1	1	2	1	2	1	1	1	2	1
B27	VULCAN,T	8	1	1	8	1	1	8	1	1	8	1	1
B28	TOW,GROUND	10	1	1	6	2	1	10	1	1	6	2	1
B29	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
B30	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
B31	STINGER	1	1	1	1	1	1	1	1	1	1	1	1
B32	CHAPPARRAL	2	1	1	2	2	1	2	2	1	2	2	1
B33	REDEYE	1	1	1	1	1	1	1	1	1	1	1	1
B34	UHIC/M	0	0	1	0	0	1	0	0	1	0	0	1
B35	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
B36	AH-1S	1	1	1	1	1	1	1	1	1	1	1	1
B37	DUMMY	1	1	1	9	2	1	1	1	1	9	2	1
B38	AH-1G	0	0	1	0	0	1	0	0	1	0	0	1
B39	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
B40	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
B41	A10	10	1	1	14	1	1	14	1	1	14	1	1
B42	LT VEH D	0	0	1	0	0	1	0	0	1	0	0	1
B43	HVY ARM D	0	0	1	0	0	1	0	0	1	0	0	1
B44	LT ARM D	0	0	1	0	0	1	0	0	1	0	0	1
B45	UH-60A/CH47	11	1	1	4	1	1	4	1	1	4	1	1
B46	DUMMY	0	1	1	0	1	1	0	1	1	0	1	1
B47	UH1H	6	1	1	4	1	1	4	1	1	4	1	1
B48	OH58A/C,OH6	7	1	1	4	1	1	4	1	1	4	1	1
B49	DUMMY	0	1	1	0	1	1	0	1	1	0	1	1
B50	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
B51	81MM MTR G	10	1	1	1	1	1	1	1	1	1	1	1
B52	DUMMY	10	1	1	1	1	1	1	1	1	1	1	1
B53	4.2IN MTR G	10	1	1	1	1	1	1	1	1	1	1	1
B54	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
B55	105MM T	0	0	1	0	0	1	0	0	1	0	0	1

Figure D-I-9. Labeled Signature Sought Element
(page 1 of 2 pages)

B56	155MM(M198)	10	1	1	1	1	1	1	1	1	1	1	1
B57	8IN SP	10	1	1	1	1	1	1	1	1	1	1	1
B58	DUMMY	0	1	1	1	1	1	1	1	1	1	1	1
B59	155MM(M114)	0	0	1	0	0	1	0	0	1	0	0	1
B60	8IN T	0	0	1	0	0	1	0	0	1	0	0	1
R01	RTRP DEEP	1	1	1	1	1	1	1	1	1	1	1	1
R02	AKS-74	1	1	1	1	1	1	1	1	1	1	1	1
R03	7.60GM	1	1	1	1	1	1	1	1	1	1	1	1
R04	RSNIPE	1	1	1	3	1	1	1	1	1	1	1	1
R05	GREN30	1	1	1	1	1	1	1	1	1	1	1	1
R06	BRDM-2	26	1	1	5	2	1	17	1	1	5	2	1
R07	BTR-60	26	1	1	5	2	1	17	1	1	5	2	1
R08	BMP-R	25	1	1	5	1	1	20	1	1	5	1	1
R09	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R10	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R11	SPG-9	20	1	1	25	1	1	25	1	1	25	1	1
R12	RPG-16	1	1	1	1	1	1	1	1	1	1	1	1
R13	AT-5	26	1	1	17	1	1	17	1	1	17	1	1
R14	AT-7	1	1	1	1	1	1	1	1	1	1	1	1
R15	RPG 7	1	1	1	4	1	1	1	1	1	1	1	1
R16	BMP2M	25	1	1	11	2	1	11	2	1	11	2	1
R17	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R18	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R19	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R20	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R21	T55	24	1	1	14	2	1	14	2	1	14	2	1
R22	T62	24	1	1	14	2	1	14	2	1	14	2	1
R23	T64	24	1	1	14	2	1	14	2	1	14	2	1
R24	T72	24	1	1	14	2	1	14	2	1	14	2	1
R25	T80	24	1	1	14	2	1	14	2	1	14	2	1
R26	ZSU-23	22	1	1	11	2	1	11	1	1	11	2	1
R27	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R28	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R29	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R30	SA-14	1	1	1	1	1	1	1	1	1	1	1	1
R31	SA-6	28	1	1	23	1	1	23	1	1	23	1	1
R32	SA-7	1	1	1	1	1	1	1	1	1	1	1	1
R33	SA-8	10	1	1	15	1	1	10	1	1	15	1	1
R34	SA-9	28	1	1	11	2	1	11	2	1	11	2	1
R35	SA-11	28	1	1	23	1	1	23	1	1	23	1	1
R36	HOPLITE	26	1	1	17	1	1	17	1	1	17	1	1
R37	HIP-E	26	1	1	17	1	1	17	1	1	17	1	1
R38	HIND-D	26	1	1	15	1	1	11	2	1	11	2	1
R39	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R40	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R41	MIG-21	50	1	1	17	1	1	17	1	1	17	1	1
R42	LT VEH D	0	0	1	0	0	1	0	0	1	0	0	1
R43	HVY ARM D	0	0	1	0	0	1	0	0	1	0	0	1
R44	LT ARM D	0	0	1	0	0	1	0	0	1	0	0	1
R45	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R46	ACRV-2 C&C	26	1	1	17	1	1	17	1	1	17	1	1
R47	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R48	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R49	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R50	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R51	M082M	50	1	1	1	1	1	1	1	1	1	1	1
R52	M120T	50	1	1	1	1	1	1	1	1	1	1	1
R53	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R54	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R55	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R56	H122T/S	50	1	1	21	1	1	21	1	1	21	1	1
R57	H152T/S	50	1	1	21	1	1	21	1	1	21	1	1
R58	L122T+	50	1	1	21	1	1	21	1	1	21	1	1
R59	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1
R60	DUMMY	0	0	1	0	0	1	0	0	1	0	0	1

Figure D-I-9. Labeled Signature Sought Element
(page 2 of 2 pages)

D-I-11. SENSING SIZE. Figure D-I-10 illustrates the element H7AFPFILES.SENS-SIZE, which shows the size of each Side 1 and Side 2 type in both open and defilade conditions. The conditions are taken from another element, H7AFPFILES.DEF-OPEN-POS, which lists the open or defilade situation for Side 1 and Side 2 for each of the 16 environments. H7AFPFILES.DEF-OPEN-POS is shown in Figure D-I-11.

D-I-12. SIGNATURE EMITTED. Figure D-I-12 illustrates the element H7AFPFILES.SIG-EMIT with the signature emitted by each of the Side 1 and Side 2 types. This labeled element is classified CONFIDENTIAL. The values in Figure D-I-12 have been changed to keep this documentation UNCLASSIFIED. This element also uses conditions from H7AFPFILE.DEF-OPEN-POS. Signatures emitted depend on open/defilade and day/night conditions. There are eight signature emitted values for each weapon system type:

light contrast during day in open
light contrast during day in defilade
light contrast during night in open
light contrast during night in defilade
heat level during day in open
heat level during day in defilade
heat level during night in open
heat level during night in defilade

D-I-13. SENSOR ATTENUATION. Figure D-I-13 illustrates H7AFPFILES.SENSOR-ATTEN. This element shows the atmospheric degradation coefficients for three different types of sensors--1=VISIBLE, 2=INFRARED, and 3=SILICON TV. Clear environments (environments 1 through 8) use the clear coefficient. The analyst can choose the degree of degradation (haze, mist, or fog) for the degraded environments 9 through 16. However, that degraded condition will then be used for all degraded environments, i.e., you cannot have haze during the day and fog during the night.

A maximum of 50 different sensors are allowed in AFP. Their assignment numbers are restricted in the following way:

Sensor numbers		**Sensor type**
1, 2, 10 12, 15, 16, 19-50	must be	optical visible band (types 1 and 2)
3, 4, 5	must be	image intensifiers (type 1)
6, 7, 8, 9, 11, 14, 17, 18	must be	thermo devices (type 3)

SENSING SIZE AS A TARGET

AFP #	TYPE SYSTEM	SENSING SIZE DEF	SENSING SIZE OPEN
B01	BTPP	.11	1.73
B02	M-16	.11	1.73
B03	7.62GM	.11	1.73
B04	MG50GM	.11	1.73
B05	M203	.11	1.73
B06	SAW	.11	1.73
B07	M113	.67	2.50
B08	90MMRR	.11	1.73
B09	DUMMY	.00	.00
B10	DUMMY	.00	.00
B11	DRAGON	.11	.87
B12	ITV-2	1.01	3.05
B13	VIPER	.11	1.30
B14	LAW	.11	1.30
B15	ITV-1	1.01	3.05
B16	IFV	.89	2.57
B17	CFV	.89	2.57
B18	1/4T TOWI	1.01	3.05
B19	M48A5	1.25	2.92
B20	M1E1	1.06	2.38
B21	M551	1.41	2.82
B22	M60A3	1.25	2.92
B23	M1	1.06	2.38
B24	M60A1	1.25	2.92
B25	M60A2	1.25	2.92
B26	DIVAD	2.70	4.50
B27	VULC1	.91	2.74
B28	1/4T TOWII	1.01	3.05
B29	DUMMY	.00	.00
B30	DUMMY	.00	.00
B31	STINGER	.11	1.73
B32	CHAP 1	2.01	3.80
B33	REDEYE	.11	1.73
B34	DUMMY	.00	.00
B35	DUMMY	.00	.00
B36	AH-1S	4.12	4.12
B37	AH-64	4.22	4.22
B38	DUMMY	.00	.00
B39	DUMMY	.00	.00
B40	DUMMY	.00	.00
B41	A10	4.41	4.41
B42	LT VEH D	.00	.00
B43	HVY ARM D	.00	.00
B44	LT ARM D	.00	.00
B45	UH-60A	3.76	3.76
B46	CEV	.01	2.92
B47	UH-1H	4.50	4.50
B48	OH-58	2.90	2.90
B49	UC3V	.01	2.96
B50	M081M	.01	1.73
B51	M081S	.01	2.5
B52	M4.2M	.01	1.73
B53	M4.2S	.01	2.5
B54	DUMMY	.00	.00
B55	DUMMY	.00	.00
B56	H155S	.01	3.29
B57	H203S	.01	3.47
B58	MLRS	.01	5.00

Figure D-I-10. Labeled Sensing Size Element
(page 1 of 2 pages)

B59	DUMMY	.00	.00
B60	DUMMY	.00	.00
R01	RTRP	.11	1.73
R02	AKS-74	.11	1.73
R03	7.60GM	.11	1.73
R04	RSNIPE	.11	1.73
R05	GREN30	.11	1.73
R06	BRDM-2	.95	2.01
R07	BTR-60	.95	2.31
R08	BMP-R	.53	2.10
R09	DUMMY	.00	.00
R10	DUMMY	.00	.00
R11	SPG-9	.55	1.73
R12	RPG-16	.11	1.73
R13	AT-5	.91	2.01
R14	AT-7	.11	1.73
R15	RPG 7	.11	1.73
R16	BMP2M	.53	2.10
R17	DUMMY	.00	.00
R18	DUMMY	.00	.00
R19	DUMMY	.00	.00
R20	DUMMY	.00	.00
R21	T55	.84	2.35
R22	T62	.84	2.40
R23	T64	.84	2.27
R24	T72	.84	2.27
R25	T80	.84	2.27
R26	ZSU-23	1.61	3.75
R27	DUMMY	.00	.00
R28	DUMMY	.00	.00
R29	DUMMY	.00	.00
R3	SA-14	0.11	1.73
R31	SA-6	1.19	3.45
R32	SA-7	.11	1.73
R33	SA-8	2.05	4.71
R34	SA-9	1.10	2.60
R35	SA-11	1.61	3.75
R36	HOPLITE	3.45	3.45
R37	HIP-E	5.00	5.00
R38	HIND-D	4.50	4.50
R39	DUMMY	.00	.00
R40	DUMMY	.00	.00
R41	MIG-25	4.50	4.50
R42	LT VEH D	.00	.00
R43	HVY ARM D	.00	.00
R44	LT ARM D	.00	.00
R45	DUMMY	.00	.00
R46	ACRV-2 C&C	.01	2.01
R47	DUMMY	.00	.00
R48	DUMMY	.00	.00
R49	DUMMY	.00	.00
R50	DUMMY	.00	.00
R51	M082M	.01	1.73
R52	M120T	.01	1.73
R53	DUMMY	.00	.00
R54	DUMMY	.00	.00
R55	DUMMY	.00	.00
R56	H122T/S	.01	2.37
R57	H152T/S	.01	2.88
R58	L122T+	.01	4.37
R59	DUMMY	.00	.00
R60	DUMMY	.00	.00

Figure D-I-10. Labeled Sensing Size Element
(page 2 of 2 pages)

ENVIRONMENTS			BLUE	RED	POSTURE
1	DAY	CLEAR	DEF	OPEN	RAPD
2	DAY	CLEAR	DEF	OPEN	STATIC
3	DAY	CLEAR	OPEN	OPEN	RADE
4	DAY	CLEAR	OPEN	DEF	BAPD
5	NIGHT	CLEAR	DEF	OPEN	RAPD
6	NIGHT	CLEAR	DEF	OPEN	STATIC
7	NIGHT	CLEAR	OPEN	OPEN	RADE
8	NIGHT	CLEAR	OPEN	DEF	BAPD
9	DAY	DEGRADE	DEF	OPEN	RAPD
10	DAY	DEGRADE	DEF	OPEN	STATIC
11	DAY	DEGRADE	OPEN	OPEN	RADE
12	DAY	DEGRADE	OPEN	DEF	BAPD
13	NIGHT	DEGRADE	DEF	OPEN	RAPD
14	NIGHT	DEGRADE	DEF	OPEN	STATIC
15	NIGHT	DEGRADE	OPEN	OPEN	RADE
16	NIGHT	DEGRADE	OPEN	DEF	BAPD

Figure D-I-11. Open And Defilade Conditions

SIGNATURE EMITTED FILE

AFP #	TYPE SYSTEM	LIGHT CONTRAST DAY OPEN	DAY DEF	NIGHT OPEN	NIGHT DEF	HEAT LEVEL DAY OPEN	DAY DEF	NIGHT OPEN	NIGHT DEF
B01	BTRP DEEP	.10	.10	.10	.10	5.00	5.00	5.00	5.00
B02	M-16	.10	.10	.10	.10	5.00	5.00	5.00	5.00
B03	M60 MG	.10	.10	.10	.10	5.00	5.00	5.00	5.00
B04	DUMMY	.10	.10	.10	.10	5.00	5.00	5.00	5.00
B05	M203GL	.10	.10	.10	.10	5.00	5.00	5.00	5.00
B06	SAW	.10	.10	.10	.10	5.00	5.00	5.00	5.00
B07	M113	.10	.20	.30	.40	1.50	0.50	1.50	0.50
B08	90RR	.10	.20	.30	.40	1.50	0.50	1.50	0.50
B09	DUMMY	.00	.00	.00	.00	0.00	0.00	0.00	0.00
B10	SNIPERM14	.10	.20	.00	.00	1.10	0.00	2.20	0.10
B11	DRAGON	.10	.10	.10	.10	6.00	6.00	6.00	6.00
B12	DUMMY	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B13	VIPER	.10	.10	.10	.10	6.00	6.00	6.00	6.00
B14	LAW	.10	.10	.10	.10	6.00	6.00	6.00	6.00
B15	TOW,ITV	.40	.20	.50	.25	0.20	0.20	0.20	0.20
B16	DUMMY	.10	.10	.10	.10	1.75	0.50	2.25	0.75
B17	DUMMY	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B18	TOW,1/4TON	.10	.10	.10	.10	0.50	0.50	2.50	2.50
B19	DUMMY	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B20	DUMMY	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B21	M551	.10	.10	.10	.10	0.50	0.75	0.50	0.75
B22	M60A3	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B23	DUMMY	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B24	M60A1	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B25	DUMMY	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B26	40MMDUSTER	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B27	VULCAN,T	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B28	TOW,GROUND	.20	.20	.50	.25	0.00	0.00	7.60	3.80
B29	DUMMY	.20	.20	.50	.25	0.00	0.00	7.60	3.80
B30	DUMMY	.20	.20	.25	.25	0.00	0.00	3.80	3.80
B31	STINGER	.10	.10	.10	.10	6.00	6.00	6.00	6.00
B32	CHAPPARRAL	.10	.10	.10	.10	3.75	0.50	4.00	0.75
B33	REDEYE	.10	.10	.10	.10	6.00	6.00	6.00	6.00
B34	UHIC/M	.10	.10	.10	.10	0.00	0.00	0.00	0.00
B35	DUMMY	.10	.10	.10	.10	0.00	0.00	0.00	0.00
B36	AH-1S	1.00	1.00	1.00	1.00	3.75	3.75	4.00	4.00
B37	DUMMY	1.00	1.00	1.00	1.00	3.75	3.75	4.00	4.00
B38	AH-1G	.40	.40	.40	.40	0.00	0.00	0.00	0.00
B39	DUMMY	.40	.40	.40	.40	0.00	0.00	0.00	0.00
B40	DUMMY	.40	.40	.40	.40	0.00	0.00	0.00	0.00
B41	A10	1.00	1.00	1.00	1.00	3.75	3.75	4.00	4.00
B42	LT VEH D	.40	.40	.40	.40	0.00	0.00	0.00	0.00
B43	HVY ARM D	.40	.40	.40	.40	0.00	0.00	0.00	0.00
B44OLT ARM D		.40	.40	.40	.40	0.00	0.00	0.00	0.00
B45	UH-60A/CH47	1.00	1.00	1.00	1.00	3.75	3.75	4.00	4.00
B46	DUMMY	.10	.10	.10	.10	0.20	0.20	0.20	0.20
B47	UH1H	1.00	1.00	1.00	1.00	3.75	3.75	4.00	4.00
B48	OH58A/C,OH6	.20	.20	.20	.20	1.75	0.00	2.25	0.00
B49	DUMMY	.40	.20	.20	.25	0.20	0.20	0.20	0.20
B50	DUMMY	.99	.20	.20	.20	0.00	0.50	0.50	0.50
B51	81MM MTR GRD	.99	.10	.10	.10	0.00	0.00	0.00	0.00
B52	DUMMY	.99	.10	.10	.10	0.00	0.00	0.00	0.00
B53	4.2IN MTR GR	.99	.10	.10	.10	0.00	0.00	0.00	0.00
B54	DUMMY	.99	.10	.10	.10	0.00	0.00	0.00	0.00
B55	105MM T	.99	.10	.10	.10	0.00	0.00	0.00	0.00
B56	155MM T(M198	.99	.40	.50	.50	15.00	15.00	15.00	15.00
B57	8IN SP	.99	.40	.50	.50	15.00	15.00	15.00	15.00
B58	DUMMY	.99	.40	.50	.50	15.00	15.00	15.00	15.00
B59	155MM T(M114	.99	.40	.40	.40	0.00	0.00	0.00	0.00
B60	8IN T	.99	.40	.40	.40	0.00	0.00	0.00	0.00
R01	RTRP DEEP	.10	.10	.10	.10	8.00	8.00	8.00	8.00
R02	AKS-74	.10	.10	.10	.10	3.00	3.00	3.00	3.00
R03	7.60GM	.10	.10	.10	.10	3.00	3.00	3.00	3.00
R04	RSNIPE	.10	.10	.10	.10	3.00	3.00	3.00	3.00
R05	GREN30	.10	.10	.10	.10	3.00	3.00	3.00	3.00

Figure D-I-12. Labeled Signature Emitted Element
(page 1 of 2 pages)

R06	BRDM-2	.40	.20	.50	.50	0.20	0.20	0.20	0.20
R07	BTR-60	.40	.20	.50	.50	0.20	0.20	0.20	0.20
R08	BMP-R	.40	.20	.50	.50	0.20	0.20	0.20	0.20
R09	DUMMY	.40	.20	.50	.50	0.00	0.50	0.00	0.00
R10	DUMMY	.40	.20	.50	.50	0.00	0.50	0.00	0.00
R11	SPG-9	.10	.10	.10	.10	6.00	6.00	6.00	6.00
R12	RPG-16	.10	.10	.10	.10	6.00	6.00	6.00	6.00
R13	AT-5	.40	.20	.50	.50	0.20	0.20	0.20	0.20
R14	AT-7	.10	.10	.10	.10	6.00	6.00	6.00	6.00
R15	RPG 7	.10	.10	.10	.10	6.00	6.00	6.00	6.00
R16	BMP2M	.40	.20	.50	.50	0.20	0.20	0.20	0.20
R17	DUMMY	.40	.20	.50	.50	0.00	0.50	0.00	0.00
R18	DUMMY	.40	.20	.50	.50	0.00	0.50	0.00	0.00
R19	DUMMY	.40	.20	.50	.50	0.00	0.50	0.00	0.00
R20	DUMMY	.40	.20	.50	.50	0.00	0.50	0.00	0.00
R21	T55	.40	.20	.50	.50	0.50	0.50	0.75	0.75
R22	T62	.40	.20	.50	.50	0.50	0.50	0.75	0.75
R23	T64	.40	.20	.50	.50	0.20	0.20	0.20	0.20
R24	T72	.40	.20	.50	.50	1.75	1.75	2.25	0.75
R25	T80	.40	.20	.50	.50	1.75	1.75	2.25	0.75
R26	ZSU-23	.40	.20	.50	.50	6.00	0.50	6.50	0.75
R27	DUMMY	.40	.40	.40	.40	0.00	0.00	0.00	0.00
R28	DUMMY	.40	.40	.40	.40	0.00	2.00	2.00	2.00
R29	DUMMY	.40	.40	.40	.40	0.00	0.00	0.00	0.00
R30	SA-14	.10	.10	.10	.10	6.00	1.00	6.50	1.00
R31	SA-6	.40	.20	.50	.50	0.50	1.00	0.75	1.00
R32	SA-7	.10	.10	.10	.10	6.00	1.00	6.50	1.00
R33	SA-8	.40	.20	.50	.40	1.75	1.00	2.25	1.00
R34	SA-9	.40	.20	.50	.50	1.75	1.00	1.75	1.00
R35	SA-11	.40	.20	.50	.50	0.50	1.00	0.75	1.00
R36	HOPLITE	.95	.95	.95	.95	3.75	3.75	4.00	4.00
R37	HIP-E	.95	.95	.95	.95	3.75	3.75	4.00	4.00
R38	HIND-D	.95	.95	.95	.95	3.75	3.75	4.00	4.00
R39	DUMMY	.95	.95	.95	.95	0.00	0.00	0.00	0.00
R40	DUMMY	.95	.95	.95	.95	0.00	0.00	0.00	0.00
R41	MIG-21	.95	.95	.95	.95	3.75	3.75	4.00	4.00
R42	LT VEH D	.95	.95	.95	.95	0.00	0.00	0.00	0.00
R43	HVY ARM D	.95	.95	.95	.95	0.00	0.00	0.00	0.00
R44	LT ARM D	.95	.95	.95	.95	0.00	0.00	0.00	0.00
R45	DUMMY	.95	.95	.95	.95	0.00	0.00	0.00	0.00
R46	ACRV-2 C&C	.40	.20	.50	.50	0.20	0.20	0.20	0.20
R47	DUMMY	.99	.40	.50	.50	0.00	0.00	0.00	0.00
R48	DUMMY	.99	.40	.50	.50	0.00	0.00	0.00	0.00
R49	DUMMY	.99	.40	.50	.50	0.00	0.00	0.00	0.00
R50	DUMMY	.99	.40	.50	.50	0.00	0.00	0.00	0.00
R51	M082M	.99	.10	.10	.10	0.00	0.00	0.00	0.00
R52	M120T	.99	.10	.10	.10	0.00	0.00	0.00	0.00
R53	DUMMY	.99	.10	.10	.10	0.00	0.00	0.00	0.00
R54	DUMMY	.99	.10	.10	.10	0.00	0.00	0.00	0.00
R55	DUMMY	.99	.10	.10	.10	0.00	0.00	0.00	0.00
R56	H122T/S	.99	.40	.40	.40	15.00	15.00	15.00	15.00
R57	H152T/S	.99	.40	.40	.40	15.00	15.00	15.00	15.00
R58	L122T+	.99	.40	.40	.40	15.00	15.00	15.00	15.00
R59	DUMMY	.99	.40	.40	.40	0.00	0.00	0.00	0.00
R60	DUMMY	.99	.40	.40	.40	0.00	0.00	0.00	0.00

Figure D-I-12. Labeled Signature Emitted Element
(page 2 of 2 pages)

SENSOR ATTENUATION FILE

ATMOSPHERIC EXTINCTION COEFFICIENTS

(COLUMNS ARE SENSOR TYPES,
ROWS ARE ATMOSPHERIC CONDITIONS)

	VISIBLE(1)	IR(2)	SILICON(3)
CLEAR	.28	.29	.28
HAZE	.56	.29	.44
MIST	1.96	.65	1.50
FOG	7.82	5.39	5.98

SENSOR #	SENSOR NAME	SENSOR TYPE	
1	UNAIDED EYE	1	VISIBLE
2	BINOCULARS(7X)	1	VISIBLE
3	STARLIGHT	1	VISIBLE
4	CREW WPN SIGHT	1	VISIBLE
5	BMP NIGHTSIGHT	1	VISIBLE
6	TOW NIGHTSIGHT	2	IR
7	DRAGON NIGHTSIGHT	2	IR
8	TANK THERMAL SIGHT	2	IR
9	FLIR(US)	2	IR
10	SI TV	3	SILICON
11	FLIR(SOVIET)	2	IR
12	VIDICON	3	SILICON
13	RADAR	1	VISIBLE
14	TANK THERMAL(SOV)	2	IR
15	LLLTV	1	VISIBLE
16	BINOCULARS(3X)	1	VISIBLE
17	THERMAL(NOT USED)	2	IR
18	THERMAL(NOT USED)	2	IR
19	BINOCULARS(N/USED)	1	VISIBLE
20	BINOCULARS(4X)	1	VISIBLE
21	BINOCULARS(13X)	1	VISIBLE
22	BINOCULARS(6X)	1	VISIBLE
23	BINOCULARS(1.9)	1	VISIBLE
24	BINOCULARS(3.5)	1	VISIBLE
25	BINOCULARS(6.6)	1	VISIBLE
26	BINOCULARS(8.0)	1	VISIBLE
27	BINOCULARS(2.0)	1	VISIBLE
28	BINOCULARS(10.0)	1	VISIBLE
29	OPTICAL(NOT USED)	1	VISIBLE
30	OPTICAL(NOT USED)	1	VISIBLE
31	OPTICAL(NOT USED)	1	VISIBLE
32	OPTICAL(NOT USED)	1	VISIBLE
33	OPTICAL(NOT USED)	1	VISIBLE
34	BINOC(9.0X)SNIPER	1	VISIBLE
35	OPTICAL(NOT USED)	1	VISIBLE
36	OPTICAL(NOT USED)	1	VISIBLE
37	OPTICAL(NOT USED)	1	VISIBLE
38	OPTICAL(NOT USED)	1	VISIBLE
39	OPTICAL(NOT USED)	1	VISIBLE
40	OPTICAL(NOT USED)	1	VISIBLE
41	OPTICAL(NOT USED)	1	VISIBLE
42	OPTICAL(NOT USED)	1	VISIBLE
43	OPTICAL(NOT USED)	1	VISIBLE
44	OPTICAL(NOT USED)	1	VISIBLE
45	OPTICAL(NOT USED)	1	VISIBLE
46	OPTICAL(NOT USED)	1	VISIBLE
47	OPTICAL(NOT USED)	1	VISIBLE
48	OPTICAL(NOT USED)	1	VISIBLE
49	OPTICAL(NOT USED)	1	VISIBLE
50	OPTICAL(NOT USED)	1	VISIBLE

Figure D-I-13. Labeled Sensor Attenuation Element

D-I-14. SENSOR MAGNIFICATION. Figure D-I-14 illustrates H7AFPFILES.SENSOR-MAG, with sensor magnification factors. Each sensor has a wide magnification factor. Narrow magnification factors are not used by the Combat Module. The sensor type is the same as the types used in the sensor attenuation element, 1=visible, 2=IR, and 3=silicon.

SENSOR MAGNIFICATION FILE

SENSOR #	SENSOR NAME	SENSOR TYPE	WIDE MAG	NARROW MAG
1	UNAIDED EYE	1	1.0	
2	BINOCULARS(7X)	1	7.0	
3	STARLIGHT (PVS-4)	1	3.8	
4	CREW WPN SIGHT TVS5	1	6.0	
5	BMP NIGHTSIGHT	1	0.0	
6	TOW NIGHTSIGHT	2	4.0	
7	DRAGON NIGHTSIGHT	2	4.0	
8	TANK THERMAL SIGHT	2	3.0	
9	FLIR(US)	2	0.0	
10	SI TV	3	0.0	
11	FLIR(SOVIET)	2	0.0	
12	VIDICON	3	0.0	
13	RADAR	1	0.0	
14	TANK THERMAL(SOV)	2	0.0	
15	LLLTV	3	0.0	
16	BINOCULARS(3X)	1	2.6	
17	THERMAL(M1 NT)	2	3.0	
18	THERMAL(CHAP NT)	2	1.0	
19	BINOCULARS(NOT USED)	1	0.0	
20	BINOCULARS(4X)IFV D	1	4.0	
21	BINOCULARS(13X)ITV D	1	13.0	
22	BINOCULARS(6X)DRAG	1	6.0	
23	BINOCULARS(1.9)1S	1	1.9	
24	BINOCULARS(3.5)T72	1	3.5	
25	BINOCULARS(6.6)BMP	1	6.6	
26	BINOC (8X)BRDM2	1	8.0	
27	BINOCULARS(2.0)	1	2.0	
28	BINOCULARS(10.0)	1	10.0	
29	BINOC(2.6X)M60 DAY	1	2.6	
30	BINOC(4X)DRAGON DAY	1	4.0	
31	M60 NIGHT 7X	1	7.0	
32	NSD-3	1	0.0	
33	PON	1	0.0	
34	OPTICAL(NOT USED)	1	0.0	
35	IR TOW 4X	2	4.0	
36	AAH DAY 4X	1	4.0	
37	OPTICAL(NOT USED)	1	0.0	
38	OPTICAL(NOT USED)	1	0.0	
39	OPTICAL(NOT USED)	1	0.0	
40	OPTICAL(NOT USED)	1	0.0	
41	OPTICAL(NOT USED)	1	0.0	
42	OPTICAL(NOT USED)	1	0.0	
43	OPTICAL(NOT USED)	1	0.0	
44	OPTICAL(NOT USED)	1	0.0	
45	OPTICAL(NOT USED)	1	0.0	
46	OPTICAL(NOT USED)	1	0.0	
47	OPTICAL(NOT USED)	1	0.0	
48	OPTICAL(NOT USED)	1	0.0	
49	OPTICAL(NOT USED)	1	0.0	
50	OPTICAL(NOT USED)	1	0.0	

Figure D-I-14. Labeled Sensor Magnification Element

D-I-15. PARTICIPATION. Figure D-I-15 shows a sample section of H7AFPFILES.PARTICIPATE. The indirect fire participation for each indirect fire weapon by posture is listed. Currently, all values are set to a default of 1.0. These values, multiplied together with the initial daily allocation of indirect shooters, compute the number of indirect shooters, and could reduce the participation rate if less than 1.0.

ARTY PARTICIPATION FILE

SHOOTER AFP #	TARGET AFP #	TARGET DESC	RAPD (ENV 1, 5,9,13) PARTICIP	STATIC (ENV 2, 6,10,14) PARTICIP	RABD (ENV 3, 7,11,15) PARTICIP	BAPD (ENV 4, 8,12,16) PARTICIP
B51	R01	RTRP	1.0	1.0	1.0	1.0
	R02	AKS-74	1.0	1.0	1.0	1.0
	R03	7.60GM	1.0	1.0	1.0	1.0
	R04	RSNIPE	1.0	1.0	1.0	1.0
	R05	GREN30	1.0	1.0	1.0	1.0
	R06	BRDM-2	1.0	1.0	1.0	1.0
	R07	BTR-60	1.0	1.0	1.0	1.0
	R08	BMP-R	1.0	1.0	1.0	1.0
	R09	DUMMY	1.0	1.0	1.0	1.0
	R10	DUMMY	1.0	1.0	1.0	1.0
	R11	SPG-9	1.0	1.0	1.0	1.0
	R12	RPG-16	1.0	1.0	1.0	1.0
	R13	AT-5	1.0	1.0	1.0	1.0
	R14	AT-7	1.0	1.0	1.0	1.0
	R15	RPG 7	1.0	1.0	1.0	1.0
	R16	BMP2M	1.0	1.0	1.0	1.0
	R17	DUMMY	1.0	1.0	1.0	1.0
	R18	DUMMY	1.0	1.0	1.0	1.0
	R19	DUMMY	1.0	1.0	1.0	1.0
	R20	DUMMY	1.0	1.0	1.0	1.0
	R21	T55	1.0	1.0	1.0	1.0
	R22	T62	1.0	1.0	1.0	1.0
	R23	T64	1.0	1.0	1.0	1.0
	R24	T72	1.0	1.0	1.0	1.0
	R25	T80	1.0	1.0	1.0	1.0
	R26	ZSU-23	1.0	1.0	1.0	1.0
	R27	DUMMY	1.0	1.0	1.0	1.0
	R28	DUMMY	1.0	1.0	1.0	1.0
	R29	DUMMY	1.0	1.0	1.0	1.0
	R30	SA-14	1.0	1.0	1.0	1.0
	R31	SA-6	1.0	1.0	1.0	1.0
	R32	SA-7	1.0	1.0	1.0	1.0
	R33	SA-8	1.0	1.0	1.0	1.0
	R34	SA-9	1.0	1.0	1.0	1.0
	R35	SA-11	1.0	1.0	1.0	1.0
	R36	HOPLITE	1.0	1.0	1.0	1.0
	R37	HIP-E	1.0	1.0	1.0	1.0
	R38	HIND-D	1.0	1.0	1.0	1.0
	R39	DUMMY	1.0	1.0	1.0	1.0
	R40	DUMMY	1.0	1.0	1.0	1.0
	R41	MIG-21	1.0	1.0	1.0	1.0
	R42	DUMMY	1.0	1.0	1.0	1.0
	R43	DUMMY	1.0	1.0	1.0	1.0
	R44	DUMMY	1.0	1.0	1.0	1.0
	R45	DUMMY	1.0	1.0	1.0	1.0
	R46	ACRV-2 C&C	1.0	1.0	1.0	1.0
	R47	DUMMY	1.0	1.0	1.0	1.0
	R48	DUMMY	1.0	1.0	1.0	1.0
	R49	DUMMY	1.0	1.0	1.0	1.0
	R50	DUMMY	1.0	1.0	1.0	1.0
	R51	M082M	1.0	1.0	1.0	1.0
	R52	M120T	1.0	1.0	1.0	1.0
	R53	DUMMY	1.0	1.0	1.0	1.0
	R54	DUMMY	1.0	1.0	1.0	1.0
	R55	DUMMY	1.0	1.0	1.0	1.0
	R56	H122T/S	1.0	1.0	1.0	1.0
	R57	H152T/S	1.0	1.0	1.0	1.0
	R58	L122T+	1.0	1.0	1.0	1.0
	R59	DUMMY	1.0	1.0	1.0	1.0
	R60	DUMMY	1.0	1.0	1.0	1.0
B52	R01	RTRP	1.0	1.0	1.0	1.0
	R02	AKS-74	1.0	1.0	1.0	1.0
	R03	7.60GM	1.0	1.0	1.0	1.0
	R04	RSNIPE	1.0	1.0	1.0	1.0
	R05	GREN30	1.0	1.0	1.0	1.0
	R06	BRDM-2	1.0	1.0	1.0	1.0
	R07	BTR-60	1.0	1.0	1.0	1.0
	R08	BMP-R	1.0	1.0	1.0	1.0
	R09	DUMMY	1.0	1.0	1.0	1.0
	R10	DUMMY	1.0	1.0	1.0	1.0
	THROUGH					
R60	B60	DUMMY	1.0	1.0	1.0	1.0

Figure D-I-15. Labeled Participation Element

Section III. OUTPUT

D-I-16. COMBAT MODULE FORMAT BASEDATA. The H7BASEDATA elements which are input to the AFP Combat Module are organized as described in paragraph D-I-2. Section II of this annex described each logical section. Therefore, this section will only show Combat Module format. Combat Module format is a logical series of arrays, all unlabeled. There are 16 H7BASEDATA elements for every AFP run; one for each of the 16 environments. Each element has 3,167 lines, although the data in the elements varies. A sample H7BASEDATA element is shown in Figure D-I-16.

Section IV. RUNSTREAMS

D-I-17. BASEDATA RUNSTREAM GENERATION. Sixteeen H7BASEDATA elements are needed for a complete AFP run, four postures x two day/night conditions x two clear/degraded conditions. The 16 elements are created in a runstream which executes 11 different programs and uses the system EDITOR. The run-stream creates the 16 elements by creating logical sections of BASEDATA and piecing those sections together in a prescribed sequence. Figure D-I-1 illustrates the runstream sequence. Note that ADDITIONALS is a collection of eight logical sections of unlabeled lines. Each section is added to its appropriate place in the BASEDATA elements using the EDITOR. The artil-lery's expected fractional damage, which is in H7ARTYEFD, is also unlabeled and is also added to the BASEDATA elements using the EDITOR. All other sections are labeled and must go through conversion programs. If the values in the section do not vary by posture, day/night, or clear/degraded conditions, the labels are merely stripped; and the resulting array is added to all 16 BASEDATA elements. If the values do vary, the program will create 16 different arrays and add the appropriate array to each of the 16 BASEDATA elements. Table D-I-2 shows the variations for each section.

D-I-18. BASEDATA RUNSTREAM ADJUSTMENT FOR INVENTORIES. Inventories may vary by posture. For example, 490 light vehicles were played in the deep in RAPD posture, but that number was changed to 440 vehicles in BAPD. In a Blue defense posture, the infantry would have dismounted, and the transport vehicles would have moved to the rear, i.e., deep. In a Blue attack pos-ture, some transport vehicles would be moving infantry to the front. This means that there may be four inventory elements. The sample runstream which follows in Figure D-I-17 is set up to handle this. It executes the inventory format conversion program four times, each time using a different input inventory element, H7AFPFILES.INVHM80/RAPD, H7AFPFILES.INVHM80/STATIC, H7AFPFILES.INVHM80/RADE, and H7AFPFILES.INVHM80/BAPD. If there is only one inventory element, i.e., they are not posture-dependent, the inventory conversion program only needs to be executed one time. Change the inventory element name on line 56 by removing the version name. Change H7AFPFILE. INVHM80/RAPD to H7AFPFILE.INVHM80. Remove lines 91-94, lines 103-106, and lines 115-118.

```
LINE #      DATA DESCRIPTION
------      MISC DATA FROM ADDITIONALS
     1         2   60   60   1      4000    10    10     5
            DETECTION
     2       20 20  5  5  5  1  1  1  1  1
     3       1  1  1  1  1  1  1  1  1  1
     4       1  1  1  1  1  1  1  1  1  1
     5       1  1  1  1  1  1  1  1  1  1
     6       1  1  1  1  1  1  1  1  1  1
     7       1  1  1  1  1  1  1  1  1  1
     8       20 20  5  5  5  1  1  1  1  1
     9       1  1  1  1  1  1  1  1  1  1
    10       1  1  1  1  1  1  1  1  1  1
    11       1  1  1  1  1  1  1  1  1  1
    12       1  1  1  1  1  1  1  1  1  1
    13       1  1  1  1  1  1  1  1  1  1
            INVENTORY
    14        389         0
    15        107         0
    16         39         0
    17          0         0
    18         75         0
    19          0         0
    20        460         0
    21          0         0
    22          0         0
    23          0         0
    24        278         0
    25          0         0
    26          0         0
    27        135         0
    28        146         0
    29          0         0
    30          0         0
    31          0         0
    32          0         0
    33          0         0
    34          0         0
    35          0         0
    36          0         0
    37        291         0
    38          0         0
    39          0         0
    40         24         0
    41          0         0
    42          0         0
    43          0         0
    44          0         0
    45         24         0
    46         83         0
    47          0         0
    48          0         0
    49         42         0
    50          0         0
    51          0         0
    52          0         0
    53          0         0
    54         24         0
    55         44         0
    56         22         0
    57         18         0
    58          0         0
    59          8         0
    60         25         0
    61         52         0
    62          0         0
    63          0         0
    64         54         0
    65          0         0
    66         53         0
    67          0         0
    68          0         0
    69         54         0
    70         12         0
    71          0         0
    72          0         0
    73          0         0
    74       1527         0
    75        480         0
```

Figure D-I-16. Combat Module Format BASEDATA
(page 1 of 5 pages)

76	153	0
77	189	0
78	63	0
79	93	0
80	411	0
81	18	0
82	0	0
83	0	0
84	0	0
85	420	0
86	27	0
87	84	0
88	480	0
89	426	0
90	0	0
91	0	0
92	0	0
93	0	0
94	243	0
95	39	0
96	126	0
97	270	0
98	78	0
99	48	0
100	0	0
101	0	0
102	0	0
103	117	0
104	21	0
105	192	0
106	24	0
107	24	0
108	0	0
109	6	0
110	6	0
111	36	0
112	0	0
113	0	0
114	24	0
115	264	0
116	54	0
117	12	0
118	0	0
119	0	0
120	0	0
121	0	0
122	0	0
123	0	0
124	12	0
125	171	0
126	0	0
127	0	0
128	0	0
129	300	0
130	60	0
131	54	0
132	0	0
133	0	0
	EXTERNAL LOSSES	
134	0.0	

LINES 135 - 252 OMITTED TO SHORTEN LENGTH OF FIGURE

253	0.0	
	NEAR	FAR
254	6	6
255	1	2
256	1	3
257	1	4
258	1	2
259	1	2
260	1	4
261	1	2
262	1	5
263	1	5
264	1	3
265	1	5
266	1	2
267	1	2
268	1	5
269	1	5
270	1	5

Figure D-I-16. Combat Module Format BASEDATA
(page 2 of 5 pages)

271	1	5
272	1	5
273	1	5
274	1	5
275	1	5
276	1	5
277	1	5
278	1	5
279	1	5
280	1	5
281	1	5
282	1	5
283	1	5
284	1	5
285	1	5
286	1	5
287	1	5
288	1	5
289	1	5
290	1	5
291	1	5
292	1	5
293	1	5
294	1	5
295	6	6
296	6	6
297	6	6
298	1	5
299	1	5
300	1	5
301	1	5
302	1	5
303	1	6
304	1	5
305	1	5
306	1	5
307	1	6
308	1	6
309	1	6
310	1	6
311	1	6
312	1	6
313	1	6
314	6	6
315	1	2
316	1	3
317	1	3
318	1	2
319	1	4
320	1	4
321	1	4
322	1	5
323	1	5
324	1	3
325	1	2
326	1	5
327	1	3
328	1	1
329	1	5
330	1	5
331	1	5
332	1	5
333	1	5
334	1	5
335	1	5
336	1	5
337	1	5
338	1	5
339	1	5
340	1	5
341	1	5
342	1	5
343	1	5
344	1	5
345	1	5
346	1	5
347	1	5
348	1	5
349	1	5
350	1	5
351	1	5
352	1	5

Figure D-I-16. Combat Module Format BASEDATA
(page 3 of 5 pages)

```
 353          1   5
 354          1   5
 355          6   6
 356          6   6
 357          6   6
 358          1   5
 359          1   5
 360          1   5
 361          1   5
 362          1   5
 363          1   5
 364          1   5
 365          1   5
 366          1   5
 367          1   6
 368          1   6
 369          1   6
 370          1   6
 371          1   6
 372          1   6
 373          1   6
           ENVIRONMENT SITES
 374          1.00
           INDIRECT SHOOTER RANGES
 375        .70  .20  .10  .0   .0   .0
 376        .60  .20  .10  .10  .0   .0
 377        .60  .20  .10  .10  .0   .0
 378        .17  .17  .16  .16  .17  .17
 379        .17  .17  .16  .16  .17  .17
 380        .25  .25  .15  .20  .10  .05
 381        .15  .15  .00  .00  .00  .70
 382        .10  .0   .0   .0   .0   .90
 383        .17  .17  .16  .16  .17  .17
 384        .17  .17  .16  .16  .17  .17
 385        .70  .20  .10  .0   .0   .0
 386        .60  .20  .10  .10  .0   .0
 387        .17  .17  .16  .16  .17  .17
 388        .17  .17  .16  .16  .17  .17
 389        .17  .17  .16  .16  .17  .17
 390        .25  .25  .15  .20  .10  .05
 391        .13  .13  .12  .16  .06  .40
 392        .20  .25  .30  .10  .10  .05
 393        .17  .17  .16  .16  .17  .17
 394        .17  .17  .16  .16  .17  .17
           REFIRE TIMES
 395          0.33   0.33   0.33   0.33   0.33   0.33   0.33   0.33   0.33   0.

LINES 396 - 1113 OMITTED

 1114         6.66   6.66   6.66   6.66   6.66   6.66   6.66   6.66   6.66   6.
           SIGNATURE SOUGHT
 1115         1  . 1      1

LINES 1116 - 1233 OMITTED

 1234         0    0      1
           SENSING SIZE
 1235       .11

LINES 1236 - 1353 OMITTED

 1354       .00
           LIGHT LEVEL
 1355         600.00
           SIGNATURE EMITTED
 1356       .13     3.0

LINES 1357 - 1474 OMITTED

 1475       .99     .0
           ATTENUATION
 1476       .28   .28   .28   .28   .28   .29   .29   .29   .29   .28   .29   .28   .28   .2
 1477       .28   .28   .29   .29   .28   .28   .28   .28   .28   .28   .28   .28   .28   .2
 1478       .28   .28   .28   .28   .28   .28   .28   .28   .28   .28   .28   .28   .28   .2
 1479       .28   .28   .28   .28   .28   .28   .28   .28
           MAGNIFICATION
 1480       1.0   7.0   3.8   6.0   0.0   4.0   4.0   3.0   0.0   0.0   0.0   0.0   0.0   0.
 1481       0.0   2.6   3.0   1.0   0.0   4.0  13.0   6.0   1.9   3.5   6.6   8.0   2.0  10.
 1482       2.6   4.0   7.0   0.0   0.0   0.0   4.0   4.0   0.0   0.0   0.0   0.0   0.0   0.
 1483       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
           BRIGHTNESS
 1484         3.00
```

Figure D-I-16. Combat Module Format BASEDATA
(page 4 of 5 pages)

```
             ARTY PARTICIPATION
  1485        1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
LINES 1486 - 1603 OMITTED
  1604        1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
             ARTY EFD
  1605          .0000000  .0000000  .0000000  .0000000  .0000000  .0006340
LINES 1606 - 2803 OMITTED
  2804          .0000000  .0000000  .0000000  .0000000  .0000000  .0000000
             PERCENT INDIRECT FIRE
  2805         1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  2806         1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
             AMMO TYPE
  2807          1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
LINES 2808 - 3165 OMITTED
  3166          1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
             PRINT SWITCHES
  3167        .FALSE. .FALSE.
```

Figure D-I-16. Combat Module Format BASEDATA
(page 5 of 5 pages)

BASEDATA section	Varies by ("-" denotes no variance)	How variation is implemented
Miscellaneous data in ADDITIONALS	-	-
Detection based on round fired	-	-
Inventory	force ratio and posture	ratio in H7AFPFILES.INV-FACTORS used as multiplier in program, also see paragraph D-I-17
External loss	-	-
Near far	-	-
Number of sites	-	-
Indirect Range Distribution	-	-
Refire	-	-
Signature sought	day/night and clear/degraded	signature varies by condition in labeled signature sought element
Sensing size	open/defilade	H7AFPFILES.OPEN-DEF-POS used to determine OPEN/DEF condition
Light level	day/night	ADDITIONALS line 15 (day) is added to environments 1-4 and 9-12; line 16 (night) is added to environments 5-8 and 13-16
Signature emitted	open/defilade and day/night	H7AFPFILES.OPEN-DEF-POS used to determine OPEN/DEF conditions; environments 1-4, 9-12 hardwired to be day, environments 5-8, 13-16 hardwired to be night
Attenuation	degraded atmospheric condition	degraded atmospheric coefficient (haze, fog, or mist) accepted by conversion program H7AFP.M-ATTEN
Magnification	-	-
Brightness	clear/degraded conditions	ADDITIONALS line 17 (clear) is added to environments 1-8; line 18 (degraded) is added to environments 9-16
Indirect fire participation	posture (although not currently used)	Participation varies by condition in labeled indirect fire participation element
Indirect fire EFD	force ratio and posture and day/night	analyst adjusts artillery elements which are used to build EFDs (see Annex D-VIII)
Percentage indirect fire	analyst discretion	analyst changes lines 19 and 20 in ADDITIONALS with @ED
Ammunition type	-	-
Print switches	analyst discretion	analyst changes line 381 in ADDITIONALS with @ED

Table D-I-3. Variable Conditions for H7BASEDATA Elements in BASEDATA Order

```
@QUAL UNCLASSIFIED
@ASG,A H7AFPFILES/XXX/XXX.
@ASG,T 30-BASEDATA.,///3000
@ED,I 30-BASEDATA.1

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.2

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.3

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.4

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.5

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.6

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.7

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.8

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.9

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.10

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.11

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.12

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.13

ADD H7AFFFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.14

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.15

ADD H7AFPFILES.ADDITIONALS 1 13
@ED,I 30-BASEDATA.16

ADD H7AFPFILES.ADDITIONALS 1 13
EXIT
@ASG,T AFP1.
@ED,QI AFP1.

ADD H7AFPFILES.INVHM80/RAPD
@ASG,T AFP2.
@ASG,T AFP3.
@ED,QI AFP3.

ADD H7AFPFILES.INV-FACTORS
@ASG,T AFP4.
@ED,QI AFP4.

ADD H7AFPFILES.DEF-OPEN-POS
@ASG,T OAFP1.
@ASG,T OAFP2.
@ASG,T OAFP3.
@ASG,T OAFP4.
@ASG,T OAFP5.
@ASG,T OAFP6.
@ASG,T OAFP7.
@ASG,T OAFP8.
@ASG,T OAFP9.
@ASG,T OAFP10.
@ASG,T OAFP11.
@ASG,T OAFP12.
@ASG,T OAFP13.
@ASG,T OAFP14.
@ASG,T OAFP15.
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream
(page 1 of 9 pages)

```
@ASG,T OAFP16.
@XQT *H7AFP.870M-INV
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.5
ADD OAFP5.
@ED,U 30-BASEDATA.9
ADD OAFP9.
@ED,U 30-BASEDATA.13
ADD OAFP13.
@ED,QI AFP1.

ADD H7AFPFILES.INVHM80/STATIC
@XQT *H7AFP.870M-INV
@ED,U 30-BASEDATA.2
ADD OAFP2.
@ED,U 30-BASEDATA.6
ADD OAFP6.
@ED,U 30-BASEDATA.10
ADD OAFP10.
@ED,U 30-BASEDATA.14
ADD OAFP14.
@ED,QI AFP1.

ADD H7AFPFILES.INVHM80/RADE
@XQT *H7AFP.870M-INV
@ED,U 30-BASEDATA.3
ADD OAFP3.
@ED,U 30-BASEDATA.7
ADD OAFP7.
@ED,U 30-BASEDATA.11
ADD OAFP11.
@ED,U 30-BASEDATA.15
ADD OAFP15.
@ED,QI AFP1.

ADD H7AFPFILES.INVHM80/BAPD
@XQT *H7AFP.870M-INV
@ED,U 30-BASEDATA.4
ADD OAFP4.
@ED,U 30-BASEDATA.8
ADD OAFP8.
@ED,U 30-BASEDATA.12
ADD OAFP12.
@ED,U 30-BASEDATA.16
ADD OAFP16.
EXIT
@ED,QI AFP1.

ADD H7AFPFILES.EXT-LOSSES
@XQT *H7AFP.870M-EXTLOSS
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP1.
@ED,U 30-BASEDATA.3
ADD OAFP1.
@ED,U 30-BASEDATA.4
ADD OAFP1.
@ED,U 30-BASEDATA.5
ADD OAFP1.
@ED,U 30-BASEDATA.6
ADD OAFP1.
@ED,U 30-BASEDATA.7
ADD OAFP1.
@ED,U 30-BASEDATA.8
ADD OAFP1.
@ED,U 30-BASEDATA.9
ADD OAFP1.
@ED,U 30-BASEDATA.10
ADD OAFP1.
@ED,U 30-BASEDATA.11
ADD OAFP1.
@ED,U 30-BASEDATA.12
ADD OAFP1.
@ED,U 30-BASEDATA.13
ADD OAFP1.
@ED,U 30-BASEDATA.14
ADD OAFP1.
@ED,U 30-BASEDATA.15
ADD OAFP1.
@ED,U 30-BASEDATA.16
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream (page 2 of 9 pages)

```
ADD OAFP1.
EXIT
@ED,QI AFP1.
ADD H7AFPFILES.NEAR-FAR
@XQT *H7AFP.870M-NEARFAR
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP1.
@ED,U 30-BASEDATA.3
ADD OAFP1.
@ED,U 30-BASEDATA.4
ADD OAFP1.
@ED,U 30-BASEDATA.5
ADD OAFP1.
@ED,U 30-BASEDATA.6
ADD OAFP1.
@ED,U 30-BASEDATA.7
ADD OAFP1.
@ED,U 30-BASEDATA.8
ADD OAFP1.
@ED,U 30-BASEDATA.9
ADD OAFP1.
@ED,U 30-BASEDATA.10
ADD OAFP1.
@ED,U 30-BASEDATA.11
ADD OAFP1.
@ED,U 30-BASEDATA.12
ADD OAFP1.
@ED,U 30-BASEDATA.13
ADD OAFP1.
@ED,U 30-BASEDATA.14
ADD OAFP1.
@ED,U 30-BASEDATA.15
ADD OAFP1.
@ED,U 30-BASEDATA.16
ADD OAFP1.
EXIT
@ED,U 30-BASEDATA.1
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.2
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.3
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.4
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.5
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.6
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.7
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.8
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.9
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.10
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.11
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.12
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.13
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.14
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.15
ADD H7AFPFILES.ADDITIONALS 14 14
@ED,U 30-BASEDATA.16
ADD H7AFPFILES.ADDITIONALS 14 14
EXIT
@ED,QI AFP1.
ADD H7AFPFILES.ARTY-RANGE
@XQT *H7AFP.870M-ARTYRNG
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP2.
@ED,U 30-BASEDATA.3
ADD OAFP3.
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream
(page 3 of 9 pages)

```
@ED,U 30-BASEDATA.4
ADD OAFP4.
@ED,U 30-BASEDATA.5
ADD OAFP5.
@ED,U 30-BASEDATA.6
ADD OAFP6.
@ED,U 30-BASEDATA.7
ADD OAFP7.
@ED,U 30-BASEDATA.8
ADD OAFP8.
@ED,U 30-BASEDATA.9
ADD OAFP9.
@ED,U 30-BASEDATA.10
ADD OAFP10.
@ED,U 30-BASEDATA.11
ADD OAFP11.
@ED,U 30-BASEDATA.12
ADD OAFP12.
@ED,U 30-BASEDATA.13
ADD OAFP13.
@ED,U 30-BASEDATA.14
ADD OAFP14.
@ED,U 30-BASEDATA.15
ADD OAFP15.
@ED,U 30-BASEDATA.16
ADD OAFP16.
EXIT
@ED,QI AFP1.

ADD H7AFPFILES.REFIRE
@XQT *H7AFP.870M-REFIRE
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP1.
@ED,U 30-BASEDATA.3
ADD OAFP1.
@ED,U 30-BASEDATA.4
ADD OAFP1.
@ED,U 30-BASEDATA.5
ADD OAFP1.
@ED,U 30-BASEDATA.6
ADD OAFP1.
@ED,U 30-BASEDATA.7
ADD OAFP1.
@ED,U 30-BASEDATA.8
ADD OAFP1.
@ED,U 30-BASEDATA.9
ADD OAFP1.
@ED,U 30-BASEDATA.10
ADD OAFP1.
@ED,U 30-BASEDATA.11
ADD OAFP1.
@ED,U 30-BASEDATA.12
ADD OAFP1.
@ED,U 30-BASEDATA.13
ADD OAFP1.
@ED,U 30-BASEDATA.14
ADD OAFP1.
@ED,U 30-BASEDATA.15
ADD OAFP1.
@ED,U 30-BASEDATA.16
ADD OAFP1.
EXIT
@ED,QI AFP1.

ADD H7AFPFILES.SIGNATURE
@XQT *H7AFP.870M-SIGS
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP2.
@ED,U 30-BASEDATA.3
ADD OAFP3.
@ED,U 30-BASEDATA.4
ADD OAFP4.
@ED,U 30-BASEDATA.5
ADD OAFP5.
@ED,U 30-BASEDATA.6
ADD OAFP6.
@ED,U 30-BASEDATA.7
ADD OAFP7.
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream
(page 4 of 9 pages)

```
@ED,U 30-BASEDATA.8
ADD OAFP8.
@ED,U 30-BASEDATA.9
ADD OAFP9.
@ED,U 30-BASEDATA.10
ADD OAFP10.
@ED,U 30-BASEDATA.11
ADD OAFP11.
@ED,U 30-BASEDATA.12
ADD OAFP12.
@ED,U 30-BASEDATA.13
ADD OAFP13.
@ED,U 30-BASEDATA.14
ADD OAFP14.
@ED,U 30-BASEDATA.15
ADD OAFP15.
@ED,U 30-BASEDATA.16
ADD OAFP16.
EXIT
@ED,QI AFP1.

ADD H7AFPFILES.SENSING-SIZE
@XQT *H7AFP.870M-SENSIZE
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP2.
@ED,U 30-BASEDATA.3
ADD OAFP3.
@ED,U 30-BASEDATA.4
ADD OAFP4.
@ED,U 30-BASEDATA.5
ADD OAFP5.
@ED,U 30-BASEDATA.6
ADD OAFP6.
@ED,U 30-BASEDATA.7
ADD OAFP7.
@ED,U 30-BASEDATA.8
ADD OAFP8.
@ED,U 30-BASEDATA.9
ADD OAFP9.
@ED,U 30-BASEDATA.10
ADD OAFP10.
@ED,U 30-BASEDATA.11
ADD OAFP11.
@ED,U 30-BASEDATA.12
ADD OAFP12.
@ED,U 30-BASEDATA.13
ADD OAFP13.
@ED,U 30-BASEDATA.14
ADD OAFP14.
@ED,U 30-BASEDATA.15
ADD OAFP15.
@ED,U 30-BASEDATA.16
ADD OAFP16.
EXIT
@ED,U 30-BASEDATA.1
ADD H7AFPFILES.ADDITIONALS 15 15
@ED,U 30-BASEDATA.2
ADD H7AFPFILES.ADDITIONALS 15 15
@ED,U 30-BASEDATA.3
ADD H7AFPFILES.ADDITIONALS 15 15
@ED,U 30-BASEDATA.4
ADD H7AFPFILES.ADDITIONALS 15 15
@ED,U 30-BASEDATA.5
ADD H7AFPFILES.ADDITIONALS 16 16
@ED,U 30-BASEDATA.6
ADD H7AFPFILES.ADDITIONALS 16 16
@ED,U 30-BASEDATA.7
ADD H7AFPFILES.ADDITIONALS 16 16
@ED,U 30-BASEDATA.8
ADD H7AFPFILES.ADDITIONALS 16 16
@ED,U 30-BASEDATA.9
ADD H7AFPFILES.ADDITIONALS 15 15
@ED,U 30-BASEDATA.10
ADD H7AFPFILES.ADDITIONALS 15 15
@ED,U 30-BASEDATA.11
ADD H7AFPFILES.ADDITIONALS 15 15
@ED,U 30-BASEDATA.12
ADD H7AFPFILES.ADDITIONALS 15 15
@ED,U 30-BASEDATA.13
ADD H7AFPFILES.ADDITIONALS 16 16
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream (page 5 of 9 pages)

```
@ED,U 30-BASEDATA.14
ADD H7AFPFILES.ADDITIONALS 16 16
@ED,U 30-BASEDATA.15
ADD H7AFPFILES.ADDITIONALS 16 16
@ED,U 30-BASEDATA.16
ADD H7AFPFILES.ADDITIONALS 16 16
EXIT
@ED,QI AFP1.

ADD H7AFPFILES.SIG-EMIT
@XQT *H7AFP.870M-SIGE
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP2.
@ED,U 30-BASEDATA.3
ADD OAFP3.
@ED,U 30-BASEDATA.4
ADD OAFP4.
@ED,U 30-BASEDATA.5
ADD OAFP5.
@ED,U 30-BASEDATA.6
ADD OAFP6.
@ED,U 30-BASEDATA.7
ADD OAFP7.
@ED,U 30-BASEDATA.8
ADD OAFP8.
@ED,U 30-BASEDATA.9
ADD OAFP9.
@ED,U 30-BASEDATA.10
ADD OAFP10.
@ED,U 30-BASEDATA.11
ADD OAFP11.
@ED,U 30-BASEDATA.12
ADD OAFP12.
@ED,U 30-BASEDATA.13
ADD OAFP13.
@ED,U 30-BASEDATA.14
ADD OAFP14.
@ED,U 30-BASEDATA.15
ADD OAFP15.
@ED,U 30-BASEDATA.16
ADD OAFP16.
EXIT
@ED,QI AFP1.

ADD H7AFPFILES.SENSOR-ATTEN
@XQT *H7AFP.870M-ATTEN
MIST
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP2.
@ED,U 30-BASEDATA.3
ADD OAFP3.
@ED,U 30-BASEDATA.4
ADD OAFP4.
@ED,U 30-BASEDATA.5
ADD OAFP5.
@ED,U 30-BASEDATA.6
ADD OAFP6.
@ED,U 30-BASEDATA.7
ADD OAFP7.
@ED,U 30-BASEDATA.8
ADD OAFP8.
@ED,U 30-BASEDATA.9
ADD OAFP9.
@ED,U 30-BASEDATA.10
ADD OAFP10.
@ED,U 30-BASEDATA.11
ADD OAFP11.
@ED,U 30-BASEDATA.12
ADD OAFP12.
@ED,U 30-BASEDATA.13
ADD OAFP13.
@ED,U 30-BASEDATA.14
ADD OAFP14.
@ED,U 30-BASEDATA.15
ADD OAFP15.
@ED,U 30-BASEDATA.16
ADD OAFP16.
EXIT
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream
(page 6 of 9 pages)

```
@ED,QI AFP1.

ADD H7AFPFILES.SENSOR-MAG
@XQT *H7AFP.870M-MAG
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP1.
@ED,U 30-BASEDATA.3
ADD OAFP1.
@ED,U 30-BASEDATA.4
ADD OAFP1.
@ED,U 30-BASEDATA.5
ADD OAFP1.
@ED,U 30-BASEDATA.6
ADD OAFP1.
@ED,U 30-BASEDATA.7
ADD OAFP1.
@ED,U 30-BASEDATA.8
ADD OAFP1.
@ED,U 30-BASEDATA.9
ADD OAFP1.
@ED,U 30-BASEDATA.10
ADD OAFP1.
@ED,U 30-BASEDATA.11
ADD OAFP1.
@ED,U 30-BASEDATA.12
ADD OAFP1.
@ED,U 30-BASEDATA.13
ADD OAFP1.
@ED,U 30-BASEDATA.14
ADD OAFP1.
@ED,U 30-BASEDATA.15
ADD OAFP1.
@ED,U 30-BASEDATA.16
ADD OAFP1.
EXIT
@ED,U 30-BASEDATA.1
ADD H7AFPFILES.ADDITIONALS 17 17
@ED,U 30-BASEDATA.2
ADD H7AFPFILES.ADDITIONALS 17 17
@ED,U 30-BASEDATA.3
ADD H7AFPFILES.ADDITIONALS 17 17
@ED,U 30-BASEDATA.4
ADD H7AFPFILES.ADDITIONALS 17 17
@ED,U 30-BASEDATA.5
ADD H7AFPFILES.ADDITIONALS 17 17
@ED,U 30-BASEDATA.6
ADD H7AFPFILES.ADDITIONALS 17 17
@ED,U 30-BASEDATA.7
ADD H7AFPFILES.ADDITIONALS 17 17
@ED,U 30-BASEDATA.8
ADD H7AFPFILES.ADDITIONALS 17 17
@ED,U 30-BASEDATA.9
ADD H7AFPFILES.ADDITIONALS 18 18
@ED,U 30-BASEDATA.10
ADD H7AFPFILES.ADDITIONALS 18 18
@ED,U 30-BASEDATA.11
ADD H7AFPFILES.ADDITIONALS 18 18
@ED,U 30-BASEDATA.12
ADD H7AFPFILES.ADDITIONALS 18 18
@ED,U 30-BASEDATA.13
ADD H7AFPFILES.ADDITIONALS 18 18
@ED,U 30-BASEDATA.14
ADD H7AFPFILES.ADDITIONALS 18 18
@ED,U 30-BASEDATA.15
ADD H7AFPFILES.ADDITIONALS 18 18
@ED,U 30-BASEDATA.16
ADD H7AFPFILES.ADDITIONALS 18 18
EXIT
@ED,QI AFP1.

ADD H7AFPFILES.PARTICIPATE
@XQT *H7AFP.870M-PART
@ED,U 30-BASEDATA.1
ADD OAFP1.
@ED,U 30-BASEDATA.2
ADD OAFP2.
@ED,U 30-BASEDATA.3
ADD OAFP3.
@ED,U 30-BASEDATA.4
ADD OAFP4.
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream (page 7 of 9 pages)

```
@ED,U 30-BASEDATA.5
ADD OAFP5.
@ED,U 30-BASEDATA.6
ADD OAFP6.
@ED,U 30-BASEDATA.7
ADD OAFP7.
@ED,U 30-BASEDATA.8
ADD OAFP8.
@ED,U 30-BASEDATA.9
ADD OAFP9.
@ED,U 30-BASEDATA.10
ADD OAFP10.
@ED,U 30-BASEDATA.11
ADD OAFP11.
@ED,U 30-BASEDATA.12
ADD OAFP12.
@ED,U 30-BASEDATA.13
ADD OAFP13.
@ED,U 30-BASEDATA.14
ADD OAFP14.
@ED,U 30-BASEDATA.15
ADD OAFP15.
@ED,U 30-BASEDATA.16
ADD OAFP16.
EXIT
@ED,U 30-BASEDATA.1
ADD *H7ARTYEFD.RAPD
@ED,U 30-BASEDATA.2
ADD *H7ARTYEFD.STATIC
@ED,U 30-BASEDATA.3
ADD *H7ARTYEFD.RADE
@ED,U 30-BASEDATA.4
ADD *H7ARTYEFD.BAPD
@ED,U 30-BASEDATA.5
ADD *H7ARTYEFD.RAPD
@ED,U 30-BASEDATA.6
ADD *H7ARTYEFD.STATIC
@ED,U 30-BASEDATA.7
ADD *H7ARTYEFD.RADE
@ED,U 30-BASEDATA.8
ADD *H7ARTYEFD.BAPD
@ED,U 30-BASEDATA.9
ADD *H7ARTYEFD.RAPD
@ED,U 30-BASEDATA.10
ADD *H7ARTYEFD.STATIC
@ED,U 30-BASEDATA.11
ADD *H7ARTYEFD.RADE
@ED,U 30-BASEDATA.12
ADD *H7ARTYEFD.BAPD
@ED,U 30-BASEDATA.13
ADD *H7ARTYEFD.RAPD
@ED,U 30-BASEDATA.14
ADD *H7ARTYEFD.STATIC
@ED,U 30-BASEDATA.15
ADD *H7ARTYEFD.RADE
@ED,U 30-BASEDATA.16
ADD *H7ARTYEFD.BAPD
@ED,U 30-BASEDATA.1
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.2
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.3
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.4
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.5
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.6
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.7
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.8
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.9
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.10
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.11
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.12
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.13
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream (page 8 of 9 pages)

```
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.14
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.15
ADD H7AFPFILES.ADDITIONALS 19 381
@ED,U 30-BASEDATA.16
ADD H7AFPFILES.ADDITIONALS 19 381
EXIT
@ED 30-BASEDATA.1,*H7BASEDATA.HM80E01
@ED 30-BASEDATA.2,*H7BASEDATA.HM80E02
@ED 30-BASEDATA.3,*H7BASEDATA.HM80E03
@ED 30-BASEDATA.4,*H7BASEDATA.HM80E04
@ED 30-BASEDATA.5,*H7BASEDATA.HM80E05
@ED 30-BASEDATA.6,*H7BASEDATA.HM80E06
@ED 30-BASEDATA.7,*H7BASEDATA.HM80E07
@ED 30-BASEDATA.8,*H7BASEDATA.HM80E08
@ED 30-BASEDATA.9,*H7BASEDATA.HM80E09
@ED 30-BASEDATA.10,*H7BASEDATA.HM80E10
@ED 30-BASEDATA.11,*H7BASEDATA.HM80E11
@ED 30-BASEDATA.12,*H7BASEDATA.HM80E12
@ED 30-BASEDATA.13,*H7BASEDATA.HM80E13
@ED 30-BASEDATA.14,*H7BASEDATA.HM80E14
@ED 30-BASEDATA.15,*H7BASEDATA.HM80E15
@ED 30-BASEDATA.16,*H7BASEDATA.HM80E16
EXIT
```

Figure D-I-17. Combat Module Format BASEDATA Conversion Runstream (page 9 of 9 pages)

D-I-19. RUNSTREAM ADJUSTMENT FOR EFD AND PERMANENT BASEDATA ELEMENTS. The analyst should make sure the correct H7ARTYEFD elements (artillery expected fractional damage) are in lines 599-629. Also, the elements in the temporary file 30-BASEDATA must be copied to appropriate elements in H7BASEDATA, lines 663-678.

D-I-20. BASEDATA CONVERSION PROGRAMS. There are 11 programs used to convert labeled BASEDATA information to Combat Module input format. These programs, together with system utility EDITOR routines, must be executed in sequence to create the 16 BASEDATA elements needed for one complete AFP run. Each program will be discussed in detail in the following paragraphs in the order that they are executed. The system EDITOR is used to add single lines of data and sections that do not have labeled format to the BASEDATA elements. See the BASEDATA generator runstream for documentation on the execution of the EDITOR routines. Figures D-I-18 through D-I-28 contain the flowcharts of the conversion programs.

a. Inventory. The source program to convert the labeled inventory to Combat Module input format is H7AFP.M-INV. The absolute program is H7AFP.870M-INV. This program reads the inventory in H7AFPFILES. Inventory elements are labeled by division, year, and sometimes posture. Example, H7AFPFILES.INVJM84 and H7AFPFILES.INVHM80/RAPD. The numbers in the labeled inventory represent one division. Force ratio factors in H7AFPFILES.INV-FACTORS give the number of divisions of each side per posture. Inventory numbers are multiplied by the factors as the model format inventories are being built, except where there is an indicator in the H7AFPFILES inventory element which says not to multiply. Error messages will be displayed if the input inventory number is not in the correct columns. The 16 Combat Module format inventories are created from this program. There is no program to convert a Combat Module formatted inventory into a labeled format.

b. External Losses. The source program to convert the labeled external losses to Combat Module input format is H7AFP.M-EXT-LOSSES. The absolute program is H7AFP.870M-EXTLOSS. This program strips the AFP System type names from the input file and write only the external loss number to the output file. External loss values do not change by environment, so one output file will be used in creating the external loss section for all 16 environments. There is no program to convert Module readable to labeled.

c. Near-Far Ranges. The source program to convert the labeled Near-Far ranges to Combat Module input format is H7AFP.M-NEARFAR. The absolute program is H7AFP.870M-NEARFAR. This program strips the AFP System type names from the input file and writes the near and far range band numbers to the output file. Near-Far ranges do not vary by environment, so one output file is used to create all 16 environments. There is no program to convert Combat Module format to labeled format.

d. Artillery Range Distribution. The source program to convert the labeled artillery ranges to Combat Module input format is H7AFP.M-ARTY-RNG. The absolute program is H7AFP.870M-ARTYRNG. The ranges vary by posture and day/night. Correspondence between the conditions (posture, day/night) and the environments is hardwired in the program. The appropriate set of values is selected from the tables in H7AFPFILES.ARTY-RANGE and moved to the correct output environment file. There is no program to convert Combat Module input format to labeled format.

e. Refire. The source program to convert labeled refire times to Combat Module input format is H7AFP.M-REFIRE. The absolute program is H7AFP.870M-REFIRE. This program removes the names from H7AFPFILES.REFIRE and arrays the refire times, 10 per line, 6 lines per AFP System type. Refire times do not vary by environment. One output file is created and used for all 16 environment output files. There is also a program to convert Combat Module formatted data to labeled format, H7AFP.H-REFIRE. That absolute program is H7AFP.870H-REFIRE. Using the refire values from a Combat Module formatted refire time file, the weapon names from an old refire time file, and the AFP System type names from the inventory file, a second labeled refire time file with the new refire values is created. This program can be used to check the refire values if changes are made directly to the refire section of the Combat Module format.

f. Signature Sought. The source program to convert labeled signature sought values to Combat Module input format is H7AFP.M-SIG-SOUGHT. The absolute program is H7AFP.870M-SIGS. There are four tables of sensor values in H7AFPFILES.SIGNATURE which vary by light conditions--clear day (environment 1-4), clear night (environment 5-8), degraded day (environment 9-12), degraded night (environment 13-16). The appropriate sensor number, type signature sought by the sensor, and the resolvable cycle are written to the correct output environment file. There is no program to convert Combat Module readable format to labeled format.

g. Sensing Size. The source program to convert labeled sensing size data to Combat Module input format is H7AFP.M-SENS-SIZE. The absolute program is H7AFP.870M-SENS-SIZE. The detected size of a target varies when it is in the open or in defilade. A posture file, H7AFPFILES.DEF-OPEN-POS, shows which side is in defilade or in the open by environment. The sensing size table in H7AFPFILES.SENS-SIZE shows sizes in the open and defilade. The appropriate size (OPEN or DEF) for the environment is written to each of the 16 output files. There is no program to convert Combat Module input format to labeled format.

h. Signature Emitted. The source program which converts labeled signature emitted data to Combat Module input format is H7AFP.M-SIG-EMIT. The absolute program is H7AFP.870M-SIGE. The labeled file H7AFPFILES.SIG-EMIT contains both light contrast and heat change data. The values vary by environment--clear day (Environment 1-4), clear night (environment 5-8), degraded day (environment 9-12), degraded night (environment 13-16). The appropriate light and heat signature emitted are written to the correct output environment file. There is no program to convert Combat Module input to labeled format.

i. Sensor Attenuation. The source program which converts labeled attenuation data to Combat Module format is H7AFP.M-ATTEN. The absolute program is H7AFP.870M-ATTEN. The attenuation values of the sensors in H7AFPFILES.SENSOR-ATTEN vary by clear and degraded environments. The degradation condition is entered via a parameter after the @XQT statement for H7AFP.870M-ATTEN as haze, mist, or fog. That condition will be used for all degraded environments. There are three types of sensors--visible light, IR, and SILICON TV. The appropriate value based on sensor type and atmospheric condition is written to each of the 16 environment output files. There is no program to convert Combat Module input format to labeled format.

j. Sensor Magnification. The source program which converts labeled sensor magnification sizes to Combat Module format is H7AFP.M-MAG. The absolute program is H7AFP.870M-MAG. The magnification sizes of the sensors in H7AFPFILES.SENSOR-MAG do not vary by environment. One output file is used in creating the magnification section for all 16 environments. There is no program to convert Combat Module format to labeled format.

k. Participation. The source program which converts labeled artillery participation values to Combat Module input format is H7AFP.M-PARTICIP. The absolute program is H7AFP.870M-PART. Artillery participation can vary by posture (currently it does not). The participation values are selected from the table in H7AFPFILES.PARTICIPATE, and written to the appropriate environment output file. There is no program to convert Combat Module input to labeled format.

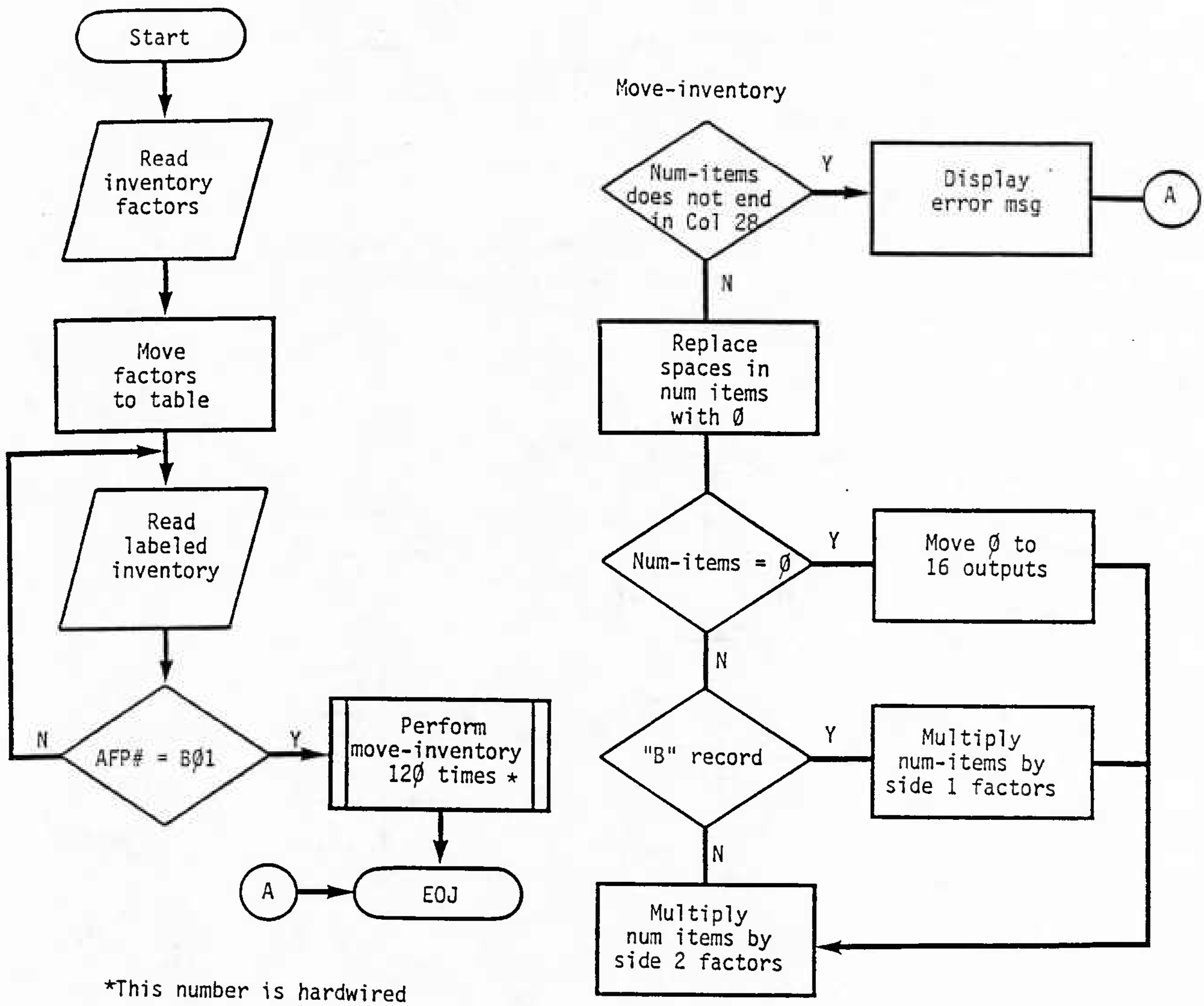


Figure D-I-18. Labeled Inventory to Combat Module Format Conversion Program

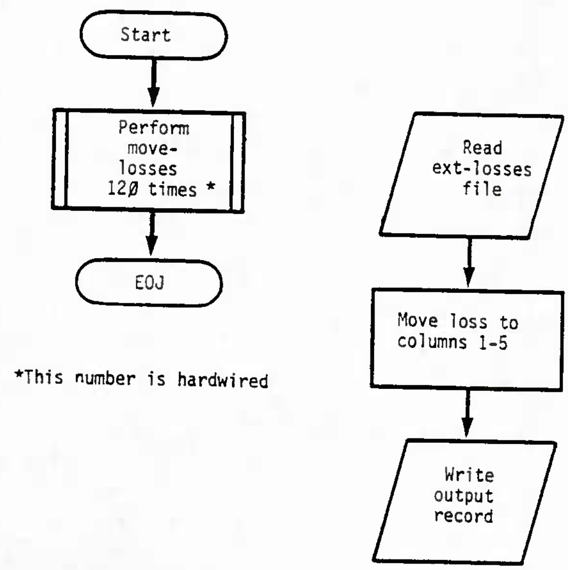


Figure D-I-19. Labeled External Loss to Combat Module Format Conversion Program

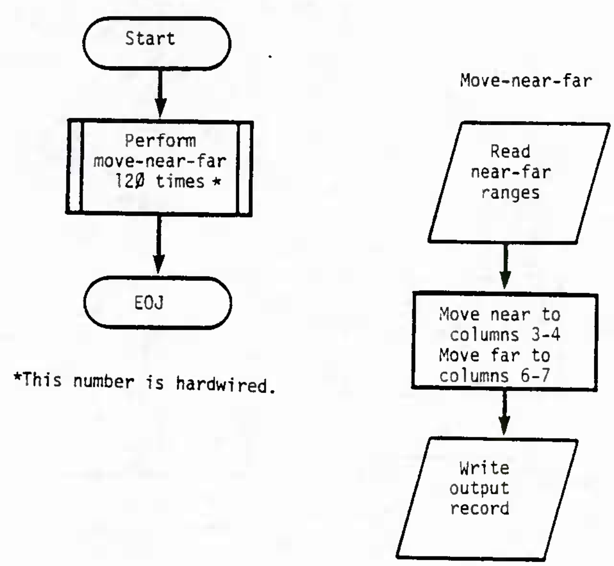


Figure D-I-20. Labeled Near-Far Ranges to Combat Module Format Conversion Program

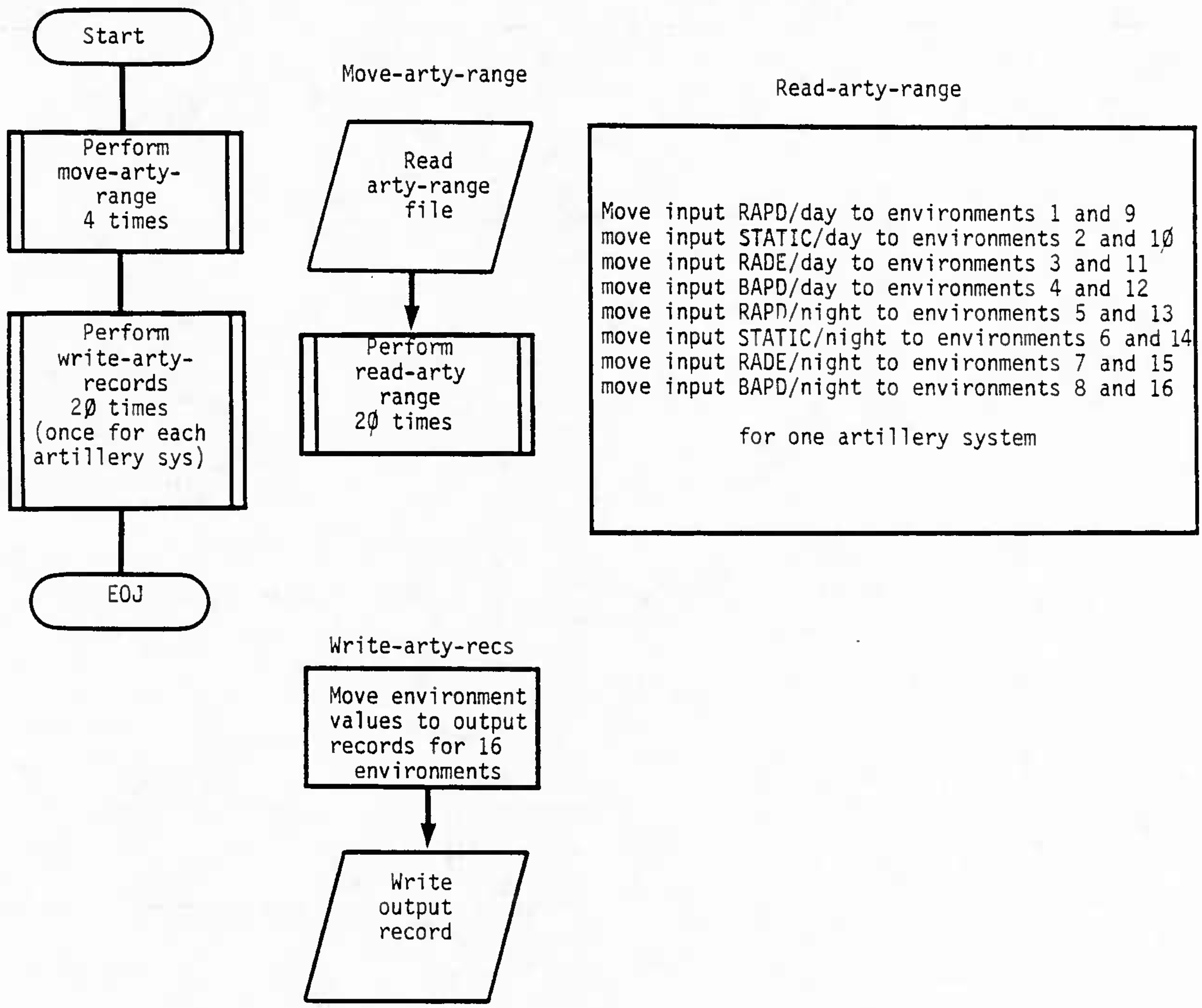


Figure D-I-21. Labeled Artillery Range Distribution to Combat Module Format Conversion Program

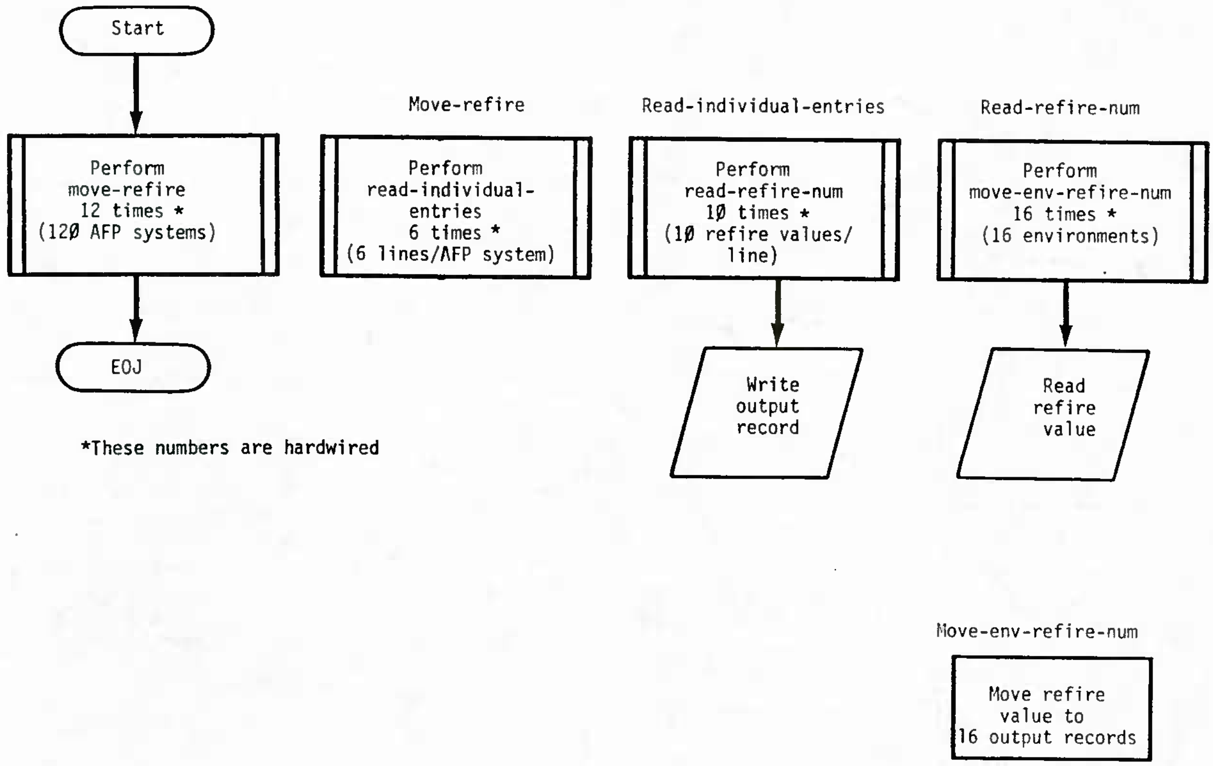


Figure D-I-22. Labeled Refire to Combat Module Format Conversion Program

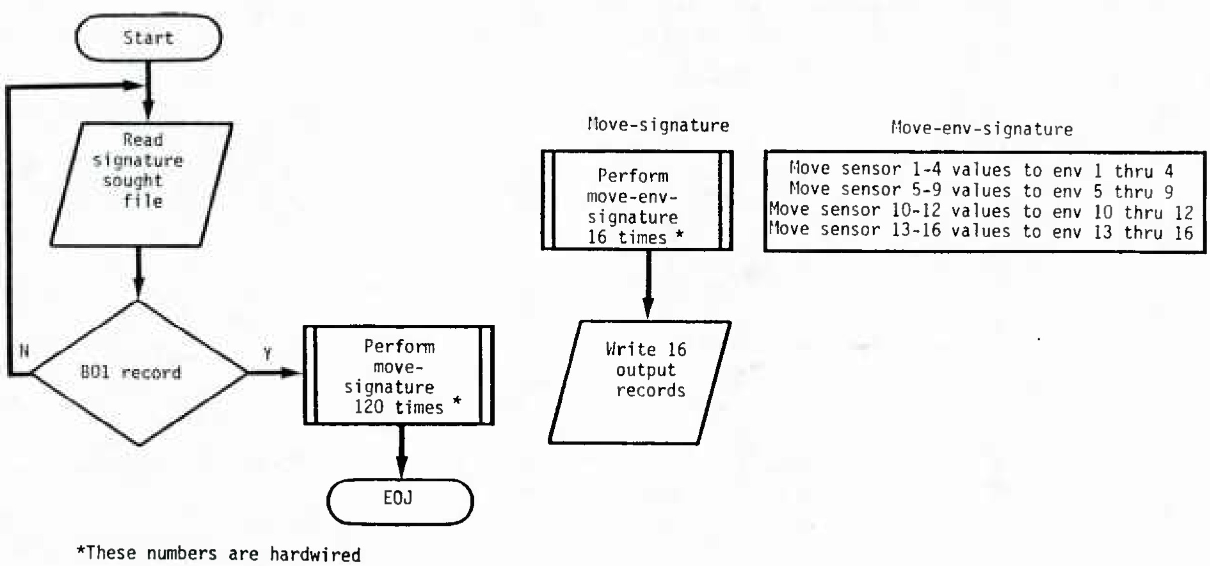


Figure D-I-23. Labeled Signature Sought to Combat Module Conversion Program

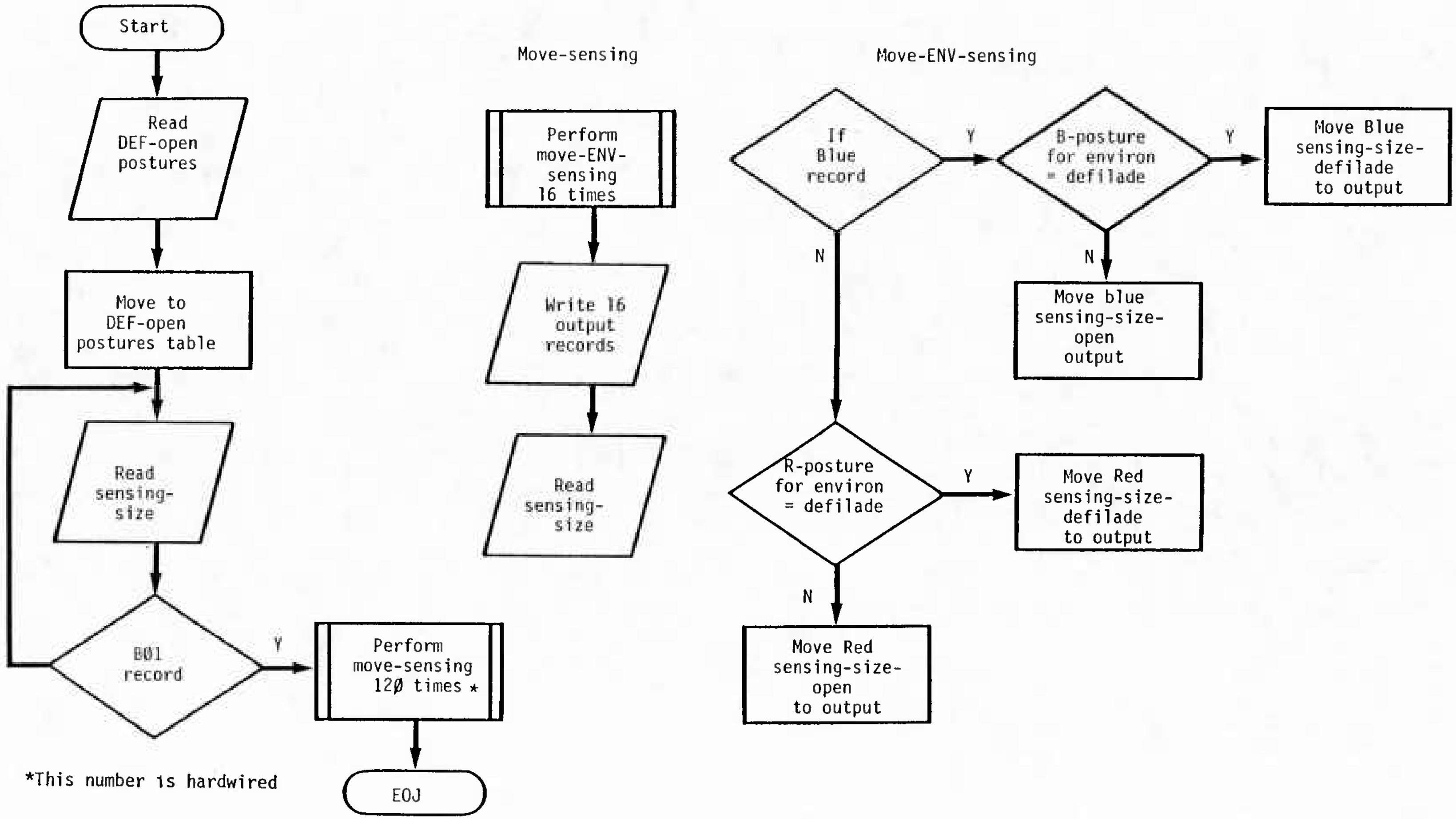


Figure D-I-24. Labeled Sensing Size to Combat Module Format Conversion Program

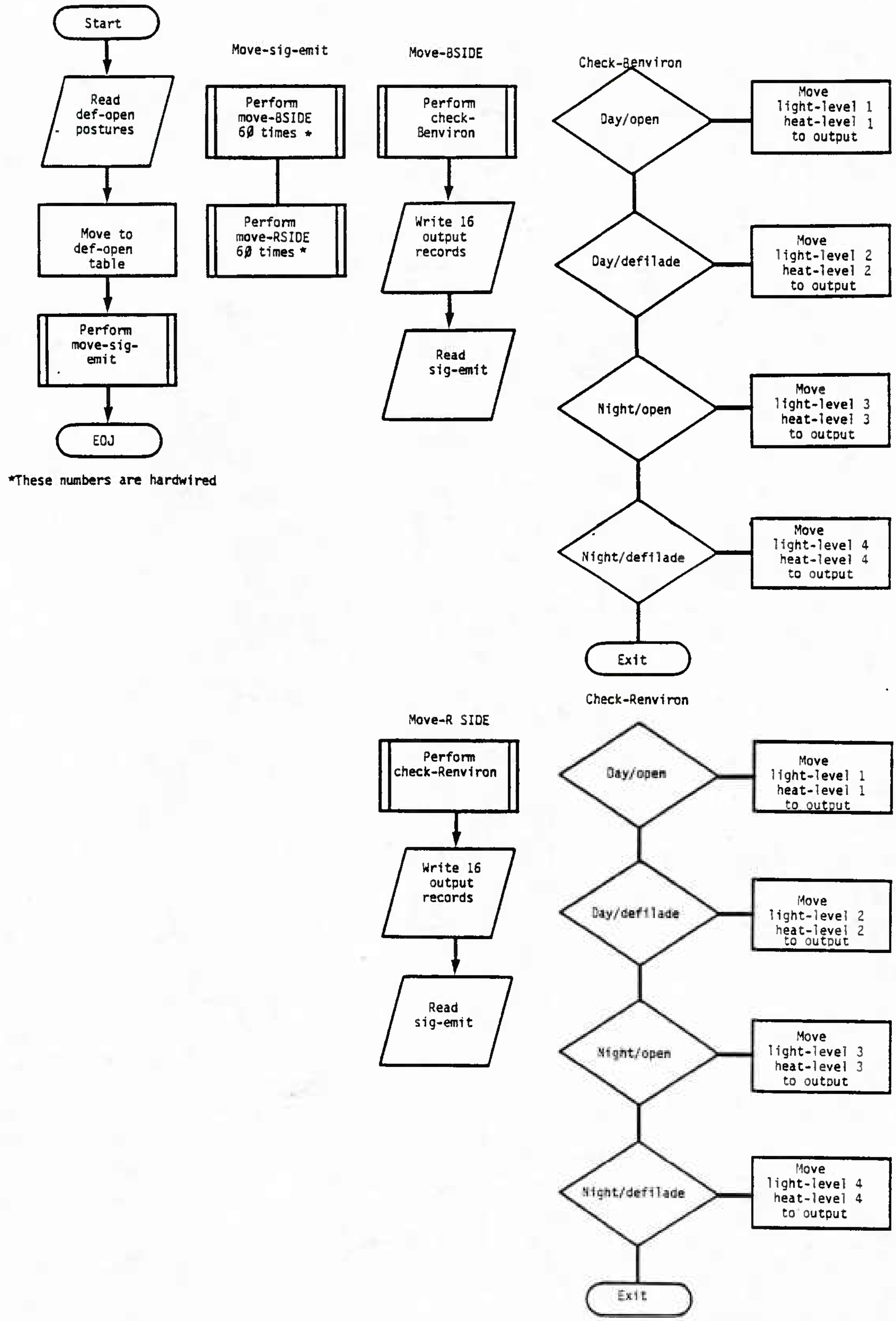


Figure D-I-25. Labeled Signature Emitted to Combat Module Format Conversion Program

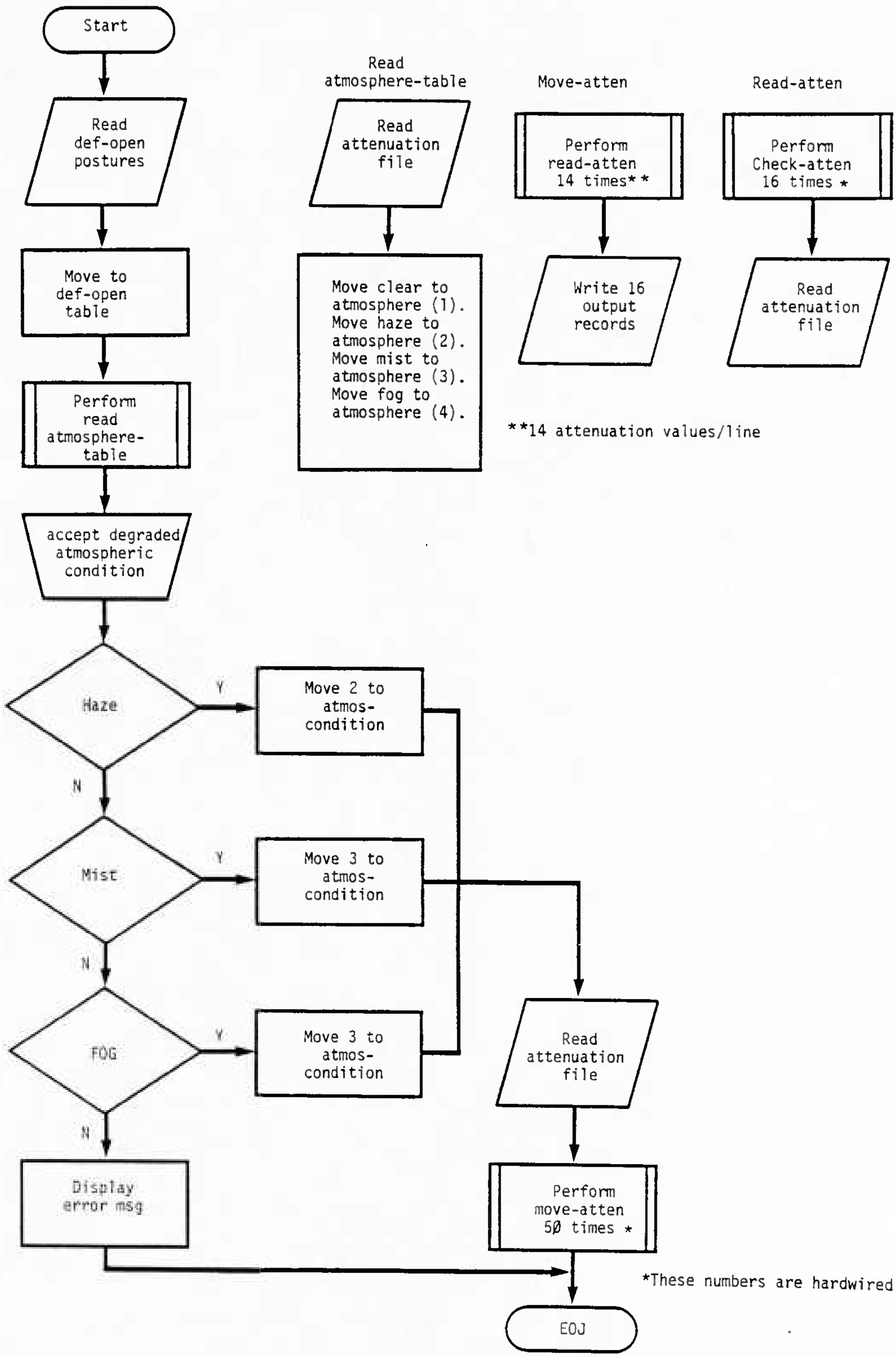


Figure D-I-26. Labeled Sensor Attenuation to Combat Module Format Conversion Program (page 1 of 2 pages)

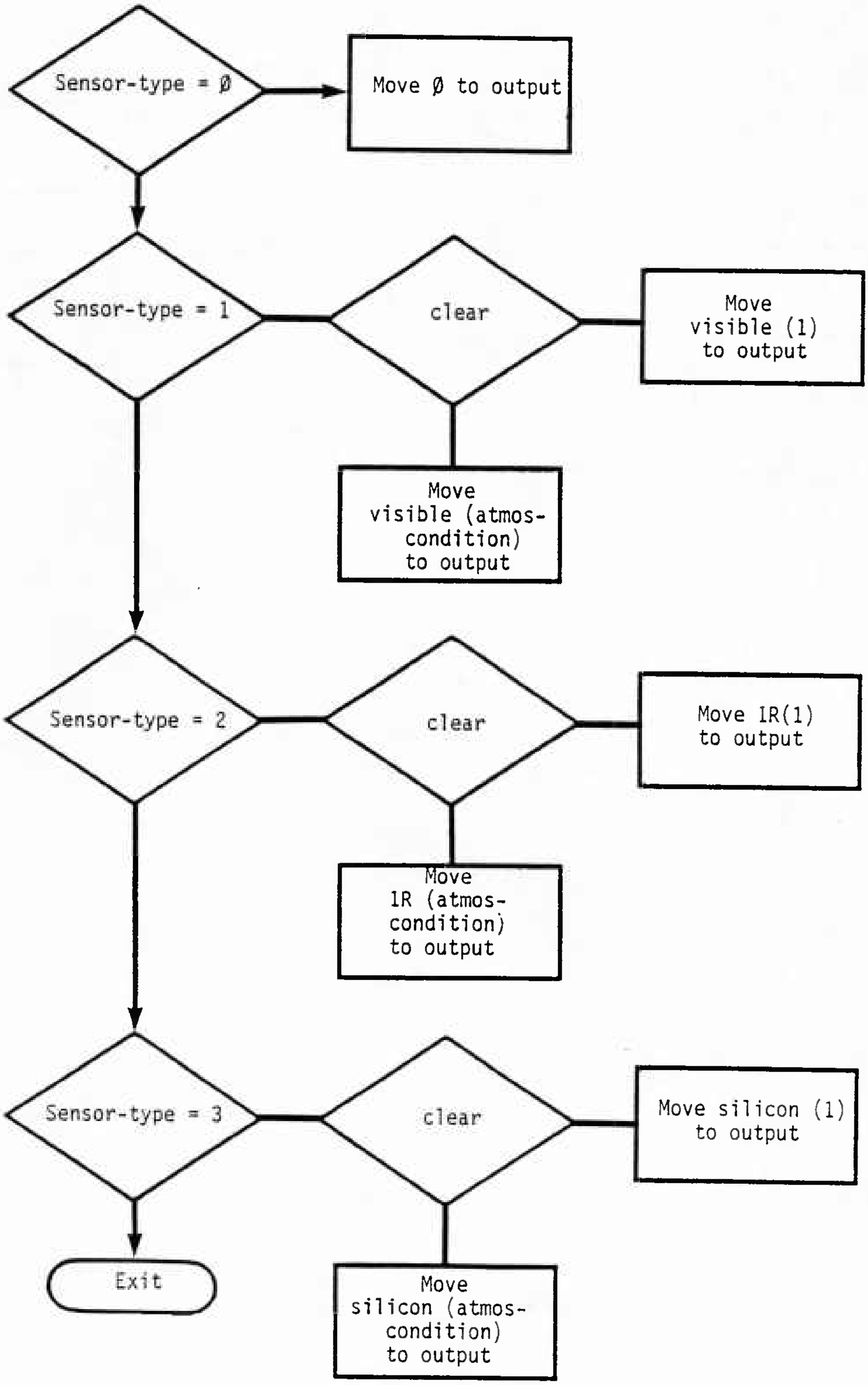


Figure D-I-26. Labeled Sensor Attenuation to Combat Module Format Conversion Program (page 2 of 2 pages)

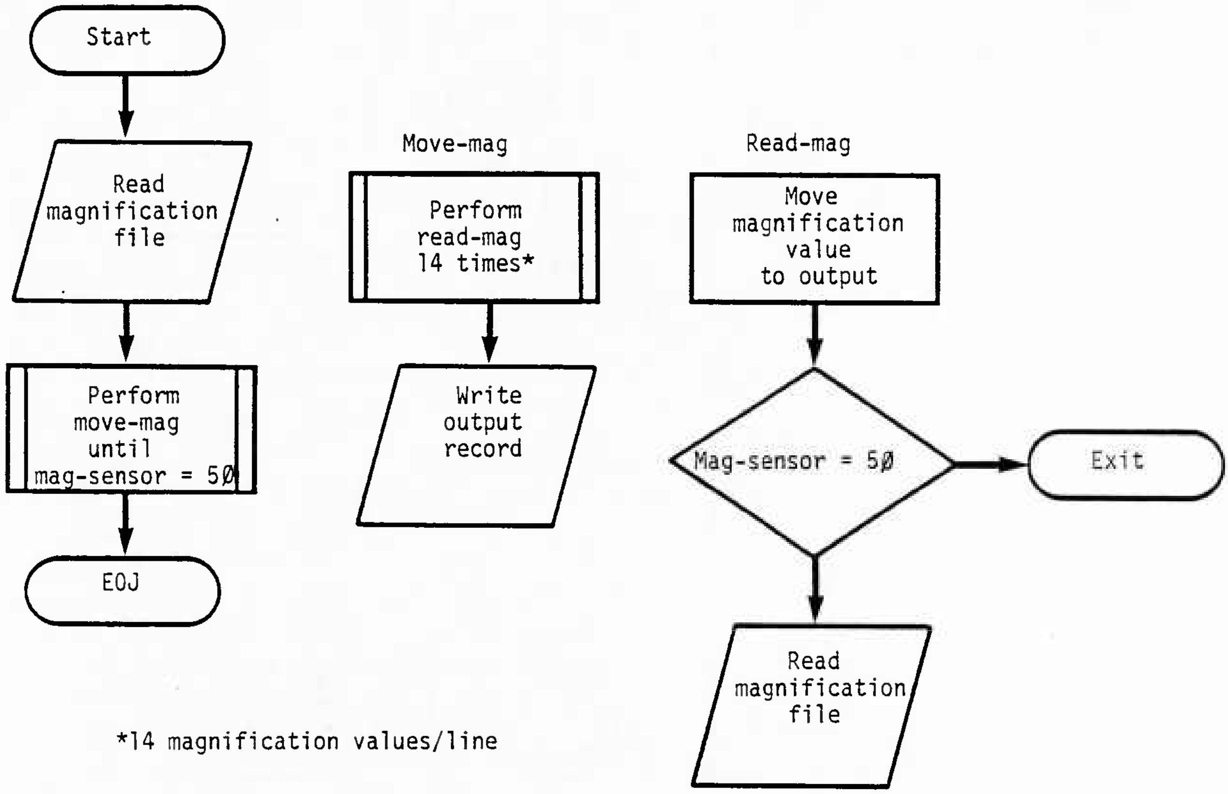


Figure D-I-27. Labeled Sensor Magnification to Combat Module Format Conversion Program

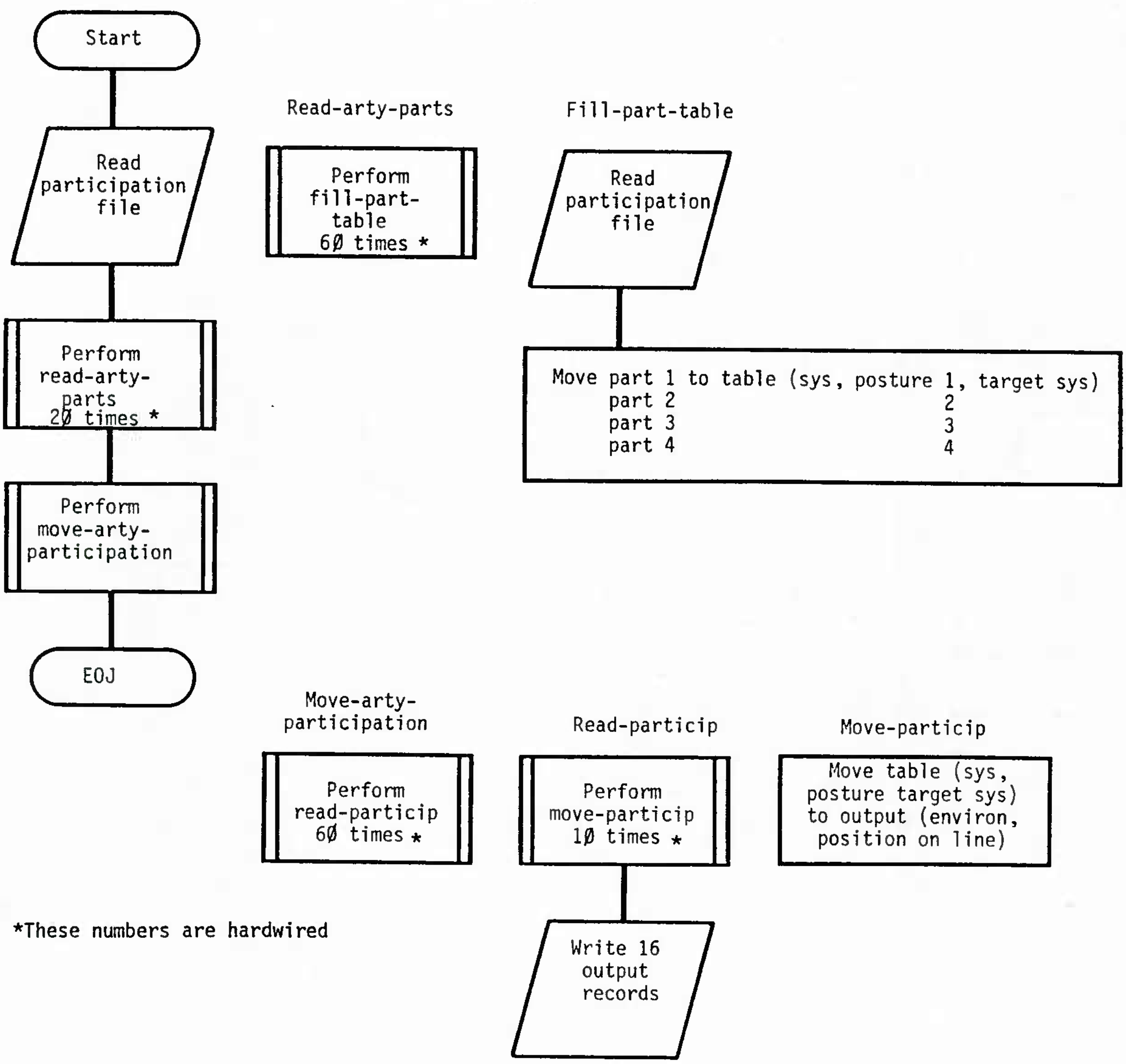


Figure D-I-28. Labeled Participation to Combat Module Format Conversion Program

ANNEX II TO APPENDIX D

AFP ENGAGEMENT PREFERENCES FILE

Section I. OVERVIEW

D-II-1. This annex describes the preference input to the Combat Module. Preferences are needed at the type on type level, i.e., B01 preferences for R01-R60 followed by B02 preferences for R01-R60, and so on for all 60 Blue types. R01 preferences for B01-B60, etc. follow the Blue preferences. A weapon type's preferences to engage the 60 opposing types must total 100 percent. Initial development of the preference file can be facilitated by using a program which accepts category versus category preference inputs and outputs the type versus type preferences required by the Combat Module. Under this scheme, all types within a category have the same preference to engage opposing types of a single category. The output file from this program can be input to the Combat Module directly or can be adjusted for type-on-type module changes. Making these changes can be facilitated by reformatting the output file into a labeled version with a conversion program. This labeled version is changed using the system EDITOR. Another program reads the changed labeled preferences and converts them back to Combat Module format. This annex documents the standard AFP preference set of four Combat Module preference files which relate to the four postures used in AFP. If more or less files are required, the user should adjust the number of preference files accordingly.

Section II. INPUT

D-II-2. A category versus category preference file example shown in Figure D-II-1. There are three logical sections. The first 13 lines are a 13 by 13 array with Side 1 categories 1-13 preferences to engage Side 2 categories 1-13. The next six lines show to which category the Side 1 weapon type belongs. First value is the category of B01, second value is the category of B02, etc. The next line, with 13 numbers, shows the percent of each Side 1 category to be allocated in direct fire duels. These three sections are repeated for Side 2. Category 13 is used in AFP for deep targets. These files are in H7PREFERENCE.RAPD, H7PREFERENCE.STATIC, H7PREFERENCE.RADE, and H7PREFERENCE.BAPD.

D-II-3. A labeled type on type preference file is shown in Figure D-II-2. A Side 1 weapon type's preference to engage each of the 60 Side 2 opponents is listed with the corresponding Side 2 preferences to engage Side 1 in the second column. This file gives preferences from a Side 1 perspective, because all the preferences for a Blue type are shown on one page. It is in H7PREF.LABELED-BLUE. A second file giving preference from a Side 2 perspective is in H7PREF.LABELED-RED. It shows preferences for Side 2 to engage Side 1 in the first preference column, and preferences for Side 2 to engage Side 1 in the next column.

```
 .5, .2,.05, .2, .0, .0, .0,.05, .0, .0, .0, .0, .0,
 .6,.15, .0, .2, .0, .0, .0,.05, .0, .0, .0, .0, .0,
 .0,.05,.15, .2, .6, .0, .0, .0, .0, .0, .0, .0, .0,
.05, .2, .2, .3, .2, .0, .0, .0, .0, .0, .0,.05, .0,
.05,.05,.15, .2, .5, .0, .0,.05, .0, .0, .0, .0, .0,
.0,.05, .0,.05, .0, .0, .0, .6, .3, .0, .0, .0, .0,
 .0, .0, .0, .0, .0, .0, .0, .2, .8, .0, .0, .0, .0,
.10, .1, .0, .2, .5, .1, .0, .0, .0, .0, .0,.0, .0,
 .0,.05,.05, .1, .1, .1, .3, .0, .0, .0, .1, .2, .0,
 .0, .0, .0, .0, .0,.0, .0, .0, .0, .2, .0,.0, .1,
 .0, .0, .0, .0, .0, .0, .0, .0, .0, .1, .2, .0, .1,
 .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0,
 .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, 1.0
 13, 1, 1, 1, 1, 1, 2, 3, 3, 3,
 3, 3, 3, 3, 3, 4, 4, 3, 5, 5,
 5, 5, 5, 5, 5, 6, 6, 3, 3, 3,
 7, 7, 7, 3, 4, 8, 8, 8, 2, 2,
 9,13,13,13,12,12,12,12,12,10,
10,10,10,10,11,11,11,11,11,11
 1., 1., 1., 1., 1., 1., 1., 1., 1., .3,0.4, 1., 1.
.65, .1, .1, .1, .0, .0, .0,.05, .0, .0, .0, .0, .0,
.15, .6, .0, .2, .0, .0, .0,.05, .0, .0, .0, .0, .0,
 .0, .1, .1, .2, .6, .0, .0, .0, .0, .0, .0, .0, .0,
 .1, .2, .2, .3, .2, .0, .0, .0, .0, .0, .0, .0, .0,
.05, .1, .2, .2, .4, .0, .0,.05, .0, .0, .0, .0, .0,
.005, .0,.005, .0, .0, .0, .0,.89, .1, .0, .0, .0, .0,
 .0, .0, .0, .0, .0, .0, .0, .2, .8, .0, .0, .0, .0,
 .0,.05, .1, .3, .4, .1, .0, .0, .0, .0, .0,.05, .0,
 .0,.05, .0,.05, .2, .1, .3, .0, .0, .0, .2, .1, .0,
 .0, .0, .0, .0, .0,.0, .0, .0, .0, .2, .0,.0, .1,
 .0, .0, .0, .0, .0, .0, .0, .0, .0, .1, .2, .0, .1,
 .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0,
 .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, 1.0
 13, 1, 1, 1, 1, 2, 2, 2, 0, 0,
 3, 3, 3, 3, 3, 4, 0, 0, 0, 0,
 5, 5, 5, 5, 5, 6, 0, 0, 0, 7,
 7, 7, 7, 7, 7, 8, 8, 8, 0, 0,
 9,13,13,13, 0,12, 0, 0, 0, 0,
10,10,0, 0, 0,11,11,11, 0, 0
 1., 1., 1., 1., 1., 1., 1., 1., 1., .3,0.4, 1., 1.
```

Figure D-II-1. Category versus Category Preferences

SIDE 1 SYSTEM: 7.62GM AFP # B03

SIDE2 SYSTEM	AFP #	PREFERENCES: SIDE1-FOR-SIDE2	SIDE2-FOR-SIDE1
RTRP D	R01	.000	.000
AKS-74	R02	.156	.123
7.60GM	R03	.156	.123
RSNIPE	R04	.156	.123
GREN30	R05	.156	.123
BRDM-2	R06	.063	.045
BTR-60	R07	.063	.045
BMP-R	R08	.063	.045
DUMMY	R09	.000	.000
DUMMY	R10	.000	.000
SPG-9	R11	.016	.000
RPG-16	R12	.016	.000
AT-5	R13	.016	.000
AT-7	R14	.016	.000
RPG 7	R15	.016	.000
BMP2M	R16	.063	.017
DUMMY	R17	.000	.000
DUMMY	R18	.000	.000
DUMMY	R19	.000	.000
DUMMY	R20	.000	.000
T55	R21	.000	.007
T62	R22	.000	.007
T64	R23	.000	.007
T72	R24	.000	.007
T80	R25	.000	.007
ZSU-23	R26	.000	.002
DUMMY	R27	.000	.000
DUMMY	R28	.000	.000
DUMMY	R29	.000	.000
SA-14	R30	.000	.000
SA-6	R31	.000	.000
SA-7	R32	.000	.000
SA-8	R33	.000	.000
SA-9	R34	.000	.000
SA-11	R35	.000	.000
HOPLIT	R36	.016	.000
HIP-E	R37	.016	.000
HIND-D	R38	.016	.000
DUMMY	R39	.000	.000
DUMMY	R40	.000	.000
MIG-21	R41	.000	.000
LT VEH	R42	.000	.000
HVY AR	R43	.000	.000
LT ARM	R44	.000	.000
DUMMY	R45	.000	.000
ACRV-2	R46	.000	.000
DUMMY	R47	.000	.000
DUMMY	R48	.000	.000
DUMMY	R49	.000	.000
DUMMY	R50	.000	.000
M0B2M	R51	.000	.000
M120T	R52	.000	.000
DUMMY	R53	.000	.000
DUMMY	R54	.000	.000
DUMMY	R55	.000	.000
H122T/	R56	.000	.000
H152T/	R57	.000	.000
L122T+	R58	.000	.000
DUMMY	R59	.000	.000
DUMMY	R60	.000	.000

Figure D-II-2. Labeled Blue Preferences

Section III. OUTPUT

D-II-4. An example of a preference file in Combat Module format is in Figure D-II-3. The preferences are in groups of 4x15 arrays. B01's preferences to engage R01-R15 are in line 1, preferences to engage R16-R30 in line 2, preferences to engage R31-R45 in line 3, and preferences to engage R46-R60 in line 4. B02's preferences follow, and so on. After all the Blue preferences, the Red preferences, R01 versus B01-B60 etc. are listed in the same 15x4 format. These preference file are in H7PREF.RAPD, H7PREF.STATIC, H7PREF.RADE, and H7PREF.BAPD.

1	.250	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
2	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
3	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.250	.250	.250	.000
4	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
5	.000	.156	.156	.155	.156	.063	.063	.063	.000	.000	.016	.016	.016	.016	.016
6	.063	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
7	.000	.000	.000	.000	.000	.016	.016	.016	.000	.000	.000	.000	.000	.000	.000
8	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
9	.000	.156	.156	.155	.156	.063	.063	.063	.000	.000	.016	.016	.016	.016	.016
10	.063	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
11	.000	.000	.000	.000	.000	.016	.016	.016	.000	.000	.000	.000	.000	.000	.000
12	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
13	.000	.156	.156	.155	.156	.063	.063	.063	.000	.000	.016	.016	.016	.016	.016
14	.063	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
15	.000	.000	.000	.000	.000	.016	.016	.016	.000	.000	.000	.000	.000	.000	.000
16	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Figure D-II-3. Combat Model Format Preference

Section IV. RUNSTREAM

D-II-5. Overall I/O flow of the preference generation process is shown in Figure D-II-4. Elements in H7PREF which contain the category versus category preferences are read by the program H7AFP.870CAT-PREF generating a type versus type preference element in H7PREF. That element is converted to a labeled format in programs H7AFP.870H-PREF-B and H7AFP.870H-PREF-R. Changes are made to the labeled elements with the system EDITOR. The changed elements are read by H7AFP.870M-PREF and are written over the old type versus type elements in H7PREF. The runstreams for each of the pieces of the process are described in the following paragraphs.

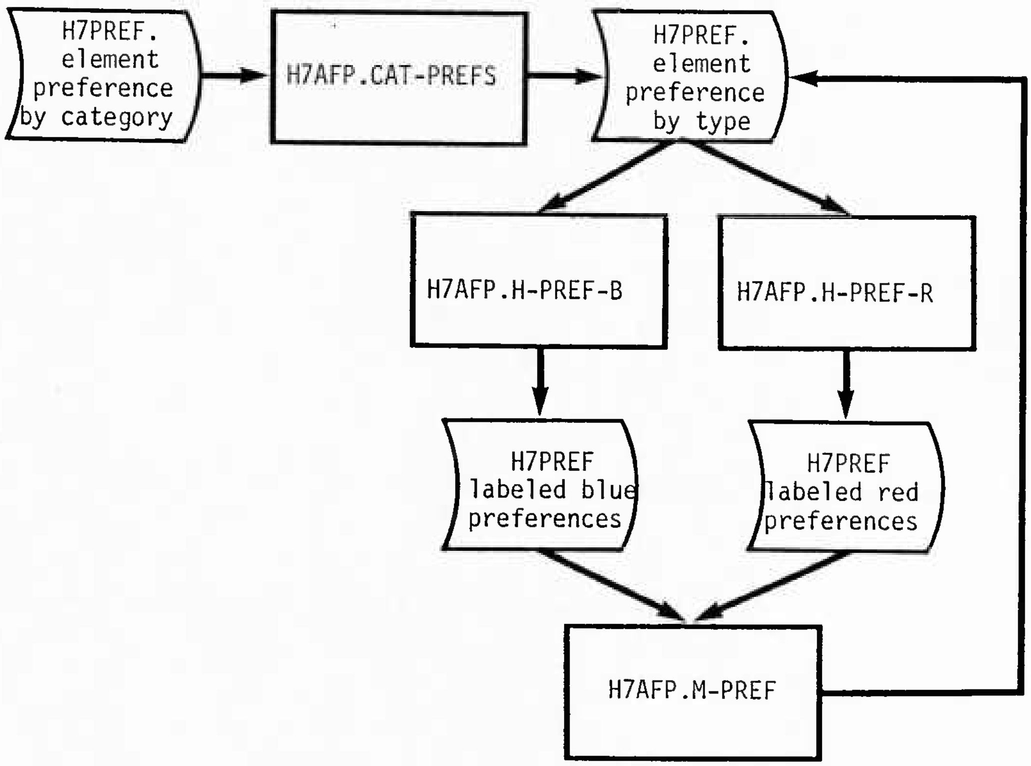


Figure D-II-4. Preference Generation I/O Flow

D-II-6. Figure D-II-5 shows the runstream to convert category preferences to type versus type, Combat Module format.

```
@XQT *H7AFP.87JCAT-PREF
@ADD *H7PREF.CAT-RAPD/H
@ED 3.,*H7PREF.RAPD/H
```

Figure D-II-5. Convert Category to Type Preference

D-II-7. Figure D-II-6 shows the runstream to convert type versus type preferences to labeled formats, both a Blue and a Red perspective.

```
@QUAL UNCLASSIFIED
@ASG,A H7AFPFILES/XXX/XXX.
@ASG,T AFP1.
@ED,GI AFP1.

ADD H7AFPFILES.INVHM80/RAPD
@ASG,T AFP2.
@ED,GI AFP2.

ADD *H7PREF.RAPD/H
@ASG,T OAFP1.
@XQT *H7AFP.870H-PREF-B
@ED OAFP1.,*H7PREF.LABELED-BLUE/RAPDH
@XQT *H7AFP.870H-PREF-R
@ED OAFP1.,*H7PREF.LABELED-RED/RAPDH
EXIT
```

Figure D-II-6. Convert Combat Module Format to Labeled Format

D-II-8. Figure D-II-7 shows the runstream to convert labeled Blue perspective preferences to Combat Module format. Use this if changes were only made to the Blue perspective file. Figure D-II-8 shows the runstream to convert labeled Red perspective preferences to Combat Module format. Use this if changes were only made to the Red perspective file. Figure D-II-9 shows the runstream to use if both the Blue and Red perspective files were changed. Many times it is easier to make Blue changes in the Blue file and not adjust Red preferences in that file because the Red preferences are spread across 60 pages.

```
@ASG,T AFP1.
@ED,GI AFP1.

ADD *H7PREF.LABELED-BLUE/RAPDH
@ASG,T OAFP1.
@XQT *H7AFP.870M-PREF
@ED OAFP1.,*H7PREF.RAPD/H
@ASG,T 12.
@ED *H7PREF.RAPD/H,12.
@XQT *H7AFP.PREFCHK
60 60
@ASG,A H7AFPFILES/XXX/XXX.
@ED,GI AFP1.

ADD H7AFPFILES.INVHM80/RAPD
@ASG,T AFP2.
@ED,GI AFP2.

ADD *H7PREF.RAPD/H
@XQT *H7AFP.870H-PREF-B
@ED OAFP1.,*H7PREF.LABELED-BLUE/RAPDH
EXIT
```

Figure D-II-7. Convert Labeled Blue Preferences

```
@ASG,T AFP1.
@ED,QI AFP1.

ADD *H7PREF.LABELED-RED/RAPDH
@ASG,T OAFP1.
@XQT *H7AFP.870M-PREF
@ED OAFP1.,*H7PREF.RAPD/H
@ASG,T 12.
@ED *H7PREF.RAPD/H,12.
@XQT *H7AFP.PREFCHK
60 60
@ASG,A H7AFPFILES/XXX/XXX.
@ED,QI AFP1.

ADD H7AFPFILES.INVHM80/RAPD
@ASG,T AFP2.
@ED,QI AFP2.

ADD *H7PREF.RAPD/H
@XQT *H7AFP.870H-PREF-R
@ED OAFP1.,*H7PREF.LABELED-RED/RAPDH
@XIT
```

Figure D-II-8. Convert Labeled Red Preferences

```
@ASG,T AFP1.
@ED,QI AFP1.

ADD *H7PREF.LABELED-BLUE/RAPDH
@ASG,T OAFP1.
@XQT *H7AFP.870M-PREF
@ED OAFP1.,*H7PREF.RAPD/H
@ED,QI AFP1.

ADD *H7PREF.LABELED-RED/RAPDH
@XQT *H7AFP.870M-PREF
@ED,U *H7PREF.RAPD/H
D 241 480
ADD OAFP1. 1 240
EXI
@ASG,T 12.
@ED *H7PREF.RAPD/H,12.
@XQT *H7AFP.PREFCHK
60 60
@ASG,A H7AFPFILES/XXX/XXX.
@ED,QI AFP1.

ADD H7AFPFILES.INVHM80/RAPD
@ASG,T AFP2.
@ED,QI AFP2.

ADD *H7PREF.RAPD/H
@XQT *H7AFP.870H-PREF-B
@ED OAFP1.,*H7PREF.LABELED-BLUE/RAPDH
@ASG,A *H7PRT1.
@USE P1.,*H7PRT1.
@XQT *H7AFP.870TOP
HM80 BLUE PREFS
@SYM,U P1.
@FREE P1.
@XQT *H7AFP.870H-PREF-R
@ED OAFP1.,*H7PREF.LABELED-RED/RAPDH
@ASG,A *H7PRT2.
@USE P1.,*H7PRT2.
@XQT *H7AFP.870TOP
HM80 RED PREFS
@SYM,U P1.
```

Figure D-II-9. Convert Labeled Blue and Red Preferences

Section V. PROGRAMS

D-II-9. The program which converts category versus category preferences to type versus type is H7AFP.CAT-PREF. The absolute program is H7AFP.870CAT-PREF. If participation and preferences by category are nonzero, the preference is proportioned among the opposing category. The program's I/O flow is in Figure D-II-10. The program flowchart is shown in Figure D-II-11, and variable definitions are in Table D-II-1.

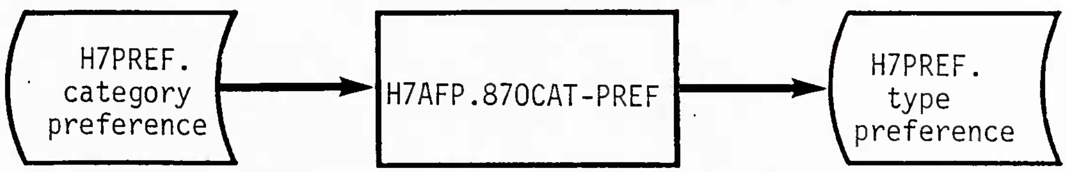
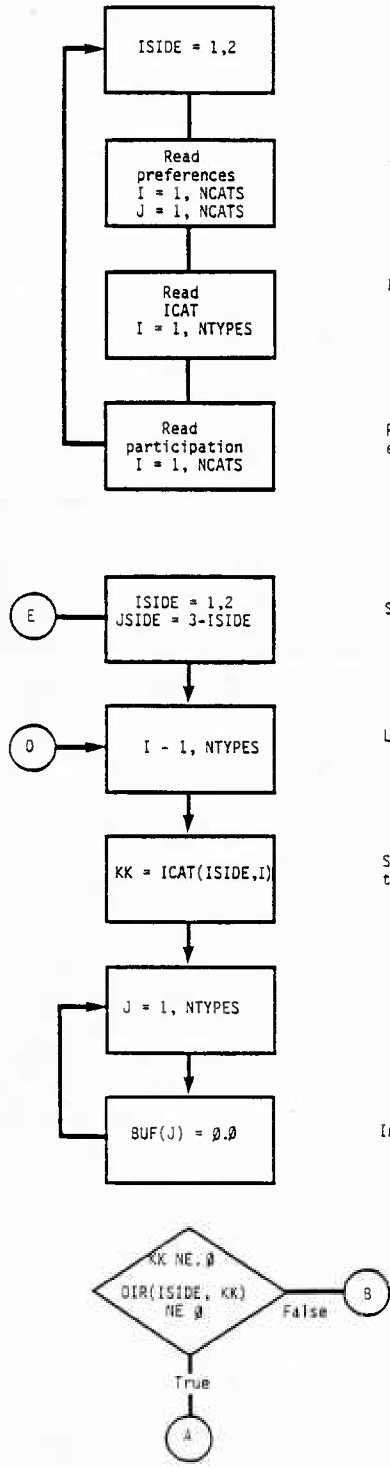


Figure D-II-10. H7AFP.CAT-PREF I/O Flow

D-II-10. There are two programs which convert the Combat Module preferences to labeled format, H7AFP.H-PREF-BLUE creates a labeled version from a Blue perspective, i.e., Side 1 (Blue) preferences are grouped by Blue type versus 60 Red types. H7AFP.H-PREF-RED creates a labeled version from a Red perspective. The absolute programs are H7AFP.870H-PREF-B and H7AFP.870H-PREF-R. I/O flow for the programs are in Figures D-II-12 and D-II-13. H-PREF-BLUE's flowchart is in Figure D-II-14. H-PREF-RED's flowchart is in Figure D-II-15.



Read preferences to engage opposite side categories for each category type.

Read the category classifications for each type.

Read the direct fire participation percentage for each category

Set shooter side and target side.

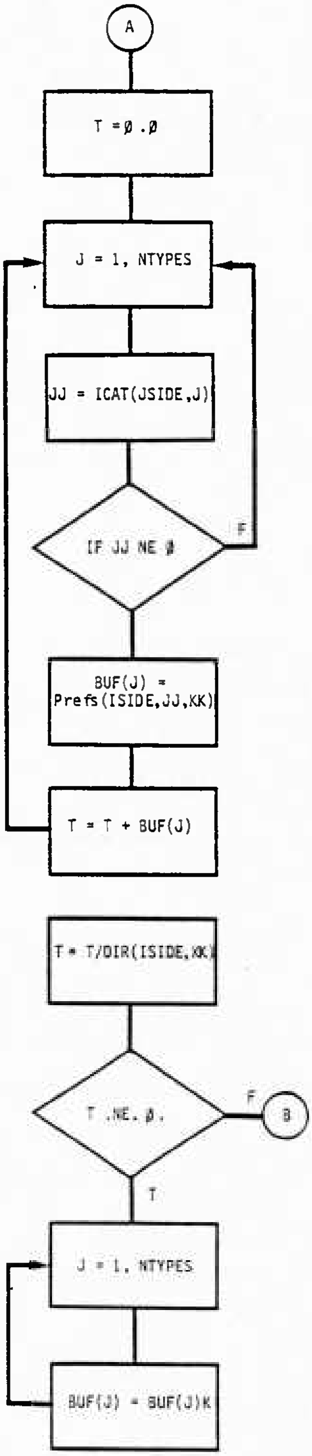
Loop over the shooter side.

Set pointer for shooter category. Give KK the value of the shooter category

Initialize output array at 0.0.

Check to see if shooter has a nonzero category preference for the target category and that participation is not equal to zero. If either category preference or participation is zero then write the type preference as zero.

Figure D-II-11. H7AFP.CAT-PREF Flowchart
(page 1 of 3 pages)



Loop over the target side.

Set pointers for target type. Give JJ the value of the target category.

If the target type is zero, go to the next type.

Give BUF(J) the shooter type KK category preference for target type JJ.

Sum the preferences for shooter type KK over all target types.

Divide the sum of the shooter preferences by the direct KK participation percentage.

If the adjusted shooter type KK preference for all target types is zero, write the preference and go to the next type.

Proportion the total shooter type KK category preference across the NTYPE targets.

Figure D-II-11. H7AFP.CAT-PREF Flowchart
(page 2 of 3 pages)

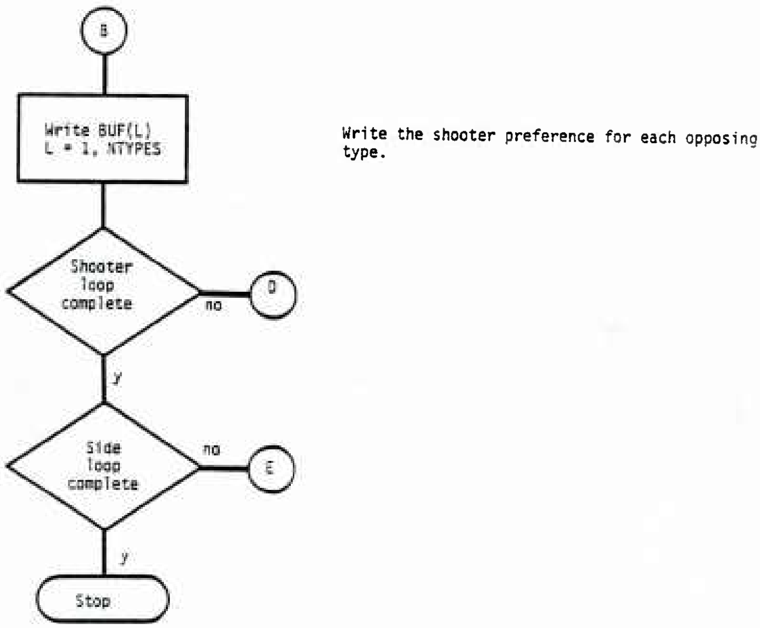


Figure D-II-11. H7AFP.CAT-PREF Flowchart (page 3 of 3 pages)

Table D-II-1. Variable Definitions

NCATS	Number of categories on each side
NTYPES	Number of weapon types on each side
PREFS	A three-dimensional array which contains the input preferences for each of the sides to engage each other on a category-by-category basis
ICAT	A two-dimensional array which contains the category classifications for each type weapon on each side
BUF	An array which contains the output values - preferences for each of the sides to engage each other on a weapon type by weapon type basis
DIR	A two-dimensional array which contains the direct fire participation percentage for each category on each side
KK	A pointer for the shooter category
JJ	A pointer for the target category
T	Variable used to transform the preference for each category into type-on-type preferences

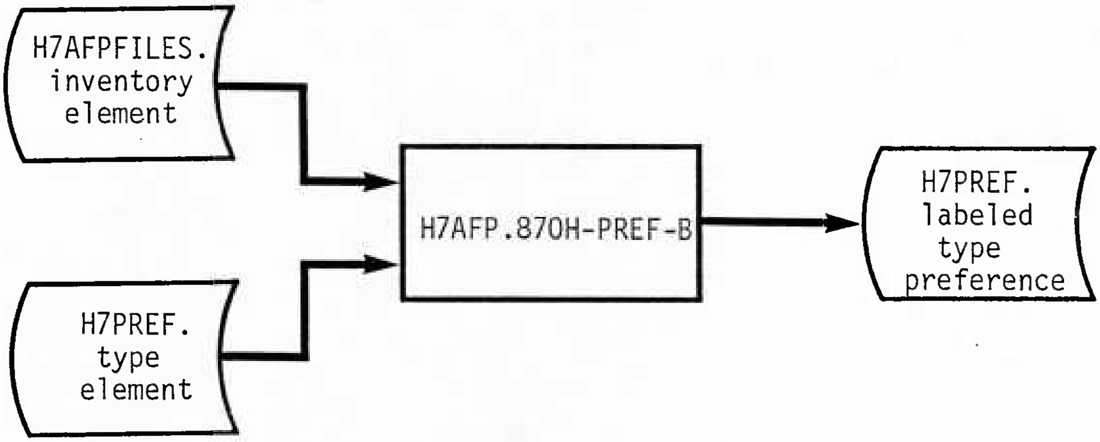


Figure D-II-12. H7AFP.H-PREF-BLUE I/O Flow

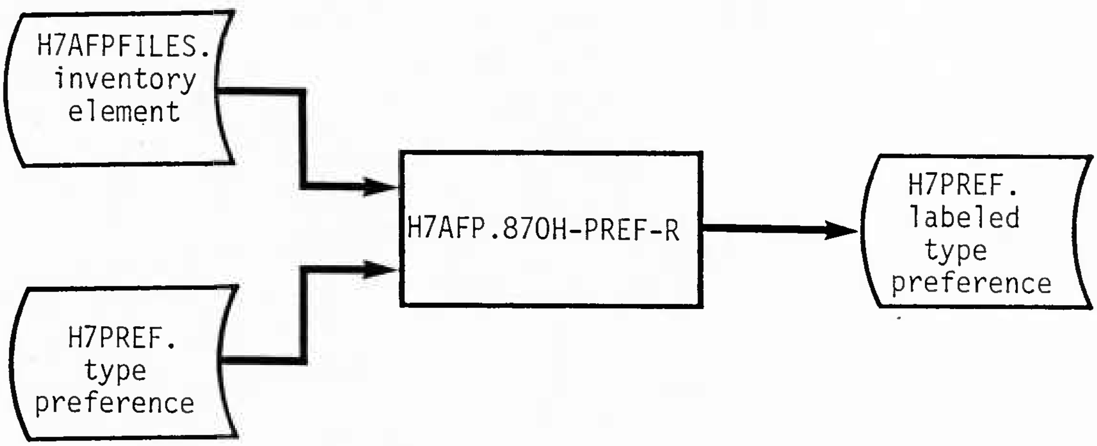


Figure D-II-13. H7AFP.H-PREF-RED I/O Flow

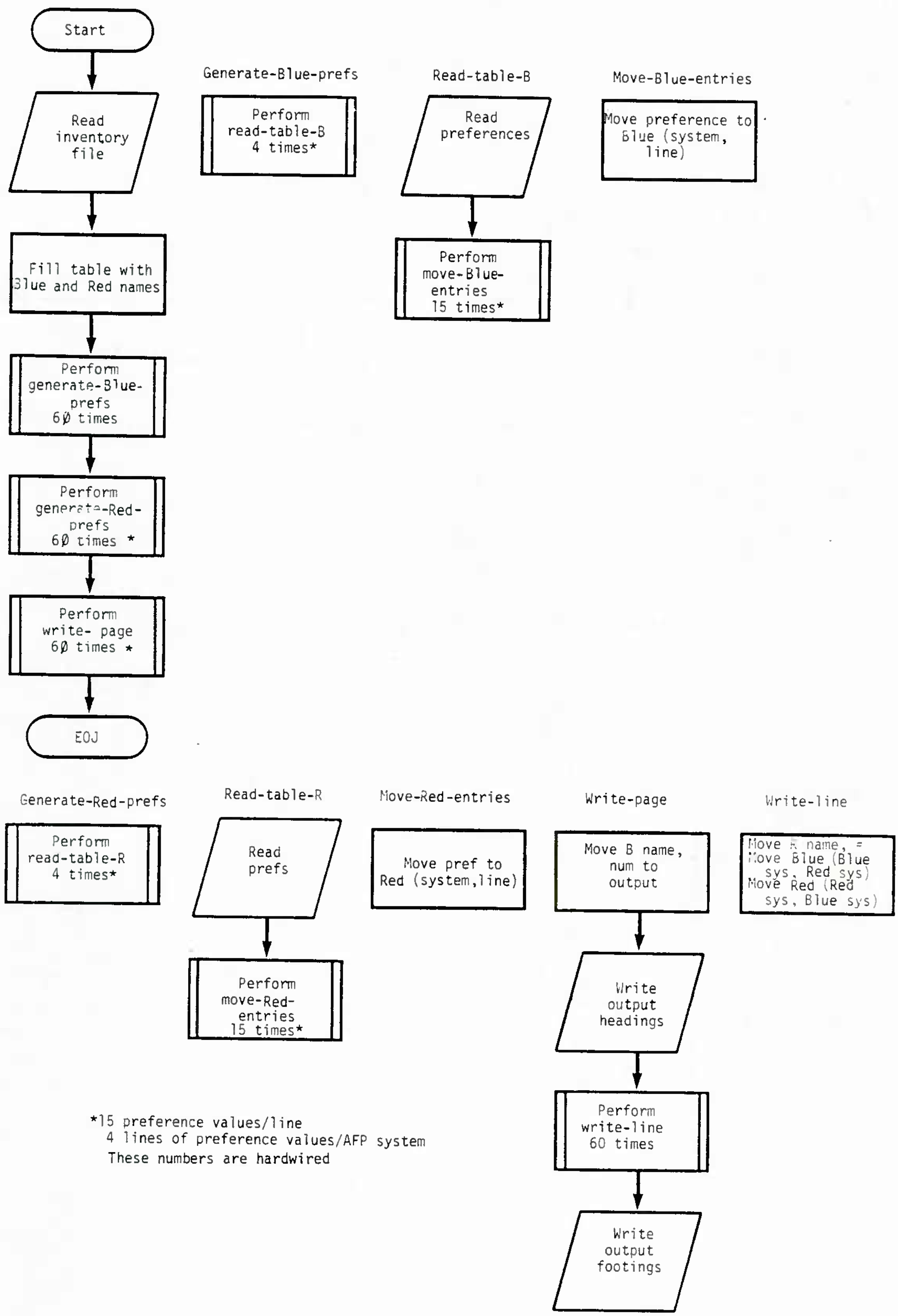


Figure D-II-14. H7AFP.H-PREF-BLUE Flowchart

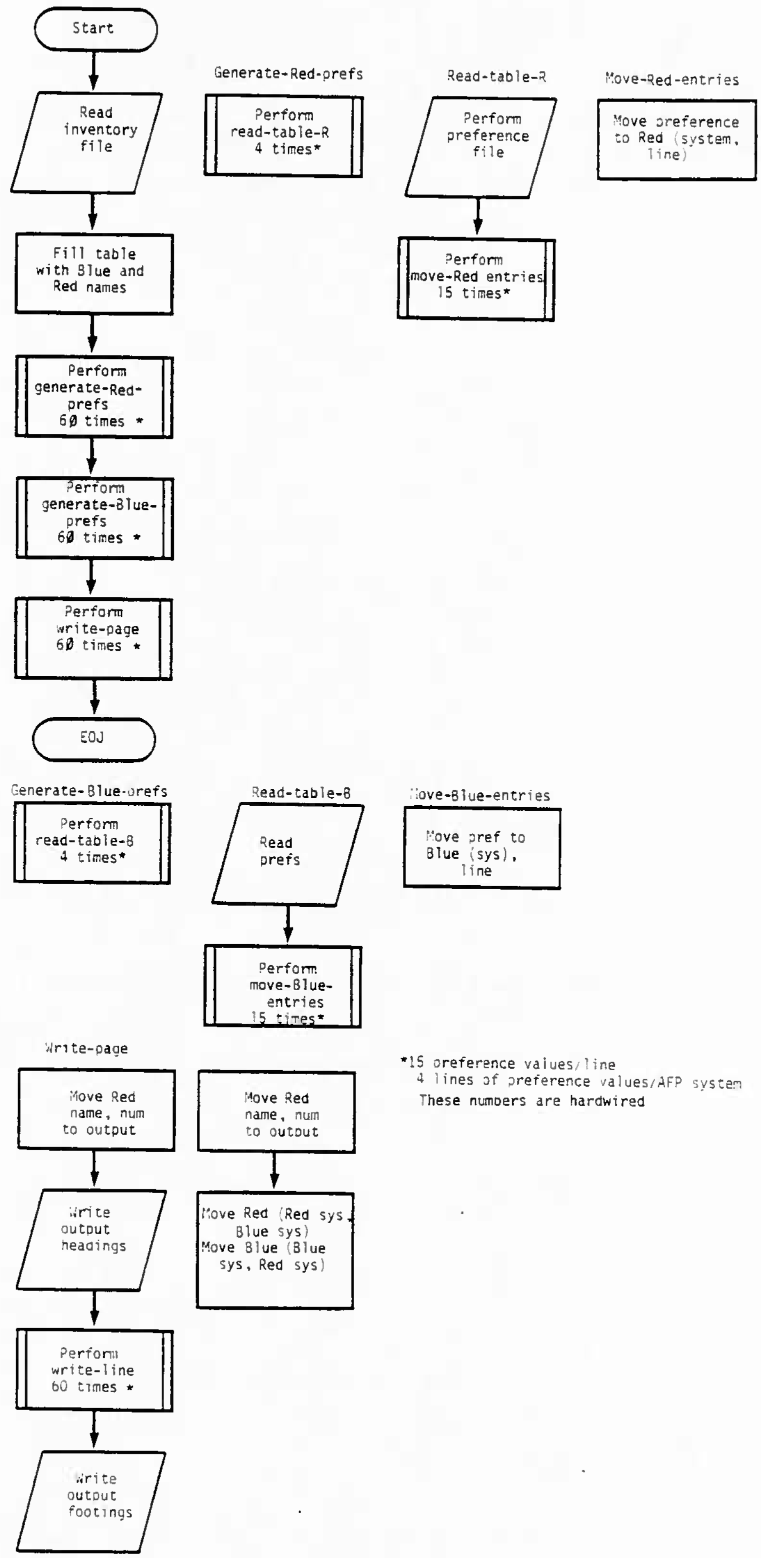


Figure D-II-15. H7AFP.H-PREF-RED Flowchart

D-II-11. There is one program which converts the labeled preferences back to Combat Module format. H7AFP.M-PREF reads the Blue perspective output from H7AFP.H-PREF-BLUE, the Red perspective output from H7AFP.H-PREF-RED. The absolute program is H7AFP.870M-PREF. Depending on how changes are made to the labeled preferences, this program may be executed once or twice. If changes are only made to the BLUE preference file, use the runstream in Figure D-II-7. If only RED changes were made, use the runstream in Figure D-II-8. However, sometimes it is easier to make BLUE changes in the BLUE file and RED changes in the RED file. Then use the runstream in Figure D-II-9. I/O flow for the program is in Figure D-II-16. The flowchart is in Figure D-II-17.

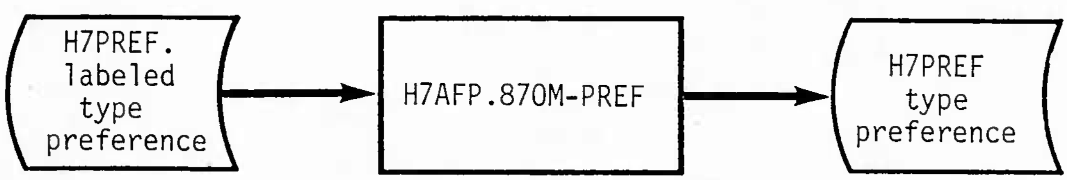


Figure D-II-16. H7AFP.M-PREF I/O Flow

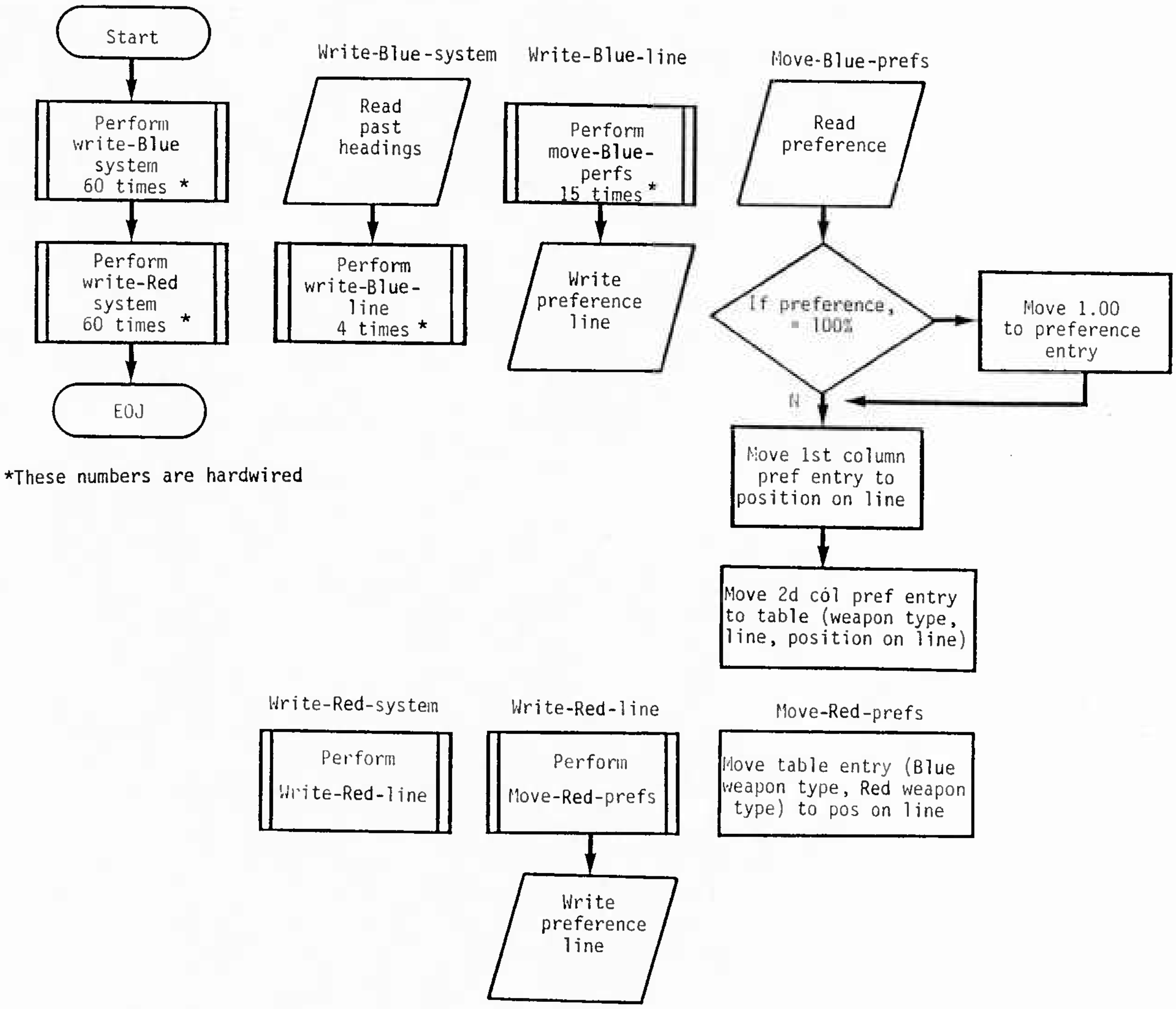


Figure D-II-17. H7AFP.M-PREF Flowchart

ANNEX III TO APPENDIX D

THE DISTRIBUTION OF DUELS TO RANGE BANDS WITHIN THE COMBAT MODULE

Section I. OVERVIEW

D-III-1. This annex describes the input file which serves as the mechanism for distributing the type-on-type duels across the six range bands employed in the Combat Module. It is important to note that each duel must have a range distribution input.

D-III-2. There are currently six range bands and one environment in each Combat Module run. The range distribution file, H7RNGDST, is used to distribute the duels at each odds class to the six ranges. The six ranges are delineated as follows:

Range 1	Range 2	Range 3	Range 4	Range 5	Range 6
250 meters	500 meters	1,000 meters	1,500 meters	2,500 meters	≥2,500 meters

A different range distribution of duels could be used for each AFP environment if deemed appropriate. Presently, only four are utilized, varying across the four combat postures. The distinct posture distributions are contained as elements in the file H7RNGDST and are employed as reflected at Table D-III-1.

Table D-III-1. Range Distributions Across Postures

	Environment	File	Element
1	Clear day Red attack prepared defense	H7RNGDST	RAPD
2	Clear day static	H7RNGDST	STATIC
3	Clear day Red attack Blue delay	H7RNGDST	RADE
4	Clear day Blue attack prepared defense	H7RNGDST	BAPD
5	Clear night Red attack prepared defense	H7RNGDST	RAPD
6	Clear night static	H7RNGDST	STATIC
7	Clear night Red attack Blue delay	H7RNGDST	RADE
8	Clear night Blue attack prepared defense	H7RNGDST	BAPD
9	Degraded day Red attack prepared defense	H7RNGDST	RAPD
10	Degraded day static	H7RNGDST	STATIC
11	Degraded day Red attack Blue delay	H7RNGDST	RADE
12	Degraded day Blue attack prepared defense	H7RNGDST	BAPD
13	Degraded night Red attack prepared defense	H7RNGDST	RAPD
14	Degraded night static	H7RNGDST	STATIC
15	Degraded night Red attack Blue delay	H7RNGDST	RADE
16	Degraded night Blue attack prepared defense	H7RNGDST	BAPD

D-III-3. RANGE DISTRIBUTION OF DUELS

a. The input range distribution exerts a powerful influence on the outcome of the direct fire duels. The range distribution, in conjunction with the engagement preferences and SSPKs, form an interrelated, complex data set that determines the results of all direct fire battles.

b. The range distribution and the engagement preference are subjectively determined by applying the doctrine and tactics of the appropriate Blue and Red force. The determination of which weapons will engage each other, and the ranges over which the engagements will be fought, is based in the combat posture (attack, defend, delay, static) being simulated by the Combat Module.

D-III-4. The range distribution files exist in two different formats: a machine readable format actually used as input to the Combat Module, and a labeled format used to facilitate data transcription and review. Additionally, there are two utility programs used to transform the data files from machine format to labeled format and vice versa. Figure D-III-1 illustrates the range distribution generation process. Source worksheets are developed using military judgment and doctrine to distribute the type-on-type duels to range bands. These data are keyed into the labeled file format. The program which converts from labeled to combat module format is then utilized to convert to the input format required by the Combat Module completing the process. A separate utility may be used to convert any extant Combat Module format range distribution file to the labeled format. This second utility program is not normally used in constructing the range distribution files but does constitute a convenient means of labeling any combat module format version for which the labeled version has not been saved or is not readily identifiable. The ensuing sections describe the data file formats and the utility programs in more detail.

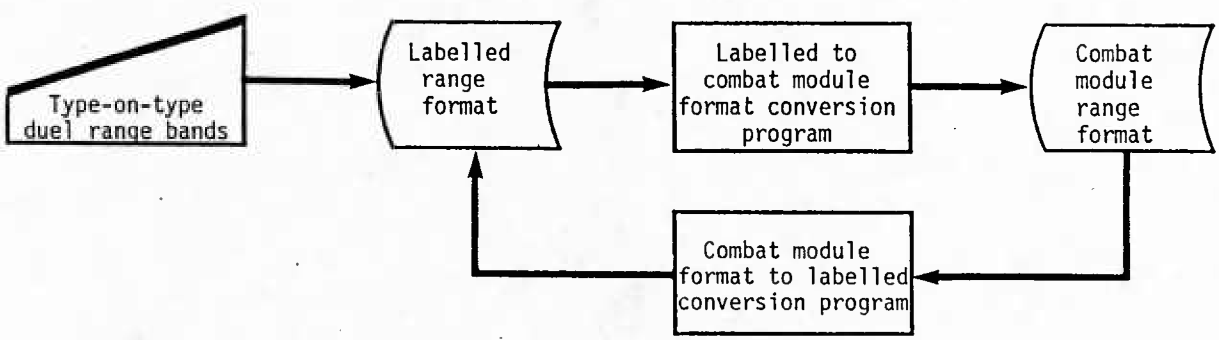


Figure D-III-1. Range Distribution Generation

Section II. INPUT

D-III-5. Figure D-III-2 is an example of the labeled range distribution file. This example portrays Side 1, type 1 (B01 BTRP) versus each of the Side 2 types, i.e., R01 through R60. The values shown are the percentage of duels (between the types) to be distributed to each of the six ranges. For this example, the duels between B01 and R01 would all be in the deep range while 70 percent of the B01 versus R02 duels would be at 250 meters, 20 percent at 500 meters, and 10 percent at 1,000 meters. Table D-III-2 depicts the file organization. Note that the file layout does not show Side 1 types nested under Side 2 types. This is because there can only be one distribution of a type-on-type duel, i.e., B01 versus R01 is the same as R01 versus B01 and requires only one distribution input. The labeled file is read by utility program, H7AFP.870M-RANGE, to produce the machine-readable version which is read by the Combat Module.

SIDE 1 AFP #	TYPE	SIDE 2 AFP #	TYPE	RANGE DISTRIBUTION (% OF CONFLICTS) 250	500	1000	1500	2500	DEEP
B11	DRAGON	R01	RTRP DEEP	90	10	0	0	0	0
		R02	AKS-74	70	20	10	0	0	0
		R03	7.60GM	90	10	0	0	0	0
		R04	RSNIPE	90	10	0	0	0	0
		R05	GREN30	90	10	0	0	0	0
		R06	BRDM-2	40	40	20	0	0	0
		R07	BTR-60	40	40	20	0	0	0
		R08	BMP-R	40	40	20	0	0	0
		R09	DUMMY	20	20	20	20	20	0
		R10	DUMMY	20	20	20	20	20	0
		R11	SPG-9	20	20	20	20	20	0
		R12	RPG-16	20	20	20	20	20	0
		R13	AT-5	40	60	0	0	0	0
		R14	AT-7	20	20	20	20	20	0
		R15	RPG 7	100	0	0	0	0	0
		R16	BMP2M	50	40	10	0	0	0
		R17	DUMMY	20	20	20	20	20	0
		R18	DUMMY	20	20	20	20	20	0
		R19	DUMMY	20	20	20	20	20	0
		R20	DUMMY	20	20	20	20	20	0
		R21	T55	50	50	0	0	0	0
		R22	T62	50	50	0	0	0	0
		R23	T64	50	50	0	0	0	0
		P24	T72	50	50	0	0	0	0
		R25	T80	50	50	0	0	0	0
		R26	ZSU-23	10	50	40	0	0	0
		R27	DUMMY	20	20	20	20	20	0
		R28	DUMMY	20	20	20	20	20	0
		R29	DUMMY	20	20	20	20	20	0
		R30	SA-14	20	20	20	20	20	0
		R31	SA-6	20	20	20	20	20	0
		R32	SA-7	20	20	20	20	20	0
		R33	SA-8	20	20	20	20	20	0
		R34	SA-9	20	20	20	20	20	0
		R35	SA-11	20	20	20	20	20	0
		R36	HOPLITE	20	20	20	20	20	0
		R37	HIP-E	20	20	20	20	20	0
		R38	HIND-D	20	20	20	20	20	0
		R39	DUMMY	20	20	20	20	20	0
		R40	DUMMY	20	20	20	20	20	0
		R41	MIG-21	20	20	20	20	20	0
		R42	LT VEH D	20	20	20	20	20	0
		R43	HVY ARM D	20	20	20	20	20	0
		R44	LT ARM D	20	20	20	20	20	0
		R45	DUMMY	20	20	20	20	20	0
		R46	ACRV-2 C&C	20	20	20	20	20	0
		R47	DUMMY	20	20	20	20	20	0
		R48	DUMMY	20	20	20	20	20	0
		R49	DUMMY	20	20	20	20	20	0
		R50	DUMMY	20	20	20	20	20	0
		R51	M082M	20	20	20	20	20	0
		R52	M120T	20	20	20	20	20	0
		R53	DUMMY	20	20	20	20	20	0
		R54	DUMMY	20	20	20	20	20	0
		R55	DUMMY	20	20	20	20	20	0
		R56	H122T/S	20	20	20	20	20	0
		R57	H152T/S	20	20	20	20	20	0
		R58	L122T+	20	20	20	20	20	0
		R59	DUMMY	20	20	20	20	20	0
		R60	DUMMY	20	20	20	20	20	0

Figure D-III-2. Labeled Range Distribution (percent of conflicts)

Table D-III-2. Range Distribution File Organization

Side 1, Type 1

Side 2 Type 1
Side 2 Type 2
Side 2 Type 3
Thru
Side 2 Type 60

Side 1, Type 2

Side 2 Type 1
Side 2 Type 2
Side 2 Type 3
Thru
Side 2 Type 60
Thru

Side 1, Type 60

Side 2 Type 1
Side 2 Type 2
Side 2 Type 3
Thru
Side 2 Type 60

Section III. OUTPUT

D-III-6. Figure D-III-3 is an example of the machine-readable or Combat Module format of the range distribution file produced by H7AFP.870M-RANGE. The first 15 lines of the file distribute the duels between Side 1, type 1 and Side 2, types 1 through 60. The first six values on line 1 are the percentage of duels between Side 1, type 1 and Side 2, type 1 distributed to each of the six range bands. The subsequent values on line 1 are for Side 1, type 1 versus Side 2, type 2 followed by Side 1, type 1 versys Side 2, type 3 and Side 1, type 1 versus Side 2, type 4. The file organization is identical to that of the labeled version shown at Figure D-III-2.

90	10	0	0	0	0	70	20	10	0	0	0	90	10	0	0	0	0	90	10	0	0	0	0
90	10	0	0	0	0	40	40	20	0	0	0	40	40	20	0	0	0	40	40	20	0	0	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
40	60	0	0	0	0	20	20	20	20	20	0	100	0	0	0	0	0	50	40	10	0	0	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
50	50	0	0	0	0	50	50	0	0	0	0	50	50	0	0	0	0	50	50	0	0	0	0
50	50	0	0	0	0	10	50	40	0	0	0	20	20	20	20	20	0	20	20	20	20	20	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0
20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0	20	20	20	20	20	0

Figure D-III-3. Combat Module Format Range Distribution

D-III-7. Utility program H7AFP.870H-RANGE takes as input the Combat Module format version of the range distribution file and creates the labeled version illustrated at Figure D-III-2.

Section IV. RUNSTREAM

D-III-8. Figure D-III-4 displays a sample runstream used to execute H7AFP.870M-RANGE to transform the range distribution file from labeled format to combat module format. Note that the element name for file H7RNGDST. must be changed in the runstream to reflect the appropriate element for the posture, i.e., RAPD, STATIC, RADE, or BAPD. Figure D-III-5 provides an overview of the runstream process. H7AFP.RNGCHK is executed after H7AFP.870M-RANGE to verify that the percentage of conflicts distributed across the six ranges total to unity. Error messages will show the side and types which do not total to 100 percent $\pm$ .01. Those ranges must be corrected or the AFP Main module will not execute. There is no message if all ranges are correct. The last section of this runstream needs the range distribution elements which have been created and generates a near-far range band element to be used in Basedata. The near values are the closets range. across all postures, at which the weapon type will shoot. The far value are the farthest ranges, across all postures.

```
@ASG,A H7AFPFILES/XXX/XXX.
@ASG,T AFP1.
@ED,GI AFP1.

ADD *H7RNGDST.LABEL-RAPD/H
@ASG,T OAFP1.
@XQT *H7AFP.870M-RANGE
@ASG,T 11.
@ED OAFP1.,11.
@XQT *H7AFP.RNGCHK
60 60
@ED OAFP1.,*H7RNGDST.RAPD-DEEP/H
@ED,GI AFP1.

ADD *H7RNGDST.LABEL-STATIC/H
@ASG,T OAFP1.
@XQT *H7AFP.870M-RANGE
@ASG,T 11.
@ED OAFP1.,11.
@XQT *H7AFP.RNGCHK
60 60
@ED OAFP1.,*H7RNGDST.STATIC-DEEP/H
@ED,GI AFP1.

ADD *H7RNGDST.LABEL-RADE/H
@ASG,T OAFP1.
@XQT *H7AFP.870M-RANGE
@ASG,T 11.
@ED OAFP1.,11.
@XQT *H7AFP.RNGCHK
60 60
@ED OAFP1.,*H7RNGDST.RADE-DEEP/H
@ED,GI AFP1.

ADD *H7RNGDST.LABEL-BAPD/H
@ASG,T OAFP1.
@XQT *H7AFP.870M-RANGE
@ASG,T 11.
@ED OAFP1.,11.
@XQT *H7AFP.RNGCHK
60 60
@ED OAFP1.,*H7RNGDST.BAPD-DEEP/H
@ED,GI AFP1.

ADD H7AFPFILES.INVHM80/RAPD
@ASG,T AFP2.
@ED,GI AFP2.

ADD *H7RNGDST.RAPD-DEEP/H
ADD *H7RNGDST.BAPD-DEEP/H
ADD *H7RNGDST.RADE-DEEP/H
ADD *H7RNGDST.STATIC-DEEP/H
@XQT *H7AFP.870RNGNF
@ED OAFP1.,H7AFPFILES.NEAR-FAR
```

Figure D-III-4. Convert Labeled Range Distribution to Combat Module Format

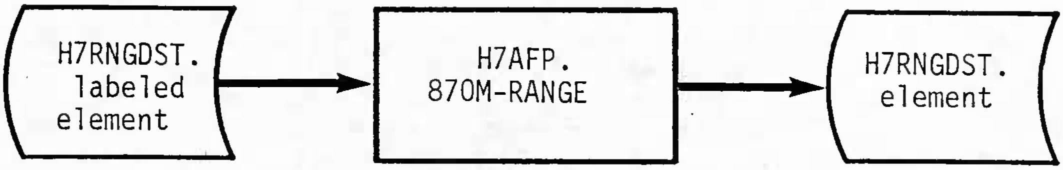


Figure D-III-5. Convert Labeled Range Distribution to Combat Module Format, I/O Flow

D-III-9. Figure D-III-6 is an example of a runstream used to convert a range distribution file from Combat Module format to a labeled format. This runstream need not be used if the labeled versions of the file are maintained. Figure D-III-7 illustrates the runstream process.

```
@QUAL UNCLASSIFIED
@ASG,A H7AFPFILES/XXX/XXX.
@ASG,T AFP1.
@ED,QI AFP1.

ADD H7AFPFILES.INVHMED/RAPD
@ASG,T AFP2.
@ED,QI AFP2.

ADD *H7RNGDST.RAPD-DEEP/H
@ASG,T OAFP1.
@XQT *H7AFP.870H-RANGE
@ED OAFP1.,*H7RNGDST.LABELED-RAPD/H
```

Figure D-III-6. Convert Combat Module Range Distribution to Labeled Format

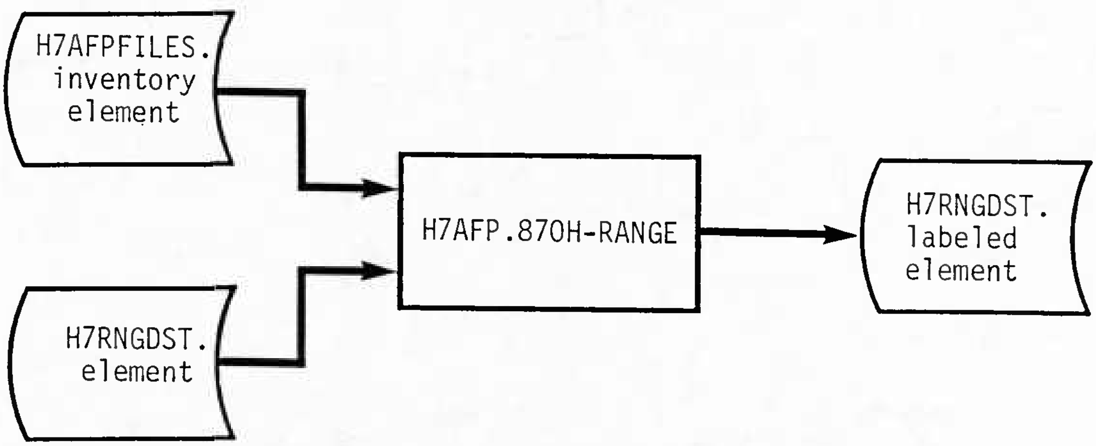
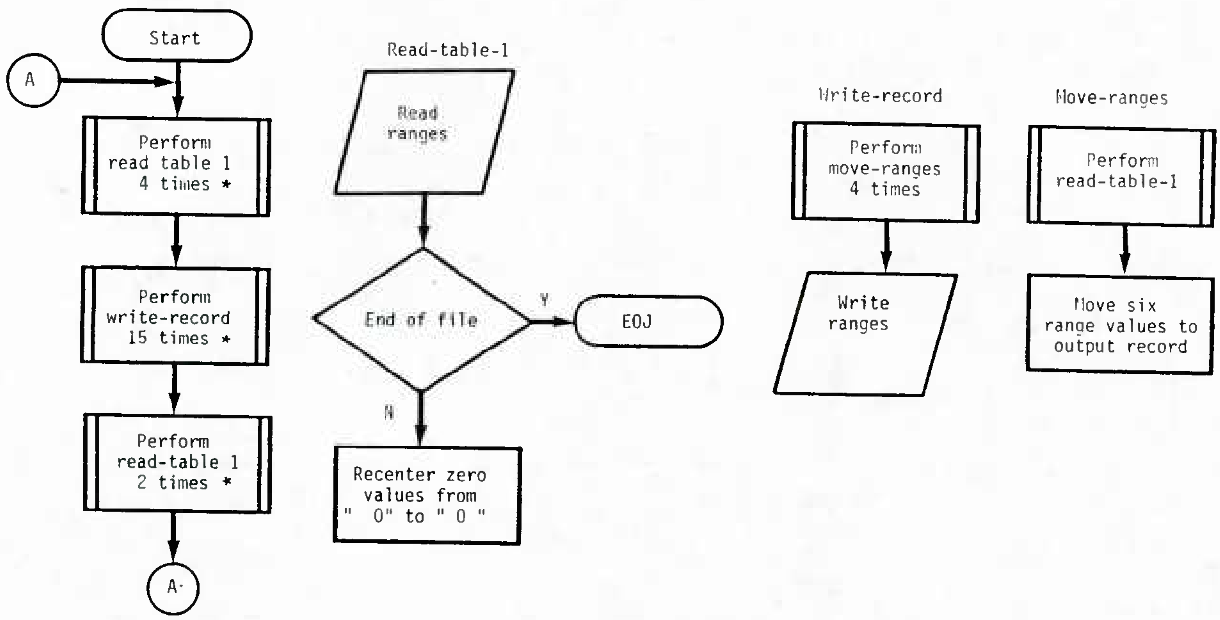


Figure D-III-7. Convert Combat Module Range Distribution to Labeled Format, I/O Flow

Section V. PROGRAM

D-III-10. The source program used to convert from labeled format to Combat Module format is H7AFP.M-RANGE-DIST, and the absolute program is H7AFP.870M-RANGE. This program reads the labeled range distribution file, strips the labels, reformats the ranges to four groups per line, and writes a file which can be used for the H7AFP.RNGDSTGEN preprocessor program. If a range distribution is zero (0), the character "0" is placed in what would normally be the 10s position. This is because only three positions are allowed per range value. If the value is "100," and if the preceding "0" was right-justified, the result would look like one value 0100. With a shifted zero, the result looks like "0 100." Figure D-III-8 contains a flow diagram of the basic logic for H7AFP.M-RANGE-DIST. Tables D-III-3 and D-III-4 show the file layouts labeled and Combat Module formats.



*These numbers are hardwired

Figure D-III-8. Flow Diagram for the Program to Convert Labeled RANGE Distribution File to Combat Module Format

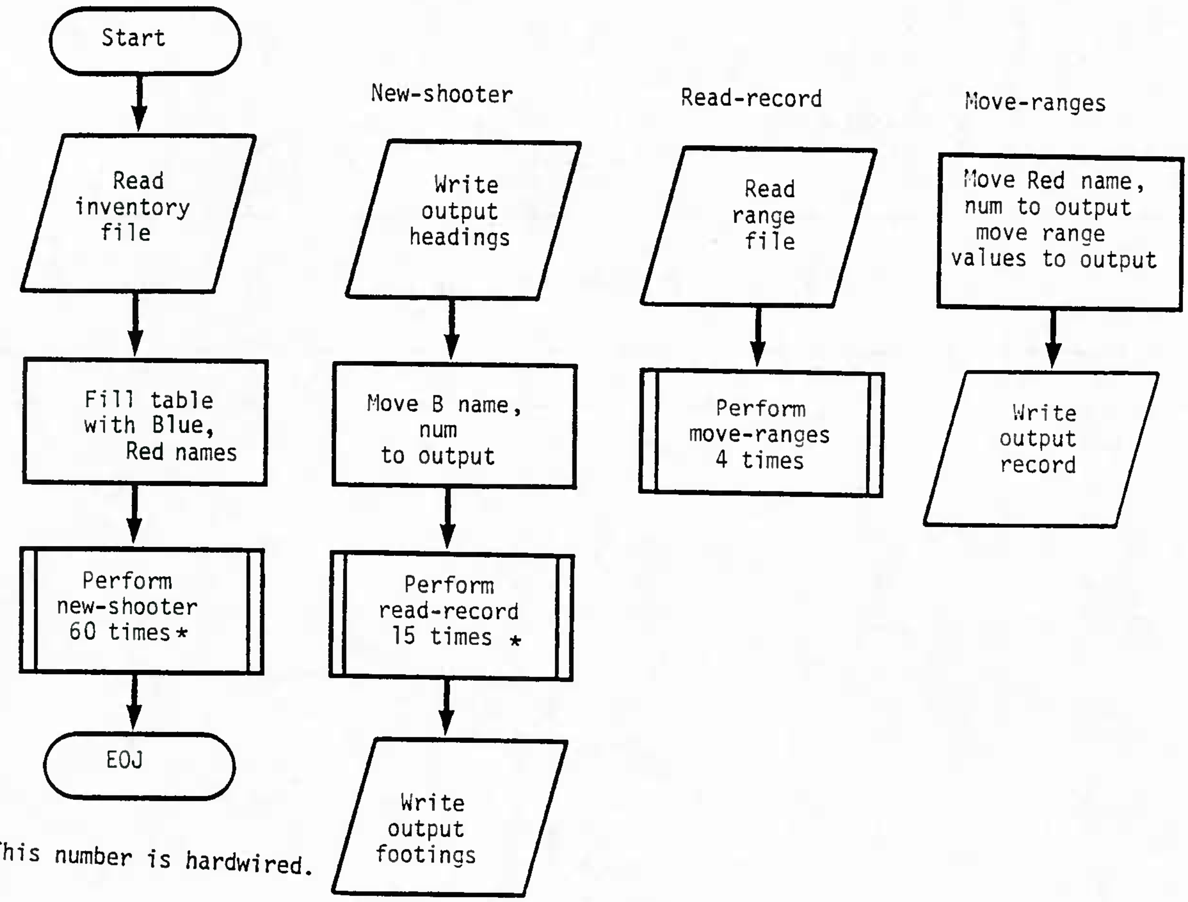
Table D-III-3. Labeled Range Distribution Format

Columns	Variable	Data description	Format
1-3	WS-SIDE1-NUM	AFP number of Side 1 type	3X
5-16	WS-SIDE1_NAME	Name of Side 1 type	12X
18-20	WS-SIDE2-NUM	AFP number of Side 2 type	3X
22-24	WS-SIDE2-NAME	Name of Side 2 type	12X
26-28	WS-RANGE1	Percent conflicts at 250 m	3X
33-35	WS-RANGE2	Percent conflicts at 500 m	3X
40-42	WS-RANGE3	Percent conflicts at 1,000 m	3X
47-49	WS-RANGE4	Percent conflicts at 1,500 m	3X
54-56	WS-RANGE5	Percent conflicts at 2,500 m	3X
61-63	WS-RANGE6	Percent conflicts deep range	3X

Table D-III-4. Combat Module Format Range Distribution

Column	Variable	Data description	Format
1-3	RANGE1 (1)	Percent conflicts at 250 m	3X
4-6	RANGE2(1)	Percent conflicts at 500 m	3X
7-9	RANGE3 (1)	Percent conflicts at 1,000 m	3X
10-12	RANGE4 (1)	Percent conflicts at 1,500 m	3X
13-15	RANGE5 (1)	Percent conflicts at 2,500 m	3X
16-18	RANGE6 (1)	Percent conflicts at deep range	3X
19-21	RANGE1 (2)	Percent conflicts at 250m	3X
22-24	RANGE2 (2)	Percent conflicts at 500 m	3X
25-27	RANGE3 (2)	Percent conflicts at 1,000 m	3X
28-30	RANGE4 (2)	Percent conflicts at 1,500 m	3X
31-33	RANGE5 (2)	Percent conflicts at 2,500 m	3X
34-36	RANGE6 (2)	Percent conflicts at deep range	3X
37-39	RANGE1 (3)	Percent conflicts at 250m	3X
40-42	RANGE2 (3)	Percent conflicts at 500 m	3X
43-45	RANGE3 (3)	Percent conflicts at 1,000 m	3X
46-48	RANGE4 (3)	Percent conflicts at 1,500 m	3X
49-51	RANGE5 (3)	Percent conflicts at 2,500 m	3X
52-54	RANGE6 (3)	Percent conflicts at deep range	3X
55-57	RANGE1 (4)	Percent conflicts at 250m	3X
58-60	RANGE2 (4)	Percent conflicts at 500 m	3X
61-63	RANGE3 (4)	Percent conflicts at 1,000 m	3X
64-67	RANGE4 (4)	Percent conflicts at 1,500 m	3X
68-70	RANGE5 (4)	Percent conflicts at 2,500 m	3X
71-73	RANGE6 (4)	Percent conflicts at deep range	3X

D-III-11. The source program to convert from Combat Module format to labeled is H7AFP.H-RNAGE-DIST, and the absolute program is H7AFP.87Ø-H-RANGE. This program reads the inventory element in H7AFPFILES to get the Side 1 and Side 2 weapon system names. The ranges are regrouped into 60 duels (Side 1 type 1 versus Side 2 types 1-60) for each of the Side 1 types. Figure D-III-9 shows the flow diagram for H7AFP.H-RANGE-DIST.



*This number is hardwired.

Figure D-III-9. Flow Diagram for the Program to Convert Combat Module Format Range Distribution to Labeled Format

ANNEX IV TO APPENDIX D

AFP TARGET CASUALTY AND CATEGORY FILE

Section I. OVERVIEW

D-IV-I. This annex describes the personnel casualty file (PCAS) which is used to compute the personnel losses incurred when a weapon type is killed. The data is in two sections. The first section lists the categories to which each Side 1 and Side 2 type belong. The second section lists the number of personnel credited as killed, per shooter target combination. Note that for super troop items these values have been multiplied by the super troop factor.

Section II. INPUT

D-IV-2. The labeled PCAS file first displays the total number of target types and the total number of shooter types for Side 1 and Side 2.

```
SIDE 1 TARGET CATEGORIES  4     SIDE 2 TARGET CATEGORIES  4
       SHOOTER CATEGORIES 13           SHOOTER CATEGORIES 13
```

Figure D-IV-1. Labeled PCAS, Target, and Shooter Types

a. The 60 Side 1 types are shown next with their corresponding AFP number, name, target type, and shooter type (Figure D-IV-2).

b. The 60 Side 2 types follow, with their AFP number, names, target types, and shooter types (Figure D-IV-3).

c. The rest of the file has the crew losses attributable to mobility/-firepower kills for every shooter-target combination. The first Side 1 type as a shooter versus each of the 60 Side 2 types as a target, with the resultant casualties, is shown in Figure D-IV-4.

d. Side 1, type 2 versus Side 2, types 1 through 60 casualties are next, through Side 1, type 60. Side 2 as a shooter follows.

SIDE 1 AFP #	SYSTEM	TARGET CATEGORY	SHOOTER CATEGORY
B01	BTRP DEEP	1	13
B02	M-16	1	1
B03	7.62GM	1	1
B04	MG50GM	1	1
B05	M203	1	1
B06	SAW	1	1
B07	M113	2	2
B08	90MMRR	1	3
B09	106MRR	2	3
B10	DUMMY	2	3
B11	DRAGON	1	3
B12	ITV-2	2	3
B13	VIPER	1	3
B14	LAW	1	3
B15	ITV-1	2	3
B16	IFV	2	4
B17	CFV	2	4
B18	1/4T TOW I	2	3
B19	M48A5	3	5
B20	M1E1	3	5
B21	M551	3	5
B22	M60A3	3	5
B23	M1	3	5
B24	M60A1	3	5
B25	M60A2	3	5
B26	DIVAD	2	6
B27	VULC1	2	6
B28	1/4T TOWII	2	7
B29	HMV W TOWI	2	3
B30	FAV TOW I	2	3
B31	STINGER	1	7
B32	CHAP1	2	7
B33	REDEYE	1	7
B34	FAV TOWII	2	3
B35	LAV25 TOWI	2	4
B36	AH-1S	4	8
B37	AH-64	4	8
B38	AH-1G	4	8
B39	M113 W M19	4	2
B40	HMV W M19	4	2
B41	A10	4	9
B42	LT VEH D	2	13
B43	HVY ARM D	3	13
B44	LT ARM D	2	13
B45	UH-60A	4	12
B46	CEV	3	12
B47	UH-1H	4	12
B48	OH-58	4	12
B49	UC3V	2	12
B50	M081M	2	10
B51	M081S	2	10
B52	M4.2M	2	10
B53	M4.2S	2	10
B54	M060	2	10
B55	H-155T	2	11
B56	H155S	2	11
B57	H203S	2	11
B58	MLRS	2	11
B59	H-105T	2	11
B60	H-203T	2	11

Figure D-IV-2. Labeled PCAS, Side 1 Categories

SIDE 2 AFP #	SYSTEM	TARGET CATEGORY	SHOOTER CATEGORY
R01	RTRP DEEP	1	13
R02	AKS-74	1	1
R03	7.60GM	1	1
R04	RSNIPE	1	1
R05	GREN30	1	1
R06	BRDM-2	2	2
R07	BTR-60	2	2
R08	BMP-R	2	2
R09	DUMMY	2	0
R10	DUMMY	2	0
R11	SPG-9	1	3
R12	RPG-16	1	3
R13	AT-5	2	3
R14	AT-7	1	3
R15	RPG 7	1	3
R16	BMP2M	2	4
R17	DUMMY	2	0
R18	DUMMY	2	0
R19	DUMMY	2	0
R20	DUMMY	2	0
R21	T55	3	5
R22	T62	3	5
R23	T64	3	5
R24	T72	3	5
R25	T80	3	5
R26	ZSU-23	2	6
R27	DUMMY	2	0
R28	DUMMY	2	0
R29	DUMMY	2	0
R30	SA-14	1	7
R31	SA-6	2	7
R32	SA-7	1	7
R33	SA-8	2	7
R34	SA-9	2	7
R35	SA-11	2	7
R36	HOPLITE	4	8
R37	HIP-E	4	8
R38	HIND-D	4	8
R39	DUMMY	4	0
R40	DUMMY	4	0
R41	MIG-21	4	9
R42	LT VEH D	2	13
R43	HVY ARM D	3	13
R44	LT ARM D	2	13
R45	DUMMY	4	0
R46	ACRV-2 C&C	2	12
R47	DUMMY	2	0
R48	DUMMY	2	0
R49	DUMMY	2	0
R50	DUMMY	2	0
R51	MO82M	2	10
R52	M120T	2	10
R53	DUMMY	2	0
R54	DUMMY	2	0
R55	DUMMY	2	0
R56	H122T/S	2	11
R57	H152T/S	2	11
R58	L122T+	2	11
R59	DUMMY	2	0
R60	DUMMY	2	0

Figure D-IV-3. Labeled PCAS, Side 2 Categories

SIDE 1 (SHOOTER) BTRP DEEP AFP # B01

SIDE 2 (TARGET) SYSTEM	AFP #	CREW LOSSES ATTRIBUTABLE TO MOBILITY/FIREPOWER KILLS
RTRP DEEP	R01	10.00
AKS-74	R02	10.00
7.60GM	R03	10.00
RSNIPE	R04	1.00
GREN30	R05	1.00
BRDM-2	R06	1.00
BTR-60	R07	4.00
BMP-R	R08	3.00
DUMMY	R09	0.00
DUMMY	R10	0.00
SPG-9	R11	1.00
RPG-16	R12	1.00
AT-5	R13	2.00
AT-7	R14	1.00
RPG 7	R15	1.00
BMP2M	R16	2.00
DUMMY	R17	1.00
DUMMY	R18	1.00
DUMMY	R19	1.00
DUMMY	R20	1.00
T55	R21	2.00
T62	R22	2.00
T64	R23	2.00
T72	R24	2.00
T80	R25	2.00
ZSU-23	R26	2.00
DUMMY	R27	1.00
DUMMY	R28	1.00
DUMMY	R29	1.00
SA-14	R30	1.00
SA-6	R31	2.00
SA-7	R32	1.00
SA-8	R33	2.00
SA-9	R34	2.00
SA-11	R35	2.00
HOPLITE	R36	2.00
HIP-E	R37	2.00
HIND-D	R38	2.00
DUMMY	R39	1.00
DUMMY	R40	1.00
MIG-21	R41	1.00
LT VEH D	R42	20.00
HVY ARM D	R43	2.00
LT ARM D	R44	30.00
DUMMY	R45	1.00
ACRV-2 C&C	R46	1.00
DUMMY	R47	1.00
DUMMY	R48	1.00
DUMMY	R49	1.00
DUMMY	R50	1.00
MO82M	R51	2.00
M120T	R52	2.00
DUMMY	R53	4.00
DUMMY	R54	1.00
DUMMY	R55	1.00
H122T/S	R56	1.00
H152T/S	R57	2.00
L122T+	R58	2.00
DUMMY	R59	1.00
DUMMY	R60	1.00

Figure D-IV-4. Labeled PCAS, Side 1, Type 1 Versus Side 2, Types 1-60

Section III. OUTPUT

D-IV-3. Figure D-IV-5 illustrates the first 72 lines of a Combat Module format PCAS element.

a. Lines 1 through 11 are related to category types of the 60 Side 1 and 60 Side 2 weapon types. Data on line 1 is not used: However, because this file is read with an unformatted read, there must be 5 digits on line 1. The second digit is the number of Side 1 target types. The third digit is the number of Side 2 target types. The fourth digit is the number of Side 1 shooter types. The fifth digit is the number of Side 2 shooter types.

b. Lines 2 and 3 have 60 numbers which are the target types of the 60 Side 1 items.

c. Lines 4 through 6 are the 60 shooter types of the Side 1 items.

d. Lines 7 and 8 are the 60 target types of the Side 2 items.

e. Lines 9 through 11 are the 60 shooter types of the Side 2 items.

f. Lines 12 through the end of the file have three columns of numbers, but only the last column is used. That column is the number of personnel credited as killed to two decimal places. Therefore, the 1000 in the third column on line 12 means when B01 kills one R01, 10.00 people are killed. The following 59 lines show B01 versus R02 through R60. Starting with line 72, B02 versus R01 through R60 personnel casualties are listed, then the rest of the Side 1 types through B60. Side 2 as a shooter follows: R01 killing B01 through B60 until the last line which is R60 killing B60.

```
1  4  4  13  13
1  1  1  1  1  1  2  1  2  2  1  2  1  1  2  2  2  2  3  3  3  3  3  3  3  2  2  2  2  2
1  2  1  2  2  4  4  4  4  4  4  2  3  2  4  3  4  4  2  2  2  2  2  2  2  2  2  2  2  2
13  1  1  1  1  1  2  3  3  3  3  3  3  3  3  4  4  3  5  5  5  5  5  5  5
6  6  3  3  3  7  7  7  3  4  8  8  8  2  2  9  13  13  13  12  12  12  12  12  10
10  10  10  10  11  11  11  11  11  11
1  1  1  1  1  2  2  2  2  2  1  1  2  1  1  2  2  2  2  2  3  3  3  3  3  2  2  2  2  1
2  1  2  2  2  4  4  4  4  4  4  2  3  2  4  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
13  1  1  1  1  2  2  2  0  0  3  3  3  3  3  4  0  0  0  0  5  5  5  5  5
6  0  0  0  7  7  7  7  7  7  8  8  8  0  0  9  13  13  13  0  12  0  0  0  0
10  10  0  0  0  11  11  11  0  0
           100          100          1000
           100          100          1000
           100          100          1000
           100          100           100
           100          100           100
           100          100           100
           300          300           400
           200          200           300
           000          000           000
           000          000           000
           100          100           100
           100          100           100
           200          200           200
           100          100           100
           100          100           100
           200          200           200
           100          100           100
           100          100           100
           100          100           100
           100          100           100
           100          100           200
           100          100           200
           100          100           200
           100          100           200
           100          100           200
           200          200           200
           100          100           100
           100          100           100
           100          100           100
           100          100           100
           200          200           200
           100          100           100
           200          200           200
           200          200           200
           200          200           200
           200          200           200
           200          200           200
           200          200           200
           100          100           100
           100          100           100
           100          100           100
           100          100          2000
           100          100           200
           100          100          3000
           100          100           100
           100          100           100
           100          100           100
           100          100           100
           100          100           100
           100          100           100
           200          200           200
           200          200           200
           200          200           400
           100          100           100
           100          100           100
           100          100           100
           200          200           200
           200          200           200
           100          100           100
           100          100           100
```

Figure D-IV-5. Combat Module Format PCAS

Section IV. RUNSTREAM

D-IV-4. There is a program to convert labeled PCAS data to Combat Module format, and another program to convert Combat Module format to labeled PCAS format. Because of the repetitive structure, it is generally easier to change the personnel losses in the Combat Module format using the system EDITOR. Unique type-on-type losses which involve only one or a few engagements may be easier to change in the labeled format.

a. The runstream to convert labeled to Combat Module format is in Figure D-IV-6.

```
@ASG,T AFP1.
@ASG,T OAFP1.
@ED,IQ AFP1.

ADD *H7PCAS.LABELED/H
EXI
@XQT *H7AFP.87CM-PCAS
@ED OAFP1.,*H7PCAS.H
```

Figure D-IV-6. PCAS Runstream to Convert Labeled to Combat Module Format

b. The runstream to convert Combat Module format to labeled is in Figure D-IV-7.

```
@ASG,A H7AFPFILES/XXX/XXX.
@ASG,T AFP1.
@ASG,T AFP2.
@ASG,T OAFP1.
@ED,IQ AFP1.

ADD H7AFPFILES.INVHMEC/PAPO
EXI
@ED,IQ AFP2.

ADD *H7PCAS.H
EXI
@XQT *H7AFP.870H-PCAS
@ED OAFP1.,*H7PCAS.LABELED/H
```

Figure D-IV-7. PCAS Runstream to Convert Combat Module Format to Labeled

Section V. PROGRAMS

D-IV-5. The program to convert labeled PCAS data to Combat Module format is H7AFP.M-PCAS. The absolute program is H7AFP.870M-PCAS. Although the labeled format has only the mobility firepower kills, the Combat Module format must have mobility kills and firepower kills as well as mobility/-firepower kills. The program which processes the PCAS data does not use the mobility or firepower information; however, because of the unformatted read in the program, there must be two data items before the mobility/fire-power kill is read. Therefore, the mobility/firepower kill is moved three times to the Combat Module format, giving mobility, firepower, and mobility/firepower the same value. The I/O flow for H7AFP.M-PCAS and its flowchart are in Figures D-IV-8 and D-IV-9.

D-IV-6. The program to convert Combat Module format PCAS to labeled format is H7AFP.M-PCAS. The absolute program is H7AFP.870M-PCAS. The mobility kills and the firepower kills (the first and second column starting on line 12) are dropped because that information is not used. The labeled format shows only mobility/firepower kills. The I/O flow for H7AFP.H-PCAS and its flowchart are Figures D-IV-10 and D-IV-11.

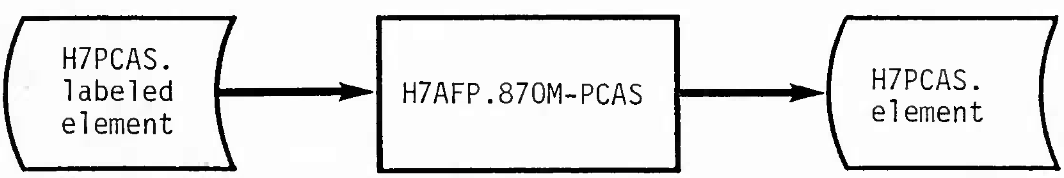
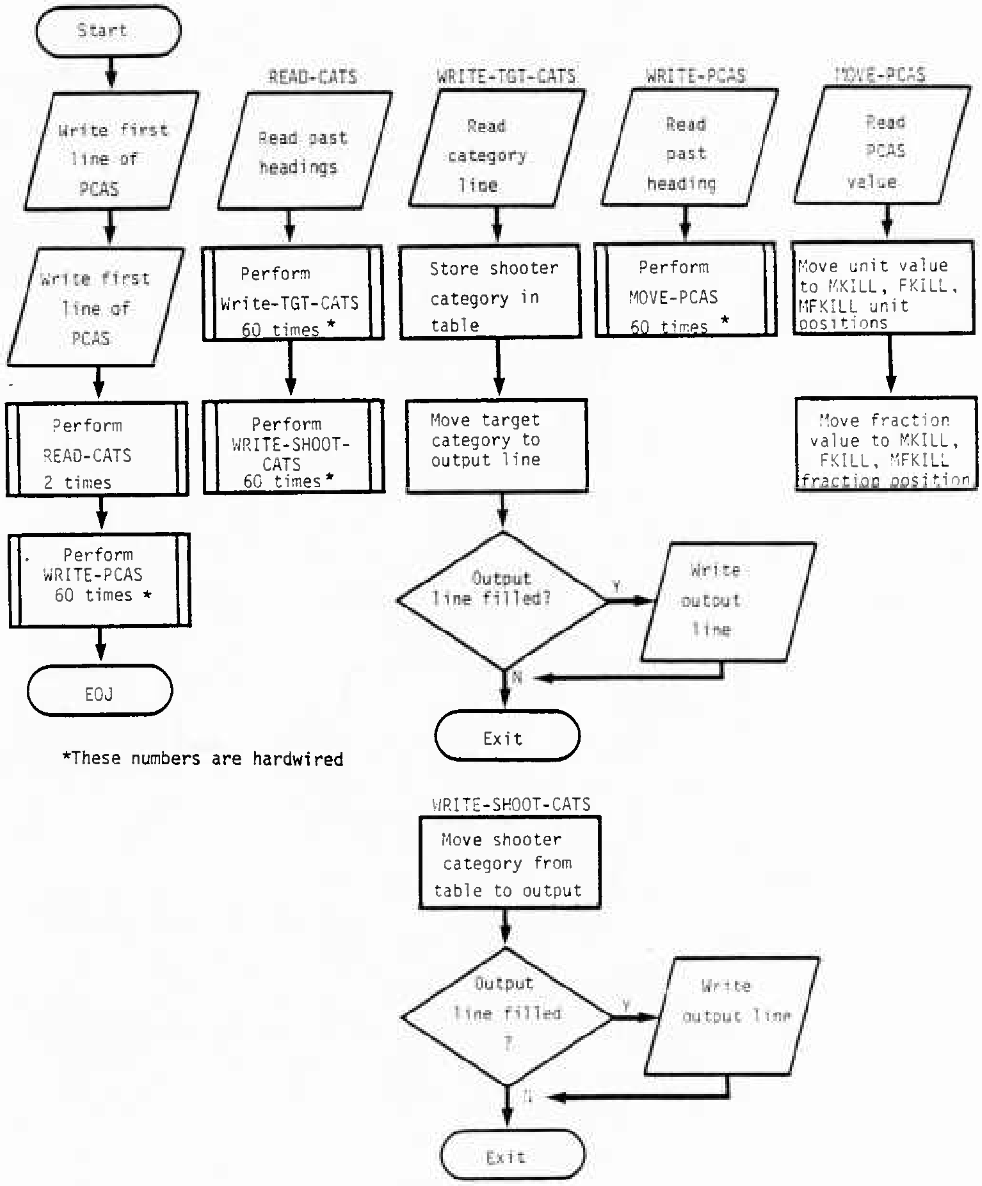


Figure D-IV-8. H7AFP.M-PCAS I/O Flow



*These numbers are hardwired

Figure D-IV-9. H7AFP.M-PCAS Flowchart

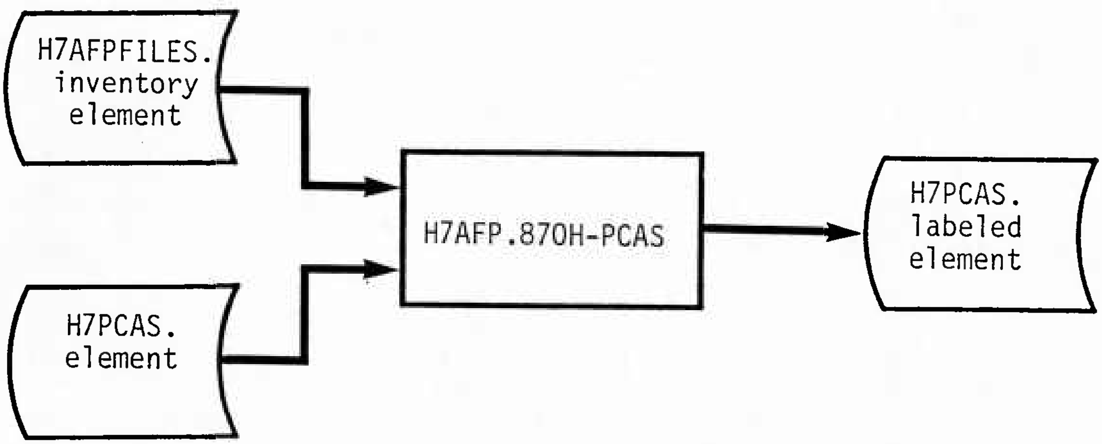


Figure D-IV-10. H7AFP.H-PCAS I/O Flow

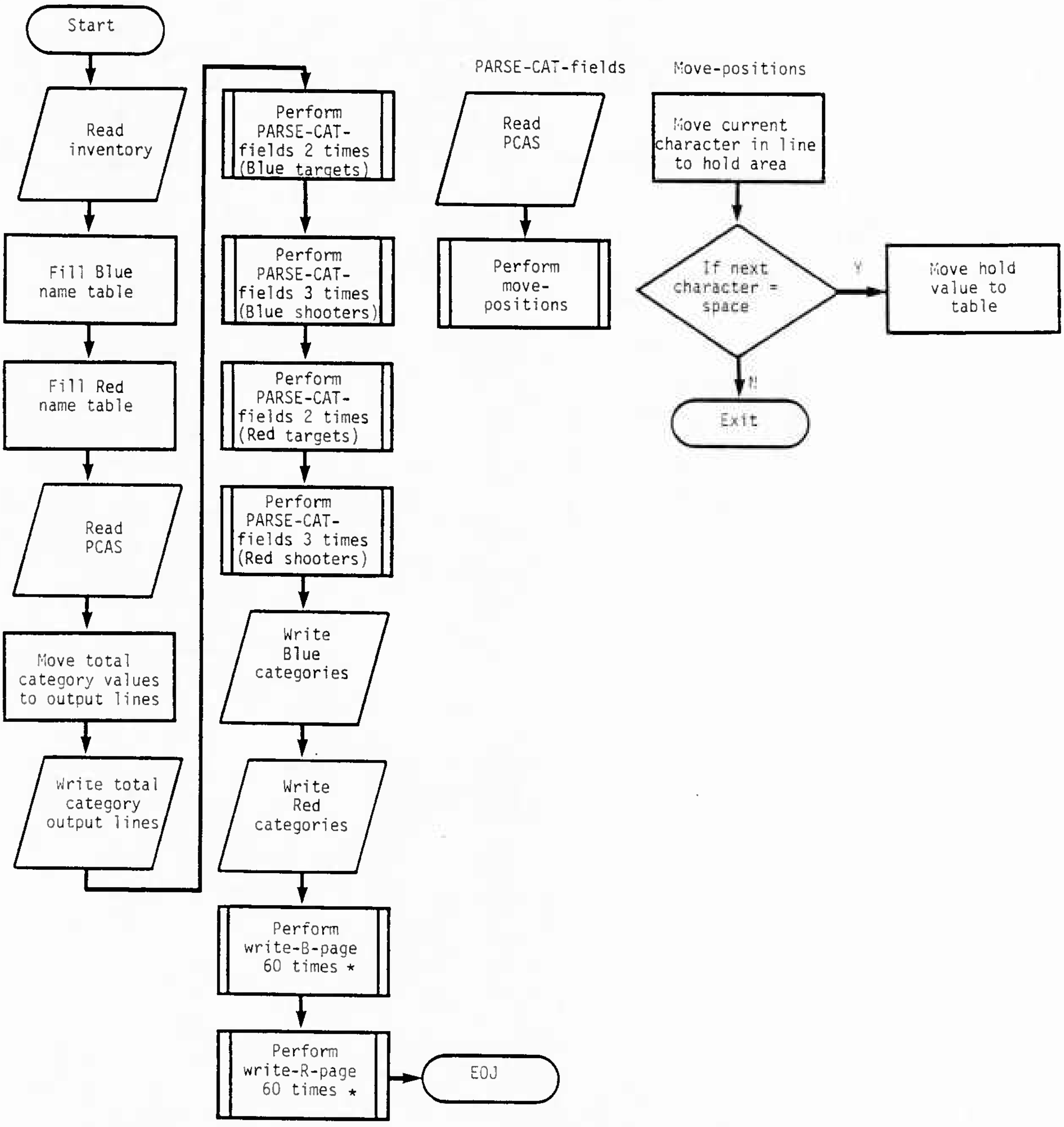


Figure D-IV-11. H7AFP.H-PCAS Flowchart
(page 1 of 2 pages)

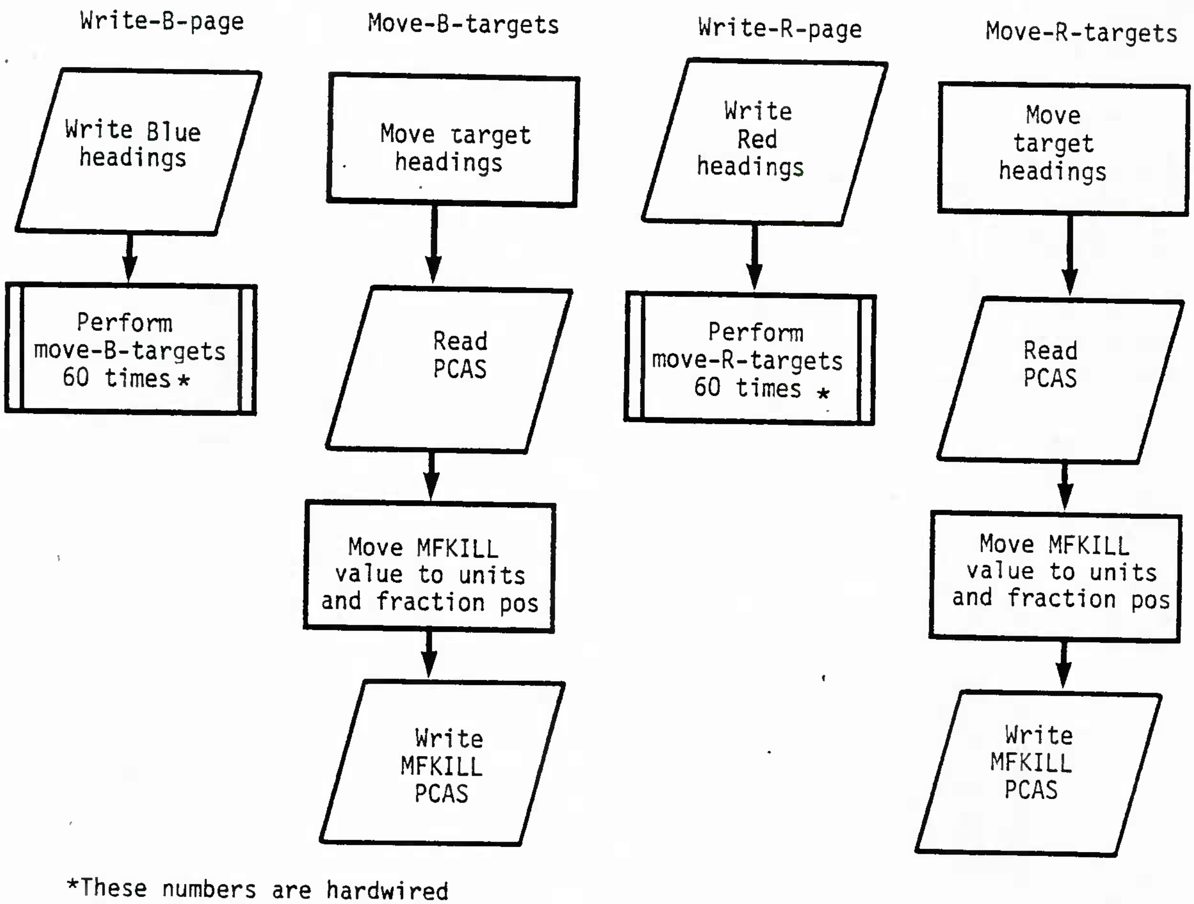


Figure D-IV-11. H7AFP.H-PCAS Flowchart (page 2 of 2 pages)

ANNEX V TO APPENDIX D

AFP ENGAGEMENT CHARACTERISTICS FILE

Section I. OVERVIEW

D-V-1. This annex describes the engagement input to the Combat Module. The engagement file contains the participation, duration of engagement, number of sites (default value of 1.00, same as ADDITIONALS), and maximum number of conflicts for each of the 60 Side 1 versus 60 Side 2 conflicts. There is a program to convert a labeled version of the engagement data to Combat Module format and a program to convert the module format to labeled. Because of the repetitive nature of the file, changes can easily be made to the Combat Module format using the system EDITOR. For specific type on type changes, it may be clearer to change the labeled version. The option of converting from one format to the other is available to be used as required. Generally, only one element in the engagement file is needed for an AFP run, i.e., no changes by posture or environment.

Section II. INPUT

D-V-2. The engagement file consists of five columns of numbers:

Column 1 - fraction of Side 1 in each engagement.

Column 2 - fraction of Side 2 in each engagement.

Column 3 - duration of engagement in minutes.

Column 4 - number of sites.

Column 5 - maximum number of conflicts.

Table D-V-1 depicts the organization of the file. Figure D-V-1 shows sample lines of a labeled engagement file. Figure D-V-2 shows sample lines of a Combat Module engagement file.

Table D-V-1. Engagement File Organization

Side 1	Type 1	vs	Side 2	Type 1
		vs	Side 2	Type 2
		:		
		:		
		:		
		vs	Side 2	Type 60
Side 1	Type 2	vs	Side 2	Type 1
		vs	Side 2	Type 2
		:		
		:		
		:		
		vs	Side 2	Type 60
Side 1	Type 60	vs	Side 2	Type 1
		vs	Side 2	Type 2
		:		
		:		
		:		
		vs	Side 2	Type 60

SIDE 1 SYSTEM: BTRP DEEP AFP # 801

SIDE2 SYSTEM	AFP #	SIDE1 % PARTICIPATION	SIDE2 % PARTICIPATION	DURATION (IN MIN)	# OF SITES	MAX # CONFLICT
RTRP DEEP	R01	100	100	2.01	1.00	4.00
AKS-74	R02	100	100	2.01	1.00	4.00
7.62GM	R03	100	100	2.01	1.00	4.00
RSNIPE	R04	100	100	2.01	1.00	4.00
GREN30	R05	100	100	2.01	1.00	4.00
BRDM-2	R06	100	100	2.01	1.00	4.00
BTR-60	R07	100	100	2.01	1.00	4.00
BMP-R	R08	100	100	2.01	1.00	4.00
DUMMY	R09	100	100	2.01	1.00	4.00
DUMMY	R10	100	100	2.01	1.00	4.00
SPG-9	R11	100	100	2.01	1.00	4.00
RPG-16	R12	100	100	2.01	1.00	4.00
AT-5	R13	100	100	2.01	1.00	4.00
AT-7	R14	100	100	2.01	1.00	4.00
RPG 7	R15	100	100	2.01	1.00	4.00
BMP2M	R16	100	100	2.01	1.00	4.00
DUMMY	R17	100	100	2.01	1.00	4.00
DUMMY	R18	100	100	2.01	1.00	4.00
DUMMY	R19	100	100	2.01	1.00	4.00
DUMMY	R20	100	100	2.01	1.00	4.00
T55	R21	100	100	2.01	1.00	4.00
T62	R22	100	100	2.01	1.00	4.00
T64	R23	100	100	2.01	1.00	4.00
T72	R24	100	100	2.01	1.00	4.00
T80	R25	100	100	2.01	1.00	4.00
ZSU-23	R26	100	100	2.01	1.00	4.00
DUMMY	R27	100	100	2.01	1.00	4.00
DUMMY	R28	100	100	2.01	1.00	4.00
DUMMY	R29	100	100	2.01	1.00	4.00
SA-14	R30	100	100	2.01	1.00	4.00
SA-6	R31	100	100	2.01	1.00	4.00
SA-7	R32	100	100	2.01	1.00	4.00
SA-8	R33	100	100	2.01	1.00	4.00
SA-9	R34	100	100	2.01	1.00	4.00
SA-11	R35	100	100	2.01	1.00	4.00
HOPLITE	R36	100	100	2.01	1.00	4.00
HIP-E	R37	100	100	2.01	1.00	4.00
HIND-D	R38	100	100	2.01	1.00	4.00
DUMMY	R39	100	100	2.01	1.00	4.00
DUMMY	R40	100	100	2.01	1.00	4.00
MIG-21	R41	100	100	2.01	1.00	4.00
LT VEH D	R42	100	100	2.01	1.00	4.00
HVY ARM D	R43	100	100	2.01	1.00	4.00
LT ARM D	R44	100	100	2.01	1.00	4.00
DUMMY	R45	100	100	2.01	1.00	4.00
ACRV-2 C&C	R46	100	100	2.01	1.00	4.00
DUMMY	R47	100	100	2.01	1.00	4.00
DUMMY	R48	100	100	2.01	1.00	4.00
DUMMY	R49	100	100	2.01	1.00	4.00
DUMMY	R50	100	100	2.01	1.00	4.00
MO82M	R51	100	100	2.01	1.00	4.00
M120T	R52	100	100	2.01	1.00	4.00
DUMMY	R53	100	100	2.01	1.00	4.00
DUMMY	R54	100	100	2.01	1.00	4.00
DUMMY	R55	100	100	2.01	1.00	4.00
H122T/S	R56	100	100	2.01	1.00	4.00
H152T/S	R57	100	100	2.01	1.00	4.00
L122T+	R58	100	100	2.01	1.00	4.00
DUMMY	R59	100	100	2.01	1.00	4.00
DUMMY	R60	100	100	2.01	1.00	4.00

Figure D-V-1. Labeled Engagement File

100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400
100	100	201	100	400

Figure D-V-2. Combat Module Format Engagement File

Section III. OUTPUT

D-V-3. Both the labeled and the Combat Module formats were described in Section II.

Section IV. RUNSTREAMS

D-V-4. The I/O flow of the process to convert labeled to Combat Module format is shown in Figure D-V-3. The runstream is shown in Figure D-V-4.

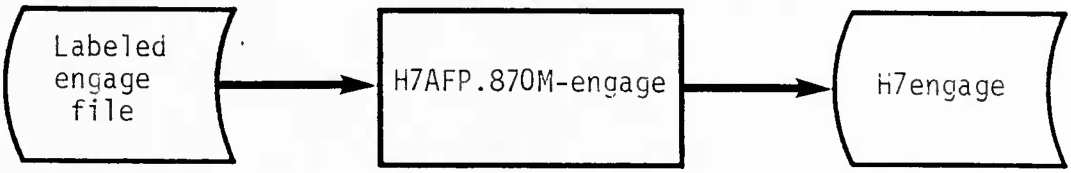


Figure D-V-3. Labeled to Combat Module Format I/O Flow

```
@ASG,T AFP1.
@ASG,T OAFP1.
@ED,IQ AFP1.

ADD *H7ENGAGE.LABELED/H
EXI
@XQT *H7AFP.870M-ENGAGE
@ED OAFP1.,*H7ENGAGE.H
```

Figure D-V-4. Labeled to Combat Module Format Runstream

D-V-5. The I/O flow of the process to convert module format to labeled is shown in Figure D-V-5. Weapon system names from the H7AFPFILES.INVENTORY element are used to label the H7ENGAGE element. The runstream is shown in Figure D-V-6.

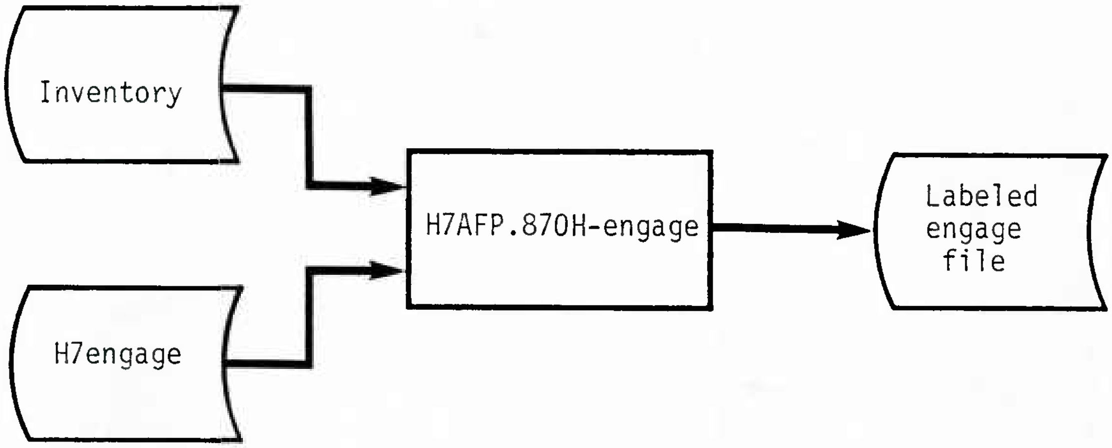


Figure D-V-5. Combat Module Format To Labeled I/O Flow

```
@ASG,A H7AFPFILES/XXX/XXX.
@ASG,T AFP1.
@ASG,T AFP2.
@ED,GI AFP1.

ADD H7AFPFILES.INVHM80/RAPD
@ED,GI AFP2.

ADD *H7ENGAGE.H
@ASG,T OAFP1.
@XQT *H7AFP.870H-ENGAGE
@ED OAFP1.,*H7ENGAGE.LABELED/H
```

Figure D-V-6. Combat Module Format to Labeled Runstream

Section V. PROGRAMS

D-V-6. The program which converts the labeled format to Combat Module format is H7AFP.M-ENGAGE. The absolute program is H7AFP.870M-ENGAGE. The program I/O flow is the same as Figure D-V-3. The program flowchart is in Figure D-V-7.

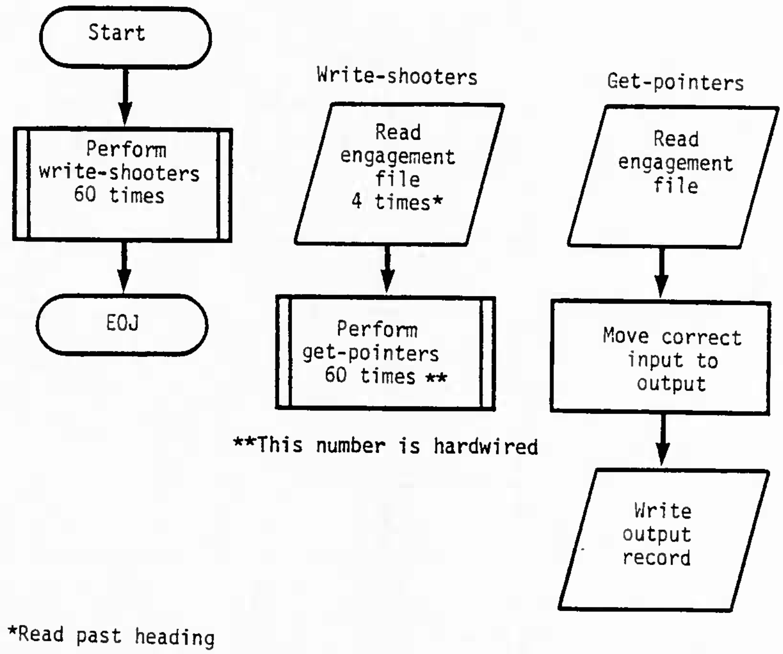


Figure D-V-7. H7AFP.M-ENGAGE Flow Chart

D-V-7. The program which converts the Combat Module format to labeled format is H7AFP.H-ENGAGE. The absolute program is H7AFP.870H-ENGAGE. The program reads weapon system names from the inventory file and labels the engagement data. The program I/O flow is the same as Figure D-V-5. The program flowchart is in Figure D-V-8.

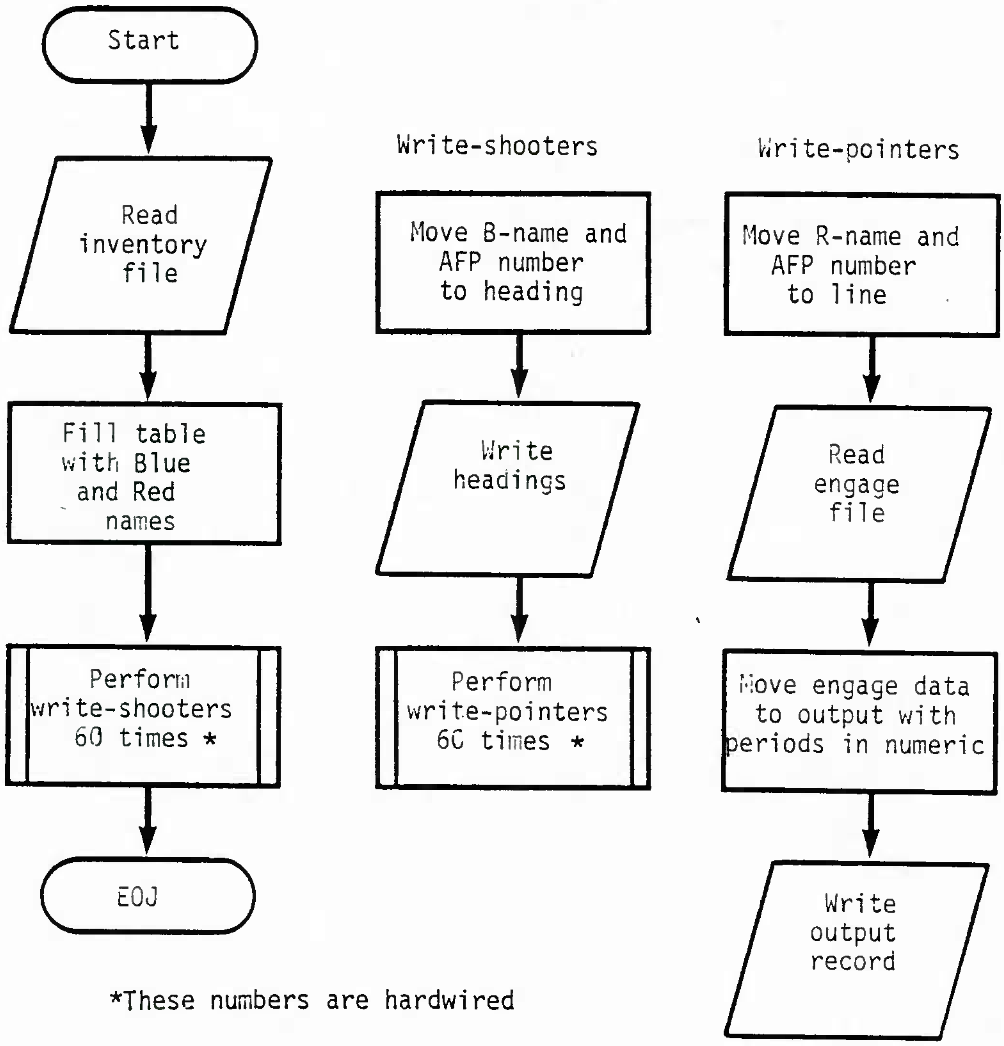


Figure D-V-8. H7AFP.H-ENGAGE Flow Chart

ANNEX VI TO APPENDIX D

AFP SSPK FILE

Section I. OVERVIEW

D-VI-1. This annex describes the process by which the SSPK values from AMMSA/BRL are modified for the AFP Combat Module. There are two pointer files, one for Side 1 (Blue) weapon types and one for Side 2 (Red) weapon types. These pointer files have shooter/target matrices which, for every shooter/target combination have: a pointer to the open SSPK value, a pointer to the defilade SSPK value, a moving target degradation pointer, a moving firer degradation pointer, and a night degradation factor. The pointers "point" to lines in other files which contain the actual values to be used. Lines in H7AFPPK contain the SSPK values for stationary firers and stationary targets in the open and in defilade. H7MOVETGT has degradation factors which are multiplied against the SSPK values if the environmental conditions call for a moving target. H7MOVEFIRER has degradation factors which are multiplied against the SSPK values if environmental conditions call for a moving firer. After all degradation factors have been taken into account, a modified SSPK for each shooter/target combination is generated. Figure D-VI-1 illustrates the I/O flow of the SSPK generation process. The program H7AFP.M-SSPK reads the above mentioned files and generates a Combat Module format modified SSPK file and a labeled format file.

Section II. INPUT

D-VI-2. There are two labeled SSPK pointer elements, H7AFPFILES.BPOINT and H7AFPFILES.RPOINT. The BPOINT element shows Blue weapon types as shooters versus the Red weapon type targets. RPOINT shows Red as the shooters. The logical organization (Table D-VI-1) is the same for both BPOINT and RPOINT. An example page from BPOINT is in Figure D-VI-2. The following information is in the pointer elements:

a. The shooter name and AFP number.

b. The weapon used by that shooter against each target.

c. The ammunition used by the weapon.

d. The target name and AFP number.

e. Pointer to the SSPK for conflicts in the open.

f. Pointer to the moving target degradation factor.

g. Pointer to the SSPK for conflicts with the target in defilade.

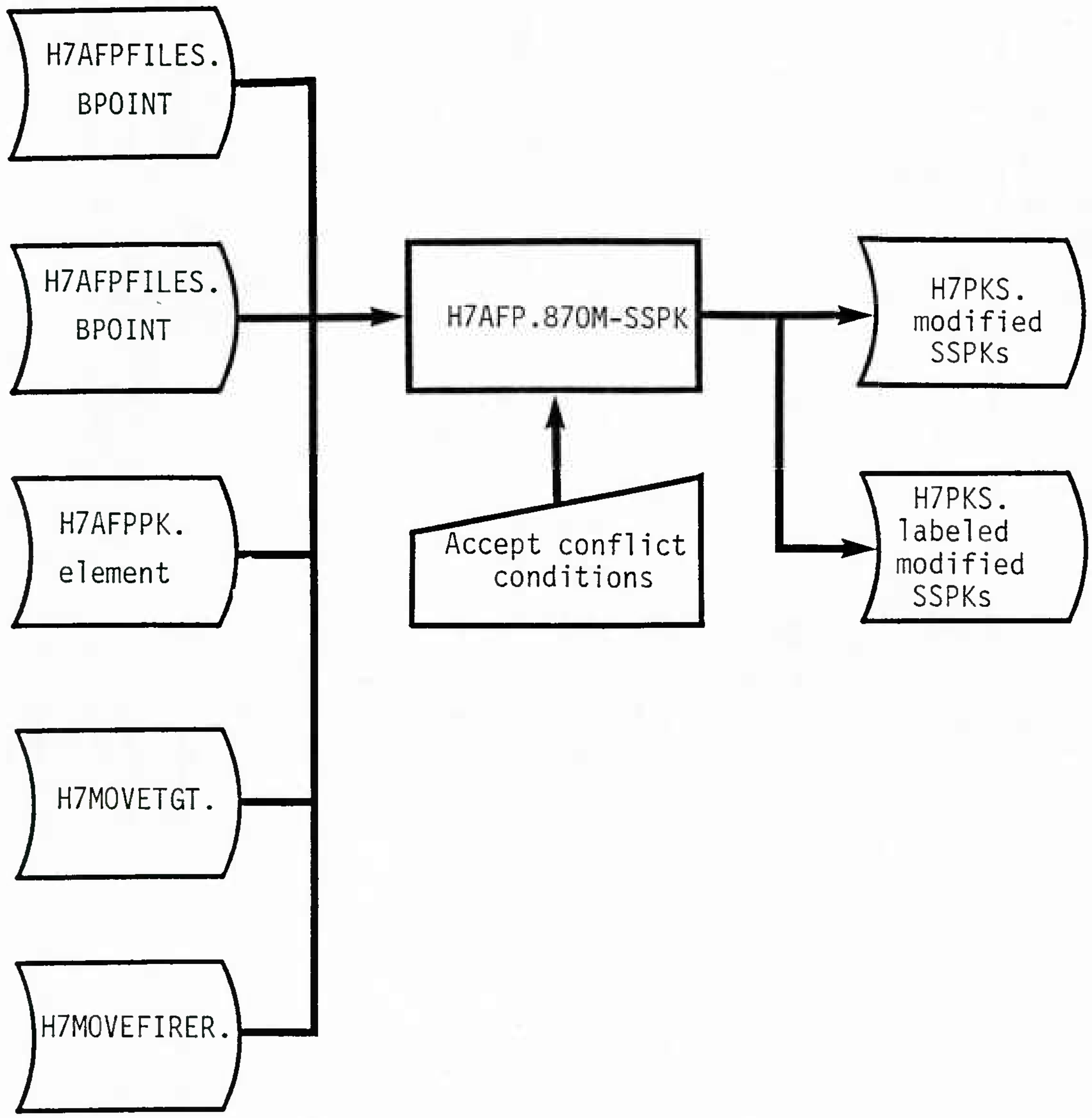


Figure D-VI-1. SSPK I/O Flow

Table D-VI-1. Logical Organization of SSPK Pointer Elements

B01 vs	R01
	R02
	:
	:
	:
	R60
B02 vs	R01
	R02
	:
	:
	:
	R60
	:
	:
	:
B60 vs	R60

SHOOTER SYSTEM: DRAGON SHOOTER #B11

SHOOTER WEAPON	TYPE AMMO	TARGET	TGT #	OPEN TGT	MOVING TGT	DEF TGT	MOVING FIRER	NIGHT DEGRAD
		RTRP	R01	627	060	627	060	.67
		AKS-74	R02	627	060	627	060	.67
		7.60GM	R03	627	060	627	060	.67
		RSNIPE	R04	627	060	627	060	.67
		GREN30	R05	627	060	627	060	.67
DRAGON	HEAT	BRDM-2	R06	151	016	152	016	.80
DRAGON	HEAT	BTR-60	R07	151	016	152	016	.80
DRAGON	HEAT	BMP-R	R08	153	016	154	016	.80
		DUMMY	R09	627	060	627	060	.67
		DUMMY	R10	627	060	627	060	.67
		SPG-9	R11	627	060	627	060	.67
		RPG-16	R12	627	060	627	060	.67
DRAGON	HEAT	AT-5	R13	163	016	164	016	.80
DRAGON	HEAT	AT-7	R14	627	060	627	060	.67
		RPG 7	R15	627	060	627	060	.67
DRAGON	HEAT	BMP2M	R16	153	016	154	016	.80
		DUMMY	R17	627	060	627	060	.67
		DUMMY	R18	627	060	627	060	.67
		DUMMY	R19	627	060	627	060	.67
		DUMMY	R20	627	060	627	060	.67
DRAGON	HEAT	T55	R21	155	016	156	016	.80
DRAGON	HEAT	T62	R22	155	016	156	016	.80
DRAGON	HEAT	T64	R23	157	016	158	016	.80
DRAGON	HEAT	T72	R24	159	016	160	016	.80
DRAGON	HEAT	T80	R25	161	016	162	016	.80
DRAGON	HEAT	ZSU-23	R26	165	016	166	016	.80
		DUMMY	R27	627	060	627	060	.67
		DUMMY	R28	627	060	627	060	.67
		DUMMY	R29	627	060	627	060	.67
		SA-14	R30	627	060	627	060	.67
		SA-6	R31	627	060	627	060	.67
		SA-7	R32	627	060	627	060	.67
		SA-8	R33	627	060	627	060	.67
		SA-9	R34	627	060	627	060	.67
		SA-11	R35	627	060	627	060	.67
		HOPLITE	R36	627	060	627	060	.67
		HIP-E	R37	627	060	627	060	.67
		HIND-D	R38	627	060	627	060	.67
		DUMMY	R39	627	060	627	060	.67
		DUMMY	R40	627	060	627	060	.67
		MIG-21	R41	627	060	627	060	.67
		DUMMY	R42	627	060	627	060	.67
		DUMMY	R43	627	060	627	060	.67
		DUMMY	R44	627	060	627	060	.67
		DUMMY	R45	627	060	627	060	.67
DRAGON	HEAT	ACRV-2 C&C	R46	151	016	152	016	.80
		DUMMY	R47	627	060	627	060	.67
		DUMMY	R48	627	060	627	060	.67
		DUMMY	R49	627	060	627	060	.67
		DUMMY	R50	627	060	627	060	.67
		M082M	R51	627	060	627	060	.67
		M120T	R52	627	060	627	060	.67
		DUMMY	R53	627	060	627	060	.67
		DUMMY	R54	627	060	627	060	.67
		DUMMY	R55	627	060	627	060	.67
		H122T/S	R56	627	060	627	060	.67
		H152T/S	R57	627	060	627	060	.67
		L122T+	R58	627	060	627	060	.67
		DUMMY	R59	627	060	627	060	.67
		DUMMY	R60	627	060	627	060	.67

Figure D-VI-2. H7AFPFILE.BPOINT Example Page

h. Pointer to the moving shooter degradation factor.

i. Night degradation factor.

D-VI-3. The actual SSPK values are in SECRET*H7AFPPK. There are no labels in the file, only six columns of numbers:

column - SSPK at 250 m
column - SSPK at 500 m
column - SSPK at 1,000 m
column - SSPK at 1,500 m
column - SSPK at 2,500 m
column - SSPK at greater than 2,500 m

Each line in the file is "pointed to" by the pointers in BPOINT and RPOINT. For example, the value 627 in the Open Tgt column of BPOINT means that the six numbers on line 627 of H7AFPPK are the SSPK values for that Blue and Red conflict combination in the open. A value of Ø96 in the Def Tgt column means line 96 has the SSPK values for that Blue and Red conflict combination with the target in defilade. There is no sequential order to this file. New SSPK values are added to the end of the file as they are needed. However, because of the pointer structure, values are not moved within, deleted from, or added to the middle of the file. The file may grow as large as 2,000 lines. At that point, M7AFP.M-SSPK, the program which reads H7AFPPK must be changed because it only allows for a 2,000 line maximum in its array. Figure D-VI-3 is an example of what H7AFPPK might look like. Because the file is SECRET, the values shown in Figure D-VI-3 have been adjusted and are not true SSPKs.

.05	.01	.00	.00	.00	.00
.04	.02	.00	.00	.00	.00
.03	.01	.00	.00	.00	.00
.45	.18	.00	.00	.00	.00
.34	.14	.00	.00	.00	.00
.22	.09	.00	.00	.00	.00
.17	.07	.00	.00	.00	.00
.95	.95	.95	.95	.95	.00
.90	.90	.90	.90	.90	.00
.62	.62	.62	.76	.76	.00
.67	.67	.67	.82	.82	.00
.85	.85	.85	.89	.89	.00
.46	.47	.67	.67	.67	.00
.91	.91	.91	.91	.91	.00
.88	.88	.88	.88	.88	.00
.93	.93	.93	.93	.93	.00
.60	.61	.63	.64	.67	.00
.64	.65	.67	.63	.72	.00
.81	.81	.81	.81	.81	.00
.65	.65	.65	.65	.65	.00
.90	.90	.90	.90	.90	.00
.88	.88	.89	.91	.94	.00
.85	.86	.87	.89	.91	.00
.91	.91	.91	.91	.91	.00
.28	.06	.02	.00	.00	.00

Figure D-VI-3. H7AFPPK Example

D-VI-4. Moving target degradation factors are in H7MOVETGT. The file is shown in Figure D-VI-4. There are six columns of numbers in this file:

1st column - degradation factor for 250 m SSPK
2d column - degradation factor for 500 m SSPK
3d column - degradation factor for 1,000 m SSPK
4th column - degradation factor for 1,500 m SSPK
5th column - degradation factor for 2,500 m SSPK
6th column - degradation factor for greater than 2,500 m SSPK

The number in the Moving Tgt column of BPOINT and RPOINT points to a line in H7MOVETGT which has the degradation factors. These factors are multiplied times the corresponding SSPK values if conditions of the conflict call for a target to be on the move. This file, like H7AFPPK, has no sequential order. As new degrading factors are needed, they are added to the end of the file. Additions to or deletions from the middle of the file are not allowed. The file may grow as large as 100 lines. If more lines are needed, change the array size in the program H7AFP.M-SSPK which reads this file.

D-VI-5. Moving firer degradation factors are in H7MOVEFIRER. The file is shown in Figure D-VI-5. This file is exactly like H7MOVETGT except the factors are for moving shooter conditions and use the Moving Firer column in BPOINT and RPOINT. It also has a maximum size of 100 lines. The array in program H7AFP.M-SSPK must be expanded if more lines are needed.

.50	.00	.00	.00	.00	.00
.50	.00	.00	.00	.00	.00
1.00	.80	.00	.00	.00	.00
1.00	.80	.50	.25	.10	.00
0.50	0.00	0.00	0.00	0.00	0.00
1.00	0.80	0.80	0.80	0.80	0.00
0.50	0.50	0.00	0.00	0.00	0.00
0.80	0.80	0.80	0.80	0.80	0.80
1.00	1.00	1.00	1.00	1.00	1.00
1.00	0.80	0.00	0.00	0.00	0.00
1.00	1.00	0.96	0.85	0.75	0.65
1.00	1.00	0.96	0.85	0.75	0.65
1.00	1.00	0.90	0.63	0.52	0.41
1.00	1.00	0.89	0.61	0.46	0.32
1.00	1.00	0.92	0.57	0.48	0.39
0.85	0.85	0.00	0.00	0.00	0.00
0.13	0.00	0.00	0.00	0.00	0.00
0.80	0.80	0.80	0.80	0.80	0.80
0.80	0.80	0.80	0.80	0.80	0.80
0.80	0.80	0.80	0.80	0.80	0.80
1.00	1.00	1.00	1.00	1.00	1.00
0.80	0.80	0.80	0.80	0.80	0.80
0.00	1.00	1.00	1.00	0.00	0.00
0.00	1.00	1.00	1.00	0.00	0.00
0.00	1.00	1.00	1.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00
0.80	0.80	0.80	0.80	0.80	0.80
0.80	0.80	0.80	0.80	0.80	0.80
0.50	0.50	0.00	0.00	0.00	0.00
0.50	0.50	0.00	0.00	0.00	0.00
0.50	0.50	0.50	0.00	0.00	0.00
0.50	0.50	0.50	0.50	0.50	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.00	1.00	1.00	0.00	0.00	0.00
0.50	0.50	0.50	0.50	0.50	0.50
0.97	0.67	0.16	0.11	0.08	0.00
0.50	0.50	0.50	0.00	0.00	0.00
1.00	1.00	0.78	0.67	0.56	0.45
1.00	1.00	0.88	0.57	0.48	0.39
1.00	1.00	0.95	0.77	0.66	0.56
1.00	1.00	0.95	0.77	0.66	0.56
1.00	1.00	0.95	0.77	0.66	0.56
0.50	0.50	0.50	0.00	0.00	0.00
0.50	0.50	0.50	0.00	0.00	0.00
0.33	0.08	0.08	0.00	0.00	0.00
0.33	0.08	0.08	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.00	1.00	1.00	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	0.88	0.60	0.48	0.36
1.00	1.00	1.00	0.00	0.00	0.00
0.50	0.50	0.50	0.00	0.00	0.00
0.50	0.50	0.50	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	0.93	0.44	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.00	1.00	1.00	1.00	1.00	0.00
0.50	0.50	0.50	0.50	0.50	0.50
0.50	0.50	0.50	0.50	0.50	0.50

Figure D-VI-4. H7MOVETGT

.30	.00	.00	.00	.00	.00
.30	.00	.00	.00	.00	.00
.60	.60	.00	.00	.00	.00
.60	.60	.60	.60	.60	.00
0.30	0.00	0.00	0.00	0.00	0.00
.60	0.60	0.60	0.00	0.00	0.00
0.30	0.30	0.00	0.00	0.00	0.00
0.60	0.60	0.60	0.60	0.60	0.60
1.00	1.00	1.00	1.00	1.00	1.00
.60	0.60	0.00	0.00	0.00	0.00
.98	.87	0.81	0.80	0.78	0.75
.98	.87	0.81	0.80	0.78	0.75
1.00	.95	0.80	0.73	0.71	0.69
1.00	1.00	0.88	0.72	0.68	0.64
1.00	.93	0.83	0.62	0.63	0.63
1.00	1.00	0.00	0.00	0.00	0.00
1.00	0.00	0.00	0.00	0.00	0.00
0.60	0.60	0.60	0.60	0.60	0.60
0.60	0.60	0.60	0.60	0.60	0.60
0.60	0.60	0.60	0.60	0.60	0.60
1.00	1.00	1.00	1.00	1.00	1.00
0.60	0.60	0.60	0.60	0.60	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.00	1.00	1.00	1.00	1.00	0.00
0.80	0.80	0.80	0.80	0.80	0.80
0.60	0.60	0.60	0.60	0.60	0.60
0.60	0.60	0.60	0.60	0.60	0.60
0.30	0.00	0.00	0.00	0.00	0.00
0.30	0.30	0.00	0.00	0.00	0.00
0.30	0.30	0.00	0.00	0.00	0.00
0.30	0.30	0.30	0.30	0.30	0.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
0.30	0.30	0.30	0.00	0.00	0.00
0.93	0.87	0.87	0.44	0.20	0.00
0.30	0.30	0.00	0.00	0.00	0.00
0.99	0.92	0.86	0.88	0.80	0.73
0.97	0.75	0.53	0.38	0.34	0.30
0.98	0.78	0.54	0.33	0.28	0.21
0.98	0.78	0.54	0.33	0.28	0.21
0.98	0.78	0.54	0.33	0.28	0.21
0.30	0.30	0.30	0.00	0.00	0.00
0.30	0.30	0.00	0.00	0.00	0.00
1.00	1.00	0.00	0.00	0.00	0.00
1.00	1.00	0.00	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	0.00	0.00	0.00
0.80	0.80	0.80	0.80	0.80	0.00
0.30	0.30	0.30	0.00	0.00	0.00
0.30	0.30	0.30	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
0.80	0.80	0.80	0.00	0.00	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00
0.60	0.60	0.60	0.60	0.60	0.60
0.60	0.60	0.60	0.60	0.60	0.60

Figure D-VI-5. H7MOVEFIRER

Section III. OUTPUT

D-VI-6. The Combat Module format modified SSPKs which are generated by the program H7AFP.M-SSPK are values from H7AFPPK which have been multiplied by moving target degradation factors, moving firer factors, and night degradation factors if those conditions were required. The structure is shown in Table D-VI-2. There are 7,200 lines, each with six values which correspond to the six SSPK range bands. The modified SSPKs are SECRET.

Table D-VI-2. Modified SSPK Element Structure

B01 vs R01 modified SSPKs
B01 vs R02
:
:
:
B01 vs R60
B02 vs R01
B02 vs R02
:
:
:
B02 vs R60
:
:
:
B60 vs R60
R01 vs B01
R01 vs B02
:
:
:
R01 vs B60
R02 vs B01
:
:
:
R60 vs B60

D-VI-7. There is a labeled modified SSPK element generated by the program H7AFP.M-SSPK. This element is also SECRET. Example lines are shown in Table D-VI-3. The SSPK values have been modified to keep this documentation UNCLASSIFIED. A header line shows the conditions of this conflict. The shooter name and AFP number are followed by target names and their AFP numbers, if any of the six SSPK values for that conflict are nonzero. If all values are zero, the next shooter name follows immediately. In the example, BTRP has SSPK values of .00 against all targets across all ranges. M-16 has .00 values against R03, R05-R31, R33-R60.

Table D-VI-3. Labeled Modified SSPK Element Structure

Blue shooter stationary vs Red open moving target at daytime

	Shooter = BTRP			Shooter number = B01			
	Shooter = M-16			Shooter number = B02			
RTRP	R01	.01	.01	.00	.00	.00	.00
AKS-74	R02	.02	.02	.00	.00	.00	.00
Gren 30	R04	.04	.00	.00	.00	.00	.00
SA-7	R32	.01	.00	.00	.00	.00	.00
	Shooter = 7.62 GM			Shooter number = B03			
RTRP	R01	.20	.10	.01	.00	.00	.00
AKS-74	R02	.20	.10	.01	.00	.00	.00
	Shooter = M650GM			Shooter number = B04			
ZSU-23	R26	.00	.00	.00	.00	.00	.01
SA-14	R30	.00	.00	.00	.00	.01	.00
SA-7	R32	.00	.00	.00	.01	.00	.00

Section IV. RUNSTREAM

D-VI-8. Figure D-VI-6 shows the runstream of the program which reads the pointer files and the SSPK values to generate modified SSPK values.

```
@QUAL UNCLASSIFIED
@ASG,T 7.
@ASG,T 8.
@ASG,T 9.
@ASG,T 10.
@ASG,T 11.
@ASG,T 12.
@ASG,T 13.
@ASG,A H7AFPPK/XXX/XXX.
@ASG,A *H7MOVETGT.
@ASG,A *H7MOVFIRER.
@ED H7AFPPK,7.
@USE 8.,*H7MOVETGT.
@USE 9.,*H7MOVFIRER.
@ED *H7AFPFILES.BPOINT,10.
@ED *H7AFPFILES.RPOINT,11.
EXIT
@XQT *H7AFP.870M-SSPK
1
2
1
1
2
1
1
1
10
8
6
1 2 3 4 5 6 42 44
1 2 3 42 44
@ED 12.,H7AFPPK.RAPD-D
@ED 13.,H7AFPPK.LABEL-RAPD-D
EXIT
```

Figure D-VI-6. Modified SSPK Generation Runstream

Section V. PROGRAM

D-VI-9. The program which generates modified SSPK values is H7AFP.M-SSPK. The absolute program is H7AFP.870M-SSPK. The flowchart for the program is in Figure D-VI-7. The program accepts conflict conditions for Side 1 (Blue) shooter and Side 2 (Red) targets: (1) open or defilade target; (2) stationary or moving target; (3) stationary or moving firer; and (4) day or night. The modified SSPK values, for Blue as shooter and Red as target are derived using BPOINT, SSPK values in H7AFPPK, and the degradation factors. Then, conflict conditions for Red as shooter and Blue as the targets are accepted. The Red modified SSPK values are derived using RPOINT, SSPK values in H7AFPPK, and the degradation factors depending on the second set of conflict conditions. The source program is shown in Figure D-VI-8.

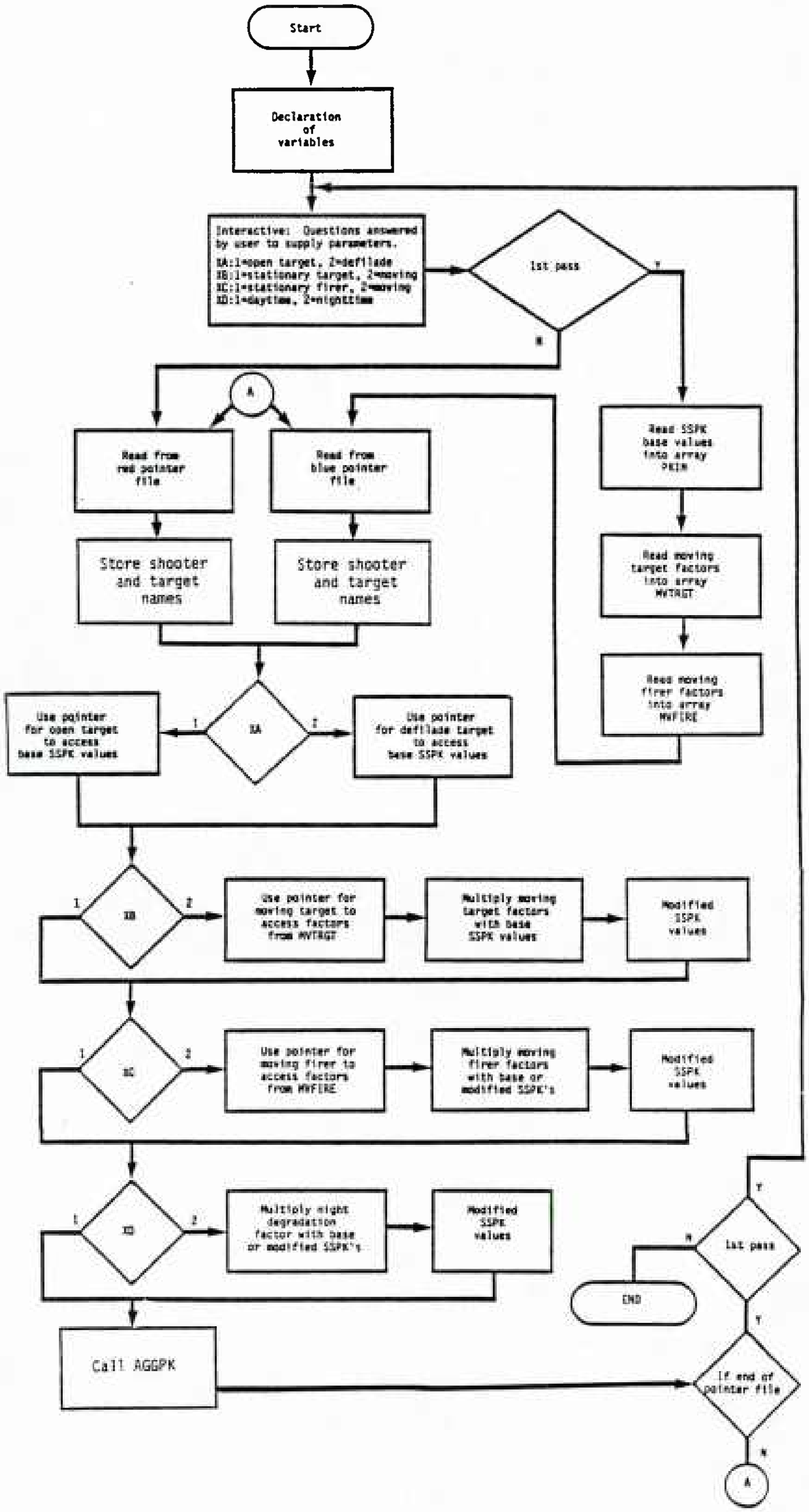


Figure D-VI-7. Modified SSPK Generation Program Flowchart (page 1 of 4 pages)

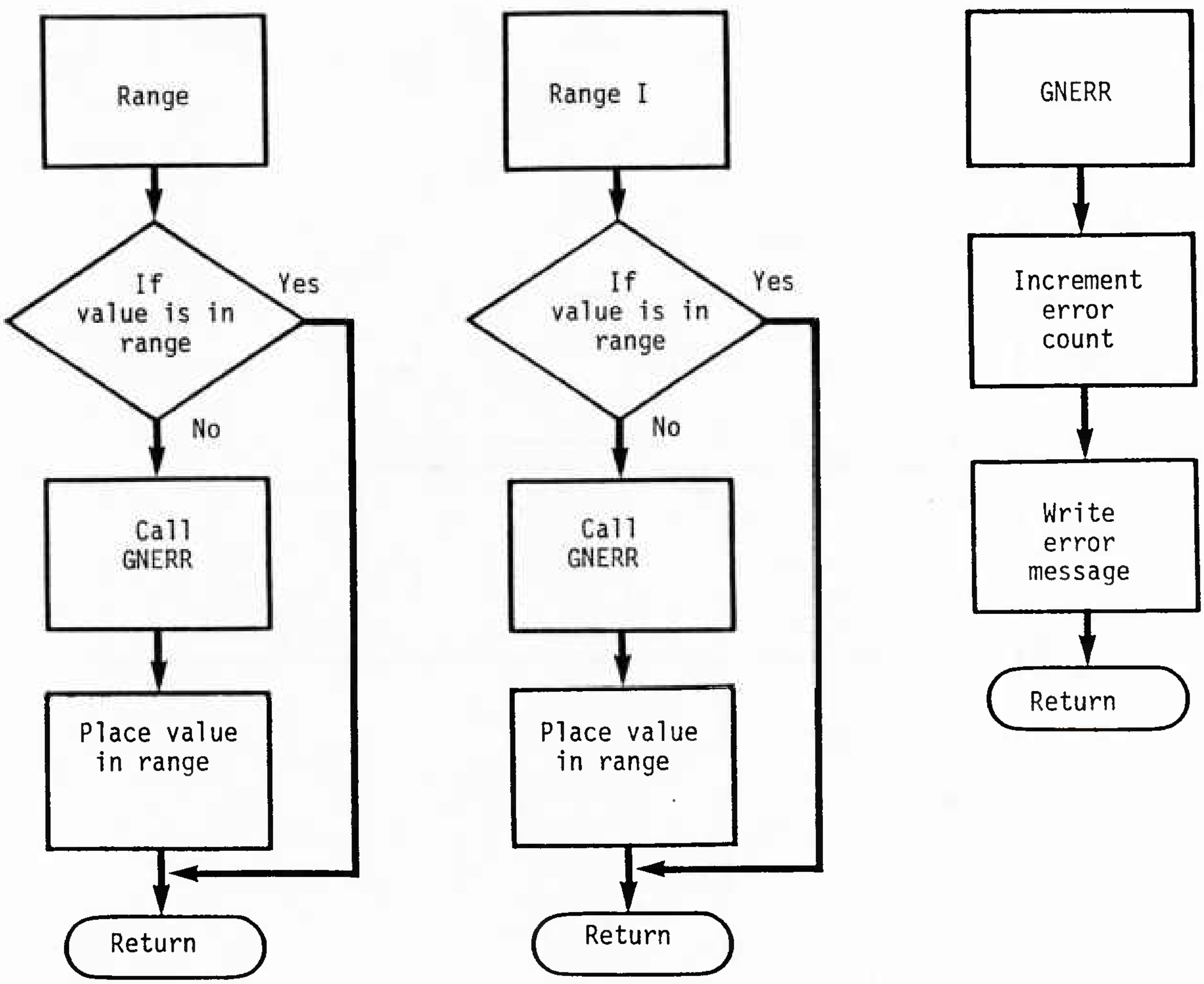


Figure D-VI-7. Modified SSPK Generation Program Flowchart (page 2 of 4 pages)

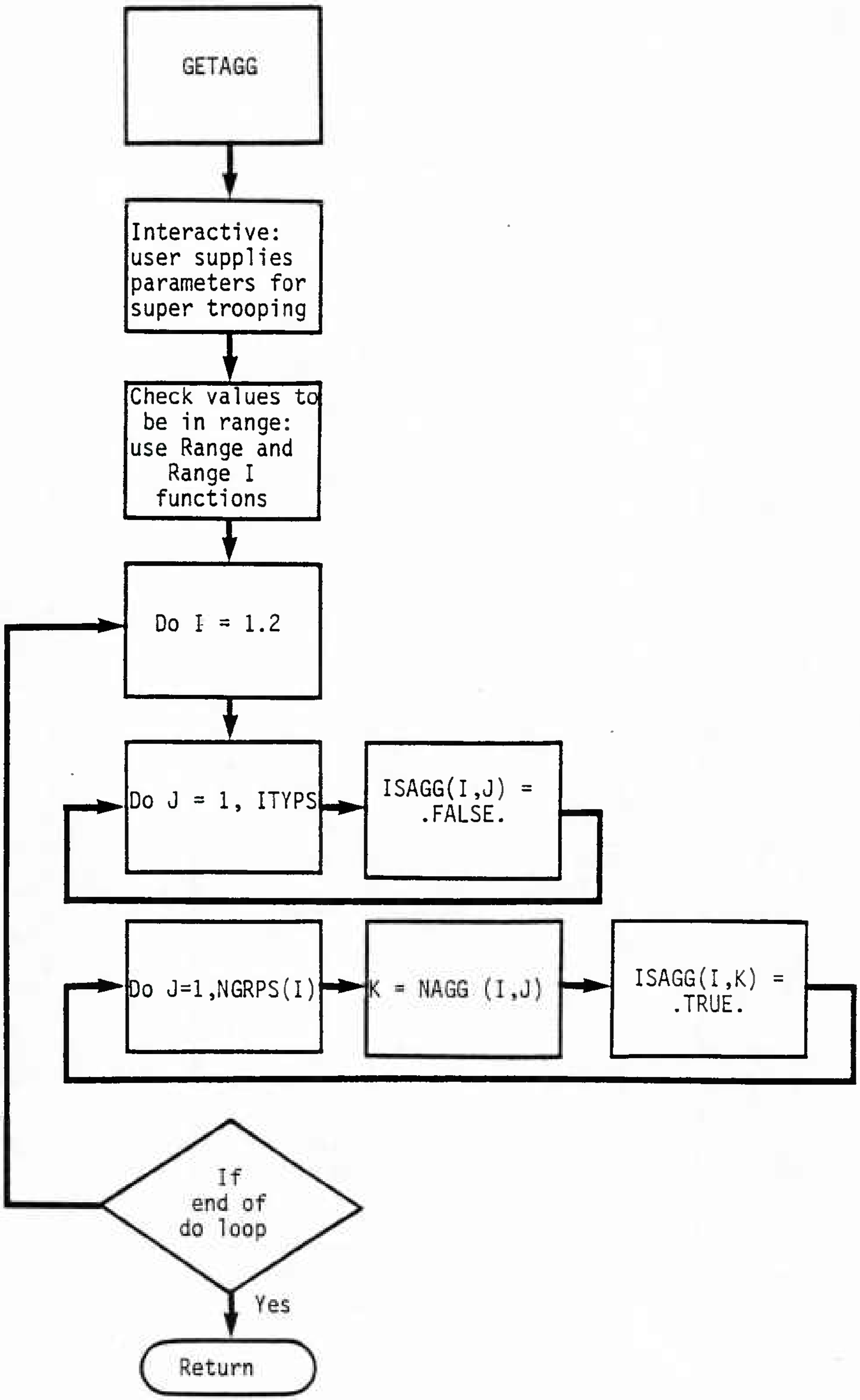


Figure D-VI-7. Modified SSPK Generation Program Flowchart
(page 3 of 4 pages)

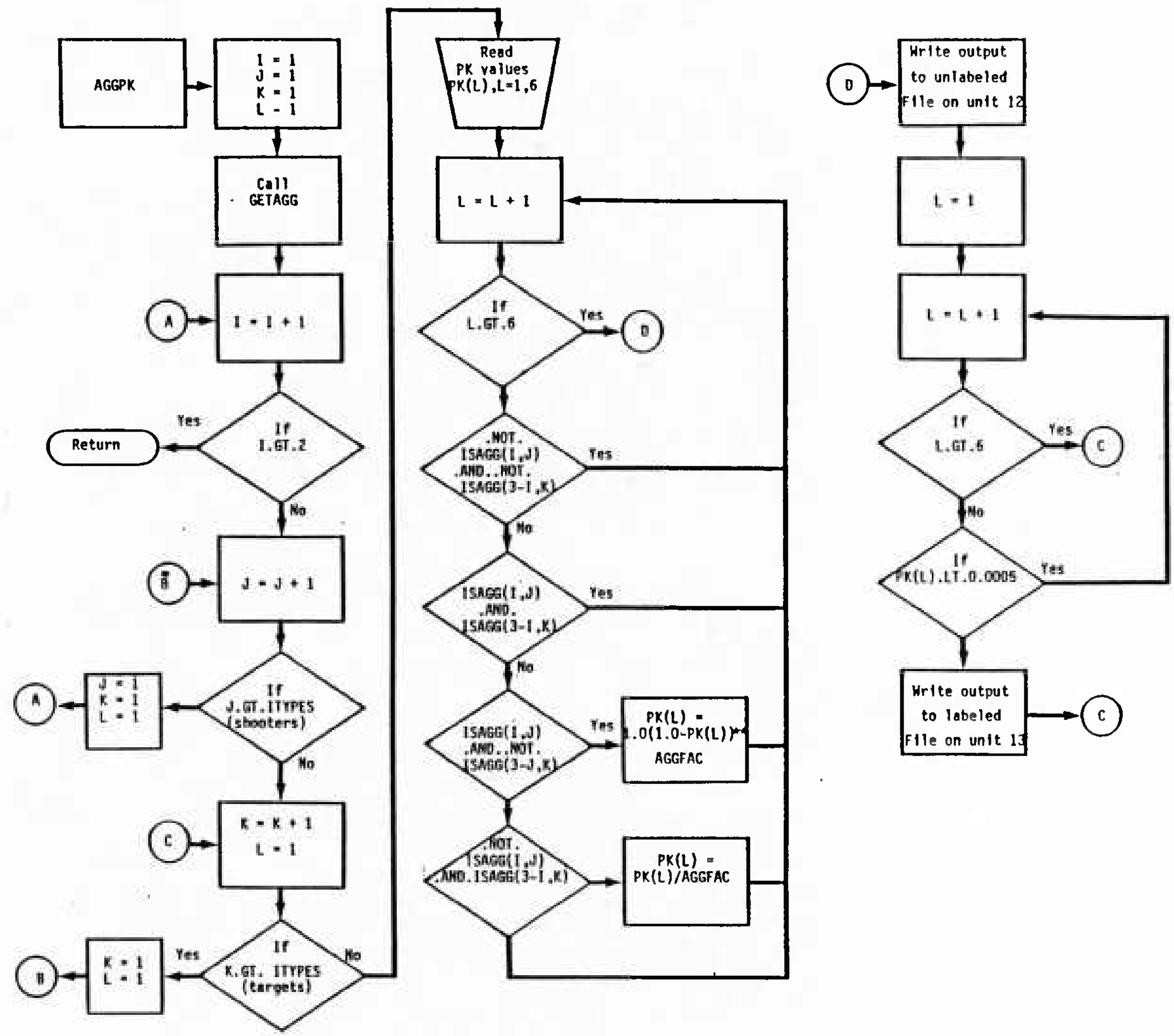


Figure D-VI-7. Modified SSPK Generation Program Flowchart
(page 4 of 4 pages)

```
      C
      C
      C             D E C L A R A T I O N    O F    D A T A
      C
      C
            COMMON/LABEL/TITLE(2),SHTNAM(2,60),TRGNAM(2,60)
            REAL PKIN(2000,6),MVTRGT(100,6),MVFIRE(100,6),TEMP(6),TEMP2(6)
            INTEGER XA,XB,XC,XD,INDEX,PFILE,LOOP
            REAL NITE
            CHARACTER*4 SCOLOR,TCOLOR
            CHARACTER*10 SHOOTR,ARRAY1(2),ARRAY2(2),ARRAY3(2)
            CHARACTER*3 SHTNUM
            CHARACTER*15 TRGNAM
            CHARACTER*39 SHTNAM
            CHARACTER*80 LINE,TITLE
            DATA ARPAY1/'   OPEN    ',' DEFILADE '/ARRAY2/'STATIONARY',
           &'  MOVING   '/ARRAY3/'DAYTIME    ','NIGHTTIME '/
      C
      C-----------------------------------------------------------------
      C
      C                    D A T A    D I C T I O N A R Y
      C
      C
      C     XA      - TARGET OPEN = 1, TARGET DEFILADE = 2
      C     XB      - TARGET STATIONARY = 1, TARGET MOVING = 2
      C     XC      - FIRER STATIONARY = 1, FIRER MOVING = 2
      C     XD      - DAYTIME = 1, NIGHTTIME = 2
      C     INDEX   - RECORD COUNTER IN POINTER FILE
      C     PFILE   - UNIT NUMBER OF POINTER FILE IN USE
      C     LOOP    - INDICATES 1ST OR 2ND PASS
      C     PKIN    - ARRAY OF BASE SSPK VALUES
      C     MVTRGT  - ARRAY OF MOVING TARGET FACTORS
      C     MVFIRE  - ARRAY OF MOVING FIRER FACTORS
      C     TEMP    - ARRAY WORKING AREA HOLDING BASE OR MODIFIED SSPK VALUES
      C     TEMP2   - ARRAY WORKING AREA HOLDING MOVING OR FIRER FACTORS
      C     NITE    - NIGHT DEGRADATION FACTOR
      C     ZERO    - LOGICAL VAPIABLE INDICATING ALL ZERO VALUES
      C     SCOLOP  - SHOOTER COLOR
      C     TCOLOR  - TARGET COLOR
      C     SHOOTR  - NAME OF SHOOTER WEAPON
      C     SHTNUM  - SHOOTEP WEAPON NUMBER
      C     TRGNAM  - ARRAY CONTAINING TARGET NAME AND NUMBEP
      C     SHTNAM  - ARRAY CONTAINING SHOOTER NAME AND NUMBER
      C     TITLE   - ARRAY CONTAINING THE TITLE OF THE REPORT
      C     LINE    - RECORD FROM POINTER FILE
      C     ARRAY1  - #
      C     APRAY2  - #   ARRAYS HOLDING PARAMETER CONDITIONS FOR CREATING
      C     APRAY3  - #   REPORT HEADINGS.
      C
      C-----------------------------------------------------------------
      C
      C
      C
      C
            LOOP=0
        1   IF(LOOP.EQ.0)THEN
                  SCOLOR='BLUE'
                  TCOLOR=' RED'
                  PFILE=10
            ELSE
                  SCOLOR=' RED'
                  TCOLOR='BLUE'
                  PFILE=11
            END IF
      C
      C
      C           THIS SECTION ASK THE USER A SET OF QUESTIONS TO
      C     OBTAIN THE PARAMETERS FOR THE PROCESSING OF THE SSPK'S
      C
      C
            WRITE(6,99)SCOLOR
       99   FORMAT(1X,T38,'MENU',/,10X,'  ANSWER THE FOLLOWING QUESTIONS TO ES
           &TABLISH THE PARAMETERS',/,10X,'  FOR SSPK MODIFICATIONS WITH ',A4,
           &' AS THE SHOOTER. ',//,
           &5X,'IS THE TARGET OPEN OR DEFILADE ?',//,10X,' 1=OPEN ',/,
           &10X,' 2=DEFILADE ')
       32   READ(5,30,ERR=31) XA
            IF((XA.GT.2).OR.(XA.LT.1)) GO TO 31
       30   FORMAT(I1)
            GO TO 33
       31   PRINT *, 'ILLEGAL ENTRY, TRY AGAIN'
```

Figure D-VI-8. Modified SSPK Generation Program (page 1 of 11 pages)

```
      GO TO 32
   33 WRITE(6,98)
   98 FORMAT(5X,'IS THE TARGET STATIONARY OR MOVING ?',//,
     &10X,' 1=STATIONARY ',/,10X,' 2=MOVING ')
  42  READ(5,30,ERR=41) XB
      IF((XB.GT.2).OR.(XB.LT.1)) GO TO 41
      GO TO 43
   41 PRINT *, 'ILLEGAL ENTRY, TRY AGAIN'
      GO TO 42
   43 WRITE(6,97)
   97 FORMAT(5X,'IS THE FIRER STATIONARY OR MOVING ?',//,
     &10X,' 1=STATIONARY ',/,10X,' 2=MOVING ')
   52 READ(5,30,ERR=51) XC
      IF((XC.GT.2).OR.(XC.LT.1)) GO TO 51
      GO TO 53
   51 PRINT *, 'ILLEGAL ENTRY, TRY AGAIN'
      GO TO 52
   53 WRITE(6,96)
   96 FORMAT(5X,'IS IT DAYTIME OR  NIGHTTIME ?',//,
     &10X,' 1=DAYTIME ',/,10X,' 2=NIGHTTIME ')
   62 READ(5,30,ERR=61) XD
      IF((XD.GT.2).OR.(XD.LT.1)) GO TO 61
       GO TO 73
   61 PRINT *, 'ILLEGAL ENTRY, TRY AGAIN'
      GO TO 62
C
C
C
   73 WRITE(6,90)
   90 FORMAT(////,T35,'P R O G R A M   W O R K I N G',////)
C
C
C           READ THE SSPK BASE VALUES,MOVING TARGER FACTORS,
C      AND THE MOVING FIRER FACTORS INTO THE APPROPRIATE ARRAYS
C      FOR FUTURE PROCESSING BY SUBSCRIPTS.
C
C
      IF(LOOP.EQ.0)THEN
      I=1
  111 READ(7,10,END=222)(PKIN(I,J),J=1,6)
   10 FORMAT(6(F10.2,1X))
      I=I+1
      GO TO 111
  222  K=1
  333 READ(8,20,END=444)(MVTRGT(K,L),L=1,6)
   20 FORMAT(6(F4.2,1X))
      K=K+1
      GO TO 333
  444 M=1
  555 READ(9,20,END=666)(MVFIRE(M,N),N=1,6)
      M=M+1
      GO TO 555
  666 CONTINUE
      END IF
C
C     STORE THE TITLES OF THE REPORTS
C
      M=LOOP+1
      WRITE(TITLE(M),5)SCOLOR,ARRAY2(XC),TCOLOR,ARRAY1(XA),
     &ARRAY2(XB),ARRAY3(XD)
    5 FORMAT(5X,A4,' SHOOTER ',A10,' VS ',A4,A10,',',A10,
     &' TARGET AT ',A10)
      INDEX=0
C
C
C           READ A RECORD FROM THE POINTER FILE AND PROCESS
C     ACCORDING TO THE PARAMETERS SUPPLIED DURING THE INTER-
C     ACTIVE PORTION OF THE PROGRAM.
C     STORE THE SHOOTER NAME AND NUMBER IN THE SHOOTER AND
C     TARGET NAME ARRAYS FOR CREATING LABELED REPORT.
C
C
      N=0
  777 READ(PFILE,70,END=888)LINE
   70 FORMAT(A80)
      INDEX=INDEX+1
      IF(INDEX.EQ.1)THEN
           IF(LINE(30:36).NE.'SHOOTER')THEN
                INDEX=0
                GO TO 777
           END IF
           SHOOTR=LINE(18:28)
```

Figure D-VI-8. Modified SSPK Generation Program
(page 2 of 11 pages)

```
                SHTNUM=LINE(39:41)
                 N=N+1
                 WRITE(SHTNAM(M,N),40)SHOOTR,SHTNUM
                 WRITE(TRGNAM(M,N),46)SHOOTR,SHTNUM
   46            FORMAT(A11,1X,A3)
   40     FORMAT('SHOOTER= ',A10,' SHOOTER NUMBER= ',A3)
          END IF
          IF(INDEX.LE.4) GO TO 777
          IF(INDEX.EQ.65) GO TO 777
          IF(INDEX.EQ.66) THEN
                INDEX=0
                GO TO 777
          END IF
          IF(XA.EQ.1)THEN
                READ(LINE,80)KOUNT
   80           FORMAT(T35,I3)
          ELSE
                READ(LINE,81)KOUNT
   81           FORMAT(T51,I3)
          END IF
          DO 15 J=1,6
          TEMP(J)=PKIN(KOUNT,J)
   15     CONTINUE
          IF(XB.EQ.2)THEN
                READ(LINE,82)KOUNT
   82           FORMAT(T42,I3)
                DO 25 J=1,6
                TEMP2(J)=MVTRGT(KOUNT,J)
                TEMP(J)=TEMP(J)*TEMP2(J)
   25           CONTINUE
          END IF
          IF(XC.EQ.2)THEN
                READ(LINE,83)KOUNT
   83           FORMAT(T57,I3)
                DO 35 J=1,6
                TEMP2(J)=MVFIRE(KOUNT,J)
                TEMP(J)=TEMP(J)*TEMP2(J)
   35           CONTINUE
          END IF
          IF(XD.EQ.2)THEN
                READ(LINE,84)NITE
   84           FORMAT(T66,F3.2)
                DO 45 J=1,6
                TEMP(J)=TEMP(J)*NITE
   45           CONTINUE
          END IF
C
C
C         WRITE THE MODIFIED SSPK VALUES TO UNIT 14 FOR FUTURE PROCESSING
C
C
          WRITE(14,50)(TEMP(J),J=1,6)
   50     FORMAT(6(F4.2,1X))
C
C
          GO TO 777
  888     WRITE(6,91)
   91     FORMAT(///,T31,'F I N I S H E D   P R O C E S S I N G',////)
C
C
C               IF THIS IS THE FIRST PASS, RETURN TO THE TOP TO
C         PROCESS RED AS A SHOOTER.
C
C
          IF(LOOP.EQ.0)THEN
                LOOP=1
                GO TO 1
          END IF
          CLOSE(14)
          REWIND 14
C
C         CALL SUBROUTINE TO COMPUTE SUPER TROOP VALUES
C
          CALL AGGPK
          WRITE(6,92)
   92     FORMAT(////,T33,'P R O G R A M   C O M P L E T E',////
         &,T35,'UNLABELED OUTPUT ON UNIT 12 ',/
         &,T35,'LABELED OUTPUT ON UNIT 13 ',//)
          STOP
          END
```

Figure D-VI-8. Modified SSPK Generation Program
(page 3 of 11 pages)

```
      SUBROUTINE AGGPK
C     ******************************************************************
C     *                                                                *
C     * AGGPK: COMPUTE PROBABILITIES OF KILL FOR AGGREGATED GROUPS     *
C     *                                                                *
C     ******************************************************************
C
      INCLUDE UNCLASSIFIED*98AFP2.PARM,LIST
      INCLUDE UNCLASSIFIED*98AFP2.ERRDAT,LIST
      REAL PK(6)
      LOGICAL ISAGG(2,ITYPS)
      COMMON /LABEL/TITLE(2),SHTNAM(2,60),TRGNAM(2,60)
      CHARACTER*80 TITLE
      CHARACTER*39 SHTNAM
      CHARACTER*15 TRGNAM
      LOGICAL ZERO
C
      NERRS = 0
C     GET AGGREGATION FACTOR AND AGGREGATED TYPES
      CALL GETAGG (AGGFAC, ISAGG)
C
C     READ, TRANSFORM, AND OUTPUT PK'S
C
      DO 600 I = 1,2  @ SHOOTING SIDE
      WRITE(13,10)TITLE(I)
10    FORMAT(A80)
        DO 700 J = 1,ITYPS  @ SHOOTING TYPE
            IF(ISAGG(I,J))WRITE(13,20)SHTNAM(I,J)
            IF(.NOT.ISAGG(I,J))WRITE(13,21)SHTNAM(I,J)
20          FORMAT(/,5X,'SUPER TROOP ',A39,/)
21          FORMAT(/,17X,A39,/)
          DO 800 K = 1,ITYPS  @ TARGET TYPE
            READ (14,51) (PK(L), L=1,6)
51          FORMAT(6(F4.2,1X))
            DO 900 L = 1,6
C             A LITTLE PARANOIA NEVER HURT ANYBODY
              PK(L) =  RANGE ('PK', 0.0, PK(L), 1.0)
              IF (NERRS .GT. 0) STOP
              IF (.NOT. ISAGG(I,J) .AND. .NOT. ISAGG(3-I,K)) THEN
                CONTINUE  @ NEITHER AGGREGATED
              ELSE IF (ISAGG(I,J) .AND. ISAGG(3-I,K)) THEN
                CONTINUE  @ BOTH AGGREGATED
              ELSE IF (ISAGG(I,J) .AND. .NOT. ISAGG(3-I,K)) THEN
                PK(L) = 1.0 - (1.0-PK(L)) ** AGGFAC  @ AGGREGATED SHOOTER
              ELSE IF (.NOT. ISAGG(I,J) .AND. ISAGG(3-I,K)) THEN
                PK(L) = PK(L) / AGGFAC  @ AGGREAGATED TARGET
              ENDIF
              PK(L) = MIN (PK(L), 0.99)
900         CONTINUE
C
C
C           WRITE ALL OF THE MODIFIED SSPK VALUES TO THE UNLABELED
C     OUTPUT FILE ON UNIT 12.  IF THE MODIFIED SSPK VALUE AT ANY
C     DISTANCE IS GREATER THAN 0.0 , THEN WRITE TO THE LABELED
C     OUTPUT FILE ON UNIT 13 ALSO.
C
C
            WRITE (12,101)
     &          (NINT(100.0*PK(L)), L=1,6)
101         FORMAT (6(I2,1X))
      ZERO=.TRUE.
      DO 55 L=1,6
C       PK(L)=TEMP(J)+0.005
      IF(PK(L).LT.0.005)THEN
           ZERO=.TRUE.
      ELSE
           ZERO=.FALSE.
           GO TO 56
      END IF
 55   CONTINUE
56    IF(ZERO)GO TO 62
      IF(ISAGG(3-I,K))WRITE(13,60)TRGNAM(3-I,K),PK
      IF(.NOT.ISAGG(3-I,K))WRITE(13,61)TRGNAM(3-I,K),PK
60    FORMAT(5X,'SUPER TROOP ',A15,5X,6(F4.2))
61    FORMAT(17X,A15,5X,6(F4.2))
62    CONTINUE
800         CONTINUE
700       CONTINUE
600   CONTINUE
      WRITE (6,63)
63    FORMAT (' AGGREGATION OF PK''S COMPLETED.')
      RETURN
      END
```

Figure D-VI-8. Modified SSPK Generation Program
(page 4 of 11 pages)

```
C
C   NAME: RANGE                          TYPE: REAL FUNCTION
C   PURPOSE: CHECK A REAL NUMBER FOR BEING IN RANGE.
C   ARGUMENTS:
C     NAME     DIMENSION     TYPE   USE  DESCRIPTION
C     ITEM     ----          C*(*)  I    NAME OF THE ITEM BEING TESTED
C     XLO      ----          R      I    LOWER BOUND
C     X        ----          R      I    THE NUMBER BEING TESTED
C     XHI      ----          R      I    UPPER BOUND
C   RETURNS: X, IF X IS WITHIN RANGE; XLO IF X IS LESS THAN XLO; OR
C   XHI IF X IS GREATER THAN XHI.
C   CALLED BY: APIN1, APIN2, GETAGG
C   CALLS: GNERR
C   COMMON BLOCKS USED: NONE
C   FILES USED: NONE.
C   LOCAL ARRAYS: NONE.
C   METHOD:  TEST TO SEE IF XLO <= X <= XHI.  IF NOT, CONSTRUCT AN ERROR
C   MESSAGE SAYING '<NAME> SHOULD BE IN THE RANGE <LOWER> TO <HIGHER>'
C   AND CALL GNERR TO ISSUE THE ERROR MESSAGE.
C
      FUNCTION RANGE (ITEM, XLO, X, XHI)
C     ******************************************************************
C     *                                                                *
C     * RANGE: CHECK A REAL NUMBER FOR BEING IN RANGE                  *
C     *                                                                *
C     ******************************************************************
      CHARACTER*(*) ITEM
      CHARACTER*70 MSG
C
      IF ((XLO .LE. X) .AND. (X .LE. XHI)) THEN
        CONTINUE
      ELSE
C       BUILD ERROR MESSAGE
        WRITE (MSG,66) X, XLO, XHI
66      FORMAT (1PG11.4,
     &    ' SHOULD BE IN THE RANGE ', 1PG11.4, ' TO ', 1PG11.4)
        CALL GNERR(ITEM,MSG)
      ENDIF
      RANGE = X
      IF (RANGE .LT. XLO) RANGE = XLO
      IF (RANGE .GT. XHI) RANGE = XHI
      RETURN
      END
```

Figure D-VI-8. Modified SSPK Generation Program
(page 5 of 11 pages)

```
C
C   NAME: RANGEI                         TYPE: INTEGER FUNCTION
C   PURPOSE: CHECK AN INTEGER FOR BEING IN RANGE.
C   ARGUMENTS:
C     NAME     DIMENSION    TYPE  USE  DESCRIPTION
C     ITEM     ----         C*(*) I    NAME OF THE ITEM BEING TESTED
C     IXLO     ----         I     I    LOWER BOUND
C     IX       ----         I     I    THE NUMBER BEING TESTED
C     IXHI     ----         R     I    UPPER BOUND
C   RETURNS: IX, IF IX IS IN RANGE; IXLO, IF IX IS LESS THAN IXLO; OR
C   IXHI IF IX IS GREATER THAN IXHI.
C   CALLED BY: APIN1, APIN2, GETAGG
C   CALLS: GNERR
C   COMMON BLOCKS USED: NONE
C   FILES USED: NONE.
C   LOCAL ARRAYS: NONE.
C   METHOD:  TEST TO SEE IF IXLO <= IX <= IXHI.  IF NOT, CONSTRUCT AN ERROR
C   MESSAGE SAYING '<NAME> SHOULD BE IN THE RANGE <LOWER> TO <HIGHER>'
C   AND CALL GNERR TO ISSUE THE ERROR MESSAGE.
C
      INTEGER FUNCTION RANGEI (ITEM, IXLO, IX, IXHI)
C     *****************************************************************
C     *                                                               *
C     * RANGEI: CHECK AN INTEGER FOR BEING IN RANGE                   *
C     *                                                               *
C     *****************************************************************
      CHARACTER*(*) ITEM
      CHARACTER*70 MSG
      IF ((IXLO .LE. IX) .AND. (IX .LE. IXHI)) THEN
        CONTINUE
      ELSE
C       BUILD ERROR MESSAGE
        WRITE (MSG,66) IX, IXLO, IXHI
66      FORMAT (I10, ' SHOULD BE IN THE RANGE ', I10, ' TO ', I10)
        CALL GNERR(ITEM,MSG)
      ENDIF
      RANGEI = IX
      IF (RANGEI .LT. IXLO) RANGEI = IXLO
      IF (RANGEI .GT. IXHI) RANGEI = IXHI
      RETURN
      END
```

Figure D-VI-8. Modified SSPK Generation Program
(page 6 of 11 pages)

```
C
C  NAME: GNERR                          TYPE: SUBROUTINE
C  PURPOSE: ISSUE AN ERROR MESSAGE RELATED TO INPUT DATA.
C  ARGUMENTS:
C     NAME     DIMENSION     TYPE  USE   DESCRIPTION
C     ITEM     ----          C*(*) I     NAME OF THE ITEM IN ERROR
C                                        (OR BLANK, IF THE MESSAGE DOES NOT
C                                        REFER TO A SPECIFIC ITEM)
C     MSG      ----          C*(*) I     THE ERROR MESSAGE
C  CALLED BY: APIN1, APIN2, ICOLOR, RANGE, RANGEI
C  CALLS: NONE
C  COMMON BLOCKS USED: ERRDAT
C  FILES USED: NONE.
C  LOCAL ARRAYS: NONE.
C  METHOD:  PRINT THE NAME OF THE ITEM IN ERROR AND THE ASSOCIATED MESSAGE.
C  INCREMENT VARIABLES NERRS (THE TOTAL NUMBER OF ERRORS DETECTED THUS FAR)
C  AND NERRSL (THE NUMBER OF ERRORS ON THE CURRENT INPUT LINE) IN COMMON BLOCK
C  ERRDAT.  IF NERRS EXCEEDS 30, ISSUE A FINAL ERROR MESSAGE AND STOP.
C
      SUBROUTINE GNERR (ITEM, MSG)
C     ****************************************************************
C     *                                                              *
C     * GNERR: ISSUE AN ERROR MESSAGE                                *
C     *                                                              *
C     ****************************************************************
C
      INCLUDE UNCLASSIFIED*98AFP2.ERRDAT,LIST
      CHARACTER*(*) ITEM, MSG
C
C     INCREMENT ERROR COUNT
      NERRS = NERRS+1
      NERRSL = NERRSL+1
C     PRINT THE MESSAGE
      IF (ITEM .EQ. ' ') THEN
        WRITE (6,61) MSG
61      FORMAT (' ERROR: ', A)
      ELSE
        WRITE (6,62) ITEM, MSG
62      FORMAT (' ERROR: ', A, ': ', A)
      ENDIF
      IF (NERRS .GT. 30) THEN
        WRITE (6,63)
63      FORMAT (' ERROR LIMIT EXCEEDED...PROCESSING STOPPED.')
        STOP
      ENDIF
      RETURN
      END
```

```
C     THIS IS ONLY A COPY OF THE PROC ERRDAT.  THE ACTUAL PROC
C     IS LOCATED ON  UNCLASSIFIED*98AFP2.ERRDAT
C
C  BLOCK NAME: ERRDAT
C  USED BY: APIN1, APIN2, APOUT, ARTPRE, GETAGG, GNERR
C  BLOCK DESCRIPTION:
C     NAME     DIMENSION        TYPE  DESCRIPTION
C     NERRS    ----             I     TOTAL NUMBER OF ERRORS DETECTED
C     NERRSL   ----             I     TOTAL NUMBER OF ERRORS DETECTED ON
C                                     THE CURRENT INPUT LINE
C
ERRDAT PROC
      COMMON /ERRDAT/ NERRS, NERRSL
 END
```

Figure D-VI-8. Modified SSPK Generation Program
(page 7 of 11 pages)

```
C  CODES USED IN NAMING THE COMMON BLOCKS
C      I            INPUT
C      M            MAIN
C      1            FRCALC
C      2            DIRIO
C      3            CNFALC
C      4            STPREP
C      5            CNFTMS
C      6            EXTKLS
C      7            DIRKLS
C      8            CNFLCI
BK1    PROC
C USED IN INPUT, MAIN, FRCALC, DIRIO, CNFALC, STPREP, EXTKLS, DIRKLS
C
       COMMON/CIT057 /NTYPS(2),CASE,PREF1,PREF2,PREF3,PREF4,INSEED,DSEED
       INTEGER CASE, PREF1, PREF2, PREF3, PREF4
       DOUBLE PRECISION DSEED
 END
BK2    PROC
C  USED IN MAIN, CNFALC, STPREP, CNFTMS, EXTKLS, DIRKLS
C
       COMMON/C34567/TYPS(2), SMALER, BIGGER, STATUS(ICONFL,ILEN),CNFLCT,
     * CSHOTS(2,ICONFL)
       INTEGER TYPS, SMALER, BIGGER, CNFLCT, CSHOTS
 END
BK3    PROC
C  USED IN INPUT, MAIN, STPREP, EXTKLS
C
       COMMON/CIM46 /IOUNIT, OUTFIL, NSTAGE, NSTAGO, NDAYS
       INTEGER OUTFIL
 END
BK4    PROC
C  USED IN CNFALC, STPREP
C
       COMMON/C34   /NODDS, ODDS(2), CONFLO(2,IRANGE,IENVIR)
       INTEGER ODDS, CONFLO
 END
BK5    PROC
C  USED IN INPUT, CNFTMS
C
       COMMON/CI5   /LIMITS(2,ITYPS,2),
     * SENSOR(2,ITYPS,IENVIR,2), SIZE(2,ITYPS,IENVIR),
     * CNTRST(2,ITYPS,IENVIR,ISSNTR), LIGHT(IENVIR), MAG(ISENSR),
     * ATTN(ISENSR,IENVIR), BRIGHT(IENVIR), RCDET(2,ITYPS), NDTCTN,
     * NS2DCT(2,ITYPS)
       INTEGER SENSOR
       REAL LIGHT,MAG
 END
BK6    PROC
C  USED IN INPUT, MAIN, EXTKLS, DIRKLS
C
       COMMON/CIM67  /NOEXT(2),EXT(2,IEXT,ITYPS,IENVIR,IRANGE)
     * ,EXTN(2,IEXT,IENVIR,IRANGE,ICDPTH)
     * ,EXTK(2,IEXT,IENVIR,IRANGE,ICDPTH), EXTPER(2,IEXT)
     * ,EXTN1(2,IEXT,IENVIR,IRANGE), SHOTS(2,IENVIR,IRANGE)
     * ,AMOTYP(2,ITYPS,ITYPS), FSHOTS(2,IEXT,IENVIR,IRANGE)
     * ,BWPNS(2,ITYPS,IENVIR,IRANGE),INDDST(2,IEXT,IRANGE),LESONS(2)
     * ,INDPAR(2,IEXT,ITYPS)
       INTEGER EXTK, EXTN1, SHOTS, AMOTYP, BWPNS
       REAL INDDST, INDPAR
       LOGICAL LESONS
 END
BK7    PROC
C  USED IN MAIN, FRCALC, EXTKLS
C
       COMMON/CM16   /IOPTR3
 END
BK8    PROC
C  USED IN INPUT, MAIN, FRCALC
C
       COMMON/CIM1   /INSERT(IPHASE,2,ITYPS), EXTLOS(2,ITYPS),
     * PHASE(IDAYS)
       INTEGER PHASE
 END
BK9    PROC
C  USED IN INPUT, CNFALC, DIRKLS
C
       COMMON/CI37   /ENVDST(IENVIR)
 END
BK10   PROC
C  USED IN MAIN, DIRIO, DIRKLS
C
```

Figure D-VI-8. Modified SSPK Generation Program
(page 8 of 11 pages)

```
      COMMON/CM27  /IOPTR4, CKILLS(2,ITYPS2,7)
      INTEGER CKILLS
 END
BK11  PROC
C  USED IN MAIN, STPREP, DIRKLS
C
      COMMON/CM47  /PKS(2,ITYPS,IPKS,IRANGE), MAXCON,
     *    IU1S, IU1T1, IU1T2, KEYU1, PK1(IRANGE)
 END
BK12  PROC
C  USED IN MAIN, CNFALC
C
      COMMON/CM3   /MAXODS
 END
BK13  PROC
C  USED IN MAIN, FRCALC, DIRIO, EXTKLS
C
      COMMON/CM126  /FORCES(2,ITYPS,ITYPS)
      INTEGER FORCES
 END
BK14  PROC
C  USED IN CNFALC,EXTKLS
C
      COMMON/C36   /CTR(2),DAYST(2)
     * ,SAVIT(4)
      INTEGER CTR , DAYST, SAVIT
 END
BK15  PROC
C  USED IN EXTKLS, DIRKLS
C
      COMMON/C67   /ATEMP(3,IPKS)
 END
BK16  PROC
C  USED IN MAIN, CNFALC, DIRIO, STPREP
      COMMON /CM234  / PROJCT(ITYPS,5)
      REAL PROJCT
 END
BK17  PROC
C  USED IN MAIN, FRCALC, EXTKLS, DIRKLS
      COMMON /CM167/ WPNS(2,ITYPS),WPNS1(2,ITYPS)
     * ,DUELS(IENVIR,IRANGE,ICDPTH,5)
     * ,TMS(2,IENVIR,IRANGE,ICDPTH,5), WPNS11(2,ITYPS)
      INTEGER WPNS, WPNS1, DUELS, WPNS11
 END
BK18  PROC
C  USED IN CNFTMS, DIRKLS
      PARAMETER RMAXT=2.0**30
 END
BK19  PROC
C  USED IN MAIN, CNFALC, EXTKLS, DIRKLS
      COMMON /CM367/ FORCS1(2)
      INTEGER FORCS1
 END
BK20  PROC
C  USED IN MAIN, INPUT, CNFALC
      COMMON /CIM3/ NENVIR
 END
BK21  PROC
C  USED IN MAIN, DIRIO, CNFALC, DIRKLS
      COMMON /CM237/ NUMWPN(ITYPS,2)
 END
BK22  PROC
C  USED IN INPUT, STPREP, CNFTMS
      COMMON /CI45  / REFIRE(2,ITYPS,ITYPS,IENVIR)
 END
BK23  PROC
C  USED IN MAIN, CNFALC, DIRKLS
      COMMON /CM37  /FORCS2(2),NOTPAR(2,IENVIR,IRANGE)
      INTEGER FORCS2,NOTPAR
 END
BK24  PROC
C  USED IN CNFALC, DIRKLS
      COMMON /C37  / RNGDST(IRANGE)
 END
BK25  PROC
C    USED IN MAIN, INPUT, CNFLC1, EXTKLS, DIRKLS
      COMMON/CIM678/ LOG(ICDPTH,2,ITYPS,9),LOGIT,STATE
      LOGICAL LOGIT,STATE
 END
BK27  PROC
C    USED FOR OUTPUT FILES
      COMMON /CM67/  TTLOS(2,ITLOS,IENVIR,IRANGE)
```

Figure D-VI-8. Modified SSPK Generation Program
(page 9 of 11 pages)

```
            INTEGER TTLOS
       END
BK28        PROC
C       USED IN CIPS, COPS, AND AFP FILE GENERATION
C
            PARAMETER ITTOVC=4, ITTOTC=12, IINF=1
C
            COMMON /0123/ TTOVC(2,ITYPS,ITTOVC), CTOVC(2,ITTOTC,ITTOVC)
          & , STOVC(2,ITTOVC), PERK(2,ITYPS), IVCM(2), ICM(2)
          & , CKILS1(ITYPS1,ITYPS2,2), PLOSS(2,ITYPS,ITYPS,3)
          & , TPTOVC(2,ITYPS), TPTOTC(2,ITYPS), NOS(2,ITYPS)
          & , CIPS1(2,ITYPS,ITTOVC),JTTOVC(2),JTTOTC(2)
          & , LOSSES(2,ITYPS), FAC(2,ITYPS)
C
            INTEGER TPTOVC, TPTOTC
C
            REAL PLOSS, CKILS1, PERK, TTOVC, CTOVC, STOVC, NOS, CIPS1, LOSSES
C
       END
BK29        PROC
C       USED IN INDLOS,TPXTP
                COMMON/COM1/INLOSS(2,ITLOS)
       END
PARM        PROC
C       USED IN INPUT, MAIN, FRCALC, DIRIO, CNFALC, STPREP,
C                  CNFTMS, EXTKLS, DIRKLS
C
C   GENERAL USAGE PARAMETERS
            PARAMETER SMALL1=.00001,SMALL2=.000001
            PARAMETER IPHASE=2,IDAYS=7,ICDPTH=5
            PARAMETER IODDS=50,IENVIR=1,IRANGE=6,ISGNTR=2
            PARAMETER ICONFL=1500,ISTGF=10000000,IPKS=4
            PARAMETER ITYPS1=60,ITYPS2=60,ITYPS=MAX(ITYPS1,ITYPS2),IREPS=1
            PARAMETER LNCNT=30
            PARAMETER IPOS=4, IDAY=2, IVIS=2, ICIPTP=4
            PARAMETER ICASE=IPOS*IVIS*IDAY , ITHTR=1, ITPD=1, ICOMBO=1
            PARAMETER IP1= ICASE*ITHTR , IP3=1
            PARAMETER IP4=ITHTR*ICASE, IP2=ITHTR*ICASE*ITPD
            PARAMETER IP5=ICOMBO*ITHTR*ICASE*ITPD
C     PARAMETERS FOR THE EXTERNAL KILLS
            PARAMETER  IEXT1=10,IEXT2=10,IEXT=MAX(IEXT1,IEXT2)
            PARAMETER  ILEN=IODDS+3, ITLOS=IEXT+1
C   PARAMETERS FOR DIRECT I/O
            PARAMETER N11=ITYPS*IENVIR,N12=IENVIR,IU1=20
             PARAMETER IRSZ1=3+IRANGE
            PARAMETER N21=ITYPS2,IU2=21
            PARAMETER NR2=IP1*ITYPS1*N21,IRSZ2=IRANGE*4+10
            PARAMETER NR6=IP1*ITYPS1,IRSZ6=5*ITYPS*6+10,IU6=25
            PARAMETER NR7=IP1*2*ITYPS,IRSZ7=2*ITYPS*8+10,IU7=26
            PARAMETER N84=IRANGE,N83=IENVIR*N84,N82=ITYPS*N83
          &    ,N81=IDAYS*N82,N80=IREPS*N81
            PARAMETER NR8=IP3*2*N80
            PARAMETER IRSZ8=ICDPTH*ITYPS*9*6+10,IU8=27
C   PARAMETERS FOR DETECT INTERFACES
            PARAMETER ISENSR=50
C   PARAMETERS FOR CKILLS DIRECT ACCESS I/O
            PARAMETER N31=ITYPS1,IU3=22
            PARAMETER NR3=IP5*IREPS*N31,IRSZ3=2*ITYPS2*7*6+10
            PARAMETER N41=ITYPS1,IU4=23
            PARAMETER NR4=IP3*IREPS*N41,IRSZ4=2*ITYPS*6+10
            PARAMETER IU5=24
            PARAMETER IRSZ5=4 + 2*2*IEXT*IENVIR*IRANGE*ICDPTH
            PARAMETER N90=IREPS, IU9=7
            PARAMETER NR9=IP5*IREPS*IDAYS,IRSZ9=2*ITYPS*ITYPS*6+10
            PARAMETER N100=IREPS, N101=IREPS*IDAYS, IU10=4
            PARAMETER NR10=IP5*N101*ITYPS1,IRSZ10=2*ITYPS2*6+10
            PARAMETER NR11=IP5*ICIPTP,IU11=7,IRSZ11=2*ITYPS*ICIPTP*10+10
            PARAMETER NR12=IP5*ICIPTP,IU12=8,IRSZ12=2*ITYPS*ICIPTP*10+10
            PARAMETER NR14=IP3*IREPS*IDAYS*ITYPS1*ITYPS2, IU14=31
          &     ,IRSZ14=(3+IENVIR*IRANGE*ICDPTH*5)+10
            PARAMETER IU15=32,IRSZ15=5+2*IENVIR*IRANGE*ICDPTH*5
C   PARAMETERS FOR NON-DIRECT ACCESS FILES
            PARAMETER N132=IREPS, N131=IDAYS*N132, IU13=9
            PARAMETER IRSZ13=2*ITLOS*IENVIR*IRANGE+4
            PARAMETER IUIN1=10, IUIN2=11, IUIN3=12, IUIN4=13, IUIN6=15
            PARAMETER IUIN7=16, IUIN8=19, IAFPIN=17, IUOUT1=18
            PARAMETER IUOUT3=33, IRSZU3=4 + IRANGE*IENVIR*IEXT*2 +
          &      IRANGE*IENVIR*2
C
C           INPUT FILES              NUMBER             UNIT            FORMAT
C
C           03PKS3A                    10               IUIN1           24F3.3
```

Figure D-VI-8. Modified SSPK Generation Program
(page 10 of 11 pages)

```
C       03RNGDST3A                11           IUIN2        24I3
C       03PREF3A                  12           IUIN3        FREE
C       03PROJ3A                  13           IUIN4        FREE
C       03KILLSD3                 15           IUIN6        FREE
C       03CIPSD3                  16           IUIN7        FREE
C       03CSCSS                   14           IUIN8        FREE
C       03AFPIN                   17           IUIN9        IN SPECS
C
C      FILE TO UNIT ASSOCIATIONS
C
C       FILE          UNIT     NUMBER      FORMAT
C
C       PKS           IU1        20        UNFORMATTED (BINARY)
C       RNGDST        IU2        21        F4.2
C       CKILLS        IU3        22        I6
C       NUMWPN        IU4        23        I6
C       EXT           IU5        24        UNFORMATTED (BINARY)
C       PROJCT        IU6        25        F6.2
C       PREF          IU7        26        F8.6
C       LOG           IU8        27        I6
C       ALLOCATE      IU9         3        I6
C       FALLOC        IU10        4        I6
C       CIPS          IU11        7        F10.3
C       COPS          IU12        8        F10.3
C       TTLOS         IU13        9       UNFORMATTED (BINARY)
C       AFPIN         IAFPIN     17       IN  SPECS
C       AFPOUT        IUOUT1     18       IN  SPECS
C       ESHOTS,SHOTS  IUOUT3     33       UNFORMATTED (BINARY)
C       DUELS         IU14       31       I5
C       TIMES         IU15       32       UNFORMATTED (BINARY)
  END
```

Figure D-VI-8. Modified SSPK Generation Program
(page 11 of 11 pages)

ANNEX VII TO APPENDIX D

PROGRAMS AND NOTES FOR FOUR COMBAT MODULE PREPROCESSORS - PREFGEN, RNGDSTGEN, PKSGEN, AND PROJGEN

Section I. OVERVIEW

D-VII-1. This annex describes briefly four preprocessors of the Combat Module. All four preprocessors transform previously prepared input data into files in the forms required by the principal program of the Combat Module. The relation of the four preprocessors to the AFP Combat Module, and AFP System in general, is portrayed in Figure D-VII-1.

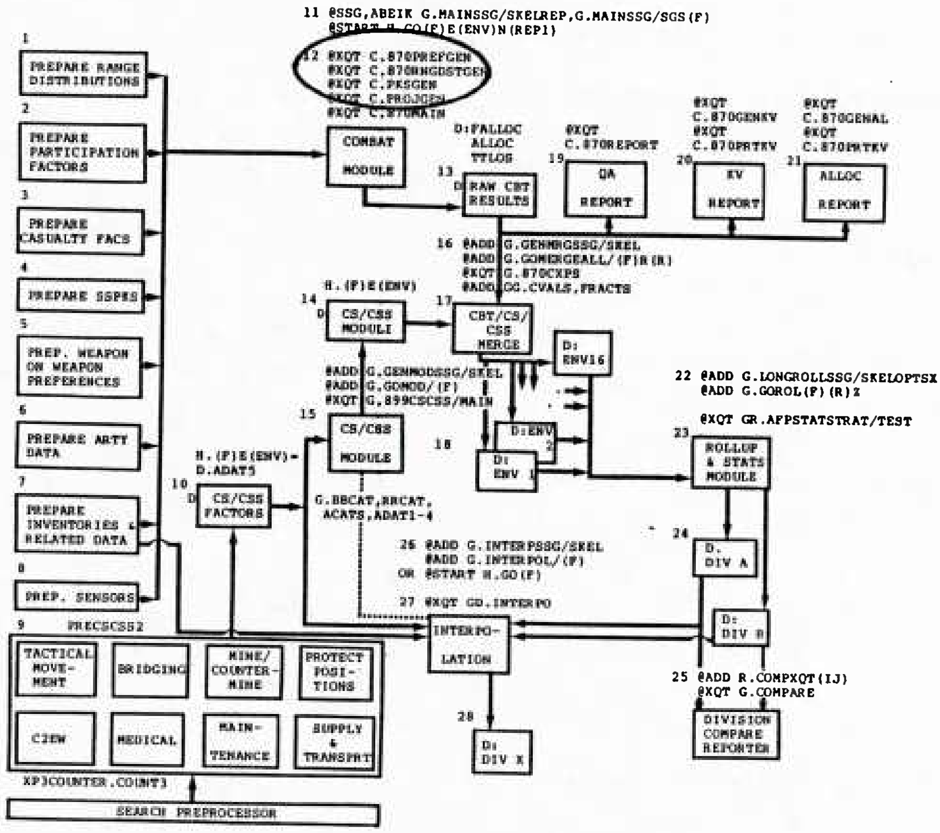


Figure D-VII-1. The Relation of Four Combat Module Preprocessors (PREFGEN, RNGDSTGEN, PKSGEN, and PROJGEN) to the Module's Principal Program (MAIN)

Section II. PREPROCESSOR PROGRAMS AND NOTES

D-VII-2. PREFGEN

a. Program PREFGEN processes the sequential data file PREF previously created in task 5 of the Combat Module input data preparation. The file

PREF and task 5 are described in Annex II to Appendix D. The file PREF contains user-specified fractional preferences for shooting weapons to engage targets (which, too, may be able to shoot). Program PREFGEN generates so-called "modified preferences" in order to reconcile the possible conflicting preferences of opponents. The original preferences of two opposing types are first moved "halfway toward their arithmetic mean." These intermediate values are then "renormalized" so that the sum of the modified preferences for each weapon equals the sum of the original preferences. The modified preferences are then output to a direct access file in the form required by the main program of the Combat Module.

b. The PREFGEN source program is listed in Figure D-VII-2.

c. Input Files. Unit 15 - basedata

Unit 12 - preference file

d. Runstream Input. String of six 1's separated by blanks.

e. Format of Sequential Symbolic Preference Input File. Free field format.

f. Method

(1) For each combination of preferences, each preference is moved halfway toward the average.

Example:	Blue Type 1 Preference for Red Type 2	Red Type 2 Preference for Blue Type 1
	.2	.4

The average is .3

Moving the Blue preference halfway toward the average:

(.2 + .3)/2 = .25

Moving the Red preference halfway toward the average:

(.4 + .3)/2 = .35

So the new (modified) preferences are:

Blue Type 1 Preference for Red Type 2	Red Type 2 Preference for Blue Type 1
.25	.35

```
UNCLASSIFIED*H7AFP(1).PREFGEN(2)
                 INCLUDE PAR*
                 INCLUDE BK1
                 INCLUDE BK3
      C
                 DEFINE FILE IU7(NR7,IRSZ7,F,NEXT7)
      C
                 PARAMETER ERROR=.02
      C
                 DIMENSION PREFS(2,ITYPS,ITYPS),PREFS1(2,ITYPS,ITYPS)
                 DIMENSION SUMS(2,ITYPS),SUMS1(2,ITYPS)
      C
                 REAL PREFS,PREFS1,SUMS,SUMS1
      C
                 INTEGER COMBO,TPD,THTR,NVIS,NDAY,NPOS
      C
                 LOGICAL ERRORS
      C
      2          CONTINUE
                 ERRORS=.FALSE.
                 READ(IUIN6,100) NDAYS,NTYPS(1),NTYPS(2)
      100        FORMAT()
                 READ(5,100) COMBO,TPD,THTR,NVIS,NDAY,NPOS
                 CASE=(NVIS-1)*IDAY*IPOS+(NDAY-1)*IPOS+(NPOS-1)
                 PREF1=(THTR-1)*ICASE+CASE
                 DO 1000 I=1,2
                   READ(IUIN3,100) ((PREFS(I,J,K),K=1,NTYPS(3-I)),
               &        J=1,NTYPS(I))
      1000       CONTINUE
                 DO 3000 I=1,2
                 DO 3000 J=1,NTYPS(I)
                   SUMS(I,J)=0.0
                   DO 2500 K=1,NTYPS(3-I)
                     SUMS(I,J)=SUMS(I,J)+PREFS(I,J,K)
      2500         CONTINUE
                   IF ( SUMS(I,J) .GT. 1.0 + ERROR ) THEN
                     WRITE(6,2600) J,I,SUMS(I,J)
      2600           FORMAT(/,1X,10(1H*),' PREFERENCES FOR TYPE ',I3,' SIDE ',I1
               &           ,3X,'SUM TO ',F9.6,1X,10(1H*) )
                     ERRORS=.TRUE.
                   ENDIF
      3000       CONTINUE
                 IF ( ERRORS ) CALL FABORT
                 DO 4000 J=1,NTYPS(1)
                 DO 4000 K=1,NTYPS(2)
                   TEMP=(PREFS(1,J,K)+PREFS(2,K,J))/2.0
                   PREFS1(1,J,K)=(PREFS(1,J,K)+TEMP)/2.0
                   PREFS1(2,K,J)=(PREFS(2,K,J)+TEMP)/2.0
      4000       CONTINUE
                 DO 5000 I=1,2
                 DO 5000 J=1,NTYPS(I)
                   SUMS1(I,J)=0.0
                   DO 5000 K=1,NTYPS(3-I)
                     SUMS1(I,J)=SUMS1(I,J)+PREFS1(I,J,K)
      5000       CONTINUE
                 DO 6000 I=1,2
                 DO 6000 J=1,NTYPS(I)
                 DO 6000 K=1,NTYPS(3-I)
                   IF ( SUMS1(I,J) .GT. .001 ) THEN
                     PREFS1(I,J,K)=(PREFS1(I,J,K)*SUMS(I,J))/SUMS1(I,J)
                   ELSE
                     PREFS1(I,J,K)=0.0
                   ENDIF
      6000       CONTINUE
                 DO 2000 I=1,2
                 DO 2000 J=1,NTYPS(I)
                   IOPTR7=(I-1)*ITYPS+J
                   IOPTR7=IOPTR7+PREF1*2*ITYPS
                   WRITE(IU7'IOPTR7,1100,ERR=1500)
               &        (PREFS1(I,J,K),K=1,NTYPS(3-I))
      1100         FORMAT(500(500F8.6))
                   GOTO 2000
      1500         WRITE(6,1600) I,J
      1600         FORMAT(30(1H*),5X,'PREF WRITE ERROR',5X,'INDICES= ',
               &        2(I5,5X),30(1H*))
                   STOP
      2000       CONTINUE
                 STOP
                 END
```

Figure D-VII-2. Source Listing of the Program PREFGEN, a Combat Module Preprocessor

(2) The original sum of a weapon's preferences is maintained.

Example: Let's say the raw preferences for Blue type 1 sum to .8 and the modified preferences sum to .9.

Each modified preference for Blue type 1 against each opponent is multiplied by .8/.9 so the new modified preferences again sum to .8.

(3) Although PREFGEN is currently "hardwired" to move original opposing preferences halfway (f = 0.5) toward their mean before renormalization, the method may be easily generalized for any fraction (f) toward the mean. Let p and q be the original opposing preferences. Let p' and q' be the opposing modified preferences before renormalization. Then

$$p' = (1 - f/2)p + (f/2)q$$

$$q' = (f/2)p + (1-f/2)q$$

The current value, f=0.5, was chosen arbitrarily as a "neutral" value. Some unbalanced weapons allocations have occurred with f=0.5. Although the imbalances are not the fault of f=0.5, a different value of f may produce better though hardly perfect weapons allocations. One disadvantage of fine tuning f is that f is not a "natural" combat parameter.

g. Output File. Unit 26, direct access.

(1) Pointer to Record.

(Side - 1) + ITYPS + ID of weapon type

(2) **Record Format.** The preferences against all opponents in F8.6 format.

D-VII-3. RNGDSTGEN

a. Program RNGDSTGEN processes the sequential data file RNGDST previously created in task 1 of the Combat Module input data preparation. The file RNGDST and task 1 are described in Annex III to Appendix D. The file RNGDST contains user-specified fractional preferences by range for the engagement of opposing weapons types, given that opposing types do engage. Program RNGDSTGEN converts the integer values in file RNGDST to real values and scales those values to decimal fractions. The program checks whether the sum of fractions across all ranges for each weapon-type-on-weapon-type pairing is within an acceptable tolerance of 1.0. Any sum out of tolerance generates an error message and sets an abort flag. If the abort flag is not set after all checks have been made, program RNGDSTGEN, the range preferences, are then output to a direct access file in the form required by the main program of the Combat Module.

b. The RNGDSTGEN source program is listed in Figure D-VII-3.

```
UNCLASSIFIED*H7AFP(1).RNGDSTGEN(0)
              INCLUDE PARM
              INCLUDE BK1
              INCLUDE BK3
      C
              DEFINE FILE IU2(NR2,IRSZ2,F,NEXT2)
      C
              PARAMETER IPLACE=2
      C
              DIMENSION IRNG(ITYPS1,ITYPS2,IRANGE)
              DIMENSION RRNG(ITYPS1,ITYPS2,IRANGE)
      C
              INTEGER IRNG,COMBO,TPD,THTR,NVIS,NDAY,NPOS
      C
              REAL RRNG
      C
              LOGICAL ERRORS
      C
      2       CONTINUE
      C
              ERRORS=.FALSE.
              READ(IUIN6,100) NDAYS,NTYPS(1),NTYPS(2)
      100     FORMAT()
              READ(5,100) COMBO,TPD,THTR,NVIS,NDAY,NPOS
              CASE=(NVIS-1)*IDAY*IPOS+(NDAY-1)*IPOS+(NPOS-1)
              PREF1=(THTR-1)*ICASE+CASE
              DO 175 J=1,NTYPS(1)
              READ(IUIN2,150) ((IRNG(J,K,L),L=1,IRANGE),K=1,NTYPS(2))
      150     FORMAT(24I3)
      175     CONTINUE
              DO 2000 J=1,NTYPS(1)
              DO 2000 K=1,NTYPS(2)
                SUM=0.0
                DO 1000 L=1,IRANGE
                  TEMP=IRNG(J,K,L)/(10.0**IPLACE)
                  RRNG(J,K,L)=TEMP
                  SUM=SUM+TEMP
      1000      CONTINUE
                IF ( (SUM .GT. 1.0+SMALL2) .OR. (SUM .LT. 1.0-SMALL2) ) THEN
                 WRITE(6,1200) J,K,SUM
      1200        FORMAT(/,1X,10(1H*),3X,'TYPES ',I3,3X,I3,3X,'SUM ',F9.6
           &        ,3X,10(1H*))
                  ERRORS=.TRUE.
                ENDIF
      2000    CONTINUE
              IF ( ERRORS ) CALL FABORT
              DO 3000 J=1,NTYPS(1)
              DO 3000 K=1,NTYPS(2)
                IOPTR2=(J-1)*N21+K
                IOPTR2=IOPTR2+PREF1*ITYPS1*ITYPS2
                WRITE(IU2'IOPTR2,2100,ERR=2500)
           &        (RRNG(J,K,L),L=1,IRANGE)
      2100      FORMAT(500(500F4.2))
                GOTO 3000
      2500      WRITE(6,2600) J,K
      2600      FORMAT(30(1H*),5X,'RNGDST WRITE ERROR',5X,'INDICES= ',
           &        2(I5,5X),30(1H*))
                STOP
      3000    CONTINUE
              STOP
              END
```

Figure D-VII-3. Source Listing of the Program RNGDSTGEN, A Combat Module Preprocessor

c. Input Files. Unit 15 - basedata

Unit 11 - range distribution

d. Runstream Input. String of six 1's separated by blanks.

e. Format of Sequential Range Distribution Input File. Each line contains 24 integers in I3 format.

f. Output File. Unit 21, direct access.

(1) Pointer to Record.

(Side 1 type ID - 1) * ITYPS2 + Side 2 type ID

(2) Record Format. The range distribution for each combination of types as real numbers in F4.2 format.

D-VII-4. PKSGEN

a. Program PKSGEN processes the sequential data file PKS previously created in task 4 of the Combat Module input data preparation. The file PKS and task 4 are described in Annex VI to Appendix D. The formatted file PKS contains SSPKs by range for weapon-type-on-weapon-type engagements. Program PKSGEN generates an unformatted sequential file in the form required by the main program of the Combat Module.

b. The PKSGEN source program is listed in Figure D-VII-4.

c. Input File. Unit 10 - PK file

d. Sequential Input File Format. Each line contains four groups, each group consisting of six integer PKS. The six PKS correspond to the six ranges.

(1) Six PKS for mobility kills.

(2) Six PKS for firepower kills.

(3) Six PKS for catastrophic kills.

(4) Six PKS for M/F kills.

Each line is read using 24F3.3 in the format.

e. The output file is sequential and is written onto Unit 20. Each line contains:

(1) The side of the shooter.

(2) The Side 1 weapon type in the conflict.

(3) The Side 2 weapon type in the conflict.

(4) The six M/F PKS corresponding to the shooting type.

```
              PROGRAM PKSGEN
      C       ****************************************************************
      C       *                                                              *
      C       * PKSGEN: GENERATE AFP PK FILE                                 *
      C       *                                                              *
      C       ****************************************************************
      C
              INCLUDE PARM,LIST
              REAL PK(2,ITYPS,ITYPS,IRANGE), PK1(IRANGE)
              INTEGER TYPE(2)
              LOGICAL ALLZ
              DEFINE FILE IU1(SDF,,IRSZ1,,IRSZ1)
      C
              DO 100 ISIDE = 1,2
                DO 200 IT1 = 1,ITYPS
                  TYPE(ISIDE) = IT1
                  DO 300 IT2 = 1,ITYPS
                    TYPE(3-ISIDE) = IT2
                    READ (IUIN1,301)
             &          (PK(ISIDE,TYPE(1),TYPE(2),I), I=1,IRANGE)
      301           FORMAT (6F3.3)
      300         CONTINUE
      200       CONTINUE
      100     CONTINUE
              DO 400 IT1 = 1,ITYPS
                TYPE(1) = IT1
                DO 500 ISIDE = 1,2
                  DO 600 IT2 = 1,ITYPS
                    TYPE(2) = IT2
                    DO 700 I = 1,IRANGE
                      PK1(I) = PK(ISIDE,TYPE(1),TYPE(2),I)
      700           CONTINUE
                    IF (.NOT. ALLZ(PK1,IRANGE))
             &            WRITE (IU1) ISIDE, TYPE(1), TYPE(2), PK1
      600         CONTINUE
      500       CONTINUE
      400     CONTINUE
              END
```

Figure D-VII-4. Source Listing of the Program PKSGEN, a Combat Module Preprocessor

D-VII-5. PROJGEN

a. Program PROJGEN processes the sequential data file ENGAGE previously created in task 2 of the Combat Module input data preparation. The file ENGAGE and task 2 are described in Annex V to Appendix D. The free-formatted file ENGAGE contains user-specified participation factors and conflict specifications by weapon type on weapon type. A participation factor, p, $0.0 \leq p \leq 1.0$, specifies that only the fraction p of weapons allocated to engage on a day do "reach" the engagements. The fraction (1-p) is not lost but becomes available for allocation on the following day. The participation factor is applied each day. Program PROJGEN scales the real input values and outputs them to a direct access file in the form required by the main program of the Combat Module.

b. The PROJGEN source program is listed in Figure D-VII-5.

```
      INCLUDE PARM
      INCLUDE BK1
      INCLUDE BK3
C
      DEFINE FILE IU6(NR6,IRSZ6,F,NEXT6)
C
      PARAMETER IPLACE=2
C
      DIMENSION PROJ1(ITYPS1,ITYPS2,5)
      DIMENSION PROJ2(ITYPS1,ITYPS2,5)
C
      REAL PROJ2
C
      INTEGER PROJ1,COMBO,TPD,THTR,NVIS,NDAY,NPOS
C
2     CONTINUE
C
      READ(IUIN6,100) NDAYS,NTYPS(1),NTYPS(2)
100   FORMAT()
      READ(5,100) COMBO,TPD,THTR,NVIS,NDAY,NPOS
      CASE=(NVIS-1)*IDAY*IPOS+(NDAY-1)*IPOS+(NPOS-1)
      PREF1=(THTR-1)*ICASE+CASE
C
      DO 1000 I=1,NTYPS(1)
      DO 1000 J=1,NTYPS(2)
        READ(IUIN4,100) (PROJ1(I,J,K),K=1,5)
1000  CONTINUE
      DO 2000 I=1,NTYPS(1)
      DO 2000 J=1,NTYPS(2)
      DO 2000 K=1,5
        PROJ2(I,J,K)=PROJ1(I,J,K)/(10.0**IPLACE)
2000  CONTINUE
      DO 3000 I=1,NTYPS(1)
        IOPTR6=I
        IOPTR6=IOPTR6+PREF1*ITYPS1
        WRITE(IU6'IOPTR6,2100,ERR=2500)
     &      ((PROJ2(I,J,K),K=1,5),J=1,NTYPS(2))
2100    FORMAT(500(500F6.2))
        GOTO 3000
2500    WRITE(6,2600) I
2600    FORMAT(30(1H*),5X,'PROJECT WRITE ERROR',5X,'INDEX= ',
     &      I5,5X,30(1H*))
        STOP
3000  CONTINUE
      STOP
      END
```

Figure D-VII-5. Source Listing of the Program PROJGEN, a Combat Module Preprocessor

c. Input Files. Unit 15 - basedata

Unit 13 - projection/participation

d. Runstream Input. String of six 1's separated by blanks.

e. **Format of Sequential Projection/Participation Input File.** Free field format--all inputs real. For each Side 1 type and Side 2 type:

(1) The participation factor for the Side 1 type.

(2) The participation factor for the Side 2 type.

(3) The conflict duration in minutes.

(4) The number of identical sites of which this battle is typical.

(5) The number of conflicts/day.

f. **Output File.** Unit 25, direct access.

(1) **Pointer to Record.** ID of Side 1 type.

(2) **Record Format.** The five numbers described above, for each Side 2 type, in F6.2 format.

ANNEX VIII TO APPENDIX D

AFP ARTILLERY PREPROCESSOR

Section I. OVERVIEW

D-VIII-1. GENERAL. This annex describes the artillery preprocessor of the AFP Firepower and Counterfirepower Module. The main program of the AFP Combat Module decomposes battle into direct fire engagements between single opposing weapon types. Area weapons (including mortars, artillery, and rockets) may fire on the direct fire engagements. However, because the main program of the Combat Module does not include explicit information about the mix and distribution of weapons within indirect fire areas, a preprocessor sensitive to the geometry and contents of target complexes is applied to generate appropriately adjusted lethalities of indirect fire. The AFP Artillery Preprocessor accepts a variety of special input data and generates so-called "effective fractional damage (EFD)" data for shooter and target weapon pairings. With some important distinctions, an EFD is the artillery analogue of an SSPK for direct fire. The SSPK of a direct fire round is a probability that the round kills a single target. An SSPK must lie on the closed interval 0.0 to 1.0. Logic within the main program of the Combat Module treats SSPKs as probabilities. The EFD of an indirect fire round is the expected damage against possibly several targets of the same type within the lethal area of the round. In principle, an EFD may exceed 1.0; however, computed EFDs are usually much smaller than 1.0. A single indirect fire round may also kill targets of different types. Hence, a single artillery round may have nonzero EFDs against several target types. If there are N target types on the battlefield, a single shot may be characterized by an N-element vector. A single round SSPK vector may have only one nonzero element, and that element must be less than 1.0. A single round EFD vector may have N nonzero elements, and the elements may exceed 1.0. The elements of such an EFD vector clearly depend on characteristics of a round and on the geometry and contents of the target area. In these terms, the artillery preprocessor's purpose is to build EFD vectors.

D-VIII-2. INPUT

a. Artillery preferences for target cluster types (including the type of round fired).

b. Target cluster compositions.

c. Target cluster locations.

d. Artillery characteristics.

e. Artillery target class composition.

f. Range distribution of artillery.

D-VIII-3. OUTPUT. Expected fractional kills per round of AFP targets.

D-VIII-4. PROCESS. The artillery preprocessor models two range bands: near (corresponding to Combat Module bands 1-5) and deep (corresponding to Combat Module range band 6).

a. Compute the modified preference of each indirect fire shooter for each type of target cluster in each of the two range bands (based on the initial preference and the number of clusters of each type).

b. Compute the expected fractional kills (EFK) per round of each class of artillery target (troops, light vehicles, heavy armor, etc., seven classes in all) in each type of target cluster in each location (open, woods, or town), via the Super-Quickie II algorithm. In the case of troops, this computation also involves their posture (standing, prone, or in foxholes) in addition to the data required for the other target classes.

c. Weight the EFKs computed in step b by the fraction of clusters in each location and sum over locations to produce the EFK of each class of target in each type of target cluster.

d. For each shooter, calculate the expected fractional kills according to the formula:

$$efk(i,r) = \sum_{k=1}^{n} mp(k,r) * efk1(k,m) * den(i,k)/class(k,m)$$

where:

n	=	the number of target cluster types
efk(i,r)	=	the expected fractional kills of AFP type i in artillery preprocessor range band r (near or deep)
mp(k,r)	=	the modified preference of this shooter for target clusters of type k in range band r
efk1(k,m)	=	the expected fractional kills of artillery target class m in target cluster type k
m	=	the artillery target class of AFP type i (1 m 7)
den(i,k)	=	the number of type i targets in target cluster type k
class (k,m)	=	the number of targets of artillery target class m in target cluster type k

Section II. INPUT FILES

D-VIII-5. TYPEMAP FILE

a. File Name: Artillery target type-mapping **Unit number:** 5

b. Description: This file maps AFP target numbers to artillery target classes. The first five lines are descriptive information disregarded by the processing program.

c. Used by: APIN1

d. Description of Fields:

Name	Position	Type	Description
SC	1	C	Target side, B or R
ID	2-3	I	AFP identification number
DESCR	10-19	C	Target description
IATCL	24	I	Artillery target class 0-7, with the following meaning: 0 = invulnerable to artillery 1 = troops 2 = light vehicles 3 = heavy armor 4 = light armor 5 = crew-served weapon 6 = light artillery 7 = heavy artillery

D-VIII-6. ARTYPREF FILE

a. File Name: Artillery preference file **Unit number:** 5

b. Description: This file specifies the preference of each artillery shooter for each type of target cluster. It is not necessary that the file include data for every possible shooter (51-60); those not included will be presumed to be "dummies" and will produce no kills. The first five lines of the file are human-readable descriptive information disregarded by the processing program.

c. Used by: APIN1

d. Description of Fields:

Name	Position	Type	Description
SC	1	C	Shooter side, B or R
ID	2-3	I	Shooter identification number, 51-60
DESCR2	8-27	C	Shooter description
TSC	30	C	Target cluster side, B or R
ITCT	31-32	I	Target cluster identification number
P	39-43	R	Preference of this shooter for this type of target complex. The total preferences of a given shooter must sum to 1

Name	Position	Type	Description
IAM	45	I	Type of ammo to be fired by this shooter at this type of target cluster, 0-2 with the following meaning: 0 = ICM 1 = HE 2 = VT

D-VIII-7. ARTYDATA FILE

a. File Name: Artillery data file **Unit number:** 5

b. Description: This file contains the technical characteristics of the artillery and ammunition. One chunk of data requires two lines in the file and corresponds to a specific combination of shooter, ammunition, class of target, and target location. For troop-class targets, their posture (standing, prone, or in foxholes) is also considered.

c. Used by: APIN2

d. Description of Fields:

Name	Position	Type	Description
(First of two lines)			
SC	1	C	Shooter side, B or R
ID	2-3	I	Shooter number, 51-60
WPNNAM	4-23	C	Shooter description
AMODES	24-33	C	Ammunition description
IATCL	34	I	Target class, 1-7
TGTDES	36-45	C	Target description. For troop-class targets, the 7th column of this field specifies their posture: S = standing P = prone F = foxhole
ILOCN	46	I	Target location: 1 = open 2 = woods 3 = town
AMOTYP	47	I	Ammunition type: 0 = ICM 1 = HE 2 = VT
NV	48-49	R	Number of volleys to be fired at the target
AL	50-54	R	Lethal area of a round (sq meters)

Name	Position	Type	Description
LV	55-59	R	Length of volley pattern (meters)
WV	60-64	R	Width of volley pattern (meters)
NR	65-67	R	Number of rounds per volley
RR	68-70	R	Round reliability

(Second of two lines)

Name	Position	Type	Description
REPM	1-5	R	Mean point of impact range error probable (meters)
DEPM	6-10	R	Mean point of impact deflection error probable (meters)
CEPM	11-15	R	Mean point of impact circular error probable (meters)
TLE	16-20	R	Target location error (meters)
REPP	21-25	R	Precision range error probable (meters)
DEPP	25-30	R	Precision deflection error probable (meters)
CEPP	31-35	R	Precision circular error probable (meters)
K	36-38	R	Pattern adjustment factor
OMEGA	39-41	R	Angle of fall, for HE (degrees)
NS	42-44	R	Number of submunitions, for ICM
RS	45-47	R	Reliability of a submunition, for ICM
LSP	48-52	R	Single round submunition pattern length (meters) for ICM
WSP	53-57	R	Single round submunition pattern width (meters) for ICM
RSP	58-62	R	Single round submunition pattern radius (meters) for ICM

D-VIII-8. TGTDENSITY FILE

a. File Name: Target density file **Unit number:** 5

b. Description: This file contains the densities (inventories) of each of the AFP weapon types in each of the artillery target clusters.

c. Used by: APIN1

d. Description of Fields:

Name	Position	Type	Description
SC	1	C	The side of the target cluster, B or R (possibly blank, in which case the line contains only IT and DEN)
JTCT	2-3	I	The identification number of the target cluster

Name	Position	Type	Description
IT	19-20	I	The AFP identification number of the target weapon type
DEN	21-29	R	The density of the target weapon within the target cluster

D-VIII-9. RNGDIST FILE

a. **File Name:** Range distribution **Unit number:** 5

b. **Description:** This file contains the distribution of each of the indirect fire shooters to the two range bands (near and deep) used by the artillery preprocessor. The first line of this file is descriptive information not processed by the program.

c. **Used by:** APIN1

d. **Description of Fields:**

Name	Position	Type	Description
SC	1	C	Shooter side, B or R
IS	2-3	I	Shooter identification number, 51-60
DIST(1)	10-13	R	Fraction of shooters assigned to band 1 (near)
DIST(2)	25-28	R	Fraction of shooters assigned to band 2 (deep)

(Fractions must sum to 1.0)

D-VIII-10. AFPINV FILE

a. **File Name:** AFP inventory file **Unit number:** 5

b. **Description:** This file contains the number of each AFP type. The first nine lines of this file are descriptive information not processed by the program. The file also includes information about the types of weapons mounted on each firing platform, but this information is disregarded by the processing program. Only the lines which contain a nonblank character in column 1 are processed fully; these are the lines which contain the inventory data.

c. **Used by:** APIN1

d. Description of Fields:

Name	Position	Type	Description
SC	1	C	Side, B or R
ID	2-3	I	AFP identification number
DESCR	10-19	C	Description
INVTRY	26-35	I	Inventory (NOTE: Although this field is read with an I type format, blanks are treated as nulls, not zeros, contrary to the usual practice. Consequently, the number need not be right-justified in the field.)

D-VIII-11. TGTDATA FILE

a. File Name: Artillery target data file **Unit number:** 5

b. Description: This file describes the dimensions of artillery target clusters.

c. Used by: APIN1

d. Description of Fields:

Name	Position	Type	Description
SC	1	C	Target cluster side, B or R
ID	2-3	C	Target cluster identification number
DESCR2	5-24	C	Target cluster description
TL	25-30	R	Length of target cluster (meters)
TW	31-36	R	Width of target cluster (meters)

D-VIII-12. TGTLOC FILE

a. File Name: Target location file **Unit number:** 5

b. Description: This file specifies the fraction of target clusters of each type to be found in each of the three types of location: open, woods, and town. It also specifies the total number of clusters of each type. The first nine lines of the file are descriptive header information not processed by the program.

c. Used by: APIN1

d. **Description of Fields:**

Name	Position	Type	Description
SC	1	C	Target cluster side, B or R
ID	2-3	I	Target cluster identification number
DESCR	4-13	C	Target cluster description
FLOC(1)	23-26	R	Fraction located in open
FLOC(2)	29-33	R	Fraction located in woods
FLOC(3)	36-39	R	Fraction located in towns
IPOST	42	I	Posture of troops in each cluster: 1 = standing 2 = prone 3 = foxhole This field may be omitted if the cluster contains no troops
NCMPLX(1)	47-53	R	Number of target clusters of this type in range band 1 (near)
NCMPLX(2)	56-62	R	Number of target clusters of this type in range band 2 (deep)

Section III. PROGRAM ARRAYS AND VARIABLES

D-VIII-13. APDAT BLOCK

a. **Block Name:** APDAT

b. **Used by:** APIN1, APIN2, APOUT, ARTPRE

c. **Block Description:**

Name	Dimension	Type	Description
The following are input data:			
NTCT	2	I	Number of target cluster types, by side
ARTYCL	2,ITYPS	I	Artillery target class, by side and AFP weapon type number (0 if target is invulnerable to artillery)
NTYPS	2	I	Number of AFP weapon types, by side
POSTUR	2,MAXTCT	I	Posture of troops, by side and target cluster type (1 = standing, 2 = prone, 3 = foxhole)
IAMMO	2, LOARTY:HIARTY	I	Type of ammo to be fired, by shooter side and indirect fire weapon number
IOPT	MAXOPT	I	Run control options (currently only one: arty data print control)

PREF	2, LOARTY:HIARTY, MAXTCT	R	The raw (unmodified) target preference, by shooter side, shooter type, and target cluster type
TL	2,MAXTCT	R	Target cluster length (meters), by side and target cluster type
TW	2,MAXTCT	R	Target cluster width (meters), by side and target cluster type
FLOCN	2,MAXTCT, NLOCN	R	Fraction of target clusters in type of location, by side, target cluster type, and location type
NCMPLX	2,MAXTCT, NRANGE	R	Density of target cluster type, by side, target cluster type, and range
CLWPN	2,MAXTCT,ITYPS	R	Composition of target cluster by AFP identification number
RNGDST	2, LOARTY:HIARTY, NRANGE	R	Distribution of shooters to range bands
WPID	2,ITYPS	C*10	Weapon identifier, by side and AFP weapon type number
CLSTID	2,MAXTCT	C*10	Target cluster identifier, by side and target cluster type

This is the end of the input data and the start of computed data:

EFK1	2, LOARTY:HIARTY, MAXTCT,NATCL	R	The expected fractional kill, by shooter side, shooter type, target cluster type, and artillery target class
MPREF	2, LOARTY:HIARTY MAXTCT	R	Modified target preference (taking density of target clusters into account) by shooter side, shooter type, and target cluster type
EFK3	2, LOARTY:HIARTY, ITYPS,NRANGE	R	Expected fractional kill, by shooter side, shooter type, AFP target type number, and range
TDEN	2,MAXTCT, NATCL	R	Target density, by side, target cluster type, and artillery target class

D-VIII-14. ERRDAT BLOCK

a. **Block Name:** ERRDAT

b. **Used by:** APIN1, APIN2, APOUT, ARTPRE, GETAGG, GNERR

c. **Block Description:**

Name	Dimension	Type	Description
NERRS	----	I	Total number of errors detected
NERRSL	----	I	Total number of errors detected on the current input line

Section IV. PROGRAM ROUTINES

D-VIII-15. ARTPRE

a. **Name:** ARTPRE **Type:** Main program

b. **Purpose:** Controls the overall process of the artillery preprocessor.

c. **Arguments:** None

d. **Called by:** None

e. **Calls:** APIN1, APIN2, APOUT, ZERO

f. **Common Blocks Used:** APDAT, ERRDAT

g. **Files Used:** Reads from unit 5.

h. **Local Arrays:** None

i. **Method:** First, read the run control option(s). Invoke APIN1 to read all of the input data except the artillery characteristics, i.e., the artillery target preferences, target cluster dimensions, target cluster locations, target inventories, artillery target class compositions, and artillery range distributions. Calculate the modified artillery target preferences, adjusting the original preferences to take the density of target clusters into account. Call APIN2 to read the artillery characteristic file and compute the initial expected fractional kills (EFKs) of artillery target classes in each target cluster type. Use the modified preferences and the target composition of each cluster to compute the EFK of each AFP type in each of the two range bands. Finally, if no errors have been detected, call APOUT to output the results.

D-VIII-16. APIN1

a. **Name:** APIN1 **Type:** Subroutine

b. **Purpose:** Read and validate first set of artillery preprocessor data (all data but the artillery tube data).

c. **Arguments:** None

d. **Called by:** ARTPRE

e. **Calls:** FLUSH, GETAGG, GNERR, ICOLOR, IZERO, PRTBOX, RANGE, RANGEI, and ZERO

f. **Common Blocks Used:** APDAT, ERRDAT

g. Files Used: Reads from unit 5.

h. Local Arrays:

Name	Dimension	Type	Description
COLOR1	2	C*1	B and R (color abbreviations)
ATCL	NATCL	C*7	Artillery target class names
CLOCN	NLOCN	C*5	Target cluster location names
XLOCN	NLOCN	R	Temporary storage for target cluster fractional locations
XNCMPL	NRANGE	R	Temporary storage for numbers of target clusters
XDIST	NRANGE	R	Temporary storage for range distribution

i. Method: APIN1 first reads the number of AFP weapon types and the number of target cluster types on each side. Next, APIN1 reads the artillery preprocessor input files (excluding the artillery tube data file) in the following order:

(1) The artillery target type-mapping file, which tells the artillery target class of each of the AFP target types.

(2) The artillery preference file, which contains the "raw" (or unmodified) preference of each type of indirect fire weapon for each type of target cluster and the type of ammo fired at the cluster.

(3) The target data file, which contains the dimensions of each target cluster type.

(4) The target density file which contains the densities of each of the AFP weapon types within each of the target clusters.

(5) The target location file, which specifies the fraction of each type of target cluster found in each type of location, the posture of the troops (if any) in each type of cluster, and the number of clusters of each type in each of the two range bands.

(6) The range distribution file which specifies the fraction of each type of indirect fire shooter allocated to each of the two range bands.

Each file is terminated by an end-of-file condition. All data is printed and validated, and any errors detected will result in error messages, but the routine will attempt to keep on processing even in the presence of errors.

D-VIII-17. APIN2

a. Name: APIN2 **Type:** Subroutine

b. Purpose: Read artillery tube data for the artillery preprocessor and calculate expected fractional kills.

c. Arguments: None

d. Called by: ARTPRE

e. Calls: GNERR, ICOLOR, INDWPS, IZERO, LOOKUP, PRTBOX, RANGE, RANGEI, and ZERO

f. Common Blocks Used: APDAT, ERRDAT

g. Files Used: Reads from unit 5.

h. Local Arrays:

Name	Dimension	Type	Description
CPOST	NPOST	C*1	One-letter posture abbreviations
COLOR1	2	C*1	R and B (color abbreviations)
IREC	2, LOARTY:HIARTY, 0:MAXAMO, 2:NATCL,NLOCN	I	Used to detect missing or duplicate data for all target classes except personnel
IRECP	2, LOARTY:HIARTY, 0:MAXAMO,NLOCN, NPOST	I	Used to detect missing or duplicate data for personnel kills

i. Method: Read and validate data from the artillery tube data file. Data from this file may or may not be listed, depending on a run option, IOPT(1). Setting IOPT(1) equal to 0 will suppress the listing of tube data, including the lethal area data, which is classified. This routine reads two records at a time (one set of data requires two input records), validates the data, and checks for a possible duplication of previous data. Records are considered to be duplicate if they refer to the same combination of shooting side, shooter, ammo type, artillery target class, and type of location. For personnel targets, their posture must match as well. If all is well, APIN2 next checks to see if the data is needed to compute EFKs within any of the target clusters. The data is relevant only if it matches the type of ammo shot at that type of cluster, and the cluster includes targets of the artillery target class specified by the tube data. For personnel targets, their posture within the cluster must also match the posture specified by the tube data. If the data is relevant according to these criteria, INDWPS is called to compute the basic EFK against the specified class of targets in the specified cluster. APIN2 weights the basic EFK figure by the fraction of clusters in the specified location and sums the result into the EFK1 array, which is used to accumulate the total EFK for that combination of cluster and target class. This cycle is repeated for each pair of input records until the end of file is encountered. When all the records have been processed, APIN2 checks to see that all the data needed was included in the input, i.e., records were found which matched each required combination of shooting side, shooter, ammo type, artillery target class, and location (and posture as well, for personnel). Any missing data will result in an error message.

D-VIII-18. APOUT

a. **Name:** APOUT **Type:** Subroutine

b. **Purpose:** Output the results of the artillery preprocessor.

c. **Arguments:** None

d. **Called by:** ARTPRE

e. **Calls:** NZRNDX, PRTBOX, PRTMR1, RIGHTJ

f. **Common Blocks Used:** APDAT, ERRDAT

g. **Files Used:** Input - none; Output - 7

h. **Local Arrays:**

Name	Dimension	Type	Description
ATCL	NATCL	C*7	Artillery target class names
COLOR1	2	C*1	B and R (color abbreviations)
ROWIND	2,IEXT	I	Indexes of nonzero matrix rows
CLUSX	2,MAXTCT	I	Indexes of nonzero matrix columns (for cluster types)
WPNX	2,ITYPS	I	Indexes of nonzero matrix columns (for AFP weapon types)
ROWLB1	2,IEXT	C*5	First set of row labels for matrices
CLUSL1	2,MAXTCT	C*5	First set of column labels (for cluster types)
WPNL1	2,MAXTCT	C*5	First set of column labels (for AFP weapon types)
ROWLB2	2,IEXT	C*10	Second set of row labels for matrices
CLUSL2	2,MAXTCT	C*10	Second set of column labels (for cluster types)
WPNL2	2,MAXTCT	C*10	Second set of column labels (for AFP weapon types)
NROWS	2	I	Counts entries used in ROWIND
NCLUSX	2	I	Counts entries used in CLUSX
NWPNX	2	I	Counts entries used in WPNX
INR	IRANGE	I	Maps Combat Module range bands to artillery preprocessor range bands

i. **Method:** APOUT first prints, in columnar form, the nonzero EFKs for each combination of target cluster type and artillery target class. Next, the routine sets up row and column labels for matrices containing the remaining data to be printed: modified preferences for target clusters and EFKs of AFP target types. For each of these two matrices and each of the two preprocessor range bands, APOUT first calls NZRNDX to set up lists of nonzero rows and columns, then calls PRTMR1 to print the matrix, complete with row and column labels and suppression of all-zero rows or columns.

Finally, the routine writes the EFKs of AFP target types to file 7 in a form suitable for inclusion in the AFP Combat Module base data file. Combat Module EFKs for range bands 1 through 5 correspond to the artillery preprocessor EFK for the near range band; those for range band 6, to the far range band.

D-VIII-19. RANGE

a. Name: RANGE **Type:** Real function

b. Purpose: Check a real number for being in range.

c. Arguments:

Name	Dimension	Type	Use	Description
ITEM	----	C*(*)	I	Name of the item being tested
XLO	----	R	I	Lower bound
X	----	R	I	The number being tested
XHI	----	R	I	Upper bound

d. Returns: X, if X is within range; XLO if X is less than XLO; or XHI if X is greater than XHI.

e. Called by: APIN1, APIN2, GETAGG

f. Calls: GNERR

g. Common Blocks Used: None

h. Files Used: None

i. Local Arrays: None

j. Method: Test to see if XLO X XHI. If not, construct an error message saying "(name) SHOULD BE IN THE RANGE (lower) to (higher)" and call GNERR to issue the error message.

D-VIII-20. RANGEI

a. Name: RANGEI **Type:** Integer function

b. Purpose: Check an integer for being in range.

c. Arguments:

Name	Dimension	Type	Use	Description
ITEM	----	C*(*)	I	Name of the item being tested
IXLO	----	I	I	Lower bound
IX	----	I	I	The number being tested
IXHI	----	R	I	Upper bound

d. Returns: IX, if IX is within range; IXLO if IX is less than IXLO; or IXHI if IX is greater than IXHI.

e. Called by: APIN1, APIN2, GETAGG

f. Calls: GNERR

g. Common Blocks Used: None

h. Files Used: None

i. Local Arrays: None

j. Method: Test to see if IXLO IX IXHI. If not, construct an error message saying "(name) SHOULD BE IN THE RANGE (lower) to (higher)" and call GNERR to issue the error message.

D-VIII-21. GNERR

a. Name: GNERR **Type:** Subroutine

b. Purpose: Issue an error message related to input data.

c. Arguments:

Name	Dimension	Type	Use	Description
ITEM	----	C*(*)	I	Name of the item in error (or blank, if the message does not refer to a specific item)
MSG	----	C*(*)	I	The error message

d. Called by: APIN1, APIN2, ICOLOR, RANGE, RANGEI

e. Calls: None

f. Common Blocks Used: ERRDAT

g. Files Used: None

h. Local Arrays: None

i. Method: Print the name of the item in error and the associated message. Increment variables NERRS (the total number of errors detected thus far) and NERRSL (the number of errors on the current input line) in common block ERRDAT. If NERRS exceeds 30, issue a final error message and stop.

D-VIII-22. FLUSH

a. **Name:** FLUSH **Type:** Subroutine

b. **Purpose:** Flush a specified number of input records.

c. **Arguments:**

Name	Dimension	Type	Use	Description
IUNIT	----	I	I	FORTRAN unit number of the file
N	----	I	I	Number of records to flush

d. **Called by:** APIN1

e. **Calls:** None

f. **Common Blocks Used:** None

g. **Files Used:** INPUT - IUNIT (above)

h. **Local Arrays:** None

i. **Method:** Read the specified number of records from the specified file, or up to the end of the file, whichever occurs first.

D-VIII-23. INDWPS

a. **Name:** INDWPS **Type:** Subroutine

b. **Purpose:** Compute expected fractional kill of targets by artillery.

c. **Arguments:**

Name	Dimension	Type	Use	Description
AMOTYP	----	I	I	Ammo type: 0 if ICM, HE otherwise
AL	----	R	I	Lethal area of a round (sq meters)
CEPM	----	R	I	Mean point of impact circular error probable (meters)
DEPM	----	R	I	Mean point of impact deflection error probable (meters)
REPM	----	R	I	Mean point of impact range error probable (meters)
CEPP	----	R	I	Precision circular error probable (meters)
DEPP	----	R	I	Precision deflection error probable (meters)
REPP	----	R	I	Precision range error probable (meters)
K	----	R	I	Pattern adjustment factor (no units)

Name	Dimension	Type	Use	Description
LT	----	R	I	Length of target area (meters)
RT	----	R	I	Radius of target area (meters)
WT	----	R	I	Width of target area (meters)
LV	----	R	I	Length of volley pattern (meters)
WV	----	R	I	Width of volley pattern (meters)
LSP	----	R	I	Single round submunition pattern length for ICM (meters)
RSP	----	R	I	Single round submunition pattern radius for ICM (meters)
WSP	----	R	I	Single round submunition pattern width for ICM (meters)
NR	----	R	I	Number of rounds in each volley (no units)
NS	----	R	I	Number of submunitions in each round for ICM (no units)
NV	----	R	I	Number of volleys (no units)
RR	----	R	I	Reliability of a round (no units)
RS	----	R	I	Reliability of a submunition for ICM (no units)
TD	----	R	I	Target density
TLE	----	R	I	Target location error in circular error probable (meters)
OMEGA	----	R	I	Angle of fall for HE (degrees)
EFK	----	R	O	Expected fractional kill
FD	----	R	O	Fractional damage

d. Called by: APIN2

e. Calls: None

f. Common Blocks Used: None

g. Files Used: None

h. Local Arrays: None

i. Method: This subroutine implements the Super-Quickie II algorithm as described in the publication "Programmable Calculator (TI-58 or TI-59) Manual for Evaluating Effectiveness of Nonnuclear Surface-to-Surface Indirect-Fire Weapons Against Area Targets," dated 2 June 1981, by Lonnie R. Minton, US Army Field Artillery School. Note that the TI-59 program and the flowchart given in that publication differ. This subroutine agrees with the flowchart.

D-VIII-24. PRTBOX

a. Name: PRTBOX **Type:** Subroutine

b. Purpose: Print a character string, framed by a box of asterisks.

c. **Arguments:**

Name	Dimension	Type	Use	Description
ICNTRL	----	I	I	Print control digit (0 or 1)
S	----	C*(*)	I	The character string to print

d. **Called by:** APIN1, APIN2, APOUT

e. **Calls:** None

f. **Common Blocks Used:** None

g. **Files Used:** None

h. **Local Arrays:** None

i. **Method:** Using the supplied ICNTRL for carriage control on the first line, print the character string framed in a box

```
*************
*           *
* LIKE THIS *
*           *
*************
```

D-VIII-25. NZRNDX

a. **Name:** NZRNDX **Type:** Subroutine

b. **Purpose:** Form lists of indexes of nonzero rows and columns in a matrix.

c. **Arguments:**

Name	Dimension	Type	Use	Description
MAT	2,ROWDIM, COLDIM	I	I	The matrix to be scanned
ROWDIM	----	I	I	The row dimension of MAT
COLDIM	----	I	I	The column dimension of MAT
ROWIND	2,ROWDIM	I	O	Lists of indexes of nonzero rows
COLIND	2,COLDIM	I	O	Lists of indexes of nonzero columns
NROWS	2	I	O	Sizes of lists of row indexes
NCOLS	2	I	O	Sizes of lists of column indexes

d. **Called by:** APOUT

e. **Calls:** None

f. **Common Blocks Used:** None

g. **Files Used:** None

h. **Local Arrays:** None

i. **Method:** The first index of MAT typically represents a side (Blue or Red). For a fixed side, say S, consider the slice of MAT consisting of MAT(S,i,j), with i and j varying. NROWS(S) is set equal to the number of nonzero rows in this slice. (A row is nonzero if it contains at least one nonzero entry.) ROWIND(S,1) through ROWIND(S,NROWS(S)) contain the indexes of the nonzero rows. Similarly, NCOLS(S) is set to the number of nonzero columns in the slice, and COLIND(S,1) through COLIND(S,NCOLS(S)) contain the indexes of the nonzero columns in the slice.

D-VIII-26. PRTMR1

a. **Name:** PRTMR1 **Type:** Subroutine

b. **Purpose:** Print a matrix.

c. **Arguments:**

Name	Dimension	Type	Use	Description
TITLE	----	C*(*)	I	A title to be printed above the matrix
MAT	2,NDIM1, NDIM2	I	I	The matrix to be printed
NDIM1	----	I	I	The second dimension of MAT
NDIM2	----	I	I	The third dimension of MAT
ROWIND	2,NDIM1	I	I	Lists of indexes of rows to be printed
COLIND	2,NDIM2	I	I	Lists of indexes of columns to be printed
NROWS	2	I	I	Sizes of lists in ROWIND
NCOLS	2	I	I	Sizes of lists in COLIND
ROWLB1	2,NDIM1	C*5	I	First set of row labels
ROWLB2	2,NDIM1	C*10	I	Second set of row labels
COLLB1	2,NDIM2	C*5	I	First set of column labels
COLLB2	2,NDIM2	C*10	I	Second set of column labels

d. **Called by:** APOUT

e. **Calls:** None

f. **Common Blocks Used:** None

g. **Files Used:** None

h. Local Arrays:

Name	Dimension	Type	Description
CELL	10	C*10	Used to hold contents of one line of print

i. Method: Consider MAT as consisting of two slices, MAT(1,*,*) and MAT(2,*,*). Each slice is considered to be a two-dimensional array and is printed separately. For the slice with first dimension S, i.e., MAT(S,*,*), only the rows listed in ROWIND(S,*) and the columns listed in COLIND(S,*) will be printed. Typically, these are the nonzero rows and columns listed by calling NZRNDX before calling PRTMR1. Ten columns per line are printed, framed by the row and column labels (both sets), and zero entries are printed as '.' rather than '0' to increase the readability of the printout. This is accomplished by setting up the contents of each line in the character array CELL before printing, with each cell containing one number in character form if nonzero or '.' if zero.

D-VIII-27. RIGHTJ

a. Name: RIGHTJ **Type:** Subroutine

b. Purpose: Right justify a character string.

c. Arguments:

Name	Dimension	Type	Use	Description
S1	----	C*(*)	I	The input character string
S2	----	C*(*)	I	The output character string

d. Called by: APOUT

e. Calls: None

f. Common Blocks Used: None

g. Files Used: None

h. Local Arrays: None

i. Method: Move S1 to S2, right-justified within S2 and padded on the left with blanks, if necessary.

D-VIII-28. ZERO

a. Name: ZERO **Type:** Subroutine

b. Purpose: Set all the elements of a real matrix to 0.

c. **Arguments:**

Name	Dimension	Type	Use	Description
A	N	R	O	The matrix to be zeroed
N	----	I	I	The dimension of A

d. **Called by:** APIN1, APIN2, ARTPRE

e. **Calls:** None

f. **Common Blocks Used:** None

g. **Files Used:** None

h. **Local Arrays:** None

i. **Method:** Set A(1) through A(N) equal to 0. This routine can also be used to zero a multidimensional matrix A by passing N as the product of the dimensions of A. This is a little tricky and doubtless reprehensible, but extremely convenient.

D-VIII-29. IZERO

a. **Name:** IZERO **Type:** Subroutine

b. **Purpose:** Set all the elements of an integer matrix to 0.

c. **Arguments:**

Name	Dimension	Type	Use	Description
IA	N	I	O	The matrix to be zeroed
N	----	I	I	The dimension of IA

d. **Called by:** APIN1, APIN2

e. **Calls:** None

f. **Common Blocks Used:** None

g. **Files Used:** None

h. **Local Arrays:** None

i. **Method:** Set IA(1) through IA(N) equal to 0. This routine can also be used to zero a multidimensional matrix IA by passing N as the product of the dimensions of IA. This is a little tricky and doubtless reprehensible, but extremely convenient.

D-VIII-30. ICOLOR

a. **Name:** ICOLOR **Type:** Subroutine

b. **Purpose:** Convert a color (B or R) to a side index (1 or 2).

c. **Arguments:**

Name	Dimension	Type	Use	Description
ID	----	C*(*)	I	Identifier to be used for any error message
CH	----	C*1	I	Color

d. **Returns:** 1 if B, 2 if R, 1 otherwise.

e. **Called by:** APIN1, APIN2

f. **Calls:** GNERR

g. **Common Blocks Used:** None

h. **Files Used:** None

i. **Local Arrays:** None

j. **Method:** Set ICOLOR accordingly if CH is B or R. If CH is neither B nor R, force ICOLOR to 1 and issue an error message via GNERR.

D-VIII-31. LOOKUP

a. **Name:** LOOKUP **Type:** Integer function

b. **Purpose:** Look up a character string in a table.

c. **Arguments:**

Name	Dimension	Type	Use	Description
S	----	C*(*)	I	The character string to be look up
STAB	(*)	C*(*)	I	The table to be searched
N	----	I	I	The number of entries to be searched

d. **Called by:** APIN2

e. **Calls:** None

f. **Common Blocks Used:** None

g. **Files Used:** None

h. Local Arrays: None

i. Method: Search STAB(1) to STAB(N) for a match to S. The value of the function returned is the index of the first match or, if no match is found, 0.

D-VIII-32. GETAGG

a. Name: GETAGG **Type:** Subroutine

b. Purpose: Get weapon types to be aggregated.

c. Arguments:

Name	Dimension	Type	Use	Description
AGGFAC	----	R	O	The aggregation factor
ISAGG	----	L	O	ISAGG(i,j) is .true. if weapon type j on side i is aggregated, .false. otherwise

d. Called by: APIN1

e. Calls: RANGE, RANGEI

f. Common Blocks Used: ERRDAT

g. Files Used: Reads from unit 5.

h. Local Arrays:

Name	Dimension	Type	Description
NGRPS	2	I	The number of aggregated types, by side
NAGG	2,ITYPS	I	List of aggregated types on each side
COLOR	2	C*4	Blue and Red

i. Method: Read data specifying the aggregated types, validate the data, and construct the logical array ISAGG accordingly. Any errors detected will result in execution of a STOP instruction. The data is read with free-format reads, in the following sequence:

(1) Aggregation factor, number of aggregated types on the Blue side, and number of aggregated types on the Red side.

(2) List of Blue aggregated types.

(3) List of Red aggregated types.

All data read is printed out for perusal by the user.

Section V. ADVICE ON ARTILLERY SETUP

D-VIII-33. BACKGROUND. The AFP scheme for decomposing the battlefield into direct fire duels between pure weapon types, subject to incoming indirect fire, requires special treatment of indirect fire weapons. Indirect fire weapons must be able to fire at direct fire weapons and at one another. AFP development took the unusual step of making the counterbattery role a variation of direct fire. Counterbattery weapons can kill one another but nothing else. In early AFP application, some weapon types were assigned almost entirely to the counterbattery role, thereby, precluding them from killing other types of targets. Weapons not assigned to the counterbattery role were available for indirect fire upon the direct fire duels (including the counterbattery versions of direct fire). Indirect firers kill any targets upon which they fire, but the indirect firers cannot be killed by any weapon. In that early AFP production, mortars, artillery, and rockets killed and lost relatively little. Review of input, logic, and output led to the conclusion that the AFP battlefield decomposition scheme had been carried too far in several respects.

a. The treatment of counterbattery fire had prevented kills of collateral targets.

b. The treatment of noncounterbattery fire had been based on underestimation of the presence of mortars, artillery, and rocket launchers in general target complexes.

c. Overall, too little fire was falling on too few targets with too little effect.

D-VIII-34. REMEDY. System tests showed that mortar, artillery, and rocket could be better approximated simply by a revised approach to input data setup.

a. Increase Firing

(1) Change: Set Additionals file entries to be 1.0 for all weapons.

(2) Rationale: The value in the Additionals file reflects the percentage of "indirect fire" that a weapon system will use as opposed to "counterbattery fire." In AFP, "counterbattery fire" is reflected by direct fire engagements between indirect weapon systems. This does not reflect the total killing power of the artillery/mortars because one does not normally shoot at just the indirect fire weapon system but other vehicles and personnel as well. When the Additionals value is set at 1.0, all firing will be in the "indirect fire" mode where firing is done against target clusters (area targets). When one round is fired at a "cluster," it may kill anything in the cluster as set up in the target cluster description file. This more accurately reflects the counterbattery/ countermortar effects. The percent of counterbattery fire can be portrayed by giving a weapon system a preference against an artillery cluster. This new portrayal also eliminates many confusing cross-checks that needed to be conducted.

(3) Previously: Because 8" and MLRS had high "counterbattery" preferences (40 percent and 90 percent respectively), they were not getting the correct number of kills. Note: With this new change, no allocations will show up for the indirect fire weapon systems, but one must ensure allocations are made between deep generic targets (BTRP and RTRP Deep, LARM, HARM, etc.) so that engagements occur.

b. Enrich Target Clusters

(1) Change: Add generic targets for the close range bands.

(2) Rationale: Target clusters that are fired on in the near range bands do not include all targets that may be killed by indirect fire. By having generic targets such as BTRP near, LVEH near, etc., this problem should be corrected.

(3) Previously: Only direct fire weapon systems were clustered.

c. Attack Air Defense Weapons

(1) Change: Increase preference of artillery for Enemy Air Defense.

(2) Rationale: Doctrinally, 8" and MLRS should have a substantial portion of their fires directed for anti-ADA missions. This may simultaneously increase their CIP since ADA systems have a higher target value if they can be killed.

(3) Previously: MICAF I had zero preference for ADA targets.

APPENDIX E

THE AFP COMBAT SUPPORT/COMBAT SERVICE SUPPORT MODULE AND ITS PREPROCESSORS

E-1. OVERVIEW. The Combat Support/Combat Service Support (CS/CSS) modulation process is designed to provide adjustments to the AFP Combat Module results to account for the impacts of eight CS/CSS functions. The Combat Module produces estimates of the kills and losses achieved and suffered by each opposing side. Those kills and losses are recorded for each type weapon versus each type opposing weapon. The Combat Module represents combat at so-called normed levels of CS/CSS support. A normed level of support corresponds most closely to standard doctrine with its implied requirements. For many reasons one or both sides may be unable to conduct combat operations at normed levels. These reasons range from deliberate decisions to provide more or less than doctrinal levels, through temporary imbalances associated with modernization and reorganization time leads and lags, to outright interference by an opponent's equipment and actions.

a. Detailed examples with sample data requirements and record formats are presented in Annex I to this appendix.

b. The relation of the AFP CS/CSS processes to the AFP System in general is portrayed in Figure E-1; the CS/CSS processes are highlighted by being enclosed within a heavy line.

c. Essentially there are two AFP CS/CSS Preprocessors depicted in Figure E-1. The first is called the Search Preprocessor; its function is to extract data from selected inventory files based on two keys--unit identification (TPSNA) and fiscal year--and to develop pertinent equipment and manpower performance factors. The second preprocessor is the Main AFP CS/CSS Preprocessor and is described further in the succeeding paragraphs.

d. The AFP approach to CS/CSS was developed in accordance with the following guidelines.

(1) Combat Support

(a) Support potential must reflect any disparity of forces--should be based on comparisons between US and its adversaries.

(b) Combat support units must be employed in their normal roles.

(c) Combat support factors must incorporate combat support equipment characteristics.

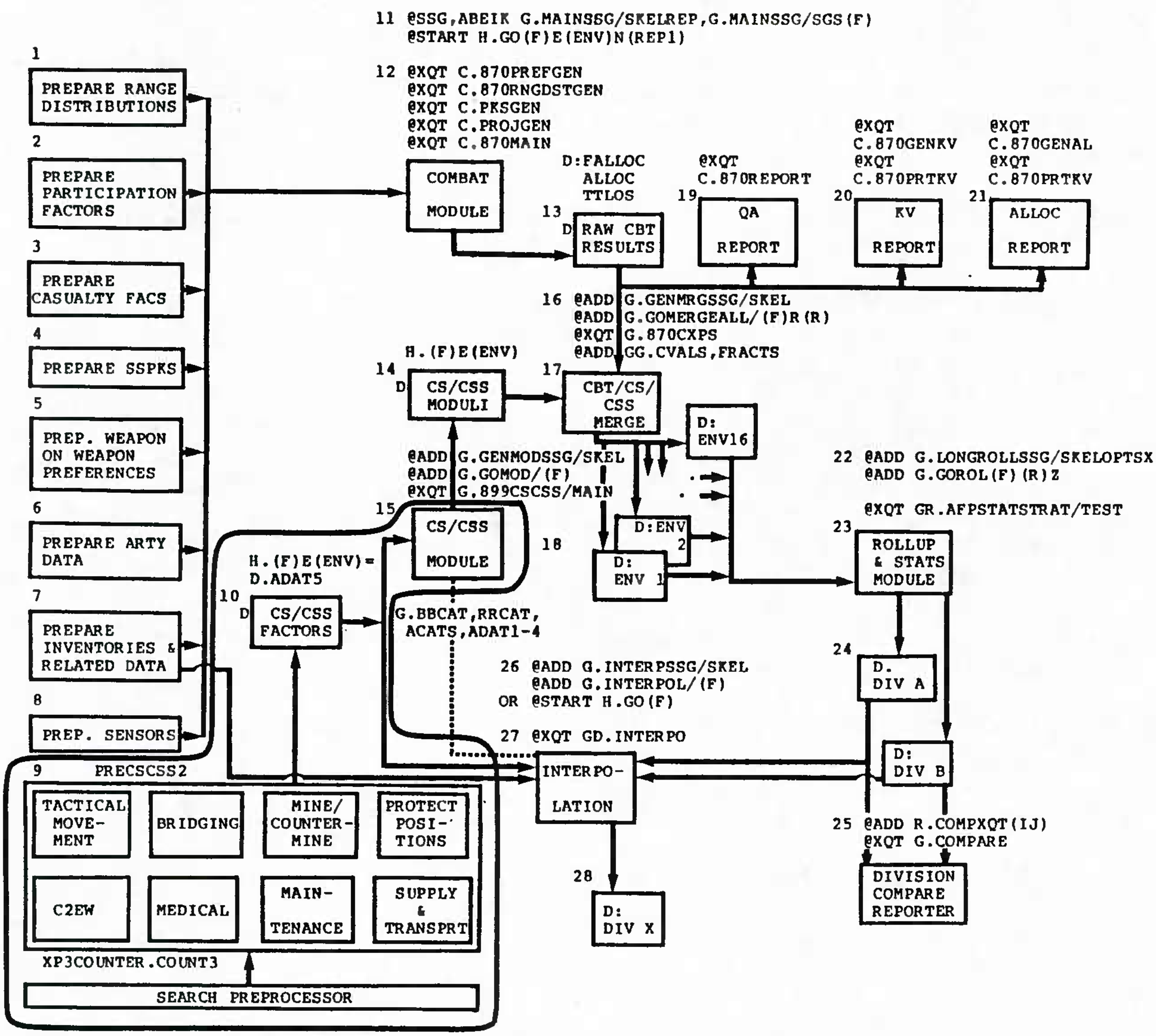


Figure E-1. The Relation of the AFP CS/CSS Preprocessors and Module to the Analysis of Force Potential (AFP) System in General

(2) **Combat Service Support**

(a) Emphasis must be on division level support--DISCOM or equivalent.

(b) Support factors must be relative to intense combat that stresses the support system and its surge capabilities.

(c) Combat service support factors must incorporate CSS equipment characteristics.

E-2. FUNCTIONAL DEFINITIONS. In general, a CS/CSS function may be represented by a measure and a countermeasure for both Blue and Red sides. For example, mine laying and mine clearing are a natural measure/countermeasure pair. Some functions possess measures without corresponding countermeasures. For example, maintenance is represented by a measure for each side but is not regarded as requiring or permitting a direct countermeasure. The eight CS/CSS functions represented within the AFP CS/CSS Module are defined as follows. (The current AFP system preserves space for a ninth, unused CS/CSS function.)

a. **Tactical Mobility**

(1) **Measure.** The measure is a weighted average (weighted with respect to inventory and to on-road or off-road travel) of the speeds of a side's ground combat systems relative to the average weighted speed of the base case ground combat systems. Similarly there is a relative weighted measure for air combat systems.

(2) **Countermeasure.** The countermeasure for a side is its divisional engineer capability to create obstacles (not including minefields) relative to the enemy capability to clear obstacles (a planning or doctrinal standard); e.g., the Blue standard was extracted from the Combat to Support Balance Study (CSBS).

b. **Bridging**

(1) **Measure.** The computation of the bridging measure is based on the force's inventory of bridges, rafts, and vehicles, and on a stylized series of gaps of specified widths which the force must cross. Three factors are considered in the computation: the fraction of the series of gaps which can be bridged; the fraction of the vehicles in the force which can cross each gap, based on their load classes; and the time required for those vehicles to cross the gaps, including the time to assemble and disassemble the bridges and rafts used.

(2) **Countermeasure.** No bridging countermeasures are represented within AFP.

c. Mine/Countermine

(1) Measure. The measure for a side is its divisional engineer capability to create minefields relative to enemy capability to clear mines.

(2) Countermeasure. A side's countermeasure is the clearing rate achievable with divisional assets relative to the opponent's mine laying rate. Consideration is given to doctrinal breaching approaches; i.e., complete breaching success may be achieved "in lanes" without clearing an entire minefield.

d. Protective Positions

(1) Measure. A side's measure is the ratio of its ability to construct protective positions for combat systems within five hours to the number of positions required.

(2) Countermeasure. Countermeasures are not represented within AFP.

e. C2EW

(1) Essentially a side's C2EW measure and its opposing countermeasure are combined because the measure for a side is computed using related countermeasure data; i.e., a side's command and control performance is degraded by opposing electronic jammers. The measure for a side is 1.0 minus the fractional reduction in combat effectiveness attributable to diminished effectiveness of command and control caused by active jamming. Three TRADOC Systems Analysis Activity (TRASANA) studies provided the rationale to relate a side's combat effectiveness and the performance of command and control equipment such as multichannel radio relay equipment, VHF and HF radios. Both sides are assumed to have deployed their assets in doctrinal fashion. Consideration is given to radio/radar and jammer equipment densities, frequency bands, and effectiveness.

f. Medical. The measure for a side is 1.0 less the fractional reduction of divisional strength attributable to shortfall in the capacity of the divisional medical battalion. Higher echelon medical facilities are assumed to perform at their designed capacities. The generation of casualties is assumed to occur at intense combat levels. (No countermeasure is represented.)

g. Maintenance. The measure for a side is 1.0 less the fractional reduction in divisional equipment availability attributable to manhour shortfalls in divisional maintenance battalion teams and shops. (No countermeasure is represented.)

h. Supply and Transportation. The measure for a side is 1.0 less any fractional shortfall in the divisional supply and transportation battalions' capability to lift daily supply requirements. Higher echelon supply and transportation units are assumed to perform at their designed levels. (No countermeasure is represented.)

E-3. CS/CSS Symbols. As appropriate to the Ith function, measures and countermeasures are developed and assigned in accordance with the following AFP symbolism:

a. **U(I) Blue countermeasure.**

b. **V(I) Blue measure.**

c. **S(I) Red countermeasure.**

d. **T(I) Red measure.**

E-4. ORGANIZATION OF APPENDIX. The development of measures and countermeasures, as and if appropriate to each CS/CSS function, is described in Annex I to Appendix E. There the Main CS/CSS and Search preprocessor programs are described. The current AFP Combat Module may represent type-on-type engagements between weapons of up to 60 types on each side. Whether or not CS/CSS measures and countermeasures apply to specific type-on-type engagements is determined from sets of tables. The tables serve as screens by weapon type, weapon category, function, measure/countermeasure, and combat environment. If a type-on-type engagement "passes" all the screens, the corresponding U, V, S, or T value is applied. And depending on combat environment, a weight is applied to each function in rolling up all functions. All these steps are performed in the AFP CS/CSS Module proper. That module outputs the rolled-up, weighted values across all CS/CSS functions as the so-called CS/CSS "moduli." The CS/CSS Module proper is described in Annex II to Appendix E.

ANNEX I TO APPENDIX E

THE AFP COMBAT SUPPORT/COMBAT SERVICE SUPPORT (CS/CSS) SEARCH AND MAIN PREPROCESSORS FOR DETERMINING MEASURES AND COUNTERMEASURES BY CS/CSS FUNCTION

Section I. OVERVIEW

E-I-1. This annex describes the development of Blue and Red measures and countermeasures for each of the eight active CS/CSS functions. (AFP structure reserves space for a ninth CS/CSS function, but that space is not currently used.) The product of the Main Preprocessor is a small file containing 2 (sides) x 2 (measure and countermeasure) x 8 (functions) = 32 significant numerical values. (The file also contains four 1.0-values for the unused function.) That file is one of the principal inputs to the CS/CSS Module proper which "rolls up" the values in accord with a complex of switches and weights; this process constructs Blue and Red CS/CSS moduli for each possible pairing of Blue and Red weapon types. The CS/CSS Module proper is described in Annex II to Appendix E.

E-I-2. It is important to note that anything and everything about CS/CSS equipment and manpower characteristics, requirements, and capabilities relative to a specific combat environment is compressed into the 32-value file generated by the CS/CSS Main Preprocessor. And because, by definition, some of those values are always 1.0, the range of CS/CSS variability must be expressed in fewer than 32 numerical values.

E-I-3. Toward accomplishment of the generation of CS/CSS factors, the two AFP CS/CSS preprocessors are designed to:

a. Read all necessary input data from a single automatically prepared file* comprised of data which represent both authorizations and on-hand strengths of personnel and equipment.

---

*This file is prepared from an interactive program called the Search Preprocessor; this program requires the user to identify the organizations (by TPSNA) and years to be analyzed by AFP. Two files will be used by the Search Preprocessor to develop the input data for the CS/CSS Main Preprocessor:

(1) An MOS file depicting the on-hand and authorized strengths for several maintenance related specialties.

(2) An item inventory file of US equipment (weapon system) quantities by fiscal year.

b. Perform a variety of arithmetic and logical operations for each CS/CSS function.

c. Generate a single file containing Blue and Red factors for measures and countermeasures (some of which may be identically 1.0) for each CS/CSS function.

E-I-4. The relation of the AFP CS/CSS Preprocessor to the AFP System in general is shown in Figure E-I-1.

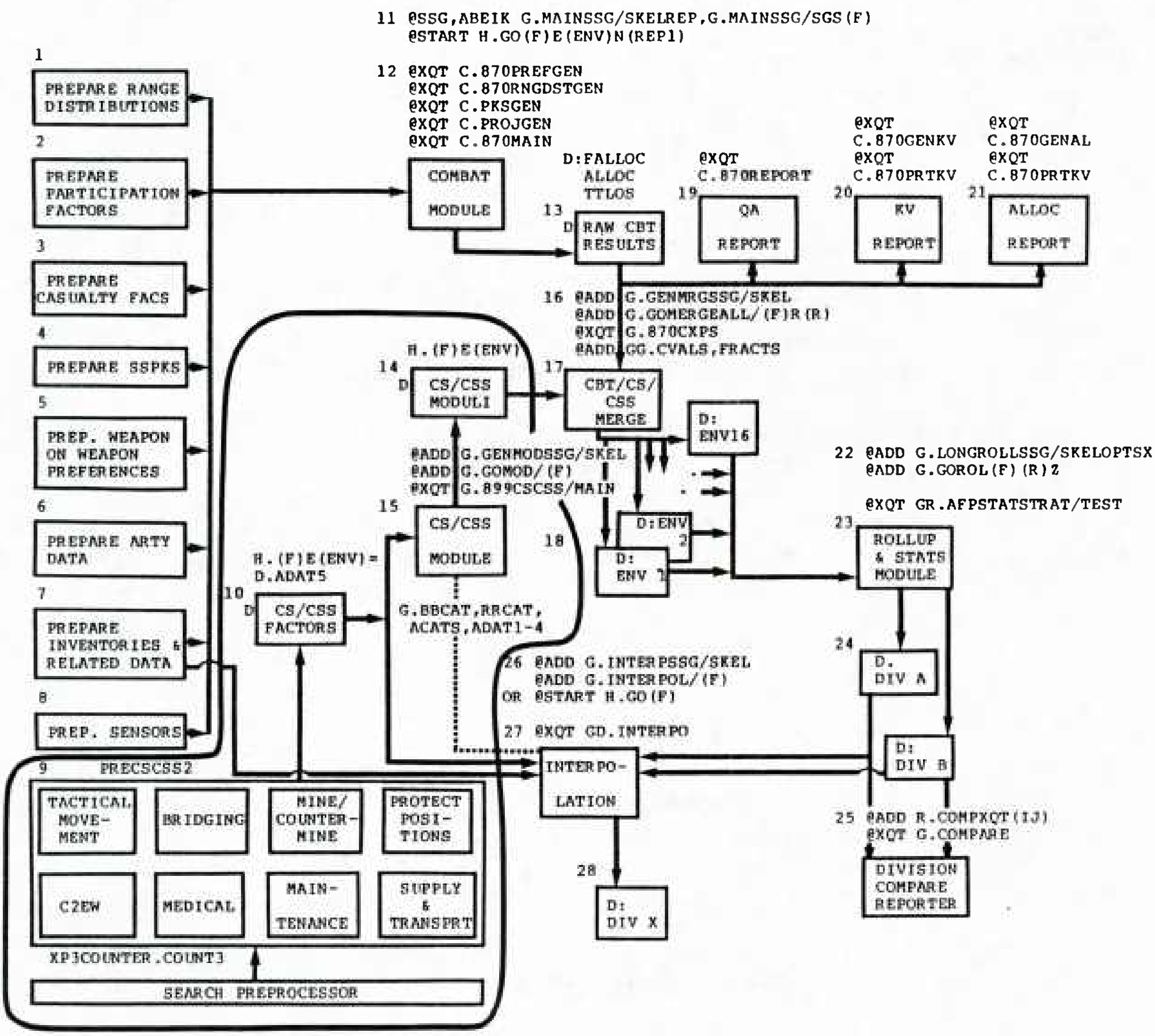


Figure E-I-1. **Relation of the AFP Combat Support/Combat Service Support Preprocessors to the AFP System in General**

Section II. INPUT

E-I-5. The following paragraphs describe the input to the AFP CS/CSS Preprocessor; note that all input data are read by FORTRAN free-format read statements:

a. Tactical Mobility. Table E-I-1 describes the tactical mobility input. Figure E-I-2 displays sample input records for generating tactical mobility factors.

Table E-I-1. Tactical Mobility Input (page 1 of 3 pages)

Field	Variable	Data description
(1st record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input where: T = use, F = read data and compute
2	FACT(1,1)	Value of Blue countermeasure, U(1)
3	FACT(1,2)	Value of Blue measure, V(1)
(if JUMP = .FALSE., next record, else GOTO next record **)		
(next record)		
1	SPDX(1)	Blue ground vehicles average speed (kph)
2	SPDX(2)	Blue aircraft average speed (kph)
3	SPDWT(2)	Ratio of average speed (two base case years) of Blue aircraft to Blue ground vehicles
(next record)		
1	NVEH	Number of Blue vehicle or Blue aircraft types
(next NVEH records)		

Table E-I-1. Tactical Mobility Input
(page 2 of 3 pages)

Field	Variable	Data description
1	DENSTY( )	Population of Blue vehicles/aircraft; subscript 1 = ground, 2 = air
2	SPEED( )	Rate of movement (kph) of Blue vehicle/aircraft types; subscript 1 = ground, 2 = air
(next record)		
1	REQEHO	Blue obstacle clearing capability (cubic meters/hour)
(next record)		
1	CAPEHO	Blue obstacle creation capability (cubic meters/hour)
(** next record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input where: T = use, F = read data
2	FACT(1,3)	Value of Red countermeasure, S(1)
3	FACT(1,4)	Value of Red measure, T(1)
(if JUMP = .FALSE., next record, else GOTO next **)		
(next record)		
1	SPDX(1)	Red ground vehicles average speed (kph)
2	SPDX(2)	Red aircraft average speed (kph)
3	SPDWT(2)	Ratio of average speed of Red aircraft to Red ground vehicles
(next record)		

Table E-I-1. Tactical Mobility Input
(page 3 of 3 pages)

Field	Variable	Data description
1	NVEH	Number of Red vehicles or Red aircraft types
(next NVEH records)		
1	DENSTY( )	Population of Red vehicles/aircraft; subscript 1 = ground, 2 = air
2	SPEED( )	Rate of movement (kph) of Red vehicle/aircraft types; subscript 1 = ground, 2 = air
(next record)		
1	REQEHO	Red obstacle clearing capability (cubic meters/hour)
(next record)		
1	CAPEHO	Red obstacle creation capability (cubic meters/hour)
(**)		

```
F          1.0        1.0
 62.86     219.1      3.53
4
841        68
328        48
21         61
55         56
3
42         227
25         204
52         220
3426
2311
F          1.0        1.0
 62.86     219.1      3.94
8
33         85
330        80
148        75
88         70
8          62.5
362        60
170        50
36         44
2
6          215
6          315
6031
8482
```

Figure E-I-2. Example of Tactical Mobility Data Input to the AFP CS/CSS Main Preprocessor

b. Bridging. Table E-I-2 describes the bridging input. Figure E-I-3 displays sample input records for generating bridging factors.

Table E-I-2. Bridging Input (page 1 of 3 pages)

Field	Variable	Data description
(1st record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(2,1)	Value of Blue countermeasure, U(2)
3	FACT(2,2)	Value of Blue measure, V(2)
(if JUMP = .FALSE., next record, else GOTO next record **)		
(next record)		
1	NCPD	Number of crossing points desired
2	TDFACT	Take-down factor (when multiplied by set-up time, yields take-down time)
(next record)		
1	BASEF1	Value of capability factor F1 in base case
2	BASEF2	Value of capability factor F2 in base case
3	BASEF3	Value of capability factor F3 in base case
(next record)		
1	NGAPS	Number of different types of gap
(next NGAPS records)		

Table E-I-2. Bridging Input
(page 2 of 3 pages)

Field	Variable	Data description
1	NGP	Frequency of occurance of this type gap
2	WGAP	Width of this type gap (meters)
(next record)		
1	NVEH	Number of different combat vehicle types
(next NVEH records)		
1	NV	Number of vehicles of this type
2	VCLAS	Load class of this type vehicle
3	VNUM	AFP identification number of type vehicle
4	VDESC	Description of this type vehicle (NOTE: This is character type data and must be enclosed in single quotation marks.)
(next record)		
1	NBRIDG	Number of different types of bridges/rafts
(next records)		
(NOTE: There will be NBRIDG groups of records consisting of (NGAPS+1) records per group.)		
1	BSETS	Number of bridge/raft sets available
2	BNUM	AFP identification number of bridge/raft
3	BDESC	Description of bridge/raft (character data)
(next NGAPS records)		

Table E-I-2. Bridging Input
(page 3 of 3 pages)

Field	Variable	Data description
1	CPSET	Number of crossing points per set for this type of gap
2	SETUP	Set-up time (minutes)
3	VCRATE	Vehicle crossing rate (vehicles/hour)
4	BCLAS	Bridge/raft load class
(** next record)		
1	JUMP	Switch for whether to use given values in record or compute values following input, T = use, F = read data
2	FACT(2,3)	Value of Red countermeasure, S(2)
3	FACT(2,4)	Value of Red measure, T(2)
(if JUMP = .FALSE., next record, else GOTO next **)		
NOTE: Currently, the Red bridging measures and countermeasures are computed off-line; therefore, input data coding for Red bridging is limited to the three preceding fields.		
(**)		

c. **Mine/Countermine.** Table E-I-3 describes the mine/countermine input. Figure E-I-4 displays sample input records for generating mine/countermine factors.

```
@XQT *98AFP.870BRGPPE
3 2.0
1.0 0.005584 1.0
4
1 15
1 60
1 200
1 400
23
 54 6 51 '81MM MORTAR'
460 11 7 'M113'
159 12 15 'M113+TOW'
0 12 0 'M577A1'
144 12 49 'UC3V'
24 12 27 'VULCAN'
24 13 32 'CHAP1'
0 16 21 'M551'
53 22 53 '4.2MSP'
0 24 12 'ITV TOWII'
54 24 56 '155SPMID9'
12 24 57 '8IN/203'
0 46 17 'CFV'
0 46 58 'MLRS'
0 50 16 'IFV'
 0 52 19 'M48A5'
206 53 24 'M60A1'
0 55 22 'M60A3'
0 55 25 'M60A2'
8 57 46 'CEV'
0 62 20 'M1E1'
0 59 26 'DIVAD'
108 59 23 'M1'
3
1 320 'MAB'
8 40 200 62
3.2 40 200 62
4 10 4.5 60
4 10 3 60
16 321 'AVLB'
1 5 550 60
0 0 0 0
0 0 0 0
0 0 0 0
2 323 'LTRB'
2 30 200 16
1 25 40 16
1 25 4.5 16
1 25 3 16
```

Figure E-I-3. Example of Bridging Data Input to the AFP CS/CSS Main Preprocessor

Table E-I-3. Mine/Countermine Input
(page 1 of 2 pages)

Field	Variable	Data description
(1st record)		
1	JUMP	Switch indicates whether to use given values in record or compute values from following input, where T = use, F = read data
2	FACT(3,1)	Value of Blue countermeasure, U(3) NOTE: U(3) will = 1.00
3	FACT(3,2)	Value of Blue measure, V(3)
(if JUMP = .FALSE., next record, else GOTO next record **)		
(next record)		
1	EQMN	Population of Blue mine laying equipment items
2	PRSMN	Population of Blue mine laying personnel, grade E-7 and below
3	LANMN	Number of lanes through Red minefield along Red division front
4	AWLMN	Average width of Red minefield lane (m)
5	ADFRT	Average Red division frontage (m)
(next record)		
1	CLRMN	Red mine clearing capability (mines/hr)
(** next record)		
1	JUMP	Switch inndicates whether to use given values in record or compute values from following input, where T = use, F = read data
2	FACT(3,3)	Value of Red countermeasure, S(3) Note: S(3) will = 1.00

Table E-I-3. Mine/Countermine Input
(page 1 of 2 pages)

Field	Variable	Data description
3	FACT(3,4)	Value of Red measure, T(3)
(if JUMP = .FALSE., next record, else GOTO next **)		
(next record)		
1	EQMN	Population of Red mine laying equipment items
2	PRSMN	Population of Red mine laying personnel, grade E-7 and below
3	LANMN	Number of lanes through Blue minefield along Blue division front
4	AWLMN	Average width of Blue minefield lane (m)
5	ADFRT	Average Blue division frontage (m)
(next record)		
1	EQCLM	Blue mine clearing capability (mines/hr)
(**)		

```
F           1.0         1.0
3000         285         28           3.736        12500
573.14
F           1.0         1.0
4800        0           12           8            30000
24
```

Figure E-I-4. Example of Mine/Countermine Data Input to the AFP CS/CSS Main Preprocessor

d. Protective Positions. Table E-I-4 describes the protective position data. Figure E-I-5 displays sample input records for generating protective position factors.

Table E-I-4. Protective Positions Input
(page 1 of 2 pages)

Field	Variable	Data description
(1st record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(4,1)	Value of Blue countermeasure, U(4)
3	FACT(4,2)	Value of Blue measure, V(4)
(if JUMP = .FALSE., next record, else GOTO next record **)		
(next record)		
1	BNPP	Number of Blue equipment types used to prepare protective positions
(next BNPP records)		
1	BDENPP	Population of Blue equipment of type
2	BRATEP	Rate of position preparation (per hour) per equipment of type
(** next record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(4,3)	Value of Red countermeasure, S(4)
3	FACT(4,4)	Value of Red measure, T(4)
(if JUMP = .FALSE., next record, else GOTO next **)		
(next record)		

Table E-I-4. Protective Positions Input
(page 2 of 2 pages)

Field	Variable	Data description
1	RNPP	Number of Red equipment types used to prepare protective positions
(next RNPP records)		
1	RDENPP	Population of Red equipment of type
2	RRATEP	Rate of position preparation (per hour) per equipment of type
(**)		

```
F          1.0        1.0
2
 11         2.00
 7         1.78
F          1.0        1.0
6
8          7.47
12         3.88
12         1.91
2          2.33
12         2.95
2          1.40
```

Figure E-I-5. Example of Protective Position Data Input to the AFP CS/CSS Main Preprocessor

e. **C2EW.** Table E-I-5 describes the C2EW input. Figure E-I-6 displays sample input records for generating C2EW factors.

Table E-I-5. C2EW Input
(page 1 of 4 pages)

Field	Variable	Data description
(1st record)		
1	JUMP	Switch indicates whether to use given values in record or compute values from following input, where T = use, F = read data
2	FACT(6,1)	Value of Blue countermeasure, U(6)
3	FACT(6,2)	Value of Blue measure, V(6)
(if JUMP = .FALSE., next record *, else GOTO next record **)		
(* next record)		
1	DNSRAD( )	Population of Blue radios/radars by type. NOTE: For this and following subscripted variables (indicated by the parentheses ( ) symbol), these subscripts pertain for Blue: 1 = Very high frequency (VHF) radios 2 = High frequency (HF) radios 3 = Multichannel radio relay equipment 4 = Countermortar/counterbattery radars
2	TOPJAM( )	Maximum possible fraction of type Blue emitters which could be jammed
3	FACJAM( )	Change in Blue combat effectiveness for each 1 % of Blue communications lost
4	WTJAM( )	Fraction of the change in Blue combat effectiveness which can be attributed to a particular type of radio/radar being jammed

Table E-I-5. C2EW Input
(page 2 of 4 pages)

Field	Variable	Data description
5	NJAM( )	Number of types of Red jammers capable of countering the similarly subscripted radio/radar
(next NJAM records)		
1	EFFJAM( )	Quantity of Blue radios/radars by type jammed by one of a type Red jammer
2	DNSJAM( )	Population of Red jammers by type capable of countering the similarly subscripted radio/radar
(Continue reading jammer records for this type Blue radio/radar until NJAM types have been read; then return to next record (*) until all four types of radios have been read. After all four types of radios have been read, proceed to ** next record)		
(** next record)		
1		Switch indicates whether to use given values in record or compute values from following input, where T = use, F = read data
2	FACT(6,3)	Value of Red countermeasure, S(6)
3	FACT(6,4)	Value of Red measure, T(6)
(if JUMP = .FALSE., next record ***, else go to next **)		
(*** next record)		
1	DNSRAD( )	The value of data read into this field is unity (1.0) NOTES: (a) The approach used to calculate Red C2EW countermeasures and measures focuses on Blue ECM (electronic countermeasure) equipment instead of Red radios/radars.

Table E-I-5. C2EW Input
(page 3 of 4 pages)

Field	Variable	Data description
		NOTES (continued): (b) Although the field naming convention is the same one used for the preceding Blue calculations, the data read into these fields has a different "meaning" in some cases (as indicated in following paragraphs). (c) Subscripted variables (indicated by the parentheses ( ) symbol) for Red C2EW are: 1 = Red HF radios 2 = Red VHF radios 3 = Red communication equipment jammed by Blue airborne jammers 4 = Red noncommunication equipment such as radars
2	TOPJAM( )	Maximum possible degradation to Red combat effectiveness corresponding to the appropriate subscript
3	FACJAM( )	Value of unity (1.0) for Red C2EW
4	WTJAM( )	Fraction of the change in Red combat effectiveness which can be attributed to a particular type of Blue jamming such as HF or VHF.
5	NJAM( )	Number of types of Blue jammers capable of countering a similarly subscripted radio/radar
(next NJAM records)		
1	EFFJAM( )	Fraction of the change in Red combat effectiveness which can be attributed to a particular type Blue jammer such as the AN/TLQ-15

Table E-I-5. C2EW Input
(page 4 of 4 pages)

Field	Variable	Data description
2	DNSJAM( )	Population of the specific type of Blue jammer targeted against Red C2EW capability

(Continue reading jammer records for this type Red C2EW capability until NJAM types have been read; then return to next record *** until all 4 types of Red C2EW capability have been read.)

(**)

```
F             1.0          1.0
2088          0.685        1.37685    0.8318     1
81.5          5
77          0.02         0.02394    0.0004     1
7.1           6
    0         0.02         0.02337    0.0004     2
1.714          3
1.714          1
13             1.00         0.19       0.1674     1
1.25          4
F             1.0          1.0
1.0           0.86         1.0        0.1016     1
0.1433        0
1.0           0.685        1.0        0.4566     5
0.0274        0
0.0274         0
0.00913        0
.0183         0
.00834        0
1.0           0.315        1.0        0.2100     2
.0228          0
.00365         0
1.0           0.40         1.0        .2317      1
.00344        0
```

Figure E-I-6. Example of C2EW Data Input to the AFP CS/CSS Main Preprocessor

f. Medical. Table E-I-6 describes the medical input. Figure E-I-7 displays sample input records for generating medical factors.

Table E-I-6. Medical Input
(page 1 of 2 pages)

Field	Variable	Data description
(1st record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(7,1)	Value of Blue countermeasure, U(7)
3	FACT(7,2)	Value of Blue measure, V(7)
(if JUMP = .FALSE., next record, else GOTO next record **)		
(next record)		
1	BDIVS	Blue divisional personnel strength supported
2	BAREAS	Blue area support strength supported, if any
3	BSFACT	Blue surge factor
4	BDWIA	Blue divisional WIA rate (per 1,000)
5	BAWIA	Blue area support WIA rate (per 1,000)
(next record)		
1	BDDNBI	Blue divisional DNBI rate (per 1,000)
2	BADNBI	Blue area support DNBI rate (per 1,000)
3	BMBCAP	Blue medical bn capability (releases per day)
4	BFEVAC	Blue fraction of casualties evacuated from division area
(** next record)		

Table E-I-6. Medical Input
(page 2 of 2 pages)

Field	Variable	Data description
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(7,3)	Value of Blue countermeasure, S(7)
3	FACT(7,4)	Value of Blue measure, T(7)
(if JUMP = .FALSE., next record, else GOTO next **)		
(next record)		
1	RDIVS	Red divisional personnel strength supported
2	RAREAS	Red area support strength supported, if any
3	RSFACT	Red surge factor
4	RDWIA	Red divisional WIA rate (per 1,000)
5	RAWIA	Red area support WIA rate (per 1,000)
(next record)		
1	RDDNBI	Red divisional DNBI rate (per 1,000)
2	RADNBI	Red area support DNBI rate (per 1,000)
3	RMBCAP	Red medical bn capability (releases per day)
4	RFEVAC	Red fraction of casualties evacuated from division area
(**)		

```
F    1.0      1.0
 19485.0   19485.0          5.0       3.17        1.73
     5.65        1.2      160.0         .5
F    1.0      1.0
 11920.0   11920.0          5.0      24.0         5.4
     8.00        1.8      500.0         .5
```

Figure E-I-7. Example of Medical Data Input to the AFP CS/CSS Main Preprocessor

g. Maintenance. Table E-I-7 describes the maintenance input. Figure E-I-8 displays sample input records for generating maintenance factors.

Table E-I-7. Maintenance Input (page 1 of 3 pages)

Field	Variable	Data description
(1st record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(8,1)	Value of Blue countermeasure, U(8)
3	FACT(8,2)	Value of Blue measure, V(8)
(if JUMP = .FALSE., next records, else GOTO next record **)		
(next records)		

Table E-I-7. Maintenance Input
(page 2 of 3 pages)

Field	Variable	Data description
1	BMCAP( )	Blue maintenance capability (number of personnel on hand) in each of i maintenance categories, i = 1 through 7. The subscripted variable BMCAP( ) corresponds to each of the following skill specialties:
		i **MOS** **Maintenance category**
		1 31E Field radio repair
		2 45B Small arms repair
		3 45L Artillery repair
		4 52D Power generator repair
		5 62B Construction equipment repair
		6 63G,H,W Automotive repair
		7 67 Aircraft repair
2	BMREQ( )	Blue maintenance personnel requirement according to TAADS (authorization documents) in each of i categories as shown above.
(** next record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(8,3)	Value of Red countermeasure, S(8)
3	FACT(8,4)	Value of Red measure, T(8)
(if JUMP = .FALSE., next records, else GOTO next **)		
(next records)		

Table E-I-7. Maintenance Input
(page 3 of 3 pages)

Field	Variable	Data description
1	RMCAP	Capability (1,000s MH/day) for maintenance category
2	RMREQ	Requirement (1,000s MH/day) for maintenance category
(**)		

```
F    1.0      1.0
   7
   627.12    11.807
  1057.26     9.835
    16.08      .256
    68.34      .446
    52.26     1.566
   217.08     2.42
    12.06      .37
T    1.0      .7017
```

Figure E-I-8. Example of Maintenance Data Input to the AFP CS/CSS Main Preprocessor

h. Supply and Transportation. Table E-I-8 describes the supply and transportation input. Figure E-I-9 displays sample input records for generating supply and transportation factors.

Table E-I-8. Supply and Transportation Input (page 1 of 3 pages)

Field	Variable	Data description
(1st record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(9,1)	Value of Blue countermeasure, U(9)
3	FACT(9,2)	Value of Blue measure, V(9)
(if JUMP = .FALSE., next record, else GOTO next record **)		
(next record)		
1	BNSTA	Number of Blue S&T equipment types
(next BNSTA records)		
1	BDST	Population of Blue equipment of type
2	BAVAIL	Availability of Blue equipment of type
3	BEFF	Capacity of Blue equipment type (meas-miles/day)
4	TCOMP	Pointer to transportation class 1 = STON 2 = gallons 3 = STON (ALOC)
(next record)		
1	BNCLAS	Number of Blue classes of supply
(next BNCLAS records)		

Table E-I-8. Supply and Transportation Input
(page 2 of 3 pages)

Field	Variable	Data description
1	BCONR	Blue required consumption by class of supply (lbs/man/day)
2	BALOC	Percent of class required by ALOC
3	BBASIC	Percent of class carried by supported units
(** next record)		
1	JUMP	Switch for whether to use given values in record or compute values from following input, T = use, F = read data
2	FACT(9,3)	Value of Red countermeasure, S(9)
3	FACT(9,4)	Value of Red measure, T(9)
(if JUMP = .FALSE., next record, else GOTO next **)		
1	RDIVS	Red division personnel strength
(next record)		
1	RNSTA	Number of Red S&T equipment types
(next RNSTA records)		
1	RDST	Population of Red equipment of type
2	RAVAIL	Availability of Red equipment of type
3	REFF	Capacity of Red equipment type (meas-miles/day)
4	TCOMP	Pointer to transportation class 1 = MTON 2 = liters 3 = MTON (ALOC)
(next record)		

Table E-I-8. Supply and Transportation Input
(page 3 of 3 pages)

Field	Variable	Data description
1	RNCLAS	Number of Red classes of supply
(next RNCLAS records)		
1	RCONR	Red required consumption by class of supply (lbs/man/day)
2	RALOC	Percent of class required by ALOC
3	RBASIC	Percent of class carried by supported units
(**)		

NOTE: The AFP treatment of the supply and transportation function requires Blue divisional personnel strength as input. That datum need not be read in this segment because the value is read earlier by the CS/CSS Main Preprocessor for one of the foregoing CS/CSS functions.

```
F          1.0        1.0
 8
  2       1.0          60.00  1
 72       1.0           5.00  1
654       1.0           2.50  1
 2       1.0        2500.00  2
 19       1.0        1300.00  2
 8       1.0           8.00  1
8          1.0      500.00   2
 63       1.0           6.35  3
8
 3.82      0.0        1.0
15.17      0.0        1.0
   .33     0.0        1.0
 8.50      0.0        1.0
13.78      0.0        1.0
6.97      0.0        1.0
   .35     0.0        1.0
 .72      0.3        0.7
F          1.0        1.0
11920
2
1          1.0        4382.5     1
2200       1.0        10449      2
2
1866.92    0.0        1.0
 42.28     0.0        1.0
```

Figure E-I-9. Example Supply and Transportation Data Input to the AFP CS/CSS Main Preprocessor

Section III. OUTPUT

E-I-6. Table E-I-9 describes the sample extract records from a file produced by the CS/CSS Main Preprocessor. Figure E-I-10 displays sample extract records; the records have the same format.

Table E-I-9. Output Description

Columns	Variable	Data description	Format
(Ith record)			
1-2	I	Counter to indicate the Ith CS/CSS function listed	I2
3-5	IENV	Numerical indication of the type environment pertinent to this output listing	I3
6-13	FACT(I,1)	Blue countermeasure factor U(I) for the Ith CS/CSS function	F8.2
14-21	FACT(I,2)	Blue measure factor V(I) for the Ith CS/CSS function	F8.2
22-29	FACT(I,3)	Red countermeasure factor S(I) for the Ith CS/CSS function	F8.2
30-37	FACT(I,4)	Red measure factor T(I) for the Ith CS/CSS function	F8.2
38-42	--		5X
43-53	NFUN(I)	Literal name of the Ith function such as Mines or C2EW	A11
54-55	--		2X
56-61	NAMPOS	Designation of the posture; e.g., STATIC, BAPD	
62-63	--		2X
64-67	FORCE	Identification of unit and fiscal year; e.g., 8I82 for the 8th Infantry Division, fiscal year 1982	A4

F	EN	B C/M	B MEAS	R C/M	R MEAS	FUNCTION	POST	FORCE
1	2	1.12	.99	2.28	.97	MOBILITY	STATIC	8I82
2	2	1.00	.92	1.00	.85	BRIDGING	STATIC	8I82
3	2	1.00	.69	1.00	1.29	MINE	STATIC	8I82
4	2	1.00	.29	1.00	1.46	PROT.POSN.	STATIC	8I82
5	2	1.00	1.00	1.00	1.00	N/A	STATIC	8I82
6	2	1.00	.98	1.00	1.17	C2EW	STATIC	8I82
7	2	1.00	1.02	1.00	1.02	MEDICAL	STATIC	8I82
8	2	1.00	.97	1.00	2.83	MAINTENANCE	STATIC	8I82
9	2	1.00	.61	1.00	.81	SUPP&TRANS	STATIC	8I82

Figure E-I-10. Example of Blue Countermeasures and Measures and Red Countermeasures and Measures by CS/CSS Function as Generated by AFP CS/CSS Main Preprocessor

Section IV. RUNSTREAM

E-I-7. Figure E-I-11 displays a sample runstream for a single execution of the two CS/CSS Preprocessors, both the Search and the Main Preprocessors.

```
1:@RUN,M/TP A1P3,E1899P2277D,UNCLASSIFIED,5,100
2:@HDG,U ******UNCLASSIFIED******
3:@QUAL UNCLASSIFIED
4:@ASG,A *P3OUTPUT.
5:@USE 20, *P3OUTPUT.
6:@ASG,T 17.
7:@ED *P3COUNTER.BASISTEST,17.
8:@USE 15., *P3TESTCS.
9:@XQT *P3COUNTER.870PRECSCSSI
```

Figure E-I-11. Sample Runstream for Execution of the AFP CS/CSS Preprocessors

Section V. PROGRAM

E-I-8. Figures E-I-12 and E-I-13 display the basic logical flow of the two AFP CS/CSS Preprocessors.

E-I-9. The following paragraphs describe the calculation of measures and countermeasures for the CS/CSS functions represented within AFP.

a. Tactical Mobility. The treatment of tactical mobility provides a first order approximation to the advantages and disadvantages of having more or less air and ground mobility as a net result of unimpeded vehicle speeds (the Blue and Red measures) and non-mine obstacles (the Blue and Red countermeasures).

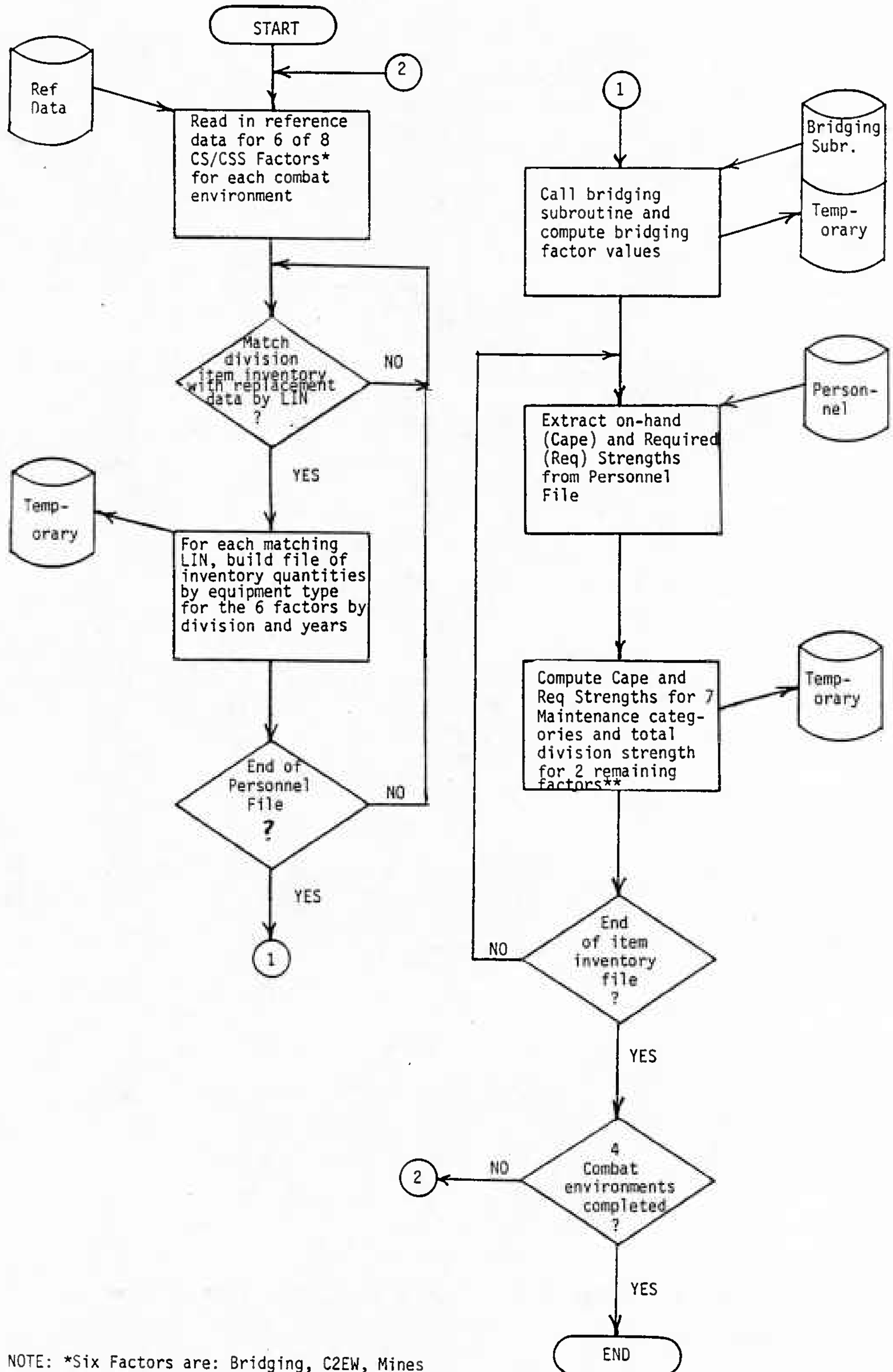
(1) Measure. Tactical mobility is measured by combining the weighted average speeds of a side's aircraft and ground vehicles. If V(1) is the Blue tactical mobility measure,

$$V(1) = \frac{\sum_{i}^{G} (N(i) * S(i))}{\sum_{i}^{GB} (N(i) * S(i))} * \frac{\sum_{i}^{GB} N(i)}{\sum_{i}^{T} N(i)} + \frac{\sum_{i}^{A} (N(i) * S(i)}{\sum_{i}^{AB} (N(i) * S(i)} * \frac{\sum_{i}^{AB} N(i)}{\sum_{i}^{T} N(i)} * M$$

where:

G = the number of Blue ground combat vehicle types
GB = the number of Blue ground combat vehicle types in the selected base case
i = index of Blue vehicle types; e.g., i = 1, 2, ...
N(i) = the number of vehicles of type i (including aircraft)
S(i) = the speed (kph) of the vehicles of type i
T = the total number of Blue ground combat vehicle and aircraft types
A = the number of Blue aircraft types
AB = the number of Blue aircraft types in the selected base case
M = ratio of the aircraft weighted average speed to ground combat vehicle weighted average speed for two combined base cases

A similar expression (for T(1)) applies to Red ground combat vehicles. It is implied that faster is better. For other measures held constant, greater speed tends to improve a side's exchange ratio (kills/losses). It is not suggested that the simple proportionality to speed holds for extremely low or extremely high speeds.



NOTE: *Six Factors are: Bridging, C2EW, Mines Protective Positions, Tactical Mobility, and Supply and Transportation.
**Remaining factors are Maintenance and Medical.

Figure E-I-12. Flow Chart--Search Preprocessor

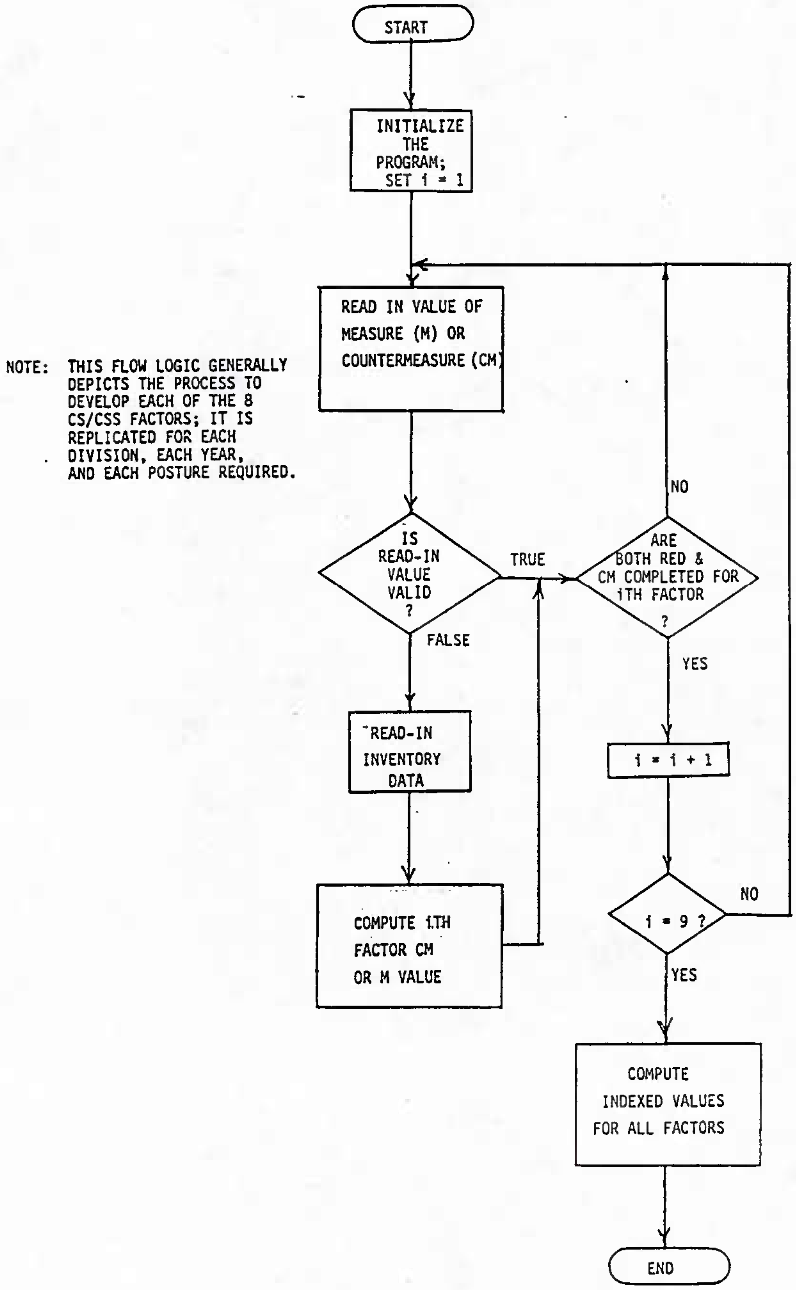


Figure E-I-13. Flow Chart--AFP CS/CSS Main Preprocessor

(2) Countermeasure. The Blue side's countermobility measure is the ratio of its non-mine obstacle creation capability to the Red side's doctrinal non-mine obstacle creation clearing capability. It is assumed the tactical commander would allocate 100 percent of his earth moving assets to countermobility and 100 percent to build protective positions. Situationally, however, the tactical commander on the ground would allocate his resources according to the scenario in progress. The Blue obstacle creation capability in packed dirt expressed in cubic meters per hour (m^3/hr) is the dig-in rate of on-hand Blue equipment. The Red obstacle clearing capability similarly expressed in m^3/hr is the excavation rate of doctrinally-prescribed Red equipment. If the Blue tactical mobility countermeasure is U(1), then,

$$U(1) = \sum_{i}^{J} (N(i) * D(i)) / \sum_{i}^{K} (N(i) * E(i))$$

where:

J = the number of Blue equipment types capable of creating obstacles
i = index of equipment types
N(i) = number of equipment of type i
D(i) = the dig-in rate (m^3/hr) of equipment of type i
K = the number of Red equipment types capable of clearing obstacles
E(i) = the excavation rate of equipment of type i

An expression similar to U(1) applies to the Red side, S(1). This would be accomplished by substituting Red doctrinal obstacle creation equipment capability in the numerator of the equation and Blue obstacle clearing equipment capability in the denominator.

b. Bridging. The treatment of bridging provides first order approximation to the advantages and disadvantages of having more or less gap-crossing capability.

(1) Measure. A side's measure is determined in a submodel as the bridging capability of a combat force using the force's bridge, raft, and vehicle inventory. The submodel uses bridge and raft inventories as well as characteristics of bridges and rafts, the vehicle inventory by weight class, and a series of gaps of specified widths which the force attempts to cross. For each gap width, the submodel computes the following:

(a) A determination whether it is possible to cross the gap, given the rafts and bridges available (without regard to vehicle inventory);

(b) The fraction of vehicles which can cross the gap via bridge or raft; and

(c) The total time required for those vehicles to cross the gap, including set-up and take-down times for bridges and rafts.

These assumptions pertain: the availability of personnel to perform set-up and take-down operations is not considered. Although not all the force may be able to cross a given width of gap, when the next gap is processed the entire force is viewed as being ready to cross it even though the entire force may not be able to do so. The bridging measure or capability V(2) is computed as follows:

$$V(2) = (F_1 * F_2 * F_3)/(\text{base } F_1 * \text{base } F_2 * \text{base } F_3)$$

where,

$$F_1 = \sum_{i}^{n} T(i) * D(i)$$

n = number of different gap widths
T(i) = the "gap weight" (i=1,2...,n)
D(i) = 1 if it is possible to cross, 0 otherwise

and

$$T(i) = (G(i) * W(i)) / (\sum_{i}^{n} G(i) * W(i))$$

where,

G(i) = the number of gaps of type i (i=1,2,...,n)
W(i) = the width of the ith type gap (i=1,2,...,n)

$$F_2 = 1/(\sum_{i}^{n} G(i) * C(i))$$

C(i) = the calculated crossing time for gap type i (i=1,2,...,n)

$$F_3 = (\sum_{i}^{n} R(i))/n$$

R(i) = the fraction of vehicles capable of crossing gap type i (i=1,2,...,n).

Base F_1, base F_2, and base F_3 = values of F_1, F_2, and F_3 in a base case

(2) **Countermeasure.** None; i.e., U(2) = S(2) = 1.0.

c. **Mine/Countermine.** The treatment of mine/countermine provides first order approximation to the advantages and disadvantages of a side's having more or less mine laying or breaching capability.

(1) **Measure.** The computations depicting a side's mine laying capability and the enemy's mine clearing capability are incorporated into a single ratio representing a side's mine and countermine functions. Both equipment and personnel capabilities are considered. Also, a side's mine laying capability is affected by the opposing side's lane clearing doctrine (i.e., number and width of lanes to be cleared) to yield effective mine laying capability. If V(3) is the Blue mine/countermine measure, then

$$V(3) = L(b)/C(r)$$

where,

L(b) = Blue capability to lay mines

and

$$L(b) = Q(b) + P(b) * 500/13 * N(r) * W(r)/0.5 * F(r)$$

where,

Q(b) = rate (mines/hr) of Blue mine laying/clearing equipment, nonmanual methods
P(b) = number of Blue personnel engaged in minelaying/clearing operations
500/13 = number of mines per hour laid by a 13-man team (12EM + 1 NCO) of engineer personnel (MOS 12B)
N(r) = doctrinal number of Red lanes to be cleared through a Blue minefield by an attacking Red division
W(r) = doctrinal width (meters) of a cleared Red lane through a minefield
0.5 = represents the assumption that one-half of the division front is protected by natural obstacles
F(r) = doctrinal width (meters) of a Red division's frontage
C(r) = Red capability to clear mines (mines/hr) is currently calculated off-line

Otherwise, T(3) is the Red mine measure such that,

$$T(3) = L(r)/C(b)$$

where,

L(r) = Red capability to lay mines calculated by replacing the (b) and (r) notations with their respective converse in the formula for L(b)
C(b) = Blue capability to clear mines (mines/hr)

and

$$C(b) = Q(b) + P(b) * 3.6/30.0$$

where,

3.6 = number of Red mines encountered per lane
30.0 = Blue doctrine specifies 30 manhours required to clear one lane that is 1 km long.

(2) Countermeasure. None; i.e., U(3) = S(3) = 1.0.

d. Protective Positions. The treatment of protective positions provides first order approximation to the advantages and disadvantages of having more or less capability to construct protective positions. Within the AFP Combat Module, combat postures are assumed to be "pure" in the sense that none or all of the side's ground systems are in defilade. That assumption directly influences the sensing sizes and SSPKs by posture employed within the Combat Module. The protective position factors provide some a posteriori adjustment to the Combat Module's raw estimates of kills and losses. In a defense intense posture, where all defenders are assumed to be in defilade, a defender's protective position factor must be = 1.0. In open postures, it is reasonable to permit factors 1.0 as well. The protective position function does not include countermeasures; i.e., countermeasure factors are all set to the default value 1.0.

(1) Measure. A side's measure is simply the ratio of its ability to construct protective positions in 5 hours to the required number of positions. The required number of positions is usually assumed equal to the total inventory of ground combat systems supported. However, to the extent that nature or peacetime work have already provided some adequate positions, the requirement may be less than the total number of ground systems. Or to the extent that a side may wish or be forced to change locations more frequently than with 5 hours' warning, the requirement may be larger than the number of ground systems. If V(4) is the Blue protective position measure,

$$V(4) = (\sum_{i}^{M} (A(i) * B(i)) * H / (F * C)$$

where,

M = the number of types of Blue systems assigned to prepare protective positions

i = the index of the Blue system types; i = 1 to M

A(i) = the number of Blue systems of type i

B(i) = the position preparation rate per hour per Blue system, adjusted if the system must perform other roles as well

H = 5 (hrs), the assumed critical time interval, may be adjusted for the number of productive H-hour productive periods that can be achieved in a combat day if moves are frequent

F = the inventory of Blue ground combat systems

C = the fraction of Blue ground systems requiring prepared positions, C 0.0 and may exceed 1.0

A similar expression (for T(4)) applies to the Red side.

(2) Countermeasure. None; i.e., U(4) = S(4) = 1.0.

Note: U(5), V(5), S(5), and T(5) are not employed in the current version of the AFP system.

e. Command, Control, and Electronic Warfare (C2EW). The treatment of C2EW provides first order approximation of the advantages and disadvantages of being more or less able to overcome an opponent's efforts to jam one's own electronic communication system and thereby disrupt command and control. A side's C2EW measure and countermeasure are combined within a single factor; the AFP convention is to regard the single factor as a measure with the countermeasure always 1.0.

(1) Measure. A side's measure is simply the ratio of its unjammed radios to the number required for electronic communication supportive of "normally intense" combat command and control activity. Because a division's inventory of radios includes some redundancy and because transmissions need not be continuous, the jamming of some radios need not induce any degradation. If V(6) is the Blue C2EW measure,

$$V(6) = 1.0 - SUM$$

where,

$$SUM = \sum_{i}^{4} F(i) * W(i) * T(i)$$

i = the index of Blue C-E system types; i = 1 to 4
F(i) = a factor describing the change in combat effectiveness corresponding to each 1 percent of communications lost by type of C-E system
W(i) = the percent (weight) of combat effectiveness change allocated to a particular type C-E system

$$T(i) = \min(M(i), \sum_{j}^{N} E(j,i) * Q(j,i)/R(i))$$

where,

M(i) = maximum possible factor by which combat effectiveness could be degraded for a particular type of C-E system
j = the index of the jammer types (j=1,2,...,N)
E(j,i) = the effective number of Blue systems of type i jammed by one Red jammer of type j
Q(j,i) = the quantity of Red jammers of type j
R(i) = the quantity of Blue C-E systems of type i

The Red measure T(6) uses formula similar to the preceding, but T(6) focuses on the effectiveness of Blue jammers by type against Red C-E systems. Note that selected data elements such as quantity of Red C-E systems jammed (i.e., DNSRAD(i)) and the jamming factor describing the change in combat effectiveness corresponding to each 1 percent of communications lost (i.e., FACJAM(i)) have been converted to a base value (1.0). If T(6) is the Red measure,

$$T(6) = 1.0 - \left(\sum_{i}^{4} (F(i) * W(i)) * \min\left(M(i), \sum_{j}^{N} E(j,i) * Q(j,i)/R(i)\right)\right)$$

where,

i = index of Blue type jammers countering Red C-E systems (i = 1 to 4)
F(i) = 1.0
W(i) = percent expressed as a decimal number indicating the effective weight of the targeted Red C-E system
M(i) = factor indicating the maximum possible effectiveness of a type Blue jammer against a type Red C-E system
E(j,i) = factor expressed as a decimal number indicating the percentage effectiveness of a particular jammer (by nomenclature) identified by subscript j
Q(i) = the quantity of Red jammers of type j
R(i) = 1.0

(2) Countermeasure. None; i.e., U(6) = S(6) = 1.0.

f. Medical. The AFP treatment of the medical function provides first order approximation to the advantages and disadvantages of a side's having more or less divisional medical capability given fixed treatment and evacuation policies and fixed higher echelon medical capability. It is assumed that only relatively small differences from norms need be considered. Blue and Red medical measures may differ from 1.0; however, the AFP representation of the medical function does not include countermeasures (hence "countermeasure" values are always set to the default value 1.0).

(1) Measure. A side's measure is based on the daily loss of division strength directly attributable to shortfall, if any, in divisional medical capability. If V(7) is the Blue medical measure,

$$V(7) = 1.0 - Q * (A - C*(1.0 + D)) / B \leq 1.0$$

where, in turn,

$$A = (F*(G + H) + L*(K + M)) * P / 1000.0$$

$$B = F + L$$

and,

A = daily admissions

B = total supported personnel strength

C = divisional medical capability, returns to duty per day

D = evacuees to higher echelon as fraction of the divisional medical battalion return to duty rate

F = division personnel strength

G = division normal WIA per day per 1,000 strength

H = division normal DNBI per day per 1,000 strength

K = support area normal WIA per day per 1,000 strength

L = support area personnel strength

M = support area normal DNBI per day per 1,000 strength

P = surge factor (combat intensity/normal intensity)

Q = number of days of concern, usually 1.0

The factor typically has a very small range. The form of the above expression was approved by the Agency's medical POC. However, note that, if there is no divisional medical battalion capability, the measure is $V(7) = 1.0 - Q * A / B$. If Q = 1.0 and the casualty rate is 2 percent per day, V(7) = 0.98. No attempt is made to reflect the impact on personnel attitude when they know there is more or less medical support available. A reduction in a side's medical factor reduces that side's kill/loss ratio, thereby tending to reduce its combat potentials. As for other functions, the range of practical interest is assumed to be relatively small. The simple proportionalities in the above expressions probably overestimate the impact of extremely small shortfalls and certainly underestimate the net effects of very large shortfalls.

A similar set of expressions (for T(7)) applies to the Red side.

(2) Countermeasure. None; i.e., U(7) = S(7) = 1.0.

g. Maintenance. The AFP treatment of the maintenance function provides first order approximation of the advantages and disadvantages of a side's having more or less divisional maintenance capability relative to planning or doctrinal requirements levels. Obviously excess maintenance capability cannot produce benefit. Within a sufficiently narrow range, the maintenance impact is assumed proportional to the capability/requirement ratio. More maintenance improves a side's kill/loss ratio and, therefore, tends to increase combat potential. The maintenance function does not include countermeasures.

(1) Measure. The measure is simply a gross capability-to-requirement average over several equipment categories. If V(8) is the Blue maintenance measure,

$$V(8) = ((\sum_{i}^{M} (C(i)/R(i)) \leq 1.0$$

where,

M = number of maintenance categories hardwired to equal 7

i = index of maintenance categories; i = 1 to M; typically: automotive, aviation, artillery, engineer, communication, power generation, small arms

C(i) = Blue maintenance capability in category i; personnel on-hand by fiscal year according to Readiness Indicator Model (RIM) data

R(i) = Blue maintenance requirement in category i according to TOE

NOTE: The maximum value for r(8) is 1.0 because there is no tradeoff of capabilities across maintenance categories. For example, a large surplus of communication maintenance capability does not compensate for a shortfall in automotive.

A similar expression (for T(8)) applies to the Red side.

(2) Countermeasure. None; i.e., U(8) = S(8) = 1.0.

h. Supply and Transportation. The AFP treatment of the supply and transportation function provides first order approximation of the advantages and disadvantages of a side's having more or less divisional supply and transportation capability relative to planning or doctrinal requirements. A capability to move more than there is available to be moved does not generate added benefit. Subject to that obvious limitation, less supply and transportation capability reduces a side's kill/loss ratio and, therefore, tends to decrease combat potential. The supply and transportation function does not include countermeasures.

(1) Measure. The measure is simply a gross capability-to-requirement weighted average over three cargo/movement categories: STON, gallons (bulk POL), and ALOC. Capabilities and requirements may be expressed in terms of "unit of measure-miles;" e.g., STON-miles. If the "miles" is set to 1.0, capabilities and requirements are then expressed in "single lift" terms. The capabilities are computed from the numbers and cargo carrying capacities of divisional transportation assets. The requirements are computed from planning factors expressed in terms of lbs/man/day for different classes of supply and for the division personnel strengths. Bulk and packaged requirements may be specified separately; e.g., for Class III by representing Class III with subclasses. For each class (or subclass) of supply, a fraction to be delivered by air may be specified. Hence, zero bulk POL may be required to be delivered over the ALOC, but nonzero quantities of packaged POL may be designated for air delivery. Units of measure are converted as appropriate (e.g., lbs to gals for bulk POL). Adjustment is made for how much of total supply must be transported by division assets. If V(9) is the Blue measure,

$$V(9) = Q1 * A / X + Q2 * E / P + Q3 * K / CC$$

where,

Q1 + Q2 + Q3 = 1.0 and Q1, Q2, and Q3 are nonnegative weights expressing the net relative importance of the three ratios. E.g., if all three are equally important, Q1 = Q2 = Q3 = 1/3.

A = Blue STON ground transport capability (STON-miles/day)

X = Blue STON ground transport requirements (STON-miles/day)

E = Blue Class III transport capability (gal-miles/day)

P = Blue Class III transport requirement (gal-miles/day)

K = Blue ALOC transport capability (STON-miles/day)

CC = Blue ALOC transport requirement (STON-miles/day)

Capabilities are computed in accord with:

$$A = \sum_{i}^{NT} (B(i) * C(1,i) * D(1,i))$$

$$E = \sum_{i}^{NT} (B(i) * C(2,i) * D(2,i))$$

$$K = \sum_{i}^{NT} (B(i) * C(3,i) * D(3,i))$$

where,

NT = the number of Blue transporter types

i = the index of the Blue transporter types; i = 1 to NT

B(i) = the number of Blue transporters of type i

C(j,i) = the availability of Blue transporter type i for use with cargo category j

D(j,i) = the capacity of Blue transporter type i for use with cargo category j; expressed in terms of unit of measure x distance. (Distance is 1.0 for single lift.)

j = index of cargo category; j = 1 to 3 for STON, gals, ALOC

Requirements are computed in accord with:

$$X = (R/2000.0) * \sum_{k}^{NC} (Y(k) * AA(k) * DT(k) * (1.0 - BB(k)))$$

$$P = R * LG * Y(k') * AA(k') * DT(k')$$

$$CC = (R/2000.0) * \sum_{k}^{NC} (Y(k) * AA(k) * BB(k))$$

where,

NC = the number of classes (subclasses) of supply

k = index of the number of classes of supply. (One of these may be for packaged POL.)

k' = index of bulk POL class

R = division personnel strength

Y(k) = consumption rate (lbs/person/day) or class of supply k

AA(k) = fraction of class k to be carried by unit

DT(k) = distance (mi) over which supply class k must be transported. Set to 1.0 for single lift.

BB(k) = fraction of class k to be carried on ALOC

LG = conversion factor: lbs to gals

and the term 2000.0 is the conversion factor: lbs to STON.

A similar expression (for T(9)) applies to the Red side. Where appropriate, use the factor 1.102 to convert from MTON to STON and the factor 0.265 to convert from liters to gallons.

(2) Countermeasure. None; i.e., U(9) = S(9) = 1.0.

E-I-10. After the values of U, V, S, and T are computed for each CS/CSS function, a subroutine (INDXCS) is called. This subroutine normalizes (or indexes) each measure/countermeasure value to the base case value.

Section VI. PROGRAM LISTINGS

E-I-11. This section contains source listings which depict both the Main and the Search Preprocessors. The programs are written using FORTRAN 77 for the Sperry UNIVAC 1108 computer.

a. Figures E-I-14 and E-I-15 are listings of the Procs used in the Main program, the Output subroutine, and the BRIDGE subroutine to the CS/CSS Search program.

b. Figure E-I-16 is a listing of the main CS/CSS Search program.

c. Figure E-I-17 is a listing of the Output subroutine to the CS/CSS Search program.

d. Figure E-I-18 is a listing of a FORTRAN Proc identifying certain COMMON blocks in the BRIDGE subroutine to the CS/CSS Search program.

e. Figure E-I-19 is a listing of the BRIDGE subroutine, with 13 external subroutines.

f. Figure E-I-20 is a listing of the CS/CSS Main Preprocessor.

g. Figure E-I-21 is a listing of the AFP CS/CSS Main Preprocessor.

h. Figure E-I-22 is a source listing of the external subroutine INDXCS to the CS/CSS Main Preprocessor.

```
BDGROU PROC
      COMMON/BRGROU/IBTAFP(60),
     *VTCLAS(60),ITAFP(60),
     *BGLOOP,
     *BCAP,
     *IBTNUM(32)
      INTEGER BGLOOP,IBFLAG
 END
```

Figure E-I-14. Listing of FORTRAN Procs Identifying COMMON Blocks, Variables, Arrays, and Parameters of the Main Program, the Output Subroutine, and the BRIDGE Subroutine for the CS/CSS Search Preprocessor

```
BPDAT PROC
      PARAMETER (MAXGAP=10, MAXVEH=30, MAXBRG=20, MAXCP=20,
     &      IBIG=1000000000, BIG=1.0E10)
C     MAXGAP = THE MAX NUMBER OF GAP TYPES
C     MAXVEH = THE MAX NUMBER OF VEHICLE TYPES
C     MAXBRG = THE MAX NUMBER OF BRIDGE/RAFT TYPES
C     MAXCP = THE MAXIMUM NUMBER OF CROSSING POINTS
C     IBIG = A REALLY BIG INTEGER
C     BIG = A REALLY BIG REAL NUMBER
C     INPUT DATA:
      INTEGER
     +     BNUM(MAXBRG),        NBRIDG,              NCPD,
     +     NGAPS,               NVEH,                VNUM(0:MAXVEH)
      REAL
     +     BASEF1,              BASEF2,              BASEF3,
     +     BCLAS(MAXBRG,MAXCP),BSETS(MAXBRG),        CPSET(MAXBRG,MAXCP),
     +     NGP(MAXGAP),         NV(MAXVEH),          SETUP(MAXBRG,MAXCP),
     +     TDFACT,              VCLAS(MAXVEH),       VCRATE(MAXBRG,MAXCP),
     +     WGAP(MAXGAP)
      CHARACTER
     +     BDESC(MAXBRG)*10,    VDESC(MAXVEH)*10
C     END OF INPUT DATA
      INTEGER
     +     IBT(MAXCP),          ISTATE(MAXCP),       IVT(MAXCP),
     +     NCB(MAXBRG),         NCPC(MAXGAP)
      REAL
     +     BLMAX,               CSCORE(MAXBRG),      ECR(MAXVEH),
     +     FRVEHC(MAXGAP),      GAPCTM(MAXGAP),      GAPWT(MAXGAP),
     +     LEFT(MAXVEH),        POSSBL(MAXGAP),      TNEXT(MAXCP),
     +     TNOW,                TOTVEH
C     POSSIBLE VALUES OF ISTATE:
      PARAMETER (KSETUP=1, KCROSS=2, KTDOWN=3, KFINI=4)
C       KSETUP: BRIDGE BEING SET UP
C       KCROSS: BEING CROSSED BY VEHICLES
C       KTDOWN: BRIDGE BEING TAKEN DOWN
C       KFINI:  FINISHED (NO ACTIVITY)
      COMMON /BPDAT/
     +     BASEF1,         BASEF2,         BASEF3,         BCLAS,
     +     BDESC,          BLMAX,          BNUM,           BSETS,
     +     CPSET,          CSCORE,         ECR,            FRVEHC,
     +     GAPCTM,         GAPWT,          IBT,            ISTATE,
     +     IVT,            LEFT,           NBRIDG,         NCB,
     +     NCPC,           NCPD,           NGAPS,          NGP,
     +     NV,             NVEH,           POSSBL,         SETUP,
     +     TDFACT,         TNEXT,          TNOW,           TOTVEH,
     +     VCLAS,          VCRATE,         VDESC,          VNUM,
     +     WGAP
 END
```

Figure E-I-15. Listing of FORTRAN Procs Identifying COMMON Blocks, Variables, Arrays, and Parameters of the Main Program, the Output Subroutine, and the BRIDGE Subroutine for the CS/CSS Search Preprocessor

```
      INTEGER STP, KT, ITP
      INCLUDE BPOAT,LIST
      INCLUDE BOGPOU,LIST
      INTEGER FILLIN,DIGIN,BOBSCP,BOBSCL
      INTEGER PLACE,CLEAR,ICOUNT
      INTEGER ICLTOT,IPLTOT
      CHARACTER*12 NAMES(60), SNAME(60)
      CHARACTER*12 NAMEB(15), BNAME(32)
      CHARACTER*12 NAMEM(15),MNAME(32)
      CHARACTER*12 NAMET(60),TNAME(60)
      CHARACTER*12 NAMEC(100),CNAME(100)
      CHARACTER*6 STR(60)
      CHARACTER*6 BSTR(32)
      CHARACTER*6 MSTR(32)
      CHARACTER*6 TSTR(60)
      CHARACTER*6 CSTR(100)
      CHARACTER*6 PSTR(20)
      CHARACTER LIN*6
      CHARACTER*12 DESCRP
      DIMENSION ISCAP(60),ISQTY(60)
      DIMENSION IBCAP(32),IBQTY(32)
      DIMENSION IMCAP(32),IMQTY(32)
      DIMENSION ITCAP(60),ITQTY(60)
      DIMENSION ICCAP(100),ICQTY(100)
      DIMENSION IPCAP(20),IPQTY(20),IFLCAP(20),IDGCAP(20),POSCAP(20)
      DIMENSION IYEAR(10)
      DIMENSION IREADU(10)
      COMMON JTPSNA(15)
      COMMON/S/ITYPE(60),HAUL(60),ITEMP(60),HTEMP(60)
      COMMON/A/ STR,ISCAP,ISQTY,ISVAR(60),NAMES
      COMMON/B/ BSTR,IBCAP,IBQTY,IBVAR(15),NAMEB
      COMMON/M/ MSTR,IMCAP,IMQTY,IMVAR(15),NAMEM
      COMMON/T/ TSTR,ITCAP,ITQTY,ITVAR(60),NAMET
      COMMON/T2/ISPED(60),ITESPD(60)
      COMMON/C/ CSTR,ICCAP,ICQTY,ICVAR(100),NAMEC
      COMMON/P/ PSTR,IPCAP,IPQTY,IPVAR(20),IFLTOT(20),IDGTOT(20),
     *POSTOT(20)
      COMMON/M2/IPLACE(20),ICLEAR(20),IMPLAT(20),ICLRT(20)
      OPEN (UNIT=16)
C
      WRITE(6,120)
  120 FORMAT(1X,'ENTER NUMBER OF YEARS YOU ARE CONSIDERING')
      WRITE(6,121)
  121 FORMAT(1X,'PRECEDE SINGLE DIGIT VALUES WITH A 0')
      READ(5,125)NUMYRS
  125 FORMAT(I2)
      ICOUNT=1
      DO 140 IS=1,NUMYRS
        WRITE(6,130)IS
  130   FORMAT(1X,'ENTER YEAR ',I2,' USING 2 DIGITS')
        READ(5,135)IYEAR(IS)
  135   FORMAT(I2)
C     ***************************************************************
C     *
C     * IREADU(IS) DETERMINES WHICH INVENTORY FILE TO READ. THESE
C     * FILES ARE ARRANGED BY YEARS.  THE IREADU STANDS FOR READ
C     * UNIT.  THE UNITS THAT THESE FILES ARE STORED IN BEGIN WITH
C     * UNIT 10.  THE RUNSTREAM OBVIOUSLY CAN BE MODIFIED AS REQUIRED
C     * FOR NEW INVENTORIES.
C     *
C     ***************************************************************
        IREADU(IS)=IYEAR(IS)-70
  140 CONTINUE
      WRITE(6,145)
  145 FORMAT(1X,'ENTER THE NUMBER OF TPSNS YOU ARE INTERESTED IN')
      WRITE(6,121)
      READ(5,150)NUMTIP
  150 FORMAT(I2)
C     IF(NUMTIP .EQ. 1)GOTO153
      DO 160 JS=1,NUMTIP
  153   WRITE(6,155)JS
   155  FORMAT(1X,'ENTER TPSN NUMBER ',I2,' USING 5 DIGITS')
        READ(5,157)JTPSNA(JS)
  157   FORMAT(I5)
C       IF(NUMTIP .EQ.1)THEN
C         JJ=1
C         GOTO 161
C       ENDIF
```

Figure E-I-16. Listing of Main CS/CSS Search Program with 12 Internal Subroutines (page 1 of 7 pages)

```
  160  CONTINUE
       DO 700 JJ=1,NUMTIP
       IF(JTPSNA(JJ).EQ.0)GOTO 700
  161  DO 600 II=1,NUMYRS
       IF(ICOUNT .GT. 1)GOTO 300
        M=0
   200 READ (21,210,END=220) LIN,DESCRP,K,ITYP,CARRY
   210 FORMAT(A6,2X,A12,2X,I2,2X,I1,2X,F3.2)
C      M IS A COUNTER THAT TIES THE LIN# WITH THE CAPBLTY CODE(ISCAP( ))
C
       M=M+1
       STR(M)=LIN
       SNAME(M)=DESCRP
       ISCAP(M)=K
       ITEMP(M)=ITYP
       HTEMP(M)=CARRY
C      WRITE(6,215)SNAME(M)
C 215  FORMAT(1X,A12)
       GOTO 200
C
   220  MB=0
   230   READ(22,235,END=240 ) LIN,DESCRP,K,IAFP
  235  FORMAT(A6,2X,A12,2X,I2,2X,I3)
        MB=MB+1
        BSTR(MB) = LIN
        BNAME(MB) = DESCRP
        IBCAP(MB) = K
       IBTAFP(MB)=IAFP
        GOTO 230
C
C
   240  MM=0
   250  READ(23,252,END=255 ) LIN,DESCRP,K,PLACE,CLEAR
   252  FORMAT(A6,2X,A12,2X,I2,2X,I4,2X,I3)
        MM=MM+1
        MSTR(MM) = LIN
        MNAME(MM) = DESCRP
        IMCAP(MM) = K
        IMPLAT(MM)=PLACE
        ICLRT(MM)=CLEAR
        GOTO 250
C
   255  MT=0
   257   READ(24,258,END=260 ) LIN,DESCRP,K,ISPEED,CLASS,IAFP
  258  FORMAT(A6,2X,A12,2X,I2,2X,I3,2X,F4.1,2X,I3)
        MT=MT+1
        TSTR(MT) = LIN
        TNAME(MT) = DESCPP
        ITCAP(MT) = K
       ITESPD(MT)=ISPEED
       VTCLAS(MT)=CLASS
       ITAFP(MT)=IAFP
        GOTO 257
C
   260 MC=0
   262   READ(25,210,END=264 ) LIN,DESCRP,K
        MC=MC+1
        CSTR(MC) = LIN
        CNAME(MC) = DESCRP
        ICCAP(MC) = K
        GOTO 262
   264 MP=0
   265 READ(26,267,END=300 ) LIN,RNPOSN,DIGIN,FILLIN,K
   267 FORMAT(A6,2X,F5.2,2X,I3,2X,I3,2X,J2)
C      WRITE(6,268)K
       MP=MP+1
C  268 FORMAT(1X,'K= ',J2)
       PSTR(MP) = LIN
       POSCAP(MP) = RNPOSN
       IDGCAP(MP)=DIGIN
       IFLCAP(MP)=FILLIN
       IPCAP(MP) = K
C      WRITE(6,269)PSTR(MP),POSCAP(MP),IDGCAP(MP),IFLCAP(MP),IPCAP(MP)
C  269 FORMAT(1X,A6,2X,F5.2,2X,I5,2X,I5,2X,J5)
       GOTO 265
C      THE BELOW READS THE DATA FILE
C
   300 READ(IREADU(II),310,END=500) ITPSNA,LIN,JOTY,ITP
 310   FORMAT(I5,67X,A6,40X,I7,1X,I3)
       KT = KT + 1
       IF(KT .EQ. 1) STP = ITP
C
```

Figure E-I-16. Listing of Main CS/CSS Search Program with 12 Internal Subroutines (page 2 of 7 pages)

```
C       THE FOLLOWING COMPARES TPSN# OF INT. W/TPSN# IN DATA FILE.
C       IF THEY ARE NOT THE SAME, THE NEXT LINE IN FILE IS COMPARED.
C
C
        IF(ITPSNA.NE.JTPSNA(JJ))GOTO 700
         CALL BRIDGE(JQTY,MB,BNAME,LIN)
        CALL MINES(JQTY,MM,MNAME,LIN)
        CALL TACMOB(JQTY,MT,TNAME,LIN)
        CALL COMMO(JQTY,MC,CNAME,LIN)
        CALL SANDT(JQTY,M,SNAME,LIN)
        CALL PROPOS(JQTY,MP,POSCAP,IDGCAP,IFLCAP,LIN)
        GOTO300
C
 500    CALL OUTPT1(STP,JJ,TCOUNT)
        CALL OUTPT2(ICLTOT,IPLTOT)
        CALL OUTPT3
        CALL OUTPT4
        CALL OUTPT5
        CALL OUTPT6(BOBSCP,BOBSCL,IOBCNT)
        CALL OUTPUT(STP,JJ,ICLTOT,IPLTOT,BOBSCR,BOBSCL,IOBCNT,
       *TCOUNT)
        DO 510 KK=1,NUMYRS
          REWIND IREADU(KK)
 510    CONTINUE
        DO 515 LL = 21,30,1
          REWIND LL
 515    CONTINUE
        DO 540 INITLZ=1,100
          IF(INITLZ.GT.60)GOTO 535
          IF(INITLZ.GT.32)GOTO 525
          IF(INITLZ.GT.20)GOTO 530
          IF(INITLZ.GT.15)GOTO 520
          IBVAR(INITLZ)=0
          IMVAR(INITLZ)=0
 520      IPVAR(INITLZ)=0
          BSETS(INITLZ)=0
 525      IBQTY(INITLZ)=0
 530      ITVAR(INITLZ)=0
          ISVAR(INITLZ)=0
  535     ICVAR(INITLZ)=0
 540    CONTINUE
        ICOUNT=ICOUNT+1
        KT=0
        NVEH=0
        NBRIDG=0
        IBFLAG=0
        IAVLB=0
 600    CONTINUE
 700    CONTINUE
        CLOSE (UNIT=16)
 750    STOP
        END
C
C       SUBROUTINE OUTPUT COMPARES LIN#S IN DATA FILE WITH LIN#S
C       IN INPUT FILES.  IF THE ARE THE SAME THE QTYS ARE ADDED.
C
C       ***************************************************************
        SUBROUTINE SANDT(JQTY,M,SNAME,LIN)
C       ***************************************************************
        CHARACTER*12 NAMES(60),SNAME(60)
C
C
        CHARACTER LIN*6
        CHARACTER*6 STR(60)
        DIMENSION ISCAP(60),ISQTY(60)
        COMMON/A/ STR,ISCAP,ISQTY,ISVAR(60),NAMES
        COMMON/S/ITYPE(60),HAUL(60),ITEMP(60),HTEMP(60)
        K=0
        DO 50 J=1,M
         IF(L IN .EQ. STR(J))K= ISCAP(J)
         IF(L IN .EQ. STR(J)) ISQTY(J)= ISQTY(J)+JQTY
         IF(LIN.EQ.STR(J))GOTO55
  50    CONTINUE
        IF(K.EQ.0)GOTO 100
 55     ISVAR(K)=ISVAR(K)+JQTY
        NAMES(K)=SNAME(J)
        ITYPE(K)=ITEMP(J)
        HAUL(K)=HTEMP(J)
C       WRITE(6,90)ISVAR(K),NAMES(K)
C 90    FORMAT(1X,'ISVAR= ',I7,3X,'NAME= ',A12)
 100    RETURN
         END
```

Figure E-I-16. Listing of Main CS/CSS Search Program with 12 Internal Subroutines (page 3 of 7 pages)

```
C
C          ***************************************************************
           SUBROUTINE MINES(JQTY,MM,MNAME,LIN)
C          ***************************************************************
C
           CHARACTER*12 NAMEM(15),MNAME(32)
           CHARACTER*6 MSTR(32)
           CHARACTER LIN*6
           DIMENSION IMCAP(32),IMQTY(32)
           COMMON/M/ MSTR,IMCAP,IMQTY,IMVAR(15),NAMEM
           COMMON/M2/IPLACE(20),ICLEAR(20),IMPLAT(20),ICLRT(20)
           K=0
           DO 50 J=1,MM
             IF(LIN .EQ. MSTR(J))K=IMCAP(J)
             IF(LIN .EQ. MSTR(J))IMQTY(J)=IMQTY(J)+JQTY
             IF(LIN.EQ.MSTR(J))GOTO55
    50     CONTINUE
           IF(K.EQ.0)GOTO 100
    55     IMVAR(K)=IMVAR(K)+JQTY
           NAMEM(K)=MNAME(J)
           IPLACE(K)=IMPLAT(J)
           ICLEAR(K)=ICLRT(J)
           WRITE(6,90)IMVAR(K),NAMEM(K),IPLACE(K),ICLEAR(K)
    90     FORMAT(1X,'IMVAR=',I7,3X,'NAME=',A12,3X,'PLAC=',I4,3X,
          *'CLEAR=',I4)
   100     RETURN
            END
C
C          ***************************************************************
            SUBROUTINE BRIDGE(JQTY,MB,BNAME,LIN)
C          ***************************************************************
C
           INCLUDE BPDAT,LIST
           INCLUDE BOGROU,LIST
            CHARACTER*12 NAMEB(15), BNAME(32)
            CHARACTER*6 BSTR(32)
            CHARACTER LIN*6
            DIMENSION IBCAP(32),IBQTY(32)
            COMMON/B/ BSTR,IBCAP,IBQTY,IBVAR(15),NAMEB
C
           K=0
           DO 50 J=1,MB
             IF(LIN .EQ. BSTR(J))K=IBCAP(J)
             IF(LIN .EQ. BSTR(J))IBQTY(J)=IBQTY(J)+JQTY
             IF(LIN.EQ.BSTR(J))GOTO55
    50     CONTINUE
    55     IF(K.EQ.0)GOTO 100
           IBVAR(K)=IBVAR(K) +JQTY
           NAMEB(K)=BNAME(J)
           IBTNUM(K)=IBTAFP(J)
   100     RETURN
           END
C
C
C          *********************************************************
           SUBROUTINE TACMOB(JQTY,MT,TNAME,LIN)
C          *********************************************************
C
           INCLUDE BPDAT,LIST
           INCLUDE BOGROU,LIST
           CHARACTER*12 NAMET(60), TNAME(60)
           CHARACTER*6 TSTR(60)
           CHARACTER LIN*6
           DIMENSION ITCAP(60),ITQTY(60)
           COMMON/T/ TSTR,ITCAP,ITQTY,ITVAR(60),NAMET
           COMMON/T2/ISPED(60),ITESPD(60)
C
           K=0
           DO 50 J=1,MT
             IF(LIN .EQ. TSTR(J))THEN
               K=ITCAP(J)
               ITQTY(J)=ITQTY(J)+JQTY
               GOTO55
           ENDIF
    50     CONTINUE
           IF(K.EQ.0)GOTO 100
    55     ITVAR(K)=ITVAR(K)+JQTY
           NAMET(K)=TNAME(J)
           ISPED(K)=ITESPD(J)
           IF(ITESPD(J).GT.130)GOTO 100
           NVEH=NVEH+1
           VNUM(NVEH)=ITAFP(J)
```

Figure E-I-16. Listing of Main CS/CSS Search Program with 12 Internal Subroutines (page 4 of 7 pages)

```
      VCLAS(NVEH)=VTCLAS(J)
      VDESC(NVEH)=TNAME(J)
      NV(NVEH)=FLOAT(JQTY)
C     WRITE(6,90)ITVAR(K),NAMET(K)
C 90  FORMAT(1X,'ITVAR= ',I7,3X,'NAMET= ',A12)
 100  RETURN
       END
C
C     ****************************************************************
       SUBROUTINE COMMO(JQTY,MC,CNAME,LIN)
C     ****************************************************************
C
       CHARACTER*12 NAMEC(100), CNAME(100)
       CHARACTER*6 CSTR(100)
       CHARACTER LIN*6
       DIMENSION ICCAP(100),ICQTY(100)
       COMMON/C/ CSTR,ICCAP,ICQTY,ICVAR(100),NAMEC
C
      K=0
      DO 50 J=1,MC
        IF(LIN .EQ. CSTR(J))K=ICCAP(J)
        IF(LIN .EQ. CSTR(J))ICQTY(J)=ICQTY(J)+JQTY
        IF(LIN.EQ.CSTR(J))GOTO55
   50 CONTINUE
      IF(K.EQ.0)GOTO 100
 55   ICVAR(K)=ICVAR(K)+JQTY
      NAMEC(K)=CNAME(J)
C     WRITE(6,90)ICVAR(K),NAMEC(K)
C 90  FORMAT(1X,'ICVAR= ',I7,3X,'NAMEC= ',A12)
 100  RETURN
       END
C
C     ****************************************************************
       SUBROUTINE PPOPOS(JQTY,MP,POSCAP,IDGCAP,IFLCAP,LIN)
C     ****************************************************************
C
       CHARACTER*6 PSTR(20)
       CHARACTER LIN*6
       DIMENSION IPCAP(20),IPQTY(20),IFLCAP(20),IDGCAP(20),POSCAP(20)
       COMMON/P/ PSTR,IPCAP,IPQTY,IPVAR(20),IFLTOT(20),IDGTOT(20),
     *POSTOT(20)
C
      K=0
      DO 50 J=1,MP
        IF(LIN .EQ. PSTR(J))K=IPCAP(J)
        IF(LIN .EQ. PSTR(J))IPQTY(J)=IPQTY(J)+JQTY
        IF(LIN.EQ.PSTR(J))GOTO55
   50 CONTINUE
      IF(K.EQ.0)GOTO 100
 55   IPVAR(K)=IPVAR(K)+JQTY
      POSTOT(K)=POSCAP(J)
      IDGTOT(K)=IDGCAP(J)
      IFLTOT(K)=IFLCAP(J)
C     WRITE(6,90)IPVAR(K),POSTOT(K),IDGTOT(K),IFLTOT(K)
C   90 FORMAT(1X,'IPVAR=',I7,3X,'POSTOT= ',F5.2,1X,'IDGTOT',I5,2X,
C    *'IFLTOT=',I5)
 100  RETURN
       END
C     SUBROUTINE OUTPT1 SIMPLY PROVIDES FOR HARD COPY OUTPUT.
C
C     ****************************************************************
      SUBROUTINE OUTPT1(STP,JJ,TCOUNT)
C     ****************************************************************
C
      INTEGER STP,TCOUNT
      CHARACTER*12 NAMES(60)
      CHARACTER*6 STR(60)
       DIMENSION ISCAP(60),ISQTY(60)
      COMMON JTPSNA(15)
       COMMON/A/ STR,ISCAP,ISQTY,ISVAR(60),NAMES
      TCOUNT=0
      WRITE(6,503)
      WRITE(6,501) STP,JTPSNA(JJ)
      WRITE(6,503)
 500   FORMAT('1',1X,'SUPPLY AND TRANSPORTATION DATA')
       WRITE(6,500)
501   FORMAT(1H1,1X,'TIME PERIOD = ',I3,2X,'TPSNA = ',J5//)
 503  FORMAT(1X,'***********************************************')
505   DO 600 K=1,60
C     WRITE(6,507)ISVAR(K)
C507    FORMAT(1X,I7)
       IF(ISVAR(K).EQ.0)GOTO600
```

Figure E-I-16. Listing of Main CS/CSS Search Program with 12 Internal Subroutines (page 5 of 7 pages)

```
          TCOUNT=TCOUNT+1
          WRITE(6,510) ISVAR(K),NAMES(K)
 510      FORMAT(1X,I7,1X,A12)
  600   CONTINUE
        RETURN
        END
C
C      PROVIDES OUTPUT FOR MINE/COUNTERMINE DATA
C
C       *****************************************************************
        SUBROUTINE  OUTPT2(ICLTOT,IPLTOT)
C       *****************************************************************
C
        DIMENSION IMCAP(32), IMQTY(32)
        CHARACTER*12 NAMEM(15)
        CHARACTER MSTR(32)*6
C
        COMMON/M/ MSTR,IMCAP,IMQTY,IMVAR(15),NAMEM
        COMMON/M2/IPLACE(20),ICLEAR(20),IMPLAT(20),ICLRT(20)
        IPLTOT=0
        ICLTOT=0
C
        WRITE(6,500)
 500    FORMAT('1',1X,'MINE/COUNTERMINE DATA')
        WRITE(6,501)
  501   FORMAT(1X,'**************************************************')
        WRITE(6,502)
 502    FORMAT(' ',1X,'MINEPLACEMENT')
        WRITE(6,503)
 503    FORMAT(' ',1X,'--LIN#-',3X,'----ITEM----',2X,'QTY',2X,'MINES/HR')
C
        DO 600 K = 1,15
          IF(IMVAR(K) .EQ. 0) GO TO 600
          IF(IPLACE(K) .EQ. 0) GO TO 600
          IPLTOT=IPLTOT+(IMVAR(K)*IPLACE(K))
C
C
          WRITE(6,504) MSTR(K),NAMEM(K),IMQTY(K),IPLACE(K)
 504      FORMAT(' ',1X,A6,2X,A12,2X,I5,2X,I4)
C
 600    CONTINUE
C
        WRITE(6,620) IPLTOT
 620    FORMAT(' ',1X,'EQUIPMENT MINELAYING CAP PER HOUR = ',I7)
        WRITE(6,630)
 630    FORMAT(' ',1X,'MINECLEARING DATA')
        WRITE(6,501)
        DO 700 K = 1,15
           IF(IMVAR(K) .EQ. 0) GO TO 700
           IF(ICLEAR(K) .EQ. 0) GO TO 700
        ICLTOT=ICLTOT+(IMVAR(K)*ICLEAR(K))
           WRITE(6,504) MSTR(K),NAMEM(K),IMQTY(K),ICLEAR(K)
 700    CONTINUE
        WRITE(6,710)ICLTOT
  710   FORMAT(1X,'EQUIPMENT MINECLEARING CAPACITY PER HOUR= ',I7)
        RETURN
        END
C
C      PROVIDES OUTPUT FOR BRIDGING DATA
C
C       *****************************************************************
        SUBROUTINE OUTPT3
C       *****************************************************************
C
        INCLUDE BDGROU,LIST
        INCLUDE BPDAT,LIST
        CHARACTER*12 NAMEB(15)
        CHARACTER*6 BSTR(32)
        DIMENSION IBCAP(32),IBQTY(32)
        COMMON/B/ BSTR,IBCAP,IBQTY,IBVAR(15),NAMEB
C
        WRITE(6,500)
 500    FORMAT('1',1X,'BRIDGING DATA')
        WRITE(6,502)
 502    FORMAT(' ',1X,'--LIN#-'2X,'----ITEM----',2X,'QTY')
C
        DO 600 K = 1,15
           IF(IBVAR(K) .EQ. 0) GO TO 600
        NBRIDG=NBRIDG+1
        IF(BSTR(K).EQ.'K97376')THEN
          BSETS(NBRIDG)=FLOAT(IBQTY(K))/30.
        ELSE
```

Figure E-I-16. Listing of Main CS/CSS Search Program with 12 Internal Subroutines (page 6 of 7 pages)

```
        BSETS(NBRIDG)=FLOAT(IBQTY(K))
      ENDIF
      BNUM(NBRIDG)=IBTNUM(K)
      BDESC(NBRIDG)=NAMEB(K)
         WRITE(6,510) BSTP(K),NAMEB(K),IBQTY(K)
510      FORMAT(1X,A6,3X,A12,2X,I3)
600   CONTINUE
      RETURN
      END
C
C     ****************************************************************
      SUBROUTINE OUTPT4
C     ****************************************************************
C
      CHARACTER*12 NAMET(60)
      CHARACTER*6 TSTR(60)
      DIMENSION ITCAP(60),ITQTY(60)
      COMMON/T/ TSTP,ITCAP,ITQTY,ITVAR(60),NAMET
C
      WRITE(6,500)
500   FORMAT('1',1X,'TACTCAL MOBILITY DATA',/)
      WRITE(6,505)
505   FORMAT(' ',1X,'SPEED (KPH)',2X,'QUANTITY')
C
      DO 600 K=1,60
       IF (ITVAR(K) .EQ. 0)GOTO 600
       WRITE(6,510)NAMET(K),ITVAR(K)
510    FORMAT(1X,A12,4X,I7)
600   CONTINUE
      RETURN
      END
C
C
C     ***************************************************************
C     *                                                             *
C     *                     SUBROUTINE OUTPT6                       *
C      ***************************************************************
C     *  SUBROUTINE OUTPT6 PRODUCES A PRINTOUT OF THE FOLLOWING:     *
C      ***************************************************************
C
C     *    A.  A LIST OF DIVISIONAL EQUIPMENT CAPABLE OF OBSTACLE/    *
C     *        PROTECTIVE POSITION CREATION AND ITS DIG IN RATE IN    *
C     *        TERMS OF PROTECTIVE POSITIONS PER HOUR.                *
C      ***************************************************************
       SUBROUTINE OUTPT6(BOBSCR,BOBSCL,IOBCNT)
C      ***************************************************************
C
       INTEGER BOBSCR,BOBSCL
       CHARACTER*6 PSTR(20)
        DIMENSION IPCAP(20),IPQTY(20)
        COMMON/P/ PSTR,IPCAP,IPQTY,IPVAR(20),IFLTOT(20),IDGTOT(20),
     *POSTOT(20)
C
       WRITE(6,500)
500    FORMAT('1',1X,'PROTECTIVE POSITION DATA',/)
       WRITE(6,505)
505    FORMAT(' ',1X,'POSITIONS/HR',4X,'QUANTITY')
C
       IOBCNT=0
       BOBSCR=0
       BOBSCL=0
C
       DO 600 K=1,20
        IF (IPVAR(K) .EQ. 0)GOTO 600
        BOBSCR = BOBSCR + ( IDGTOT(K) * IPVAR(K) )
        IF(IDGTOT(K).NE.0)IOBCNT=IOBCNT+1
        BOBSCL = BOBSCL + (  IFLTOT(K) * IPVAR(K) )
        IF(POSTOT(K).EQ.0.)GO TO 600
        WRITE(6,510)POSTOT(K),IPVAR(K)
510     FORMAT(3X,F5.2,4X,I5)
600    CONTINUE
C
       WRITE(6,520)BOBSCR
 520   FORMAT(1X,'BLUE OBSTACLE CREATION RATE = ',I7)
       WRITE(6,530)BOBSCL
 530   FORMAT(1X,'BLUE OBSTACLE CLEARANCE RATE = ',I7)
       RETURN
       END
C
C      ****************************************************************
       SUBROUTINE OUTPT5
C      ****************************************************************
C
       CHARACTER*12 NAMEC(100)
       CHARACTER*6 CSTP(100)
       DIMENSION ICCAP(100),ICQTY(100)
       COMMON/C/ CSTP,ICCAP,ICQTY,ICVAR(100),NAMEC
C
       WRITE(6,500)
500    FORMAT('1',1X,'COMMUNICATIONS DATA',/)
       WRITE(6,505)
505    FORMAT(' ',1X,'EMMITER TYPE',2X,'QUANTITY')
C
       DO 600 K=1,20
        IF (ICVAR(K) .EQ. 0)GOTO 600
        WRITE(6,510)NAMEC(K),ICVAR(K)
510     FORMAT(1X,A12,4X,I7)
600    CONTINUE
       RETURN
       END
```

Figure E-I-16. Listing of Main CS/CSS Search Program with 12 Internal Subroutines (page 7 of 7 pages)

```
      SUBROUTINE OUTPUT(STP,JJ,ICLTOT,IPLTOT,BOBSCR,BOBSCL,IOBCNT,
     *TCOUNT)
      INCLUDE BDGPOU,LIST
      INCLUDE BPDAT,LIST
      INTEGER GNDCNT,AIRCNT,BOBSCR,BOBSCL,TCOUNT
      INTEGER IPLTOT,ICLTOT,STP
      INTEGER NUMENG,NUMPER
      REAL BMAINF
      CHARACTER*6 POSTUR(4)
      CHARACTER*6 TSTR(60)
      CHARACTER*6 CSTR(100)
      CHARACTER*6 PSTR(20)
      CHARACTER*6 STR(60)
      CHARACTER*5 IDENT,IDNAME
      DIMENSION ISCAP(60),ISQTY(60)
      DIMENSION ICCAP(100),ICQTY(100)
      DIMENSION IPCAP(20),IPQTY(20),IFLCAP(20),IDGCAP(20),POSCAP(20)
      DIMENSION ITCAP(60),ITQTY(60)
      CHARACTER*12 NAMES(60), SNAME(60)
      CHARACTER*12 NAMET(60),TNAME(60)
      CHARACTER*12 NAMEC(100),CNAME(100)
      DIMENSION RMAINT(4),RAMMO(4),RPOL(4),BSURGE(4),RSURGE(4)
      DIMENSION REDBRG(4)
      COMMON JTPSNA(15)
      COMMON/S/ITYPE(60),HAUL(60),ITEMP(60),HTEMP(60)
      COMMON/A/ STR,ISCAP,ISQTY,ISVAR(60),NAMES
      COMMON/B/ BSTR,IBCAP,IBQTY,IBVAR(15),NAMEB
      COMMON/M/ MSTR,IMCAP,IMQTY,IMVAR(15),NAMEM
      COMMON/T/ TSTR,ITCAP,ITQTY,ITVAR(60),NAMET
      COMMON/T2/ISPED(60),ITESPD(60)
      COMMON/C/ CSTR,ICCAP,ICQTY,ICVAR(100),NAMEC
      COMMON/P/ PSTR,IPCAP,IPQTY,IPVAR(20),IFLTOT(20),IDGTOT(20),
     *POSTOT(20)
      COMMON/M2/IPLACE(20),ICLEAR(20),IMPLAT(20),ICLRT(20)
      POSTUR(1)='RAPD'
      POSTUR(2)='STATIC'
      POSTUR(3)='RADE'
      POSTUR(4)='RAPD'
      RMAINT(1)=.3072
      RMAINT(2)=.8781
      RMAINT(3)=.4398
      RMAINT(4)=.5857
      RAMMO(1)=2223.53
      RAMMO(2)=1866.92
      RAMMO(3)=1866.92
      RAMMO(4)=2223.53
      RPOL(1)=105.7
      RPOL(2)=42.29
      RPOL(3)=147.98
      RPOL(4)=63.42
C
C
C     ************************************************************
C     * 1984 SURGE FACTORS FOR MICAFII STUDY ARE FOLLOWING
C     ************************************************************
C
C
      BSURGE(1)=2.65
      BSURGE(2)=1.0
      BSURGE(3)=   9.65
      BSURGE(4)=1.47
      RSURGE(1)=0.52
      RSURGE(2)=1.0
      RSURGE(3)=0.16
      RSURGE(4)=1.17
      REDBRG(1)=.88
      REDBRG(2)=.85
      REDBRG(3)=.88
      REDBRG(4)=.85
      IBFLAG=0
      CALL PRPROG(BMAINF,NUMENG,NUMPER,STP,JJ)
      DO 900 RGLOOP=1,4
      AIRCNT=0
      GNDCNT=0
      IF(RGLOOP.NE.1)GOTO4
    2 READ(28,3,END=4)KTPSN,IDNAME
    3  FORMAT(I5,5X,A5)
       IF (KTPSN.EQ.JTPSNA(JJ))THEN
```

Figure E-I-17. Listing of Output Subroutine to Search Program with One Internal Subroutine (page 1 of 6 pages)

```
            IDENT=IDNAME
            GOTO 4
          ENDIF
        GOTO 2
      4 WRITE(20,5)BGLOOP,POSTUR(BGLOOP),IDENT,INT(STP/10)
      5 FORMAT(9X,I2,8X,A6,5X,A5,I2)
        WRITE(20,10)
     10 FORMAT(9X,'F',9X,'1.0',7X,'1.0')
        WRITE(20,20)
     20 FORMAT(9X,'77.10',5X,'227.60',4X,'3.02')
C
C       * NOTE! GNDCNT IS THE NUMBER OF GROUPS OF TACTICAL VEHICLES,
C       * WHICH ARE GROUPED BY COMMON TACTICAL SPEEDS.GNDCNT REPRESENTS
C       *GROUND VEHICLE COUNT
C
        DO 25 I=1,60
          IF(ISPED(I).EQ.0)GOTO 25
          IF(ISPED(I).GT.130)GOTO23
          GNDCNT=GNDCNT+1
          GOTO 25
     23 AIRCNT=AIRCNT+1
     25 CONTINUE
        WRITE(20,30)GNDCNT
     30 FORMAT(8X,I2)
        DO 50 I=1,60
            IF(ISPED(I).EQ.0)GO TO 50
            IF(ISPED(I).GT.130)GO TO 50
            WRITE(20,40)ITVAR(I),ISPED(I)
     40     FORMAT(8X,I4,6X,I3)
     50 CONTINUE
C
        WRITE(20,55)AIRCNT
     55 FORMAT(9X,I2)
        DO 70 I=1, 60
          IF(ISPED(I).EQ. 0)GO TO 70
          IF(ISPED(I).LE. 130)GOTO 70
          WRITE(20,60)ITVAR(I),ISPED(I)
     60   FORMAT(8X,I4,6X,I3)
     70 CONTINUE
C
C       * THE FOLLOWING IS THE RED OBSTACLE CLEARING RATE, AND MAY
C       * VARY DEPENDING ON THE TYPE OF OPPOSING FORCE.
C
        WRITE(20,80)
     80 FORMAT(9X,'3426')
C
C       * THE FOLLOWING, BOBSCR, REPRESENTS THE BLUE OBSTACLE
C       * CREATION RATE (CUBIC METERS/HOUR).
C
        WRITE(20,90)BOBSCR
     90 FORMAT(8X,I5)
        WRITE(20,10)
C
C       * THE FOLLOWING ARE FIGURES THAT DEAL WITH RED TACTICAL MOB-
C       * ILITY.  THE FIRST FIGURE IS THE WEIGHTED AVG GROUND VEHICLE
C       * SPEED.  THE SECOND FIGURE IS THE WEIGHTED AVG HELICOPTER
C       * SPEED.  THE THIRD FIGURE IS THE SECOND FIGURE DIVIDED BY
C       * FIRST.  THESE NUMBERS WILL VARY WITH THE OPPOSING FORCE.
C
C
        WRITE(20,100)
    100 FORMAT(9X,'77.10',5X,'227.60',4X,'3.94')
C
C       * THE FOLLOWING IS THE NUMBER OF GROUPS OF RED GROUND TAC
C       * VEHICLES, GROUPED BY TACTICAL VEHICLE SPEED, FOLLOWED BY
C       * THE NUMBER OF VEHICLES AND THEIR SPEEDS IN KPH.
C
        WRITE(20,110)
    110 FORMAT(9X,'8'/9X,'33',8X,'85'/9X,'330',7X,'80'/9X,'148',7X,
       *'75'/9X,'88',8X,'70'/9X,'8',9X,'62.5'/9X,'362',7X,'60'/9X,'170',
       *7X,'50'/9X,'36',8X,'44')
C
C      * THE FOLLOWING IS THE NUMBER OF GROUPS OF RED HELICOPTERS,
C      * THEIR QUANTITIES (LHS) AND THEIR SPEEDS IN KPH.
C
        WRITE(20,120)
    120 FORMAT(9X,'2'/9X,'6',9X,'215'/9X,'6',9X,'315')
C
C      * THE FOLLOWING IS THE BLUE OBSTACLE CLEARANCE RATE IN TERMS
C      * OF CUBIC METERS OF SOIL PER HOUR.  IT IS COMPUTED VIA SUB-
C      * ROUTINES PROPSN AND OUTPT6.
C
```

Figure E-I-17. Listing of Output Subroutine to Search Program with One Internal Subroutine (page 2 of 6 pages)

```
      WRITE(20,130)BOBSCL
  130 FORMAT(8X,I5)
C
C     * THE FOLLOWING IS THE RED OBSTACLE CREATION RATE AND
C     * VARIES WITH THE OPPOSING FORCE.
C
      WRITE(20,140)
  140 FORMAT(9X,'8482')
C
C     * THE FOLLOWING DEALS WITH BLUE BRIDGING FACTOR.  THE VALUE IN
C     * 3RD COLUMN VARIES FROM DIVISION TO DIVISION, DEPENDING ON
C     * HAND BRIDGING ASSETS AND DIVISIONAL ASSETS & DIVISIONAL
C     * VEHICLE.  THIS VALUE IS DETERMINED BY THE BRIDGE SUB-
C     * ROUTINE, OUTPT AND A BRIDGE CAPABILITY FACTOR ROUTINE.
C     *
      BCAP=0
      IF(NBRIDG.NE.0)CALL BRGPRE
      REWIND 30
      WRITE(20,150)BCAP
  150 FORMAT(9X,'T',9X,'1.0',7X,F8.3)
C
C     * THE FOLLOWING IS AS ABOVE, EXCEPT FOR RED FORCES, AND WAS
C     * DETERMINED VIA THE 'BEAN COUNT' METHOD.  IT VARIES AS THE
C     * RED FORCES VARY.
C
      WRITE(20,160)REDBRG(BGLOOP)
  160 FORMAT(9X,'T',9X,'1.0',7X,F4.3)
      WRITE(20,10)
C
C     * THE FOLLOWING CONCERNS MINEPLACEMENT CAPABILITY. THE 1ST
C     * COLUMN FIGURES REPRESENT BLUE EQUIPMENT MINELAYING CAPA-
C     * BILITY (IPLTOT).  THE 2ND FIGURE REPRESENTS THE NUMBER OF
C     * DIVISIONAL PERSONNEL (E-7 AND BELOW) THAT HVE BEEN TRAIN-
C     * ED TO EMPLACE MINEFIELDS (BASED ON MOS).  THIS FIGURE
C     * CURRENTLY MUST BE PLACED IN THE OUTPUT FILE SEPARATELY
C    * BUT CAN BE AUTOMATED.  THE 3RD FIGURE (28) REPRESENTS THE
C     * NUMBER OF LANES REQUIRED IN A RED DIVISION FRONT.  THIS
C     * WILL PROBABLY REMAIN CONSTANT FOR ALL RED DIVISIONS, BUT
C     * SHOUULD PROBABLY BE VALIDATED WITH CHANGES IN RED FORCE
C     * STRUCTURE.
C     * THE 4TH FIGURE REPPRESENTS LANE WIDTH. THE 5TH FIGURE
C     * REPRESENTS THE WIDTH OF A RED DIVISION FRONT IN METERS.
C     * IT SHOULD ALSO BE VALIDATED WITH THE TYPE OF RED DIVISION
C     * AND THE TYPE OF TERRAIN THE BATTLE IS FOUGHT IN.
C
      WRITE(20,180)IPLTOT,NUMENG
  180 FORMAT(7X,I5,7X,I3,7X,'28',8X,'3.736',5X,'12500')
C
C
C      *THE FOLLOWING FIGURE REPRESENTS RED EQUIPMENT MINECLEAR-
C      *ING CAPABILITY, AND WILL VARY WITH CHANGES IN RED FORCES
C
C
      WRITE(20,190)
  190 FORMAT(9X,'573.14')
      WRITE(20,10)
C
C
C     * THE FOLLOWING FIGURES ARE SIMILAR TO THE ABOVE, BUT FOR
C     * THE RED FORCES. 1ST FIGURE IS RED M.F. EQUIPMENT LAYING
C     * CAPABILITY. 2ND FIGURE IS # OF PERSONNEL CAPABLE OF M.F.
C     * INSTALLATION.  THE 3RD FIGURE REPRESENTS # OF LANES RE-
C     * QUIRED PER BLUE DIVISION FRONT.  THE 5TH FIGURE REPRE-
C     * SENTS THE WIDTH OF AN U.S. DIVISION FRONT.
C
C
      WRITE(20,200)
  200 FORMAT(9X,'4800',6X,'0',9X,'12',8X,'8',9X,'30000')
C
C
C     * THE FOLLOWING REPRESENTS TOTAL BLUE MINECLEARING EQUIP
C     * CAPABILITY.
C
C
      WRITE(20,210)ICLTOT
  210 FORMAT(9X,I5)
      WRITE(20,10)
C
C
C     * THE FOLLOWING REPRESENTS THE NUMBER OF TYPES OF BLUE
C     * PROTECTIVE POSITION CREATION EQUIPMENT.
C
```

Figure E-I-17. Listing of Output Subroutine to Search Program with One Internal Subroutine (page 3 of 6 pages)

```
C
      WRITE(20,230)IOBCNT
  230 FORMAT(8X,I2)
C
C
C     * THE LEFT HAND COLUMN OF THE FOLLOWING REPRESENTS THE
C     * QUANTITY OF EQUIPMENT, AND THE RIGHT COLUMN REPRESENTS
C     * THE NUMBER OF PROTECTIVE POSITIONS PER HOUR IT CAN
C     * CREATE.
C
C
      DO 240 K=1,20
        IF((POSTOT(K).EQ.0.).OR.(IPVAR(K).EQ.0))GO TO 240
        WRITE(20,235)IPVAR(K),POSTOT(K)
  235   FORMAT(9X,I2,7X,F5.2)
  240 CONTINUE
C
      WRITE(20,10)
C
C     *************************************************************
C     * THE NEXT VALUE REPRESENTS THE # OF GROUPS OF RED ENGI-
C     * NEER EQUIPMENT CAPABLE OF CREATING PROTECTIVE POSITIONS.
C     * FOLLOWING THAT VALUE ARE TWO COLUMNS REPRESENTING THE
C     * QUANTITY OF THE EQUIPMENT IN EACH GROUP AND THE # OF
C     * POSITIONS THAT CAN BE CONSTRUCTED PER HOUR PER PIECE
C     * OF EQUIPMENT.  THE # OF GROUPS OF EQUIPMENT, THEIR QUAN-
C     * TITIES AND CREATION RATES CAN BE EXPECTED TO CHANGE IF
C     * THE RED FORCE CHANGES.
C     *************************************************************
C
      WRITE(20,250)
  250 FORMAT(9X,'6'/9X,'8',9X,'7.47'/9X,'12',8X,'3.88'/9X,'12',
     *8X,'1.91'/9X,'2',9X,'2.33'/9X,'12',8X,'2.95'/9X,'2',9X,
     *'1.40')
C
      WRITE(20,10)
C
C     *************************************************************
C     * THE FOLLOWING DEALS WITH COMMUNICATIONS AND JAMMING.
C     * THE INTEGER VALUES ON THE LEFT REPRESENT RADIO QUANTI-
C     * TIES.THE INTEGER VALUES ON THE RIGHT REPRESENT THE #
C     * OF RED JAMMERS.  THE REAL NUMBERS IN BETWEEN ARE MAJIC
C     * NUMBERS.
C     *************************************************************
C
      WRITE(20,260)ICVAR(1)
  260 FORMAT(8X,I5,6X,'0.685',5X,'1.37685',3X,'0.8318',4X,'1')
C
      WRITE(20,262)
  262 FORMAT(9X,'81.5',6X,'5')
C
      WRITE(20,264)ICVAR(2)
  264 FORMAT(8X,I3,6X,'0.02',6X,'0.02394',3X,'0.0004',4X,'1')
C
      WRITE(20,266)
  266 FORMAT(9X,'7.1',7X,'6')
C
C     *************************************************************
C     * THE FOLLOWING DEALS WITH MULTI-CHANNEL.
C     *************************************************************
C
      WRITE(20,268)ICVAR(4)
  268 FORMAT(9X,I4,6X,'0.02'6X,'0.02337',3X,'0.0004',4X,'2')
      WRITE(20,270)
  270 FORMAT(9X,'1.714',6X,'3'/9X,'1.714',6X,'1')
C
      WRITE(20,272)ICVAR(3)
  272 FORMAT(8X,I2,9X,'1.00',6X,'0.19',6X,'0.1674',4X,'1')
C
      WRITE(20,273)
  273 FORMAT(9X,'1.25',6X,'4')
C
C
      WRITE(20,10)
C
C     *************************************************************
C     * THE FOLLOWING DEALS WITH AN/TLQ 15
C     *************************************************************
C
      WRITE(20,274)ICVAR(12)
  274 FORMAT(9X,'1.0',7X,'0.86',6X,'1.0',7X,'0.1016',4X,'1'/9X,
     *'0.1433',3X,I2)
```

Figure E-I-17. Listing of Output Subroutine to Search Program with One Internal Subroutine (page 4 of 6 pages)

```
      WRITE(20,275)
 275  FORMAT(9X,'1.0',7X,'0.685',5X,'1.0',7X,'0.4566',4X,'5')
C
C     ***********************************************************
C     * THE FOLLOWING DEALS WITH AN/TLQ 34, AN/TLQ 17, PIRANA,
C     * AN/MLQ 33,AND AN/GLQ-3
C     ***********************************************************
C
      WRITE(20,276)ICVAR(06),ICVAR(07),ICVAR(08),ICVAR(09),
     *ICVAR(11)
 276  FORMAT(9X,'0.0274',3X,I2/9X,'0.0274',4X,I2/9X,'0.00913',
     *3X,I2/9X,'.0183',4X,I2/9X,'.00834',3X,I2)
C
C     ***********************************************************
C     *THE FOLLOWING DEALS WITH AN/ALQ 151 AND UAVVHF
C     ***********************************************************
C
      WRITE(20,278)ICVAR(10),ICVAR(13)
 278  FORMAT(9X,'1.0',7X,'0.315',5X,'1.0',7X,'0.2100',4X,'2'/9X,
     *'.0228',5X,I2/9X,'.00365',4X,I2)
C
C     ***********************************************************
C     *THE FOLLOWING DEALS WITH BLUE AV RADAR
C     ***********************************************************
C
      WRITE(20,280)ICVAR(14)
 280  FORMAT(9X,'1.0',7X,'0.40',6X,'1.0', 7X,'.2317',5X,'1'/9X,
     *'.00344',3X,I2)
C
      WRITE(20,10)
C
      WRITE(20,290)NUMPER,NUMPER,RSURGE(BGLOOP)
 290  FORMAT(9X,I5,4X,I5,4X,F5.2,7X,'5.90',6X,
     *'1.73')
      WRITE(20,292)
 292  FORMAT(9X,'5.65',6X,'1.2',7X,'160.0',5X,'.5')
      WRITE(20,10)
C
C     ***********************************************************
C     * THE FOLLOWING DEALS WITH RED DIVISION STRENGTH/MEDICAL
C     ***********************************************************
C
      WRITE(20,294)RSURGE(BGLOOP)
 294  FORMAT(9X,'11920',4X,'11920',4X,F5.2,7X,'18.0',6X,'5.1')
C
      WRITE(20,296)
 296  FORMAT(9X,'6.00',6X,'1.7',7X,'500.0',5X,'.371')
C
C
      WRITE(20,300)BMAINF
 300  FORMAT(9X,'T',9X,'1.0',7X,F6.4)
C
C     ***********************************************************
C     *THE FOLLOWING DEALS WITH THE RED MAINTENANCE FACTOR,
C     * AND VARIES WITH POSTURE
C
C     ***********************************************************
C
      WRITE(20,310)RMAINT(BGLOOP)
 310  FORMAT(9X,'T',9X,'1.0',7X,F6.4)
C
      WRITE(20,10)
C
C     ***********************************************************
C     * THE FOLLOWING DEALS WITH S AND T ASSETS AND CAPABILITIES
C     ***********************************************************
C
      WRITE(20,315)TCOUNT
 315  FORMAT(9X,I2)
C
      DO 330 ISTCNT=1,60
        IF(ISVAR(ISTCNT).EQ.0)GOTO330
        WRITE(20,320)ISVAR(ISTCNT),HAUL(ISTCNT),ITYPE(ISTCNT)
 320    FORMAT(8X,I4,6X,'1.0',7X,F8.2,2X,I1)
 330  CONTINUE
C
      WRITE(20,340)
 340  FORMAT(9X,'8')
C
      DO 410 INDIVD=1,8
        IF (BGLOOP.EQ.2) GOTO 355
        IF (BGLOOP.EQ.3) GOTO 365
```

Figure E-I-17. Listing of Output Subroutine to Search Program with One Internal Subroutine (page 5 of 6 pages)

```
           IF (BGLOOP.EQ.4) GOTO 375
           READ(27,350)CONSUM
   350     FORMAT(F5.2)
           GOTO 390
   355     READ(27,360)CONSUM
   360     FORMAT(9X,F5.2)
           GOTO 390
   365     READ(27,370)CONSUM
   370     FORMAT(18X,F5.2)
           GOTO 390
   375   READ(27,380)CONSUM
   380   FORMAT(27X,F5.2)
   390     IF (INDIVD.EQ.8) GOTO 400
           WRITE(20,394)CONSUM
   394     FORMAT(9X,F5.2,5X,'0.0',7X,'1.0')
           GOTO 410
   400     WRITE(20,405)CONSUM
   405     FORMAT(9X,F5.2,5X,'0.3',7X,'0.7')
   410   CONTINUE
         WRITE(20,10)
C
C        ***********************************************************
C        * THE FOLLOWING IS RED DIVISION STRENGTH
C
C        ***********************************************************
C
         WRITE(20,420)
   420   FORMAT(9X,'11920')
         WRITE(20,430)
   430   FORMAT(9X,'2'/9X,'1',9X,'1.0',7X,'4382.5',4X,'1')
         WRITE(20,435)
   435   FORMAT(9X,'3200',6X,'1.0',7X,'10449 ',5X,'2')
C
C
         WRITE(20,440)RAMMO(BGLOOP)
   440   FORMAT(9X,'2'/9X,F7.2,3X,'0.0',7X,'1.0')
         WRITE(20,450)FPOL(BGLOOP)
   450   FORMAT(9X,F6.2,4X,'0.0',7X,'1.0')
         REWIND 27
   900   CONTINUE
         IBFLAG=0
         RETURN
         END
C
C
C
C
         SUBROUTINE PRPROG(BMAINF,NUMENG,NUMPER,STP,JJ)
C        ********************************************
C
         INTEGER NUMENG,NUMPER,IUYEAR,STP
         REAL BMAINF
         COMMON JTPSNA(15)
     5   READ(29,10,END=30)KKTPSN,IUYEAR,BMAINF,NUMENG,NUMPER
    10   FORMAT(1X,I5,2X,I2,2X,F5.4,2X,I3,2X,I5)
         IF((JTPSNA(JJ).EQ.KKTPSN).AND.(INT(STP/10).EQ.IUYEAR))THEN
           GOTO 30
         ENDIF
         GOTO 5
    30   RETURN
         END
```

Figure E-I-17. Listing of Output Subroutine to Search Program with One Internal Subroutine (page 6 of 6 pages)

```
      ERRDAT PROC
            COMMON /ERRDAT/ NERRS, NERRSL
       END
```

Figure E-I-18. Listing of FORTRAN Proc Identifying Certain COMMON Blocks in the BRIDGE Subroutine to the Search Program

```
      SUBROUTINE BRGPRE
C     ******************************************************************
C     *                                                                *
C     * BRGPRE: AFP BRIDGE PRE-PROCESSOR                               *
C     *                                                                *
C     ******************************************************************
C
      INCLUDE BPDAT,LIST
      INCLUDE ERRDAT,LIST
      INCLUDE BDGROU,LIST
C
      NERRS = 0
C     GET INPUT DATA
      CALL BPIN
C     ANY ERRORS PREVENT FURTHER PROCESSING
      IF (NERRS .GT. 0) STOP
C     PROCESS EACH TYPE OF GAP IN TURN
      DO 100 IG = 1,NGAPS
C       SEE WHAT VEHICLES CAN CROSS, WHAT BRIDGES CAN BE USED
        CALL BPGAP(IG)
        IF (POSSEL(IG) .GT. 0.0) THEN
C         NUMBER OF CROSSING POINTS TO BE USED:
          NCP = MIN(NCPD, NCPC(IG))
C         COMPUTE CROSSING TIME
          CALL BPCROS (IG, NCP, CT1)
          IF (NCP .LT. NCPC(IG)) THEN
C           TRY AGAIN WITH ONE ADDITIONAL CROSSING POINT
            CALL BPCROS (IG, NCP+1, CT2)
            CT1 = MIN(CT1, CT2)
          ENDIF
          GAPCTM(IG) = CT1
        ELSE
          GAPCTM(IG) = 0.0
        ENDIF
100   CONTINUE
C     COMPUTE BRIDGING CAPABILITY
      CALL BPCAP
      RETURN
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 1 of 15 pages)

```
      SUBROUTINE BPNEXT (NCP, TNEW, ICP)
C     ************************************************************
C     *                                                          *
C     * BPNEXT: TIME OF NEXT EVENT                               *
C     *                                                          *
C     ************************************************************
C
      INCLUDE BPDAT,LIST
C
      ICP = 1
      TNEW = TNEXT(1)
      DO 100 JCP = 1,NCP
        IF (TNEXT(JCP) .LT. TNEW) THEN
          ICP = JCP
          TNEW = TNEXT(JCP)
        ENDIF
100   CONTINUE
      IF (TNEW .GT. TNOW) THEN
        WRITE (6,60) TNOW, (VNUM(IVT(JCP)), JCP=1,NCP)
60      FORMAT (1X, F8.2, (T10, 30I4))
      ENDIF
CCCCCCC DEBUG START
C       IF (TNEW .GT. TNOW) THEN
C         WRITE (6,61) TNOW
C61       FORMAT ('CTIME = ', F8.2)
C         WRITE (6,62) 'CROSS PT', (JCP, JCP=1,NCP)
C62       FORMAT (1X, A, T15, 20I5)
C         WRITE (6,62) 'STATE', (ISTATE(JCP), JCP=1,NCP)
C         WRITE (6,62) 'BRIDGE', (IBT(JCP), JCP=1,NCP)
C         WRITE (6,62) 'VEHICLE', (IVT(JCP), JCP=1,NCP)
C         WRITE (6,63) 'NEXT TIME', (TNEXT(JCP), JCP=1,NCP)
C63       FORMAT (1X, A, T15, 20F5.1)
C         WRITE (6,64) 'VEHICLES', (LEFT(IV), IV=1,NVEH)
C64       FORMAT (1X, A, T10, 10F10.1)
C         WRITE (6,64) 'EFF RATE', (ECR(IV), IV=1,NVEH)
C       ENDIF
CCCCCCC DEBUG END
      RETURN
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines
(page 2 of 15 pages)

```
      SUBROUTINE GNERR (ITEM, MSG)
C     ******************************************************************
C     *                                                                *
C     * GNERR: ISSUE AN ERROR MESSAGE                                  *
C     *                                                                *
C     ******************************************************************
C
      INCLUDE ERRDAT.LIST
      CHARACTER*(*) ITEM, MSG
C
C     INCREMENT ERROR COUNT
      NERRS = NERRS+1
      NERRSL = NERRSL+1
C     PRINT THE MESSAGE
      IF (ITEM .EQ. ' ') THEN
        WRITE (6,61) MSG
61      FORMAT (' ERROR: ', A)
      ELSE
        WRITE (6,62) ITEM, MSG
62      FORMAT (' ERROR: ', A, ': ', A)
      ENDIF
      IF (NERRS .GT. 30) THEN
        WRITE (6,63)
63      FORMAT (' ERROR LIMIT EXCEEDED...PROCESSING STOPPED.')
        STOP
      ENDIF
      RETURN
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 3 of 15 pages)

```
      FUNCTION RANGE (ITEM, XLO, X, XHI)
C     **********************************************************************
C     *                                                                    *
C     * RANGE: CHECK A REAL NUMBER FOR BEING IN RANGE                       *
C     *                                                                    *
C     **********************************************************************
      CHARACTER*(*) ITEM
      CHARACTER*70 MSG
C
      IF ((XLO .LE. X) .AND. (X .LE. XHI)) THEN
        CONTINUE
      ELSE
C       BUILD ERROR MESSAGE
        WRITE (MSG,66) X, XLO, XHI
66      FORMAT (1PG11.4,
     &     ' SHOULD BE IN THE RANGE ', 1PG11.4, ' TO ', 1PG11.4)
        CALL GNERR(ITEM,MSG)
      ENDIF
      RANGE = X
      IF (RANGE .LT. XLO) RANGE = XLO
      IF (RANGE .GT. XHI) RANGE = XHI
      RETURN
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 4 of 15 pages)

```
      SUBROUTINE BPCROS (IG, NCP, CTIME)
C     ****************************************************************
C     *                                                              *
C     * BPCROS: COMPUTE TIME REQUIRED TO CROSS A GAP                 *
C     *                                                              *
C     ****************************************************************
C
      INCLUDE BPDAT,LIST
C
      WRITE (6,61) WGAP(IG)
61    FORMAT ('0CROSSING TIME CALCULATIONS FOR ', F6.0, ' METER GAP')
      WRITE (6,62) (I, I=1,NCP)
62    FORMAT ('0TIME', (T10, 30I4))
      WRITE (6,63) (' ---', I=1,NCP)
63    FORMAT (' --------', (T10, 30A4))
C     INITIALIZE VARIABLES
      TNOW = 0.0
      DO 100 IV = 1,NVEH
        IF (VCLAS(IV) .LE. BLMAX) THEN
          LEFT(IV) = NV(IV)
        ELSE
          LEFT(IV) = 0.0
        ENDIF
        ECR(IV) = 0.0
100   CONTINUE
C     ASSIGN BRIDGES TO CROSSING POINTS
      CALL BPASG (IG, NCP)
      DO 200 ICP = 1,NCP
        WRITE (6,64) TNOW, 'SETTING UP', BDESC(IBT(ICP)), ICP
64      FORMAT (1X, F8.2, T10, A20, 1X, A10, ' AT POINT ', I3)
        ISTATE(ICP) = KSETUP
        IVT(ICP) = 0
        TNEXT(ICP) = TNOW + SETUP(IBT(ICP),IG) / 60.0  @ SETUP IN HOURS
200   CONTINUE
C     GET THE TIME OF THE NEXT EVENT
      CALL BPNEXT (NCP, TNEW, ICP)
300   IF (TNEW .GE. BIG) GOTO 399
        DELTA = TNEW-TNOW
        TNOW = TNEW
C       COMPUTE NUMBERS OF VEHICLES LEFT
        DO 400 IV = 1,NVEH
          LEFT(IV) = LEFT(IV) - ECR(IV) * DELTA
          IF (LEFT(IV) .LT. 0.001) LEFT(IV) = 0.0
400     CONTINUE
C       MODIFY THE STATE OF THE CROSSING POINT
        IF (ISTATE(ICP) .EQ. KSETUP) THEN
          WRITE (6,64) TNOW, 'STARTING CROSSING', BDESC(IBT(ICP)), ICP
          ISTATE(ICP) = KCROSS
        ELSE IF (ISTATE(ICP) .EQ. KCROSS) THEN
C         REALLOCATION WILL TAKE CARE OF IT
        ELSE IF (ISTATE(ICP) .EQ. KTDOWN) THEN
C         REPLACE THE BRIDGE, IF FEASIBLE
          CALL BPSCOR (IG)  @ RE-COMPUTE CROSSING SCORES
C         FIND THE BRIDGE WITH THE BIGGEST SCORE
          IB = 0
          SBIG = 0.0
          DO 500 JB = 1,NBRIDG
            IF ((CSCORE(JB) .GT. SBIG) .AND. (NCB(JB) .GT. 0)) THEN
              IB = JB
              SBIG = CSCORE(JB)
            ENDIF
500       CONTINUE
          IF (IB .GT. 0) THEN
C           ASSIGN BRIDGE TO THIS CROSSING POINT
            IBT(ICP) = IB
            NCB(IB) = NCB(IB)-1
            WRITE (6,64) TNOW, 'SETTING UP', BDESC(IBT(ICP)),
     &            ICP
            ISTATE(ICP) = KSETUP
            TNEXT(ICP) = TNOW + SETUP(IB,IG) / 60.0
          ELSE
            ISTATE(ICP) = KFINI
            TNEXT(ICP) = BIG
          ENDIF
        ELSE
          STOP 13  @ ERRONEOUS STATE
        ENDIF
C       REALLOCATE VEHICLES TO CROSSING POINTS
        CALL BPREAL(IG,NCP)
C       GET THE NEXT EVENT
        CALL BPNEXT (NCP, TNEW, ICP)
      GOTO 300
399   CONTINUE
      CTIME = TNOW
      RETURN
      DEBUG SUBCHK
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 5 of 15 pages)

```
      INTEGER FUNCTION RANGEI (ITEM, IXLO, IX, IXHI)
C     ******************************************************************
C     *
C     * RANGEI: CHECK AN INTEGER FOR BEING IN RANGE                     *
C     *                                                                *
C     ******************************************************************
      CHARACTER*(*) ITEM
      CHARACTER*70 MSG
      IF ((IXLO .LE. IX) .AND. (IX .LE. IXHI)) THEN
        CONTINUE
      ELSE
C       BUILD ERROR MESSAGE
        WRITE (MSG,66) IX, IXLO, IXHI
66      FORMAT (I10, ' SHOULD BE IN THE RANGE ', I10, ' TO ', I10)
        CALL GNERR(ITEM,MSG)
      ENDIF
      RANGEI = IX
      IF (RANGEI .LT. IXLO) RANGEI = IXLO
      IF (RANGEI .GT. IXHI) RANGEI = IXHI
      RETURN
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 6 of 15 pages)

```
      SUBROUTINE QKSRTI (A, INDEX, N)
C
C     SORT A(N) IN ASCENDING ORDER VIA THE ARRAY 'INDEX'
C     I.E., AT THE CONCLUSION OF THE SUBROUTINE, INDEX IS TO
C     CONTAIN A PERMUTATION OF THE INTEGERS 1...N SUCH THAT
C     A(INDEX(I)) <= A(INDEX(J)) WHENEVER I<=J.  THE ARRAY A
C     IS UNCHANGED.
C     ADOPTED FROM 'QUICKSORT', PROGRAM 2.11 IN 'ALGORITHMS + DATA
C     STRUCTURES = PROGRAMS' BY N. WIRTH
C
      REAL A(*), X
      INTEGER INDEX(*), ITEMP
      INTEGER N
      PARAMETER MAXSTK = 36
C     THIS STACK SIZE IS ADEQUATE TO SORT 2**36 ITEMS
      INTEGER STACKL (MAXSTK), STACKR (MAXSTK), S, I, J, L, R
C     INITIALIZE INDEX TO 1...N
      DO 5 I = 1,N
        INDEX(I) = I
5     CONTINUE
C     STACK REQUEST TO SORT A[1..N]
      S = 1
      STACKL (1) = 1
      STACKR (1) = N
      R = N
10    CONTINUE
C       TAKE TOP REQUEST FROM STACK
        L = STACKL (S)
        R = STACKR (S)
        S = S - 1
20      CONTINUE
C       SPLIT A(L)...A(R)
          I = L
          J = R
          X = A (INDEX((L+R)/2))
30        CONTINUE
40          IF (A(INDEX(I)) .GE. X)  GOTO 50
              I = I + 1
              GOTO 40
50        CONTINUE
60          IF (X .GE. A(INDEX(J))) GOTO 70
              J = J - 1
              GOTO 60
70          CONTINUE
            IF (I .LE. J) THEN
              ITEMP = INDEX(I)
              INDEX(I) = INDEX(J)
              INDEX(J) = ITEMP
              I = I + 1
              J = J - 1
            ENDIF
          IF (I .LE. J ) GOTO 30
          IF (J-L .LT. R-I) THEN
            IF (I .LT. R) THEN
C             STACK REQUEST FOR SORTING RIGHT PARTITION
              S = S + 1
              STACKL (S) = I
              STACKR (S) = R
            ENDIF
C           CONTINUE SORTING LEFT PARTITION
            R = J
          ELSE
            IF (L .LT. J) THEN
C             STACK REQUEST FOR SORTING LEFT PARTITION
              S = S + 1
              STACKL (S) = L
              STACKR (S) = J
            ENDIF
C           CONTINUE SORTING RIGHT PARTITION
            L = I
          ENDIF
          IF (L .LT. R ) GOTO 20
        IF (S .NE. 0) GOTO 10
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 7 of 15 pages)

```
      SUBROUTINE BPGAP(IG)
C     ****************************************************************
C     *                                                              *
C     * BPGAP: COMPUTE CAPABILITY TO CROSS A GAP                     *
C     *                                                              *
C     ****************************************************************
C
      INCLUDE BPDAT,LIST
C
C     DETERMINE THE MAXIMUM LOAD CLASS SUPPORTED BY BRIDGES AND THE
C     NUMBER OF CROSSING POINTS THE BRIDGE INVENTORY CAN SUPPORT
      BLMAX = 0.0
      NCPC(IG) = 0
      DO 100 IB=1,NBRIDG
C        CROSSING POINTS FOR THIS BRIDGE TYPE:
         NC = FSETS(IB) * CPSET(IB,IG) + 0.0001
         NCPC(IG) = NCPC(IG) + NC
         IF (NC .GT. 0) THEN
            BLMAX = MAX(BLMAX,BCLAS(IB,IG))
         ENDIF
100   CONTINUE
      IF (BLMAX .EQ. 0.0) THEN
C        CAN'T CROSS IT
         POSSBL(IG) = 0.0
         FRVEHC(IG) = 0.0
      ELSE
C        CAN CROSS IT
         POSSBL(IG) = 1.0
C        COMPUTE THE FRACTION OF VEHICLES THAT CAN CROSS
         TOTCRS = 0.0
         DO 200 IV = 1,NVEH
            IF (VCLAS(IV) .LE. BLMAX) THEN
               TOTCRS = TOTCRS + NV(IV)
            ENDIF
200      CONTINUE
         FRVEHC(IG) = TOTCRS / TOTVEH
      ENDIF
CCCCCCC DEBUG START
C        WRITE (6,61) IG, POSSBL(IG), FRVEHC(IG)
C61      FORMAT (' GAP ', I2, ' POSSBL = ', F3.0, ' FRVEHC = ', F8.4)
CCCCCCC DEBUG END
      RETURN
      DEBUG SUBCHK
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 8 of 15 pages)

```
      SUBROUTINE BPIN
C     ****************************************************************
C     *                                                              *
C     * BPIN: DATA INPUT ROUTINE FOR BRIDGE PRE-PROCESSOR            *
C     *                                                              *
C     ****************************************************************
C
      INCLUDE BPDAT.LIST
      INCLUDE ERRDAT.LIST
       INCLUDE BDGROU.LIST
      DIMENSION CPSETT(12,4),VCRATT(12,4),BCLAST(12,4),SETUPT(12,4),
     *IBAFP(12)
      INTEGER RANGEI
C     FORMATS:
61    FORMAT(1X, 'NUMBER OF CROSSING POINTS DESIRED', 5X, I5)
62    FORMAT(1X, 'TAKE-DOWN FACTOR', 20X, F8.3)
63    FORMAT(1X, 'BASE CASE VALUES: F1 =', 1X, F7.4, 2X, 'F2 =', 1X,
     &  F10.6, 2X, 'F3 =', 1X, F7.4)
64    FORMAT(1X, 'NUMBER OF GAP TYPES =', 1X, I5)
65    FORMAT(1X, '# GAPS', 4X, 'WIDTH (METERS)')
66    FORMAT(1X, '------', 4X, '--------------')
67    FORMAT(1X, F6.0, 4X, F14.0)
68    FORMAT(1X, 'NUMBER OF VEHICLE TYPES =', 1X, I5)
69    FORMAT(1X, '# VEHICLES', 4X, 'LOAD CLASS', 4X, 'AFP #', 4X,
     &  'DESCRIPTION')
70    FORMAT(1X, '----------', 4X, '----------', 4X, '-----', 4X,
     &  '-----------')
71    FORMAT(1X, F10.0, 4X, F10.1, 4X, I5, 4X, A10)
72    FORMAT(1X, 'NUMBER OF BRIDGE/RAFT TYPES =', 1X, I5)
73    FORMAT(32X, 'GAP WIDTH   CROSS PTS   SETUP   CROSS RATE')
74    FORMAT(1X, '# SETS   AFP #   DESCRIPTION   (METERS)', 4X,
     &  'PER SET', 5X, '(MIN)   (PER HR)', 5X, 'LOAD CLASS')
75    FORMAT(1X, '------   -----   ----------- ', 2X,
     &  '---------   ---------   -----   ----------   ----------')
76    FORMAT(1X, F6.1, 3X, I5, 3X, A10)
77    FORMAT(32X, F9.0, 3X, F9.3, 3X, F5.0, 3X, F10.3, 3X, F10.1)
C
      WRITE(6,2000)BGLOOP
 2000 FORMAT( 1X,'BGLOOP IS EQUAL TO ',I1)
       IF (BGLOOP .EQ. 4)THEN
         NCPD=7
         GOTO 78
       ENDIF
         NCPD=2
 78    TDFACT=2
      WRITE (6,61) NCPD
      NCPD = RANGEI ('# CROSS PTS', 1, NCPD, MAXCP-1)
      WRITE (6,62) TDFACT
      TDFACT = RANGE ('TAKE-DOWN FACT', 0.0, TDFACT, BIG)
       BASEF1=1.0
       BASEF2=.005815
      IF(BGLOOP.EQ.4)BASEF2=.011014
       BASEF3=1.0
      WRITE (6,63) BASEF1, BASEF2, BASEF3
      BASEF1 = RANGE ('BASE F1', 1.0/BIG, BASEF1, BIG)
      BASEF2 = RANGE ('BASE F2', 1.0/BIG, BASEF2, BIG)
      BASEF3 = RANGE ('BASE F3', 1.0/BIG, BASEF3, BIG)
C
C        GAP DATA
C
      CALL PRTBOX (0, 'GAP DATA')
       NGAPS=4
      WRITE (6,64) NGAPS
      NGAPS = RANGEI ('# GAP TYPES', 1, NGAPS, MAXGAP)
      WRITE (6,65)
      WRITE (6,66)
       WGAP(1)=15
       WGAP(2)=60
       WGAP(3)=200
       WGAP(4)=400
       NGP(1)=30
       NGP(2)=7
       NGP(3)=3
       NGP(4)=1
      DO 100 IG = 1,NGAPS
        WRITE (6,67) NGP(IG), WGAP(IG)
        NGP(IG) = RANGE ('# GAPS', 0.0, NGP(IG), BIG)
        WGAP(IG) = RANGE ('WIDTH', 0.0, WGAP(IG), BIG)
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 9 of 15 pages)

```
  100   CONTINUE
C
C          VEHICLE DATA
C
        CALL PRTBOX (0, 'VEHICLE DATA')
        WRITE (6,68) NVEH
        NVEH = RANGEI ('# VEH TYPES', 1, NVEH, MAXVEH)
        WRITE (6,69)
        WRITE (6,70)
        TOTVEH = 0.0
        VNUM(0) = 0
        DO 200 IV = 1,NVEH
          WRITE (6,71) NV(IV), VCLAS(IV), VNUM(IV), VDESC(IV)
          NV(IV) = RANGE ('# VEHICLES', 0.0, NV(IV), BIG)
          VCLAS(IV) = RANGE ('CLASS', 0.0, VCLAS(IV), BIG)
          TOTVEH = TOTVEH + NV(IV)
  200   CONTINUE
        IF (TOTVEH .EQ. 0.0) CALL GNEPR (' ', 'NO VEHICLES IN FORCE')
C
C          BRIDGE DATA
C
        CALL PRTBOX (0, 'BRIDGE/RAFT DATA')
        WRITE (6,72) NBRIDG
        NBRIDG = RANGEI ('# BRIDGE TYPES', 1, NBRIDG, MAXBRG)
        WRITE (6,73)
        WRITE (6,74)
        WRITE (6,75)
C
C       ***************************************************************
C       ***************************************************************
C       THE FOLLOWING IS A TEMPOPARY STORAGE AREA FOR VARIABLES
C       SOME OF WHICH WILL LATER BE TRANSFERED INTO THE VARIABLES
C       CPSET( ),SETUP( ),VCRATE( ), AND BCLAS( ).
C       ***************************************************************
C
C
        DO 240 NN=1,12
          READ(30,256,END=260)IBAFP(NN),
      * CPSETT(NN,1),SETUPT(NN,1),VCRATT(NN,1),BCLAST(NN,1),
      * CPSETT(NN,2),SETUPT(NN,2),VCRATT(NN,2),BCLAST(NN,2),
      * CPSETT(NN,3),SETUPT(NN,3),VCRATT(NN,3),BCLAST(NN,3),
      * CPSETT(NN,4),SETUPT(NN,4),VCRATT(NN,4),BCLAST(NN,4)
  240     CONTINUE
   256 FORMAT(I3,1X,F3.1,F4.0,F4.0,F3.0,F3.1,1X,F4.0,F4.0,F3.0,1X,
      *F2.0,F4.0,F3.1,1X,F3.0,F2.0,F4.0,F2.0,F3.0)
  260   DO 300 IB = 1,NBRIDG
          WRITE (6,76) BSETS(IB), BNUM(IB), BDESC(IB)
          BSETS(IB) = RANGE ('# SETS', 0.0, BSETS(IB), BIG)
          DO 270 NN=1,12
            IF(BNUM(IB).EQ.IBAFP(NN))THEN
              DO 280 NM = 1,4
                CPSET(IB,NM)=CPSETT(NN,NM)
                SETUP(IB,NM)=SETUPT(NN,NM)
                VCRATE(IB,NM)=VCRATT(NN,NM)
                BCLAS(IB,NM)=BCLAST(NN,NM)
  280         CONTINUE
                GOTO 290
            ENDIF
  270     CONTINUE
  290     DO 400 IG = 1,NGAPS
            WRITE (6,77) WGAP(IG), CPSET(IB,IG), SETUP(IB,IG),
     &                VCRATE(IB,IG), BCLAS(IB,IG)
            CPSET(IB,IG) = RANGE ('# CROSS PTS', 0.0, CPSET(IB,IG), BIG)
            IF (CPSET(IB,IG) .GT. 0.0) THEN
              SETUP(IB,IG) = RANGE ('SETUP', 0.0, SETUP(IB,IG), BIG)
              VCRATE(IB,IG) = RANGE ('CROSS RATE',0.001,VCRATE(IB,IG),BIG)
              BCLAS(IB,IG) = RANGE ('CLASS', 0.0, BCLAS(IB,IG), BIG)
            ENDIF
  400     CONTINUE
  300   CONTINUE
        RETURN
        DEBUG SUBCHK
        END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 10 of 15 pages)

```
      SUBROUTINE BPSCOR (IG)
C     ****************************************************************
C     *                                                              *
C     * BPSCOR: COMPUTE CROSSING POINT SCORES OF BRIDGES             *
C     *                                                              *
C     ****************************************************************
C
      INCLUDE BPDAT,LIST
      DO 100 IB = 1,NBRIDG
        IF (NCB(IB) .LE. 0) THEN
          CSCORE(IB) = 0.0
        ELSE
C         COMPUTE NUMBER OF VEHICLES CAPABLE OF CROSSING THIS BRIDGE
          VC = 0.0
          DO 200 IV = 1,NVEH
            IF (VCLAS(IV) .LE. BCLAS(IB,IG)) THEN
              VC = VC + LEFT(IV)
            ENDIF
200       CONTINUE
          TEMP = 1.0 + SETUP(IB,IG) + SETUP(IB,IG)*TDFACT
          CSCORE(IB) = CPSET(IB,IG) * VCRATE (IB,IG) * VC / TEMP
        ENDIF
100   CONTINUE
CCCCCCC DEBUG START
C       WRITE (6,61) IG, (CSCORE(IB), IB=1,NBRIDG)
C61     FORMAT (' GAP ', I2, /, ' SCORES ', (T10, 10F8.2))
C       WRITE (6,62) (NCB(IB), IB=1,NBRIDG)
C62     FORMAT (' NCB', (T10, 10I8))
CCCCCCC DEBUG END
      RETURN
      DEBUG SUBCHK
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 11 of 15 pages)

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 11 of 15 pages)

```
       SUBROUTINE BPCAP
C      ****************************************************************
C      *                                                              *
C      * BPCAP: COMPUTE TOTAL BRIDGE-CROSSING CAPABILITY              *
C      *                                                              *
C      ****************************************************************
C
       INCLUDE BPDAT,LIST
       INCLUDE BDGROU,LIST
C      FORMATS:
61     FORMAT(48X, 'CROSSING', 4X, 'FRACTION')
62     FORMAT(1X, 'GAP    WIDTH (M)    # GAPS    WEIGHT', 3X,
      &  'POSSIBLE    TIME (HR)    VEHICLES')
63     FORMAT(1X, '---   ---------    ------    ------', 3X,
      &  '--------   ---------   --------')
64     FORMAT(1X, I3, 3X, F9.0, 3X, F6.1, 3X, F6.2, 3X, F3.1, 8X,
      &  F9.2, 3X, F8.2)
65     FORMAT(1X, 'F1 =', 1X, F9.5, 3X, 'F2 =', 1X, F10.6, 3X,
      &  'F3 =', 1X, F9.5)
66     FORMAT(1X, 'BRIDGING CAPABILITY =', 1X, F8.3)
C
C      COMPUTE GAP WEIGHTS
       SUM = 0.0
       DO 100 IG = 1,NGAPS
         GAPWT(IG) = NGP(IG) * WGAP(IG)
         SUM = SUM + GAPWT(IG)
100    CONTINUE
       DO 200 IG = 1,NGAPS
         GAPWT(IG) = GAPWT(IG) / SUM
200    CONTINUE
C      COMPUTE FIRST FACTOR
       F1 = 0.0
       DO 300 IG = 1,NGAPS
         F1 = F1 + GAPWT(IG) * POSSBL(IG)
300    CONTINUE
C      COMPUTE SECOND FACTOR
       SUM = 0.0
       DO 400 IG = 1,NGAPS
         SUM = SUM + NGP(IG) * GAPCTM(IG)
400    CONTINUE
       IF (SUM .GT. 0.0) THEN
         F2 = 1.0 / SUM
       ELSE
         F2 = 1.0
       ENDIF
C      COMPUTE THIRD FACTOR
       SUM = 0.0
       DO 500 IG = 1,NGAPS
         SUM = SUM + FRVEHC(IG)
500    CONTINUE
       F3 = SUM / FLOAT(NGAPS)
C      TOTAL CAPABILITY:
       BCAP = F1 * F2 * F3 / (BASEF1 * BASEF2 * BASEF3)
C
C      PRINT RESULTS
C
       CALL PRTBOX (0, 'RESULTS')
       WRITE (6,61)
       WRITE (6,62)
       WRITE (6,63)
       DO 600 IG = 1,NGAPS
         WRITE (6,64) IG, WGAP(IG), NGP(IG), GAPWT(IG), POSSBL(IG),
      &         GAPCTM(IG), FRVEHC(IG)
600    CONTINUE
       WRITE (6,*) ' '
       WRITE (6,65) F1, F2, F3
       WRITE (6,*) ' '
       WRITE (6,66) BCAP
       RETURN
       DEBUG SUBCHK
       END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 12 of 15 pages)

```
      SUBROUTINE BPASG (IG, NCP)
C     ******************************************************************
C     *                                                                *
C     * BPASG: INITIAL ASSIGNMENT OF BRIDGES TO CROSSING POINTS        *
C     *                                                                *
C     ******************************************************************
C
      INCLUDE BPDAT,LIST
      INTEGER INDEX(MAXBRG)
C
C     COMPUTE NUMBER OF CROSSING POINTS EACH TYPE BRIDGE CAN SUPPORT
      DO 100 IB = 1,NBRIDG
        NCB(IB) = BSETS(IB) * CPSET(IB,IG) + 0.0001
100   CONTINUE
C     COMPUTE CROSSING POINT SCORES OF BRIDGES
      CALL BPSCOR (IG)
C     SORT BRIDGES BY CROSSING POINT SCORE
      CALL QKSRTI (CSCORE, INDEX, NBRIDG)
      IBR = NBRIDG  @ START WITH HIGHEST-SCORING BRIDGE
      IB = INDEX(IBR)
      DO 200 ICP = 1,NCP  @ CROSSING POINT LOOP
        IF (NCB(IB) .LE. 0) THEN
C         USE THE NEXT TYPE OF BRIDGE, IN DESCENDING CPSCOR ORDER
          IBR = IBR-1       @ BRIDGE RANK
          IB = INDEX(IBR)   @ BRIDGE TYPE
        ENDIF
C       ASSIGN THE BRIDGE TO THE CROSSING POINT
        IBT(ICP) = IB
C       DECREMENT THE NUMBER OF CROSSING POINTS LEFT FOR THIS TYPE
        NCB(IB) = NCB(IB) - 1
200   CONTINUE
      RETURN
      DEBUG SUBCHK
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 13 of 15 pages)

```
      SUBROUTINE PRTBOX(ICNTRL, S)
C     ***************************************************************
C     *                                                             *
C     * PRTBOX: PRINT A CHARACTER STRING FRAMED IN A BOX            *
C     *                                                             *
C     ***************************************************************
      CHARACTER*(*) S
C
      L = LEN(S)
      WRITE (6,61) ICNTRL, ('*', I = 1,L+4)
61    FORMAT (I1, 132A1)
      WRITE (6,62) '*', (' ', I=1,L+2), '*'
62    FORMAT (' ', 132A1)
      WRITE (6,63) '*', ' ', S, ' ', '*'
63    FORMAT (' ', 5A)
      WRITE (6,62) '*', (' ', I=1,L+2), '*'
      WRITE (6,64) ('*', I=1,L+4)
64    FORMAT (' ', 132A1)
      WRITE (6,65)
65    FORMAT (' ')
      RETURN
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 14 of 15 pages)

```
      SUBROUTINE BPREAL (IG, NCP)
C     ****************************************************************
C     *                                                              *
C     * BPREAL: REALLOCATE VEHICLES TO CROSSING POINTS               *
C     *                                                              *
C     ****************************************************************
C
      INCLUDE BPDAT,LIST
C     FORMATS:
64      FORMAT (1X, F8.2, T10, A20, 1X, A10, ' AT POINT ', I3)
C
      DO 100 IV = 1,NVEH
        ECR(IV) = 0.0
100   CONTINUE
      DO 200 ICP = 1,NCP
        IF (ISTATE(ICP) .EQ. KCROSS) THEN
          IB = IBT(ICP)
          BCL = BCLAS(IB,IG)
C         FIND THE HEAVIEST VEHICLE LEFT THAT THE BRIDGE CAN SUPPORT
          IV = 0
          VCL = -1.0
          DO 300 JV = 1,NVEH
            IF ((LEFT(JV) .GT. 0.0) .AND. (VCLAS(JV) .LE. BCL) .AND.
     &            (VCLAS(JV) .GT. VCL)) THEN
              IV = JV
              VCL = VCLAS(JV)
            ENDIF
300       CONTINUE
          IF (IV .GT. 0) THEN
C           ASSIGN VEHICLE TO THIS CROSSING POINT
            IVT(ICP) = IV
            ECR(IV) = ECR(IV) + VCRATE(IB,IG)
          ELSE
C           TAKE IT DOWN
            WRITE (6,64) TNOW, 'TAKING DOWN', BDESC(IBT(ICP)),
     &            ICP
            ISTATE(ICP) = KTDOWN
            IVT(ICP) = 0
            TNEXT(ICP) = TNOW + SETUP(IB,IG) * TDFACT / 60.0
          ENDIF
        ENDIF
200   CONTINUE
C     COMPUTE TIME OF NEXT EVENT FOR CROSSING POINTS WITH VEHICLES
      DO 400 ICP = 1,NCP
        IF (ISTATE(ICP) .EQ. KCROSS) THEN
          IV = IVT(ICP)
          DELTA = LEFT(IV) / ECR(IV)
          TNEXT(ICP) = TNOW + DELTA
        ENDIF
400   CONTINUE
      RETURN
      DEBUG SUBCHK
      END
```

Figure E-I-19. Listing of BRIDGE Subroutine to Search Program with 13 External Subroutines (page 15 of 15 pages)

```
@MAP,E ,*P3COUNTER.B7DCOUNT4
IN *P3COUNTER.BRGPREI
IN *P3COUNTER.BPINI
IN *P3COUNTER.BPGAP
IN *P3COUNTER.BPSCOR
IN *P3COUNTER.GNERR
IN *P3COUNTER.RANGE
IN *P3COUNTER.RANGEI
IN *P3COUNTER.QKSRTI
IN *P3COUNTER.PRTPOX
IN *P3COUNTER.BPREAL
IN *P3COUNTER.BPNEXT
IN *P3COUNTER.BPCROS
IN *P3COUNTER.BPCAP
IN *P3COUNTER.BPASG
IN *P3COUNTER.COUNT4
IN *P3COUNTER.OUTPUTB
```

Figure E-I-20. Listing of the Mapping Routine for CS/CSS Search Program

```
C234567
C PROGRAM TO CALCULATE CS/CSS FACTORS AND OUTPUT U,V,S,T VALUES
C W. R. HILLISON 30 AUG 83
C MODIFIED G.F. COOPER -- 30 MAR 84, 3 MAY 84
C MODIFIED BY JOHN WARREN -- 21 MAY 84
C MODIFIED BY TOM LINGAN--15 AUG 84
C FTN COMPILER USED
      DIMENSION
     +     FACT(9,4)
      CHARACTER NFUN(9)*11,NAMPOS*6,FORCE*5
      LOGICAL JUMP,JMPT(2),JMPBM
      REAL EQMN(2), PRSMN(2), LANMN(2), AWLMN(2), ADFRT(2), CAPMN(2),
     &     EQCLM(2), CLRMN(2), TDENS(2,2), TSPEED(2), SPDWT(2), SPDX(2)
      DATA NFUN/
     1 'MOBILITY   ','BRIDGING   ',
     2 'MINE       ','PROT.POSN. ',
     3 'N/A        ','C2EW       ',
     4 'MEDICAL    ','MAINTENANCE',
     5 'SUPP&TRANS '/
C     SET ALL ARRAY FACT VALUES TO 1.0
      DO 2000 JJ=1,176
      DO 2 I=1,9
        DO 1 J=1,4
          FACT(I,J)=1.000
    1   CONTINUE
    2 CONTINUE
      JMPBM=.FALSE.
      JMPT(1)=.FALSE.
      JMPT(2)=.FALSE.
C       INPUT CORRESPONDING ENVIRONMENT AND POSTURE NAME
      READ(20,32) IENV,NAMPOS,FORCE
      WRITE(15,50)FORCE,IENV
   50 FORMAT('@ED,I *H7CSCSS.',A5,'E',I2)
   32 FORMAT(8X,I3,8X,A6,7X,A5)
C
C     FACTOR 1: MOBILITY
C
      DO 100 ISIDE = 1,2
        L = 2*ISIDE-1
        READ (20,*) JUMP, (FACT(1,J), J=L,L+1)
        IF (.NOT. JUMP) THEN
          JMPT(ISIDE) = .TRUE.
          READ (20,*) SPDX(1), SPDX(2), SPDWT(2)
          SPDWT(1) = 1.0
          SUMD = 0.0
          DO 110 K = 1,2  @ 1=GROUND, 2=AIR
            TDENS(ISIDE,K) = 0.0
            TSPEED(K) = 0.0
            READ (20,*) NVEH
            DO 120 IVEH = 1,NVEH
              READ (20,*) DENSTY, SPEED
              TDENS(ISIDE,K) = TDENS(ISIDE,K) + DENSTY
              TSPEED(K) = TSPEED(K) + SPEED * DENSTY
  120       CONTINUE
            SUMD = SUMD + TDENS(ISIDE,K)
  110     CONTINUE
C         COMPUTE MOBILITY MEASURE:
          FACT(1,L+1) = 0.0
          DO 130 K = 1,2
            FACT(1,L+1) = FACT(1,L+1) +
     &          TSPEED(K) * SPDWT(K) / (SPDX(K) * SUMD)
  130     CONTINUE
          READ (20,*) RECEHO
          READ (20,*) CAPEHO
C         MOBILITY COUNTER-MEASURE:
          FACT(1,L) = CAPEHO / RECEHO
        ENDIF
  100 CONTINUE
C               BRIDGING FUNCTION 2
  200 READ(20,*) JUMP,(FACT(2,J),J=1,2)
      IF(JUMP) GOTO 250
C               READ NUMBER OF BLUE AND RED BRIDGE TYPES
      READ (20,*) BNBRG
C               READ BLUE DENSITY/LENGTH OF SPAN(METERS)
      DO 8 I=1,BNBRG
        READ (20,*) BBDEN,BSPAN
        TBSPAN=BBDEN*BSPAN
        SBSPAN=SBSPAN+TBSPAN
```

Figure E-I-21. Listing of AFP CS/CSS Main Preprocessor
(Page 1 of 5 pages)

```
    8 CONTINUE
C               COMPUTE V VALUE FOR BLUE FUNCTION 2--BRIDGING
      FACT(2,2)=SBSPAN/360.
C               READ RED DENSITY/LENGTH OF SPAN(METERS)
  250 READ(20,*) JUMP,(FACT(2,J),J=3,4)
      IF(JUMP) GOTO 300
      READ(20,*) RNBRG
      DO 9 I=1,RNBRG
         READ (20,*) RBDEN,RSPAN
         TRSPAN=RBDEN*RSPAN
         SRSPAN=SRSPAN+TRSPAN
    9 CONTINUE
C               COMPUTE T VALUE FOR RED FUNCTION 2--BRIDGING
      FACT(2,4)=SRSPAN/360.
C
C     FACTOR 3: MINES
C
300   CONTINUE
      DO 350 ISIDE = 1,2
         JSIDE = 3-ISIDE
         L = 2*ISIDE - 1
         READ (20,*) JUMP, (FACT(3,J), J=L,L+1)
         IF (.NOT. JUMP) THEN
           READ(20,*) EQMN(ISIDE), PPSMN(ISIDE), LANMN(JSIDE),
     &           AWLMN(JSIDE), ADFRT(JSIDE)
           IF (JSIDE .EQ. 1) THEN
             READ (20,*) EQCLM(1)
             CLRMN(1) = EQCLM(1) + PRSMN(1) * 3.6 / 30.0
           ELSE
             READ (20,*) CLRMN(2)
           ENDIF
           CAPMN(ISIDE) = (EQMN(ISIDE) + PRSMN(ISIDE) * 500.0 / 13.0) *
     &           LANMN(JSIDE) * AWLMN(JSIDE) / (0.5 * ADFRT(JSIDE))
           FACT(3,2*ISIDE) = CAPMN(ISIDE) / CLRMN(JSIDE)
         ENDIF
350   CONTINUE
C               FUNCTION 4 PROTECTIVE POSITIONS
C               READ NUMBERS OF BLUE AND RED PROTECTIVE POSITION EQUIP
C               TYPES
  400 READ(20,*) JUMP,(FACT(4,J),J=1,2)
      IF(JUMP) GOTO 450
      IF(JMPT(1)) GOTO 401
      WRITE (15,98)
   98 FORMAT(' *** NEED TDENS(1,1) -- SKIP')
      GOTO 450
  401 READ (20,*) BNPP
C               READ DENSITY AND RATE(POSITIONS/HR) BLUE
      SBDENP=0
      DO 10 I=1,BNPP
         READ (20,*) BDENPP,BRATEP
         TBDENP=BDENPP*BRATEP
         SBDENP=SBDENP+TBDENP
   10 CONTINUE
C               COMPUTE BLUE V VALUE FOR FUNCTION 4 PROTECTIVE POSITIONS"
      IF(.NOT.JMPT(1)) GOTO 450
      FACT(4,2)=(SBDENP*10.)/TDENS(1,1)
C               READ DENSITY AND RATE(POSITIONS/HR) RED
  450 READ(20,*) JUMP,(FACT(4,J),J=3,4)
      IF(JUMP) GOTO 600
      IF(JMPT(2)) GOTO 451
      WRITE (15,97)
   97 FORMAT(' *** NEED TDENS(2,1) -- SKIP')
      GOTO 600
  451 READ(20,*) RNPP
      SRDENP=0.0
      DO 11 I=1,RNPP
         READ (20,*) RDENPP,RRATEP
         TRDENP=RDENPP*RRATEP
         SRDENP=SRDENP+TRDENP
   11 CONTINUE
C               COMPUTE RED T VALUE FOR FUNCTION 4 PROTECTIVE POSITIONS
      IF(.NOT.JMPT(2)) GOTO 600
      FACT(4,4)=(SRDENP*10.)/TDENS(2,1)
C
C     FUNCTION 6: CREW
C
600   CONTINUE
      DO 610 ISIDE = 1,2
         L = 2*ISIDE-1
         READ (20,*) JUMP, (FACT(6,J), J=L,L+1)
         IF (.NOT. JUMP) THEN
           SUM = 0.0
```

Figure E-I-21. Listing of AFP CS/CSS Main Preprocessor
(Page 2 of 5 pages)

```
              DO 620 IBAND = 1,4
C                READ # RADIOS, TOP EFFECTS,
C                   JAMMING FACTOR, JAMMER WEIGHT, # JAMMER TYPES:
                 READ (20,*) DNSRAD, TOPJAM, FACJAM, WTJAM, NJAM
                 TOTJAM = 0.0
                 DO 630 I = 1,NJAM
C                   READ EFFECTIVENESS, # OF JAMMERS OF THIS TYPE:
                    READ (20,*) EFFJAM, DNSJAM
                    TOTJAM = TOTJAM + EFFJAM * DNSJAM
 630             CONTINUE
                 IF (DNSRAD .EQ. 0.0) THEN
                    T = TOPJAM
                 ELSE
                    T = MIN (TOPJAM, TOTJAM / DNSRAD)
                 ENDIF
                 SUM = SUM + T * FACJAM * WTJAM
 620          CONTINUE
              FACT(6,L+1) = 1.0 - SUM
           ENDIF
 610    CONTINUE
C                   FUNCTION 7 MEDICAL
    700 READ(20,*) JUMP,(FACT(7,J),J=1,2)
        IF(JUMP) GOTO 750
        JMPBM=.TRUE.
C                   READ BLUE DIV/AREA SUPPORTED STRENGTH, SURGE FACTOR,
C                   WIA/DNBI,
C                   (PER 1000),MED BN CAPABILITY(ADMINS/DAY),FRACTION
C                   EVACUATED
        READ (20,*) BDIVS,BAREAS,BSFACT,BDWIA,BAWIA,BDDNBI,BADNBI,BMBCAP,
       +BFEVAC
        TBDIVS=((((BDIVS*BDWIA)+(BDIVS*BDDNBI))+((BAREAS*BAWIA)+(BAREAS*BA
       +DNBI)))*BSFACT)/1000.
C                   COMPUTE BLUE V VALUE FOR FUNCTION 4--MEDICAL
        TSS=BDIVS+BAREAS
        FACT(7,2)=1-(((TBDIVS-BMBCAP)+(BMBCAP*BFEVAC))/TSS)
C                   1BDDNBI,BADNBI,BMBCAP,BFEVAC,TBDIVS,TSS
C                   200 FORMAT(11F9.1)
C                   *************************************************************
C                   **** NEED TO WORK ON FOR RED  MEDICAL MILLIRON 4
C                   SEPT*********
    750 READ(20,*) JUMP,(FACT(7,J),J=3,4)
        IF(JUMP) GOTO 800
C                   READ RED DIV/AREA SUPPORTED STRENGTH, SURGE FACTOR,
C                   WIA/DNBI,
C                   (PER 1000),MED BN CAPABILITY(ADMINS/DAY),FRACTION
C                   EVACUATED
        READ (20,*) RDIVS,RAREAS,RSFACT,RDWIA,RAWIA,RDDNBI,RADNBI,RMBCAP,
       +RFEVAC
        TRDIVS=((((RDIVS*RDWIA)+(RDIVS*RDDNBI))+((RAREAS*RAWIA)+(RAREAS*RA
       +DNBI)))*RSFACT)/1000.
C                   COMPUTE RED T VALUE FOR FUNCTION 4--MEDICAL
        TSS=RDIVS+RAREAS
        FACT(7,4)=1-(((TRDIVS-RMBCAP)+(RMBCAP*RFEVAC))/TSS)
C                   **** SET TO OLD VALUE
C                   *********
C                   ********************************
C                   ********************************
C                   FUNCTION 8 MAINTENANCE
C                   READ NUMBER OF MATERIEL CATEGORIES BLUE,RED
    800 READ(20,*) JUMP,(FACT(8,J),J=1,2)
        IF(JUMP) GOTO 850
        READ (20,*) BNMCAT
C                   READ BLUE MAJOR MATERIAL(CAPABILITY,REQUIREMENT)
        DO 15 I=1,BNMCAT
          READ (20,*) BMCAP,BMREQ
          TBMREQ=TBMREQ+BMREQ
          TBMCAP=TBMCAP+BMCAP
     15 CONTINUE
        TBMCAP=TBMCAP/365.
        FACT(8,2)=TBMCAP/TBMREQ
C                   SET BLUE V VALUE TO 1.0 IF GREATER THAN 1.0
        IF  (FACT(8,2).GT.1.0) FACT(8,2)=1.0
C                   COMPUTE RED T VALUE--MAINTENANCE
    850 READ(20,*) JUMP,(FACT(8,J),J=3,4)
        IF(JUMP) GOTO 900
        READ(20,*) RNMCAT
        DO 16 I=1,RNMCAT
C                   READ RED MAJOR MATERIAL(CAPABILITY,REQUIREMENT)
          READ (20,*) RMCAP,RMREQ
          TRMREQ=TRMREQ+RMREQ
          TRMCAP=TRMCAP+RMCAP
     16 CONTINUE
```

Figure E-I-21. Listing of AFP CS/CSS Main Preprocessor
(Page 3 of 5 pages)

```
      FACT(8,4)=TRMCAP/TRMREQ
C               IF RED T VALUE GREATER THAN 1.0-- SET TO 1.0
      IF (FACT(8,4).GT.1.0) FACT(8,4)=1.0
C               ****************************************************
C               ***** SET RED VALUE TO OLD VALUE.  I DON'T LIKE THE*****
C               ***** THIS WAS DONE--- NEEDS MORE WORK             *****
C               ***** MILLIRON 4 SEPT 83                           *****
C               ****************************************************
      FACT(8,4)=.74
C               FACTOR 9 SUPPLY AND TRANSPORTATION
C               S/T CAPABILITY
C               BROKEN DOWN INTO THREE COMPONENTS
C               STONS,GALLONS(BULK POL),ALOC
C               READ NUMBER OF BLUE  S/T ASSETS
  900 READ(20,*) JUMP,(FACT(9,J),J=1,2)
      IF(JUMP) GOTO 950
      IF(JMPBM) GOTO 901
      WRITE (15,96)
   96 FORMAT(' *** NEED BDIVS -- SKIP')
      GOTO 950
  901 READ (20,*) BNSTA
C               COMPUTE BLUE CAPABILITY
C               READ BLUE DENSITY S/T ASSETS,%EFFECTIVNESS(STON/GAL)
C               TCOMP(POINTER TO COMPONENT DEFINED AT LINE 144
C               1 = STON, 2 = GALLONS, 3 = ALOC (STONS)
      DO 20 I=1,BNSTA
        READ (20,*) BDST,BAVAIL,BEFF,TCOMP
        IF (IFIX(TCOMP).GT.1) GO TO 18
C               COMPUTE STON CAPABILITY
        GBCAPG=BDST*BAVAIL*BEFF
        TBCAPG=TBCAPG+GBCAPG
        GO TO 20
   18   IF (IFIX(TCOMP).EQ.3) GO TO 19
C               COMPUTE GALLONS(BULK POL) CAPABILITY COMPONENT
        GBCAPP=BDST*BAVAIL*BEFF
        BCAPBP=BCAPBP+GBCAPP
        GO TO 20
   19   CONTINUE
C               COMPUTE BLUE ALOC CAPABILITY COMPONENT
        GBCAPA=BDST*BAVAIL*BEFF
        TBCAPA=TBCAPA+GBCAPA
   20 CONTINUE
C               COMPUTE BLUE S/T REQUIREMENT
C               BLUE DIVISION STRENGTH READ IN AT LINE 100(BDIVS)
C               CONVERSION LBS TO GAL   .152
C               READ NUMBER OF BLUE CLASSES OF SUPPLY TO BE CONSIDERED
      READ (20,*) BNCLAS
C               COMPUTE BLUE REQUIREMENT
      DO 23 I=1,BNCLAS
        READ (20,*) BCONR,BALOC,BBASIC
        IF (I.EQ.2) GO TO 22
        GBREQG=((BCONR*BDIVS*BBASIC)/2000.)*(1-BALOC)
        GBREQA=((BCONR*BDIVS*BBASIC)/2000.)*BALOC
        TBREQG=TBREQG+GBREQG
        TBREQA=TBREQA+GBREQA
        GO TO 23
   22   BREQBP=(BCONR*BDIVS*BBASIC)*.152
   23 CONTINUE
C               COMPUTE COMPONENT FACTORS
C               COMPUTE STON GROUND FACTOR
      BFSTG=TBCAPG/TBREQG
C               COMPUTE CLASS III BULK POL FACTOR(GALLONS)
      BFBPOL=BCAPBP/BREQBP
C               COMPUTE BLUE ALOC FACTOR
      BFALOC=TBCAPA/TBREQA
CC      WRITE (15,35) TBCAPG,TBREQG,BCAPBP,BREQBP,TBCAPA,TBREQA
CC   35 FORMAT(6F12.4)
      IF (BFSTG.GT.1.0) BFSTG=1.0
      IF (BFBPOL.GT.1.0) BFBPOL=1.0
      IF (BFALOC.GT.1.0) BFALOC=1.0
C               COMPUTE BLUE V VALUE FOR FUNCTION 9-- S/T
      IF(.NOT.JMPBM) GOTO 950
      FACT(9,2)=(BFSTG+BFBPOL+BFALOC)/3.
      IF (FACT(9,2).GT.1.0) FACT(9,2)=1.0
C               COMPUTE RED CAPABILITY
C               READ RED DIVISION STRENGTH
C               READ RED NUMBER OF S + T ASSETS
  950 READ(20,*) JUMP,(FACT(9,J),J=3,4)
      IF(JUMP) GOTO 1000
      READ (20,*) RDIVS
      READ (20,*) RNSTA
C               READ RED DENSITY S/T ASSETS,%EFFECTIVNESS(MTON/LITERS)
```

Figure E-I-21. Listing of AFP CS/CSS Main Preprocessor (Page 4 of 5 pages)

```
C               TCOMP(POINTER TO COMPONENT DEFINED AT LINE 144
C               1 = STON, 2 = GALLONS, 3 = ALOC (MTONS)
      DO 27 I=1,RNSTA
         READ (20,*) RDST,RAVAIL,REFF,TCOMP
         IF (IFIX(TCOMP).GT.1) GO TO 25
C               COMPUTE STON CAPABILITY REQUIREMENT
C               CONVERSION METRIC TONS  TO STON  1.102
         GRCAPG=(RDST*RAVAIL*REFF)*1.102
         TRCAPG=TRCAPG+GRCAPG
         GO TO 27
   25    IF (IFIX(TCOMP).EQ.3) GO TO 26
C               COMPUTE GALLONS(BULK POL) CAPABILITY COMPONENT
C               CONVERSION LITERS TO GALLONS
         GRCAPP=(RDST*RAVAIL*REFF)*.264
         RCAPBP=RCAPBP+GRCAPP
         GO TO 27
   26    CONTINUE
C               COMPUTE RED ALOC CAPABILITY COMPONENT
         GRCAPA=(RDST*RAVAIL*REFF)*1.102
         TRCAPA=TRCAPA+GRCAPA
   27 CONTINUE
C               COMPUTE RED REQUIREMENT
C               READ NUMBER OF RED CLASSES OF SUPPLY TO BE CONSIDERED
      READ (20,*) RNCLAS
      DO 29 I=1,RNCLAS
         READ (20,*) RCONR,RALOC,RBASIC
         IF (I.EQ.2) GO TO 28
         GRREQG=((RCONR*RDIVS*RBASIC)/2000.)*(1-RALOC)
         GRREQA=((RCONR*RDIVS*RBASIC)/2000.)*RALOC
         TRREQG=TRREQG+GRREQG
         TRREQA=TRREQA+GRREQA
         GO TO 29
   28    RREQBP=(RCONR*RDIVS*RBASIC)*.152
   29 CONTINUE
C               COMPUTE COMPONENT FACTORS
C               COMPUTE STON GROUND FACTOR
      RFSTG=TRCAPG/TRREQG
C               COMPUTE CLASS III BULK POL FACTOR(GALLONS)
      RFBPOL=RCAPBP/RREQBP
C               COMPUTE RED ALOC FACTOR
C               RFALOC = TRCAPA/TRREQA
C               RFALOC SET TO 1.0--NO RED ALOC SO DIVIDE FAULT IF LEFT AT
C               0.
      RFALOC=1.0
      IF (RFSTG.GT.1.0) RFSTG=1.0
      IF (RFBPOL.GT.1.0) RFBPOL=1.0
      IF (RFALOC.GT.1.0) RFALOC=1.0
      FACT(9,4)=(RFSTG+RFBPOL+RFALOC)/3.
 1000 CONTINUE
      CALL INDXCS (FACT)
C     WRITE (6,31)
C  31 FORMAT(
C    1 /' F EN  B C/M   B MEAS  R C/M   R MEAS',6X,
C    2  'FUNCTION      PCST   FORCE'/
C    3  ' - --  ------  ------  ------  ------',5X,'-',
C    4  '----------  ------  -----'/)
C     WRITE (6,30) (I,IENV,(FACT(I,J),J=1,4),NFUN(I),NAMPOS,FORCE,
C    1 I=1,9)
   30 FORMAT (I2,I3,4F8.2,5X,A11,2X,A6,2X,A5)
      WRITE (15,30) (I,IENV,(FACT(I,J),J=1,4),NFUN(I),NAMPOS,FORCE,
     1 I=1,9)
      WRITE(15,51)
   51 FORMAT('@EOF')
 2000 CONTINUE
      STOP
      END
```

Figure E-I-21. Listing of AFP CS/CSS Main Preprocessor
(Page 5 of 5 pages)

```
      SUBROUTINE INDXCS (FACT)
C     ****************************************************************
C     *                                                              *
C     * INDXCS: INDEX CS AND CSS FACTORS                             *
C     *                                                              *
C     ****************************************************************
C
      REAL FACT(9,4)
      REAL INDEX(9,4), REF(9,4)
C
C     READ REFERENCE FACTORS
      READ (18,51,END=999) ((REF(I,J), J=1,4), I=1,9)
51    FORMAT (5X, 4F8.0)
      WRITE (6,61)
61    FORMAT ('0CS/CSS REFERENCE FACTORS:')
      WRITE (6,62) ((REF(I,J), J=1,4), I=1,9)
62    FORMAT (5X, 4F8.2)
C     1-FILL INDEX MATRIX
      DO 100 J = 1,4
        DO 200 I = 1,9
          INDEX(I,J) = 1.0
200     CONTINUE
100   CONTINUE
C     SET UP INDEXES
      INDEX(1,1) = REF(1,1)
      INDEX(1,2) = REF(1,2)
      INDEX(1,3) = REF(1,1)
      INDEX(1,4) = REF(1,2)
      INDEX(2,2) = REF(2,2)
      INDEX(2,4) = REF(2,2)
      INDEX(3,2) = REF(3,2)
      INDEX(3,4) = REF(3,2)
      INDEX(6,2) = REF(6,2)
      INDEX(6,4) = REF(6,2)
      INDEX(7,2) = REF(7,2)
      INDEX(7,4) = REF(7,2)
      INDEX(8,2) = REF(8,2)
      INDEX(8,4) = REF(8,4)
      INDEX(9,2) = REF(9,2)
      INDEX(9,4) = REF(9,2)
C     ADJUST FACTORS BY INDEXES
      DO 300 J = 1,4
        DO 400 I = 1,9
          FACT(I,J) = FACT(I,J) / INDEX(I,J)
          FACT(I,J) = MIN (FACT(I,J), 2.0)  @ TOP IT AT 2
400     CONTINUE
300   CONTINUE
999   CONTINUE
       REWIND 18
      RETURN
      END
```

Figure E-I-22. Listing of the External Subroutine to the CS/CSS Main Preprocessor

ANNEX II TO APPENDIX E

THE AFP COMBAT SUPPORT/COMBAT SERVICE SUPPORT (CS/CSS) MODULE PROPER

Section I. OVERVIEW

E-II-1. To a large extent, the hard work of CS/CSS analysis must precede use of the CS/CSS Module proper. The development of Blue and Red measures and countermeasures by CS/CSS function (described in Annex E-1) is considered among preprocessing to the CS/CSS Module proper. The CS/CSS Module itself is designed to accept previously derived factors plus a battery of "switches" and then quite simply turn the appropriate switches on and off in the computation of CS/CSS moduli. In this view, the CS/CSS Module does little more than roll up CS/CSS functions subject to nested screening for applicability to each Blue and Red weapon type pairing and output the results in a formatted file for later input to the AFP CBT/CS/CSS Merge Module.

E-II-2. Toward accomplishment of calculation and output of CS/CSS moduli, the AFP CS/CSS Module is designed to:

a. Accept input files specifying the weapon categories to which Blue and Red weapon types belong.

b. Accept an input file specifying whether Blue and Red weapon types versus types are to be affected by CS/CSS factors in general.

c. Accept an input file specifying for each Blue and Red weapon type and each CS/CSS function whether the corresponding measures and countermeasures apply.

d. Accept an input file specifying for each CS/CSS function and each combat environment whether the Blue and Red measures and countermeasures apply.

e. Accept an input file specifying the relative weight ascribed to each CS/CSS function in each combat environment.

f. Accept an input file specifying the CS/CSS factors by function for both Blue and Red measures and countermeasures.

g. Accept an input file specifying for each CS/CSS function and each combination of a Blue and Red weapon category with a Red weapon category whether the Blue and Red measures and countermeasures apply.

h. Accept a record specifying several case identifiers.

i. Apply the above input Blue weapon type by Red weapon type in the calculation of Blue and Red CS/CSS moduli.

j. Generate a file containing Blue and Red CS/CSS moduli for each Blue and Red weapon type pairing.

E-II-3. The CS/CSS Module may be viewed as applying a succession of screens to determine just which CS/CSS measures and countermeasures apply. All CS/CSS factors, prior to screening, are assumed to have default values 1.0 in relation to each Blue/Red weapon type pairing. If all screening with respect to weapon types, weapon categories, side, functions, measures and countermeasures, and combat environment is passed for a specific combination, then a value from the nondefault table of CS/CSS factors is retrieved to replace the default 1.0 in the computation. There are at least two aspects of this process that strike many newcomers as odd. First, the "different" screens are often partially redundant. This redundancy sometimes seems to overtest a combination, and, indeed, it may. No harm is done; the extra screening just takes a little longer. Second, a retrieved "nondefault" CS/CSS factor may be 1.0, the same as the default value. Preprocessing may, indeed, yield measure and countermeasure factors of 1.0 for some functions. Replacing an assumed 1.0 by a computed 1.0 is, of course, unneccessary and could have been avoided at the expense of another test. No harm is done either way.

E-II-4. The relation of the AFP CS/CSS Module itself to the AFP System in general is shown in Figure E-II-1.

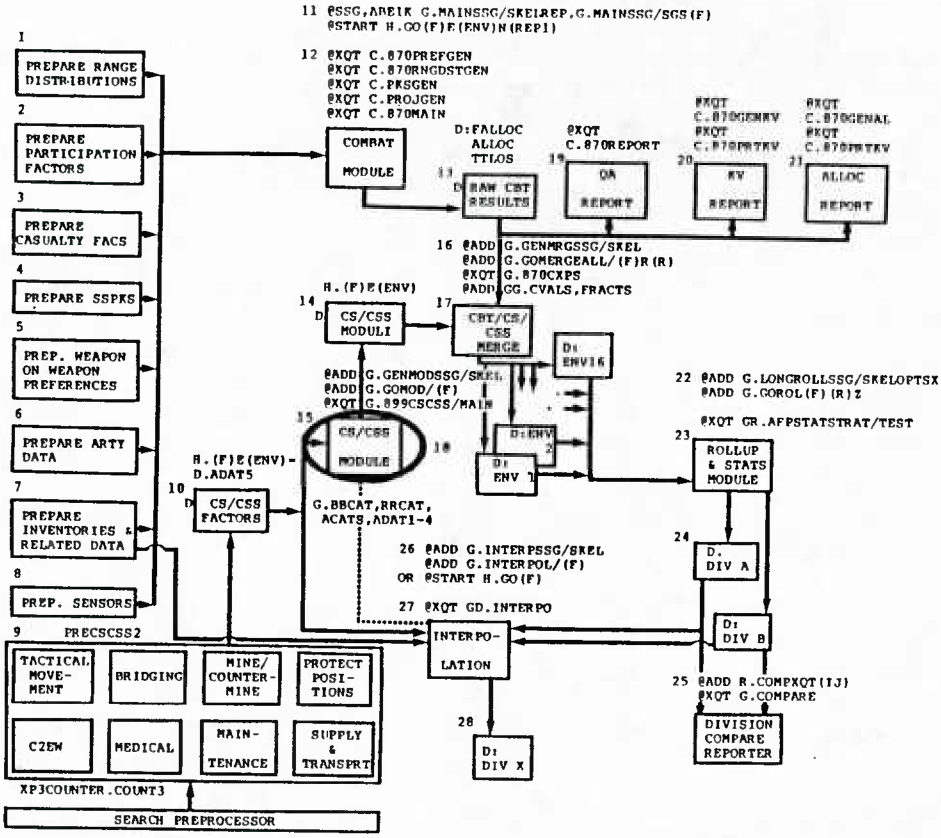


Figure E-II-1. Relation of the AFP Combat Support/Combat Service Support (CS/CSS) Module to the AFP System in General

Section II. INPUT

E-II-5. The following paragraphs describe the input to the AFP CS/CSS Module.

a. Start data record.

Columns	Variables	Data description	Format
1	---	---	1X
4-5	ITHTR	Theater code, e.g., E=Europe	A3
6-10	ITPD	Time period	I4
11-13	IVIS	Visibility 1=clear 2=degraded 2=defense light 3=delay 4=attack	I3
17-19	IDAY	Day or night 1=day 2=night	I3

b. Weapon type to weapon category link, Blue side. An example to fill array LBCAT() is provided in Figure E-II-2.

Columns	Variables	Data description	Format
(first record)			
1-3	---	---	3X
4-6	LBCAT(1)	Links weapon #1 to its category	I3
7-9	LBCAT(2)	Links weapon #2 to its category	I3
10-12	LBCAT(3)	Links weapon #3 to its category	I3
.			
.			
31-33	LBCAT(10)	Links weapon #10 to its category	I3
(Nth record)			
1-3	---	---	3X
4-6	LBCAT(N-1)*10+1)	Links weapon # (N-1)*10+1 to its category	I3
.			
.			
31-33	LBCAT(N*10)	Links weapon #N*10 to its category	I3
	(to a 6th record for a total of 60 Blue weapon types)		

```
@DATA,L 31BBCAT
DATA 9R1 SL74T9 03/06/84 12:14:46 (1)
     1.              1  1  1  1  1  2  2  2  2  2
     2.              3  3  3  3  3  4  4  4  4  5
     3.              5  5  5  5  5  6  6  6  6  6
     4.              7  7  7  7  7  8  8  8  8  8
     5.              9  9  9  9 12 12 12 12 12 12
     6.             10 10 10 10 10 11 11 11 11 11
END DATA. ERRORS: NONE. TIME:  0.505 SEC. IMAGE COUNT: 6
```

Figure E-II-2. Listing of Sample Data Assigning Blue Weapon Types (1-60) to Weapon Categories (1-12)

c. Weapon type to weapon category link, Red side. An example to fill array LRCAT() is provided in Figure E-II-3.

Columns	Variables	Data description	Format
	(records are similar to those for LBCAT above)		
		LRCAT(1)	
		.	
		.	
		LRCAT(60)	

```
@DATA,L 31RRCAT.
DATA 9R1 SL74T9 03/06/84 12:15:11 (1)
     1.              1  1  1  1  1  2  2  2  2  2
     2.              3  3  3  3  3  4  4  4  4  4
     3.              5  5  5  5  5  6  6  6  6  7
     4.              7  7  7  7  7  8  8  8  8  8
     5.              9  9  9  9  9 12 12 12 12 12
     6.             10 10 10 10 10 11 11 11 11 11
END DATA. ERRORS: NONE. TIME:  0.506 SEC. IMAGE COUNT: 6
```

Figure E-II-3. Listing of Sample Data Assigning Red Weapon Types (1-60) to Weapon Categories (1-12)

d. Weapon type screen. An example to fill arrays BW() and RW() is shown in Figure E-II-4. A value of "Y" turns on a weapon; a value of "N" turns off a weapon type. "Y" subjects a weapon type to subsequent screens; an "N" bypasses further screens for that weapon type. Furthermore, an "N" suppresses the output of the corresponding modulus. The CBT/CS/CSS Module regards "missing" moduli to possess the default value "1.0".

```
@DATA,L 31ADAT1.
DATA 9R1 SL74T9 03/06/84 12:12:45 (1)
     1.          YYYYYYYYYY
     2.          YYYYYYYYYY
     3.          YYYYYYYYYY
     4.          YYYYYYYYYY
     5.          YYYYYYYYYY
     6.          YYYYYYYYYY
     7.          YYYYYYYYYY
     8.          YYYYYYYYYY
     9.          YYYYYYYYYY
    10.          YYYYYYYYYY
    11.          YYYYYYYYYY
    12.          YYYYYYYYYY
END DATA. ERRORS: NONE. TIME:  0.508 SEC. IMAGE COUNT: 12
```

Figure E-II-4. Listing of Sample Data Specifying Whether (Y) or Not (N) CS/CSS Logic is Applicable to Blue (1-60) and Red (1-60) Weapon Types

Columns	Variables	Data description	Format
(first record)			
1-5	---	---	5X
6	BW(1)	Switch for Blue type #1	A1
7	BW(2)	Switch for Blue type #2	A2
.			
.			
15	BW(10)		
	(five similar records for Blue types #11 through #60)		
(seventh record)			
1-5	---	---	5X
6	RW(1)	Switch for Red type #1	A1
.			
.			
15	RW(10)		
	(five similar records for Red types #11 through #60)		

e. Countermeasure/measure by side by CS/CSS function by weapon type screen. Figure E-II-5 displays sample records for filling the arrays UFUN(), VFUN(), SFUN(), and TFUN(). A value of "Y" subjects the combination to additional screening. A value of "N" bypasses further screening, leaving the corresponding countermeasure or measure set to the default value of "1.0". A complete file contains 60 records if the AFP system is configured for 60 weapon types.

```
@DATA,L 31ADAT2.
DATA 9R1 SL74T9 03/06/84 12:13:06 (1)
  1.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
  2.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
  3.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
  4.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
  5.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
  6.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
  7.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
  8.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
  9.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 10.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 11.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 12.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 13.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 14.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 15.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 16.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 17.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 18.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 19.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 20.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 21.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 22.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 23.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 24.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 25.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 26.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 27.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 28.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 29.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 30.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 31.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 32.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 33.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 34.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 35.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 36.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 37.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 38.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 39.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 40.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 41.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 42.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 43.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 44.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 45.          YNYN NNNN NYNY NNNN NNNN YYYY NYNY NYNY NYNY
 46.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 47.          NNNN NNNN NNNN NNNN NNNN YYYY NYNY NYNY NYNY
 48.          NNNN NNNN NNNN NNNN NNNN YYYY NYNY NYNY NYNY
 49.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 50.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 51.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 52.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 53.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 54.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 55.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 56.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 57.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 58.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 59.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
 60.          YYYY NYNY YYYY NYNY NNNN YYYY NYNY NYNY NYNY
END DATA. ERRORS: NONE. TIME:  0.733 SEC. IMAGE COUNT: 60
```

Figure E-II-5. Listing of Sample Data Specifying by Countermeasure/Measure by Side by CS/CSS Function by Blue and Red Weapon Types Whether (Y) or Not (N) Subsequent CS/CSS Logic Is Applicable to the Combination (an "N" leaves the corresponding factor set to the default value = 1.0)

Columns	Variables	Data description	Format
(Ith record)			
1-5	---	---	5X
6	UFUN(1,I)	Blue countermeasure, first CS/CSS function, Ith Blue weapon	A1
7	VFUN(1,I)	Blue measure, first CS/CSS function, Ith Blue weapon	A1
8	SFUN(1,I)	Red countermeasure, first CS/CSS function, Ith Red weapon	A1
9	TFUN(1,I)	Red measure, first CS/CSS function, Ith Red weapon	A1
10	---	---	1X
.			
.			
46	UFUN(9,I)	Blue countermeasure, ninth CS/CSS function, Ith Blue weapon	A1
47	VFUN(9,I)	Blue measure, ninth CS/CSS function, Ith Blue weapon	A1
48	SFUN(9,I)	Red countermeasure, ninth CS/CSS function, Ith Red weapon	A1
49	TFUN(9,I)	Red measure, ninth CS/CSS function, Ith Red weapon	A1

f. Countermeasure/measure by side by CS/CSS function by combat environment screen. Figure E-II-6 displays sample records for filling the arrays EUFUN(), EVFUN(), ESFUN(), and ETFUN(). A value of "Y" subjects the combination to additional screening for the current weapon pairing. A value of "N" bypasses further screening, leaving the corresponding countermeasure or measure set to the default value of "1.0". A complete file contains 16 records, one for each combat environment.

```
@DATA,L 31ADAT3.
DATA 9R1 SL74T9 03/06/84 12:13:27 (1)
     1.               YYNY NYNY NYYN NYNN NNNN YYYY NYNY NYNY NYNY
     2.               YYNY NYNY NYYN NYNN NNNN YYYY NYNY NYNY NYNY
     3.               YYNY NYNY NYYN NNNN NNNN YYYY NYNY NYNY NYNY
     4.               NYYY NYNY YNNY NNNY NNNN YYYY NYNY NYNY NYNY
     5.               YYNY NYNY NYYN NYNN NNNN YYYY NYNY NYNY NYNY
     6.               YYNY NYNY NYYN NYNN NNNN YYYY NYNY NYNY NYNY
     7.               YYNY NYNY NYYN NNNN NNNN YYYY NYNY NYNY NYNY
     8.               NYYY NYNY YNNY NNNY NNNN YYYY NYNY NYNY NYNY
     9.               YYNY NYNY NYYN NYNN NNNN YYYY NYNY NYNY NYNY
    10.               YYNY NYNY NYYN NYNN NNNN YYYY NYNY NYNY NYNY
    11.               YYNY NYNY NYYN NNNN NNNN YYYY NYNY NYNY NYNY
    12.               NYYY NYNY YNNY NNNY NNNN YYYY NYNY NYNY NYNY
    13.               YYNY NYNY NYYN NYNN NNNN YYYY NYNY NYNY NYNY
    14.               YYNY NYNY NYYN NYNN NNNN YYYY NYNY NYNY NYNY
    15.               YYNY NYNY NYYN NNNN NNNN YYYY NYNY NYNY NYNY
    16.               NYYY NYNY YNNY NNNY NNNN YYYY NYNY NYNY NYNY
END DATA. ERRORS: NONE. TIME:  0.510 SEC. IMAGE COUNT: 16
```

Figure E-II-6. Listing of Sample Data Specifying by Countermeasure/Measure by Side by CS/CSS Function by Combat Environment Whether (Y) or Not (N) Subsequent CS/CSS Logic Is Applicable to the Combination (an "N" leaves the corresponding factor set to the default value = 1.0)

Columns	Variables	Data description	Format
(Ith record)			
1-5	---	---	5X
6	EUFUN(1,I)	Blue countermeasure, first CS/CSS function, Ith combat environment	A1
7	EVFUN(1,I)	Blue measure, first CS/CSS function, Ith combat environment	A1
8	ESFUN(1,I)	Red countermeasure, first CS/CSS function, Ith combat environment	A1
9	ETFUN(1,I)	Red measure, first CS/CSS function, Ith combat environment	A1
10	---	---	1X
.			
.			
46	EUFUN(9,I)	Blue countermeasure, ninth CS/CSS function, Ith combat environment	A1
47	EVFUN(9,I)	Blue measure, ninth CS/CSS function, Ith combat environment	A1
48	ESFUN(9,I)	Red countermeasure, ninth CS/CSS function, Ith combat environment	A1
49	ETFUN(9,I)	Red measure, ninth CS/CSS function, Ith combat environment	A1

g. Countermeasure/measure by side by CS/CSS function by Red weapon category by Blue weapon category screen. Figure E-II-7 displays sample records for filling the arrays UCAT(), VCAT(), SCAT(), and TCAT(). If screening by all preceding switches has been passed, a value of "Y" causes retrieval of the nondefault value for the corresponding countermeasure or measure. A value of "N" causes the corresponding countermeasure or measure to be set to the default value "1.0". A complete file contains 108 records (12 weapon categories x 9 CS/CSS functions).

Columns	Variables	Data description	Format
((K-1)*9+Ith record)			
1-4	---	---	4X
5	UCAT(I,1,K)	Blue countermeasure, Ith CS/CSS function, first Red category, Kth Blue category	A1
6	VCAT(I,1,K)	Blue measure, Ith CS/CSS function, first Red category, Kth Blue category	A1
7	SCAT(I,1,K)	Red countermeasure, Ith CS/CSS function, first Red category, Kth Blue category	A1
8	TCAT(I,1,K)	Red measure, Ith CS/CSS function, first Red category, Kth Blue category	A1
9	---	---	1X
.			
.			
60	UCAT(I,12,K)	Blue countermeasure, Ith CS/CSS function, 12th Red category, Kth Blue category	A1
61	VCAT(I,12,K)	Blue measure, Ith CS/CSS function, 12th Red category, Kth Blue category	A1
62	SCAT(I,12,K)	Red Countermeasure, Ith CS/CSS function, 12th Red category, Kth Blue category	A1
63	TCAT(I,12,K)	Red measure, Ith CS/CSS function, 12th Red category, Kth Blue category	A1

```
@DATA,L 31ACATS.
DATA 9R1 SL74T9 03/06/84 12:14:12 (1)
     1.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY NYYN NYYN YYYY YYYY YYYY
     2.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
     3.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY YNNY YNNY YYYY YYYY YYYY
     4.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
     5.          NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN
     6.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY
     7.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
     8.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
     9.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    10.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY NYYN NYYN YYYY YYYY YYYY
    11.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
    12.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY YNNY YNNY YYYY YYYY YYYY
    13.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
    14.          NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN
    15.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY
    16.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    17.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    18.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    19.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY NYYN NYYN YYYY YYYY YYYY
    20.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
    21.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY YNNY YNNY YYYY YYYY YYYY
    22.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
    23.          NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN
    24.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY
    25.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    26.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    27.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    28.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY NYYN NYYN YYYY YYYY YYYY
    29.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
    30.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY YNNY YNNY YYYY YYYY YYYY
    31.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
    32.          NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN NNNN
    33.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY
    34.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    35.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    36.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNY
    37.          YYYY YYYY YYYY YYYY YYYY YYYY YYYY NYYN NYYN YYYY YYYY YYYY
    38.          NYNY NYNY NYNY NYNY NYNY NYNY NYNY NYNN NYNN NYNY NYNY NYNY
```

Figure E-II-7. Listing of Sample Data Specifying by Countermeasure/Measure by Side by CS/CSS Function by Red Weapon Category (1-12) by Whether (Y) or Not (N) the Corresponding CS/CSS Factor Is Applicable to the Combination (an "N" leaves the factor set to the default value = 1.0) (page 1 of 2 pages)

39.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YNNY	YNNY	YYYY	YYYY	YYYY
40.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
41.	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
42.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY
43.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
44.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
45.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
46.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	NYYN	NYYN	YYYY	YYYY	YYYY
47.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
48.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YNNY	YNNY	YYYY	YYYY	YYYY
49.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
50.	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
51.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY
52.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
53.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
54.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
55.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	NYYN	NYYN	YYYY	YYYY	YYYY
56.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
57.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YNNY	YNNY	YYYY	YYYY	YYYY
58.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
59.	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
60.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY
61.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
62.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
63.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
64.	YNNY	YNNY	YNNY	YNNY	YNNY	YNNY	YNNY	NNNN	NNNN	YNNY	YNNY	YNNY
65.	NNNY	NNNY	NNNY	NNNY	NNNY	NNNY	NNNY	NNNN	NNNN	NNNY	NNNY	NNNY
66.	NYYN	NYYN	NYYN	NYYN	NYYN	NYYN	NYYN	NNNN	NNNN	NYYN	NYYN	NYYN
67.	NNNY	NNNY	NNNY	NNNY	NNNY	NNNY	NNNY	NNNN	NNNN	NNNY	NNNY	NNNY
68.	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
69.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY
70.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
71.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
72.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
73.	YNNY	YNNY	YNNY	YNNY	YNNY	YNNY	YNNY	NNNN	NNNN	YNNY	YNNY	YNNY
74.	NNNY	NNNY	NNNY	NNNY	NNNY	NNNY	NNNY	NNNN	NNNN	NNNY	NNNY	NNNY
75.	NYYN	NYYN	NYYN	NYYN	NYYN	NYYN	NYYN	NNNN	NNNN	NYYN	NYYN	NYYN
76.	NNNY	NNNY	NNNY	NNNY	NNNY	NNNY	NNNY	NNNN	NNNN	NNNY	NNNY	NNNY
77.	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
78.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY
79.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
80.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
81.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
82.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	NYYN	NYYN	YYYY	YYYY	YYYY
83.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
84.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YNNY	YNNY	YYYY	YYYY	YYYY
85.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
86.	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
87.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY
88.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
89.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
90.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
91.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	NYYN	NYYN	YYYY	YYYY	YYYY
92.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
93.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YNNY	YNNY	YYYY	YYYY	YYYY
94.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
95.	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
96.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY
97.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
98.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
99.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
100.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	NYYN	NYYN	YYYY	YYYY	YYYY
101.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
102.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YNNY	YNNY	YYYY	YYYY	YYYY
103.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNN	NYNN	NYNY	NYNY	NYNY
104.	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
105.	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY	YYYY
106.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
107.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY
108.	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY	NYNY

END DATA. ERRORS: NONE. TIME: 1.041 SEC. IMAGE COUNT: 108

Figure E-II-7. Listing of Sample Data Specifying by Countermeasure/Measure by Side by CS/CSS Function by Red Weapon Category (1-12) by Whether (Y) or Not (N) the Corresponding CS/CSS Factor Is Applicable to the Combination (an "N" leaves the factor set to the default value = 1.0) (page 2 of 2 pages)

h. CS/CSS function weights by combat environment. Figure E-II-8 displays sample records for filling array A(). Each record provides the weights for a single combat environment. The weights express the relative importance or significance of CS/CSS functions in a combat environment. A complete file contains 16 records.

```
@DATA,L 31ADAT4.
DATA 9R1 SL74T9 03/06/84 12:13:40 (1)
     1.            .500   .500 1.000 1.000   .000   .250 1.000 1.000 1.000
     2.            .500   .500 1.000 1.000   .000   .250 1.000 1.000 1.000
     3.            .750   .750 1.250  .500   .000 1.000 1.000 1.000 1.000
     4.           1.000 1.000 1.000  .250   .000   .250 1.000 1.000 1.000
     5.            .500   .500 1.000 1.000   .000   .250 1.000 1.000 1.000
     6.            .500   .500 1.000 1.000   .000   .250 1.000 1.000 1.000
     7.            .750   .750 1.250  .500   .000 1.000 1.000 1.000 1.000
     8.           1.000 1.000 1.000  .250   .000   .250 1.000 1.000 1.000
     9.            .500   .500 1.000 1.000   .000   .250 1.000 1.000 1.000
    10.            .500   .500 1.000 1.000   .000   .250 1.000 1.000 1.000
    11.            .750   .750 1.250  .500   .000 1.000 1.000 1.000 1.000
    12.           1.000 1.000 1.000  .250   .000   .250 1.000 1.000 1.000
    13.            .500   .500 1.000 1.000   .000   .250 1.000 1.000 1.000
    14.            .500   .500 1.000 1.000   .000   .250 1.000 1.000 1.000
    15.            .750   .750 1.250  .500   .000 1.000 1.000 1.000 1.000
    16.           1.000 1.000 1.000  .250   .000   .250 1.000 1.000 1.000
END DATA. ERRORS: NONE. TIME:  0.550 SEC. IMAGE COUNT: 16
```

Figure E-II-8. Listing of Sample Data Specifying CS/CSS Function Weights by Combat Environment

Columns	Variables	Data description	Format
(Ith record)			
1-5	---	---	5X
6-11	A(1,I)	Weight for first CS/CSS function in Ith combat environment	F6.4
12-17	A(2,I)	Weight for 2nd CS/CSS function in Ith combat environment	F6.4
.			
.			
54-59	A(9,I)	Weight for 9th CS/CSS function in Ith combat environment	F6.4

i. CS/CSS functional factors by countermeasure/measure, by side, and by function. Figure E-II-9 displays sample records for filling the array A(). A complete file consists of nine records. (In the current AFP formulation, the "fifth" CS/CSS function is "empty.")

```
                         1         2         3
                1234567890123456789012345678901234567
                -------------------------------------
Tactical movement      .96    1.15     .95    1.12
Bridging              1.00    1.05    1.00    1.22
Mine/countermine       .85    1.10    1.05    1.14
Protect. positions    1.00     .90    1.00     .87
not used              1.00    1.00    1.00    1.00
C2EW                   .80    1.00     .90    1.00
Medical               1.00     .99    1.00     .99
Maintenance           1.00     .88    1.00     .83
Supply & trans-       1.00     .95    1.00    1.00
  portation
```

Figure E-II-9. Listing of Sample Data Specifying the CS/CSS Countermeasure/Measure Factors by Side by CS/CSS Function

Columns	Variables	Data description	Format
(Ith record)			
1-5	----	---	5X
6-13	U(I)	Blue countermeasure factor for the Ith CS/CSS function	F8.4
14-21	V(I)	Blue measure factor for the Ith CS/CSS function	F8.4
22-29	S(I)	Red countermeasure factor for the Ith CS/CSS function	F8.4
30-37	T(I)	Red measure factor for the Ith CS/CSS function	F8.4

Section III. OUTPUT

E-II-6. The usual practice is to execute the AFP CS/CSS Module for a specific combat environment and for specific Blue and Red division types. A complete set of CS/CSS moduli corresponding to Blue and Red weapon type pairings is output to a file (Unit 25) as type 200 records. If there are 60 Blue and 60 Red weapon types, separate Blue and Red moduli are output for each of the 3,600 pairings for a total of 3,600 Blue and 3,600 Red moduli. Figure E-II-10 displays sample extracted records from a standard CS/CSS Module output file. The fields of each record have the following significance:

Columns	Variables	Data description	Format
(any record)			
1-6	ISCNT	Identifier of modulus record; always ISCNT=200	I6
7-9	ITHTR	Theater identifier	A3
10-13	ITPD	Timeframe	I4
14-16	IVIS	Visibility condition	I3
17-19	IPOS	Posture identifier	I3
20-22	IDAY	Day/night identifier	I3
23-27	IB	Blue weapon type identifier	I5
28-37	BF	Blue CS/CSS modulus	F10.6
38-42	IR	Red weapon type identifier	I5
43-52	RF	Red CS/CSS modulus	F10.6

ISCNT	Thea-ter	Time pd	Vis	Pos-ture	Day	Blue Wpn ID	Blue Modulus	Red Wpn ID	Red Modulus
200	E	1	1	1	1	1	.987073	1	1.194461
200	E	1	1	1	1	1	.987073	2	1.194461
200	E	1	1	1	1	1	.987073	3	1.194461
200	E	1	1	1	1	1	.987073	4	1.194461
200	E	1	1	1	1	1	.987073	5	1.194461
200	E	1	1	1	1	1	.987073	6	1.194461
200	E	1	1	1	1	1	.987073	7	1.194461
200	E	1	1	1	1	1	.987073	8	1.194461
200	E	1	1	1	1	1	.987073	9	1.194461
200	E	1	1	1	1	1	.987073	10	1.194461
•									
•									
•									
200	E	1	1	1	1	1	.987073	60	1.194461
200	E	1	1	1	1	2	.987073	1	1.194461
200	E	1	1	1	1	2	.987073	2	1.194461
200	E	1	1	1	1	2	.987073	3	1.194461
200	E	1	1	1	1	2	.987073	4	1.194461
200	E	1	1	1	1	2	.987073	5	1.194461
200	E	1	1	1	1	2	.987073	6	1.194461
200	E	1	1	1	1	2	.987073	7	1.194461
200	E	1	1	1	1	2	.987073	8	1.194461
200	E	1	1	1	1	2	.987073	9	1.194461
200	E	1	1	1	1	2	.987073	10	1.194461
•									
•									
•									
200	E	1	1	1	1	2	.987073	60	1.194461
200	E	1	1	1	1	3	.987073	1	1.194461
200	E	1	1	1	1	3	.987073	2	1.194461
200	E	1	1	1	1	3	.987073	3	1.194461
200	E	1	1	1	1	3	.987073	4	1.194461
200	E	1	1	1	1	3	.987073	5	1.194461
200	E	1	1	1	1	3	.987073	6	1.194461
200	E	1	1	1	1	3	.987073	7	1.194461
200	E	1	1	1	1	3	.987073	8	1.194461
200	E	1	1	1	1	3	.987073	9	1.194461
200	E	1	1	1	1	3	.987073	10	1.194461
•									
•									
•									
200	E	1	1	1	1	3	.987073	60	1.194461
200	E	1	1	1	1	4	.987073	1	1.194461
200	E	1	1	1	1	4	.987073	2	1.194461
200	E	1	1	1	1	4	.987073	3	1.194461
200	E	1	1	1	1	4	.987073	4	1.194461
200	E	1	1	1	1	4	.987073	5	1.194461
200	E	1	1	1	1	4	.987073	6	1.194461
200	E	1	1	1	1	4	.987073	7	1.194461
200	E	1	1	1	1	4	.987073	8	1.194461
200	E	1	1	1	1	4	.987073	9	1.194461
200	E	1	1	1	1	4	.987073	10	1.194461

Figure E-II-10. Example Extract Records from File of Blue and Red CS/CSS Moduli Output by the AFP CS/CSS Module

Section IV. RUNSTREAM

E-II-7. The CS/CSS Module must be executed as part of full-scale AFP system application. That is, in AFP system work involving execution of the AFP Combat Module, Combat Module output should always be processed by the CBT/CS/CSS Merge Module. The CBT/CS/CSS Merge Module requires, in addition to output of the Combat Module, CS/CSS moduli produced by the CS/CSS Module. But whereas, the CBT/CS/CSS Merge Module should be executed every time the Combat Module is run (e.g., 160 times in the production of H-series mechanized division potentials involving 10 replications of the Combat Module in each of 16 combat environments), the CS/CSS Module must be run only enough times to produce the needed number of distinct sets of CS/CSS moduli. In the case just mentioned, only four sets of moduli were needed, one for each of the usual AFP postures: RAPD, STATIC, RADE, and BAPD. Those sets were applied for those postures in both day and night and in both clear and degraded visibility. Four different sets of CS/CSS moduli were needed for the J-series mechanized division production. Hence, production of CS/CSS moduli for the H- and J-series division required a total of eight executions of the CS/CSS Module. The runstreams for execution of the CS/CSS Module were generated by a short SSG program. That SSG program is shown in Figure E-II-11. The CS/CSS Module is not executed as part of the AFP interpolation process described in Appendix I. Modified routines from the CS/CSS Module are imbedded within the AFP Interpolation Module. The Interpolation Module requires many of the same input as does the CS/CSS Module. The Interpolation Module requires its own special runstreams; these may be generated by an SSG program described in Appendix I.

```
@SSG,BK
SGS
FORCEIN  HM00
FORCEOUT HM80
YEAR 80
FFILE H7CSCSSI
ENV 01 02 03 04
ENVO 01 06 11 04
IVIS 1  1  1  1
IPOS 1  2  3  4
IDAY 1  1  1  1
FIL ADAT1 ADAT2 ADAT3 ADAT4 ACATS BBCAT RRCAT
NUMS  11    12    13    14    16    17    18
@EOF
SKEL
*INCREMENT F TO [FORCEIN,1]
*BRKPT,K G6GECTEST.GOMOD/[FORCEOUT,1,F,1]
#HDG UNCLASSIFIED CSCSS MOD [FORCEOUT,1,F,1]
#ELT,L G6GECTEST.GOMOD/[FORCEOUT,1,F,1]
#ASG,A [FFILE,1,1,1].      .CSCSS FACTORS IN
#ASG,A H7CSCSS.            .CSCSS MODULI OUT
#ASG,T 25.,///1000   .TEMP OUTPUT
#ASG,T 15.
*INCREMENT A TO [FIL,1]
#ASG,T [NUMS,1,A,1].,///100
#ED G6GECTEST.[FIL,1,A,1],[NUMS,1,A,1].
*LOOP     .A
*INCREMENT IE TO [ENV,1]
#ED [FFILE,1,1,1].[FORCEIN,1,F,1][[ENV,1,IE,1],15.
#DATA,L 15.
#END
#XQT G6GECTEST.899CSCSS/MAIN
    E  [YEAR,1,F,1]  [IVIS,1,IE,1]  [IPOS,1,IE,1]  [IDAY,1,IE,1]
#ED 25.,H7CSCSS.[FORCEOUT,1,F,1][[ENVO,1,IE,1]
#ERS 15.
#ERS 25.
*LOOP     .IE
*INCREMENT A TO [NUMS,1]
#FREE [NUMS,1,A,1].
*LOOP     .A
#FREE 15.
#FREE 25.
*LOOP     .F
@ECF
@EOF
```

Figure E-II-11. Example SSG Program for the Generation of CS/CSS Module Runstreams

a. SGS Section. The SGSs specify the symbols and controls necessary to generate correct runstreams. The examples in Figure E-II-11 are correct for generation of one runstream involving four executions of the CS/CSS Module for the H-series mechanized division. Different specifications of the SGSs are required for, say, the J-series division. However, symbols for both the H- and J-series divisions could be included within the SGS definitions leading to the generation of both runstreams in a single execution of the SSG program. Suitable SGS definitions can lead to generation of many runstreams in a single execution of the SSG program.

(1) FORCEIN. The SGS "FORCEIN" specifies the symbol(s) appearing in the names of elements containing the appropriate CS/CSS factors. In the example, the single symbol "HMOO" leads to generation of a single runstream. If the same SGS included a second symbol, say "JMOO," a second runstream would be generated. Of course, a second symbol here means that additional symbols must be included in some other SGSs as well.

(2) FORCEOUT. The SGS "FORCEOUT" specifies the symbol(s) to appear in the names of the output runstream element and CS/CSS Module elements. In the example, the FORCEIN and FORCEOUT symbols differ; in general, the symbols may be the same or different as convenient. If the SGS FORCEIN includes a second symbol, so too should the SGS FORCEOUT.

(3) YEAR. The SGS "YEAR" specifies a year symbol that will appear in each CS/CSS moduli record output. If the SGS FORCEIN includes a second symbol, so too should SGS YEAR.

(4) FFILE. The SGS "FFILE" specifies the name of the file containing the elements providing input CS/CSS factors. The elements corresponding to all postures or environments and all divisions specified in SGS FORCEIN must be located in the single file specified.

(5) ENV. The SGS "ENV" specifies the environmental symbols that appear in the element names of the input CS/CSS factors. The same symbols must apply to all divisions specified in SGS FORCEIN. In the example shown, only four environmental symbols are specified. From 1 to 16, such symbols are permitted.

(6) ENVO. The SGS "ENVO" specifies the environmental symbols that appear in the element names of the output CS/CSS moduli. The same symbols must apply to all divisions specified in SGS FORCEIN. In the example shown, only four environmental symbols are specified. From 1 to 16, such symbols are permitted, but the numbers of symbols in SGSs ENV and ENVO should be the same. Note that the symbols defined in the example SGSs ENV and ENVO are not identical. The symbols may be identical or different, as convenient. The differences shown in the example are simply the result of different outlooks by the AFP CS/CSS and combat teams. (Note that the posture is the same in environments 2 and 6. Also, the posture is the same in environments 3 and 11.)

(7) IVIS. The SGS "IVIS" specifies the visibility symbols that will appear in the CS/CSS moduli output records. As many symbols should appear in SGS IVIS as in SGS ENV. The same symbols are used for all divisions defined in SGS FORCEIN.

(8) IPOS. The SGS "IPOS" specifies the posture symbols that will appear in the CS/CSS moduli output records. As many symbols should appear in SGS IPOS as in SGS ENV. The same symbols are used for all divisions defined in SGS FORCEIN.

(9) IDAY. The SGS "IDAY" specifies the day/night symbols that will appear in the CS/CSS moduli output records. As many symbols should appear in SGS IDAY as in SGS ENV. The same symbols are used for all divisions defined in SGS FORCEIN.

(10) FIL. The SGS "FIL" specifies the names of elements containing data input to the CS/CSS Module. The data are described in the INPUT section.

(11) NUMS. The SGS "NUMS" specifies the logical device numbers of temporary files into which the corresponding elements named in SGS FIL are copied prior to execution of the CS/CSS Module.

b. SKELeton Section. In accord with the above-described SGSs, the SKEL section generates as many runstream elements (assumed to be used as ADD not as START elements) as there are divisions defined in SGS FORCEIN. Note that the SKEL is based on the assumption that runstream elements will all be breakpointed to file G6GECTEST and that all CS/CSS moduli output elements will be written to file H7CSCSS. Both these file names may be changed within the SKEL, or, if name changes are expected to be frequent, new SGSs may be defined in the SGS section and referenced in the SKEL section.

c. Runstream Example. Figure E-II-12 displays an example runstream generated by the example program in Figure E-II-11. Note that, because the SGS in the example specified four environments, the single runstream includes four executions of the CS/CSS Module (@XQT G6GECTEST.899CSCSS/MAIN).

```
@HDG UNCLASSIFIED CSCSS MOD HM80
@ELT,L G6GECTEST.GOMOD/HM80
@ASG,A H7CSCSSI.        .CSCSS FACTORS IN
@ASG,A H7CSCSS.           .CSCSS MODULI OUT
@ASG,T 25.,///1000    .TEMP OUTPUT
@ASG,T 15.
@ASG,T 11.,///100
@ED G6GECTEST.ADAT1,11.
@ASG,T 12.,///100
@ED G6GECTEST.ADAT2,12.
@ASG,T 13.,///100
@ED G6GECTEST.ADAT3,13.
@ASG,T 14.,///100
@ED G6GECTEST.ADAT4,14.
@ASG,T 16.,///100
@ED G6GECTEST.ACATS,16.
@ASG,T 17.,///100
@ED G6GECTEST.BBCAT,17.
@ASG,T 18.,///100
@ED G6GECTEST.RRCAT,18.
@ED H7CSCSSI.HM00F01,15.
@DATA,L 15.
@END
@XQT G6GECTEST.899CSCSS/MAIN
    E   80  1  1  1
@ED 25.,H7CSCSS.HM80E01
@ERS 15.
@ERS 25.
@ED H7CSCSSI.HM00E02,15.
@DATA,L 15.
@END
@XQT G6GECTEST.899CSCSS/MAIN
    E   80  1  2  1
@ED 25.,H7CSCSS.HM80E06
@ERS 15.
@ERS 25.
@ED H7CSCSSI.HM00E03,15.
@DATA,L 15.
@END
@XQT G6GECTEST.899CSCSS/MAIN
    E   80  1  3  1
@ED 25.,H7CSCSS.HM80E11
@ERS 15.
@ERS 25.
@ED H7CSCSSI.HM00E04,15.
@DATA,L 15.
@END
@XQT G6GECTEST.899CSCSS/MAIN
    E   80  1  4  1
@ED 25.,H7CSCSS.HM80E04
@ERS 15.
@ERS 25.
@FREE 11.
@FREE 12.
@FREE 13.
@FREE 14.
@FREE 16.
@FREE 17.
@FREE 18.
@FREE 15.
@FREE 25.
```

Figure E-II-12. Example Runstream for Execution of the AFP Combat Support/Combat Service Support (CS/CSS) Module

Section V. PROGRAM

E-II-8. Figure E-II-13 displays the basic logical flow of the AFP CS/CSS Module.

E-II-9. The source listings of the main program and subprograms of the AFP CS/CSS Module are displayed in Figures E-II-14 through E-II-21. The source listings include some intralinear comments. The following paragraphs provide additional commentary.

E-II-10. Figure E-II-14 presents the source listing of the main program of the CS/CSS Module.

a. The main program declares most of the reference arrays used in the CS/CSS Module. Most of these are given short definitions in lines 25-45 of the source listing of subprogram CSCSS shown later in Figure E-II-15. Because of their importance throughout the module, the arrays and their dimensioning parameters are defined at somewhat greater length in the following paragraphs.

(1) M is a parameter specifying the number of Blue weapon types.

(2) N is a parameter specifying the number of Red weapon types.

(3) NFUNS is a parameter specifying the number of CS/CSS functions.

(4) NENV is a parameter specifying the number of combat environments.

(5) NCATS is a parameter specifying the number of weapon categories.

(6) BW(M), the array BW() stores a 'Y' (yes) or 'N' (no) indicating whether the m-th Blue weapon type is to be subject to subsequent CS/CSS logic.

(7) RW(N), the array RW() stores a 'Y' (yes) or 'N' (no) indicating whether the n-th Red weapon type is to be subject to subsequent CS/CSS logic.

(8) UFUN(NFUNS,M), the array UFUN() stores a 'Y' (yes) or 'N' (no) indicating whether the m-th Blue weapon type is to be subject to subsequent CS/CSS logic for the countermeasure of the nfuns-th CS/CSS function.

(9) VFUN(NFUNS,M), the array VFUN() stores a 'Y' (yes) or 'N' (no) indicating whether the m-th Blue weapon type is to be subject to subsequent CS/CSS logic for the measure of the nfuns-th CS/CSS function.

(10) SFUN(NFUNS,N), the array SFUN() stores a 'Y' (yes) or 'N' (no) indicating whether the n-th Red weapon type is to be subject to subsequent CS/CSS logic for the countermeasure of the nfuns-th CS/CSS function.

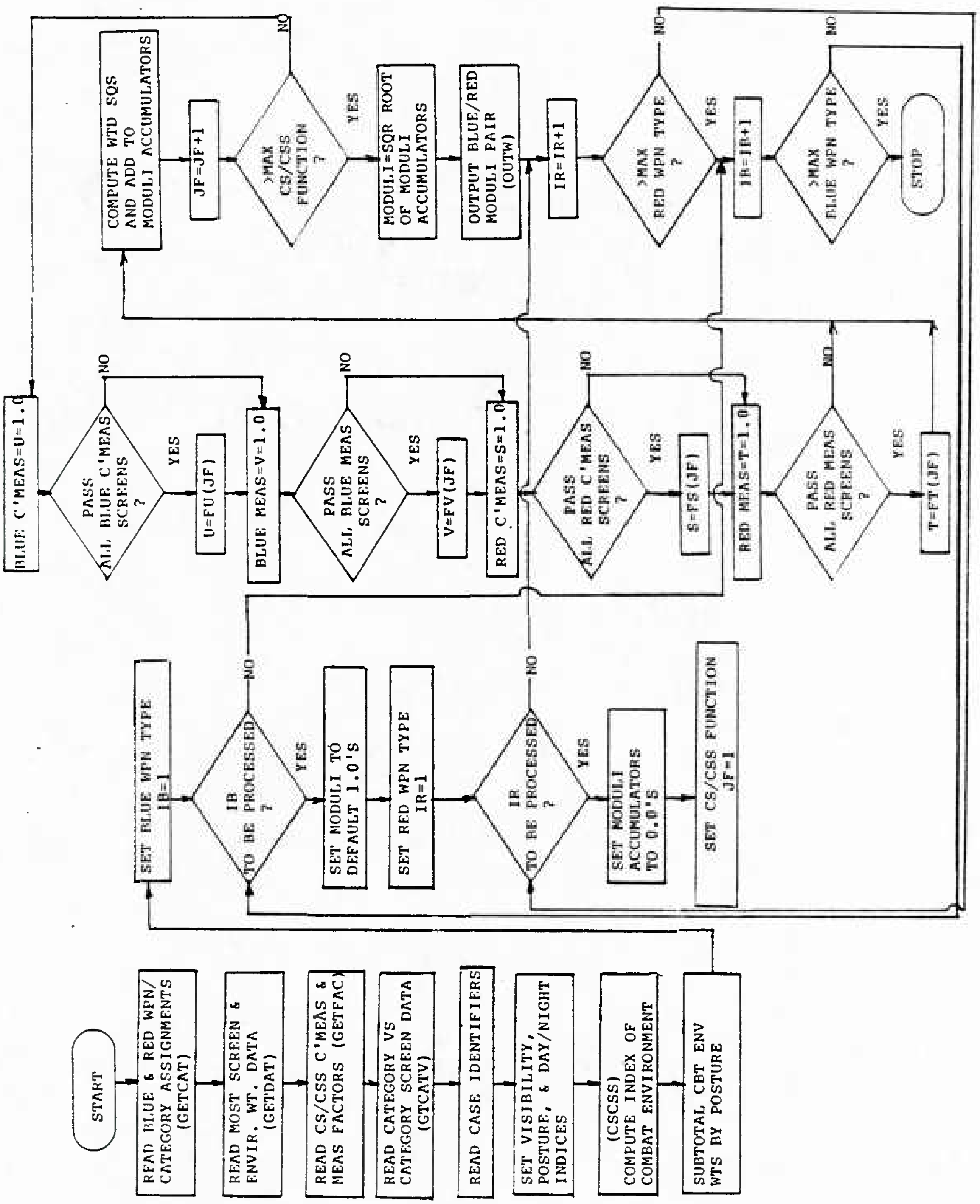


Figure E-II-13. Flow Diagram of the Basic Logic of the AFP Combat Support/Combat Service Support (CS/CSS) Module

```
      PARAMETER M=60,N=60,NFUNS=9,NENV=16,NCATS=12
C
      CHARACTER*1 BW(M),RW(N),UFUN(NFUNS,M),VFUN(NFUNS,M),
     *SFUN(NFUNS,N),TFUN(NFUNS,N),EUFUN(NFUNS,NENV),EVFUN(NFUNS,NENV),
     *ESFUN(NFUNS,NENV),ETFUN(NFUNS,NENV),UCAT(NFUNS,NCATS,NCATS),
     *VCAT(NFUNS,NCATS,NCATS),SCAT(NFUNS,NCATS,NCATS),
     *TCAT(NFUNS,NCATS,NCATS)
C
      CHARACTER ITHTR*3, RDERR*80
C
      DIMENSION A(NFUNS,NENV),LBCAT(M),LRCAT(N)
C
      COMMON/FACTOR/U(NFUNS),V(NFUNS),S(NFUNS),T(NFUNS)
      COMMON/AWRK/IWRK,ISCNT,ITHTR,ITPD,IVISX,IPOSX,IDAYX
C
      DATA NPOS,NDAY/4,2/
C
C
C
      INFILE=17
      CALL GETCAT(LBCAT,M,'BBCAT-FILE',INFILE)
      INFILE=18
      CALL GETCAT(LRCAT,N,'RRCAT-FILE',INFILE)
C
      CALL GETDAT(M,N,NFUNS,NENV,UFUN,VFUN,SFUN,TFUN,
     *EUFUN,EVFUN,ESFUN,ETFUN,BW,RW,A)
C
      CALL GETFAC(U,V,S,T,NFUNS)
C
      CALL GTCATV(UCAT,VCAT,SCAT,TCAT,NFUNS,NCATS)
C
      IWRK=25
      ISCNT=200
      READ (5,100,ERR=110) ITHTR,ITPD,IVIS,IPOS,IDAY
  100 FORMAT(1X,A3,I4,3I3)
      GO TO 120
  110 READ(0,115)RDERR
  115 FORMAT(A80)
      PRINT*, 'ERR IN READING INPUT VARIABLES:'
      PRINT*, '               RECORD=',RDERR
      STOP
  120 PRINT*,'AT END - INPUT VARIABLES'
C     ITHTR='  E'
C     ITPD=1
C     DO 50 IVIS=1,1
      IVISX=IVIS
C     DO 40 IPOS=1,1
      IPOSX=IPOS
      JPOS=4
      IF(IPOSX.EQ.4) JPOS=1
C     DO 30 IDAY=1,1
      IDAYX=IDAY
      CALL CSCSS(M,N,NFUNS,IVIS,IPOS,IDAY,NENV,UFUN,VFUN,SFUN,TFUN,
     *EUFUN,EVFUN,ESFUN,ETFUN,BW,RW,NPOS,NDAY,A,JPOS,
     *LBCAT,LRCAT,UCAT,VCAT,SCAT,TCAT,NCATS)
C  30 CONTINUE
C  40 CONTINUE
C  50 CONTINUE
      STOP 'DONE'
      END
```

Figure E-II-14. Source Listing of Main Program CS/CSS/MAIN of the AFP CS/CSS Module

(11) TFUN(NFUNS,N), the array TFUN() stores a 'Y' (yes) or 'N' (no) indicating whether the n-th Red weapon type is to be subject to subsequent CS/CSS logic for the measure of the nfuns-th CS/CSS function.

(12) EUFUN(NFUNS,NENV), the array EUFUN() stores a 'Y' (yes) or 'N' (no) indicating whether subsequent CS/CSS logic is to be applied for the Blue countermeasure of the nfuns-th CS/CSS function in the nenv-th combat environment.

(13) EVFUN(NFUNS,NENV), the array EVFUN() stores a 'Y' (yes) or 'N' (no) indicating whether subsequent CS/CSS logic is to be applied for the Blue measure of the nfuns-th CS/CSS function in the nenv-th combat environment.

(14) ESFUN(NFUNS,NENV), the array ESFUN() stores a 'Y' (yes) or 'N' (no) indicating whether subsequent CS/CSS logic is to be applied for the Red countermeasure of the nfuns-th CS/CSS function in the nenv-th combat environment.

(15) ETFUN(NFUNS,NENV), the array ETFUN() stores a 'Y' (yes) or 'N' (no) indicating whether subsequent CS/CSS logic is to be applied for the Red measure of the nfuns-th CS/CSS function in the nenv-th combat environment.

(16) UCAT(NFUNS,NCATS,NCATS), the array UCAT() store a 'Y' (yes) or 'N' (no) indicating whether subsequent CS/CSS logic is to be applied for the Blue countermeasure of the nfuns-th CS/CSS function involving pairings of a j-th Red category weapon with a k-th Blue category weapon.

(17) VCAT(NFUNS,NCATS,NCATS), the array VCAT() stores a 'Y' (yes) or 'N' (no) indicating whether subsequent CS/CSS logic is to be applied for the Blue measure of the nfuns-th CS/CSS function involving pairings of a j-th Red category weapon with a k-th Blue category weapon.

(18) SCAT(NFUNS,NCATS,NCATS), the array SCAT() STORES A 'Y' (yes) or 'N' (no) indicating whether subsequent CS/CSS logic is to be applied for the Red countermeasure of the nfuns-th CS/CSS function involving pairings of a j-th Red category weapon with a k-th Blue category weapon.

(19) TCAT(NFUNS,NCATS,NCATS), the array TCAT() stores a 'Y' (yes) or 'N' (no) indicating whether subsequent CS/CSS logic is to be applied for the Red Measure of the mfuns-th CS/CSS function involving pairings of a j-th Red category weapon with a k-th Blue category weapon.

(20) A(NFUNS,NENV), the array A() stores numerical weights (usually on the interval 0.00 - 1.25) to be applied as multipliers of the nfuns-th component (corresponding to the nfuns-th CS/CSS function) of the CS/CSS modulus for nenv-th combat environment.

(21) LBCAT(M), the array LBCAT() stores the index of the ncats-th weapon category corresponding to the m-th Blue weapon type.

(22) LRCAT(N), the array LRCAT() stores the index of the ncats-th weapon category corresponding to the n-th Red weapon type.

(23) U(NFUNS), the array U() stores the Blue countermeasure factors corresponding to the nfuns-th CS/CSS functions.

(24) V(NFUNS), the array V() stores the Blue measure factors corresponding to the nfuns-th CS/CSS functions.

(25) S(NFUNS), the array S() stores the Red countermeasure factors corresponding to the nfuns-th CS/CSS functions.

(26) T(NFUNS), the array T() stores the Red measure factors corresponding to the nfuns-th CS/CSS functions.

b. The main program declares arrays, initializes a few variables, calls data input routines, calls the principal subprogram CSCSS to compute and output moduli, and terminates the execution of the module. The call to the subprogram CSCSS originally lay within a triply nested loop structure over combat environments; however, the limiting statements of the loops have been converted to comment statements inasmuch as only one combat environment is processed in a single execution of the module.

(1) Line 1 declares the parameters defined in paragraph E-II-10a immediately above.

(2) Lines 3-7, 11, and 14 declare the arrays defined in paragraph E-II-10a immediately above.

(3) Line 21 calls subprogram GETCAT to input and store the weapon categories corresponding to the Blue weapon types.

(4) Line 23 calls subprogram GETCAT to input and store the weapon categories corresponding to the Red weapon types.

(5) Lines 25 and 26 call subprogram GETDAT to input and store data for the 11 arrays (along with their dimensions) specified in the argument list and defined in paragraph E-II-10a immediately above.

(6) Line 28 calls subprogram GETFAC to input and store the Blue and Red countermeasure and measure factors for the CS/CSS functions. The factors are stored in the four arrays specified in the argument list as defined in paragraph E-II-10a above.

(7) Line 30 calls subprogram GTCATV to input and store the data for the four arrays specified in the argument list and defined in paragraph E-II-10a above.

(8) Line 32 sets scratch variable IWRK to 25, the index of the unit to which CS/CSS moduli are to be written.

(9) Line 33 sets scratch variable ISCNT to 200, the index of the type records to be output.

(10) Line 34 reads several case and environment identifiers whose values, except for posture, are not used for anything significant in the current implementation of the program. The identifiers are included within the records containing CS/CSS moduli at output time, but the values may have to be changed via the system Editor in order to be acceptable as input by the CBT/CS/CSS Merge Module.

(11) Line 36 transfers control beyond the error message sequence (lines 37-41) invoked in the event of a read error in line 34.

(12) Line 42 prints message to the effect that data input was successful.

(13) Line 46 sets scratch variable IVISX to the previously read visibility index.

(14) Line 48 sets scratch variable IPOSX to the previously read (Blue) posture index.

(15) Line 49 sets scratch variable JPOS to an assumed Red attack posture index, '4', corresponding to the first three Blue postures.

(16) However, in Blue posture 4 (Blue attack), Red is defending, so set the Red scratch variable posture index to 1 (defend) line 50.

(17) Line 52 sets scratch variable IDAYX to the previously read day/night index.

(18) Lines 53-55 call subprogram CSCSS to generate the CS/CSS moduli in accord with the data already read. The argument list of the call contains the addresses and dimensions of the arrays needed and defined in paragraph E-II-10a above.

(19) Line 59 provides normal termination of the CS/CSS Module.

E-II-11. Figure E-II-15 presents the source listing of the principal subprogram, CSCSS, of the CS/CSS Module. Subprogram CSCSS possesses a rather lengthy list of formal arguments. Most of these arguments are the addresses and dimensions already defined in paragraph E-II-10a on the main program of the CS/CSS Module. All critical data have been input to the CS/CSS Module before entry to subprogram CSCSS.

```
      SUBROUTINE CSCSS(M,N,NFUNS,IVIS,IPOS,IDAY,NENV,
     *UFUN,VFUN,SFUN,TFUN,EUFUN,EVFUN,ESFUN,ETFUN,
     *BW,RW,NPOS,NDAY,A,JPOS,LBCAT,LRCAT,
     *UCAT,VCAT,SCAT,TCAT,NC)
C
      PARAMETER NBSTEP=1, NRSTEP=1
C
      CHARACTER*1 BW(M),RW(N),UFUN(NFUNS,M),VFUN(NFUNS,M),
     *SFUN(NFUNS,N),TFUN(NFUNS,N),EUFUN(NFUNS,NENV),EVFUN(NFUNS,NENV),
     *ESFUN(NFUNS,NENV),ETFUN(NFUNS,NENV),NO,
     *UCAT(NFUNS,NC,NC),VCAT(NFUNS,NC,NC),SCAT(NFUNS,NC,NC),
     *TCAT(NFUNS,NC,NC)
C
      DIMENSION A(NFUNS,NENV),LBCAT(M),LRCAT(N)
C
      NO='N'
C
C     A U,V,S, OR T ELEMENT IS BUILT AS THE DIAGONAL OF A HYPER-
C     PARALLELOPIPED FOR THE L-TH ENVIRONMENT:
C       ELE = SQRT(SUM(A(K,L)*F(K)^2;K)/SUM(A(K,L);K))
C
C     IN CURRENT FORM, DOES NOT DEPEND ON FORCE MASSES OR RATIO!!!
C
C     ARRAYS & THINGS:
C       A(K,L)             WT OF K-TH FUNCT IN L-TH ENVIRONMENT.
C       BF                 VLUE NET FACTOR OVER ALL FUNCTS. L ENVIR.
C       RF                 RED NET FACTOR OVER ALL FUNCTS. L ENVIR.
C       BW(I)              SW, WHETHER VLUE TYPE I AFFECTED.
C       RW(J)              SW, WHETHER RED TYPE J AFFECTED.
C       UFUN(K,I)          SW, WHETHER K FUNCT AFFECTS B-TYPE I U'S.
C       VFUN(K,I)          SW, WHETHER K FUNCT AFFECTS B-TYPE I V'S.
C       SFUN(K,J)          SW, WHETHER K FUNCT AFFECTS R-TYPE J S'S.
C       TFUN(K,J)          SW, WHETHER K FUNCT AFFECTS R-TYPE J T'S.
C       EUFUN(K,L)         SW, WHETHER K FUNCT & L ENVIR AFFECT U'S.
C       EVFUN(K,L)         SW, WHETHER K FUNCT & L ENVIR AFFECT V'S.
C       ESFUN(K,L)         SW, WHETHER K FUNCT & L ENVIR AFFECT S'S.
C       ETFUN(K,L)         SW, WHETHER K FUNCT & L ENVIR AFFECT T'S.
C       UCAT(K,J,I)        SW, WHETHER K FUNCTION, R-TYPE J, AND B-TYPE I
C                              AFFECT U'S.
C       VCAT(K,J,I)        SW, WHETHER K FUNCTION, R-TYPE J, AND B-TYPE I
C                              AFFECT V'S.
C       SCAT(K,J,I)        SW, WHETHER K FUNCTION, R-TYPE J, AND B-TYPE I
C                              AFFECT S'S.
C       TCAT(K,J,I)        SW, WHETHER K FUNCTION, R-TYPE J, AND B-TYPE I
C                              AFFECT T'S.
C
C     COMPUTE INDEX OF ENVIRONMENT
C
      LENV=(IVIS-1)*NPOS*NDAY+(IDAY-1)*NPOS+IPOS
      LENVJ=(IVIS-1)*NPOS*NDAY+(IDAY-1)*NPOS+JPOS
```

Figure E-II-15. Source Listing of Subprogram CSCSS
of the AFP CS/CSS Module
(page 1 of 2 pages)

```
      C
      C        COMPUTE NORM
      C
               AC=0.0
               AD=0.0
               DO 50 JF=1,NFUNS
               AC=AC+A(JF,LENV)
               AD=AD+A(JF,LENVJ)
            50 CONTINUE
      C
               DO 1000 IB=1,M,NBSTEP
               IF (BW(IB).EQ.NO) GO TO 1000
               IBCAT=LBCAT(IB)
               DO 900 IR=1,N,NRSTEP
               BF=1.0
               RF=1.0
               IF (RW(IR).EQ.NO) GO TO 900
               IRCAT=LRCAT(IR)
               RR=0.0
               BB=0.0
               DO 800 JF=1,NFUNS
               U=1.0
               IF(UFUN(JF,IB).EQ.NO) GO TO 710
               IF(EUFUN(JF,LENV).EQ.NO) GO TO 710
               IF(UCAT(JF,IRCAT,IBCAT).EQ.NO) GO TO 710
               U=FU(JF)
           710 V=1.0
               IF(VFUN(JF,IB).EQ.NO) GO TO 720
               IF(EVFUN(JF,LENV).EQ.NO) GO TO 720
               IF(VCAT(JF,IRCAT,IBCAT).EQ.NO) GO TO 720
               V=FV(JF)
           720 S=1.0
               IF(SFUN(JF,IR).EQ.NO) GO TO 730
               IF(ESFUN(JF,LENV).EQ.NO) GO TO 730
               IF(SCAT(JF,IRCAT,IBCAT).EQ.NO) GO TO 730
               S=FS(JF)
           730 T=1.0
               IF(TFUN(JF,IR).EQ.NO) GO TO 740
               IF(ETFUN(JF,LENV).EQ.NO) GO TO 740
               IF(TCAT(JF,IRCAT,IBCAT).EQ.NO) GO TO 740
               T=FT(JF)
           740 CONTINUE
               UVST=(U*V)/(S*T)
               UVST=UVST*UVST
               BB=BB+A(JF,LENV)*UVST
               RR=RR+A(JF,LENVJ)/UVST
           800 CONTINUE
               BF=SQRT(BB/AC)
               RF=SQRT(RR/AD)
               CALL OUTW(BF,RF,IB,IR)
           900 CONTINUE
          1000 CONTINUE
               RETURN
               END
      C
```

Figure E-II-15. Source Listing of Subprogram CSCSS
of the AFP CS/CSS Module
(page 2 of 2 pages)

a. The CSCSS subprogram, on a single call (the current normal implementation of the subprogram), is basically a triply-nested loop structure over Blue weapon types (the outer loop), Red weapon types (the next inner loop), and the CS/CSS functions (the innermost loop).

(1) Ultimately, the products of the CS/CSS Module are to be used to "modulate" results of the AFP Combat Module. The basic output of the Combat Module are estimated kills and losses by each Blue and Red weapon type under assumed normed CS/CSS conditions for a single combat environment. But generally, forces need not and are not equipped and manned to perform CS/CSS functions at exactly normed levels under all combat environments. Therefore, it is necessary to modulate (adjust) the raw output of the Combat Module to compensate for the differences between normed and estimated CS/CSS levels of support and between opposing sides. The CS/CSS preprocessing treats each CS/CSS function separately--in effect, without regard to interaction among CS/CSS functions. There is an abundant literature on the effects of separate CS/CSS functions. Those references are uneven in depth and reliability. Nevertheless, the best of those references comprise the primary sources on which AFP's CS/CSS preprocessing is based. The literature is much less rich with respect to the net effects of combinations of the CS/CSS functions at different support levels for the AFP combat environments. The next paragraph addresses the simpler issue of what the CS/CSS Module would do if there were only a single CS/CSS function about which to worry. The paragraph after that addresses the more difficult issue of how to combine several CS/CSS functions.

(2) For each Blue/Red weapon type pair, subprogram CSCSS applies from one to a variety of screening tests to determine which, if any, CS/CSS functions and their corresponding Blue and Red measures and countermeasures are to be applied at other than 1.0 default values. Each CS/CSS function involves a term of the form:

$(U \times V) \ / \ (S \times T)$ for Blue, and its reciprocal

$(S \times T) \ / \ (U \times V)$ for Red, where:

U represents the countermeasure for Blue

V represents the measure for Blue

S represents the countermeasure for Red

T represents the measure for Red

Suppose there is only one CS/CSS function. The U, V, S, and T values are first all assumed equal to 1.0. However, if logical tests are passed, one or more of these assumed values are replaced by values read and stored beforehand by the main program. Of course, a replacement value may be 1.0, but typically the replacement value is something other than 1.0. If there were only one CS/CSS function, the term (UxV)/(SxT) alone would be used later in the AFP CBT/CS/CSS Merge Module as a simple multiplier of the kills of the Red weapon type achieved by the Blue weapon type in the so-called "global exchange ratio," or GER method. In the so-called "local exchange ratio," or LER method, the same term would be applied by the CBT/CS/CSS Merge Module as a simple multiplier of the ratio of the kills of the Red type to the losses by the Blue type (in that particular type-on-type engagement).

(3) Now suppose that there are only two CS/CSS functions. These imply two Blue terms of the now familiar form (UxV)/(SxT). To distinguish CS/CSS functions, introduce (for the i-th CS/CSS function) the new term UVST (i). The obvious problem now is to construct a multiplier that depends (in the assumed two CS/CSS function case) on both UVST (1) and UVST (2). The dominant schools of thought sharply divided between means of additive and multiplicative functions. That is, some analysts favored the form A = (UVST(1) + UVST(2))/2 (an arithmetic mean), and others favored G = SQRT(UVST(1) x UVST(2)) (a geometric mean). Both these forms have obvious extensions to more CS/CSS functions. The AFP development team favored a third approach that seemed to provide a better foundation for later generalization. Like several other aspects of AFP, the approach may best be described as heuristic. CS/CSS functions are assumed to be definable in a multidimensional space in which both direction and length are significant. In the special case of just two CS/CSS functions, the underlying space is assumed to be two-dimensional. In general two CS/CSS functions may be considered as mutually supportive, independent, or even counterproductive. In CS/CSS space the extent to which CS/CSS functions are related is associated with their relative direction. Two functions known or assumed to be perfectly contributing to each other in kind "point" in the same direction. Two functions in perfect conflict "point" in opposite directions. Two functions that are independent are considered "perpendicular" to each other. Yet even independent functions are considered to have a net effect greater than either alone. Two independent functions are considered analogous to the adjacent sides of a rectangle with their net result analogous to the diagonal of the rectangle. Given that the functions are already properly scaled, their net effect then has the form:

NET = SQRT(UVST(1)^2 + UVST(2)^2)

This simple Pythagorean rule is next generalized to admit weights expressing the relative importance of functions and perhaps adjusting for scale:

NET = SQRT((a(1) x UVST(1)^2 + a(2) x UVST(2)^2)/(a(1) + a(2)))

For two CS/CSS functions, this last expression is just the basic form chosen for the AFP CS/CSS modulus. The form is extended directly to all CS/CSS functions. The weights are permitted to depend on combat environment, L, yielding a current form:

NET(L) = SQRT(SUM(a(i,L)*UVST(i);i)/SUM(a(i,L);i)) where SUM (;i) represents summation with respect to the index i. The form yields the CS/CSS modulus as the diagonal of a hyperrectangle with weighted (rescaled) edges. As perhaps appropriate at a later date, the underlying figure may be generalized to a hyperparallelopiped permitting somewhat longer or shorter diagonals as the extent to which functions overlap or underlap becomes better understood. The form is considered useful for relatively small changes in functions relative to their norms. It is not argued that the form is satisfactory for one or more functions at zero levels of support. The form, in keeping with the usual "static" emphasis within AFP, is not intended to reflect the dynamics of support leads and lags.

b. Lines 1-3 provide the formal argument list of subprogram CSCSS. As already noted above for the main program, most of the arguments are addresses and dimensions of arrays. The arrays have been described in paragraph E-II-10.a. Short definitions are also provided in lines 25-45 of subprogram CSCSS. "SW" represents a "yes/no SWitch."

c. Lines 49 and 50 set working variables LENV and LENVJ to the Blue and Red combat environment indices (1-NENV) to be referenced later in extracting the appropriate environmental weights from array A().

d. Lines 54-59 set working variables AC and AD to the sums of the appropriate environmental weights for Blue and Red, respectively.

e. Lines 61 and 102 define the bounds of the loop over Blue weapon types. In normal use, NBSTEP = 1 so that all Blue weapon types are examined. Larger steps have been used for test purposes.

f. Line 62 checks whether the Blue weapon type is to be processed. If not, subprogram CSCSS does not output any moduli for the Blue weapon type. When the CBT/CS/CSS Merge Module processes the file of CS/CSS moduli, all "missing values" are assumed to be 1.0.

g. Line 63 sets scratch variable IBCAT to the weapon category index of Blue weapon type IB.

h. Lines 64 and 101 define the bounds of the loop over Red weapon types. In normal use, NRSTEP = 1 so that all Red weapon types are examined. Larger steps have been used for test purposes.

i. Lines 65 and 66 set scratch variables BF and RF to 1.0 as the default moduli for Blue and Red respectively.

j. Line 67 checks whether the Red weapon types is to be processed. if not, subprogram CSCSS does not output a record for the IB/IR weapon types pairing. When the CBT/CS/CSS Merge Module processes the file of CS/CSS moduli, all "missing values" are assumed to be 1.0.

k. Line 68 sets scratch variable IRCAT to the weapon category of Red weapon type IR.

l. Lines 69 and 70 set scratch variables RR and BB to 0.0 in preparation for receiving the weighted sums of squares of factors over CS/CSS functions for the current IB/IR weapon type pairing.

m. Lines 71 and 97 define the bounds of the loop over CS/CSS functions.

n. Line 72 sets scratch variable U to the default value 1.0 for the Blue measure corresponding to CS/CSS function JF.

o. Lines 73-75 apply tests to check whether the nondefault Blue countermeasure is applicable for the current combinations of:

(1) Blue weapon type IB and CS/CSS function JF.

(2) CS/CSS function JF and combat environment LENV.

(3) CS/CSS function JF, Red weapon category IRCAT, and Blue weapon category IBCAT.

p. Only if all three tests are non-'N' does line 76 call function FU to return a nondefault value for the Blue countermeasure.

q. Line 77 sets scratch variable V to the default value 1.0 for the Blue measure corresponding to CS/CSS function JF.

r. Lines 78-80 apply tests to check whether the nondefault Blue measure is applicable for the current combinations of:

(1) Blue weapon type IB and CS/CSS function JF.

(2) CS/CSS function JF and combat environment LENV.

(3) CS/CSS function JF, Red weapon category IRCAT, and Blue weapon category IBCAT.

s. Only if all three tests are non-'N' does line 81 call function FV to return a nondefault value for the Blue measure.

t. Line 82 sets scratch variable S to the default value 1.0 for the Red countermeasure corresponding to CS/CSS function JF.

u. Lines 83-85 apply tests to check whether the nondefault Red countermeasure is applicable for the current combinations of:

(1) Red weapon type IR and CS/CSS function JF.

(2) CS/CSS function JF and combat environment LENV.

(3) CS/CSS function JF, Red weapon category IRCAT, and Blue weapon category IBCAT.

v. Only if all three tests are non-'N' does line 86 call function FS to return a nondefault value for the Red countermeasure.

w. Line 87 sets scratch variable T to the default value 1.0 for the Red measure corresponding to CS/CSS function JF.

x. Lines 88-90 apply tests to check whether the nondefault Red measure is applicable for the current combinations of:

(1) Red weapon type IR and CS/CSS function JF.

(2) CS/CSS function JF and combat environment LENV.

(3) CS/CSS function JF, Red weapon category IRCAT, and Blue weapon category IBCAT.

y. Only if all three tests are non-'N', does line 91 call function FT to return a nondefault value for the Red measure.

z. Line 93 sets scratch variable UVST to the ratio of above-determined countermeasure and measure factors.

aa. Line 94 simply squares UVST.

ab. Lines 95 and 96 update the working variables BB and RR with the weighted partial sum through the current CS/CSS function JF. BB accumulates the sum for Blue, and RR accumulates the sum for Red. Note that for Red, the reciprocal of UVST is applied.

ac. Lines 98 and 99 sets scratch variables BF and RF to the CS/CSS moduli of Blue weapon type IB and Red weapon type IR, respectively, for the engagement of those two specific types. Note that in keeping with the description in paragraph a(3) above, BF and RF are the "diagonals of the rectangular hyperparallelopiped with weighted or rescaled edges."

ad. Line 100 calls subprogram to output a standard type 200 AFP record for the pairing of Blue type IB and Red type IR. The record is to contain the weapon type identifiers and the Blue (BF) and Red (RF) CS/CSS moduli for the IB/IR type pairing.

ae. Line 103 returns control to the main program of the CS/CSS Module.

E-II-12. Figure E-II-16 presents source listings for function subprograms FU, FV, FS, and FT of the CS/CSS Module. The functions are all trivial, serving merely to return values from the corresponding arrays U(), V(), S(), and T() given the index of the CS/CSS function of interest. The CS/CSS Module structure was originally designed in anticipation of later generalization of the scheme for determining the factors corresponding to Blue and Red measures and countermeasures. However, as of this writing, no need had risen requiring anything more than the simple referencing of arrays.

```
      FUNCTION FU(JZ)
      COMMON/FACTOR/U(9),V(9),S(9),T(9)
      FU=U(JZ)
      RETURN
      END
C
      FUNCTION FV(JZ)
      COMMON/FACTOR/U(9),V(9),S(9),T(9)
      FV=V(JZ)
      RETURN
      END
C
      FUNCTION FS(JZ)
      COMMON/FACTOR/U(9),V(9),S(9),T(9)
      FS=S(JZ)
      RETURN
      END
C
      FUNCTION FT(JZ)
      COMMON/FACTOR/U(9),V(9),S(9),T(9)
      FT=T(JZ)
      RETURN
      END
```

Figure E-II-16. Source Listing of Subprograms FU, FV, FS, and FT of the AFP CS/CSS Module

E-II-13. Figure E-II-17 presents the source listing of subprogram GETCAT of the CS/CSS Module. GETCAT is called from the main program to read and store indices specifying to what weapon category each weapon type belongs. The formal arguments of GETCAT represent:

a. LCAT is the address of the array that is to receive the weapon category indices. The main program maintains separate arrays for Blue and Red weapons. Hence, the main program calls GETCAT twice, once for each of the Blue and Red arrays.

b. N is the length of array LCAT.

c. FNAME is the name associated with the input file. The name is printed in the event of error or successful completion of input.

d. INFILE is the unit number on which the input data are to be found.

```
          SUBROUTINE GETCAT(LCAT,N,FNAME,INFILE)
          CHARACTER RDERR*80,FNAME*10
          DIMENSION LCAT(N)
          READ(INFILE,100,ERR=110) (LCAT(I),I=1,N)
      100 FORMAT(3X,10I3)
          GO TO 120
      110 READ(0,115) RDERR
      115 FORMAT(A80)
          PRINT*, 'ERR IN READING ', FNAME, '  RECORD= ', RDERR
          STOP
      120 PRINT*, 'AT END - ', FNAME
          RETURN
          END
```

Figure E-II-17. Source Listing of Subprogram GETCAT of the AFP CS/CSS Module

E-II-14. Figure E-II-18 presents the source listing of subprogram GETDAT of the CS/CSS Module. GETDAT is called once from the main program in order to read and store four categories of data. The formal arguments of GETDAT correspond to actual arguments already defined for the main program. The arguments are the addresses and dimensions of arrays.

a. Lines 12-16 are devoted to the arrays specifying whether CS/CSS logic is to be applied to specific weapon types--array BW() for Blue, array RW() for Red.

b. Lines 18-24 are devoted to the arrays specifying whether further CS/CSS logic is to be applied for Blue and Red measures and countermeasures for specific combinations of CS/CSS functions and weapon types.

c. Lines 26-31 are devoted to the arrays specifying whether further CS/CSS logic is to be applied for Blue and Red measures and couuntermeasures for specific combinations of CS/CSS functions and combat environments.

d. Lines 33-41 are devoted to the arrays specifying the weights to be applied by function and combat environment in building CS/CSS moduli over all functions.

```
1           SUBROUTINE GETDAT(M,N,NFUNS,NENV,UFUN,VFUN,
2          *SFUN,TFUN,EUFUN,EVFUN,ESFUN,ETFUN,BW,RW,A)
3     C
4           CHARACTER RDERR*80,FNAME*10
5     C
6           CHARACTER*1 BW(M), RW(N), UFUN(NFUNS,M), VFUN(NFUNS,M),
7          *SFUN(NFUNS,NENV), TFUN(NFUNS,NENV), EUFUN(NFUNS,NENV),
8          *EVFUN(NFUNS,NENV), ESFUN(NFUNS,NENV), ETFUN(NFUNS,NENV)
9     C
10          DIMENSION A(NFUNS,NENV)
11    C
12          FNAME='ADAT1 FILE'
13          READ(11,100,ERR=310) (BW(I),I=1,M)
14          READ(11,100,ERR=310) (RW(I),I=1,N)
15      100 FORMAT(5X,10A1)
16          PRINT*, 'AT END - ', FNAME
17    C
18          FNAME='ADAT2 FILE'
19          DO 150 I=1,M
20          READ(12,200,ERR=310) (UFUN(K,I),VFUN(K,I),SFUN(K,I),
21         *TFUN(K,I),K=1,NFUNS)
22      150 CONTINUE
23      200 FORMAT(5X,9(4A1,1X))
24          PRINT*, 'AT END - ', FNAME
25    C
26          FNAME='ADAT3 FILE'
27          DO 250 L=1,NENV
28          READ(13,200,ERR=310) (EUFUN(K,L),EVFUN(K,L),ESFUN(K,L),
29         *ETFUN(K,L),K=1,NFUNS)
30      250 CONTINUE
31          PRINT*, 'AT END - ', FNAME
32    C
33          FNAME='ADAT4 FILE'
34          READ(14,300,ERR=310) ((A(K,L),K=1,NFUNS),L=1,NENV)
35      300 FORMAT(5X,9F6.4)
36          GO TO 320
37      310 READ(0,315) RDERR
38      315 FORMAT(A80)
39          PRINT*, 'ERR IN READING ', FNAME, '  RECORD= ', RDERR
40          STOP
41      320 PRINT*, 'AT END - ', FNAME
42          RETURN
43          END
```

Figure E-II-18. Source Listing of Subprogram GETDAT of the AFP CS/CSS Module

E-II-15. Figure E-II-19 presents the source listing of subprogram GETFAC of the CS/CSS Module. GETFAC is called once from the main program in order to read and store the nondefault values of factors corresponding to the Blue and Red measures and countermeasures by CS/CSS function. The formal arguments of GETFAC correspond to actual arrays and their length defined within the main program's description above.

```
      SUBROUTINE GETFAC(U,V,S,T,NFUNS)
      CHARACTER*30 RDERR
      DIMENSION U(NFUNS),V(NFUNS),S(NFUNS),T(NFUNS)
C
      READ(15,100,ERR=110) (U(I),V(I),S(I),T(I),I=1,NFUNS)
  100 FORMAT(5X,4F8.4)
      GO TO 120
  110 READ(0,115) RDERR
  115 FORMAT(A80)
      PRINT*, 'ERR IN READING ADATS FILE', '  RECORD= ', RDERR
      STOP
  120 PRINT*, 'AT END - ADATS FILE'
      RETURN
      END
```

Figure E-II-19. Source Listing of Subprogram GETFAC of the AFP CS/CSS Module

E-II-16. Figure E-II-20 presents the source listing of subprogram GTCATV of the CS/CSS Module. GTCATV is called once from the main program to read and store data specifying for Blue and Red measures and countermeasures whether further CS/CSS logic is to be applied for specific combinations of CS/CSS functions, Red weapon categories, and Blue weapon categories. The formal arguments of GTCATV correspond to actual arguments already defined in the main program as arrays and their dimensions.

E-II-17. Figure E-II-21 presents the source listing of subprogram OUTW of the CS/CSS Module. OUTW is called by subprogram CSCSS every time a record of AFP standard type 200 is to be output. A single such record contains a number of case identifiers, but the principal output data are the CS/CSS moduli for Blue weapon type IB and Red weapon type IR in their paired engagement.

a. Argument BF is the Blue CS/CSS modulus.

b. Argument RF is the Red CS/CSS modulus.

c. Argument IB is the index of the Blue weapon type.

d. Argument IR is the index of the Red weapon type.

```
      SUBROUTINE GTCATV(UCAT,VCAT,SCAT,TCAT,NFUNS,NC)
C
      CHARACTER*1 UCAT(NFUNS,NC,NC),VCAT(NFUNS,NC,NC),
     *SCAT(NFUNS,NC,NC),TCAT(NFUNS,NC,NC)
C
      CHARACTER*80 RDERR
C
C
      DO 60 K=1,12
      DO 50 I=1,9
      READ(16,100,ERR=110) (UCAT(I,J,K),VCAT(I,J,K),
     *SCAT(I,J,K),TCAT(I,J,K),J=1,12)
   50 CONTINUE
   60 CONTINUE
  100 FORMAT(4X,12(1X,4A1))
      GO TO 120
  110 READ(0,115)RDERR
  115 FORMAT(A80)
      PRINT*, 'ERR IN READING ACATS FILE', '  RECORD= ', RDERR
      STOP
  120 PRINT*, 'AT END - ACATS FILE'
      RETURN
      END
```

Figure E-II-20. Source Listing of Subprogram GTCATV of the AFP CS/CSS Module

```
      SUBROUTINE OUTW(BF,FF,IB,IR)
C
      COMMON/AWRK/IWRK,ISCNT,ITHTR,ITPD,IVIS,IPOS,IDAY
C
      WRITE(IWRK,100) ISCNT,IB,BF,IR,FF
  100 FORMAT(1X,I5,'   E   1   1   1   1',I5,F10.6,I5,F10.6)
      RETURN
      END
```

Figure E-II-21. Source Listing of Subprogram OUTW of the AFP CS/CSS Module

E-II-18. Figure E-II-22 presents a listing of the MAP element for collection of the program elements of the CS/CSS Module.

```
@MAP    ,G6GECTEST.899CSCSS/MAIN
IN   G6GECTEST.899CSCSS/MAIN
IN   G6GECTEST.CSCSS
IN   G6GECTEST.GETCAT
IN   G6GECTEST.GETDAT
IN   G6GECTEST.GETFAC
IN   G6GECTEST.GTCATV
IN   G6GECTEST.OUTW
END
```

Figure E-II-22. Listing of the MAP Element for the Collection of Program Elements of the AFP CS/CSS Module

APPENDIX F

THE AFP CBT/CS/CSS MERGE MODULE

F-1. OVERVIEW

a. The AFP Combat/Combat Service/Combat Service Support (CBT/CS/CSS) Merge Module is designed to:

(1) Accept some input to and results from a single execution of the AFP Combat Module giving, among other items, the allocations of weapons to type-on-type engagements and the losses in engagements for one combat environment.

(2) Accept CS/CSS moduli as a file from the AFP CS/CSS Module. That file contains CS/CSS moduli for each side for each possible pairing of 60 Blue weapon types against 60 Red weapon types. That is, the file contains 2 x 60 x 60 x = 7,200 CS/CSS moduli.

(3) Accept a table of input target values for use in converting estimates of target kills into partial scalar combat potentials. On user option, the target values may be applied to all elements of partial combat potentials.

(4) Accept a table of input fractional life factors for projecting partial combat potentials to the fractions of lifetimes corresponding to shooting weapon types.

(5) Calculate partial (i.e., for only one combat environment) combat potentials for each weapon type and both Red and Blue divisions by the standard and one alternative method.

(6) Output a file (for the standard method) containing weapon and division partial combat potentials in standard AFP combat potential format.

b. The relation of the AFP CBT/CS/CSS Merge Module to the AFP System in general is portrayed in Figure F-1. There the module is highlighted by being enclosed in an oval.

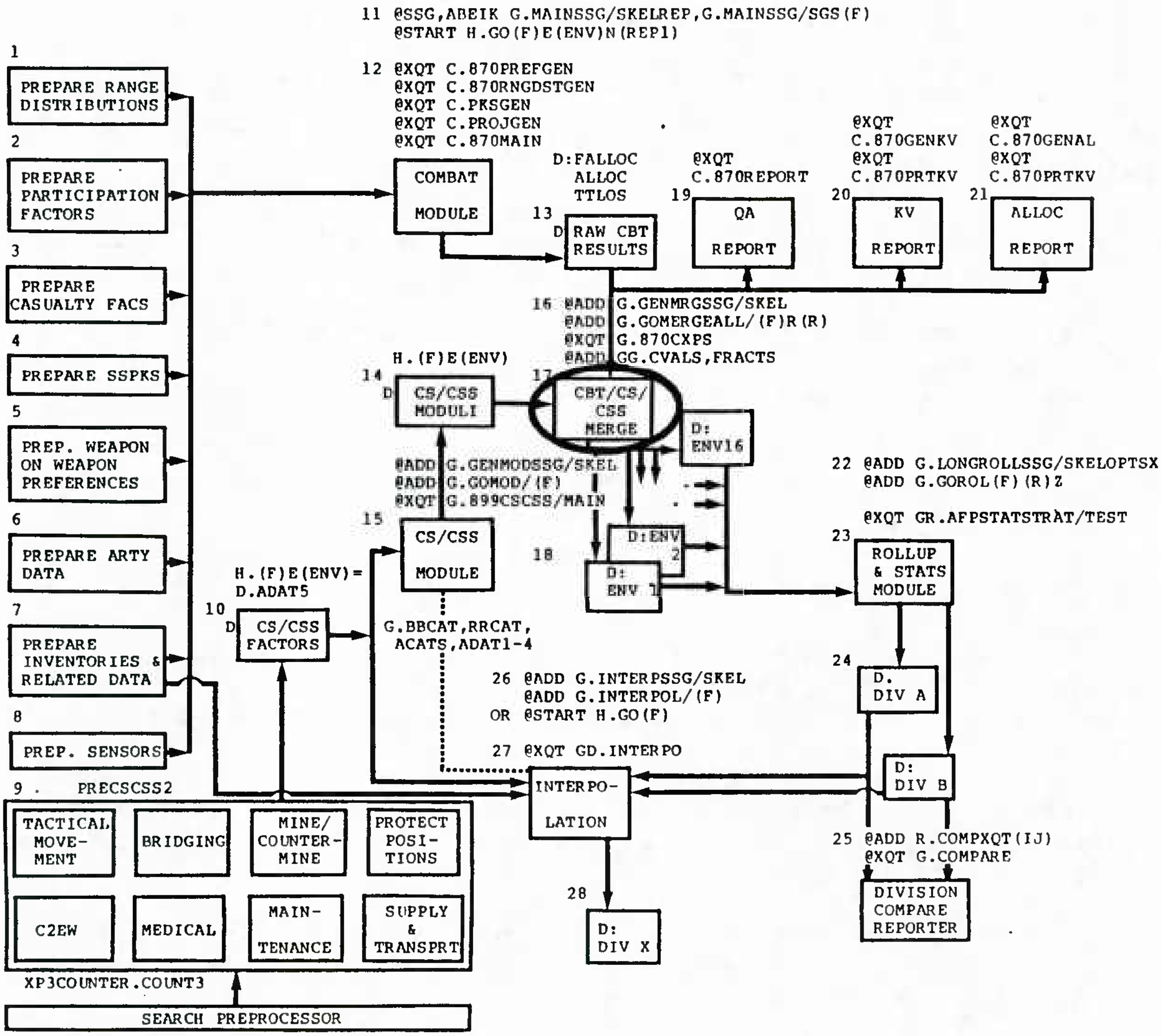


Figure F-1. Relation of the AFP CBT/CS/CSS Merge Module to the AFP System in General

F-2. INPUT

a. The primary input to the AFP CBT/CS/CSS Merge Module are the files of Combat Module output and some input and the file of CS/CSS moduli from the CS/CSS Module.

(1) The files from the Combat Module are formatted direct access. The contents of these files cannot be inspected or listed by the common UNIVAC system utilities. The CBT/CS/CSS Merge Module contains special interface routines for accessing Combat Module files.

(2) An extract from the CS/CSS Module-produced file of CS/CSS moduli for input to the CBT/CS/CSS Merge Module is shown in Figure F-2. With the AFP System configured for 60 Blue and 60 Red weapon types, a full CS/CSS moduli file contains 3,600 records. The 3,600 records correspond to all pairings of 60 Blue weapon types with 60 Red weapon types. The first 60 records correspond to Blue weapon type #1 versus each of the 60 Red weapon types in order. The second 60 records correspond to Blue weapon type #2 versus each of the 60 Red weapon types in order. Thus, the file may be considered to consist of 60 sets of 60 records. Each record contains a Blue CS/CSS modulus and a Red modulus. The fields of the CS/CSS moduli file are described in some detail in the output section of Appendix E. Very brief descriptions of the fields of records illustrated in Figure F-2 are provided in the following paragraphs.

(a) Field 1 contains the record identifier "200" for all records within a file of the CS/CSS Module.

(b) The subfields of Field 2 may contain special identifiers for such attributes as theater, case, and combat environment. Some such identifiers are included in Figure F-2. However, in current AFP practice, identification of complete files is usually done by application of file naming conventions.

(c) Field 3 contains a numerical identifier of the Blue weapon type corresponding to the record.

(d) Field 4 contains the Blue CS/CSS modulus for the pairing of the Blue weapon type identified in Field 3 with the Red weapon type identified in Field 5.

(e) Field 5 contains a numerical identifier of the Red weapon type corresponding to the record.

(f) Field 6 contains the Red CS/CSS modulus for the pairing of the Red weapon type identified in Field 5 with the Blue weapon type identified in Field 3.

FIELDS

1	2					3	4	5	6
200	E	1	1	1	1	20	.966148	40	1.220659
200	E	1	1	1	1	20	.966148	41	1.220659
200	E	1	1	1	1	20	.966148	42	1.220659
200	E	1	1	1	1	20	.966148	43	1.220659
200	E	1	1	1	1	20	.966148	44	1.220659
200	E	1	1	1	1	20	.966148	45	1.220659
200	E	1	1	1	1	20	.987073	46	1.194461
200	E	1	1	1	1	20	.965332	47	1.220696
200	E	1	1	1	1	20	.965332	48	1.220696
200	E	1	1	1	1	20	.987073	49	1.194461
200	E	1	1	1	1	20	.987073	50	1.194461
200	E	1	1	1	1	20	.987073	51	1.194461
200	E	1	1	1	1	20	.987073	52	1.194461
200	E	1	1	1	1	20	.987073	53	1.194461
200	E	1	1	1	1	20	.987073	54	1.194461
200	E	1	1	1	1	20	.987073	55	1.194461
200	E	1	1	1	1	20	.987073	56	1.194461
200	E	1	1	1	1	20	.987073	57	1.194461
200	E	1	1	1	1	20	.987073	58	1.194461
200	E	1	1	1	1	20	.987073	59	1.194461
200	E	1	1	1	1	20	.987073	60	1.194461
200	E	1	1	1	1	21	.987073	1	1.194461
200	E	1	1	1	1	21	.987073	2	1.194461
200	E	1	1	1	1	21	.987073	3	1.194461
200	E	1	1	1	1	21	.987073	4	1.194461
200	E	1	1	1	1	21	.987073	5	1.194461
200	E	1	1	1	1	21	.987073	6	1.194461
200	E	1	1	1	1	21	.987073	7	1.194461
200	E	1	1	1	1	21	.987073	8	1.194461
200	E	1	1	1	1	21	.987073	9	1.194461
200	E	1	1	1	1	21	.987073	10	1.194461
200	E	1	1	1	1	21	.987073	11	1.194461
200	E	1	1	1	1	21	.987073	12	1.194461
200	E	1	1	1	1	21	.987073	13	1.194461
200	E	1	1	1	1	21	.987073	14	1.194461
200	E	1	1	1	1	21	.987073	15	1.194461
200	E	1	1	1	1	21	.987073	16	1.194461
200	E	1	1	1	1	21	.987073	17	1.194461
200	E	1	1	1	1	21	.987073	18	1.194461
200	E	1	1	1	1	21	.987073	19	1.194461
200	E	1	1	1	1	21	.987073	20	1.194461
200	E	1	1	1	1	21	.987073	21	1.194461
200	E	1	1	1	1	21	.987073	22	1.194461
200	E	1	1	1	1	21	.987073	23	1.194461
200	E	1	1	1	1	21	.987073	24	1.194461
200	E	1	1	1	1	21	.987073	25	1.194461
200	E	1	1	1	1	21	.987073	26	1.194461
200	E	1	1	1	1	21	.987073	27	1.194461
200	E	1	1	1	1	21	.987073	28	1.194461
200	E	1	1	1	1	21	.987073	29	1.194461
200	E	1	1	1	1	21	.987073	30	1.194461

Figure F-2. Example Extract Records from the CS/CSS Moduli File Output by the AFP CS/CSS Module for Input to the CBT/CS/CSS Module

b. Secondary, but still essential, input to the CBT/CS/CSS Merge Module consists of target values and fractional life factors.

(1) Figure F-3 displays a set of input target values. The table has been set up in unusual fashion in order to accommodate a relatively late generalization of the original AFP notion of target value. Formerly, all light armored vehicles killed were considered to generate the same contribution to the partial scalar combat potential. All heavy armor kills generated the same contribution to combat potential (the standard heavy armor target value exceeded the standard light armor target value). Aircraft kills, whether rotary or fixed wing, generated equal additions to combat potentials although the standard aircraft target value exceeded the light and heavy armor values. Crew lost with a vehicle or aircraft added slightly to the net target value. Most nonvehicularly-mounted weapons were considered to have target value only to the extent of their crew losses. Thus, a ground-mounted machinegun or a shoulder-fired missile possessed target value only to the extent that its crew was killed. That former approach was judged to provide too little discrimination--in theory and in practice. Just such a simplified treatment of target values is still permitted within the current system. However, the current system permits each of the 120 (60 Blue and 60 Red) weapon types to have a unique target value. Furthermore, crew members of different weapon types may have different target values. Weapon-unique target values are probably as much as theory requires. Practice requires correct target values but necessarily relies heavily on judgment. Recall that four-valued combat potentials consist of personnel, light armor, heavy armor, and aircraft components. At first thought, it may seem appropriate to implement the generalized target value scheme in the form of a four-component target value vector. In fact, the generalization could have been introduced by means of a two-component target value vector. The approach selected combines the alternatives, yielding a five-component target value vector for each target. The fields in records illustrated in Figure F-3 have the following significance:

(a) Field 1. The side: 1 = Blue, 2 = Red.

(b) Field 2. The weapon type number: 1 to 60 for each of Blue and Red.

(c) Field 3. The target value of each crew member lost.

(d) Field 4. The target value as a light armored vehicle.

(e) Field 5. The target value as a heavy armored vehicle.

(f) Field 6. The target value as an aircraft.

(g) Field 7. The target value as other than a weapon corresponding to Fields 4-6.

	FIELDS 1	2	3	4	5	6	7
	-	-	---	---	---	---	---
1	1	1	0.1	0.0	0.0	0.0	0.0
2	1	2	0.1	0.0	0.0	0.0	0.0
3	1	3	0.1	0.0	0.0	0.0	0.2
4	1	4	0.1	0.0	0.0	0.0	0.2
5	1	5	0.1	0.0	0.0	0.0	0.2
6	1	6	0.1	1.0	0.0	0.0	0.0
7	1	7	0.1	1.0	0.0	0.0	0.0
8	1	8	0.1	0.0	0.0	0.0	0.2
9	1	9	0.0	0.0	0.0	0.0	0.0
10	1	10	0.1	2.0	0.0	0.0	0.0
11	1	11	0.1	0.0	0.0	0.0	1.5
12	1	12	0.1	2.0	0.0	0.0	0.0
13	1	13	0.1	0.0	0.0	0.0	0.2
14	1	14	0.1	0.0	0.0	0.0	0.2
15	1	15	0.1	2.0	0.0	0.0	0.0
16	1	16	0.1	2.0	0.0	0.0	0.0
17	1	17	0.1	2.0	0.0	0.0	0.0
18	1	18	0.1	2.0	0.0	0.0	0.0
19	1	19	0.1	0.0	4.0	0.0	0.0
20	1	20	0.1	0.0	4.0	0.0	0.0
21	1	21	0.1	0.0	4.0	0.0	0.0
22	1	22	0.1	0.0	4.0	0.0	0.0
23	1	23	0.1	0.0	4.0	0.0	0.0
24	1	24	0.1	0.0	4.0	0.0	0.0
25	1	25	0.1	0.0	4.0	0.0	0.0
26	1	26	0.1	16.0	0.0	0.0	0.0
27	1	27	0.1	2.0	0.0	0.0	0.0
28	1	28	0.1	2.0	0.0	0.0	0.0
29	1	29	0.1	2.0	0.0	0.0	0.0
30	1	30	0.1	2.0	0.0	0.0	0.0
31	1	31	0.1	0.0	0.0	0.0	1.0
32	1	32	0.1	8.0	0.0	0.0	0.0
33	1	33	0.1	0.0	0.0	0.0	1.0
34	1	34	0.0	0.0	0.0	0.0	0.0
35	1	35	0.0	0.0	0.0	0.0	0.0
36	1	36	0.1	0.0	0.0	10.0	0.0
37	1	37	0.1	0.0	0.0	10.0	0.0
38	1	38	0.1	0.0	0.0	0.0	0.0
39	1	39	0.1	0.0	0.0	0.0	0.0
40	1	40	0.0	0.0	0.0	0.0	0.0
41	1	41	0.1	0.0	0.0	6.0	0.0
42	1	42	0.1	1.0	0.0	0.0	0.0
43	1	43	0.1	0.0	4.0	0.0	0.0
44	1	44	0.1	1.0	0.0	0.0	0.0
45	1	45	0.1	0.0	0.0	5.0	0.0
46	1	46	0.1	0.0	3.0	0.0	0.0
47	1	47	0.1	0.0	0.0	5.0	0.0
48	1	48	0.1	0.0	0.0	5.0	0.0
49	1	49	0.1	1.0	0.0	0.0	0.0
50	1	50	0.0	0.0	0.0	0.0	0.0
51	1	51	0.1	0.0	0.0	0.0	1.0
52	1	52	0.1	0.0	0.0	0.0	1.0
53	1	53	0.1	1.5	0.0	0.0	0.0
54	1	54	0.0	0.0	0.0	0.0	0.0
55	1	55	0.0	0.0	0.0	0.0	0.0
56	1	56	0.1	2.5	0.0	0.0	0.0
57	1	57	0.1	2.5	0.0	0.0	0.0
58	1	58	0.1	2.5	0.0	0.0	0.0
59	1	59	0.0	0.0	0.0	0.0	0.0
60	1	60	0.0	0.0	0.0	0.0	0.0

Figure F-3. Example of Data Input Element Providing Target Category Value to the CBT/CS/CSS Merge Module (Page 1 of 2 pages)

61	2	1	0.1	0.0	0.0	0.0	0.0
62	2	2	0.1	0.0	0.0	0.0	0.0
63	2	3	0.1	0.0	0.0	0.0	0.2
64	2	4	0.1	0.0	0.0	0.0	0.0
65	2	5	0.1	0.0	0.0	0.0	0.2
66	2	6	0.1	1.0	0.0	0.0	0.0
67	2	7	0.1	1.0	0.0	0.0	0.0
68	2	8	0.1	1.0	0.0	0.0	0.0
69	2	9	0.0	0.0	0.0	0.0	0.0
70	2	10	0.0	0.0	0.0	0.0	0.0
71	2	11	0.1	0.0	0.0	0.0	0.4
72	2	12	0.1	0.0	0.0	0.0	0.4
73	2	13	0.1	2.0	0.0	0.0	0.0
74	2	14	0.1	0.0	0.0	0.0	0.4
75	2	15	0.1	0.0	0.0	0.0	0.4
76	2	16	0.1	2.0	0.0	0.0	0.0
77	2	17	0.0	0.0	0.0	0.0	0.0
78	2	18	0.0	0.0	0.0	0.0	0.0
79	2	19	0.0	0.0	0.0	0.0	0.0
80	2	20	0.0	0.0	0.0	0.0	0.0
81	2	21	0.1	0.0	4.0	0.0	0.0
82	2	22	0.1	0.0	4.0	0.0	0.0
83	2	23	0.1	0.0	4.0	0.0	0.0
84	2	24	0.1	0.0	4.0	0.0	0.0
85	2	25	0.1	0.0	4.0	0.0	0.0
86	2	26	0.1	16.0	0.0	0.0	0.0
87	2	27	0.0	0.0	0.0	0.0	0.0
88	2	28	0.0	0.0	0.0	0.0	0.0
89	2	29	0.0	0.0	0.0	0.0	0.0
90	2	30	0.1	0.0	0.0	0.0	1.0
91	2	31	0.1	16.0	0.0	0.0	0.0
92	2	32	0.1	0.0	0.0	0.0	1.0
93	2	33	0.1	8.0	0.0	0.0	0.0
94	2	34	0.1	8.0	0.0	0.0	0.0
95	2	35	0.1	16.0	0.0	0.0	0.0
96	2	36	0.1	0.0	0.0	10.0	0.0
97	2	37	0.1	0.0	0.0	10.0	0.0
98	2	38	0.1	0.0	0.0	10.0	0.0
99	2	39	0.0	0.0	0.0	0.0	0.0
100	2	40	0.0	0.0	0.0	0.0	0.0
101	2	41	0.1	0.0	0.0	6.0	0.0
102	2	42	0.1	1.0	0.0	0.0	0.0
103	2	43	0.1	0.0	4.0	0.0	0.0
104	2	44	0.1	1.0	0.0	0.0	0.0
105	2	45	0.0	0.0	0.0	0.0	0.0
106	2	46	0.1	1.0	0.0	0.0	0.0
107	2	47	0.0	0.0	0.0	0.0	0.0
108	2	48	0.0	0.0	0.0	0.0	0.0
109	2	49	0.0	0.0	0.0	0.0	0.0
110	2	50	0.0	0.0	0.0	0.0	0.0
111	2	51	0.1	0.0	0.0	0.0	1.0
112	2	52	0.1	1.5	0.0	0.0	0.0
113	2	53	0.0	0.0	0.0	0.0	0.0
114	2	54	0.0	0.0	0.0	0.0	0.0
115	2	55	0.0	0.0	0.0	0.0	0.0
116	2	56	0.1	2.5	0.0	0.0	0.0
117	2	57	0.1	2.5	0.0	0.0	0.0
118	2	58	0.1	2.5	0.0	0.0	0.0
119	2	59	0.0	0.0	0.0	0.0	0.0
120	2	60	0.0	0.0	0.0	0.0	0.0

Figure F-3. Example of Data Input Element Providing Target Category Value to the CBT/CS/CSS Merge Module (Page 2 of 2 pages)

Fields 3 and 7 would be sufficient. Any value in Fields 4-6 could have been entered in Field 7 with identical contribution to partial and final scalar combat potentials. Notice that a weapon's value as a target is not an AFP result; target values are pure inputs. A weapon's partial or final scalar combat potential is an AFP result dependent on the numbers of targets killed (an AFP result) and the value of those targets killed (not an AFP result). Target values may be applied, at user option, in one of two ways. If program input variable TVALON=.FALSE., target values are incorporated on the fifth elements of partial combat potential, the so-called scalar elements. The first four elements are unweighted estimates of personnel, light armored vehicles, heavy armored vehicles, and aircraft, respectively. If TVALON=.TRUE., the first four elements of partial combat potentials are estimates weighted by target values. In addition, the former "personnel only" elements become "personnel plus other weapon" elements, weighted by their corresponding target values. "Other" weapons typically include small arms and shoulder SAMs. "Other" weapons have nonzero entries for the fifth component (Field 7) of target value. However applied to date, target values have been AFP input largely independent of AFP results. As AFP development progressed, some target values were modified to reflect some AFP results. Discussions among analysts inside and outside CAA revealed differing philosophies and preferences. Some analysts want the "AFP loop" closed so that target values become a pure result of the AFP process. Currently, the AFP System does not operate quickly enough to imbed the modules in an iterative or recursive scheme to converge on target values equal to combat potentials (CIPs). Other analysts favor target values depending on capital investment instead of combat estimates. Then, too, there are "middle of the roaders" who prefer some blend of the extremes. At a somewhat deeper level, a case can be made for making the weapon preferences input to the AFP Combat Module depend directly on target values. As of this writing, "how best" and "who best" to determine target values remains a subject of debate.

(2) Figure F-4 displays an example of fractional lifetime factors. The first version of the AFP System expressed all combat potentials relative to a 25 percent depletion of shooter weapons. Combat potential then expressed an estimate of targets killed for 25 percent loss to shooters. However, the vulnerability and pace of combat differs so much among weapons that the original approach was considered to provide too little useful discrimination. Two systems might have exactly the same exchange ratios, implying that each would achieve the same number of kills by the time it had lost an equal fraction of its own strength. The first system might reach its "standard" result in 30 minutes; the second system might reach the "same" result only after 30 days. Few commanders or analysts would want to regard the two systems as equal. The underlying problem recurs in almost all attempts to map dynamics into static measures where both results achieved and resources expended may vary. If static measure is taken to involve just results at a fixed time, then resource expenditure is left totally uncertain. On the other hand, if static measure is taken to involve just the results achieved for a fixed resource expenditure, then time to achieve is left totally uncertain. Practical military issues

usually involve consideration of achievement, cost to achieve, and time to achieve. All three of these considerations may vary among different weapon types and circumstances. Any compression of these related but separate considerations into a single measure involves some loss of information and implies acceptance of some, perhaps much, uncertainty. The AFP developers decided to apply a hybrid approach as being less misleading than either of the constant-time and constant-resource extremes. The hybrid approach introduces three categories of relevant lifetime. The first lifetime category includes the usual direct fire weapons: their potential is assessed at their half-life, estimated kills achieved for 50 percent loss of their own strength. The second lifetime category includes the usual indirect fire weapons: their potential is assessed at the 2-week point in a projected campaign, estimated kills achieved for a 2-weeks' expenditure of ammunition at accepted planning rates. The third lifetime category includes self-consuming weapons (typically many kinds of platformless rockets or any rounds in short supply relative to campaign length and planned rate of fire): their potential is assessed at the point of consumption of basic load or estimated theater stockage level. The fields in records illustrated in Figure F-4 have the following significance:

(a) Field 1. The weapon type number: 1 to 60 for both Blue and Red weapons.

(b) Field 2. The adjustment to Blue weapon potentials relative to an assumed standard half-life. Hence, a factor of 1.0 simply maintains half-life as the point of reference. A factor of 0.33 corresponds to a net of one-sixth life; relative to an assumed 2-week half-life, the factor 0.33 corresponds to roughly 2+ days as fire weapons, whose achievements but not losses, are recorded. (As a useful arithmetic simplification and artifice, most indirect fire weapons populations are regarded as suffering exactly one loss during the AFP Combat Module estimation process.) The indirect fire factor then is an estimate of how may times as many rounds the indirect fire weapon would expend if the Combat Module estimated a 2-week rather than 2-day campaign.

(c) Field 3. The factors in Field 3 apply to Red weapons in exactly the same way as those in Field 2 apply to Blue weapons.

F-3. OUTPUT

a. The normal output of the CBT/CS/CSS Merge Module takes two forms. The first report is a recapitulation (with some extensions) of some of the results from the Combat Module. The second report (also saved as a mass storage file) presents unmodulated and modulated partial (in the sense of being from a single combat environment) five-valued scores and CIPs for each Blue and Red weapon with nonzero combat potential. The report also presents the unmodulated and modulated CIPs for the correspondings Blue and Red divisions. Each of the two types of reports is illustrated and described in the following paragraphs.

1	1	1.0	1.0
2	2	1.0	1.0
3	3	1.0	1.0
4	4	1.0	1.0
5	5	1.0	1.0
6	6	1.0	1.0
7	7	1.0	1.0
8	8	1.0	1.0
9	9	1.0	1.0
10	10	1.0	1.0
11	11	1.0	1.0
12	12	1.0	1.0
13	13	1.0	1.0
14	14	1.0	1.0
15	15	1.0	1.0
16	16	1.0	1.0
17	17	1.0	1.0
18	18	1.0	1.0
19	19	1.0	1.0
20	20	1.0	1.0
21	21	1.0	1.0
22	22	1.0	1.0
23	23	1.0	1.0
24	24	1.0	1.0
25	25	1.0	1.0
26	26	1.0	1.0
27	27	1.0	1.0
28	28	1.0	1.0
29	29	1.0	1.0
30	30	1.0	0.16
31	31	0.16	0.16
32	32	0.16	0.16
33	33	0.16	0.16
34	34	0.16	0.16
35	35	0.16	0.16
36	36	1.0	1.0
37	37	1.0	1.0
38	38	1.0	1.0
39	39	1.0	1.0
40	40	1.0	1.0
41	41	1.0	1.0
42	42	1.0	1.0
43	43	1.0	1.0
44	44	1.0	1.0
45	45	1.0	1.0
46	46	1.0	1.0
47	47	1.0	1.0
48	48	1.0	1.0
49	49	1.0	1.0
50	50	1.0	1.0
51	51	5.56	5.56
52	52	27.0	27.0
53	53	27.0	27.0
54	54	27.0	27.0
55	55	27.0	27.0
56	56	30.0	30.0
57	57	30.0	30.0
58	58	30.0	30.0
59	59	1.0	1.0
60	60	1.0	1.0

Figure F-4. Example of Data Input Element Providing Fractional Lifetime Factors to the CBT/CS/CSS Merge Module

b. Figure F-5 displays sample records extracted from the CBT/CS/CSS Merge Module's recapitulation of some intermediate results a step or two beyond raw output of the Combat Module. The intermediate results are several steps removed from being partial combat potentials. The intermediate results are not expressed in terms of exchange ratios, nor have the raw results been projected to the fixed-fractional lifetimes or the fixed-time points appropriate for each weapon. The report contains only one record for a shooting weapon. In general, AFP reports of partial or final combat potential contain four records per shooting weapon type, separate records for: unmodulated score, unmodulated CIP, modulated score, and modulated CIP. The fields in records of the type portrayed in Figure F-5 have the following significance.

(1) Field 1 identifies the side to which the record applies: Blue = 1, Red = 2.

1	2		3	4	5	6	7		8
	SHOOTER		PERS	LVEH	LVEH	ACFT	SCALAR		
SIDE= 1	IDWPN= 1	KILLS=	.000	.000	.000	.000	.000	LOSSES=	399.000
SIDE= 1	IDWPN= 2	KILLS=	387.000	.000	.000	.000	30.900	LOSSES=	87.000
SIDE= 1	IDWPN= 3	KILLS=	224.000	.000	.000	.000	24.200	LOSSES=	35.000
SIDE= 1	IDWPN= 5	KILLS=	364.000	8.000	.000	.000	47.000	LOSSES=	54.000
SIDE= 1	IDWPN= 6	KILLS=	256.000	.000	.000	.000	27.800	LOSSES=	50.000
SIDE= 1	IDWPN= 11	KILLS=	91.000	23.000	43.000	.000	225.100	LOSSES=	85.000
SIDE= 1	IDWPN= 12	KILLS=	44.000	8.000	15.000	.000	93.400	LOSSES=	2.000
SIDE= 1	IDWPN= 16	KILLS=	599.000	138.000	41.000	.000	433.100	LOSSES=	27.000
SIDE= 1	IDWPN= 17	KILLS=	237.000	77.000	23.000	.000	233.300	LOSSES=	5.000
SIDE= 1	IDWPN= 23	KILLS=	645.000	64.000	139.000	7.000	994.700	LOSSES=	56.000
SIDE= 1	IDWPN= 26	KILLS=	9.000	7.000	.000	3.000	37.900	LOSSES=	1.000
SIDE= 1	IDWPN= 31	KILLS=	5.000	.000	.000	5.000	50.500	LOSSES=	29.000
SIDE= 1	IDWPN= 32	KILLS=	1.000	.000	.000	1.000	10.100	LOSSES=	.000
SIDE= 1	IDWPN= 36	KILLS=	197.000	36.000	9.000	.000	491.400	LOSSES=	53.000
SIDE= 1	IDWPN= 41	KILLS=	.000	.000	.000	.000	.000	LOSSES=	56.000
SIDE= 1	IDWPN= 43	KILLS=	.000	.000	.000	.000	.000	LOSSES=	.000
SIDE= 1	IDWPN= 44	KILLS=	.000	.000	.000	.000	.000	LOSSES=	20.000
SIDE= 1	IDWPN= 51	KILLS=	.000	.000	.000	.000	.000	LOSSES=	.000
SIDE= 1	IDWPN= 53	KILLS=	1520.000	17.000	.000	.000	165.000	LOSSES=	.000
SIDE= 1	IDWPN= 56	KILLS=	2.000	.000	.000	2.000	.000	LOSSES=	3.000
SIDE= 1	IDWPN= 57	KILLS=	2884.000	21.000	.000	.000	333.400	LOSSES=	2.000
SIDE= 2	IDWPN= 1	KILLS=	.000	.000	.000	.000	.000	LOSSES=	397.000
SIDE= 2	IDWPN= 2	KILLS=	319.000	.000	.000	.000	57.700	LOSSES=	156.000
SIDE= 2	IDWPN= 3	KILLS=	355.000	.000	.000	.000	45.800	LOSSES=	32.000
SIDE= 2	IDWPN= 4	KILLS=	165.000	.000	.000	.000	32.200	LOSSES=	57.000
SIDE= 2	IDWPN= 5	KILLS=	61.000	.000	.000	.000	14.400	LOSSES=	24.000
SIDE= 2	IDWPN= 6	KILLS=	63.000	2.000	.000	.000	10.400	LOSSES=	26.000
SIDE= 2	IDWPN= 7	KILLS=	378.000	5.000	.000	.000	51.000	LOSSES=	122.000

Figure F-5. Example Extract from the CBT/CS/CSS Merge Module Report of Raw Results (with some extensions) from the Combat Module

(2) Field 2 provides the number identifying the weapon type: 1 to 60. Fields 1 and 2 together provide unique reference to a weapon type. Field 2 alone is not sufficient because both sides may have weapon identification numbers from 1 through 60.

(3) Field 3 provides a raw count of personnel losses/casualties inflicted by weapons of the type identified in Fields 1 and 2 during the course of the preceding Combat Module run. The personnel count is not a raw result of the Combat Module. The values reported in records of the type shown in Figure F-5 are generated by the CBT/CS/CSS Merge Module as the result of having accumulated the results of references to a loss/casualty table for each "raw" weapon/platform loss output by the Combat Module. The contents of Field 3 are thus the sum of crew losses estimated to occur as the result of weapon losses. If input program variable TVALON=.TRUE., the count includes other weapons, and personnel and other weapons are weighted by target values.

(4) Field 4 provides a raw count of light armored vehicle losses inflicted by the weapons of the type identified in Fields 1 and 2 during the course of the preceding Combat Module run. In general, the count includes losses of several different weapon types, all those identified as light armored vehicles within a special cross-reference table. If input program variable TVALON=.TRUE., the count is weighted by target values.

(5) Field 5 provides a raw count of heavy armored vehicle losses inflicted by the weapons of the type identified in Fields 1 and 2 during the course of the preceding Combat Module run. In general, the count includes losses of several different weapon types, all those identified as heavy armored vehicles, within a special cross-reference table. If input program variable TVALON=.TRUE., the count is weighted by target values.

(6) Field 6 provides a raw count of aircraft losses inflicted by the weapons of the type identified in Fields 1 and 2 during the course of the preceding Combat Module run. In general, the count includes losses of several different aircraft types, all those weapons identified as aircraft, within a special cross-reference table. If input program variable TVALON=.TRUE., the count is weighted by target values.

(7) Field 7 provides a raw scalar value achieved by the weapons of the type identified in Fields 1 and 2 during the course of the preceding Combat Module run. The values shown in Field 7 are not raw results of the Combat Module. Rather, the values are the results of accumulation (by CBT/CS/CSS Merge Module) of target values extracted from the table described in paragraph 4 above for each target loss generated in the Combat Module.

(8) Field 8 provides a raw count of the losses suffered during the preceding Combat Module run by the weapons of the type identified in Fields 1 and 2 while achieving the results shown in Fields 3 through 7. Obviously, the information in Field 8 does not discriminate how the losses occurred by target type engaged, although just such paired data are provided by the Combat Module and used (for other purposes) within the CBT/CS/CSS Merge Module.

c. Figure F-6 displays sample records extracted from a report (or file) of partial combat potential generated by the AFP CBT/CS/CSS Merge Module. Only a subset of the records is shown; the total number of records within a report depends on the number of different types of weapons analyzed within the Combat Module and whether the weapons were successful in inflicting losses. A weapon type that scores no kills is not reported. It is not unusual for a weapon type to appear in the report for some combat environment but to be missing from the report for another combat environment, e.g., a weapon may score well in daylight, but, if it cannot "see" in the dark, it cannot kill anything in nighttime combat environments and will be scoreless in those environments. The significance of fields in records of the types displayed in Figure F-6 is described in the following paragraphs. In general, there may be four records for each weapon type, one for each of unmodulated score, unmodulated CIP, modulated score, and modulated CIP. There should be two records for each side's division, one each for unmodulated COP and modulated COP.

(1) Field 1 contains record identifiers corresponding to the keys tabulated in Figure A-3 of Appendix A. Note that, because the report (or file) contains only partial (one-combat environment) potentials, all the record identifiers are less than 100. A report of file of final potentials (producible from another AFP module) has the same format as portrayed in Figure F-6, but all its record identifiers must exceed 100.

	FIELD 1	2	3	4	5	6	7	8	9
17.	10	E 1 1 1 1	16	1191.458	84.925	113.229	.000	26.959	1 1 1
18.	30	E 1 1 1 1	16	5.784	.412	.550	.000	.131	1 1 1
19.	50	E 1 1 1 1	16	1197.489	85.355	113.802	.000	27.096	1 1 1
20.	70	E 1 1 1 1	16	5.813	.414	.552	.000	.132	1 1 1
21.	10	E 1 1 1 1	17	644.042	47.542	48.500	.000	13.075	1 1 1
22.	30	E 1 1 1 1	17	5.776	.426	.435	.000	.117	1 1 1
23.	50	E 1 1 1 1	17	647.302	47.782	48.746	.000	13.141	1 1 1
24.	70	E 1 1 1 1	17	5.805	.429	.437	.000	.118	1 1 1
25.	10	E 1 1 1 1	20	3252.010	234.271	345.906	5.000	83.696	1 1 1
26.	30	E 1 1 1 1	20	9.855	.710	1.048	.015	.254	1 1 1
27.	50	E 1 1 1 1	20	3272.702	235.457	347.657	5.448	84.554	1 1 1
28.	70	E 1 1 1 1	20	9.917	.714	1.054	.017	.256	1 1 1
29.	10	E 1 1 1 1	26	118.500	6.625	.000	8.000	9.006	1 1 1
30.	30	E 1 1 1 1	26	3.703	.207	.000	.250	.281	1 1 1
31.	50	E 1 1 1 1	26	120.453	6.659	.000	8.717	9.732	1 1 1
32.	70	E 1 1 1 1	26	3.764	.208	.000	.272	.304	1 1 1
					* * *				
157.	20	E 1 1 1 1	51	1.706	.000	.000	.000	.004	1 1 1
158.	40	E 1 1 1 1	51	.059	.000	.000	.000	.000	1 1 1
159.	60	E 1 1 1 1	51	1.780	.000	.000	.000	.005	1 1 1
160.	80	E 1 1 1 1	51	.061	.000	.000	.000	.000	1 1 1
161.	20	E 1 1 1 1	52	51.161	6.395	.000	.000	.810	1 1 1
162.	40	E 1 1 1 1	52	.839	.105	.000	.000	.013	1 1 1
163.	60	E 1 1 1 1	52	53.377	6.672	.000	.000	.845	1 1 1
164.	80	E 1 1 1 1	52	.875	.109	.000	.000	.014	1 1 1
165.	20	E 1 1 1 1	56	21.396	1.297	.000	.000	.192	1 1 1
166.	40	E 1 1 1 1	56	.181	.011	.000	.000	.002	1 1 1
167.	60	E 1 1 1 1	56	22.322	1.353	.000	.000	.201	1 1 1
168.	80	E 1 1 1 1	56	.189	.011	.000	.000	.002	1 1 1
					* * *				
177.	11	E 1 1 1 1	0	16413.689	1193.812	704.285	112.000	373.358	1 1 1
178.	51	E 1 1 1 1	0	16494.709	1197.939	705.834	122.042	384.262	1 1 1
179.	21	E 1 1 1 1	0	1052.358	180.632	13.746	125.153	148.850	1 1 1
180.	61	E 1 1 1 1	0	1144.605	187.630	14.106	150.793	175.517	1 1 1

Figure F-6. Example Extract Records from the CBT/CS/CSS Merge Module Output File of Partial Combat Potentials

(2) The subfields of Fields 2 and 9 are filled with blanks or zeros in the examples shown. These subfields provide the means to include special identifiers that may be applied but are not needed if filed and named carefully in the first place. In AFP work to date, file naming has been sufficient to preserve the integrity of data sets.

(3) Field 3 contains an identifying number corresponding to a specific weapon type. Although it is not apparent from Figure F-6, Blue and Red weapon types may have the same identifying number. However, because the record identifiers in Field 1 fully distinguish Blue and Red records, there is no ambiguity with respect to weapon type after all.

(4) Field 4 contains the first or personnel component of the four-valued partial combat potential for each weapon of division. If input program variable TVALON=.TRUE., the field contains target-value-weighted personnel plus other weapon results.

(5) Field 5 contains the second or light armored vehicle component of the four-valued partial combat potential for each weapon of division. If input program variable TVALON=.TRUE., the field contains target-value-weighted results.

(6) Field 6 contains the third or heavy armored vehicle component of the four-valued partial combat potential for each weapon or division. If input program variable TVALON=.TRUE., the field contains target-value-weighted results.

(7) Field 7 contains the fourth or aircraft component of the four-valued partial combat potential for each weapon or division. If input program variable TVALON=.TRUE., the field contains target-value-weighted results.

(8) Field 8 contains the partial scalar combat potential for each weapon or division. Partial scalar potential is often referred to as the fifth component of five-valued partial combat potential. Scalar potential, as frequently noted, is not independent of four-valued potential.

F-4. RUNSTREAM. This section describes runstream of generation only for special standalone use of the CBT/CS/CSS Merge Module.

a. Normal Use. In normal AFP practice, the CBT/CS/CSS Merge Module is not executed in a standalone mode. The Merge Module usually is executed one or more times within a combined Combat Module and CBT/CS/CSS Merge Module run. Generation of these normal runstreams is described in paragraph D-4 of Appendix D on the Combat Module.

b. Special, Standalone Use. The need for special, standalone operation of the CBT/CS/CSS Merge Module may arise because critical CS/CSS data may be unavailable at the time of Combat Module runs or because a variation in CS/CSS data, target values, or shooter fractional lifetimes must be assessed. Standalone operation of the Merge Module is the more awkward and requires that normally discarded Combat Module output be retained. Each execution of the Combat Module generates (among other things) three usually temporary files or elements, ALLOC, FALLOC, AND TTLOS (named differently in production), used by (among other programs) the CBT/CS/CSS Merge Module. In regular AFP practice, these files are used soon after generation and discarded before another Combat Module execution. In full AFP production for 10 replications per combat environment, this generate-use-discard cycle occurs 160 times. For the special standalone, Merge Module mode, 3 x 160 = 480 files or elements must be saved for later use. And, it has sometimes been the case, two separate divisions or forces have been "in production" at the same time, 960 files or elements must be saved. Note that the 960 files or elements specially saved must be in addition to several hundred saved in the regular process. The full-scale special process puts a very heavy burden of the Agency's file storage and mainframe production to the point of very nearly excluding non-AFP users and customers. The standalone process is not recommended for AFP production. On the other hand, the standalone process can be very useful as a research method for small-scale (for few combat environments and or few replications) sensitivity analyses.

c. SSG Program for Standalone Runstsreams. An SSG program for generation of runstreams for standalone use of the CBT/CS/CSS Merge Module is shown in Figure F-7.

(1) SGS Section. The SGSs specify the symbols and controls necessary to generate correct runstreams. The examples in Figure F-7 are correct for the generation of a single runstream for an H-series division over the first four combat environments and two replications. Changes to specific SGSs identified below would, for example, lead to a runstream including all 16 combat environments over 10 replications. No run so large has been attempted. Indeed, a single run so large would be ill-advised. The generation of several shorter runs would be safer in many respects.

(a) PREFIX. The SGS "PREFIX" specifies a user-ID that prefixes filenames needed within Merge Module runs.

(b) FORCE. The SGS "FORCE" specifies the symbol identifying the division or force of current concern. The symbol may be used in filename, element names, and printed output headings.

(c) OJTPD. The SGS "OJTPD" specifies a year symbol to appear in output records.

(d) OKTHTR. The SGS "OKTHTR" specifies a theater symbol to appear in output records.

```
@SSG,BK
SGS
PREFIX H7
FORCE HM80
OJTPD '' 80''
OKTHTR ''  E''
OJVIS 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
OJDAY 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2
OJPOS 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
IBFOR ''  3456''
IRFOR ''  4567''
ICOMBO ''  1''
IJTPD 80
IJTHTR 1
ENV 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
POSTS 01 02 03 04 01 02 03 04 01 02 03 04 01 02 03 04
REP 1 2
R1 1
R2 2
ENVFIRST 1
ENVLAST 4
CASE NEW
CASEX N
RATIO ''1 3'' ''1 1'' ''1 4'' ''3 1'' ;
      ''1 3'' ''1 1'' ''1 4'' ''3 1'' ;
      ''1 3'' ''1 1'' ''1 4'' ''3 1'' ;
      ''1 3'' ''1 1'' ''1 4'' ''3 1''
BNODIV 41
RNODIV 41
MRGABS ''30GECTEST.870CXPS''
USEFILES N
TVALON  TRUE
@EOF
SKEL
*DEFINE ASGX
*EDIT ON
*CREATE SGS: XFILE [PREFIX,1,1,1][FORCE,1,1,1] [ENV,1,IE,1][
R[REP,1,R,1]
#ASG,A [XFILE,1,1,1][#1][XFILE,1,2,1].
#USE [#2].,[XFILE,1,1,1][#1][XFILE,1,2,1].
*REMOVE SGS XFILE
*END
*DEFINE ELEX
#ERS [#2].
#COPY,I [PREFIX,1,1,1][FORCE,1,1,1].[#1][ENV,1,IE,1]R[REP,1,R,1],[#2].
*END
*SET R TO 1
*BRKPT,K 30GECTEST.GOMERGEALL/[FORCE,1,1,1]R[REP,1,R,1]
#ASG,A H7BASEDATA.
#ASG,T 14.
```

Figure F-7. Example Setup of SSG Program for Generating Runstream for Separate Execution(s) of AFP CBT/CS/CSS Merge Module
(Page 1 of 3 pages)

```
#ASG,T 15.
#ASG,T 30.
#ASG,T SORT9.,///1000
#ASG,A 30PCAS3.
#ASG,T 16.
#ED 30PCAS3.FJE-1,16.
*IF [USEFILES,1,1,1] = N
#ASG,A [PREFIX,1,1,1][FORCE,1,1,1].
#ASG,A H7AFP.
#ASG,T 3.,///1000     .ALLOC
#ASG,T 4.,///1000     .FALLOC
#ASG,T 10.,///1000     .TTLOS
#USE 9.,10.
#ASG,T 29.
*END
*INCREMENT IE FPOM [ENVFIRST,1,1,1] TO [ENVLAST,1,1,1]
#ED H7BASEDATA.[FORCE,1,1,1]E[ENV,1,IE,1],15.
*IF [USEFILES,1,1,1] = Y
*CREATE SGS CSCSS [PREFIX,1,1,1]CSCSS[POSTS,1,IE,1][CASEX,1,1,1]
#ASG,A [CSCSS,1,1,1].
#ED    [CSCSS,1,1,1].,14.
*REMOVE SGS CSCSS
*ELSE
#ED H7AFP.[FORCE,1,1,1]M[POSTS,1,IE,1],14.
*END
*INCREMENT R FROM [R1,1,1,1] TO [R2,1,1,1]
#HDG UNCLASSIFIED MERGE [FORCE,1,1,1]E[ENV,1,IE,1] R[REP,1,P,1] [CASE,1,1,1]
*IF [USEFILES,1,1,1] = Y
#ASG,A [PREFIX,1,1,1][FORCE,1,1,1]E[ENV,1,IE,1][CASEX,1,1,1][REP,1,R,1].
#USE 29.,[PREFIX,1,1,1][FORCE,1,1,1]E[ENV,1,IE,1][CASEX,1,1,1][REP,1,R,1].
*PROCESS ASGX A 3
*PROCESS ASGX F 4
*PROCESS ASGX T 9
*ELSE
*PROCESS ELEX A 3
*PROCESS ELEX F 4
*PROCESS ELEX T 10
#USE 9.,10.
*END
#SORT,ES
VOLUME=SMALL
KEYW=3,35,36,B,A:4,35,36,B,A:2,35,36,B,A:1,35,36,B,A
FILEIN=9.
FILEOUT=SORT9.
#EOF
#USE 9.,SORT9.
#XQT [MRGABS,1,1,1]
*. IMAGE GIVING INDICFS IN CBT POT OUTPUT RECORDS
*EDIT ON
[ICOMBO,1,1,1]&
```

Figure F-7. Example Setup of SSG Program for Generating Runstream for Separate Execution(s) of AFP CBT/CS/CSS Merge Module
(Page 2 of 3 pages)

```
[OJTPD,1,1,1]&
[OKTHTR,1,1,1]&
  [OJVIS,1,IE,1]&
  [OJDAY,1,IE,1]&
  [OJPOS,1,IE,1]&
[IBFOR,1,1,1]&
[IRFOR,1,1,1]
TVALON     [TVALON,1,1,1]
*. IMAGE GIVING INDICES OF INPUT CS/CSS MODULI
*EDIT ON
[ICOMBO,1,1,1]&
  [IJTPD,1,1,1]&
  [IJTHTR,1,1,1]&
*SET IEX TO [POSTS,1,IE,1]
  [OJVIS,1,IEX,1]&
  [OJDAY,1,IEX,1]&
  [OJPOS,1,IEX,1]
[RATIO,1,IE,1]
*EDIT ON
 [BNODIV,1] &
*INCREMENT IBNO TO [BNODIV,1]
 [BNODIV,1,IBNO,1]&
*LOOP    .IBNO
*EDIT OFF
*EDIT ON
 [RNODIV,1] &
*INCREMENT IRNO TO [RNODIV,1]
 [RNODIV,1,IRNO,1]&
*LOOP    .IRNO
*EDIT OFF
#ADD 30GLOBAL.CVALS
#ADD 30GLOBAL.FRACTS
*IF [USEFILES,1,1,1] = N
#ED 29.,[PREFIX,1,1,1][FORCE,1,1,1].E[ENV,1,IE,1]R[REP,1,R,1]
#DATA,L 29.
#END
#ERS 29.
*ELSE
#DATA,L 29.
#END
#FREE 29.
*END
#ERS 9.
#ERS 30.
*LOOP    .R
#ERS 14.
#ERS 15.
*LOOP    .IE
#FREE 9.
#FREE 14.
#FREE 15.
#FREE 16.
#FREE 30.
@EOF
@EOF
```

Figure F-7. Example Setup of SSG Program for Generating Runstream for Separate Execution(s) of AFP CBT/CS/CSS Merge Module
(Page 3 of 3 pages)

(e) **OJVIS.** The SGS "OJVIS" specifies symbols corresponding to clear (1) and degraded (2) visibility for the AFP standard 16 combat environments. A symbol appears in output records and may appear in the input CS/CSS moduli records.

(f) **OJDAY.** The SGS "OJDAY" specifies symbols corresponding to day-time (1) and nighttime (2) for the AFP standard 16 combat environments. A symbol appears in output records and may appear in the input CS/CSS moduli records.

(g) **OJPOS.** The SGS "OJPOS" specifies symbols corresponding to RAPD (1), STATIC (2), RADE (3), and BAPD (4) postures for the AFP standard 16 combat environments. A symbol appears in output records and may appear in input CS/CSS moduli records.

(h) **IBFOR.** The SGS "IBFOR" specifies a Blue division or force numeric identifer for inclusion in output records.

(i) **IRFOR.** The SGS "IRFOR" specifies a Red division or force numeric identifier for inclusion in output records.

(j) **ICOMBO.** The SGS "ICOMBO" specifies a case symbol appearing in CS/CSS moduli input and partial combat potential output records. The symbol has been set to bbl in all AFP work to date.

(k) **IJTPD.** The SGS "IJTPD" specifies a year symbol appearing in input CS/CSS moduli records.

(l) **IJTHTR.** The SGS "IJTHTR" specifies a theater symbol appearing in input CS/CSS moduli records.

(m) **ENV.** The SGS "ENV" specifies symbols corresponding to the AFP standard 16 combat environments. The symbols may be used in a variety of names or headings.

(n) **POSTS.** The SGS "POST" specifies symbols for the postures (RAPD = 01, STATIC = 02, RADE = 03, BAPD = 04) corresponding to the AFP standard 16 combat environments. The symbols appear in the names of CS/CSS moduli elements. POSTS assures that correct CS/CSS moduli are matched combat environments.

(o) REP. The SGS "REP" specifies symbols corresponding to one or more replications of the Combat Module. The example in Figure F-7 shows only the symbols "1" and "2." The list could be longer because the following two SGSs specify just which consecutive replications are to be processed by the generated runstream. The symbols must match characters appearing in input and output files or elements. Note that the generated runstream element names includes the first symbol in REP. Hence, there is some danger of overwriting a runstream element before it has been used if several runstream generations precede runstream executions--unless the first entry in REP is changed between runstream generations.

(p) R1. The SGS "R1" specifies the position of the first replication symbol in SGS REP to be included in the generated runstream. The runstream will include all replications corresponding to the closed REP set positions R1 to R2.

(q) R2. The SGS "R2" specifies the position of the last replication symbols in SGS REP to be included in the generated runstream. The runstream will include all replications corresponding to the closed REP set position R1 to R2.

(r) ENVFIRST. The SGS "ENVFIRST" specifies the position of the first combat environment symbol in SGS ENV to be included in the generated runstream. The runstream will include all environments corresponding to the closed ENV set positions ENVFIRST to ENVLAST.

(s) ENVLAST. The SGS "ENVLAST" specifies the position of the last combat environment symbol in SGS ENV to be included in the generated runstream. The runstream will include all environments corresponding to the closed ENV set positions ENVFIRST to ENVLAST.

(t) CASE. The SGS "CASE" specifies an identifier to appear in the headings of printed output.

(u) CASEX. The SGS "CASEX" specifies an identifier appearing in the CS/CSS filename (only if the CS/CSS moduli input are in file format). The identifier is of no concern if CS/CSS moduli are in elements (the current preferred practice).

(v) RATIO. The SGS "RATIO" specifies the number of Blue and of Red divisions by AFP standard combat environments. In the example in Figure F-7, "13" represents one Blue and three Red divisions in each of the environments (1, 5, 9, 13) corresponding to posture RAPD.

(w) BNODIV. The SGS "BNODIV" specifies the AFP weapon type number of Blue nondivisional weapons. The CBT/CS/CSS Merge Module excludes nondivisional weapons from COP computations.

(x) RNODIV. The SGS "RONDIV" specifies the AFP weapon type number of Red nondivisional weapons. The CBT/CS/CSS Merge Module excludes nondivisional weapons from COP computations.

(y) MRGABS. The SGS "MRGABS" specifies the file and element name for the Merge Module's current absolute program element.

(z) USEFILES. The SGS "USEFILES" specifies whether the principal input and output of the Merge Module are files (Y)es) or elements (N)o).

(aa) TVALON. The SGS "TVALON" specifies whether or not the first four components of partial combat potentials are to be weighted by target values. TVALON=.FALSE. suppresses the weighting. TVALON=.TRUE. forces weighting. All replications of all combat environments should be processed at the same setting of TVALON for rollups and interpolations to be consistent.

(2) **SKELeton Section.** In accord with the above-described SGSs, the SKEL section shown in Figure F-7 generates a single runstream element for the specified division or force, combat environments, and replications. Note that the SKEL is based on the assumption that some file names are permanent. Those "permanent" names may be changed within the SKEL section; or, if name changes are expected to be frequent, new SGSs may be added to the SGS section and referenced in the SKEL section.

F-5. PROGRAM

a. The CBT/CS/CSS Merge Module program is more confusing than most of the AFP System's other modules, short of the Combat Module itself. No one set out to make the program confusing. It is just that the current version is the result of several changes in the scope of the program. Each change was implemented by patches to the preceding version without ever starting over. The prototype was developed on a microcomputer using a FORTRAN compiler permitting a maximum of three-dimensional arrays. Partly because of that dimensioning limitation and partly to keep Blue and Red clearly delineated, many separate arrays and program loops for Blue and Red data were implemented in the prototype. Although those distinctions were preserved in the mainframe version of the module, later extensions to the module often introduced higher-dimensioned arrays in which the distinction between Blue and Red was limited to the difference between "1" and "2" as indices within one dimension of an array. From the beginning, artillery was handled differently from direct fire, sometimes by special statements within an otherwise direct fire sequence, sometimes by a separate block of statements following a direct fire sequence of instructions. Perhaps most confusing of all, the original method (based on "local exchange ratios") for computing partial (in the sense of pertaining to a single instead of all 16 environments) combat potentials was retained as a variant after the now standard method (based on "global exchange ratios") was implemented. The current version of the module still deactivated code for writing to a combat environment rollup file that is not used. The admirable and professional goal of "cleaning up the module" has had to be ignored in the face of higher priority requirements to make other modules work. Throughout system development, the strategy has been to build those things that did not exist, to fix those things that did not work correctly, and last to improve things that worked correctly but awkwardly. The operator/programer may find it helpful to keep in mind several special considerations.

(1) The CBT/CS/CSS Merge Module treats direct fire and indirect fire weapons differently.

(2) The notions of "lifetime potentials" differ among weapons types.

(3) The module concurrently builds Blue and Red potentials.

(4) Partial potentials are developed largely concurrently by two different methods.

(5) Four-valued and scalar partial combat potentials (i.e., five-valued partial potentials) are developed concurrently.

(6) Unmodulated and modulated partial potentials are developed concurrently.

(7) Division potentials are accumulated as the module works its way through weapon inventories weapon type by weapon type.

b. Figure F-8 displays the basic logic of subroutine TARTY of the CBT/CS/CSS Merge Module program. The basic strategy of the module is to process Combat Module results Blue weapon type by Blue weapon type. For each Blue weapon type, results of matchups against Red weapons are processed Red weapon type by Red weapon type. And for each Blue/Red weapon type-on-type matchup, results are processed day by day (normally only for 2 days). Much updating of several different multidimensional working arrays occurs over and over again as days, Red weapon types, and Blue weapon types are cycled. Some updating is done without reference to CS/CSS moduli in order to compute unmodulated potentials. But an equal amount of updating is done with reference to the CS/CSS moduli in order to compute the modulated potentials. It is not until the working arrays have been filled that output to the partial combat potential file is begun. The essence of the CBT/CS/CSS Merge Module is the computation of unmodulated and modulated scores, CIPs, and COPs from Combat Module output and CS/CSS moduli (the latter only for modulated quantities). The computation is performed by two methods. The standard method is usually called the "global exchange ratio" method, or the GER method. The variant (actually the original) is called the "local exchange ratio" method, or the LER method. The following paragraphs are intended to clarify the distinctions between the methods, first in a simplified way and then in more detail. The examples are all from the point of view of the Blue side as "shooters." Similar examples could have been constructed from the Red side's point of view inasmuch as the Combat Module and the CBT/CS/CSS Merge Module treat both sides symmetrically. Although the Merge Module treats only one combat environment per run, the module develops unmodulated and modulated scores, CIPs, and COPs by two methods for both Blue and Red sides concurrently. Some patience is required to unravel the intertwined processes and to grasp the similarities and differences.

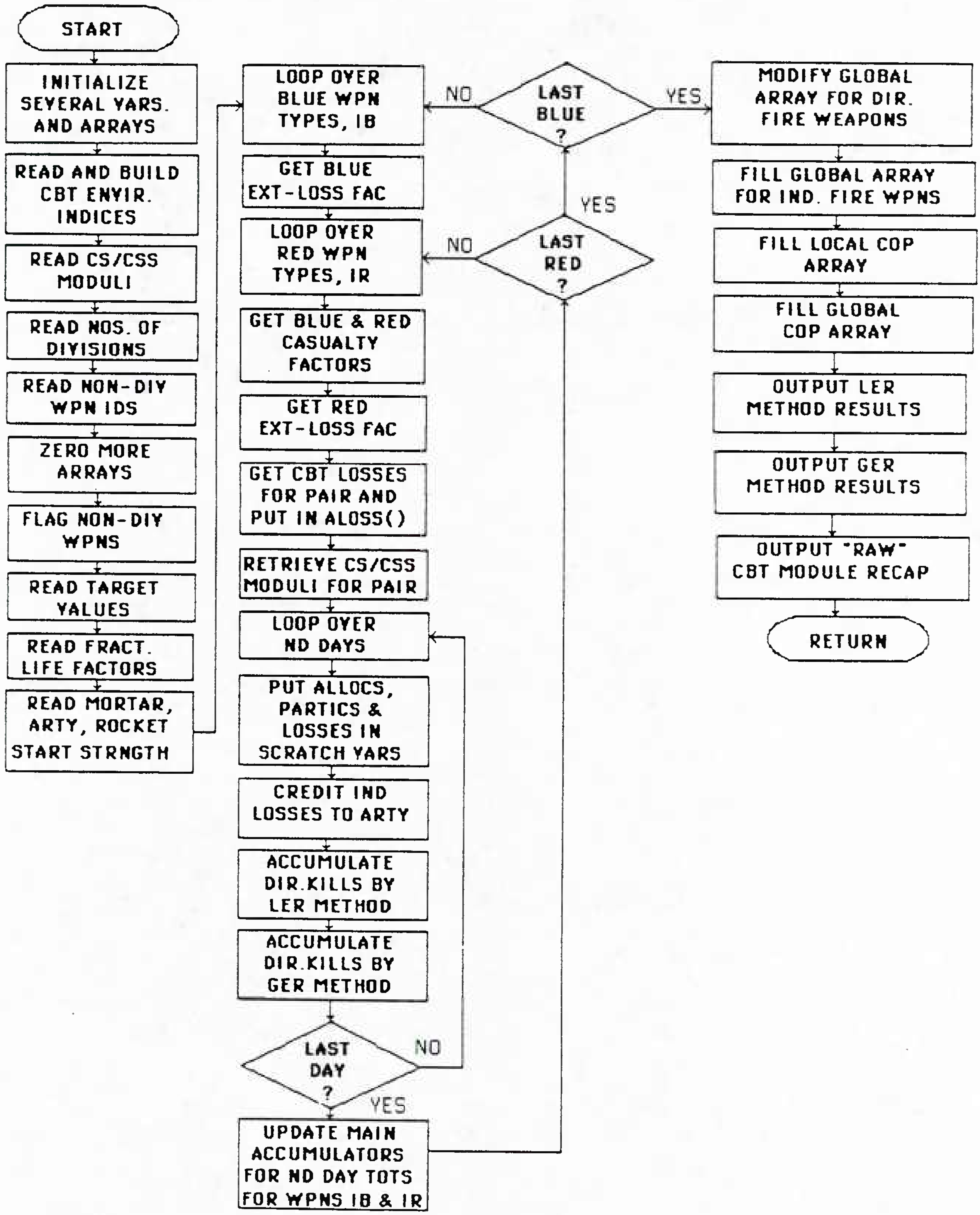


Figure F-8. Flow Diagram of the Basic Logic of Subroutine TARTY of the CBT/CS/CSS Merge Module

(1) The principal difference between the "global" and "local" exchange ratio methods lies in whether exchange ratios are computed for each type-on-type engagement (as in the "local" method) or only after kills and losses have been separately summed over all engagements (as in the "global" method). The difference, then, is in the exact sequence of summation and division operations. In highly simplified form, the difference involves terms of the form--

(a) Global:

$$\text{GER(IB)} = \sum_{\text{IR}} \sum_{\text{ID}} \frac{\text{(Red weapons of type IR killed by Blue weapons of type IB on day ID)}}{\text{(Blue weapons of type IB lost to all causes over all days)}}$$

(b) Local:

$$\text{LER(IB)} = \text{(1/ND)} * \sum_{\text{IR}} \sum_{\text{ID}} \frac{\text{(Red weapons of type IR killed by Blue weapons of type IB on day ID)}}{\text{(Blue weapons of type IB lost in engagement with Red type IR on day ID)}}$$

(2) The symbols used in the following paragraphs to make the GER and LER notions more specific in relation to the determination of partial combat potentials are defined in the following subparagraphs. The fact that a single execution of the Combat Module or CBT/CS/CSS Merge Module applies to a single combat environment is disregarded except in paragraph (4) below. Otherwise, the following symbols should also be indexed by IE to indicate the corresponding combat environment. The CBT/CS/CSS Merge Module program and subprograms apply a different symbolism and somewhat different orders of summation from the examples below.

(a) IB is the index of the IBth Blue weapon type.

(b) IR is the index of the IRth Red weapon type.

(c) IT is the index of the ITth component of five-valued partial combat potentials. The components are: personnel, light armored vehicles, heavy armored vehicles, aircraft, and weighted scalar.

(d) ID is the index of the IDth day represented within the AFP Combat Module.

(e) ND is the total number of days represented within the AFP Combat Module.

(f) GLOSS(IB) is the sum of losses of weapon type IB to all causes over all days. GLOSS is "global" in the sense of being a sum over all Red shooters and over all days. In the standard 60 by 60 engagement of matrix of the Combat Module, a Blue weapon type is allocated within a single row across 60 Red columns, and a Red weapon type is allocated within a single column across 60 Blue rows. Hence, GLOSS(IB) is the sum of Blue losses within a complete row of the 60 by 60 engagement matrix.

(g) LLOSS(IB,IR,ID) is the loss of Blue weapon type IB to all causes related to engagement with Red weapon type IR on Day-ID. LLOSS is "local" is the sense of referring to losses incurred on a single day within a single "cell" of the standard 60 by 60 engagement matrix of the Combat Module.

(h) DBL(IB,IR) is the sum of direct fire losses of Blue weapon type IB to Red weapon type IR over all days.

(i) DBL(IB,IR,ID) is the direct fire losses of Blue weapon type IB to Red weapon type IR on Day-ID.

(j) DRL(IT,IR,IB) is the sum of direct fire losses within target category IT or Red weapon type IR to Blue weapon type IB over all days.

(k) DRL(IT,IR,IB,ID) is the direct fire losses within target category IT or Red weapon type IR to Blue weapon type IB in Day-ID.

(l) IBL(IB,IR) is the sum of indirect fire losses of Blue weapon type IT to Red indirect fire weapon type IR over all days.

(m) IBL(IB,IR,ID) is the indirect fire losses of Blue weapon type IB to Red indirect fire weapon type IR on Day-ID.

(n) EBL(IB,IR) is the sum of external losses of Blue weapon type IB related to engagements with Red weapon type IR over all days.

(o) EBL(IB,IR,ID) is the external losses of Blue weapon type IB related to engagements with Red weapon type IR on Day-ID.

(p) ERL(IT,IR,IB) is the sum of the external losses within target category IT of Red weapon type IR related to engagement with Blue weapon type IB over all days.

(q) ERL(IT,IR,IB,ID) is the external losses within target category IT of Red weapon type IR related to engagements with Blue weapon type IB on Day-ID.

(r) F is a factor specifying the fraction of external losses to a target which are to be credited to a shooter. In general, the factor F results in a splitting of external losses between direct and indirect activities.

(s) AVAIL(IB) is the number of Blue weapons of type IB input to the Combat Module as part of the starting inventory. That inventory may include several divisions.

(t) ALLOC(IB,IR,ID) is the number of Blue weapons of type IB allocated against Red weapon type IR on Day-ID.

(u) NBD is the number of Blue divisions represented in Combat Module results. NBD is applied frequently as a multiplier or divisor as appropriate as the CBT/CS/CSS Merge Module viewpoint shifts from all weapons of a type to a single divisions's worth.

(v) USCORE(IT,IB) is the ITth component of the unmodulated score of one division's worth of Blue weapon type IB.

(w) UCIP(IT,IB) is the ITth component of the unmodulated CIP of Blue weapon type IB.

(x) MSCORE(IT,IB) is the ITth component of the modulated score of one division's worth of Blue weapon type IB.

(y) MCIP(IT,IB) is the ITth component of the modulated CIP of Blue weapon type IB.

(z) FRACT(IB) is the fractional lifetime factor for Blue weapon type IB, e.g., the net factor is usually 0.5 (for half lifetimes) for ordinary DIRECT fire weapons.

(aa) UCOP(IT) is the ITth target category component of unmodulated COP.

(ab) MCOP(IT) is the ITth target category component of modulated COP.

(3) The GER method requires computation along the following lines.

(a) For Blue direct fire weapons in general;

$$GLOSS(IB) = \sum_{IR} (DBL(IB,IR) + IBL(IB,IR) + EBL(IB,IR))$$

If GLOSS(IB) = 0.0, then GLOSS(IB) - 1.0.

The losses to Blue weapon type IB are summed over all uses of that weapon before any division to determine exchange ratio. To avoid division by 0.0, 1.0 is the minimum permitted loss.

USCORE(IT,IB) = FRACT(IB) * AVAIL(IB)/(GLOSS(IB) * NBD) *

$$\sum_{IR} (DRL(IT,IR,IB) + F * ERL(IT,IR,IB))$$

The unmodulated score is thus purely a quotient of sums, not a sum of quotients.

UCIP(IT,IB) = USCORE(IT,IB) * NBD/AVAIL(IB)

The unmodulated CIP is simply a mean score per weapon.

MSCORE(IT,IB) = (FRACT(IB) * AVAIL(IB) / (GLOSS(IB) * NBD)) *

$$\sum_{IR} (DRL(IT,IR,IB) + F * ERL(IT,IR,IB)) * CSCSS(IB,IR)$$

The modulated score, too, is a quotient of sums, not a sum of quotients.

MCIP(IT,IB) = MSCORE(IT,IB) * NBD / AVAIL(IB)

(b) For Blue indirect fire weapons in general:

USCORE(IT,IB) = FRACT(IB / NBD) *

$$\sum_{IR} (DRL(IT,IR,IB) + F * ERL(IT,IR,IB))$$

It is implied that the indirect firer loses only 1.0 weapon.

UCIP(IT,IB) = USCORE(IT,IB) * NBD / AVAIL (IB)

MSCORE(IT,IB) = (FRACT(IB) / NBD) *

$$\sum_{IR} (DRL(IT,IR,IB) + F * ERL(IT,IR,IB)) * CSCSS(IB,IR)$$

MCIP(IT,IB) = MSCORE(IT,IB) * NBD / AVAIL(IB)

(c) The division COPs;

$$UCOP(IT) = \sum_{IB} USCORE(IT,IB)$$

$$MCOP(IT) = \sum_{IB} MCSCORE(IT,IB)$$

(4) The LER method required computation of the following:

(a) For Blue direct fire weapons in general:

LLOSS(IB,IR,ID) = DBL(IB,IR,ID) + IBL(IB,IR,ID) + EBL(IB,IR,ID)

If LLOSS(IB,IR,ID) = 0.0, then LLOSS(IB,IR,ID) = 1.0.

Here, the sum of losses applies to a single type-on-type engagement on a single day.

$$USCORE(IT,IB) = (FRACT(IB) / (ND * NBD)) * \sum_{IR} \sum_{ID} \frac{(DRL(IT,IR,IB,ID) + F * ERL(IT,IR,IB,ID)) * ALLOC(IB,IR,ID)}{LLOSS(IB,IR,ID)}$$

The unmodulated score is a sum of quotients and not simply a quotient of sums as in the GER case.

UCIP(IT,IB) = USCORE(IT,IB) * NBD / AVAIL(IB)

$$MSCORE(IT,IB) = (FRACT(IB) / (ND * NBD)) * \sum_{IR} CSCSS(IB,IR) * \sum_{ID} \frac{(DRL(IT,IR,IB,ID) + F*ERL(IT,IR,IB,ID)) *ALLOC(IB,IR,ID)}{LLOSS(IB,IR,ID)}$$

MCIP(IT,IB) = MSCORE(IT,IB) * NBD / AVAIL(IB)

(b) For Blue indirect fire weapons in general:

LLOSS(IB) = DBL(IB) + IBL(IB) + EBL(IB)

If LLOSS(IB) = 0.0, then LLOSS(IB) = 1.0.

Unlike in the GER case, the "actual losses" of indirect fire weapons are considered except that 1.0 is the minimum permitted loss.

$$USCORE(IT,IB) = FRACT(IB) / (ND * NBD * LLOSS(IB)) * \sum_{IR} \sum_{ID} (DRL(IT,IR,IB,ID) + IRL(IT,IR,IB,ID) + F*ERL(IT,IR,IB,ID))$$

$$UCIP(IT,IB) = USCORE(IT,IB) * NBD / AVAIL(IB)$$

$$MSCORE(IT,IB) = (FRACT(IB) / (ND * NBD * LLOSS(IB)))* \sum_{IR} CSCSS(IB,IR)* \sum_{ID} (DRL(IT,IR,IB,ID) +IRL(IT,IR,IB,ID)+F*ERL(IT,IR,IB,ID))$$

$$MCIP(IT,IB) = MSCORE(IT,IB) * NBD / AVAIL(IB)$$

(c) The divisions COPs;

$$UCOP(IT) = \sum_{IB} USCORE(IT,IB)$$

$$MCOP(IT) = \sum_{IB} MSCORE(IT,IB)$$

(5) The AFP CBT/CS Merge Module does not roll up partial combat potentials over all 16 combat environments. That function is performed within the AFP Rollup and Stats Module. In terms of the above notation, an example of the computation performed within the Rollup and Stats Module is:

$$MCOP(IT) = \sum_{IE}^{16} MCOP(IT,IE) * ENV(IE)$$

Where the partial MCOPs (IT) above have been extended to MCOP(IT,IE) to show dependence on corresponding combat environment, IE. The term MCOP has been replaced to indicate that the environmentally weighted sum of partial

COPs is a final COP. Similar summations lead to the full variety of final unmodulated and modulated scores, CIPs, and COPs for both Blue and Red divisions.

c. **GGGTARTY** is the current version of the source program of the principal subprogram of the CBT/CS/CSS Merge Module. The GGGTARTY source program is listed in Figure F-9.

(1) The principal reference arrays are initialized at load time or early in program execution.

(a) BAA(10). The array BAA stores the starting strengths of Blue mortar and artillery weapon types. The indexing of BAA(I) for indirect weapon types for I = 1 to 10 relative to all weapon tyes W = 1 to 60 is W = LIMD + I. Array RAA is the Red counterpart of Blue array BAA().

(b) RAA(10). The array RAA() stores the starting strengths of Red mortar and artillery weapon types. The indexing of RAA() is the same as for Blue array BAA() described in (1) immediately above. Array BAA() is the Blue counterpart of Red array RAA().

(c) CSCSS(2,60,60). The array CSCSS() receives and stores CS/CSS module generated earlier by the separate AFP CS/CSS Module. An array element CSCSS(I,J,K) is indexed;

1. I=1 to 2 for Blue and Red sides.

2. If I=1, then

   J=1 to 60 for the 60 Blue weapon types.

   K=1 to 60 for the 60 Red weapon types.

3. If I=2, then

   J=1 to 60 for the 60 Red weapon types.

   K=1 to 60 for the 60 Blue weapon types.

(d) BNDX(60). The array BNDX() receives and stores the weapon indices of any Blue weapon which are nondivisional. If there are only N nondivisional Blue weapon types, then only the first N elements of BNDX() are filled. For example, during AFP system development only Blue weapon type 41 was nondivisional. Hence, N=1, and BNDX(1)=41. As a result of entries in BNDX(), the program sets the corresponding logical value in the separate array BNDW() to .TRUE. for nondivisional. Array RNDX() is the Red counterpart of Blue array BNDX().

```
          SUBROUTINE TARTY
C            15 JUN 83 -- G.E.C.
C            COMBINES COMBAT MODULE OUTPUT WITH CS/CSS FACTORS
C          12 APR 83 -- CHANGE MODULI & OUTPUT ID'S -- G.E.C.
C        ALGORITHM AND MOST OF CODE BY G. COOPER. MODIFIED BY J.WARREN
C          AND S. BRAVY
C            EXTENDED 24-28 OCT 83 BY G.E.C. TO PROVIDE GLOBAL
C            (I.E., ROW-WISE & COLUMN-WISE EXCHANGE RATIOS --
C            [ROW KILLS]/[ROW LOSSES]
C            [COLUMN KILLS/COLUMN LOSSES]
C            EXTENDED 4 NOV 83 TO APPLY GENERALIZED CATEGORY
C            WEIGHTS AND THREE NEW 'STOPPING' CRITERIA --
C              * 0.5-LIFE FOR DIRECT FIRE GROUND SYSTEMS
C              * BASIC LOAD DEPLETION FOR SAMS
C              * FORTNIGHT DURATION FOR MORTARS, ARTY, & MLRS
C              EXTENDED 9/84 FOR CVALS() OPTION TVALON IN POTENTIALS
C                ELEMENTS 1-5
          PARAMETER ITHTR=1
C          INCLUDE PARM,LIST
C          INCLUDE BK1,LIST
C          INCLUDE BK3,LIST
          DIMENSION ALOSS(14,2,2),BCAS(4),RCAS(4)
          DIMENSION BT(5,2),RT(5,2),BAL(10),RAL(10),BAA(10),RAA(10)
          DIMENSION B(5,2,60),R(5,2,60),ABTOT(60),ARTOT(60)
          DIMENSION CSCSS(2,60,60)
          DIMENSION USCOR(2,60,4),MSCOR(2,60,4)
          DIMENSION UCIP(2,60,4),MCIP(2,60,4)
C         DIMENSION UCOP(2,4),MCOP(2,4),SIZES(2)
          DIMENSION BNDX(60),RNDX(60),BNDW(60),RNDW(60)
          DIMENSION OMEGAE(5,4,60,2),OMEGAC(5,2,2),ZLOSS(60,2)
          DIMENSION DIVS(2),ISNO(4,2),START(60,2),NTYPS(2)
          REAL MSCOR,MCIP,MCOP
          INTEGER COMBO,BNDX,RNDX,CASE,SIZES
C
C         DEFINE FILE IU11(NR11,IRSZ11,F,NEXT11)
C         DEFINE FILE IU12(NR12,IRSZ12,F,NEXT12)
C
          CHARACTER*3 KTHTR,LTHTR
          CHARACTER*3 ATHTR(ITHTR)
          LOGICAL TEST,BNDW,RNDW,TVALON
C
C            B(I,J,K)      BLUE KILLS
C              I= 1          PERSONNEL+OTHER
C                 2          LT VEHS
C                 3          HVY VEHS
C                 4          AIRCRAFT
C                 5          SCALAR
C              J= 1          UNMODULATED
C                 2          MODULATED
C              K= 1,M        KTH WEAPONS TYPE
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 1 of 15 pages)

```
C            R(I,J,K)      RED KILLS          SUBSCRIPTS LIKE B(I,J,K)
C            BAL(I)        BLUE ARTY LOSSES, TYPE LIMD+I
C            RAL(I)        RED ARTY LOSSED, TYPE LIMD+I
C            BAA(I)        BLUE ARTY STARTING STRENGTH, TYPE LIMD+1
C            RAA(I)        RED ARTY STARTING STRENGTH, TYPE LIMD+1
C            OMEGAE(I,J,K,L)      GLOBAL SCORES & CIPS
C              I=1,5                 PERS+OTHER,...,SCALAR
C              J=1,4                 USCORE,UCIP,MSCORE,MCIP
C              K=1,NTYPS(IS)         WPN TYPE
C              L=1,2                 BLUE,RED
C            OMEGAC(I,J,K)        GLOBAL COPS
C              I=1,5                 PERS+OTHER,...,SCALAR
C              J=1,2                 UCOP,MCOP
C              K=1,2                 BLUE,RED
C            ZLOSS(I,J)           DIRECT & INDIRECT LOSSES
C              I=1,NTYPS(IS)         WPN TYPE
C              J=1,2                 BLUE,RED
C
          COMMON/CIM1/DUM(240),EXTLOS(2,60)
          COMMON/SOR/ISOR
          COMMON/GLOBAL/IBFOR,IRFOR,JCASE
          COMMON/AWRK/IWRK,ISCNT,JTHTRZ,JTPDZ,JVISZ,JPOSZ,JDAYZ
          COMMON/CHAR/KTHTR
          COMMON/CATVAL/CVALS(5,60,2)
          COMMON/FRACTS/FRAX(60,2)
C         DATA ATHTR/'  E'/
          DATA IDIM/14/,NTYPS/2*60/,IUIN8/14/
          DATA ZTOL/0.2/,M,N/2*60/
          DATA FRACLF/0.50/,ND/2/
          DATA LIMD/50/,IPOS/4/,IDAY/2/
          DATA CSCSS/7200*1.0/
          DATA BNDW/60*.FALSE./,RNDW/60*.FALSE./
          DATA ALOSS/56*0.0/
          DATA ISNO/10,30,50,70,20,40,60,80/
C
C         MINFO RECORD FORMAT
C       1 FORMAT(I5,A3,I4,3I3,I5,5F10.3,2I6,I5)
C         FOR OUTPUT OF ISCNT,JTHTR,JTPD,JVIS,JPOS,JDAY,IWID,XPERS,
C           XLVEH,XHVEH,XACFT,XSCLR,IBFOR,IRFOR,JCASE
          CALL OUTPAR
          READ(5,2) TVALON
        2 FORMAT(10X,L5)
          KLIM=5
          KSTEP=1
          IF(TVALON) THEN
            KLIM=1
            KSTEP=4
          ENDIF
          ISOR=6
          IWRK=30
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 2 of 15 pages)

```
C       IBFOR=1
C       IRFOR=1
        CVOPT=1.0
        CALL ZERO(BT,10)
        CALL ZERO(RT,10)
        CALL ZERO(B,600)
C
C     READ IN THE CSCSS DATA
C
        READ(5,13) COMBO,JTPD,JTHTR,JVIS,JDAY,JPOS
 13     FORMAT()
C       KTHTR=ATHTR(JTHTR)
        CASE=(JVIS-1)*IDAY*IPOS+(JDAY-1)*IPOS+(JPOS-1)
        JCASE=CASE
        PREF1=0
        PREF2=0
C        PREF3=(JTHTR-1)*ICASE+CASE
C        PREF4=(COMBO-1)*ITHTR*ITPD*ICASE+(JTPD-1)*ITHTR*ICASE+PREF3
 15     READ(IUIN8,23,END=50) LSCNT,LTHTR,LTPD,LVIS,LPOS,LDAY
     &      ,LBWID,XBMOD,LRWID,XRMOD
 23     FORMAT(I6,A3,I4,I3,I3,I3,I5,F10.3,I5,F10.3)
        TEST=(LSCNT .EQ. 200 ) .AND. (KTHTR .EQ. LTHTR)
        TEST=TEST .AND. (JTPD .EQ. LTPD)
        TEST=TEST .AND. (JVIS .EQ. LVIS)
        TEST=TEST .AND. (JPOS .EQ. LPOS)
        TEST=TEST .AND. (JDAY .EQ. LDAY)
        IF ( TEST ) THEN
          CSCSS(1,LBWID,LRWID)=XBMOD
          CSCSS(2,LRWID,LBWID)=XRMOD
        ENDIF
        GOTO 15
 50     CONTINUE
C     NUMBER OF BLUE AND RED DIVISIONS
        READ(5,13) DIVS(1), DIVS(2)
C        DIVS(1)=NBDIV
C        DIVS(2)=NRDIV
C     READ THE BLUE NON-DIVISIONAL WEAPON COUNT AND INDICES
        READ(5,13) NBNDIV,(BNDX(I),I=1,NBNDIV)
C     READ THE RED NON-DIVISIONAL WEAPON COUNT AND INDICES
        READ(5,13) NRNDIV,(RNDX(I),I=1,NRNDIV)
C
        CALL ZERO(R,600)
        CALL ZERO(ABTOT,M)
        CALL ZERO(ARTOT,N)
        CALL ZERO(BAA,10)
        CALL ZERO(RAA,10)
        CALL ZERO(BAL,10)
        CALL ZERO(RAL,10)
        CALL ZERO(OMEGAE,2400)
        CALL ZERO(OMEGAC,20)
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 3 of 15 pages)

```
      CALL ZERO(ZLOSS,120)
C   SET UP POINTERS TO NON-DIVISIONAL RESOURCES
      DO 60 I=1,NBNDIV
        BNDW(BNDX(I))=.TRUE.
60    CONTINUE
      DO 70 I=1,NRNDIV
        RNDW(RNDX(I))=.TRUE.
70    CONTINUE
C
C       GET CATEGORY VALUES
      CALL GETCAT
C
C       GET FRACTIONAL LIFE ADJUSTERS
C
      CALL GETFRX
C
C     GET ARTY STARTING STRENGTHS
      CALL GETART(BAA,RAA,LIMD,M,N)
C     DIRECT FIRE ENGAGEMENTS ARE ANALYZED AS TYPE-ON-TYPE
C     EXCHANGE RATIOS & ALLOCATIONS.  MORTARS & ARTY ARE
C     ANALYZED AS (GLOBAL TOTAL KILLS)/(GLOBAL TOTAL LOSSES).
C     LOOP OVER BLUE WPN TYPES
      DO 10 IB=1,NTYPS(1)
C     BLUE EXTERNAL LOSS FACTOR
      EXB=EXTLOS(1,IB)
C     LOOP OVER RED WPN TYPES
      DO 20 IR=1,NTYPS(2)
C     GET BLUE WPN CASUALTY VALUES
      CALL GETCA(2,IR,IB,BCAS)
C     GET RED WPN CASUALTY VALUES
      CALL GETCA(1,IB,IR,RCAS)
C     RED EXTERNAL LOSS FACTOR
      EXR=EXTLOS(2,IR)
C     GET CBT MODULE RESULTS FOR THIS MATCHUP
      CALL GETLOS(IB,IR,ALOSS,IDIM,ND)
C     GET BLUE & RED CS/CSS FACTORS FOR THIS PAIR
      BMOD=CSCSS(1,IB,IR)
      RMOD=CSCSS(2,IR,IB)
C
      BSUM=0.0
      RSUM=0.0
C     LOOP OVER DAYS
C       GLOBAL KILL ACCUMULATORS OVER ND DAYS
      GBNET=0.0
      GRNET=0.0
      DO 1000 ID=1,ND
C     ALLOCATIONS
      AB=ALOSS(1,1,ID)
      AR=ALOSS(1,2,ID)
C     PARTICIPATIONS
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 4 of 15 pages)

```
      PB=ALOSS(2,1,ID)
      PR=ALOSS(2,2,ID)
C     DIRECT LOSSES
      DB=ALOSS(3,1,ID)
      DR=ALOSS(3,2,ID)
C     EXTERNAL LOSSES
      EB=(EXB/(1.0-EXB))*ALOSS(1,1,ID)
      ER=(EXR/(1.0-EXR))*ALOSS(1,2,ID)
C     THRU 100, SUM INDIRECT LOSSES      ALSO CREDIT INDIRECT WPNS
      BI=0.0
      RI=0.0
      DO 100 L=5,IDIM
      LX=L+LIMD-4
      XB=ALOSS(L,1,ID)
      XR=ALOSS(L,2,ID)
      IF((XB.LE.0.01).AND.(XR.LE.0.01)) GOTO 100
      BI=BI+XB
      RI=RI+XR
      BMOD1=CSCSS(1,IB,LX)
      RMOD1=CSCSS(2,LX,IB)
      BMOD2=CSCSS(1,LX,IR)
      RMOD2=CSCSS(2,IR,LX)
C     CREDIT KILLS TO INDIRECT WEAPONS
      DO 90 IV=1,4
      CV=CVALS(IV,IR,2)
      IF(TVALON) CVOPT=CV
      BU=RCAS(IV)*XR
      B(IV,1,LX)=B(IV,1,LX)+BU*CVOPT
      B(5,1,LX)=B(5,1,LX)+BU*CV
      BM=BU*BMOD2
      B(IV,2,LX)=B(IV,2,LX)+BM*CVOPT
      B(5,2,LX)=B(5,2,LX)+BM*CV
      CV=CVALS(IV,IB,1)
      IF(TVALON) CVOPT=CV
      RU=BCAS(IV)*XB
      R(IV,1,LX)=R(IV,1,LX)+RU*CVOPT
      R(5,1,LX)=R(5,1,LX)+RU*CV
      RM=RU*RMOD1
      R(IV,2,LX)=R(IV,2,LX)+RM*CVOPT
      R(5,2,LX)=R(5,2,LX)+RM*CV
   90 CONTINUE
      DO 95 I15=KLIM,5,KSTEP
      B(I15,1,LX)=B(I15,1,LX)+XR*CVALS(5,IR,2)
      B(I15,2,LX)=B(I15,2,LX)+XR*BMOD2*CVALS(5,IR,2)
      R(I15,1,LX)=R(I15,1,LX)+XB*CVALS(5,IB,1)
      R(I15,2,LX)=R(I15,2,LX)+XB*RMOD1*CVALS(5,IB,1)
   95 CONTINUE
  100 CONTINUE
C     FOR INDIR FIRE WPN, IGNORE THE TYPE-ON-TYPE ALLOCATION
      IF(IB.GT.LIMD) AB=1.0
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 5 of 15 pages)

```
      IF(IR.GT.LIMD) AR=1.0
      ABTOT(IB)=ABTOT(IB)+AB
      ARTOT(IR)=ARTOT(IR)+AR
      IF(ID.EQ.1) THEN
        START(IB,1)=START(IB,1)+AB
        START(IR,2)=START(IR,2)+AR
      ENDIF
      BTOT=DB+EB+BI
      RTOT=DR+ER+RI
      BNET=DB
      X=DB+BI
      IF(X.GE.ZTOL) BNET=BNET+EB*DB/X
      RNET=DR
      X=DR+RI
      IF(X.GE.ZTOL) RNET=RNET+ER*DR/X
      IF(BTOT.GE.ZTOL) THEN
        BXR=RNET/BTOT
      ELSE
        BXR=RNET
      ENDIF
C     FOR INDIR FIRE WPN, TAKE KILLS, NOT XCHANGE RATIO
      IF(IB.GT.LIMD) BXR=RNET
      IF(RTOT.GE.ZTOL) THEN
        RXR=BNET/RTOT
      ELSE
        RXR=BNET
      ENDIF
C     FOR INDIR FIRE WPN, TAKE KILLS, NOT XCHANGE RATIO
      IF(IR.GT.LIMD) RXR=BNET
      BSUM=BSUM+AB*BXR
      RSUM=RSUM+AR*RXR
      IF(IB.GT.LIMD) BAL(IB-LIMD)=BAL(IB-LIMD)+BTOT
      IF(IR.GT.LIMD) RAL(IR-LIMD)=RAL(IR-LIMD)+RTOT
C
C       UPDATE GLOBAL ROW/COLUMN EXCHANGE ARRAYS
C
      ZLOSS(IB,1)=ZLOSS(IB,1)+BTOT
      ZLOSS(IR,2)=ZLOSS(IR,2)+RTOT
      GBNET=GBNET+BNET
      GRNET=GRNET+RNET
C
 1000 CONTINUE
      BSCORM=BSUM*BMOD
      RSCORM=RSUM*RMOD
C     UPDATE MAIN ACCUMULATOR ARRAYS
      DO 1800 IV=1,4
      CV=CVALS(IV,IR,2)
      IF(TVALON) CVOPT=CV
C     UNMODULATED
      BU=RCAS(IV)*BSUM
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 6 of 15 pages)

```
            GBU=RCAS(IV)*GRNET
      C     MODULATED
            BM=RCAS(IV)*BSCORM
      C     COMPONENT OF 4-VECTOR
            B(IV,1,IB)=B(IV,1,IB)+BU*CVOPT
            OMEGAE(IV,1,IB,1)=OMEGAE(IV,1,IB,1)+GBU*CVOPT
      C     SCALAR
            B(5,1,IB)=B(5,1,IB)+BU*CV
            OMEGAE(5,1,IB,1)=OMEGAE(5,1,IB,1)+GBU*CV
            B(IV,2,IB)=B(IV,2,IB)+BM*CVOPT
            OMEGAE(IV,3,IB,1)=OMEGAE(IV,3,IB,1)+GBU*BMOD*CVOPT
            B(5,2,IB)=B(5,2,IB)+BM*CV
            OMEGAE(5,3,IB,1)=OMEGAE(5,3,IB,1)+GBU*BMOD*CV
      C     SIMILAR SCHEME AS FOR BLUE ABOVE
            CV=CVALS(IV,IB,1)
            IF(TVALON) CVOPT=CV
            RU=BCAS(IV)*RSUM
            GRU=BCAS(IV)*GBNET
            RM=BCAS(IV)*RSCORM
            R(IV,1,IR)=R(IV,1,IR)+RU*CVOPT
            OMEGAE(IV,1,IR,2)=OMEGAE(IV,1,IR,2)+GRU*CVOPT
            R(5,1,IR)=R(5,1,IR)+RU*CV
            OMEGAE(5,1,IR,2)=OMEGAE(5,1,IR,2)+GRU*CV
            R(IV,2,IR)=R(IV,2,IR)+RM*CVOPT
            OMEGAE(IV,3,IR,2)=OMEGAE(IV,3,IR,2)+GRU*RMOD*CVOPT
            R(5,2,IR)=R(5,2,IR)+RM*CV
            OMEGAE(5,3,IR,2)=OMEGAE(5,3,IR,2)+GRU*RMOD*CV
       1800 CONTINUE
      C
            CVR5=CVALS(5,IR,2)
            CVB5=CVALS(5,IB,1)
            DO 1810 I15=KLIM,5,KSTEP
            B(I15,1,IB)=B(I15,1,IB)+BSUM*CVR5
            OMEGAE(I15,1,IB,1)=OMEGAE(I15,1,IB,1)+GRNET*CVR5
            B(I15,2,IB)=B(I15,2,IB)+BSCORM*CVR5
            OMEGAE(I15,3,IB,1)=OMEGAE(I15,3,IB,1)+GRNET*BMOD*CVR5
            R(I15,1,IR)=R(I15,1,IR)+RSUM*CVB5
            OMEGAE(I15,1,IR,2)=OMEGAE(I15,1,IR,2)+GBNET*CVB5
            R(I15,2,IR)=R(I15,2,IR)+RSCORM*CVB5
            OMEGAE(I15,3,IR,2)=OMEGAE(I15,3,IR,2)+GBNET*RMOD*CVB5
       1810 CONTINUE
      C
      20    CONTINUE
      10    CONTINUE
      C
            F=FRACLF
            XND=ND
      C
      C       FILL GLOBAL ARRAY FOR CIPS, DIRECT FIRE...AND
      C       NORMALIZE SCORES TO FRACTIONAL LIVES
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 7 of 15 pages)

```
      DO 2800 IS=1,2
      DD=DIVS(IS)
      DO 2700 IW=1,LIMD
      GZLOS=ZLOSS(IW,IS)
      IF(GZLOS.LE.0.0) GZLOS=1.0
      GZ=F/GZLOS
      DO 2600 IV=1,5
      OMEGAE(IV,2,IW,IS)=OMEGAE(IV,1,IW,IS)*GZ
      OMEGAE(IV,1,IW,IS)=OMEGAE(IV,1,IW,IS)*GZ*START(IW,IS)/DD
      OMEGAE(IV,4,IW,IS)=OMEGAE(IV,3,IW,IS)*GZ
      OMEGAE(IV,3,IW,IS)=OMEGAE(IV,3,IW,IS)*GZ*START(IW,IS)/DD
 2600 CONTINUE
 2700 CONTINUE
 2800 CONTINUE
C
C         FILL GLOBAL ARRAY FOR INDIRECT FIRE
C
      DO 2890 IS=1,2
      DO 2880 IW=LIMD+1,NTYPS(IS)
      IIW=IW-LIMD
      IF(IS.EQ.1) THEN
         GZLOS=BAA(IIW)
         GAA=BAA(IIW)/DIVS(1)
      ELSE
         GZLOS=RAA(IIW)
         GAA=RAA(IIW)/DIVS(2)
      ENDIF
      IF(GZLOS.LE.0.0) GZLOS=1.0
      GZ=F/GZLOS
      DO 2870 IV=1,5
      IF(IS.EQ.1) THEN
         ZG=B(IV,1,IW)*GZ
         OMEGAE(IV,1,IW,1)=ZG*GAA
         OMEGAE(IV,2,IW,1)=ZG
         ZG=B(IV,2,IW)*GZ
         OMEGAE(IV,3,IW,1)=ZG*GAA
         OMEGAE(IV,4,IW,1)=ZG
      ELSE
         ZG=R(IV,1,IW)*GZ
         OMEGAE(IV,1,IW,2)=ZG*GAA
         OMEGAE(IV,2,IW,2)=ZG
         ZG=R(IV,2,IW)*GZ
         OMEGAE(IV,3,IW,2)=ZG*GAA
         OMEGAE(IV,4,IW,2)=ZG
      ENDIF
 2870 CONTINUE
 2880 CONTINUE
 2890 CONTINUE
C
C
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 8 of 15 pages)

```
C       FILL ARRAY FOR COPS
C
        DO 3000 IB=1,NTYPS(1)
C     SKIP IF A NON-DIVISIONAL BLUE WEAPON
        IF ( BNDW(IB) ) GOTO 3000
        FUZZ=1.0
        IF(IB.GT.LIMD) THEN
         BALX=BAL(IB-LIMD)
         IF(BALX.LE.0.5) BALX=1.0
         FUZZ=XND*BAA(IB-LIMD)/BALX
        ENDIF
        DO 2900 IV=1,5
        BT(IV,1)=BT(IV,1)+B(IV,1,IB)*FUZZ*FRAX(IB,1)
        BT(IV,2)=BT(IV,2)+B(IV,2,IB)*FUZZ*FRAX(IB,1)
 2900   CONTINUE
 3000   CONTINUE
        DO 4000 IR=1,NTYPS(2)
C     SKIP IF A RED NON-DIVISIONAL TYPE
        IF ( RNDW(IR) ) GOTO 4000
        FUZZ=1.0
        IF(IR.GT.LIMD) THEN
         RALX=RAL(IR-LIMD)
         IF(RALX.LE.0.5) RALX=1.0
         FUZZ=XND*RAA(IR-LIMD)/RALX
        ENDIF
        DO 3900 IV=1,5
        RT(IV,1)=RT(IV,1)+R(IV,1,IR)*FUZZ*FRAX(IR,2)
        RT(IV,2)=RT(IV,2)+R(IV,2,IR)*FUZZ*FRAX(IR,2)
 3900   CONTINUE
 4000   CONTINUE
C
C         FILL GLOBAL ARRAY FOR COPS--DIRECT & INDIRECT FIRE
C
        DO 3760 IS=1,2
        DO 3750 IW=1,NTYPS(IS)
        IF((IS.EQ.1).AND.BNDW(IW)) GOTO 3750
        IF((IS.EQ.2).AND.RNDW(IW)) GOTO 3750
        DO 3740 IV=1,5
        OMEGAC(IV,1,IS)=OMEGAC(IV,1,IS)+OMEGAE(IV,1,IW,IS)*
      1  FRAX(IW,IS)
        OMEGAC(IV,2,IS)=OMEGAC(IV,2,IS)+OMEGAF(IV,3,IW,IS)*
      1  FRAX(IW,IS)
 3740   CONTINUE
 3750   CONTINUE
 3760   CONTINUE
C
C       OUTPUT RECORDS TO MINFO
C
C        FACX=F/XND
C        DO 5000 IB=1,NTYPS(1)
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 9 of 15 pages)

```
C          FRX=FRAX(IB,1)
C          FAC=FACX
C          FAC1=ABTOT(IB)
CC         IF INDIR WPN, SET GLOBAL ALLOC (BAA) & LOSS (BAL)
C          IF(IB.GT.LIMD) THEN
C           FAC1=1.0
C           BALX=BAL(IB-LIMD)
C           IF(BALX.LE.0.5) BALX=1.0
C           FAC=BAA(IB-LIMD)*F/BALX
C          ENDIF
C          IF(FAC1.LE.0.01) GOTO 5000
C          FAC=FAC*FRX/DIVS(1)
CC         UNMOD BLUE SCORES FOR WPN
C          CALL FILL(1,IB,USCOR,1,B,FAC)
C          CALL OUTREC(10,IB,B(1,1,IB),FAC)
CC         UNMOD BLUE CIP FOR WPN
C          FAC2=F/FAC1
C          IF(IB.GT.LIMD) FAC2=F/BALX
C          FAC2=FAC2*FRX
C          CALL FILL(1,IB,UCIP,1,B,FAC2)
C          CALL OUTREC(30,IB,B(1,1,IB),FAC2)
CC         MOD BLUE SCORES FOR WPN
C          CALL FILL(1,IB,MSCOR,2,B,FAC)
C          CALL OUTREC(50,IB,B(1,2,IB),FAC)
CC         MOD BLUE CIP FOR WPN
C          CALL FILL(1,IB,MCIP,2,B,FAC2)
C          CALL OUTREC(70,IB,B(1,2,IB),FAC2)
C   5000   CONTINUE
C          DO 5100 IR=1,NTYPS(2)
C          FRX=FRAX(IR,2)
C          FAC=FACX
C          FAC1=ARTOT(IR)
CC         IF INDIR WPN, SET GLOBAL ALLOC (RAA) & LOSS (RAL)
C          IF(IR.GT.LIMD) THEN
C           FAC1=1.0
C           RALX=RAL(IR-LIMD)
C           IF(RALX.LE.0.5) RALX=1.0
C           FAC=RAA(IR-LIMD)*F/RALX
C          ENDIF
C          IF(FAC1.LE.0.01) GOTO 5100
C          FAC=FAC*FRX/DIVS(2)
CC         UNMOD RED SCORES FOR WPN
C          CALL FILL(2,IR,USCOR,1,R,FAC)
C          CALL OUTREC(20,IR,R(1,1,IR),FAC)
CC         UNMOD RED CIP FOR WPN
C          FAC2=F/FAC1
C          IF(IR.GT.LIMD) FAC2=F/RALX
C          FAC2=FAC2*FRX
C          CALL FILL(2,IR,UCIP,1,R,FAC2)
C          CALL OUTREC(40,IR,R(1,1,IR),FAC2)
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 10 of 15 pages)

```
CC      MOD RED SCORES FOR WPN
C       CALL FILL(2,IR,MSCOR,2,R,FAC)
C       CALL OUTREC(60,IR,R(1,2,IR),FAC)
CC      MOD RED CIP FOR WPN
C       CALL FILL(2,IR,MCIP,2,R,FAC2)
C       CALL OUTREC(80,IR,R(1,2,IR),FAC2)
C 5100  CONTINUE
C       FACB=FACX/DIVS(1)
CC      UNMOD BLUE COP
C       CALL OUTREC(11,0,BT(1,1),FACB)
CC      MOD BLUE COP
C       CALL OUTREC(51,0,BT(1,2),FACB)
C       FACR=FACX/DIVS(2)
CC      UNMOD RED COP
C       CALL OUTREC(21,C,RT(1,1),FACR)
CC      MOD RED COP
C       CALL OUTREC(61,0,RT(1,2),FACR)
C
C         OUTPUT GLOBAL SCORES,CIPS, & COPS
C            1. SCORES & CIPS
        DO 5190 IS=1,2
        DO 5180 IW=1,NTYPS(IS)
        DO 5170 IREC=1,4
        CALL OUTGLO(ISNO(IREC,IS),IW,OMEGAE(1,IREC,IW,IS),FRAX(IW,IS))
 5170   CONTINUE
 5180   CONTINUE
 5190   CONTINUE
C            2. COPS
C               A. BLUE UCOP
        CALL OUTGLO(11,0,OMEGAC(1,1,1),1.0)
C               B. BLUE MCOP
        CALL OUTGLO(51,0,OMEGAC(1,2,1),1.0)
C               C. RED UCOP
        CALL OUTGLO(21,0,OMEGAC(1,1,2),1.0)
C               D. RED MCOP
        CALL OUTGLO(61,0,OMEGAC(1,2,2),1.0)
C
C         OUTPUT RAW KILLS & LOSSES
C
        DO 6000 IS=1,2
        DD=DIVS(IS)
        DO 5800 IW=1,LIMD
        IF(START(IW,IS).LE.0.01) GOTO 5800
        GZLOS=ZLOSS(IW,IS)
        IF(GZLOS.LE.0.01) GZLOS=1.0
        GZ=(GZLOS*DD)/(FRACLF*START(IW,IS))
        CALL OUTRAW(IS,IW,OMEGAE(1,1,IW,IS),GZ,ZLOSS(IW,IS))
 5800   CONTINUE
        DO 5900 IW=LIMD+1,NTYPS(IS)
        IF((IS.EQ.1).AND.(BAA(IW-LIMD).LE.0.01)) GOTO 5900
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 11 of 15 pages)

```
            IF((IS.EQ.2).AND.(RAA(IW-LIMD).LE.0.01)) GOTO 5900
            IF(IS.EQ.1) THEN
              GZLOS=BAL(IW-LIMD)
            ELSE
              GZLOS=RAL(IW-LIMD)
            ENDIF
            GZ=DD/FRACLF
            CALL OUTRAW(IS,IW,OMEGAE(1,1,IW,IS),GZ,GZLOS)
      5900  CONTINUE
      6000  CONTINUE
C
C
C            DO 5200 IC=1,ICIPTP
C              UCOP(1,IC)=BT(IC,1)*FACB
C              UCOP(2,IC)=RT(IC,1)*FACR
C5200        CONTINUE
C            DO 5300 IC=1,ICIPTP
C              MCOP(1,IC)=BT(IC,2)*FACB
C              MCOP(2,IC)=RT(IC,2)*FACR
C5300        CONTINUE
C            SIZES(1)=M
C            SIZES(2)=N
C            IOPTRB=1
C            IOPTRB=IOPTRB+PREF4*ICIPTP
C            WRITE(IU11'IOPTRB,10000,ERR=20000)
C         &      (((UCIP(I,J,K),K=1,ICIPTP),J=1,SIZES(I)),I=1,2)
C10000      FORMAT(500(500F10.3))
C            IOPTRB=2
C            IOPTRB=IOPTRB+PREF4*ICIPTP
C            WRITE(IU11'IOPTRB,10000,ERR=20000)
C         &      (((USCOR(I,J,K),K=1,ICIPTP),J=1,SIZES(I)),I=1,2)
C            IOPTRB=4
C            IOPTRB=IOPTRB+PREF4*ICIPTP
C            WRITE(IU11'IOPTRB,10000,ERR=20000)
C         &      ((UCOP(I,K),K=1,ICIPTP),I=1,2)
C            GOTO 20200
C20000      CONTINUE
C            WRITE(6,20100)IOPTRB
C20100      FORMAT(1X,30(1H*),5X,'UNMODIFIED 4-VECTOR WRITE ERROR'
C         &     ,1X,'IOPTRB= ',I1,5X,30(1H*))
C            STOP
C20200      CONTINUE
C            IOPTRC=1
C            IOPTRC=IOPTRC+PREF4*ICIPTP
C            WRITE(IU12'IOPTRC,10000,ERR=30000)
C         &      (((MCIP(I,J,K),K=1,ICIPTP),J=1,SIZES(I)),I=1,2)
C            IOPTRC=2
C            IOPTRC=IOPTRC+PREF4*ICIPTP
C            WRITE(IU12'IOPTRC,10000,ERR=30000)
C         &      (((MSCOR(I,J,K),K=1,ICIPTP),J=1,SIZES(I)),I=1,2)
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 12 of 15 pages)

```
C         IOPTRC=4
C         IOPTRC=IOPTRC+PREF4*ICIPTP
C         WRITE(IU12'IOPTRC,10000,ERR=30000)
C       &      ((MCOP(I,K),K=1,ICIPTP),I=1,2)
C         GOTO 30200
C30000    CONTINUE
C         WRITE(6,30100)IOPTRC
C30100    FORMAT(1X,30(1H*),5X,'MODIFIED 4-VECTOR WRITE ERROR'
C       &      ,1X,'IOPTRC= ',I1,5X,30(1H*))
C         STOP
C30200    CONTINUE
          RETURN
C       DEBUG SUBCHK,INIT,SUBTRACE
C       AT 2
C       TRACE ON
        END
C
C
C         SUBROUTINE OUTREC(ISCNT,IDW,R,FACTOR)
CC
C         DIMENSION R(5),S(5)
C         CHARACTER*3 KTHTR
C         COMMON/AWRK/IWRK,IDUM,JTHTR,JTPD,JVIS,JPOS,JDAY
C         COMMON/GLOBAL/IBFOR,IRFOR,JCASE
C         COMMON/CHAR/KTHTR
CC
C         DO 50 I=1,5
C         S(I)=R(I)*FACTOR
C     50  CONTINUE
C         DO 100 I=1,5
C         IF(S(I).GT.0.0) GOTO 200
C    100  CONTINUE
C         RETURN
CC
C    200  WRITE(IWRK,1)ISCNT,KTHTR,JTPD,JVIS,JPOS,JDAY,IDW,
C       1 S(1),S(2),S(3),S(4),S(5),IBFOR,IRFOR,JCASE
C         RETURN
CC
C      1  FORMAT(I5,A3,I4,3I3,I5,5F10.3,2I6,I5)
C         END
CC
C         SUBROUTINE FILL(I,J,A,K,B,FAC)
C         DIMENSION A(2,60,4),B(5,2,60)
CC
CC     I = SIDE ID
CC     J = WEAPON ID
CC     A = ARRAY TO BE FILLED
CC     K = 1 IF UNMODIFIED AND 2 IF MODIFIED
CC     B = ARRAY CONTAINING NUMBER
CC     FAC = MULTIPLICATION FACTOR
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 13 of 15 pages)

```
CC
C        DO 100 IC=1,4
C           A(I,J,IC)=B(IC,K,J)*FAC
C100     CONTINUE
C        RETURN
C        END
C
       SUBROUTINE ZERO(X,N)
       DIMENSION X(N)
       DO 100 I=1,N
       X(I)=0.0
   100 CONTINUE
       RETURN
       END
C
       SUBROUTINE OUTGLO(ISCNT,IDW,R,FRX)
       DIMENSION R(5),S(5)
       CHARACTER*3 KTHTR
       COMMON/AWRK/IWRK,IDUM,JTHTR,JTPD,JVIS,JPOS,JDAY
       COMMON/GLOBAL/IBFOR,IRFOR,JCASE
       COMMON/CHAR/KTHTR
C
       DO 50 I=1,5
       IF(R(I).GT.0.0) GOTO 100
    50 CONTINUE
       RETURN
C
   100 DO 200 I=1,5
       S(I)=FRX*R(I)
   200 CONTINUE
C
       WRITE(29,1) ISCNT,KTHTR,JTPD,JVIS,JPOS,JDAY,IDW,
      1 S(1),S(2),S(3),S(4),S(5),IBFOR,IRFOR,JCASE
       RETURN
C
     1 FORMAT(I5,A3,I4,3I3,I5,5F10.3,2I6,I5)
       END
C
       SUBROUTINE GETCAT
       COMMON/CATVAL/CVALS(5,60,2)
C
       CALL ZERO(CVALS,600)
C
   100 READ(5,1,END=1000) ISIDE,IDW,(CVALS(J,IDW,ISIDE),J=
      1 1,5)
       GOTO 100
     1 FORMAT(2I3,5F7.3)
C
  1000 RETURN
C
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 14 of 15 pages)

```
      END
C
      SUBROUTINE OUTRAW(IS,IW,R,GZ,GZLOS)
      DIMENSION R(5),S(5)
C
      DO 100 I=1,5
      S(I)=R(I)*GZ
  100 CONTINUE
C
      WRITE(6,1) IS,IW,(S(I),I=1,5),GZLOS
      RETURN
C
    1 FORMAT(' SIDE=',I2,' IDWPN=',I3,' KILLS=',5F10.3,
     1 '   LOSSES=',F10.3)
C
      END
C
      SUBROUTINE GETFRX
      COMMON/FRACTS/FRAX(60,2)
C
      CALL ZERO(FRAX,120)
C
  100 READ(5,1,END=1000) IW,FRAX(IW,1),FRAX(IW,2)
      GOTO 100
    1 FORMAT()
C
 1000 RETURN
C
      END
C
      SUBROUTINE OUTPAR
      COMMON/AWRK/IWRK,ISCNT,JTHTR,JTPD,JVIS,JPOS,JDAY
      COMMON/GLOBAL/IBFOR,IRFOR,JCASE
      COMMON/CHAR/KTHTR
      CHARACTER*3 KTHTR
      READ(5,1) ICOMBO,JTPD,KTHTR,JVIS,JDAY,JPOS,IBFOR,IRFOR
      RETURN
    1 FORMAT(2I3,A3,3I3,2I6)
C
      END
```

Figure F-9. Source Listing of the Main Subprogram (TARTY in Element GGGTARTY) and Several Subroutines of the CBT/CS/CSS Merge Module (Page 15 of 15 pages)

(e) **BNDW(60).** The logical array BNDW() identifies Blue weapons as nondivisional assets as appropriate. The array is initialized to .FALSE. throughout. Then, depending on input to array BNDX() described in (4) immediately above, elements of BNDW() are reset to .TRUE. for those Blue weapons earmarked as nondivisional. Array RNDW() is the Red counterpart of Blue array BNDW(). An array element BNDW(I) is indexed I=1 to 60 for the 60 Blue weapons types.

(f) **RNDX(60).** The array RNDX() receives and stores the weapon indices of any Red weapons which are nondivisional. If there are only N nondivisional Red weapon types, then only the first N elements of RNDX() are filled. For example, during AFP system development, only Red weapon type 41 was nondivisional. Hence, N=1, and RNDX(1)=41. As a result of entries in RNDX(), the program sets the corresponding logical value in the separate array RNDW() to .TRUE. for nondivisional weapons. Array BNDX() is the Blue counterpart of Red array RNDX().

(g) **RNDW(60).** The logical array RNDW() identifies Red weapons as nondivisional assets as appropriate. The array is initialized to .FALSE. throughout. Then, depending on input to array RNDX() described in (6) immediately above, corresponding elements of RNDW() are reset to .TRUE. for those Red weapons earmarked as nondivisional. Array BNDW() is the Blue counterpart of Red array RNDW(). An array element RNDW(I) is indexed I=1 to 60 for the 60 Red weapon types.

(h) **CVALS(5,60,2).** The array CVALS() receives and stores target values described in paragraph F-2.b(1). An array element CVALS(I,J,K) is indexed;

<u>1</u>. I=1 to 5 for the corresponding weapon type's target value in terms of: personnel, light armored vehicle, heavy armored vehicle, aircraft, or anything else.

<u>2</u>. J=1 to 2 for the two sides: Blue=1, Red=2.

<u>3</u>. K=1 to 2 for the two sides: Blue=1, Red=2.

(i) **FRAX(60,2).** The array FRAX() receives and stores lifetime adjustment factors described in paragraph F-2.b(2). An array element FRAX(I,J) is indexed--

<u>1</u>. I=1 to 60 for the 60 weapon types on the corresponding side.

<u>2</u>. J=1 to 2 for the two sides: Blue=1, Red=2.

(j) **DIVS(2).** The array DIVS() receives and stores the number of divisions involved on each side. The array element DIVS(1) is indexed I = 1 to 2 for sides: Blue=1, Red=2.

(k) ISNO(4,2). The array ISNO() stores the record type identifiers that must appear in program output files to assure correct interpretation of output in terms of unmodulated and modulated scores and CIPs. An array element ISNO(I,J) is indexed;

1. I=1 to 4 for: unmodulated score, unmodulated CIP, modulated score, modulated CIP.

2. J=1 to 2 for side: Blue=1, Red=2.

(2) The contents of some small scratch arrays are frequently changed completely during program execution.

(a) BCAS(4). The array BCAS() specifies how many losses are to be charged to each target category for a loss of one of the Blue weapon type currently under examination. Every time program control shifts to a different Blue weapon type, array BCAS() is filled with the values corresponding to the current Blue weapon type. Typically, no more than two elements of BCAS() are nonzero for any one weapon type. Array RCAS() is the Red counterpart of Blue array BCAS(). An element BCAS(I) is indexed I= 1 to 4: personnel lost, light armored vehicles lost, heavy armored vehicles lost, aircraft lost.

(b) RCAS(4). The array RCAS() specifies how many losses are charged to each target category for a loss of one of the Red weapon types currently under examination. Every time program control shifts to a different Red weapon type, array RCAS() is filled with the values corresponding to the current Red weapon type. Typically, no more than two elements of RCAS() are nonzero for any one weapon type. Array BCAS() is the Blue counterpart of Red array RCAS(). An element RCAS(I) is indexed I =1 to 4: personnel lost, light armored vehicles lost, heavy armored vehicles lost, aircraft lost.

(c) ALOSS(14,2,2). The array ALOSS() collects in one place critical information about engagements between a specific Blue weapon type and a specific Red weapon type. Whenever program control shifts to consideration of a different Blue/Red weapon matchup, array ALLOSS() is completely refilled with the corresponding results from the Combat Module. Combat Module results related to both direct fire and indirect fire and entered. An array element ALOSS(I,J,K) is indexed;

1. I=1 to 14 for the type of Combat Module result (some "result" may be a throughput of Combat Module input).

- **I=1.** The number of weapons of the current type allocated to the current type-on-type engagement.

- **I=2.** The number of weapons of the current type participating in the current type-on-type engagement.

- **I=3.** The number of weapons of the current type lost to direct fire by the opposing type.

- **I=4.** The number of weapons of the current type regarded as lost to external causes. Although this value continues to be extracted from Combat Module information, the estimate of external losses applied within the CBT/CS/CSS Merge Module is computed differently.

- **I=5 to 14.** The number of weapons of the current type lost opposing indirect fire weapons of indirect fire weapon type I-4 or general weapon type W = LIMD + I-4, where LIMD = 50, the number of direct fire weapon types.

2. J=1 to 2 for side: Blue=1, Red=2.

3. K=1 to 2 for day. All applications of the AFP System to date have been confined to consideration of 2-days of Combat Module activity. A change to some other number of days, ND, implies that array ALOSS() be redimensioned to ALOSS(14,2,ND).

(3) The principal working arrays for accumulating results may be updated frequently during execution as program control loops over Blue weapon types, Red weapon types, and days.

(a) B(5,2,60). The losses inflicted by Blue weapon types are accumulated in B() in accordance with the LER method of projecting losses. Array R() is the Red counterpart of Blue array B(). An array element B(I,J,K) is indexed:

1. I=1 to 5 for the components of the five-valued form of losses inflicted: personnel, light armored vehicles, heavy armored vehicles, aircraft, and the weighted (in accord with target values) rolled-up scalar.

2. J=1 to 2 for the unmodulated and modulated losses inflicted. AFP terms, the losses are closely related to scores. That is, the losses are those achieved by all weapons of the given type. At this stage, it is not necessary to accumulate CIPs. The calculation of CIPs can occur much later simply by division of the scores by the numbers of weapons subject to adjustment for lifetime factors. In other words, it is not necessary for the index J to vary from 1 to 4.

3. K=1 to 60 for the 60 different Blue weapon types permitted.

(b) R(5,2,60). The losses inflicted by Red weapon types are accumulated in R() in accord with the LER method of projecting losses. Array B() is the Blue counterpart of Red array R(). An array element R(I,J,K) is indexed similarly to the scheme described for B() in F-5.c.(3)(a) above with the obvious difference that the K index for R() references a Red weapon type.

(c) BT(5,2). The array BT() serves as a Blue accumulator array for losses inflicted by Blue weapons over all Blue weapon types in the determination of Blue COPs in accord with the LER method of projecting losses. BT(I,J) may be considered as the result of summing B(I,J,K) with respect to all 60 values of index K. Array RT() is the Red counterpart of Blue array BT(). The array element BT(I,J) is indexed in accord with the scheme described for the I and J indices under (a) above.

(d) RT(5,2). The array RT() serves as a Red accumulator array for losses inflicted by Red weapons over all Red weapon types in the determination of Red COPs in accord with the LER method of projecting losses. RT(I,J) may be considered as the result of summing R (I,J,K) with respect to all 60 of the values of index K. Array BT() is the Blue counterpart of array RT(). The array element RT(I,J) is indexed in accord with the scheme described for the I and J indices under (a) above.

(e) BAL(10). The array BAL() serves as a Blue accumulator array for losses suffered by Blue mortar and artillery weapon types. In the current standard version of the AFP System, both Blue and Red inventories may include 10 types of indirect firers as weapon types 51-60. The first 50 weapon types on each side are considered direct fire weapons only; this program limit is represented by the constant LIMD = 50. Hence, the Ith indirect fire weapon type is the (LIMD + Ith) weapon type within the full weapon sequence of types 1-60. Array RAL() is the Red counterpart of Blue array BAL().

(f) RAL(10). The array RAL() serves as a Red accumulator array for losses suffered by Red mortar and artillery weapon types. Array RAL() is the Red counterpart of Blue array BAL(), and the remarks about BAL in F-5.c.(3)(e) immediately above with respect to the relation of weapon type indexing apply to RAL() as well.

(g) ABTOT(60). The array ABTOT() serves as an accumulator of Blue weapon allocations by type over all days. Array ARTOT() is the Red counterpart of Blue array ABTOT(). An element of array ABTOT(I) is indexed I=1 to 60 for the 60 different Blue weapon types.

(h) ARTOT(60). The array ARTOT() serves as an accumulator of Red weapon allocations by type over all days. Array ABTOT() is the Blue counterpart of Red array ARTOT(). An element of array ARTOT(I) is indexed I=1 to 60 for the 60 different Red weapon types.

(i) START(60,2). The array START() serves as an accumulator of Blue and Red weapon allocations by type only for the first day represented within the AFP Combat Module. The information accumulated within START() is used in accord with the GER method of estimating partial combat potentials. An array element START(I,J) is indexed:

1. I=1 to 60 for the 60 weapon types on the corresponding side.

2. J=1 to 2 for side: Blue=1, Red=2.

(j) OMEGAE(5,4,60,2). The array OMEGAE() accumulates information on losses inflicted with later modification by the losses suffered as accumulated in another array, ZLOSS(). The array OMEGAE() is the principal array used in the develop of scores and CIPs in accord with the GER method of estimating partial combat potentials. An array element OMEGAE(I,J,K,L) is indexed;

1. I=1 to 5 for the five components of five-valued potentials: personnel, light armored vehicles, heavy armored vehicles, aircraft, and weighted (in accord with CVALS() target values) scalar.

2. J=1 to 4 for: unmodulated score, unmodulated CIP, modulated score, modulated CIP. All these are partial in the sense of relating to only one combat environment.

3. K=1 to 60 for the 60 weapon types on the corresponding side.

4. L=1 to 2 for side: Blue=1, Red=2.

(k) OMEGAC(5,2,2). The array OMEGAC accumulates information in the determination of COPs in accord with the GER method of estimating partial combat potentials. An array element OMEGAC(I,J,K) is indexed:

1. I=1 to 5 for the five components of five-valued potentials: personnel, light armored vehicles, heavy armored vehicles, aircraft, and weighted (in accord with CVALS() target values) scalar.

2. J=1 to 2 for: unmodulated COP, modulated COP.

3. K=1 to 2 for side: Blue=1, Red=2.

(l) ZLOSS(60,2). The array ZLOSS() accumulates the losses suffered to both direct and indirect fire for use in the determination of partial combat potentials in accord with the GER method. An array element ZLOSS(I,J) is indexed;

1. I=1 to 60 for the 60 weapon types suffering losses on a corresponding side.

2. J=1 to 2 for the side suffering the loss: Blue=1, Red=2.

(4) Several arrays were defined for use in a currently abandoned method for rolling up partial combat potentials over the 16 combat environments. Although the arrays are retained and updated within the current version of the program, the arrays need not be mentioned here beyond their identification:

(a) USCOR(2,60,4).

(b) MSCOR(2,60,4).

(c) UCIP(2,60,4).

(d) MCIP(2,60,4).

(e) UCOP(2,4).

(f) MCOP(2,4).

d. The GGGTARTY source listing in Figure F-9 includes some intralinear comments. The following paragraphs provide some additional commentary.

(1) Lines 17 through 75 provide the needed declarative statements, many to establish the arrays described above. Some scalar character and logical variables are also declared. Many lines begin with "C." Some such lines are ordinary comments, but many others contain deactivated code.

(2) Lines 77 through 84 initialize a number of arrays and variables with values and identifiers needed throughout processing.

(3) Line 90 begins the executable statements. Line 90 calls subprogram OUTPAR to read the values of several indices and parameters to be included in the module's output records of partial combat potentials. Lines 91 through 103 initialize several variables. TVALON controls whether target values are to be included in the output partial combat potentials' first four elements and whether other weapons (e.g., small arms and SAMs) are to be counted (at their target values) with personnel.

(4) Lines 104 through 106 zero some of the accumulator arrays.

(5) Line 110 reads several case and combat environment identifiers against which records in the CS/CSS moduli file will later be checked. It is essential that the values read at this point from the runstream be identical to the values that appear in the CS/CSS file. However, in current AFP practice, some values used have no particular significance. As noted elsewhere, most AFP data control, relative to division type or period and to combat environment, is provided at the higher level of file name conventions.

(6) Line 113, largely for reference, computes the combat environment index (1-16) from the visibility (clear, degraded), day (daytime, nighttime), and posture (defense intense, defense light, delay, attack) indices just read. If the actual indices are included in both the runstream and the CS/CSS moduli file, the following logic will apply a meaningful, precise test.

(7) Lines 119 and 120 read a single record from the CS/CSS moduli file. An end-of-file transfers control to just beyond the CS/CSS moduli input sequence.

(8) Lines 122 through 130 apply the test to indices within a record from the CS/CSS moduli file, and, provided that the record satisfies the test, store the moduli with the corresponding pair of elements within array CSCSS().

(9) Line 134 reads and stores the numbers of Blue and Red divisions given in the runstream. The numbers should be equal to the numbers of divisions implied by the weapon inventories as they were input to the Combat Module. Recall that the Combat Module "knows" inventory quantities but never "knows" that those quantities are equivalent to NBDIV and NRDIV Blue and Red divisions, respectively. The CBT/CS/CSS Merge Module needs the numbers of divisions so that later it can normalize total achievements to the partial potentials "per division's worth."

(10) Lines 138 and 140 read the identifying numbers of the nondivisional weapon types within the Blue and Red inventories.

(11) Lines 142 through 151 zero a number of working and accumulator arrays.

(12) Lines 153 through 158 use the previously read identifiers of nondivisional weapon types to set the corresponding nondivisional program flags to .TRUE. within arrays BNDW() and RNDW().

(13) Line 161 calls subroutine GETCAT to reads weapon target values into the common array CVALS().

(14) Line 165 calls subroutine GETFRX to read fractional lifetime factors in common array FRAX().

(15) Line 168 calls subroutine GETART to obtain the starting strengths of Blue and Red mortar and artillery weapons.

(a) Argument BAA is the address of array BAA() where Blue starting strengths are to be stored for later reference.

(b) Argument RAA is the address of array RAA() where Red starting strengths are to be stored for later reference.

(c) Argument LIMD is the number of direct fire weapon types in each of the Blue and Red inventories. The direct fire weapon types are always the first LIMD of all the weapon types. In all AFP work to date, there have been 50 direct fire weapon types; hence, LIMD = 50, and the 10 indirect fire weapon types have been the 51st through 60th among all weapon types.

(d) Argument M is the total number of Blue weapon types. In all work to date, M=60.

(e) Argument N is the total number of Red weapon types. In all work to date, N=60.

(16) Lines 173 through 344 define the limits of the major processing loop within GGGTARTY. All possible direct fire engagements are processed with this, the outer loop, being over all the Blue weapon types = NTYPS(1) = 60. (The next inner loop is over all the Red weapon types. And within that loop is yet another loop over Combat Module days.)

(17) Line 175 sets the scratch variable EXB to the external loss factor appropriate to Blue weapon type IB by means of a function reference to EXTLOS.

(a) Argument "1" identifies the side as Blue.

(b) Argument IB is the index of the current Blue weapon type.

(18) Lines 177 and 343 define the outer limits of the program loop over all Red weapon types = NTYPS(2) = 60. Because this loop lies inside the Blue weapon type loop, all Red weapon types are checked for each Blue weapon type. At this stage, both Blue and Red weapon type indices are defined. Hence, the program now must examine the type-on-type engagements between Blue IB and Red IR weapons.

(19) Line 179 calls subroutine GETCA to fill scratch array BCAS() with how many losses are to be charged to each target category for a loss of one of Blue weapon type IB.

(a) Argument "2" is the target side, Blue.

(b) Argument IR is the index of the Red shooting weapon type.

(c) Argument IB is the index of the current Blue target weapon type.

(d) Argument BCAS is the address of the scratch array to be filled.

(20) Line 181 calls subroutine GETCA to fill scratch array RCAS() with how many losses are to be charged to each target category for a loss of one of Red weapon type IR.

(a) Argument "1" is the target side, Red.

(b) Argument IB is the index of the Blue shooting weapon type.

(c) Argument IR is the index of the current Red target weapon type.

(d) Argument RCAS is the address of the scratch array to be filled.

(21) Line 183 sets the scratch variable EXR to the external loss factor appropriate to Red weapon type IR by means of a function reference to EXTLOS.

(a) Argument "2" identifies the side as Red.

(b) Argument IR is the index of the current Red weapon type.

(22) Line 185 calls subroutine GETLOS to fill the scratch array ALOSS() with Combat Module results for the engagements between Blue weapon type IB and Red weapon type IR.

(a) Argument IB is the Blue weapon type.

(b) Argument IR is the Red weapon type.

(c) Argument ALOSS is the address of the scratch array.

(d) Argument IDIM (always 14 in current AFP work) is the number of pieces of data to be returned for each weapon type for each Combat Module day.

(e) Argument ND (always 2 in current AFP work) is the number of Combat Module days for which data are to be returned.

(23) Lines 187 and 188 put the corresponding CS/CSS moduli into scratch variables BMOD and RMOD.

(24) Lines 190 through 195 zero some scratch variables before entering the day loop.

(25) Lines 196 through 292 define the limits of a loop over the number of days represented in the Combat Module. In all work to date ND = 2.

(26) Lines 198 through 205 put some of the data from scratch array ALOSS() into scratch variables with names intended to have some mnemonic value.

(a) AB and AR are the numbers of Blue and Red weapons allocated to the type IB on type IR engagement at the beginning of Day-ID.

(b) PB and PR are the numbers of Blue and Red weapons participating in the type IB on type IR engagement at the beginning of Day-ID.

(c) DB and DR are the numbers of losses suffered on Day-ID by types IB and IR to direct fire by IR and IB types, respectively. DB and DR must be the raw losses as determined by the Combat Module. Note that, in general, those losses may be the result of duels at as many as six ranges in each of four (the current standard) conflicts on Day-ID.

(27) Lines 207 and 208 set scratch variables EB and ER to the external losses to weapon types IB and IR on Day-ID. If EXB and EXR are 0.0, then there are no external losses.

(28) Lines 210 and 211 zero scratch variables BI and RI before a loop over indirect fire weapon types.

(29) Lines 212 and 248 define the outer limits of a loop involving indirect fire weapons. The loop has two major purposes. The simpler purpose is to total the losses suffered on Day-ID by weapon types IB and IR to indirect fire weapons firing on the type IB on type IR direct fire engagement. The somewhat more involved purpose is to credit the indirect fire weapons with their kills of weapon types IB and IR. Toward this purpose, accumulator arrays must be updated. Some of the updates require application of CS/CSS moduli. Note that the loop index in line 212 is defined over a subset of indexed references to array ALOSS().

(a) Line 213 sets the scratch variable LX to the general weapon index of the corresponding indirect fire weapon types. Both Blue and Red indirect fire weapons of the same index are processed concurrently.

(b) Lines 214 and 215 set scratch variables XB and XR to the losses suffered on Day-ID to Red and Blue indirect firers of type LX from the corresponding elements of ALOSS(). If both XB and XR are less than 1.0, line 216 jumps to the end of the indirect fire loop.

(c) Lines 217 and 218 update the scratch summation variables BI and RI by adding the losses on Day-ID inflicted by the currently considered indirect fire weapon types LX. When the indirect fire loop is finally exited, BI and RI will contain the total losses inflicted by indirect fire on weapon types IB and IR on Day-ID during the direct fire engagement between types IB and IR.

(d) Lines 219 through 222 set four scratch variables to the CS/CSS moduli corresponding to the pairings of the currently considered direct and indirect fire weapons. The moduli are needed in the next steps where the kills inflicted by the indirect fire weapons must be credited to the indirect fire weapons.

(e) Lines 224 and 241 define the limits of a loop in which the kills inflicted by the indirect fire weapons will be credited to those weapons in the appropriate target categories (i.e., components of five-valued partial combat potentials--including the weighted scalar component). The loop treats both Blue and Red, unmodulated and modulated, and "local" and "global" exchange ratio methods. (For indirect fire weapons, there is no distinction between "local" and "global" exchange ratio methods.)

1. Line 225 sets scratch variable CV to the target category (personnel, light armored vehicles, heavy armored vehicles, and aircraft) weight in category IV corresponding to a kill of a Red weapon type IR. Line 226 sets CVOPT TO CV IF TVALON=.TRUE., i.e., if target values are to be included in the first four elements of potentials.

2. Line 227 sets scratch variable BU to the number of kills to be credited to Blue indirect fire weapon type LX in target category IV for XR kills of Red weapon type IR.

3. Line 228 updates the Blue accumulator array B() by adding the CVOPT-weighted unmodulated kills of Red type IR to the credit of Blue indirect fire weapon type LX in target category IV. CVOPT = 1.0 if TVALON=.FALSE.; otherwise, CVOPT = target value.

4. Line 229 updates the Blue accumulator array B() by adding in the weighted credit for kills of Red type IR to the scalar component B (5,.,.).

5. Line 230 sets scratch variable BM to the modulated kills of Red type IR.

6. Line 231 updates the Blue accumulator array B() by adding the modulated kills of Red type IR to the credit of blue indirect fire weapon type LX in target category IV. See (29)(e)1. and 3. above for comments about CVOPT.

7. Line 232 updates the Blue accumulator array B() by adding in the weighted credit for kills of Red type IR to the scalar component B(5,.,.).

8. Lines 233 through 240 apply the logic of lines 225 through 232 to give the corresponding credit to Red indirect fire weapon type LX for kills of Blue weapon type IB. Of course, for the Red weapon, the appropriate elements of Red accumulator array R() are updated.

9. Lines 242 through 247 update the Blue and Red accumulator arrays B() and R() for the indirect fire weapons of type LX in those cases where the above kills do not fall in the normal target categories: light armored vehicles, heavy armored vehicles, or aircraft. Elements of the form CVALS(5,.,.) represent the target category weights or values for targets which do not fall into the normal target categories--e.g., dismounted machineguns. The loop parameters were set in accord with TVALON in lines 93 through 98. If TVALON=.TRUE., the elements of the form (CVALS(5,.,.) also affect the first elements (originally personnel only) of combat potential.

(30) At this point, attention shifts from the indirect fire updates back to updating for the direct fire weapons of types IB and IR involved in the direct fire engagement. One possibly confusing aspect of the Combat Module and the treatment of its results here is when mortars and artillery engage one another in counterbattery duels. Those counterbattery actions are considered direct fire engagments. But just as for any other direct fire engagements, some indirect fire falls on the dueling weapons. Recall that part of the input to the Combat Module splits the total inventories of mortars and artillery between indirect and direct (counterbattery) roles. Hence, while the fraction of mortars and artillery devoted to counterbattery roles may be firing at one another, the fraction of the same type weapons devoted to indirect fire may be firing on the counterbattery duels.

(31) Lines 250 and 251 reset scratch variables AB and AR, the numbers of weapons of types IB and IR allocated to the IB-on-IR direct fire duel to 1.0 for any type that is greater than LIMD--i.e., for any mortar or artillery weapon. This step is needed to assure that mortars and artillery be treated correctly in terms of global exchange ratios for both the overall local and global exchange ratio methods or estimating partial combat potentials. The two methods are named in accord with the ways in which the direct fire weapons are treated.

(32) Lines 252 and 253 update the counts of weapons allocated in arrays ABTOT() and ARTOT() for use in the local exchange ratio method.

(33) Lines 254 through 257 update the counts of weapons allocated only on the first day in array START().

(34) The following sequence through line 277 generates the Blue and Red estimates of local exchange ratio for the IB-on-IR pairing on Day-ID. The objective is to estimate a ratio in the form:

(local exchange ratio) = (net target kills)/(total shooter losses)

where:

(a) The net target kills include the targets killed by the direct fire opponents (shooters) plus a pro rata share of the external losses suffered by the targets. The pro rata share of external target losses attributable to the shooter is taken to be:

$$\text{(pro rata external losses)} = \frac{\text{(total external losses) * (direct losses)}}{\text{(direct losses) + (indirect losses)}}$$

(b) The total shooter losses include shooter losses to both direct and indirect fire and the external losses.

(35) Lines 258 and 259 set scratch variables BTOT and RTOT to the sums of direct, external, and indirect losses suffered by weapon types IB and IR on Day-ID, but only with respect to the IB-on-IR type engagement. The losses of types IB and IR in their direct fire engagements with other type weapons are not represented here. These are the losses of types IB and IR considered as shooters.

(36) Lines 260 through 262 leave scratch variable BNET set to the number of losses of Blue type IB considered as a target of Red type IR. Line 260 first sets BNET to the direct losses suffered by Blue type IB. Line 261 sets scratch variable X to the sum of direct and indirect losses suffered by Blue type IB. If Blue type IB did indeed suffer direct or indirect losses, line 262 adds to BNET the pro rata share of external losses suffered by Blue type IB.

(37) Lines 263 through 265 apply logic similar to that described in paragraph (34) immediately above to set scratch variable RNET to the number of losses of Red type IR considered as a target of Blue type IB.

(38) Lines 266 through 270 calculate the local exchange ratio for Blue shooter type IB with Red type IR as a target. An underlying difficulty of the local exchange ratio method surfaces here. If the shooter suffers no direct, indirect, or external losses, straightforward estimation of exchange ratio would involve a division by zero. Hence, the division is performed only if Blue type IB does suffer loss. Otherwise, Blue type IB is implied to have lost one weapon. The difficulty is compounded by the numerous Combat Module engagements involving small numbers of weapons subject to stochastic detection, SSPKs, and refire times. Zero shooter losses are a frequent Combat Module result. Just such practical considerations led to addition of the GER method to the CBT/CS/CSS Merge Module. If the shooter is a mortar or artillery type, the exchange ratio is set to kills without explicit reference to losses, although the effect is as though one loss were assumed at this point.

(39) Lines 272 through 277 calculate the local exchange ratio on Day-ID for Red shooter type IR with Blue type IB as target. The logic is similar to that described in paragraph F-5.d.(36) immediately above for Blue type IB as shooter.

(40) Lines 280 and 281 update scratch variables BSUM and RSUM with Day-ID's weighted estimates of the exchange ratios. The weights are simply the numbers allocated shooters. Hence, the dimension of BSUM is really "Red kills," and the dimension of RSUM is "Blue kills." Neither variable is dimensioned as an exchange ratio after all.

(41) Lines 282 and 283 update the accumulator arrays BAL() and RAL() with the losses of types IB and IR only if the types are mortar or artillery, i.e., greater than LIMD.

(42) Lines 287 and 288 update the accumulator array ZLOSS with the direct, indirect, and external losses of types IB and IR for use in the GER method.

(43) Lines 289 and 290 update the scratch variables GBNET and GRNET with the net losses attributed to the opposing direct fire weapons for use outside the day loop in the GER method.

(44) Line 292 ends the day loop for a specific direct fire pair of weapon types IB and IR.

(45) Lines 293 through 341 update the main accumulator arrays for the direct fire pair of weapon types IB and IR based on the results of ND days of engagements between those two weapon types. Recall that kills scored by indirect fire weapons have already been credited to the indirect weapons within the above loops for some of the same accumulator arrays--but for weapon indices other than IB and IR, of course.

(46) Lines 293 and 294 set scratch variables BSCORM and RSCORM to modulated versions of the scores achieved by types IB and IR in their engagements over the preceding ND days.

(47) Lines 296 and 328 define the outer limits of a loop over the first four components of five-valued partial combat potentials.

(48) Line 297 sets scratch variable CV to the IVth component of target category value of weight for weapon type IR considered as a target. Line 298 sets CVOFT to CV 4 TVALON=.TRUE.

(49) Line 300 sets scratch variable BU to the unmodulated IVth component of the score to be credited to Blue IB in accord with the LER method.

(50) Line 301 sets scratch variable GBU to the unmodulated IVth component of the score to be credited to Blue type IB in accord with the GER method.

(51) Line 303 sets scratch variable BM to the modulated IVth component of the score to be credited to Blue type IB in accord with the LER method.

(52) Line 305 updates the LER accumulator array B() with the increment for the unmodulated IVth component of score earned by Blue type IB in its engagements with Red type IR. CVOPT is 1.0, or as a target value depending on TVALON.

(53) Line 306 updates the GER accumulator array OMEGAE() with the increment for the unmodulated IVth component of score earned by Blue type IB in its engagements with Red type IR.

(54) Lines 308 and 309 update the arrays B() and OMEGAE() for the unmodulated scalar (fifth) components of score.

(55) Lines 310 through 313 complete the updates of arrays B() and OMEGAE() for the modulated components. See comments about CVOPT in (48) and (52) above.

(56) Lines 315 through 327 update the accumulator arrays R() and OMEGAE() for the unmodulated and modulated components of score earned by Red weapon type IR against Blue weapon type IB just as lines 297 through 313, as discussed above, do for Blue weapon type IB against Red weapon type IR.

(57) Lines 330 through 340 complete the update of the scalar (and first if TVALON=.TRUE.) components of accumulator arrays B(), R(), and OMEGAE() for those Red and Blue weapon types which do not belong to the regular light armored vehicle, heavy armored vehicle, or aircraft target categories.

(58) Line 343 bounds the loop over Red weapon types IR considered as direct fire weapons in the type-on-type engagements.

(59) Line 344 bounds the loop over Blue weapon types IB considered as direct fire weapons in the type-on-type engagements.

(60) At this point, all information needed from Combat Module output for this combat environment have been accepted by the CBT/CS/CSS Merge Module. Within the GER method, it remains to divide the contents of accumulator array OMEGAE() by global losses for the corresponding weapon types and to apply fractional lifetime modifiers. It remains to sum the contents of accumulator arrays built at the weapon-type level over weapon-types to compute COP totals. As usual, indirect fire weapons require some special handling. And finally, the contents of the accumulator arrays are fed five values at a time to the output routines for final modification, if appropriate, and for output to the partial combat potentials files.

(61) Line 346 sets scratch variable F to the standard fractional lifetime (0.5) subject to later modification on a weapon-by-weapon basis in accord with the contents of the reference array FRAX().

(62) Line 347 sets scratch variable XND to the number of days represented in the Combat Module.

(63) Lines 351 through 364 bound a triply-nested loop structure that makes all but one final adjustment to unmodulated and modulated scores and CIPs of direct fire weapons in accord with the GER method. The fractional lifetime modifiers are applied later. This loop structure does divide accumulated kills by accumulated losses to provide global exchange ratios. The structure multiplies scores by starting strengths and divides scores by the numbers of divisions. To avoid division by zero, zero losses are set to 1.0, arbitrarily, of course. Looping is perfomed over side, weapon type, and potential component--from outer to inner loops.

(64) Lines 368 through 398 bound a triply-nested loop structure that makes all but one final adjustment to unmodulated and modulated scores and CIPs of indirect fire weapons in accord with the GER method. As above for direct fire weapons, the fractional lifetime modifiers are applied later. Because the GER method applies to artillery in all cases, the mortar and artillery parts of the OMEGAE() array can be constructed from the otherwise LER arrays B() and R() and the availability arrays BAA() and RAA(). Regardless of the losses to indirect fire weapons estimated by the Combat Module, the CBT/CS/CSS Merge Module's GER implies 1.0 losses to each indirect fire weapon type. To avoid division by zero, zero availabilities are set to 1.0. Looping is performed over side, weapon type, and potential component--from outer to inner loops.

(a) Line 370 sets scratch variable to the indirect fire weapon type, which is always the general weapon type, minus LIMD, the number of direct fire weapon types.

(b) Because Blue and Red weapon availabilities are in different arrays BAA() and RAA(), an IF--ENDIF block is applied in lines 381 through 395.

(c) Line 378 guards against subsequent division by zero.

(d) Line 379 sets scratch variable GZ to the standard fractional lifetime (0.5) divided by the weapon availability.

(e) Lines 380 and 396 define the limits of a loop over the five components of five-valued partial combat potentials.

(f) Because Blue and Red weapon kills are in different arrays B() and R(), an IF--ENDIF block is applied in lines 381 through 395.

1. Lines 382 and 389 set scratch variable ZG to the unmodulated kills component from arrays B() and R().

2. Lines 383, 384, 390, and 391 put the unmodulated partial score and CIP components in array OMEGAE().

3. Lines 385 and 392 set scratch variable ZG to the modulated kills component from arrays B() and R().

4. Lines 386, 387, 393, and 394 put the modulated partial score and CIP components in array OMEGAE().

(65) Lines 403 through 430 fill the LER partial COP arrays BT() and RT(). Adjustments for fractional lifetimes are applied as the COP arrays are filled.

(a) Lines 403 and 416 define the bounds of a doubly-nested loop structure for computation of Blue partial COPs. The outer loop is over Blue weapon types; the inner loop is over the five components of five-valued potentials.

(b) Line 405 checks whether the current weapon type is nondivisional. If so, the weapon's results are not included within Blue COP.

(c) Line 406 sets scratch variable FUZZ to 1.0.

(d) Lines 407 through 411 apply only if the Blue weapon is an indirect fire type. In the LER method, a nontrivial global exchange ratio is computed for indirect fire weapons. As usual, to avoid division by zero, zero losses are arbitrarily set to 1.0.

(e) Lines 412 through 415 loop over the five components of five-valued potentials and update the Blue partial COP accumulator array BT() with unmodulated (line 413) and modulated (line 414) increments adjusted by the fractional lifetime modifiers.

(f) Lines 417 through 430 define the bounds of a doubly nested loop structure for computation of Red partial COPs. The structure is similar to that already described for lines 386 through 399 for Blue weapons.

(66) Lines 434 through 445 define the bounds of a triply-nested loop structure for filling the accumulator array OMEGAC() with partial COPs by summation over weapon types from values contained in the weapon accumulator array OMEGAE(). Adjustments for fractional lifetimes are made during the summation. Looping is performed over side, weapon type, and potential component--from outer to inner loops.

(a) Lines 436 and 437 check for nondivisional weapon types. Values corresponding to nondivisional weapons are skipped.

(b) Lines 439 and 440 accumulate the unmodulated components of partial COPs.

(c) Lines 441 and 442 accumulate the modulated components of partial COPs.

(d) Line 445 marks the end of the GER method COP computation.

(67) Lines 449 through 536 output the score, CIP, and COP files for both LER and GER methods. An output routine is called each time an output record is to be output. The routines that output to the LER method file may apply some factor(s) before output. The routines that output to the GER method file make no further modifications prior to output. Both sets of routines do check the components of five-valued potentials. If no component is nonzero, output of a record is suppressed. That is, only weapons with nonzero partial combat potential are included in files output by the CBT/CS/CSS Merge Module.

(a) Deactivated lines 449 through 517 would output to the LER method file.

1. Line 449 sets scratch variable FACX to the standard fractional lifetime divided by the number of days represented in the Combat Module.

2. Lines 450 and 478 define the bounds of a loop over Blue weapon types.

a. Lines 451 sets scratch variable FRX to the fractional lifetime factor corresponding to Blue weapon type IB.

b. Lines 452 and 453 set scratch variables FAC and FAC1 to values used in setting final adjustments, if any, to the contents of the accumulator array B() before output by the output routines. Different modifiers are applied to a single five-vector from array B() to generate a partial score and a partial CIP.

c. Lines 455 through 460 reset scratch variable FAC if the Blue weapon is an indirect fire type. The instructions apply the global losses of Blue type IB. However, if IB suffered no losses, its losses are set to 1.0.

d. Line 461 checks whether any of weapon type IB were avail-able in the first place. If not, there is no need to compute or output anything for this weapon type.

e. Line 462 resets FAC by multiplying its former value by the fractional lifetime factor and dividing by the number of Blue divisions. FAC is now the appropriate modifier for use in generating a partial score from array B().

f. Ignore the lines which call subroutine FILL.

g. Line 465 calls subroutine OUTREC to output a five-valued unmodulated partial score for Blue weapon type IB. Argument "10" identifies the record type. Argument IB is the identifier of the Blue weapon type. Argument B(1,1,IB) is the address of the appropriate five-vector within array B(). Argument FAC is the multiplier that makes the five vector a partial score.

h. Line 467 sets scratch variable FAC2 to the tentative value needed as the multiplier of the same five-vector from B() to convert the five-vector to a partial CIP.

i. But if weapon type IB is an indirect fire weapon, line 468 resets FAC2 to mortar/artillery form.

j. Line 469 resets FAC2 by multiplying the last value by the fractional lifetime factor.

k. Line 471 calls subroutine OUTREC to output a five-valued unmodulated partial CIP for Blue weapon type IB. Argument "30" identifies the record type. Argument IB identifies the weapon type. Argument B(1,1,IB), the same as in the preceding call, is the address of the appropriate five-vector within array B(). Argument FAC2 is the multiplier that makes the five-vector a partial CIP.

l. Line 474 calls subrouting OUTREC to output a five-valued modulated partial score for Blue weapon type IB. Argument "50" identifies the record type. Argument IB identifies the weapon type. Argument B(1,2,IB) is the address of the appropriate five-vector within the array B(). Argument FAC is the multiplier that makes the referenced five-vector a partial score.

m. Line 477 calls subrouting OUTREC to output a five-valued modulated partial CIP for Blue weapon type IB. Argument "70" identifies the record type. Argument IB identifies the Blue weapon type. Argument B(1,2,IB), the same as in the preceding call, is the address of the appropriate five-vector within the array B(). Argument FAC2 is the multiplier that makes the five-vector a partial CIP.

3. Lines 479 and 507 define the bounds of a loop over Red weapon types. The logic is the same, with references to different arrays and with a different identifier, as described for the output of Blue scores and CIP by lines 433 through 461 and as described in 2. immediately above.

4. Lines 508 through 517 output the partial COPs.

a. Line 508 sets scratch variable FACB to the value appropriate as the multiplier of five-vectors in array BT() to make them partial COPs.

b. Line 510 calls subroutine OUTREC to output a five-valued unmodulated partial COP for the Blue division. Argument "11" identifies the record type. Argument "0" identifies the "weapon type" as a division. Argument BT(1,1) is the address of the appropriate five-vector within array BT(). Argument FACB is the multiplier of the five-vector.

c. Line 512 calls subroutine OUTREC to output a five-valued modulated partial COP for the Blue division. Argument "51" identifies the record type. Argument "0" identifies the "weapon type" as a division. Argument BT(1,2) is the address of the appropriate five-vector within array BT(). Argument FACB is the multiplier of the five-vector.

d. Line 513 sets scratch variable FACR to the value appropriate as the multiplier of five-vectors in array RT() to make them partial COPs.

e. Line 515 calls subroutine OUTREC to output a five-valued unmodulated partial COP for the Red division. Argument "21" identifies the record type. Argument "0" identifies the "weapon type" as a division. Argument RT(1,1) is the address of the appropriate five-vector within array RT(). Argument FACR is the multiplier of the five-vector.

f. Line 517 calls subroutine OUTREC to output a five-valued modulated partial COP for the Red division. Argument "61" identifies the record type. Argument "0" identifies the "weapon type" as a division. Argument RT(1,2) is the address of the appropriate five-vector within array R;t(). Argument FACR is the multiplier of the five-vector.

(b) Lines 521 through 536 output to the GER method file of partial scores, CIPs, and COPs.

1. Lines 521 and 527 define the bounds of a triply-nested loop structure for the output of scores and CIPs for Blue and Red weapons. The scores and CIPs without final adjustment for fractional lifetimes already

exist in accumulator array OMEGAE(). Looping is performed over side, weapon type, and record potential type (unmodulated score, unmodulated CIP, modulated score, and modulated CIP) from outer to inner loops). Line 524 calls subroutine OUTGLO to output a five-valued partial potential type.

a. Argument ISNO(IREC,IS) is the identifier of the record type. The identifier depends on both the potential/record type (IREC) and side (IS).

b. Argument IW is the identifier of weapon type.

c. Argument OMEGAE (I,REC,IW,IS) is the address of the appropriate type of five-vector. The five-vector is indexed to its first component (1), the potential/record type (IREC), the weapon type (IW), and the side (IS).

d. Argument FRAX(IW,IS) is the fractional lifetime multiplier appropriate for weapon type IW on side IS. Multiplication of the five-vector addressed by OMEGA(1,IREC,IW,IS) by FRAX(IW,IS) yields the desired type partial potential five-vector.

2. Lines 530 through 536 output the unmodulated and modulated COPs for Blue and Red divisions. Lines 530, 532, 534, and 536 each make a call to subroutine OUTGLO. The first call outputs the Blue unmodulated partial COP. The second call outputs the modulated Blue partial COP. The third call outputs the unmodulated Red partial COP. And finally, the fourth call outputs the modulated Red partial COP.

a. The first argument of OUTGLO is the identifier of the record type.

b. The second argument of OUTGLO, here "0", is the "weapon type" identifier for a division.

c. The third argument of OUTGLO is the address of the appropriate five-vector within the partial COP accumulator array OMEGAC(). The indexing of an element OMEGAC(I,J,K) is to: I=1 for the first com-ponent of a five-vector; J=1 for unmodulated, J=2 for modulated; and K=1 for Blue, K=2 for Red.

(68) Lines 540 through 560 output a recapitulation (with some extensions) of some of the "raw" results of the Combat Module. Note that if TVALON=.TRUE., all elements of OMEGAE() are no longer "raw" inasmuch as all raw kills are then weighted by target values; also, the original "personnel" elements then include other weapons. The report is described in paragraph F-3.b. and is illustrated in Figure F-5. The report is intended to provide for quick checks of numbers before they have been modified extensively by the CBT/CS/CSS Merge process. That the report is printed late in the execution of the CBT/CS/CSS Merge Module is no cause for worry unless the module aborts beforehand. The sequence of instructions loops over side, Blue and Red, as usual. For a given side, the logic loops over direct fire

weapon types (lines 542 through 548) and then over indirect fire weapon types (lines 549 through 559). The separate loops for direct and indirect fire weapons are used because information about losses is retrieved differently for those two weapon classes. Weapon types at zero inventory levels are skipped. The information pertaining to a single weapon type is output by a call to subroutine OUTRAW. The arguments of OUTRAW are:

(a) IS identifies the side.

(b) IW identifies the weapon type.

(c) OMEGAE(1,1,IW,IS) is the address of an unmodulated partial score five-vector corresponding to weapon type IW on side IS from which "raw kills" can be recovered by simple arithmetic.

(d) GZ is the factor by which the five-vector is to be multiplied within subroutine OUTRAW to recover a five-vector of "raw kills" by the standard target categories and their weighted scalar.

(e) ZLOSS(IW,IS) or GZLOS is the losses suffered by weapon type IW on side IS.

(69) Deactivated lines 563 through 611 may be ignored.

(70) Line 612 returns program control to the main program for termination of execution of the CBT/CS/CSS Merge Module, i.e., the real work of the module is complete at this point.

e. Several of the subroutines called by the GGGTARTY version of TARTY, the principal subprogram of the CBT/CS/CSS Merge module, described in paragraphs F-5.c. and d. above are also listed in Figure F-9, beginning at line 619. Some of the subroutines have been deactivated.

(1) **Subroutine OUTREC.** This deactivated subroutine outputs a five-valued partial potential type record in accord with the LER method. It is called four times for each weapon type--for unmodulated score, unmodulated CIP, modulated score, and modulated CIP, respectively. If all five components of a partial potential are zero, no record is output. The subroutine is called four times to output the partial COPs.

(a) The formal arguments of subroutine OUTREC are:

1. ISCNT identifies the type of record by type of potential and side.

2. IDW identifies the "weapon type," currently 1 to 60 for actual weapons and "0" for a division.

3. R is the address of the appropriate five-vector to be output after that vector has been multiplied by FACTOR.

4. FACTOR is the multiplier by which the given five-vector must be multiplied before output. FACTOR itself may reflect the prior result of adjusting for the fractional lifetime factor, the number of days, and the number of divisions.

(b) Lines 627 through 629 put the product of scalar FACTOR and the given five-vector in scratch vector S().

(c) Lines 630 through 632 check whether any component of the five-vector is greater than zero. If so, control goes to the output statement (line 635).

(d) Otherwise, control is transferred back to the calling subprogram (line 633) without output of a record.

(e) Lines 635 through 636 output a record. Only ISCNT, IDW, and S() are significant. The other members of the output list are identifiers not used in AFP work to date.

(2) **Subroutine FILL.** This subroutine, lines 642 through 656, may be ignored.

(3) **Subroutine ZERO.** This trivial subroutine, lines 658 through 664, is called to zero a real array. It is called several times during initialization within the main subprogram for arrays of different dimensions and lengths.

(4) **Subroutine OUTGLO.** This subroutine outputs a five-valued partial potential type record in accord with the GER method. It is called four times for each weapon type--for unmodulated score, unmodulated CIP, modulated score, and modulated CIP, respectively. If all five components of a partial potential are zero, no record is output. The subroutine is also called four times to output the partial COPs.

(a) The formal arguments of subroutine OUTGLO are:

1. ISCNT identifies the type of record by type of potential and side.

2. IDW identifies the "weapon type," currently 1 to 60 for actual weapons and "0" for a division.

3. R is the address of the appropriate five-vector to be output after that vector has been multiplied by FRX.

4. FRX is the multiplier by which the given five-vector must be multiplied before output. FRX itself is the lifetime adjustment factor in the case of weapon types. It is 1.0 in the case of COPs because the lifetime adjustment factors have to have been applied beforehand during the summing of weapon types over the entire divisions.

(b) Lines 673 through 675 check whether any component of the five-vector is greater than zero. If so, control goes to line 678 for more processing.

(c) Otherwise, control is transferred back to the calling program (line 659) without output of a record.

(d) Lines 678 though 680 put the product of scalar FRX and the given five-vector in scratch vector S().

(e) Lines 682 and 683 output a record. Only ISCNT, IDW, and S() are significant. The other members of the output list are identifiers not used in AFP work to date.

(5) Subroutine GETCAT. Subroutine GETCAT is called once from the principal subprogram to read and store the target category values needed for computation of the weighted scalar component of five-vector partial potentials. The values are stored in array CVALS() for later reference by the main subprogram.

(6) Subroutine OUTRAW. Subroutine OUTRAW is called from the main subprogram once for each weapon type with nonzero inventory. It outputs a line within the CBT/CS/CSS Merge Module report recapitulating (with some extensions) some of the "raw" output of the Combat Module. As noted above, if TVALON=.TRUE., the output values are not raw but are weighted by target values.

(a) The formal arguments of OUTRAW are:

1. IS identifies the side: Blue=1, Red=2.

2. IW identifies the weapon type.

3. R is the address of the appropriate five-vector giving an unmodulate score.

4. GZ is a multiplier to be applied to the given five-vector in order to extract the "raw" kills and weighted scalar.

5. GZLOS is the number of losses suffered by weapon type IW.

(b) Lines 706 through 708 put the product of scalar GZ and the given five-vector in scratch array S().

(c) Line 710 outputs a line of the report.

(d) Line 711 returns control to the main subprogram.

(7) Subroutine GETFRX. Subroutine GETFRX is called once from the principal subprogram to read and store the fractional lifetime factors needed for computation of partial combat potentials. The values are stored in array FRAX() for later reference by the main subprogram.

(8) Subroutine OUTPAR. Subroutine OUTPAR is called once from the principal subprogram to read and store index and parameter values as identifiers to be included with the module's output records of partial combat potentials.

f. Figure F-10 provides a listing of the MAP element for collection of the program elements of the CBT/CS/CSS Merge Module.

```
@MAP,E ,30GECTEST.57CCXPS
IN 300YVEY5.CXPS
IN 300YVEY5.INPUT
IN 300YVEY5.CFLDEF
IN 300YVEY5.CINIT
IN 30GECTEST.GGGTARTY
IN 300YVEY5.GETLOS
IN 300YVEY5.GETCA
IN 300YVEY5.GETART
IN 300YVEY5.DIRLOS
IN 300YVEY5.INDLOS
IN 300YVEY5.KEYCMP
END
```

Figure F-10. Listing of the MAP Element for Collection of the Program Elements of the CBT/CS/CSS Merge Module

APPENDIX G

THE AFP ROLLUP AND STATS MODULE

Section I. OVERVIEW

G-1. The AFP Rollup and Stats Module is designed to:

a. Accept intermediate results from the AFP CBT/CS/CSS Merge Module for an organization (division) in each of up to 16 combat environments for each of one or more random number seeds in the form of a separate file or separate element for each environment/seed combination.

b. Accept a set of 16 combat environmental weights to be applied in rolling up the intermediate results from the CBT/CS/CSS Merge Module.

c. Calculate the weighted sum of the partial combat potentials contained in the intermediate result files. The sums of weighted partial combat potentials are the final AFP estimates of combat potentials of equipment and organization (division) for both Blue and Red--friend and threat.

d. Output a new file containing the combat potentials for both sides' equipment and divisions.

e. Calculate and display some comparisons of partial combat potentials among environments. Simple (unweighted) arithmetic means are provided for each type of equipment within each of:

(1) The four postures: RAPD, STATIC, RADE, and BAPD.

(2) The two times of day: day and night.

(3) The two visibility conditions: clear and degraded.

Each of (1) through (3) provides a different stratification of all the intermediate files. The simple (unweighted) means and standard deviations across all files are also provided.

f. Calculate and display the mean modulated scalar CIPs by combat environment for each weapon type. If two or more replications are rolled up, standard deviations by combat environment are also displayed.

G-2. The relation of the AFP Rollup and Stats Module to the AFP System, in general, is portrayed in Figure G-1. There the module is highlighted by being enclosed in an oval.

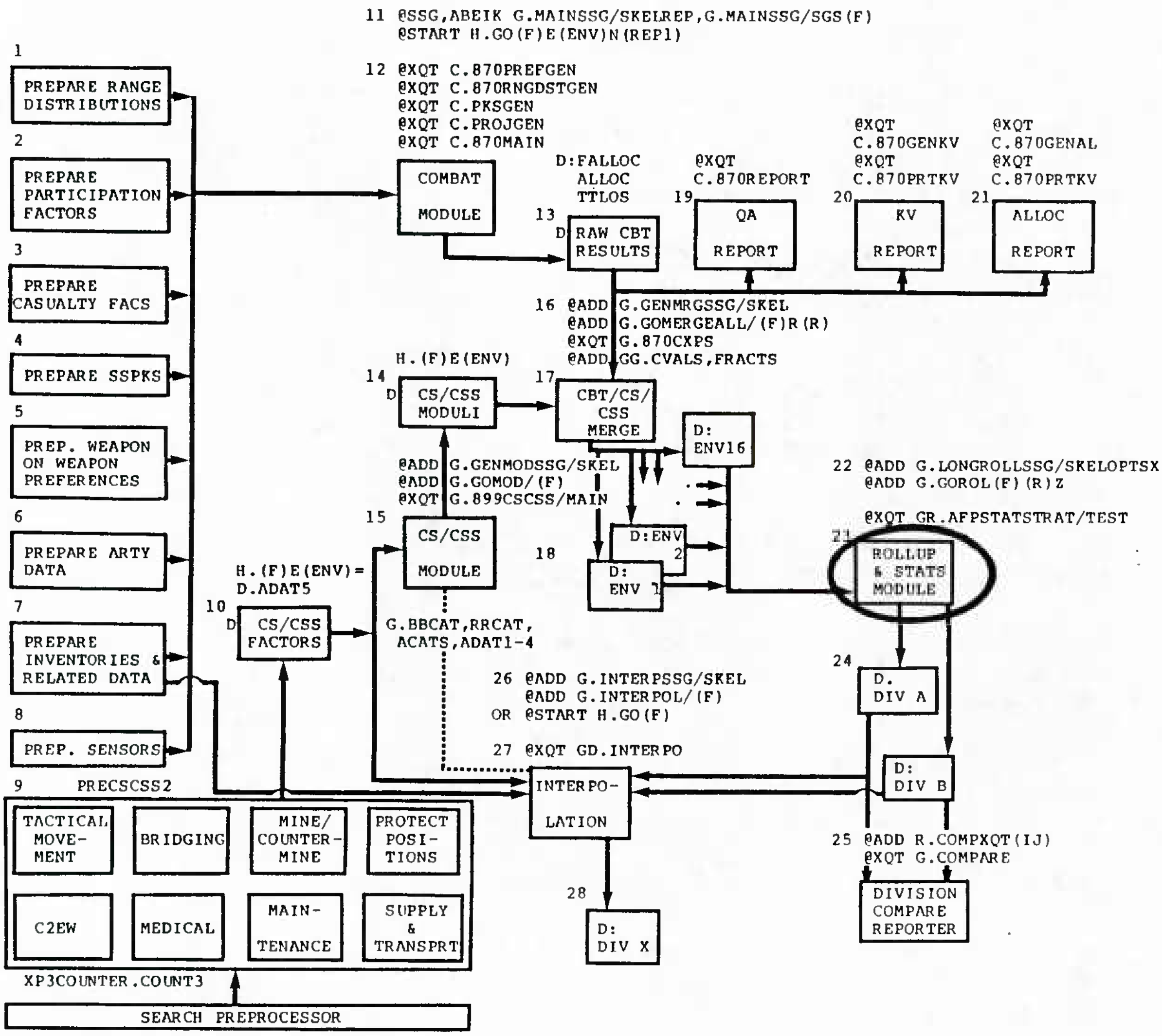


Figure G-1. Relation of the AFP Rollup and Stats Module to the AFP System in General

Section II. INPUT

G-3. The primary input to the AFP Rollup and Stats Module are the files of partial combat potentials from the AFP CBT/CS/CSS Merge Module. A separate file for each combination of combat environment and random number seed is required.

a. Figure G-2 provides an example of the form of the files from the CBT/CS/CSS Merge Module. The fields are described in paragraph 14 of Appendix B. Note, however, that the identifiers in field 1 in Figure G-2 are all less than 100, thereby identifying the contents of Figure G-2 as partial combat potentials. Figure G-2 corresponds to Figure B-7 of Appendix B. As noted in Appendix B, files containing partial and final combat potentials have the same format with different identifiers in field 1. Those identifiers are described in paragraph 12 and Figure B-3 of Appendix B.

	FIELD								
	1	2	3	4	5	6	7	8	9
17.	10	E 1 1 1 1	16	1191.458	84.925	113.229	.000	26.959	1 1 1
18.	30	E 1 1 1 1	16	5.784	.412	.550	.000	.131	1 1 1
19.	50	E 1 1 1 1	16	1197.489	85.355	113.802	.000	27.096	1 1 1
20.	70	E 1 1 1 1	16	5.813	.414	.552	.000	.132	1 1 1
21.	10	E 1 1 1 1	17	644.042	47.542	48.500	.000	13.075	1 1 1
22.	30	E 1 1 1 1	17	5.776	.426	.435	.000	.117	1 1 1
23.	50	E 1 1 1 1	17	647.302	47.782	48.746	.000	13.141	1 1 1
24.	70	E 1 1 1 1	17	5.805	.429	.437	.000	.118	1 1 1
25.	10	E 1 1 1 1	20	3252.010	234.271	345.906	5.000	83.696	1 1 1
26.	30	E 1 1 1 1	20	9.855	.710	1.048	.015	.254	1 1 1
27.	50	E 1 1 1 1	20	3272.702	235.457	347.657	5.448	84.554	1 1 1
28.	70	E 1 1 1 1	20	9.917	.714	1.054	.017	.256	1 1 1
29.	10	E 1 1 1 1	26	118.500	6.625	.000	8.000	9.006	1 1 1
30.	30	E 1 1 1 1	26	3.703	.207	.000	.250	.281	1 1 1
31.	50	E 1 1 1 1	26	120.453	6.659	.000	8.717	9.732	1 1 1
32.	70	E 1 1 1 1	26	3.764	.208	.000	.272	.304	1 1 1
					* * *				
157.	20	E 1 1 1 1	51	1.706	.000	.000	.000	.004	1 1 1
158.	40	E 1 1 1 1	51	.059	.000	.000	.000	.000	1 1 1
159.	60	E 1 1 1 1	51	1.780	.000	.000	.000	.005	1 1 1
160.	80	E 1 1 1 1	51	.061	.000	.000	.000	.000	1 1 1
161.	20	E 1 1 1 1	52	51.161	6.395	.000	.000	.810	1 1 1
162.	40	E 1 1 1 1	52	.839	.105	.000	.000	.013	1 1 1
163.	60	E 1 1 1 1	52	53.377	6.672	.000	.000	.845	1 1 1
164.	80	E 1 1 1 1	52	.875	.109	.000	.000	.014	1 1 1
165.	20	E 1 1 1 1	56	21.396	1.297	.000	.000	.192	1 1 1
166.	40	E 1 1 1 1	56	.181	.011	.000	.000	.002	1 1 1
167.	60	E 1 1 1 1	56	22.322	1.353	.000	.000	.201	1 1 1
168.	80	E 1 1 1 1	56	.189	.011	.000	.000	.002	1 1 1
					* * *				
177.	11	E 1 1 1 1	0	16413.689	1193.812	704.285	112.000	373.358	1 1 1
178.	51	E 1 1 1 1	0	16494.709	1197.939	705.834	122.042	384.262	1 1 1
179.	21	E 1 1 1 1	0	1052.358	180.632	13.746	125.153	148.850	1 1 1
180.	61	E 1 1 1 1	0	1144.605	187.630	14.106	150.793	175.517	1 1 1

Figure G-2. Example Extract Records of Output File of Partial Combat Potentials (for a single combat environment) from the AFP CBT/CS/CSS Merge Module for Input to the AFP Rollup and Stats Module

b. As a minimum, the Rollup and Stats Module requires four files in the format of Figure G-2. A separate file is required for each of the combat environments. The module will accept more than one file for each environment. The feature is provided to permit rollup (and analysis) for one or more sets of files corresponding to different random number seeds used by the Combat Module. However, the number of files should be the same for each environment. Although the module can accept different numbers of files for different environments, the results produced would not be correct, simple, and weighted means. The AFP operator must provide the number of complete sets of files as part of input to the module. If two or more sets of intermediate files exist, the AFP operator may apply the Rollup and Stats Module to different combinations of those sets. If intermediate results exist for each of two seeds, the module may be applied to the seed 1 set, the seed 2 set, and the combined seeds 1 and 2 set in three separate runs. Each run will, in general, yield somewhat different results. Therefore, the "final combat potential" files should be uniquely named.

c. Of course, correct file names must be provided to the Rollup and Merge Module. The AFP operator is responsible for supplying the correct names within the module's runstream. An aid to runstream generation is described in paragraph G-9. Provided that file names are consistent with the suggested naming conventions, the operator can easily generate runstreams for one or more random number seeds per environment.

G-4. Important, but only secondary, input required by the Rollup and Stats Module consists of the 16 environmental weights. The weights are usually supplied by the AFP customer. Weights should be nonnegative and sum to 1.0. The module accepts unformatted records containing the environmental weights. An example of environmental weight input is included in the sample runstream described in G-11.

Section III. OUTPUT

G-5. The AFP Rollup and Stats Module is designed to generate only two forms of output. The principal product is a file containing final equipment and division combat potentials for both friendly and threat forces. The secondary output is a stratified analysis of potentials by posture, time of day, and visibility.

G-6. Figure G-3 portrays extracts from a sample output file containing final combat potentials. The format of the output file is the same as that of all the input files. As noted above, field 1 contains record identifiers. In an output file from the Rollup and Stats Module, all these identifiers are greater than 100. In an input file to the Rollup and Stats Module, all the identifiers in field 1 are less than 100.

FIELD

	1	2	3	4	5	6	7	8	9
25.	110	E 1 0 0 0	16	991.096	67.506	90.345	.000	21.586	1 1 1
26.	130	E 1 0 0 0	16	4.737	.327	.436	.000	.104	1 1 1
27.	150	E 1 0 0 0	16	990.053	67.287	90.312	.000	21.555	1 1 1
28.	170	E 1 0 0 0	16	4.779	.325	.435	.000	.104	1 1 1
29.	110	E 1 0 0 0	17	524.247	35.921	45.623	.000	11.156	1 1 1
30.	130	E 1 0 0 0	17	4.628	.318	.400	.000	.098	1 1 1
31.	150	E 1 0 0 0	17	523.781	35.855	45.722	.000	11.161	1 1 1
32.	170	E 1 0 0 0	17	4.621	.317	.401	.000	.098	1 1 1
33.	110	E 1 0 0 0	20	2669.735	157.594	180.532	3.791	51.092	1 1 1
34.	130	E 1 0 0 0	20	7.992	.473	.538	.011	.153	1 1 1
35.	150	E 1 0 0 0	20	2673.525	157.388	180.920	4.131	51.470	1 1 1
36.	170	E 1 0 0 0	20	8.000	.472	.539	.012	.154	1 1 1
37.	110	E 1 0 0 0	26	105.556	4.346	.000	8.111	8.841	1 1 1
38.	130	E 1 0 0 0	26	3.175	.131	.000	.244	.265	1 1 1
39.	150	E 1 0 0 0	26	107.007	4.320	.000	8.807	9.538	1 1 1
40.	170	E 1 0 0 0	26	3.218	.130	.000	.264	.286	1 1 1
					*				
					*				
					*				
169.	120	E 1 0 0 0	51	3.160	.074	.000	.000	.016	1 1 1
170.	140	E 1 0 0 0	51	.129	.003	.000	.000	.001	1 1 1
171.	160	E 1 0 0 0	51	3.329	.079	.000	.000	.017	1 1 1
172.	180	E 1 0 0 0	51	.136	.003	.000	.000	.001	1 1 1
173.	120	E 1 0 0 0	52	45.216	7.383	.013	.000	.902	1 1 1
174.	140	E 1 0 0 0	52	.741	.121	.000	.000	.015	1 1 1
175.	160	E 1 0 0 0	52	47.456	7.748	.013	.000	.947	1 1 1
176.	180	E 1 0 0 0	52	.778	.127	.000	.000	.015	1 1 1
177.	120	E 1 0 0 0	56	19.989	2.611	.047	.000	.335	1 1 1
178.	140	E 1 0 0 0	56	.169	.022	.000	.000	.003	1 1 1
179.	160	E 1 0 0 0	56	20.952	2.749	.049	.000	.352	1 1 1
180.	180	E 1 0 0 0	56	.178	.023	.000	.000	.003	1 1 1
					*				
					*				
					*				
189.	111	E 1 0 0 0	0	11916.452	897.693	489.232	103.601	293.738	1 1 1
190.	151	E 1 0 0 0	0	11919.645	894.350	488.403	112.592	302.274	1 1 1
191.	121	E 1 0 0 0	0	898.871	184.892	12.760	89.643	112.274	1 1 1
192.	161	E 1 0 0 0	0	975.283	193.200	13.275	106.670	131.446	1 1 1

Figure G-3. Example Extract Records from File of Final Combat Potentials (for all 16 combat environments) as Output by the AFP Rollup and Stats Module

G-7. Figure G-4 displays an extract of the printed statistical analysis across postures, time of day, and visibility. Field 1 contains the same identifiers used in the file of final combat potentials. However, the statistical analysis may contain 10 or 12 lines for every record in the final combat potentials file. The extract in Figure G-4 is for just one weapon type. Each statistical stratification shown in Figure G-4 is performed over the full set of input to the Rollup and Stats Module. If one file (corresponding to a single random number seed input to the Combat Module) is input for each of the 16 combat environments, then each of the postures shown in Figure G-4 is averaged over four environments, each of the times of day is averaged over eight environments, each of the visibilities is averaged over eight environments, and the lines labelled "WTD.MEANS" and WTD.STD.DEVS." are suppressed. The columns through "SCALAR" in Figure G-4 need no new explanation. The last three columns do require some comment.

SCORES, CIPS, & COPS BY POSTURE--

									OPP.LOSS	REL.POS2	DIFF
110	POSTURE= 1	IDWPN=	2	1302.064	.000	.000	.000	137.580	RLOS= 45.350	REF= 16.292	DIFF= -29.563
110	POSTURE= 2	IDWPN=	2	456.967	.000	.000	.000	48.876	RLOS= 48.876	REF= 48.876	DIFF= .000
110	POSTURE= 4	IDWPN=	2	26.594	.000	.000	.000	2.780	RLOS= 8.339	REF= 145.627	DIFF= 139.287
110	TIME=DAY	IDWPN=	2	359.509	.000	.000	.000	36.971			
110	TIME=NIGHT	IDWPN=	2	547.353	.000	.000	.000	57.647			
110	CLEAR	IDWPN=	2	393.279	.000	.000	.000	41.443			
110	DEGRADED	IDWPN=	2	504.583	.000	.000	.000	53.175			
110	RAW MEANS		2	448.931	.000	.000	.000	47.309			
110	RAW STD.DEVS.		2	46.119	.000	.000	.000	5.077			
110	WTD. MEANS		2	440.708	.000	.000	.000	46.513			
110	WTD.STD.DEVS.		2	14.422	.000	.000	.000	1.532			
130	POSTURE= 1	IDWPN=	2	24.112	.000	.000	.000	2.548	RLOS= .349	REF= .307	DIFF= -.042
130	POSTURE= 2	IDWPN=	2	8.811	.000	.000	.000	.922	RLOS= .922	REF= .922	DIFF= .000
130	POSTURE= 4	IDWPN=	2	.497	.000	.000	.000	.052	RLOS= .155	REF= 2.767	DIFF= 2.611
130	TIME=DAY	IDWPN=	2	6.539	.000	.000	.000	.690			
130	TIME=NIGHT	IDWPN=	2	10.171	.000	.000	.000	1.071			
130	CLEAR	IDWPN=	2	7.312	.000	.000	.000	.771			
130	DEGRADED	IDWPN=	2	9.398	.000	.000	.000	.990			
130	RAW MEANS		2	8.355	.000	.000	.000	.880			
130	RAW STD.DEVS.		2	.892	.000	.000	.000	.094			
130	WTD. MEANS		2	8.196	.000	.000	.000	.865			
130	WTD.STD.DEVS.		2	.275	.000	.000	.000	.028			
150	POSTURE= 1	IDWPN=	2	1481.711	.000	.000	.000	156.562	RLOS= 52.187	REF= 17.484	DIFF= -34.703
150	POSTURE= 2	IDWPN=	2	501.131	.000	.000	.000	52.451	RLOS= 52.451	REF= 52.451	DIFF= .000
150	POSTURE= 4	IDWPN=	2	25.640	.000	.000	.000	2.670	RLOS= 8.010	REF= 157.354	DIFF= 149.344
150	TIME=DAY	IDWPN=	2	387.961	.000	.000	.000	40.930			
150	TIME=NIGHT	IDWPN=	2	616.280	.000	.000	.000	64.911			
150	CLEAR	IDWPN=	2	440.941	.000	.000	.000	46.472			
150	DEGRADED	IDWPN=	2	563.300	.000	.000	.000	59.370			
150	RAW MEANS		2	502.121	.000	.000	.000	52.921			
150	RAW STD.DEVS.		2	54.698	.000	.000	.000	5.772			
150	WTD. MEANS		2	493.463	.000	.000	.000	52.089			
150	WTD.STD.DEVS.		2	16.425	.000	.000	.000	1.739			
170	POSTURE= 1	IDWPN=	2	27.439	.000	.000	.000	2.899	RLOS= .966	REF= .330	DIFF= -.637
170	POSTURE= 2	IDWPN=	2	9.455	.000	.000	.000	.990	RLOS= .990	REF= .990	DIFF= .000
170	POSTURE= 4	IDWPN=	2	.478	.000	.000	.000	.050	RLOS= .149	REF= 2.969	DIFF= 2.820
170	TIME=DAY	IDWPN=	2	7.236	.000	.000	.000	.763			
170	TIME=NIGHT	IDWPN=	2	11.450	.000	.000	.000	1.206			
170	CLEAR	IDWPN=	2	8.197	.000	.000	.000	.864			
170	DEGRADED	IDWPN=	2	10.489	.000	.000	.000	1.105			
170	RAW MEANS		2	9.343	.000	.000	.000	.985			
170	RAW STD.DEVS.		2	1.014	.000	.000	.000	.107			
170	WTD. MEANS		2	9.176	.000	.000	.000	.968			
170	WTD.STD.DEVS.		2	.312	.000	.000	.000	.032			
110	POSTURE= 1	IDWPN=	3	1984.598	.000	.000	.000	210.178	RLOS= 70.359	REF= 16.010	DIFF= -54.350
110	POSTURE= 2	IDWPN=	3	458.810	.000	.000	.000	48.029	RLOS= 48.029	REF= 48.029	DIFF= .000
110	POSTURE= 3	IDWPN=	3	91.113	.000	.000	.000	9.111	RLOS= 1.028	REF= 12.207	DIFF= 10.979
110	POSTURE= 4	IDWPN=	3	38.347	.000	.000	.000	4.052	RLOS= 12.157	REF= 144.386	DIFF= 131.229
110	TIME=DAY	IDWPN=	3	822.447	.000	.000	.000	86.840			
110	TIME=NIGHT	IDWPN=	3	438.986	.000	.000	.000	46.345			
110	CLEAR	IDWPN=	3	382.318	.000	.000	.000	40.350			
110	DEGRADED	IDWPN=	3	879.114	.000	.000	.000	92.835			
110	RAW MEANS		3	630.716	.000	.000	.000	66.592			
110	RAW STD.DEVS.		3	110.706	.000	.000	.000	11.760			
110	WTD. MEANS		3	856.394	.000	.000	.000	90.532			
110	WTD.STD.DEVS.		3	160.603	.000	.000	.000	17.142			
130	POSTURE= 1	IDWPN=	3	68.434	.000	.000	.000	7.247	RLOS= 2.416	REF= .577	DIFF= -1.844
130	POSTURE= 2	IDWPN=	3	16.306	.000	.000	.000	1.715	RLOS= 1.715	REF= 1.715	DIFF= .000
130	POSTURE= 3	IDWPN=	3	1.468	.000	.000	.000	.147	RLOS= .377	REF= .420	DIFF= [illegible]

Figure G-4. Example Extract Records from Report of Stratified Partial Combat Potentials as Output by the AFP Rollup and Stats Module

a. The column "OPP.LOSS" translates scalar scores by posture into the implied losses per opposing division. In the Figure G-4 example, Blue scores are converted to Red losses by division of the corresponding Red to Blue division ratios. Because Posture 2 is at 1:1 division ratio, the SCALAR and OPP.LOSS entries are equal. For Posture 1, the OPP.LOSS values are one-third the SCALAR values because Posture 1 is "fought" at 3:1 Red to Blue division ratio.

b. The column "REL.POS2" gives values for all postures based on Posture 2 under the assumptions that Blue weapons are in defilade and Red weapons are in the open (exactly the case only for Postures 1 and 2) and that the underlying scalar loss law is Lanchester linear.

c. The column "DIFF" presents the difference: REL.POS2 - OPP.LOS. DIFF should be 0.0 for Posture 2, the base posture. In results to date, DIFF for Posture 1 is often relatively small suggesting that the AFP Combat Module is often consistent with the above assumptions with regard to many weapon types. DIFF for Postures 3 and 4 may be relatively large because Blue weapons are in the open in both postures and Red weapons are in defilade in Posture 4. DIFF values may be scanned quickly for unusual results.

d. Figure G-5 presents an example of a report produced by the AFP Rollup and Stats Module. For each weapon type on each side, the report presents the mean modulated scalar CIP by combat environment.

(1) A "CIP line" in the report is in the form

```
                                         1    16
SIDE = (1 or 2) WPN = (1 to 60) n.nn...n.nn...n.nn
```

SIDE = 1 for Blue, SIDE = 2 for Red

n.nn in column i is the mean CIP for combat environment i
n.nn in the right most column is the weighted CIP over all combat environments

(2) If two or more replications are rolled up, standard deviations within combat environments are displayed in lines of the form

```
STD.DEVS.                                 1     16
                                        n.nn...n.nn
```

n.nn in column i is the estimated standard deviation for the CIP in the same position in the preceding line.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
SIDE= 1 WPN= 2	1.32	.58	.00	.10	4.02	.81	.00	.01	2.26	1.69	.00	.09	3.93	.98	.03	.00	.97
STD.DEVS.	.35	.07	.00	.02	.51	.10	.00	.00	.65	.55	.00	.01	.59	.09	.00	.00	
SIDE= 1 WPN= 3	4.39	.68	.09	.27	5.91	1.16	.00	.02	16.68	4.29	.44	.25	6.01	1.24	.09	.00	3.54
STD.DEVS.	.66	[illegible]	.00	.04	.60	.19	.02	.00	16.54	3.32	.21	.04	.84	.17	.02	.00	
SIDE= 1 WPN= 5	1.86	1.16	.00	.39	2.68	2.32	.00	.23	3.39	7.58	.00	.40	2.45	2.43	.00	.24	1.56
STD.DEVS.	.32	[illegible]	.00	.05	.78	.96	.00	.07	.75	2.81	.00	.03	1.04	1.35	.00	.08	
SIDE= 1 WPN= 6	4.14	.72	.46	.34	6.71	1.26	.52	.02	13.72	2.77	3.89	.28	6.77	1.24	.51	.00	3.52
STD.DEVS.	.63	.03	.01	.05	1.22	.13	.02	.00	9.93	.68	.48	.04	1.34	.13	.03	.00	
SIDE= 1 WPN=11	1.82	1.29	.12	.45	3.14	2.09	.10	.47	4.61	3.61	.15	.43	3.44	2.69	.10	.44	1.67
STD.DEVS.	.19	.11	.03	.09	.30	.42	.13	.07	.49	.80	.06	.06	.43	.57	.04	.07	
SIDE= 1 WPN=12	7.35	3.78	2.02	6.29	15.96	5.17	1.99	1.85	8.44	5.35	1.98	2.57	17.80	5.04	1.83	1.80	5.70
STD.DEVS.	1.73	.77	.31	2.30	3.21	.94	.34	.51	2.89	1.23	.20	.53	3.32	.74	.35	.54	
SIDE= 1 WPN=13	.00	.03	.00	.00	.00	.05	.00	.00	.00	.06	.00	.00	.00	.03	.00	.00	.01
STD.DEVS.	.00	.07	.00	.00	.00	.02	.00	.00	.00	.03	.00	.00	.00	.01	.00	.00	
SIDE= 1 WPN=16	13.34	3.95	6.05	2.92	14.82	4.13	5.84	1.53	6.45	4.19	3.37	1.36	17.16	4.71	5.53	1.25	6.35
STD.DEVS.	1.47	.61	.31	.33	1.56	.45	.33	.23	.68	.42	.19	.14	1.54	.75	.37	.15	
SIDE= 1 WPN=17	17.71	4.84	6.66	5.07	23.82	4.63	6.43	1.17	7.98	6.06	3.29	1.32	27.56	6.23	6.59	.91	7.99
STD.DEVS.	2.93	.66	.63	.94	3.95	1.22	.47	.18	1.79	1.56	.35	.16	5.27	.92	.67	.13	
SIDE= 1 WPN=22	5.64	6.25	1.87	1.68	6.87	7.59	2.08	.96	4.87	5.36	1.74	.97	6.65	7.45	1.98	1.07	4.15
STD.DEVS.	.69	1.89	.15	.20	1.45	2.85	.11	.08	.99	1.65	.18	.15	.77	3.99	.17	.15	
SIDE= 1 WPN=23	8.76	7.04	2.19	2.47	11.77	9.46	2.33	1.41	12.41	9.92	2.13	1.25	11.78	9.56	2.30	1.46	6.48
STD.DEVS.	1.20	1.59	.22	[illegible]	1.50	3.57	.22	.23	2.18	4.80	.20	.18	1.75	2.44	.13	.15	
SIDE= 1 WPN=26	11.59	12.18	10.86	15.16	21.95	11.57	13.48	15.16	3.13	11.57	4.74	4.74	13.39	12.18	11.72	15.16	10.74
STD.DEVS.	7.78	.00	2.48	.00	13.45	1.93	6.95	.00	.66	1.93	2.49	2.56	8.13	.00	4.85	.00	
SIDE= 1 WPN=31	3.17	5.60	.05	2.50	.00	2.10	1.44	.00	3.27	6.00	.05	1.17	.00	.00	.05	1.92	2.29
STD.DEVS.	1.69	1.26	.01	.51	.00	.32	1.07	.00	1.04	.00	.01	.76	.00	.00	.01	1.01	
SIDE= 1 WPN=32	6.82	5.83	5.85	6.35	6.82	5.83	5.85	6.35	6.82	5.83	5.90	6.35	6.82	5.83	5.85	6.35	6.28
STD.DEVS.	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.15	.00	.00	.00	.00	.00	
SIDE= 1 WPN=36	12.19	14.76	13.78	12.16	1.62	6.90	1.61	5.49	6.55	9.22	1.89	6.41	2.18	6.19	1.32	8.51	6.33
STD.DEVS.	2.40	7.14	3.57	3.51	.89	2.49	.68	2.10	2.74	2.66	.73	1.31	1.51	3.02	.78	2.81	
SIDE= 1 WPN=53	1.51	1.39	1.90	.19	3.27	1.21	3.57	.56	4.58	2.28	3.08	.35	3.51	2.15	2.97	.50	2.03
STD.DEVS.	.45	.40	.78	.13	1.21	.61	1.70	.25	1.16	.81	1.14	.20	1.35	.46	.51	.31	
SIDE= 1 WPN=56	2.89	1.40	5.03	.58	5.27	2.16	6.63	1.38	8.06	3.36	7.92	1.49	5.39	2.21	5.53	1.29	3.79
STD.DEVS.	1.53	.89	1.72	.38	2.19	.99	2.44	.49	2.62	.74	1.38	.50	1.97	1.45	2.35	.63	
SIDE= 1 WPN=57	2.64	4.50	3.07	4.88	1.33	1.57	3.46	4.38	1.17	1.23	3.11	3.45	2.97	2.62	4.53	4.41	3.05
STD.DEVS.	2.96	3.74	2.00	1.47	2.78	2.35	2.84	2.23	2.67	1.80	3.30	1.64	3.05	4.30	3.36	2.01	
SIDE= 1 WPN=58	.00	.00	.00	1.00	.00	.00	.00	1.14	.00	.00	.00	1.08	.00	.00	.00	1.03	.24
STD.DEVS.	.00	.00	.00	.62	.00	.00	.00	.98	.00	.00	.00	.48	.00	.00	.00	.95	
SIDE= 2 WPN= 2	.03	.03	.00	.84	.00	.04	.00	.00	.05	.03	.00	.82	.00	.34	.00	.00	.18
STD.DEVS.	.01	.01	.00	.07	.00	.01	.00	.00	.01	.01	.00	.05	.00	.01	.00	.00	
SIDE= 2 WPN= 3	.08	.20	.02	2.93	.07	.24	.00	.00	.15	.21	2.65	2.68	.08	.17	.03	.00	.94
STD.DEVS.	.01	.06	.00	.48	.02	.06	.00	.00	.03	.09	.52	.55	.02	.05	.03	.00	
SIDE= 2 WPN= 4	.00	.00	.02	.27	.00	.00	.01	.25	.00	.00	.19	.23	.00	.00	.03	.00	.34
STD.DEVS.	.00	.00	.00	.03	.00	.00	.00	.08	.00	.00	.03	.34	.00	.00	.03	.00	
SIDE= 2 WPN= 5	.00	.00	.02	.37	.00	.00	.00	.00	.00	.00	1.43	.33	.00	.00	.00	.00	.27
STD.DEVS.	.00	.00	.01	[illegible]	.00	.00	.00	.00	.00	.00	.40	.11	.00	.00	.03	.00	
SIDE= 2 WPN= 6	.18	1.18	.31	.77	.13	1.03	.32	4.00	.93	.73	1.24	4.01	.13	1.07	.32	2.93	1.10
STD.DEVS.	.09	[illegible]	.01	.19	.07	.32	.01	1.84	.61	.19	.69	1.57	.07	.32	.01	.70	
SIDE= 2 WPN= 7	.18	1.41	.34	.99	.12	1.12	.35	3.23	.49	.56	1.38	5.17	.12	.99	.35	3.65	1.16
STD.DEVS.	.02	.17	.01	.12	.02	.15	.01	.45	.14	.10	.15	.76	.03	.17	.01	.77	
SIDE= 2 WPN= 8	.00	.87	.15	.35	.01	1.14	.16	1.28	.17	1.02	.14	2.97	.00	.75	.19	2.27	.59
STD.DEVS.	.00	.78	.02	.07	.01	.48	.3	.82	.27	1.03	.30	.38	.01	.37	.03	.96	

Figure G-5. Extract from Example Report of Modulated Scalar CIPs and Corresponding Standard Deviations as Output by the AFP Rollup and Stats Module

Section IV. RUNSTREAM

G-8. This section describes a generic program for generating AFP Rollup and Stats Module runstreams and provides some examples of generated runstreams for a few of the most interesting and useful cases. Familiarity with the UNIVAC Symstream language and SSG processor is assumed.

G-9. Runstream generation is intended to simplify several possible problems in applying the Rollup and Stats Module.

a. The module must be "fed" the correct set or sets of intermediate files or elements from the AFP CBT/CS/CSS Merge Module. The setting of a single SGS value within the runstream generator determines whether the resulting runstream will be setup to accept partial combat potentials from files or elements. For a rollup over 16 combat environments (normal application) and two replications (typical but as many as 10 reps may be included), 32 files or elements are correctly assigned or copied and then read by the module. The runstream generator creates the 96 runstream images.

b. If several different sets of files or elements (each set corresponding to a different random number seed's results from the AFP Combat Module) exist, the appropriate combination of sets in accord with naming conventions must be applied, and the number of sets must be specified. The runstream generator performs these tasks from a few SGS definitions. If fewer than two sets are rolled up, some of the standard statistical measures are suppressed because one "replication" is insufficient for estimating standard deviations within a single combat environment.

c. The 16 weights (summing to 1.0), one for each combat environment, must be provided for use in constructing the weighted summations of partial combat potentials over all combat environments and seeds.

d. If files rather than elements are being used throughout for partial and final combat potentials, then a file to receive the final combat potentials must be assigned and used. The runstream generator takes care of this task in accord with the same SGS mentioned in G-9a, above.

e. In the event that the intermediate partial combat potential files do not reflect scores of single divisions, a feature is provided within the Rollup and Stats Module to permit adjustments to single division scores.

G-10. It is certainly an advantage to apply a consistent scheme in naming AFP files and elements. The runstream generator requires that a consistent scheme exist from division to division, combat environment to combat environment, and replication to replication. The Rollup and Stats Module involves, at most, just two classes of AFP files. In the case in which elements (preferred) instead of files are used to save partial and final combat potentials, only a single permanent file is needed for a run. Very many temporary files may be required during a rollup run, however.

a. In the case of separate files for each environment/replication pair, partial combat potential files input should have names of the form:

(two-character user ID)(four-character division name)E(two-digit environment ID) R one- or two-digit replication ID)

e.g., H7HM80E01R1.

The final combat potential output file should have a name of the form:

(two-character user ID)(four-character division name)R16(R or X)(one- or two-digit indicator of number of replications included)

e.g., H7HM80R16X2.

b. In the case of separate elements of input and output potentials all within a single file, the file name should be of the form:

(two-character user ID)(four-character division name)

e.g., H7HM80.

The input elements should have names of the form:

E(two-character combat environment ID)R(one- or two-digit replication ID)

e.g., E01R1.

The output element should have a name of the form:

R16(R or X)(one- or two-digit indicator of the number of replications included)

e.g., R16X2.

G-11. Figure G-6 displays a sample generic SSG program for generating a set of Rollup and Stats Module runstreams. The sample is set up for a specific application: division 3M in year 84 for all 16 combat environments over two replications with combat potentials in elements, not files.

a. SGS Section. The order of SGSs is not important; their contents are critical.

(1) The SGS "FORCE" provides the name of the division of interest.

(2) The SGS "ENV" defines the symbols representing the 16 combat environments.

(3) The SGS "REPS" specifies which replications are to be included in the rollup. A "1" would limit the rollup to rep 1. A "1 2 3" would include reps corresponding to the first three seeds supplied to the AFP Combat Module.

(4) The SGS "ESET" defines the symbol used in the output file or element to record the number of combat environments included in the rollup. ESET should be 16 for a full rollup.

(5) The SGS "RESET" defines the symbol used in the output files or element to record the number of replications included in the rollup. Care must be taken to avoid overwriting prior results that should be retained. For example, separate rollups over reps 1 and 2 and over 3 and 4 should not, in general, both have RSET at X2.

(6) The SGS "USER" specifies the user ID, H7 in the example.

(7) the SGS "KTHTR" specifies a theater symbol to be included in output records. The example E signifies Europe.

```
@SSG,BK
SGS
FORCE HM80
ENV 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
REPS  1 2 3 4 5 6 7 8 9 10
ESET 16
RSET X10
KTHTR ''  E''     .3
JTPD  ''  30''    .4
IBFOR ''  2001''  .6
IRFOR ''  1000''  .6
JCASE '' 8032''   .5
USEFILES N
SNAME H7HM80
INLIST Y
SORT Y
USER H7
SSGLIST N
@EOF
SKEL
*BRKPT,K G6GECTEST.GOROL[FORCE,1,1,1][REPS,1]Z
#HDG UNCLASSIFIED AFP [FORCE,1,1,1] ROLLUP REP[RSET,1,1,1]
*IF [SSGLIST,1,1,1] = Y
#ELT,L G6GECTEST.LONGROLLSSG/SKELOPTSX
#ELT,L G6GECTEST.GOROL[FORCE,1,1,1][REPS,1]Z
*END
*IF [USEFILES,1,1,1] = Y
*INCREMENT IE TO [ENV,1]
*INCREMENT R TO [REPS,1]
#ASG,A [USER,1,1,1][FORCE,1,1,1]E[ENV,1,IE,1]R[REPS,1,R,1].
*LOOP    .R
*LOOP      .IE
#ASG,A [USER,1,1,1][FORCE,1,1,1]R[ESET,1,1,1][RSET,1,1,1].
#USE 30.,[USER,1,1,1][FORCE,1,1,1]R[ESET,1,1,1][RSET,1,1,1].
*ELSE
#ASG,A [SNAME,1,1,1].
#ASG,T 30.
#ASG,T WARNING.,///10000
#ASG,A G6RESULTS.
#ERS G6RESULTS.
*END
#ASG,T 18.
#ERS 30.
#XQT G6ROLLUP.AFPSTATSTRAT/TEST
*. IMAGE GIVING INDICES IN CBT POT OUTPUT RECORDS
*EDIT ON
[KTHTR,1,1,1]&
[JTPD,1,1,1]&
   0  0  0&
[IBFOR,1,1,1]&
```

Figure G-6. Example of SSG Program for the Generation of Runstreams for Execution of the AFP Rollup and Stats Module (page 1 of 2 pages)

```
[IRFOR,1,1,1]&
[JCASE,1,1,1]
 .126 .126 .042 .126 .054 .054 .036 .036
 .128 .032 .096 .064 .016 .040 .016 .008
*EDIT ON
[REPS,1].0 &
*IF [USEFILES,1,1,1] = Y
F
*ELSE
T
*END
*INCREMENT IE TO [ENV,1]
*INCREMENT IT TO [REPS,1]
*EDIT ON
 [ENV,1,IE,1]  1  1  1  1  &
[USER,1,1,1][FORCE,1,1,1]&
*IF [USEFILES,1,1,1] = N
.&
*END
E[ENV,1,IE,1]P[REPS,1,IT,1]&
*IF [USEFILES,1,1,1] = Y
.
*ELSE
*EDIT OFF
*END
*LOOP     .IT
*LOOP     .IE
  0  1  1  1  1  DONE
#DATA,L 18.
#END
#FREE 18.
*IF [USEFILES,1,1,1] = N
#ED 30.,[SNAME,1,1,1].R[ESET,1,1,1][RSET,1,1,1]
*END
#DATA,L  30.
#END
*IF [SORT,1,1,1] = Y
#ASG,T SORTOUT.
#SORT,S
VOLUME=SMALL
KEY=1,5,CH,A:67,10,CH,A
FILEIN=30.
FILEOUT=SORTOUT.
#EOF
#DATA,L SORTOUT.
#END
*END
*IF [INLIST,1,1,1] = Y
*INCREMENT IE TO [ENV,1]
*INCREMENT R1 TO [REPS,1]
*EDIT ON
*IF [USEFILES,1,1,1] = Y
#DATA,L &
*ELSE
#ELT,L &
*END
[USER,1,1,1][FORCE,1,1,1]&
*IF [USEFILES,1,1,1] = N
.&
*END
E[ENV,1,IE,1]P[REPS,1,R1,1]&
*IF [USEFILES,1,1,1] = Y
.
#END
*ELSE
*EDIT OFF
*END
*LOOP     .R1
*LOOP     .IE
*END
*IF [USEFILES,1,1,1] = N
#BRKPT PRINT$
#SYM,U G6RESULTS.,3
*END
@EOF
@EOF
```

Figure G-6. Example of SSG Program for the Generation of Runstreams for Execution of the AFP Rollup and Stats Module (page 2 of 2 pages)

(8) The SGS "JTPD" specifies a two-digit year symbol to be included in output records. The example 84 corresponds to the year included in the division name, 3M84.

(9) The SGS "IBFOR" specifies a four- to six-digit division identifier to be included in output records. A TPSN is a logical choice for division identifier.

(10) The SGS "IRFOR" specifies a four- to six-digit threat identifier to be included in output records.

(11) The SGS "JCASE" provides an up to six-digit identifier of the case being considered for inclusion in output records.

(12) The SGS "USEFILES" specifies whether the rollup is to be performed using input and output files (Y) or elements (N).

(13) The SGS "SNAME" specifies the name of the file containing input elements and receiving the output element. SNAME is critical only if USEFILES is N, as in the example.

(14) The SGS "INLIST" specifies whether the input partial combat potentials are to be listed. INLIST set to Y causes the potentials for all input combat environments and replications to be listed. N suppresses the listing of input.

(15) The SGS "SORT" specifies whether the output final combat potentials are to be sorted and listed in ascending scalar potential order.

(16) The SGS "SSGLIST" specifies whether the SSG program and generated runstream are to be listed in the rollup output.

b. SKELeton Section. The SKEL section of Figure G-6 creates names of files and elements in accord with the above SGS definitions. The SKEL logic loops over combat environments and replications as appropriate. The generated runstream element is saved for inspection prior to execution. The SKEL section is hardly the most general program imaginable. Special AFP cases or conditions may require modification of both SGS and SKEL. However, many special, one-time variations may be handled most easily by editing a standard generated runstream.

G-12. RUNSTREAM EXAMPLES. Figures G-7 through G-9 provide examples of Rollup and Stats Module runstreams generated by the runstream generator.

a. Figure G-7 is an example runstream for a division over 16 combat environments and 1 replication and using input and output elements.

b. Figure G-8 is an example runstream for a division over 16 combat environments and 6 replications and using input and output elements.

c. Figure G-9 is an example runstream for a division over 16 combat environments and 2 replications and using input and output files without listing of input files.

```
@HDG UNCLASSIFIED AFP JM84 ROLLUP REPX1
@ASG,A H7JM84.
@ASG,T 30.
@ERS 30.
@XQT G6ROLLUP.AFPSTATSTRAT/TEST
  E  84  0  0  0  2001  1000 8432
 .126 .126 .042 .126 .054 .054 .036 .036
 .126 .032 .096 .064 .016 .040 .016 .008
1.0 T
 01  1  1  1  1  H7JM84.E01R1
 02  1  1  1  1  H7JM84.E02R1
 03  1  1  1  1  H7JM84.E03R1
 04  1  1  1  1  H7JM84.E04R1
 05  1  1  1  1  H7JM84.E05R1
 06  1  1  1  1  H7JM84.E06R1
 07  1  1  1  1  H7JM84.E07R1
 08  1  1  1  1  H7JM84.E08R1
 09  1  1  1  1  H7JM84.E09R1
 10  1  1  1  1  H7JM84.E10R1
 11  1  1  1  1  H7JM84.E11R1
 12  1  1  1  1  H7JM84.E12R1
 13  1  1  1  1  H7JM84.E13R1
 14  1  1  1  1  H7JM84.E14R1
 15  1  1  1  1  H7JM84.E15R1
 16  1  1  1  1  H7JM84.E16R1
  0  1  1  1  1  DONE
@ED 30.,H7JM84.R16X1
@DATA,L  30.
@END
@ASG,T SORTOUT.
@SORT,S
VOLUME=SMALL
KEY=1,5,CH,A:67,10,CH,A
FILEIN=30.
FILEOUT=SORTOUT.
@EOF
@DATA,L SORTOUT.
@END
@ELT,L  H7JM84.E01R1
@ELT,L  H7JM84.E02R1
@ELT,L  H7JM84.E03R1
@ELT,L  H7JM84.E04R1
@ELT,L  H7JM84.E05R1
@ELT,L  H7JM84.E06R1
@ELT,L  H7JM84.E07R1
@ELT,L  H7JM84.E08R1
@ELT,L  H7JM84.E09R1
@ELT,L  H7JM84.E10R1
@ELT,L  H7JM84.E11R1
@ELT,L  H7JM84.E12R1
@ELT,L  H7JM84.E13R1
@ELT,L  H7JM84.E14R1
@ELT,L  H7JM84.E15R1
@ELT,L  H7JM84.E16R1
```

Figure G-7. First Example of an SSG-generated Runstream for Execution of the AFP Rollup and Stats Module

```
@HDG UNCLASSIFIED AFP JM84 ROLLUP REPY6
@ASG,A H7JM84.
@ASG,T 30.
@ASG,T WARNING.,///10000
@ASG,A G6RESULTS.
@FRS G6RESULTS.
@ASG,T 18.
@FRS 30.
@XGT G6ROLLUP.AFPSTATSTRAT/TEST
  E   84   0   0   0   3001   1000 8432
 .126 .126 .042 .126 .054 .054 .036 .036
 .126 .032 .096 .064 .016 .040 .016 .008
6.0  T
 01   1   1   1   1   H7JM84.E01R1
 01   1   1   1   1   H7JM84.E01R2
 01   1   1   1   1   H7JM84.E01R3
 01   1   1   1   1   H7JM84.E01R4
 01   1   1   1   1   H7JM84.E01R5
 01   1   1   1   1   H7JM84.E01R6
 02   1   1   1   1   H7JM84.E02R1
 02   1   1   1   1   H7JM84.E02R2
 02   1   1   1   1   H7JM84.E02R3
 02   1   1   1   1   H7JM84.E02R4
 02   1   1   1   1   H7JM84.E02R5
 02   1   1   1   1   H7JM84.E02R6
 03   1   1   1   1   H7JM84.E03R1
 03   1   1   1   1   H7JM84.E03R2
 03   1   1   1   1   H7JM84.E03R3
 03   1   1   1   1   H7JM84.E03R4
 03   1   1   1   1   H7JM84.E03R5
 03   1   1   1   1   H7JM84.E03R6
 04   1   1   1   1   H7JM84.E04R1
 04   1   1   1   1   H7JM84.E04R2
 04   1   1   1   1   H7JM84.E04R3
 04   1   1   1   1   H7JM84.E04R4
 04   1   1   1   1   H7JM84.E04R5
 04   1   1   1   1   H7JM84.E04R6
 05   1   1   1   1   H7JM84.E05R1
 05   1   1   1   1   H7JM84.E05R2
 05   1   1   1   1   H7JM84.E05R3
 05   1   1   1   1   H7JM84.E05R4
 05   1   1   1   1   H7JM84.E05R5
 05   1   1   1   1   H7JM84.E05R6
 06   1   1   1   1   H7JM84.E06R1
 06   1   1   1   1   H7JM84.E06R2
 06   1   1   1   1   H7JM84.E06R3
 06   1   1   1   1   H7JM84.E06R4
 06   1   1   1   1   H7JM84.E06R5
 06   1   1   1   1   H7JM84.E06R6
 07   1   1   1   1   H7JM84.E07R1
```

Figure G-8. Second Example of an SSG-generated Runstream for Execution of the AFP Rollup and Stats Module (page 1 of 5 pages)

51	07	1	1	1	1	H7JM84.E07R2
52	07	1	1	1	1	H7JM84.E07R3
53	07	1	1	1	1	H7JM84.E07R4
54	07	1	1	1	1	H7JM84.E07R5
55	07	1	1	1	1	H7JM84.E07R6
56	08	1	1	1	1	H7JM84.E08R1
57	08	1	1	1	1	H7JM84.E08R2
58	08	1	1	1	1	H7JM84.E08R3
59	08	1	1	1	1	H7JM84.E08R4
60	08	1	1	1	1	H7JM84.E08R5
61	08	1	1	1	1	H7JM84.E08R6
62	09	1	1	1	1	H7JM84.E09R1
63	09	1	1	1	1	H7JM84.E09R2
64	09	1	1	1	1	H7JM84.E09R3
65	09	1	1	1	1	H7JM84.E09R4
66	09	1	1	1	1	H7JM84.E09R5
67	09	1	1	1	1	H7JM84.E09R6
68	10	1	1	1	1	H7JM84.E10R1
69	10	1	1	1	1	H7JM84.E10R2
70	10	1	1	1	1	H7JM84.E10R3
71	10	1	1	1	1	H7JM84.E10R4
72	10	1	1	1	1	H7JM84.E10R5
73	10	1	1	1	1	H7JM84.E10R6
74	11	1	1	1	1	H7JM84.E11R1
75	11	1	1	1	1	H7JM84.E11R2
76	11	1	1	1	1	H7JM84.E11R3
77	11	1	1	1	1	H7JM84.E11R4
78	11	1	1	1	1	H7JM84.E11R5
79	11	1	1	1	1	H7JM84.E11R6
80	12	1	1	1	1	H7JM84.E12R1
81	12	1	1	1	1	H7JM84.E12R2
82	12	1	1	1	1	H7JM84.E12R3
83	12	1	1	1	1	H7JM84.E12R4
84	12	1	1	1	1	H7JM84.E12R5
85	12	1	1	1	1	H7JM84.E12R6
86	13	1	1	1	1	H7JM84.E13R1
87	13	1	1	1	1	H7JM84.E13R2
88	13	1	1	1	1	H7JM84.E13R3
89	13	1	1	1	1	H7JM84.E13R4
90	13	1	1	1	1	H7JM84.E13R5
91	13	1	1	1	1	H7JM84.E13R6
92	14	1	1	1	1	H7JM84.E14R1
93	14	1	1	1	1	H7JM84.E14R2
94	14	1	1	1	1	H7JM84.E14R3
95	14	1	1	1	1	H7JM84.E14R4
96	14	1	1	1	1	H7JM84.E14R5
97	14	1	1	1	1	H7JM84.E14R6
98	15	1	1	1	1	H7JM84.E15R1
99	15	1	1	1	1	H7JM84.E15R2
100	15	1	1	1	1	H7JM84.E15R3

Figure G-8. Second Example of an SSG-generated Runstream for Execution of the AFP Rollup and Stats Module (page 2 of 5 pages)

```
      15   1   1   1   1    H7JM84.E15R4
      15   1   1   1   1    H7JM84.E15R5
      15   1   1   1   1    H7JM84.E15R6
      16   1   1   1   1    H7JM84.E16R1
      16   1   1   1   1    H7JM84.E16R2
      16   1   1   1   1    H7JM84.E16R3
      16   1   1   1   1    H7JM84.E16R4
      16   1   1   1   1    H7JM84.E16R5
      16   1   1   1   1    H7JM84.E16R6
       0   1   1   1   1    DONE
    @DATA,L 18.
    @END
    @FREE 18.
    @ED 30.,H7JM84.R16X6
    @DATA,L  30.
    @END
    @ASG,T SORTOUT.
    @SORT,S
    VOLUME=SMALL
    KEY=1,5,CH,A:67,10,CH,A
    FILEIN=30.
    FILEOUT=SORTOUT.
    @EOF
    @DATA,L SORTOUT.
    @END
    @ELT,L H7JM84.E01R1
    @ELT,L H7JM84.E01R2
    @ELT,L H7JM84.E01R3
    @ELT,L H7JM84.E01R4
    @ELT,L H7JM84.E01R5
    @ELT,L H7JM84.E01R6
    @ELT,L H7JM84.E02R1
    @ELT,L H7JM84.E02R2
    @ELT,L H7JM84.E02R3
    @ELT,L H7JM84.E02R4
    @ELT,L H7JM84.E02R5
    @ELT,L H7JM84.E02R6
    @ELT,L H7JM84.E03R1
    @ELT,L H7JM84.E03R2
    @ELT,L H7JM84.E03R3
    @ELT,L H7JM84.E03R4
    @ELT,L H7JM84.E03R5
    @ELT,L H7JM84.E03R6
    @ELT,L H7JM84.E04R1
    @ELT,L H7JM84.E04R2
    @ELT,L H7JM84.E04R3
    @ELT,L H7JM84.E04R4
    @ELT,L H7JM84.E04R5
    @ELT,L H7JM84.E04R6
    @ELT,L H7JM84.E05R1
```

Figure G-8. Second Example of an SSG-generated Runstream for Execution of the AFP Rollup and Stats Module (page 3 of 5 pages)

```
@ELT,L  H7JM84.E05R2
@ELT,L  H7JM84.E05R3
@ELT,L  H7JM84.E05R4
@ELT,L  H7JM84.E05R5
@ELT,L  H7JM84.E05R6
@ELT,L  H7JM84.E06R1
@ELT,L  H7JM84.E06R2
@ELT,L  H7JM84.E06R3
@ELT,L  H7JM84.E06R4
@ELT,L  H7JM84.E06R5
@ELT,L  H7JM84.E06R6
@ELT,L  H7JM84.E07R1
@ELT,L  H7JM84.E07R2
@ELT,L  H7JM84.E07R3
@ELT,L  H7JM84.E07R4
@ELT,L  H7JM84.E07R5
@ELT,L  H7JM84.E07R6
@ELT,L  H7JM84.E08R1
@ELT,L  H7JM84.E08R2
@ELT,L  H7JM84.E08R3
@ELT,L  H7JM84.E08R4
@ELT,L  H7JM84.E08R5
@ELT,L  H7JM84.E08R6
@ELT,L  H7JM84.E09R1
@ELT,L  H7JM84.E09R2
@ELT,L  H7JM84.E09R3
@ELT,L  H7JM84.E09R4
@ELT,L  H7JM84.E09R5
@ELT,L  H7JM84.E09R6
@ELT,L  H7JM84.E10R1
@ELT,L  H7JM84.E10R2
@ELT,L  H7JM84.E10R3
@ELT,L  H7JM84.E10R4
@ELT,L  H7JM84.E10R5
@ELT,L  H7JM84.E10R6
@ELT,L  H7JM84.E11R1
@ELT,L  H7JM84.E11R2
@ELT,L  H7JM84.E11R3
@ELT,L  H7JM84.E11R4
@ELT,L  H7JM84.E11R5
@ELT,L  H7JM84.E11R6
@ELT,L  H7JM84.E12R1
@ELT,L  H7JM84.E12R2
@ELT,L  H7JM84.E12R3
@ELT,L  H7JM84.E12R4
@ELT,L  H7JM84.E12R5
@ELT,L  H7JM84.E12R6
@ELT,L  H7JM84.E13R1
@ELT,L  H7JM84.E13R2
@ELT,L  H7JM84.E13R3
```

Figure G-8. Second Example of an SSG-generated Runstream for Execution of the AFP Rollup and Stats Module (page 4 of 5 pages)

```
@ELT,L  H7JM84.E13R4
@ELT,L  H7JM84.E13R5
@ELT,L  H7JM84.E13R6
@ELT,L  H7JM84.E14R1
@ELT,L  H7JM84.E14R2
@ELT,L  H7JM84.E14R3
@ELT,L  H7JM84.E14R4
@ELT,L  H7JM84.E14R5
@ELT,L  H7JM84.E14R6
@ELT,L  H7JM84.E15R1
@ELT,L  H7JM84.E15R2
@ELT,L  H7JM84.E15R3
@ELT,L  H7JM84.E15R4
@ELT,L  H7JM84.E15R5
@ELT,L  H7JM84.E15R6
@ELT,L  H7JM84.E16R1
@ELT,L  H7JM84.E16R2
@ELT,L  H7JM84.E16R3
@ELT,L  H7JM84.E16R4
@ELT,L  H7JM84.E16R5
@ELT,L  H7JM84.E16R6
@BRKPT  PRINT$
@SYM,U  G6RESULTS.
```

Figure G-8. Second Example of an SSG-generated Runstream for Execution of the AFP Rollup and Stats Module (page 5 of 5 pages)

```
@HDG UNCLASSIFIED AFP 3M84 ROLLUP REPY2
@ELT,L G6GECTEST.LONGROLLSSG/SKELOPTS
@ELT,L G6GECTEST.G0ROL3M842X
@ASG,A H73M84E01R1.
@ASG,A H73M84E01R2.
@ASG,A H73M84F02R1.
@ASG,A H73M84E02R2.
@ASG,A H73M84E03R1.
@ASG,A H73M84E03R2.
@ASG,A H73M84E04R1.
@ASG,A H73M84E04R2.
@ASG,A H73M84E05R1.
@ASG,A H73M84E05R2.
@ASG,A H73M84F06R1.
@ASG,A H73M84E06R2.
@ASG,A H73M84E07R1.
@ASG,A H73M84E07R2.
@ASG,A H73M84E08R1.
@ASG,A H73M84E08R2.
@ASG,A H73M84E09R1.
@ASG,A H73M84F09R2.
@ASG,A H73M84F10R1.
@ASG,A H73M84E10R2.
@ASG,A H73M84E11R1.
@ASG,A H73M84E11R2.
@ASG,A H73M84E12R1.
@ASG,A H73M84E12R2.
@ASG,A H73M84E13R1.
@ASG,A H73M84F13R2.
@ASG,A H73M84E14R1.
@ASG,A H73M84F14R2.
@ASG,A H73M84E15R1.
@ASG,A H73M84F15R2.
@ASG,A H73M84E16R1.
@ASG,A H73M84E16R2.
@ASG,A H73M84R16X2.
@USE 30.,H73M84R16X2.
@ERS 30.
@XQT G6ROLLUP.AFPSTATSTRAT
  E  84  0  0  0  2001  1000 8432
 .126 .126 .042 .126 .054 .054 .036 .036
 .128 .032 .096 .064 .016 .040 .016 .008
2.0
 01   1   1   1   1   H73M84E01R1.
 01   1   1   1   1   H73M84E01R2.
 02   1   1   1   1   H73M84F02R1.
 02   1   1   1   1   H73M84E02R2.
 03   1   1   1   1   H73M84E03R1.
 03   1   1   1   1   H73M84E03R2.
 04   1   1   1   1   H73M84F04R1.
```

Figure G-9. Third Example of an SSG-generated Runstream for Execution of the AFP Rollup and Stats Module (page 1 of 2 pages)

```
        04   1   1   1   1   H73M84E04R2.
        05   1   1   1   1   H73M84E05R1.
        05   1   1   1   1   H73M84E05R2.
        06   1   1   1   1   H73M84E06R1.
        06   1   1   1   1   H73M84E06R2.
        07   1   1   1   1   H73M84E07R1.
        07   1   1   1   1   H73M84E07R2.
        08   1   1   1   1   H73M84E08R1.
        08   1   1   1   1   H73M84E08R2.
        09   1   1   1   1   H73M84E09R1.
        09   1   1   1   1   H73M84E09R2.
        10   1   1   1   1   H73M84E10R1.
        10   1   1   1   1   H73M84E10R2.
        11   1   1   1   1   H73M84E11R1.
        11   1   1   1   1   H73M84E11R2.
        12   1   1   1   1   H73M84E12R1.
        12   1   1   1   1   H73M84E12R2.
        13   1   1   1   1   H73M84E13R1.
        13   1   1   1   1   H73M84E13R2.
        14   1   1   1   1   H73M84E14R1.
        14   1   1   1   1   H73M84E14R2.
        15   1   1   1   1   H73M84E15R1.
        15   1   1   1   1   H73M84E15R2.
        16   1   1   1   1   H73M84E16R1.
        16   1   1   1   1   H73M84E16R2.
         0   1   1   1   1   DONE
      @DATA,L  30.
      @END
      @ASG,T SORTOUT.
      @SORT,S
      VOLUME=SMALL
      KEY=1,5,CH,A:67,10,CH,A
      FILEIN=30.
      FILEOUT=SORTOUT.
      @EOF
      @DATA,L SORTOUT.
      @END
```

Figure G-9. Third Example of an SSG-generated Runstream for Execution of the AFP Rollup and Stats Module (page 2 of 2 pages)

Section V. PROGRAM

G-13. Apart from its input and output functions, the AFP Rollup and Stats Module is nothing more than a stratified adding and averaging program with the side role of computing some standard deviations. The only chance for confusion arises because the program, in effect, concurrently adds and averages several different though related data streams. Figure G-10 portrays the basic logic of the AFP Rollup and Stats Module. Logical file 29 is used repeatedly as the source of intermediate partial combat potentials. In current applications, the program successively attaches 16 or 32 physical files to unit 29. Final combat potentials are output to unit 30. Most of the arithmetic performed by the module is devoted to updating multidimensional arrays with simple and weighted partial sums of potentials corresponding to different shooters by side with their unmodulated and modulated scores and CIPs. Arrays for the grand totals by side are also updated throughout. Elements of several of the arrays are divided by appropriate counters to yield arithmetic means. Concurrent with the ordinary summations, some sums of squares are maintained for use in determination of standard deviations.

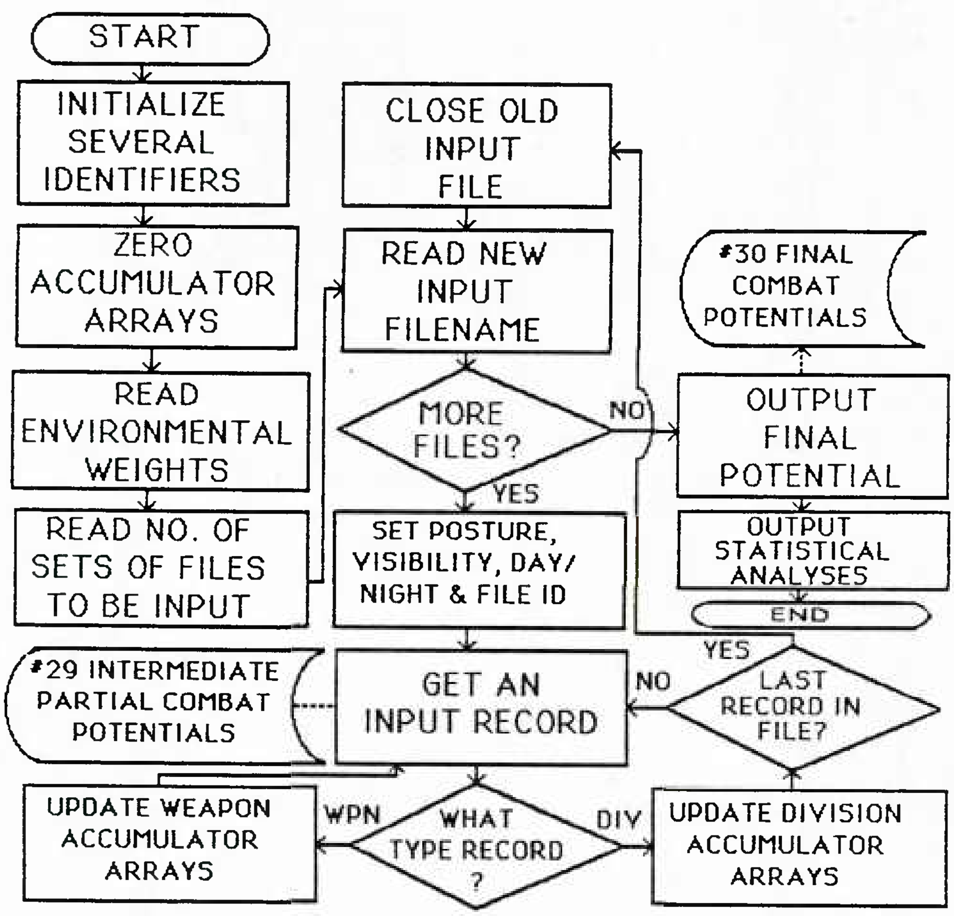


Figure G-10. Flow Diagram of the Basic Logic of the AFP Rollup and Stats Module

G-14. AFPSTATSTRAT is the current version of the source program for the AFP Rollup and Stats Module. The AFPSTATSTRAT source program is listed in Figure G-11.

a. The principal working arrays for accumulating results are:

(1) WVAL(5,4,60,2). Final combat potentials of weapons are developed within this array. During execution of the program, weighted intermediate partial combat potentials over all 16 combat environments and one or two random number seeds are accumulated within the array. An array element WVAL(I,J,K,L) is indexed:

(a) I=1 to 5 for the components of five-valued form of combat potentials: personnel, light armored vehicles, heavy armored vehicles, aircraft, and the weighted, rolled-up scalar.

(b) J=1 to 4 for the four kinds of potentials: unmodulated score, unmodulated CIP, modulated score, and modulated CIP.

(c) K=1 to 60 for the 60 different weapon types permitted.

(d) L=1 to 2 for the two sides: Blue and Red.

(2) COP(5,2,2). Final combat potentials of divisions (COPs) are developed within this array. During program execution, weighted partial COPs over all 16 combat environments and one or two random number seeds are accumulated within the array. An array element COP(I,J,K) is indexed:

(a) I=1 to 5 as in a(1)(a) above.

(b) J=1 to 2 for the two kinds of COPs: unmodulated COP and modulated COP.

(c) K=1 to 2 for the two sides: Blue and Red.

(3) XVAL(5,4,60,2). Simple arithmetic mean combat potentials (scores and CIPs) for weapon types are developed within this array. XVAL and WVAL would contain exactly the same results if the combat environmental weights applied in developing WVAL were all 1/16. The indexing of the elements XVAL(I,J,K,L) is exactly the same as for WVAL() above.

(4) X2VAL(5,4,60,2). The squares of the intermediate partial combat potentials are accumulated within this array for use in the determination of standard deviations across all 16 combat environments. The indexing of elements X2VAL(I,J,K,L) is exactly the same as for XVAL() and WVAL() above.

(5) XCOP(5,2,2). Simple arithmetic mean combat potentials (COPs) for divisions are developed within this array. XCOP and COP would contain exactly the same results if the combat environmental weights applied in developing COP() were all 1/16. The indexing of elements XCOP(I,J,K) is exactly the same as for COP() above.

```
      C     ROLLUP TARTY OUTPUT OVER ALL ENVIRONMENTS
      C     G. E. C.  26 SEP 83
      C      WITH MEANS AND STD. DEVS.    4 OCT 83
      C       WITH POSTURE SUB-MEANS  13 OCT 83
      C        WITH DAY/NIGHT SUBMEANS  21 OCT 83
      C        WITH VISIBILITY SUBMEANS 21 OCTOBER 83
      C         MODIFIED TO HANDLE .LE. 16 ENVIRONMENTS 20 MAR 84
            DIMENSION WVAL(5,4,60,2),COP(5,2,2),ENVWTS(16),ISCNTW(6,2),
           1 NTYPS(2),ISCNTS(12),X(5),ISCNTR(6,2),ISCNTT(12),JRECS(12),
           2 DIVM(2),DIVD(2)
            DIMENSION XVAL(5,4,60,2),X2VAL(5,4,60,2),
           1 XCOP(5,2,2),X2COP(5,2,2)
            DIMENSION PXVAL(5,4,60,2,4),PXCOP(5,2,2,4)
            DIMENSION DXVAL(5,4,60,2,2),DXCOP(5,2,2,2)
            DIMENSION VXVAL(5,4,60,2,2),VXCOP(5,2,2,2)
            DIMENSION MDAY(16),ENVSW(16)
            DIMENSION DAYCNT(2),VISCNT(2),POSCNT(4)
            EQUIVALENCE (ISCNTW,ISCNTS),(ISCNTR,ISCNTT)
            COMMON/AWRK/IWRK,IDUM,KTHTR,JTPD,JVIS,JPOS,JDAY
            COMMON/GLOBAL/IBFOR,IRFOR,JCASE
            COMMON/NAME/FNAME
            COMMON/ENVS/EWTTOT
            COMMON/STRAT/IREP,XSUM(5,4,16,60,2),XSQ(5,4,16,60,2),
           1 CSUM(5,2,16,2),CSQ(5,2,16,2),XSTD(5,4,60,2),CSTD(5,2,2)
            CHARACTER*50 INSTR
            CHARACTER*20 FNAME,ONAME,DONE
            CHARACTER*3 KTHTR
            LOGICAL AUS1,AUS2,ENVSW,ELTSW
            DATA ISCNTW/
           1 110,130,150,170,111,151,
           2 120,140,160,180,121,161/
            DATA NTYPS/2*60/
            DATA ISCNTR/
           1 10,30,50,70,11,51,
           2 20,40,60,80,21,61/
            DATA JRECS/
           1 1,2,3,4,1,2,
           2 1,2,3,4,1,2/
            DATA MDAY/1,1,1,1,2,2,2,2,1,1,1,1,2,2,2,2/
            DATA ENVSW/16*.FALSE./
            DATA DAYCNT/2*0.0/,VISCNT/2*0.0/,POSCNT/4*0.0/
      C
      C     READ OUTPUT RECORD IDENTIFIERS
            READ(5,4) KTHTR,JTPD,JVIS,JPOS,JDAY,IBFOR,IRFOR,JCASE
          4 FORMAT(A3,I4,3I3,2I6,I5)
      C
            N=0
      C     LOGICAL DESTINATION OF ROLLUP OUTPUT
            IWRK=30
            IPRNT=18
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 1 of 12 pages)

```
C       LOGICAL SOURCE OF TARTY FILES
        ISOR=29
        DONE='DONE'
C
        CALL ZERO(WVAL,2400)
        CALL ZERO(COP,20)
        CALL ZERO(XVAL,2400)
        CALL ZERO(X2VAL,2400)
        CALL ZERO(XCOP,20)
        CALL ZERO(X2COP,20)
        CALL ZERO(PXVAL,9600)
        CALL ZERO(PXCOP,80)
        CALL ZERO(DXVAL,4800)
        CALL ZERO(VXVAL,4800)
        CALL ZERO(DXCOP,40)
        CALL ZERO(VXCOP,40)
        CALL ZERO(XSUM,38400)
        CALL ZERO(XSQ,38400)
        CALL ZERO(CSUM,320)
        CALL ZERO(CSQ,320)
        CALL ZERO(XSTD,2400)
        CALL ZERO(CSTD,20)
C       READ THE ENVIRONMENTAL WEIGHTS
        READ(5,1) (ENVWTS(I),I=1,16)
      1 FORMAT()
C
C         READ THE NUMBER OF REPS EXPECTED & ELTSW SETTING
          READ(5,1) REPS,ELTSW
        IF(ELTSW) ISOR=5
        IREP=IFIX(REPS)
        IF(ELTSW) CALL FACSF('&BRKPT PRINTS/WARNING')
        GOTO 95
C
     90 CLOSE(ISOR)
     95 CALL SETFIL(FNAME,IENV,DIVM(1),DIVD(1),DIVM(2),DIVD(2))
        IF(FNAME.EQ.DONE) GOTO 2000
        EWT=ENVWTS(IENV)/REPS
        ENVSW(IENV)=.TRUE.
        N=N+1
        LPOS=1+MOD(IENV-1,4)
        IF(IENV.LT.9) THEN
          LVIS=1
        ELSE
          LVIS=2
        ENDIF
        LDAY=MDAY(IENV)
        POSCNT(LPOS)=POSCNT(LPOS)+1.0
        VISCNT(LVIS)=VISCNT(LVIS)+1.0
        DAYCNT(LDAY)=DAYCNT(LDAY)+1.0
        ONAME=FNAME
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 2 of 12 pages)

```
      WRITE(18,3) FNAME
    3 FORMAT(1X,A20)
      IF(ELTSW) THEN
        INSTR='@ADD,L '//ONAME
      ELSE
        INSTR='@USE 29.,'//ONAME//'.'
      ENDIF
      CALL FACSF(INSTR)
C
  100 CALL GETREC(ISCNT,IWPN,X,ISOR)
      DO 200 I=1,12
      IF(ISCNT.NE.ISCNTT(I)) GOTO 200
      IS=I
      GOTO 300
  200 CONTINUE
      WRITE(IPRNT,2) ISCNT
    2 FORMAT(/' UNRECOGNIZABLE RECORD TYPE=',I5/)
      IERR1=IERR1+1
      IF(IERR1.LT.10) GOTO 100
      STOP 'IERR1'
C
  300 IREC=JRECS(IS)
      IF(IS.LT.7) THEN
       ISIDE=1
      ELSE
       ISIDE=2
      ENDIF
C
      IF((IS.EQ.5).OR.(IS.EQ.6).OR.(IS.EQ.11).OR.(IS.EQ.12)) GOTO 500
C
C     HERE IF WEAPON RECORD READ
C
C     CIPS DO NOT NEED ADJUSTMENT FOR NUMBER OF DIVISIONS
C
      ZZ=1.0
      ZZ2=1.0
      IF(MOD(IS,2).EQ.0) THEN
       XX=EWT
      ELSE
       XX=EWT*DIVM(ISIDE)/DIVD(ISIDE)
      ZZ=DIVM(ISIDE)/DIVD(ISIDE)
      ZZ2=ZZ*ZZ
      ENDIF
C
      DO 450 I=1,5
      XY=X(I)
      WVAL(I,IREC,IWPN,ISIDE)=WVAL(I,IREC,IWPN,ISIDE)+
     1  XX*XY
      XVAL(I,IREC,IWPN,ISIDE)=XVAL(I,IREC,IWPN,ISIDE)+ZZ*XY
      X2VAL(I,IREC,IWPN,ISIDE)=X2VAL(I,IREC,IWPN,ISIDE)+ ZZ2*XY*XY
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 3 of 12 pages)

```
           PXVAL(I,IREC,IWPN,ISIDE,LPOS)=PXVAL(I,IREC,IWPN,ISIDE,LPOS)
          1 + ZZ*XY
           DXVAL(I,IREC,IWPN,ISIDE,LDAY)=DXVAL(I,IREC,IWPN,ISIDE,
          1 LDAY)+ZZ*XY
           VXVAL(I,IREC,IWPN,ISIDE,LVIS)=VYVAL(I,IREC,IWPN,ISIDE,
          1 LVIS)+ZZ*XY
           XSUM(I,IREC,IENV,IWPN,ISIDE)=XSUM(I,IREC,IENV,IWPN,ISIDE)+
          1 ZZ*XY
           XSQ(I,IREC,IENV,IWPN,ISIDE)=XSQ(I,IREC,IENV,IWPN,ISIDE)+
          1 ZZ2*XY*XY
      450  CONTINUE
           GOTO 100
C
C          HERE IF COP RECORD READ
      500  ZZ=DIVM(ISIDE)/DIVD(ISIDE)
           ZZ2=ZZ*ZZ
C
           DO 550 I=1,5
           XY=X(I)
           COP(I,IREC,ISIDE)=COP(I,IREC,ISIDE)+XY*EWT*ZZ
           XCOP(I,IREC,ISIDE)=XCOP(I,IREC,ISIDE)+ZZ*XY
           X2COP(I,IREC,ISIDE)=X2COP(I,IREC,ISIDE)+ZZ2*XY*XY
           PXCOP(I,IREC,ISIDE,LPOS)=PXCOP(I,IREC,ISIDE,LPOS)+
          1 ZZ*XY
           DXCOP(I,IREC,ISIDE,LDAY)=DXCOP(I,IREC,ISIDE,LDAY)+
          1 ZZ*XY
           VXCOP(I,IREC,ISIDE,LVIS)=VXCOP(I,IREC,ISIDE,LVIS)+
          1 ZZ*XY
           CSUM(I,IREC,IENV,ISIDE)=CSUM(I,IREC,IENV,ISIDE)+ZZ*XY
           CSQ(I,IREC,IENV,ISIDE)=CSQ(I,IREC,IENV,ISIDE)+ZZ2*XY*XY
      550  CONTINUE
C          ISCNT = 61 IS LAST RECORD OF AN ENVIRONMENT
           IF((ISCNT.EQ.61).AND.ELTSW) GOTO 95
           IF(ISCNT.EQ.61) GOTO 90
           GOTO 100
C
C          OUTPUT GRAND REPORT
     2000  IF(ELTSW) THEN
              CALL FACSF('@BRKPT PRINT$')
              CALL FACSF('@BRKPT PRINT$/G6RESULTS')
              CALL FACSF('@FREE WARNING.')
           ENDIF
           EWTTOT=0.0
           DO 2100 IE=1,16
           IF(ENVSW(IE)) EWTTOT=EWTTOT+ENVWTS(IE)
     2100  CONTINUE
           EWTTOT=1.0/EWTTOT
C
C          FOR WEAPONS BY SIDE
           DO 5000 ISIDE=1,2
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 4 of 12 pages)

```
            DO 4000 IWPN=1,NTYPS(ISIDE)
            DO 3000 IREC=1,4
            CALL OUTREX(ISCNTW(IREC,ISIDE),IWPN,WVAL(1,IREC,IWPN,
     1      ISIDE))
 3000       CONTINUE
 4000       CONTINUE
 5000       CONTINUE
C
C           FOR COPS BY SIDE
C
            DO 7000 ISIDE=1,2
            DO 6000 IREC=1,2
            CALL OUTREX(ISCNTW(IREC+4,ISIDE),0,COP(1,IREC,ISIDE))
 6000       CONTINUE
 7000       CONTINUE
C
C
            WRITE(6,5)
    5       FORMAT('1SCORES, CIPS, & COPS BY POSTURE--'/)
            IF(IREP.GT.1) CALL STRATS(ENVWTS)
C
C             OUTPUT WEAPONS BY POSTURE, DAY/NIGHT, CLEAR/DEGRADED
C
            DO 10000 ISIDE=1,2
            DO 9900 IWPN=1,NTYPS(ISIDE)
            AUS2=.FALSE.
            DO 9800 IREC=1,4
            AUS1=.FALSE.
            DO 9700 KPOS=1,4
            CALL OUTPOS(ISCNTW(IREC,ISIDE),IWPN,PXVAL(1,IREC,IWPN,
     1      ISIDE,KPOS),KPOS,POSCNT,AUS1,AUS2,PXVAL(5,IREC,IWPN,
     2      ISIDE,2),ISIDE)
 9700       CONTINUE
            IF(AUS1) WRITE(6,7)
            AUS1=.FALSE.
            DO 9720 KDAY=1,2
            CALL OUTDAY(ISCNTW(IREC,ISIDE),IWPN,DXVAL(1,IREC,
     1      IWPN,ISIDE,KDAY),KDAY,DAYCNT,AUS1,AUS2)
 9720       CONTINUE
            IF(AUS1) WRITE(6,7)
            AUS1=.FALSE.
            DO 9740 KVIS=1,2
            CALL OUTVIS(ISCNTW(IREC,ISIDE),IWPN,VXVAL(1,IREC,
     1      IWPN,ISIDE,KVIS),KVIS,VISCNT,AUS1,AUS2)
 9740       CONTINUE
            IF(AUS1) THEN
             WRITE(6,7)
    7        FORMAT(1CH ----------)
             CALL OUTSTT(ISCNTW(IREC,ISIDE),IWPN,XVAL(1,
     1        IREC,IWPN,ISIDE),X2VAL(1,IREC,IWPN,ISIDE),N)
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 5 of 12 pages)

```
         IF(IREP.GT.1) CALL OUTWT(ISCNTW(IREC,ISIDE),IWPN,
     1    WVAL(1,IREC,IWPN,ISIDE),XSTD(1,IREC,IWPN,ISIDE))
         WRITE(6,6)
        ENDIF
 9800   CONTINUE
        IF(AUS2) WRITE(6,6)
 9900   CONTINUE
10000   CONTINUE
    6   FORMAT(' ')
C
C         OUTPUT COPS BY POSTURE, DAY/NIGHT, CLEAR/DEGRADED
C
        DO 11000 ISIDE=1,2
        DO 10900 IREC=1,2
        DO 10800 KPOS=1,4
        CALL OUTPOS(ISCNTW(IREC+4,ISIDE),0,PXCOP(1,IREC,ISIDE,
     1  KPOS),KPOS,POSCNT,AUS1,AUS2,PXCOP(5,IREC,ISIDE,2),ISIDE)
10800   CONTINUE
        WRITE(6,7)
        DO 10820 KDAY=1,2
        CALL OUTDAY(ISCNTW(IREC+4,ISIDE),0,DXCOP(1,IREC,
     1  ISIDE,KDAY),KDAY,DAYCNT,AUS1,AUS2)
10820   CONTINUE
        WRITE(6,6)
        DO 10840 KVIS=1,2
        CALL OUTVIS(ISCNTW(IREC+4,ISIDE),0,VXCOP(1,IREC,
     1  ISIDE,KVIS),KVIS,VISCNT,AUS1,AUS2)
10840   CONTINUE
        WRITE(6,7)
        CALL OUTSTT(ISCNTW(IREC+4,ISIDE),0,
     1  XCOP(1,IREC,ISIDE),X2COP(1,IREC,ISIDE),N)
        IF(IREP.GT.1) CALL OUTWT(ISCNTW(IREC+4,ISIDE),0,
     1  COP(1,IREC,ISIDE),CSTD(1,IREC,ISIDE))
        WRITE(6,6)
10900   CONTINUE
        WRITE(6,6)
11000   CONTINUE
        CALL BYENV(WVAL,COP)
        STOP 'OK'
C
        SUBROUTINE ZERO(X,N)
        DIMENSION X(N)
        DO 100 I=1,N
        X(I)=0.0
  100   CONTINUE
        RETURN
        END
C
        SUBROUTINE OUTREX(ISCNT,IDWPN,X)
        DIMENSION A(5),X(5)
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 6 of 12 pages)

```
      COMMON/ENVS/EWTTOT
C
      CHARACTER*3 KTHTR
      COMMON/AWRK/IWRK,IDUM,KTHTR,JTPD,JVIS,JPOS,JDAY
      COMMON/GLOBAL/IBFOR,IRFOR,JCASE
      DO 100 I=1,5
      IF(X(I).GT.0.0) GOTO 200
  100 CONTINUE
      RETURN
C
  200 DO 300 I=1,5
  300 A(I)=EWTTOT*X(I)
C
      WRITE(IWRK,1) ISCNT,KTHTR,JTPD,JVIS,JPOS,JDAY,IDWPN,
     1 A(1),A(2),A(3),A(4),A(5),IBFOR,IRFOR,JCASE
      RETURN
C
    1 FORMAT(I5,A3,I4,3I3,I5,5F10.3,2I6,I5)
      END
C
      SUBROUTINE GETREC(ISCNT,IWPN,X,ISOR)
      DIMENSION X(5)
      COMMON/NAME/FNAME
      CHARACTER*20 FNAME
      READ(ISOR,1,END=100,ERR=50,IOSTAT=K) ISCNT,KTHTR,
     1 JTPD,JVIS,JPOS,JDAY,IWPN,X(1),X(2),X(3),X(4),X(5)
   40 IR=IR+1
      RETURN
    1 FORMAT(I5,A3,I4,3I3,I5,5F10.3,2I6,I5)
C
   50 IERR=IOC()
      WRITE(6,3) IERR
    3 FORMAT(' IOERR TYPE=',I6)
      IF(IERR.EQ.1015) GOTO 40
      WRITE(18,2) IR,FNAME
      STOP 'IERR3'
C
  100 WRITE(18,2) IR, FNAME
      STOP 'IERR2'
    2 FORMAT (' LAST RECORD READ = ', I6,2X,A20)
      END
C
      SUBROUTINE GETFIL(FNAME,IENV,DIVM1,DIVD1,DIVM2,DIVD2)
      CHARACTER*20 FNAME
      READ(5,1) IENV,DIVM1,DIVD1,DIVM2,DIVD2,FNAME
      RETURN
    1 FORMAT(I3,4F3.0,2X,A20)
      END
C
      SUBROUTINE OUTSTT(ISCNT,IDWPN,A,B,N)
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 7 of 12 pages)

```
      DIMENSION A(5),B(5),X1(5),X2(5)
C
      DO 100 I=1,5
      IF(A(I).GT.0.0) GOTO 200
  100 CONTINUE
      RETURN
C
  200 RN=N
      SRN=SQRT(RN)
      RN1=(RN-1.0)*RN
      DO 300 I=1,5
      X1(I)=A(I)/RN
      X2(I)=SQRT(ABS((RN*B(I)-A(I)*A(I))))/RN1)/SRN
  300 CONTINUE
C
      WRITE(6,1) ISCNT,'RAW MEANS      ',IDWPN,(X1(I),I=1,5)
      WRITE(6,1) ISCNT,'RAW STD.DEVS.  ',IDWPN,(X2(I),I=1,5)
      RETURN
 1    FORMAT(I5,1X,A15,I5,5F10.3)
      END
C
      SUBROUTINE OUTPOS(ISCNT,IDWPN,A,KPOS,RN4,AUS1,AUS2,REF,ISIDE)
      DIMENSION A(5),X(5),RN4(4),RATS(4,2)
      CHARACTER*5 LOS(2)
      LOGICAL AUS1,AUS2
      DATA RATS/
     1 0.333333,1.0,0.25,3.0,
     2 3.0,1.0,4.0,0.333333/
      DATA LOS/'RLOS=','BLOS='/
C
      DO 100 I=1,5
      IF(A(I).GT.0.0) GOTO 200
  100 CONTINUF
      RETURN
  200 XR=0.0
      IF(RN4(2).GT.0.0) XR=REF/RN4(2)
      DO 300 I=1,5
      X(I)=A(I)/RN4(KPOS)
  300 CONTINUE
      XLOS=X(5)*RATS(KPOS,ISIDE)
      XREF=XR*RATS(KPOS,ISIDE)
      DIF=XREF-XLOS
C
      AUS1=.TRUE.
      AUS2=.TRUE.
      WRITE(6,1) ISCNT,KPOS,IDWPN,(X(I),I=1,5),LOS(ISIDE),
     1 XLOS,XREF,DIF
      RETURN
C
    1 FORMAT(I5,' POSTURE=',I2,' IDWPN=',I3,5F10.3,1X,A5,F10.3,
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 8 of 12 pages)

```
     1 ´ REF=´,F10.3,´ DIFF=´,F10.3)
C
      END
C
      SUBROUTINE OUTDAY(ISCNT,IDWPN,A,KDAY,RN8,AUS1,AUS2)
      DIMENSION A(5),X(5),RN8(2)
      CHARACTER*5 DAY(2)
      LOGICAL AUS1,AUS2
C
      DATA DAY/´DAY  ´,´NIGHT´/
C
      DO 100 I=1,5
      IF(A(I).GT.0.0) GOTO 200
  100 CONTINUE
      RETURN
C
  200 DO 300 I=1,5
      X(I)=A(I)/RN8(KDAY)
  300 CONTINUE
C
      AUS1=.TRUE.
      AUS2=.TRUE.
      WRITE(6,1) ISCNT,DAY(KDAY),IDWPN,(X(I),I=1,5)
      RETURN
C
    1 FORMAT(I5,´ TIME=´,A5,´ IDWPN=´,I3,5F10.3)
C
      END
C
      SUBROUTINE OUTVIS(ISCNT,IDWPN,A,KVIS,RN8,AUS1,AUS2)
      DIMENSION A(5),X(5),RN8(2)
      CHARACTER*8 VIS(2)
      LOGICAL AUS1,AUS2
C
      DATA VIS/´CLEAR   ´,´DEGRADED´/
C
      DO 100 I=1,5
      IF(A(I).GT.0.0) GOTO 200
  100 CONTINUE
      RETURN
C
  200 DO 300 I=1,5
      X(I)=A(I)/RN8(KVIS)
  300 CONTINUE
C
      AUS1=.TRUE.
      AUS2=.TRUE.
      WRITE(6,1) ISCNT,VIS(KVIS),IDWPN,(X(I),I=1,5)
      RETURN
C
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 9 of 12 pages)

```
    1 FORMAT(I5,3X,A8,' IDWPN=',I3,6F10.3)
C
      END
C
      SUBROUTINE STRATS(ENVWTS)
C        G.E.C. -- 4/28/84
      COMMON/STRAT/IREP,XSUM(5,4,16,60,2),XSQ(5,4,16,60,2),
     1 CSUM(5,2,16,2),CSQ(5,2,16,2),XSTD(5,4,60,2),CSTD(5,2,2)
      DIMENSION ENVWTS(16)
      R=FLOAT(IREP)
      R1=FLOAT(IREP-1)
      RR1=R*R1
C
      DO 3000 IS=1,2
      DO 2000 IE=1,16
      E=ENVWTS(IE)**2
      DO 1000 IC=1,5
      DO 800 IREC=1,4
      DO 700 IW=1,60
      XSTD(IC,IREC,IW,IS)=XSTD(IC,IREC,IW,IS)+E*
     1 (XSQ(IC,IREC,IE,IW,IS)-(XSUM(IC,IREC,IE,IW,IS)**2)/R)
  700 CONTINUE
  800 CONTINUE
      DO 900 ICOP=1,2
      CSTD(IC,ICOP,IS)=CSTD(IC,ICOP,IS)+E*
     1 (CSQ(IC,ICOP,IE,IS)-(CSUM(IC,ICOP,IE,IS)**2)/R)
  900 CONTINUE
 1000 CONTINUE
 2000 CONTINUE
      DO 2900 IC=1,5
      DO 2800 IREC=1,4
      DO 2700 IW=1,60
      XSTD(IC,IREC,IW,IS)=SQRT(ABS(XSTD(IC,IREC,IW,IS)/RR1))
 2700 CONTINUE
 2800 CONTINUE
      DO 2850 ICOP=1,2
      CSTD(IC,ICOP,IS)=SQRT(ABS(CSTD(IC,ICOP,IS)/RR1))
 2850 CONTINUE
 2900 CONTINUE
 3000 CONTINUE
      RETURN
      END
C
      SUBROUTINE OUTWT(ISCNT,IDWPN,A,B)
C        G.E.C. -- 4/28/84
      DIMENSION A(5),B(5)
C
      DO 100 I=1,5
      IF(A(I).GT.0.0) GOTO 200
  100 CONTINUE
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 10 of 12 pages)

```
      RETURN
C
  200 WRITE(6,1) ISCNT,'WTD. MEANS      ',IDWPN,(A(I),I=1,5)
      WRITE(6,1) ISCNT,'WTD.STD.DEVS.   ',IDWPN,(B(I),I=1,5)
      RETURN
    1 FORMAT(I5,1X,A15,I5,5F10.3)
      END
C
      SUBROUTINE BYENV(WVAL,COP)
C      G.E.C. -- 5/24/84
      COMMON/STRAT/IREP,XSUM(5,4,16,60,2),XSQ(5,4,16,60,2),
     1 CSUM(5,2,16,2),CSQ(5,2,16,2),XSTD(5,4,60,2),CSTD(5,2,2)
      DIMENSION WVAL(5,4,60,2),SCRTCH(16),VCRTCH(16),COP(5,2,2)
      LOGICAL SDEVS
      SDEVS=.FALSE.
      IF(IREP.GT.1) SDEVS=.TRUE.
      R2=0.0
      R=FLOAT(IREP)
      R1=R-1.0
      IF(SDEVS) R2=1.0/(R*R1)
      CALL EHEAD
C      MCIPS
      DO 3000 IS=1,2
      DO 1000 IW=1,60
      DO 900 IE=1,16
      X=XSUM(5,4,IE,IW,IS)
      X2=XSQ(5,4,IE,IW,IS)
      SCRTCH(IE)=X/R
      IF(SDEVS) VCRTCH(IE)=SQRT(ABS(R2*(R*X2-X*X)))
  900 CONTINUE
      CALL ELINE(SDEVS,IW,IS,SCRTCH,VCRTCH,WVAL(5,4,IW,IS))
 1000 CONTINUE
 3000 CONTINUE
C      MCOPS
      DO 4000 IS=1,2
      DO 3900 IE=1,16
      X=CSUM(5,2,IE,IS)
      X2=CSQ(5,2,IE,IS)
      SCRTCH(IE)=X/R
      IF(SDEVS) VCRTCH(IE)=SQRT(ABS(R2*(R*X2-X*X)))
 3900 CONTINUE
      CALL CLINE(SDEVS,IS,SCRTCH,VCRTCH,COP(5,2,IS))
 4000 CONTINUE
      RETURN
      END
C
      SUBROUTINE EHEAD
C      G.E.C. -- 5/24/84
      WRITE(6,1) (I,I=1,16)
      RETURN
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 11 of 12 pages)

```
    1 FORMAT('1',16X,16(I3,3X))
      END
C
      SUBROUTINE ELINE(SDEVS,IW,IS,X,Y,Z)
C      G.E.C. -- 5/24/84
      DIMENSION X(16),Y(16)
      LOGICAL SDEVS
      COMMON/LINE/N
      DO 100 I=1,16
      IF(X(I).GT.0.0) GOTO 200
  100 CONTINUE
      RETURN
C
  200 NLINE=NLINE+2
      IF(NLINE.GE.80) THEN
       NLINE=0
       CALL EHEAD
      ENDIF
      WRITE(6,1) IS,IW,(X(I),I=1,16),Z
    1 FORMAT(/' SIDE=',I2,' WPN=',I2,17F6.2)
      IF(.NOT.SDEVS) RETURN
      NLINE=NLINE+1
      WRITE(6,2) (Y(I),I=1,16)
    2 FORMAT('   STD.DEVS.      ',16F6.2)
      END
C
      SUBROUTINE CLINE(SDEVS,IS,X,Y,Z)
C      G.E.C. -- 6/1/84
      DIMENSION X(16),Y(16),KX(16),KY(16)
      LOGICAL SDEVS
      COMMON/LINE/N
      DO 100 I=1,16
      IF(X(I).GT.0.0) GOTO 200
  100 CONTINUE
      RETURN
C
  200 NLINE=NLINE+2
      IF(NLINE.GE.80) THEN
       NLINE=0
       CALL EHEAD
      ENDIF
      DO 1000 I=1,16
      KX(I)=X(I)
 1000 CONTINUE
      KZ=Z
      WRITE(6,1) IS,(KX(I),I=1,16),KZ
    1 FORMAT(/' SIDE=',I2,' MCOP= ',17I6)
      IF(.NOT.SDEVS) RETURN
      NLINE=NLINE+1
      DO 2000 I=1,16
      KY(I)=Y(I)
 2000 CONTINUE
      WRITE(6,2) (KY(I),I=1,16)
    2 FORMAT('   STD.DEVS.      ',16I6)
      END
```

Figure G-11. Source Listing of the Main and Subprograms of the AFP Rollup and Stats Module (page 12 of 12 pages)

(6) **X2COP(5,2,2).** The squares of the intermediate partial combat potentials are accumulated within this array for use in the determination of standard deviations across all 16 combat environments for divisions. The indexing of elements X2COP(I,J,K) is exactly the same as for COP() and XCOP() above.

(7) **PXVAL(5,4,60,2,4).** Stratified simple arithmetic mean combat potentials for each weapon type for each combat posture are developed within this array. For the first four indices I, J, K, and L, the indexing of elements PXVAL(I,J,K,L,M) is exactly the same as for WVAL() above. PXVAL() possesses a fifth index, M. M=1 to 4 for the four combat postures: RAPD, STATIC, RADE, and BAPD.

(8) **PXCOP(5,2,2,4).** Stratified simple arithmetic mean combat potentials for each division for each combat posture are developed within this array. For the first three indices I, J, and K, the indexing of elements of PXCOP() is exactly the same as for COP() above. PXCOP() possesses a fourth index, K. K=1 to 4 for the four combat postures: RAPD, STATIC, RADE, and BAPD.

(9) **DXVAL(5,4,60,2,2).** Stratified simple arithmetic mean combat potentials for each weapon type for each of daytime and nighttime are developed within this array. The indices I, J, K, and L have the same significance as for WVAL() above. The fifth index of DXVAL(I,J,K,L,M) is M. M=1 to 2 for the two diurnal conditions: daytime and nighttime.

(10) **DXCOP(5,2,2,2).** Stratified simple arithmetic mean combat potentials for each division for daytime and nighttime are developed within this array. The indices I, J, and K have the same significance as for COP() above. The fourth index of DXCOP(I,J,K,L) is L. L=1 to 2 for the two diurnal conditions: daytime and nighttime.

(11) **VXVAL(5,4,60,2,2).** Stratified simple arithmetic mean combat potentials for each weapon type for each visibility condition are developed within this array. The indices I, J, K, and L have the same significance as for WVAL() above. The fifth index of VXVAL(I,J,K,L,M) is M. M=1 to 2 for the two visibility conditions: clear and degraded.

(12) **VXCOP(5,2,2,2).** Stratified simple arithmetic mean combat potentials for each division for each visibility condition are developed within this array. The indices I, J, and K have the same significance as for COP above. The fourth index of VXCOP(I,J,K,L) is L. L=1 to 2 for the two visibility conditions: clear and degraded.

(13) **XSUM(5,4,16,60,2).** For weapons, array XSUM() stores sums of partial combat potentials by combat environment for use in computations of weighted stratified variances over two or more replications. The array XSUM() generalizes the array WVAL(), above, by the inclusion of an additional index, JE. An array element XSUM(I,J,JE,K,L) is indexed similarly to WVAL(I,J,K,L) with the addition that JE = 1 to 16 for combat environments.

(14) XSQ(5,4,16,60,2). For weapons, array XSQ() stores sums of squares of partial combat potentials by combat environment for use in computations of weighted stratified variances over two or more replications. An array element XSQ(I,J,JE,K,L) is indexed exactly as is XSUM() immediately above.

(15) XSTD(5,4,60,2). For weapons, array XSTD() stores weighted, stratified standard deviations of partial combat potentials as developed from arrays XSUM() and XSTD(). An array element XSTD(I,J,K,L) is indexed exactly as WVAL() above.

(16) CSUM(5,2,16,2). For divisions, array CSUM() stores sums of partial COPs by combat environment for use in computations of weighted, stratified variances in COPs over two or more replications. The array CSUM() generalizes the array COP() by the inclusion of an additional index, JE. An array element CSUM(I,J,JE,K) is indexed similarly to COP(I,J,K) with the addition of JE = 1 to 16 for combat environments.

(17) CSQ(5,2,16,2). For divisions, array CSQ() stores sums of squares of COPs by combat environment for use in computations of weighted, stratified variances in COPs over two or more replications. An array CSQ(I,J,JE,K) is indexed exactly as is CSUM() immediately above.

(18) CSTD(5,2,2). For divisions, array CSTD() stores weighted standard deviations of COPs as developed from arrays CSUM() and CSQ(). An array element CSTD(I,J,K) is indexed exactly as COP() above.

b. Several small arrays are used to store needed data.

(1) ISCNTR(6,2). The identifiers of record types within the intermediate partial combat potential files are stored in this array for comparison with input records read. An array element ISCNTR(I,J) is indexed:

(a) I=1 to 6 for the input record type: unmodulated score, unmodulated CIP, modulated score, modulated CIP, unmodulated COP, and modulated COP. All these refer on input, of course, to partial combat potentials.

(b) J=1 to 2 for sides: Blue and Red.

(2) ISCNTW(6,2). The identifiers of record types output to the final combat potentials file are stored for reference and use during output. An array element ISCNTW(I,J) is indexed:

(a) I=1 to 6 for the output record type: unmodulated score, unmodulated CIP, modulated score, modulated CIP, unmodulated COP, and modulated COP.

(b) J=1 to 2 for sides: Blue and Red.

(3) ENVWTS(16). The weights to be used in summing partial combat potentials over the 16 combat environments are input to this array for reference and used during the actual weighted summing. The weights should be nonnegative and sum to 1.0. An array element ENVWTS(I) is indexed: I=1 to 16 for the normal combat environments, consecutively.

(4) JRECS(12). This array stores identifiers linking the input record types to the indices of the kinds of potentials being developed. There are 12 input record identifiers: 10, 30, 50, 70, 11, 51, 20, 40, 60, 80, 21, and 61. The elements of JRECS() have values: 1, 2, 3, 4, 1, 2, 1, 2, 3, 4, 1, and 2. The first through fourth and seventh through tenth JRECS() values provide the indices to the unmodulated score, unmodulated CIP, modulated score, and modulated CIP positions within the weapon-related accumulator arrays described above. The fifth, sixth, eleventh, and twelfth JRECS() values provide the indices to the unmodulated COP and modulated COP positions within the division-related accumulator arrays described above.

(5) MDAY(16). The arrays store a "1" for daytime and a "2" for nighttime in those elements corresponding to the normal sequence of combat environments.

(6) DIVM(2). This array provides temporary storage of an adjusting multiplier for each of Red and Blue during the processing of an intermediate partial combat potential file. An element from DIVM() is always used with a corresponding element from the sister array DIVD().

(7) DIVD(2). This array provides temporary storage of an adjusting divisor for each of Red and Blue during the processing of an intermediate partial combat potential file. An element from DIVD() is always used with a corresponding element from the sister array DIVM(). Partial potentials, P, are adjusted to new values Q in accord with the transformation Q = P * DIVM(I)/DIVD(I), where I indexes the side. As noted earlier in this appendix, the feature is provided in the event that an input file contains partial potentials for more than a single division. When the potentials do reflect a single division, both DIVM(I) and DIVD(I) should be 1.0.

c. X(5) serves as the program buffer for repeatedly receiving five-valued sets of intermediate partial potentials during the reading of the many input files. Values temporarily stored in X() are transferred to the appropriate accumulator arrays during processing.

G-15. The AFPSTATS source listing in Figure G-11 includes some intralinear comments. The following paragraphs provide some additional commentary.

a. Lines 8-28 in Figure G-11 provide the needed declarative statements, mostly for the arrays described above. Some scalar character and logical variables are also declared.

b. Lines 29-41 initialize reference arrays with identifiers and indices needed for correct processing and accumulation of input partial combat potentials.

c. Line 44 begins the executable statements. Lines 44-45 initialize several variables from values in the input stream. JVIS, JPOS, and JDAY, by convention, are zero-valued identifiers in final combat potential output records. The final potentials are weighted over all environments; "0" symbolizes all environments.

d. Line 47 initializes the "files read" counter N.

e. Lines 49-53. Logical unit 30 is the output file for final combat potentials. Logical unit 29 is the source of input partial combat potential files. Logical unit 6 is the standard FORTRAN/UNIVAC print file. "DONE" is an end-of-input files marker.

f. Lines 55-72 zero the accumulator arrays.

g. Lines 74-75 provide for the input of the 16 combat environmental weights, unformatted.

h. Lines 78-81 get the single-valued quantity specifying the number of replications (that is, random number seeds), and hence, the number of file sets to be input. The lines also get the logical value ELTSW specifying whether partial potentials are to be read as elements. If elements are to be read, the source is set to logical unit #5. Records within the elements contain more than 80 characters. All UNIVAC system warnings to that effect are diverted to temporary file WARNING.

i. Line 82. The first time through, there is no need to close unit 29.

j. Line 85. Gets the name of a file to be read, the combat environment to which it corresponds, and the Blue and Red division adjustment factors (usually 1.0). The name may be the end-of-files marker, "DONE".

k. Line 86. Checks to see whether all the files have been read. If so, jump to begin output of final combat potentials.

l. Lines 87-88. Set EWT to the corresponding combat environmental weight divided by the number of replications (seeds). Set switch for the corresponding environment to .TRUE.

m. Line 89. Increments the counter of files read, even though have not read any of the current file yet!

n. Lines 90-96

(1) Set LPOS to the combat posture of the current input file. The expression given returns the correct posture over combat environments 1 to 16.

(2) **Lines 91-95.** Set LVIS to 1 if clear and 2 if degraded visibility.

(3) **Line 96.** Sets LDAY to 1 if daytime and to 2 if nighttime.

o. Lines 97-99. Increment stratified environment counters.

p. Lines 100-102. Save and print the current file name. Name is printed in the event something fails; operator then can tell how far along the module had run before trouble hit.

q. Lines 103-107. Concatenate an EXEC level instruction depending on whether files or elements are to be read.

r. Line 102. Calls FACSF to attach the physical file to logical unit 29 or add the physical element to logical unit 5.

s. Line 110. Gets a record from the current file.

t. Lines 111-115. Determine what type of record was just read.

u. Lines 116-120. Here only if record was not identifiable. Go read another record only if fewer than 10 such failures have occurred.

v. Lines 122-127. Line 122 sets the index of the kind of partial combat potential supplied in the input record. Lines 123-127 set the side index for Blue (1) or Red (2).

w. Line 129. Line 129 checks whether the record applies to an entire division; and, if so, transfers control to statement 500 for division processing.

x. Lines 135-162. This section is for the processing of the partial combat potentials of a weapon type.

(1) Weapons scores are adjusted, but CIPs do not require adjustment. The arithmetic applied later is the same for both scores and CIPs. But the adjustment factors ZZ and ZZ2 are left at 1.0 for CIPs (lines 135 and 136).

(2) An even-numbered record index corresponds to a CIP. Odd numbers correspond to scores. Hence, the IF clause (lines 137-143) simply sets the variable XX to the environmental weight for use with CIPs but modifies the CIPs and the adjustment factors with DIVM (ISIDE) and DIVD (ISIDE). The latter two factors are usually 1.0 anyway.

(3) Lines 145-161 loop over the components of five-valued partial combat potential. Each component is stored temporarily in variable XY. Then with or without modification by ZZ or ZZ2, as appropriate, XY or (XY * XY) is added to the corresponding elements of the accumulator arrays in succession.

(4) Line 162 transfers control to retrieve another record.

y. Lines 165-185. This section is for the processing of partial combat potentials of a division.

(1) Set the ZZ and ZZ2 division strength modifiers in lines 165 and 166. The modifiers are usually 1.0.

(2) Lines 168-181 loop over the components of five-valued partial COPs. Each component is stored temporarily in variable XY. Then, with modification by ZZ or ZZ2 as appropriate, XY or (XY * XY) is added to the corresponding elements of the division accumulator arrays in succession.

(3) A record of type 61 is the last record for an environment, and hence, the last record in a partial combat potentials file. In line 184 the record type is checked, and, if it is type 61, control is transferred to get another file name. If the record is not of type 61, line 185 transfers control to get another record within the same file.

z. Lines 188-215. This section outputs final combat potentials for weapons and entire divisions record-by-record to the output file.

(1) Lines 188-192, if partial combat potentials have been input as elements, direct further output to the file G6RESULTS and release the superfluous warnings accumulated to this point. If a fatal error occurs earlier, run output continues to go to file WARNINGS.

(2) Lines 193-196 form the sum of weights only for the results of combat environments included input.

(3) Lines 200-207 output weapon potentials for each of four kinds of potential, for each weapon type, and for each side. A call to OUTREX outputs a single record. The starting address of a five-valued potential is sent to OUTREX as a corresponding address within WVAL().

(4) Lines 211-215 output division potentials for each of two kinds of COP for each side. A call to OUTREX outputs a single record. The starting address of a five-valued potential is sent to OUTREX as a corresponding address with COP().

aa. Lines 218-258. This section outputs stratified simple arithmetic means of weapon combat potentials to the standard print file. The outer loop (lines 224 and 258) is over Blue and Red. The next inner loop (lines 225 and 257) is over weapon types. Lines 229-233 output the means within each combat posture via calls to OUTPOS. Lines 236-239 output the means for daytime and nighttime via calls to OUTDAY. Lines 242-245 output the means for clear and degraded visibility via calls to OUTVIS. All the subroutine calls include an argument giving the address of five-valued potentials within the corresponding accumulator arrays. A call to OUTSTT (line 249) outputs the mean and standard deviation over all (nonstratified) combat environments. If more than one replication has been processed, a call to OUTWT outputs stratified means and standard deviations (line 251).

ab. Lines 263-287. This section outputs stratified divisional potentials (COPs) to the standard print file. The outer loop (lines 263 and 287) is over Blue and Red. The next inner loop (lines 264 and 285) is over unmodulated and modulated COPs. Lines 265-268 output the means within each combat posture via four calls to OUTPOS. Lines 270-273 output the means for daytime and nighttime via two calls to OUTDAY. Lines 275-278 output the means for clear and degraded visibility via two calls to OUTVIS. All the subroutine calls include an argument giving the address of five-valued potentials within the corresponding accumulator arrays. A call to OUTSTT (line 280) outputs the mean and standard deviation over all (nonstratified) combat environments. If more than one replication has been processed, a call to OUTWT outputs stratified means and standard deviations (line 282).

ac. Line 288 calls subroutine BYENV to calculate and display modulated scalar CIPs by combat environment, by weapon, and by side. If two or more replications have been rolled up, standard deviations by combat environment also are displayed.

ad. Line 289 normally terminates execution of the main program of the Rollup and Stats Module.

G-16. The remaining lines of Figure G-11 provide source listings of the subroutines called by the main program of the AFP Rollup and Stats Module. The following paragraphs provide brief commentary on the subroutines.

a. Lines 291-297. The subroutine ZERO simply fills a real array with zeros.

b. Lines 299-319. Subroutine OUTREX outputs a single record of five-valued potential to the final combat potentials file. The routine is used to output both weapon and division potentials. The routine does not output a record if all five components of combat potential are zero.

(1) Argument ISCNT is the identifier of record type.

(2) Argument IDWPN is the identifier of the weapon type. A division is identified as a weapon of type 0.

(3) Argument X is a five-component array containing the five-valued combat potential to be output.

c. Lines 321-341. Subroutine GETREC reads a record from a partial combat potentials file. If an "unexpected" end-of-file is encountered, the subroutine terminates execution of the entire module. Several of the fields read from a record are not used. Errors of type 1015 are ignored; they are simply "long record" warnings; no data beyond 80 characters are needed.

(1) Argument ISCNT is the record type identifier and is returned by the subroutine.

(2) Argument IWPN is the identifier of the weapon type and is returned by the subroutine.

(3) Argument X is a five-element real array which receives the five-valued partial potential from a record and is returned by the subroutine.

(4) Argument ISOR is passed to the subroutine as the number of the logical unit of the input file being read.

d. Lines 343-345. Subroutine GETFIL reads an input file name, the combat environment index, and the Blue and Red division quantity modifiers.

(1) Argument FNAME returns the input file name or the end-of-file name marker "DONE."

(2) Argument IENV returns the index (1 to 16) of the combat environment corresponding to the file name.

(3) Argument DIVM1 returns the Blue division quantity multiplier modifier.

(4) Argument DIVD1 returns the Blue division quantity divisor modifier.

(5) Argument DIVM2 returns the Red division quantity multiplier modifier.

(6) Argument DIVD2 returns the Red division quantity divisor modifier.

e. Lines 350-370. Subroutine OUTSTT completes the computation of simple, unweighted means and standard deviations of the components of one kind of combat potential for a weapon or division and outputs the results. Lines 333-335 assure that at least one component is nonzero before continuing computation; otherwise, OUTSTT returns without outputting anything to the statistical report. Lines 358-364 complete computation of mean and standard deviation for each component of potential. Lines 366-367 output the mean and standard deviation for each component.

(1) Argument ISCNT is the identifier of the kind of potential to be processed and output.

(2) Argument IDWPN is the identifier of the weapon type. A "0" denotes a division, not a single weapon type.

(3) Argument A is the address of the five-element real array containing the sums by combat potential component over all combat environments and random number seed sets.

(4) Argument B is the address of the five-element real array containing the sums of the squares by combat potential component over all environments and random number seed sets.

(5) Argument N is the number of quantities summed in accumulating the elements of arrays A() and B(). N, more simply, is the number of input files.

f. Lines 372-403. Subroutine OUTPOS completes the computation of simple, unweighted means of the components of one kind of combat potential for one combat posture for a weapon or division and outputs the results. Lines 381-383 assure that at least one component is nonzero before continuing computation; otherwise, OUTPOS returns without outputting anything to the statistical report. Lines 385-389 complete the computation of the mean for each component of potential. Lines 394-395 turn on logical switches for underlining and line feed control within the main program, and output the mean for each component.

(1) Argument ISCNT is the identifier of the kind of potential to be processed and output.

(2) Argument IDWPN is the identifier of the weapon type. A "0" denotes a division not a single weapon type.

(3) Argument A is the address of the five-element real array containing the sums by combat potential component over the corresponding combat posture and random number seed sets.

(4) Argument KPOS is the identifier of the corresponding combat posture.

(5) Argument RN4 is the number of quantities summed in accumulating the elements of array A().

(6) Arguments AUS1 and AUS2 are returned as "TRUE" if OUTPOS does output values. The logical variables are provided to control underlining and line feed within the main program.

g. Lines 405-428. Subroutine OUTDAY completes the computation of simple, unweighted means of the components of one kind of combat potential for daytime and nighttime for a weapon or division and outputs the results. Lines 412-414 assure that at least one component is nonzero before continuing computation; otherwise, OUTDAY returns without outputting anything to the statistical report. Lines 417-419 complete the computation of the mean for each component of potential. Lines 421-423 turn on logical switches for underlining and line feed control within the main program, and output the mean for each component.

(1) Argument ISCNT is the identifier of the kind of potential to be processed and output.

(2) Argument IDWPN is the identifier of the weapon type. A "0" denotes a division not a single weapon type.

(3) Argument A is the address of the five-element real array containing the sums by combat potential component over the corresponding daytime or nighttime condition and random number seed sets.

(4) Argument KDAY is the identifier of the corresponding daytime or nighttime condition.

(5) Argument RN8 is the number of quantities summed in accumulating the elements of array A().

(6) Arguments AUS1 and AUS2 are returned as "TRUE" if OUTDAY does output values. The logical variables are provided to control underlining and line feed within the main program.

h. Lines 430-453. Subroutine OUTVIS completes the computation of simple, unweighted means of the components of one kind of combat potential for clear or degraded visibility for a weapon or division and outputs the results. Lines 437-439 assure that at least one component is nonzero before continuing computation; otherwise, OUTVIS returns without outputting anything to the statistical report. Lines 442-444 complete the computation of the mean for each component of potential. Lines 446-448 turn on logical switches for underlining and line feed control within the main program, and output the mean for each component of potential.

(1) Argument ISCNT is the identifier of the kind of potential to be processed and output.

(2) Argument IDWPN is the identifier of the weapon type. A "0" denotes a division not a single weapon type.

(3) Argument A is the address of the five-element real array containing the sums by combat potential component over the corresponding clear or degraded visibility and random number seed sets.

(4) Argument KVIS is the identifier of the corresponding clear or degraded visibility condition.

(5) Argument RN8 is the number of quantities summed in accumulating the elements of array A().

(6) Arguments AUS1 and AUS2 are returned as "TRUE" if OUTVIS does output values. The logical variables are provided to control underlining and line feed within the main program.

i. Lines 455-492. Subroutine STRATS completes the computation of weighted stratified standard deviations of scores, CIPs, and COPs. Argument ENVWTS is the address of the 16-element real array containing combat environmental weights. The subroutine's loop structures construct the standard deviations for both sides, for all components of scores, CIPs, and COPS, and for all weapons. For any one component of combat potential, the weighted stratified mean is of the form:

$$E(x) = \frac{1}{R} \sum_{i=1}^{16} w_i \sum_{j=1}^{R} x_{ij}$$

and the weighted stratified standard deviation is of the form:

$$(\text{Std. Dev. } (E(x)))^2 = \frac{1}{R} \sum_{i=1}^{16} w_i^2 \left(\frac{1}{R(R-1)} \left(R \sum_{j=1}^{R} x_{ij}^2 - \left(\sum_{j=1}^{R} x_{ij}\right)^2\right)\right)$$

where:

x_{ij} = the value of the variate for the jth replication in the ith combat environment

w_i = the weight for combat environment i

R = the replications (this must be the same for all combat environment)

j. Lines 494-507. Subroutine OUTWT outputs a set of stratified means and standard deviations.

k. Lines 509-545. Subroutine BYENV computes the means and, if more than two replications have been rolled up, the standard deviations of modulated scalar CIPs and COPs by combat environment by weapon type by side. BYENV calls subroutine ELINE to output one or two (plus blank) report lines for each weapon type. The argument WVAL is the address of the array where weighted mean scalar CIPs (among other values) have been stored. Argument COP is the address of the COPs only.

l. Lines 547-552. Subroutine EHEAD advances a page and writes the row "12...16" as the simple heading within CIPs by combat environment reports.

m. Lines 554-575. Subroutine ELINE is called by subroutine BYENV to output report lines for a weapon type. A "CIP line" is output only if at least one mean CIP is nonzero. A "standard deviation line" is output only if a CIP line has been output and if two or more replications have been rolled up.

n. Lines 577-605. Subroutine CLINE is called by subroutine BYENV to output COP lines for an entire division. A "standard deviation line" is output only if two or more replications have been rolled up.

APPENDIX H

THE AFP DIVISION COMPARE REPORTER

H-1. OVERVIEW

a. The AFP Division Compare Reporter is designed to:

(1) Accept final type weapon combat potentials (scores and CIPs) from the AFP Rollup and Stats Module or from the Interpolation Module for six "different" divisions. "Different" divisions may mean different versions of the same division, stages during transition from an old to modernized division inventory, or different kinds of divisions. Although the module's viewpoint is primarily for Blue divisions, the potentials of threat weapons are also available from those files produced by the Rollup and Stats Module (but not from the Interpolation Module).

(2) Accept operator-supplied lists of identifiers of weapons to be compared within subreports. Blue and Red weapons may be compared within a single subreport.

(3) Output the requested subreports.

b. The Division Compare Reporter provides the basis for comparison of the variations among weapon potentials from random number seed to random number seed, from weapon to weapon, from division to division, and from side to side (Blue versus Red). The Reporter is limited in that six input files are required. Hence, the Reporter cannot provide a comparison until six sets of combat potentials are available from other modules.

c. The relation of the AFP Division Compare Reporter to the AFP System in general is portrayed in Figure H-1. There the Reporter is highlighted by being enclosed in an oval. Figure H-1 oversimplifies the requirement for input to the Reporter by making it appear that only two input files are required: one file for a Division A and a second file for a Division B. As noted above, six files are required.

H-2. INPUT. Paragraph H-1 above set the stage for discussion of the Reporter's input requirements. The AFP operator must supply six file names. The files must contain final combat potentials in standard AFP output record format with correct record identifiers. Files produced by the AFP Rollup and Stats Module or by the Interpolation Module meet these requirements. The operator must provide two symbols identifying the two columns in subreport headings. The operator must also provide lists of weapon and side identifiers to control which weapons appear in subreports. The number of such lists also controls the number of subreports generated.

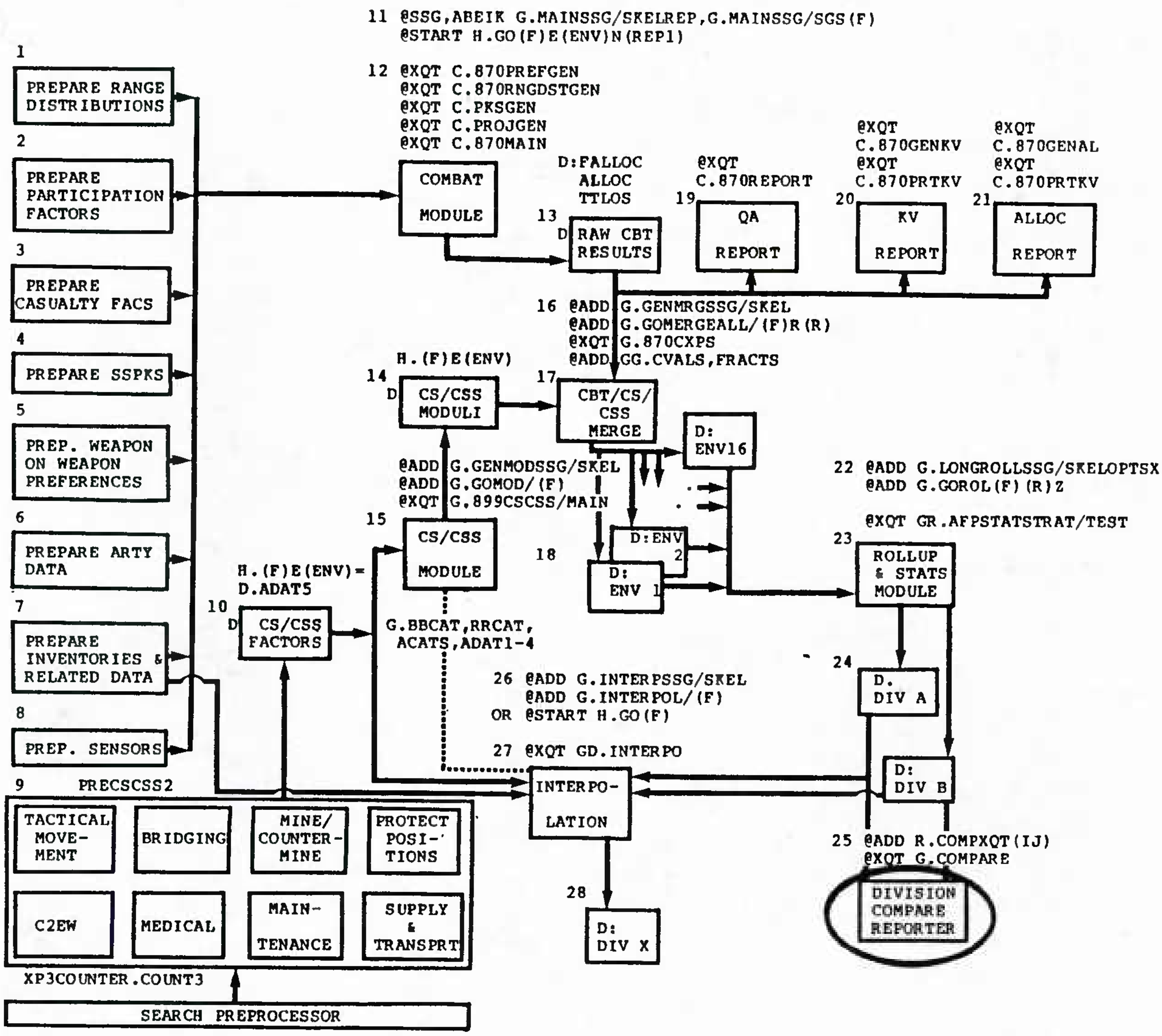


Figure H-1. Relation of the AFP Division Compare Reporter to the AFP System in General

a. Figure H-2 displays extracts from an acceptable input file containing final combat potentials for a single division (and threat). The example is the same as shown for the output of the AFP Rollup and Stats Module in Figure G-3 of Appendix G. The fields of records are described in paragraph 14 of Appendix B. In an output file from the Rollup and Stats Module or from the Interpolation Module, record identifiers (Field 1) should be greater than 100. Five-valued combat potentials are contained in the input records.

FIELD

	1	2	3	4	5	6	7	8	9
25.	110	E 1 0 0 0	16	991.096	67.506	90.345	.000	21.586	1 1 1
26.	130	E 1 0 0 0	16	4.737	.327	.436	.000	.104	1 1 1
27.	150	E 1 0 0 0	16	990.053	67.287	90.312	.000	21.555	1 1 1
28.	170	E 1 0 0 0	16	4.779	.325	.435	.000	.104	1 1 1
29.	110	E 1 0 0 0	17	524.247	35.921	45.623	.000	11.156	1 1 1
30.	130	E 1 0 0 0	17	4.628	.318	.400	.000	.098	1 1 1
31.	150	E 1 0 0 0	17	523.781	35.855	45.722	.000	11.161	1 1 1
32.	170	E 1 0 0 0	17	4.621	.317	.401	.000	.098	1 1 1
33.	110	E 1 0 0 0	20	2669.735	157.594	180.532	3.791	51.092	1 1 1
34.	130	E 1 0 0 0	20	7.992	.473	.538	.011	.153	1 1 1
35.	150	E 1 0 0 0	20	2673.525	157.388	180.920	4.131	51.470	1 1 1
36.	170	E 1 0 0 0	20	8.000	.472	.539	.012	.154	1 1 1
37.	110	E 1 0 0 0	26	105.556	4.346	.000	8.111	8.841	1 1 1
38.	130	E 1 0 0 0	26	3.175	.131	.000	.244	.265	1 1 1
39.	150	E 1 0 0 0	26	107.007	4.320	.000	8.807	9.538	1 1 1
40.	170	E 1 0 0 0	26	3.218	.130	.000	.264	.286	1 1 1
					*				
					*				
					*				
169.	120	E 1 0 0 0	51	3.160	.074	.000	.000	.016	1 1 1
170.	140	E 1 0 0 0	51	.129	.003	.000	.000	.001	1 1 1
171.	160	E 1 0 0 0	51	3.329	.079	.000	.000	.017	1 1 1
172.	180	E 1 0 0 0	51	.136	.003	.000	.000	.001	1 1 1
173.	120	E 1 0 0 0	52	45.216	7.383	.013	.000	.902	1 1 1
174.	140	E 1 0 0 0	52	.741	.121	.000	.000	.015	1 1 1
175.	160	E 1 0 0 0	52	47.456	7.748	.013	.000	.947	1 1 1
176.	180	E 1 0 0 0	52	.778	.127	.000	.000	.015	1 1 1
177.	120	E 1 0 0 0	56	19.989	2.611	.047	.000	.335	1 1 1
178.	140	E 1 0 0 0	56	.169	.022	.000	.000	.003	1 1 1
179.	160	E 1 0 0 0	56	20.952	2.749	.049	.000	.352	1 1 1
180.	180	E 1 0 0 0	56	.178	.023	.000	.000	.003	1 1 1
					*				
					*				
					*				
189.	111	E 1 0 0 0	0	11916.452	897.693	489.232	103.601	293.738	1 1 1
190.	151	E 1 0 0 0	0	11919.645	894.350	488.403	112.592	302.274	1 1 1
191.	121	E 1 0 0 0	0	898.871	184.892	12.760	89.643	112.274	1 1 1
192.	161	E 1 0 0 0	0	975.283	193.200	13.275	106.670	131.446	1 1 1

Figure H-2. Example Extract Records of File of Final Combat Potentials Output From the AFP Rollup and Stats Module for Input to the AFP Division Compare Reporter

b. The Reporter operator must input the names of the files to be read by the Reporter. The operator must also identify to the Reporter which of the six needed files corresponds to the name given. For example, the record:

```
               1         2         3         4         5         6
      123456789012345678901234567890123456789012345678901234567890

         4  G6ROLJ16S1TG.
```

directs the Reporter to treat file G6ROLJ16S1TG. as the fourth file. A record of the form:

```
               1         2         3         4         5         6
      123456789012345678901234567890123456789012345678901234567890

         0  DONE
```

serves as an end-of-files marker to the Reporter.

c. The Reporter requires one-character symbols representing the two report columns for insertion in the headings of subreports. Quite simply, if an H-series and a J-series armored division are being compared, "H J" will serve to put H's and J's in subreport headings.

d. To produce a subreport comparing three weapon types, say Blue types 2, 3, and 4, the operator must include the following record within the Reporter runstream:

```
               1         2         3         4         5         6
      123456789012345678901234567890123456789012345678901234567890

      3 1  2  1  3  1  4
```

The quotation marks are not to be included in the runstream. The first "3" specifies that the subreport is to include three weapon types. The " 1 2" specifies that the first weapon is to be from Side 1 and is to be of Type 2. The operator must refer to weapon ID and nomenclature lists to assure that proper weapons are being specified. The " 1 3" specifies that the second weapon also is to be from Side 1 and is to be of Type 3. Finally, the " 1 4" specifies that the third and final weapon to be included within the subreport is also to be from Side 1 and is to be of Type 4. To obtain a subreport containing only Blue Type 20 and Red Type 21, the operator must include a record within the runstream:

```
               1         2         3         4         5         6
      123456789012345678901234567890123456789012345678901234567890

      2 1 20  2 21
```

The sample runstream given in the next section directs production of six subreports. The same weapon type may be specified for inclusion in different subreports. An "@EOF" within the runstream indicates to the Reporter that no more subreports are required.

H-3. OUTPUT

a. The AFP Division Compare Reporter produces as many subreports as the operator specifies. An example of one such subreport is illustrated in Figure H-3.

b. As shown in Figure H-3, each weapon type reported leads to a separate section within a subreport. Each such section presents 12 lines of information. Three successive groups of four rows repeat a standard pattern of unmodulated score (U SCORE), unmodulated CIP (U CIP), modulated score (M SCORE), and modulated CIP (M CIP). The first group of four rows reports potentials from the files containing results for the first and fourth divisions. The second group of four rows reports potentials for the second and fifth divisions. The third group of four rows reports the results for the third and sixth divisions. If a weapon is not included within the inventory of a division or it achieves zero combat potential, the corresponding entries within the section of the subreport are zero-filled. Because a subreport directive within the Reporter runstream may specify only one weapon type, a subreport may contain only one weapon type section. Just such a subreport may be desirable. Although it does not provide any comparison between weapons, it obviously does provide comparison for a single weapon among different divisions or among different versions of the same division.

H-4. RUNSTREAM

a. Largely because so few complete sets of comparable division files have been produced to date, no generic runstream generator has been developed. But the nature of Division Compare Reporter runstreams is such that a generic runstream generator probably would save very little time and effort; it might even cost more.

b. Two examples of Reporter runstreams are provided in Figures H-4 and H-5.

(1) Figure H-4. Here the H- and J-series armored divisions are of interest.

(a) Lines 2-13 assign the six files containing the final combat potentials of interest.

(b) Line 26 directs execution of the Reporter absolute element, G6ROLLUP.COMPARE.

ITEM			A-SERIES PERS	A-SERIES LVEH	A-SERIES HVEH	A-SERIES ACFT	A-SERIES SCALAR	B-SERIES PERS	B-SERIES LVEH	B-SERIES HVEH	B-SERIES ACFT	B-SERIES SCALAR
15/1	1 & 4:	U SCORE	0.	0.	0.	0.	0.	2502.	143.	226.	4.	55.
		U CIP	.000	.000	.000	.000	.000	7.514	.431	.674	.012	.166
		M SCORE	0.	0.	0.	0.	0.	2504.	142.	227.	5.	56.
		M CIP	.000	.000	.000	.000	.000	7.515	.429	.676	.013	.167
	2 & 5:	U SCORE	0.	0.	0.	0.	0.	2837.	172.	135.	3.	47.
		U CIP	.000	.000	.000	.000	.000	8.741	.515	.403	.01	.139
		M SCORE	0.	0.	0.	0.	0.	2843.	172.	135	4.	47.
		M CIP	.000	.000	.000	.000	.000	8.485	.514	135.	.011	.14
	3 & 6:	U SCORE	0.	0.	0.	0.	0.	2670.	158.	181.	4.	51.
		U CIP	.000	.000	.000	.000	.000	7.992	.473	.538	.011	.153
		M SCORE	0.	0.	0.	0.	0.	2674.	157.	181	4.	51.
		M CIP	.000	.000	.000	.000	.000	8	.472	.539	.012	.154

ITEM			PERS	LVEH	HVEH	ACFT	SCALAR	PERS	LVEH	HVEH	ACFT	SCALAR
16/1	1 & 4:	U SCORE	1926.	99.	209.	3.	46.	0.	0.	0.	0.	0.
		U CIP	5.725	.298	.615	.008	.135	.000	.000	.000	.000	.000
		M SCORE	1601.	82.	174.	3.	38.	0.	0.	0.	0.	0.
		M CIP	4.757	.246	.511	.008	.113	.000	.000	.000	.000	.000
	2 & 5:	U SCORE	1863.	82.	185.	3.	41.	0.	0.	0.	0.	0.
		U CIP	5.522	.244	.543	.008	.12	.000	.000	.000	.000	.000
		M SCORE	1550.	68.	153.	3.	34.	0.	0.	0.	0.	0.
		M CIP	4.593	.202	.451	.008	.101	.000	.000	.000	.000	.000
	3 & 6:	U SCORE	1895.	68.	193.	3.	43.	0.	0.	0.	0.	0.
		U CIP	5.624	.271	.579	.008	.127	.000	.000	.000	.000	.000
		M SCORE	1575.	75.	164.	3.	36.	0.	0.	0.	0.	0.
		M CIP	4.675	.224	.481	.008	.107	.000	.000	.000	.000	.000

ITEM			PERS	LVEH	HVEH	ACFT	SCALAR	PERS	LVEH	HVEH	ACFT	SCALAR
17/1	1 & 4:	U SCORE	0.	0.	0.	0.	0.	392.	500.	107.	0.	1234.
		U CIP	.000	.000	.000	.000	.000	2.030	4.332	1.003	.000	9.154
		M SCORE	0.	0.	0.	0.	0.	401.	508.	120.	0.	1337.
		M CIP	.000	.000	.000	.000	.000	2.082	4.470	1.132	.000	10.087
	2 & 5:	U SCORE	0.	0.	0.	0.	0.	375.	543.	115.	0.	1396.
		U CIP	.000	.000	.000	.000	.000	1.945	4.446	1.076	.000	11.247
		M SCORE	0.	0.	0.	0.	0.	387.	555.	121.	0.	1433.
		M CIP	.000	.000	.000	.000	.000	1.998	4.521	1.111	.000	11.668
	3 & 6:	U SCORE	0.	0.	0.	0.	0.	402.	380.	97.	0.	897.
		U CIP	.000	.000	.000	.000	.000	2.111	3.937	.943	.000	7.960
		M SCORE	0.	0.	0.	0.	0.	399.	375.	92.	0.	879.
		M CIP	.000	.000	.000	.000	.000	2.102	3.890	.898	.000	7.739

Figure H-3. Example Subreport Produced by the AFP Division Compare Report--Providing a Comparison of Final Combat Potentials Between Divisions and Among Weapons and for two Different Combat Module Stating Seeds

(c) Lines 27-32 specify to the Reporter which file is to be treated as the Nth file.

(d) Line 33 serves as the end-of-files marker.

(e) Line 34 provides the one-character symbols for use in the headings of subreports.

(f) Lines 35-40 specify the subreports required in accord with the rules described in paragraph 4d above.

(g) Line 41 serves as the end-of-subreport specifications marker.

```
 1   @HDG UNCLASSIFIED 3M80 3M84 COMPARE
 2   @ASG,A H73M80R1601.
 3   @USE T1.,H73M80R1601.
 4   @ASG,A H73M80R1602.
 5   @USE T2.,H73M80R1602.
 6   @ASG,A H73M80R16X2.
 7   @USE T3.,H73M80R16X2.
 8   @ASG,A H73M84R1601.
 9   @USE T4.,H73M84R1601.
10   @ASG,A H73M84R1602.
11   @USE T5.,H73M84R1602.
12   @ASG,A H73M84R16X2.
13   @USE T6.,H73M84R16X2.
14   @DATA,L T1.
15   @END
16   @DATA,L T2.
17   @END
18   @DATA,L T3.
19   @END
20   @DATA,L T4.
21   @END
22   @DATA,L T5.
23   @END
24   @DATA,L T6.
25   @END
26   @XQT G6ROLLUP.COMPARE
27      1  T1.
28      2  T2.
29      3  T3.
30      4  T4.
31      5  T5.
32      6  T6.
33      0  DONE
34    A B
35   1 1  7
36   4 1 15   1 12   1 16   1 17
37   2 2 13   2 16
38   3 1 24   1 25   1 23
39   5 2 21   2 22   2 23   2 24   2 25
40   4 1 51   1 53   1 56   1 57
41   @EOF
```

Figure H-4. First Example of Runstream for Execution of the AFP Division Compare Reporter

(2) **Figure H-5.** Here six files, including some from the Rollup and Stats Module and some from the Interpolation Module, are designated for comparison. In this example, the USE statement is applied to show clearly the order of files to be reported. Prior to execution of the Reporter, the six files are listed for reference.

```
@HDG UNCLASSIFIED 3M80 3181-3183 3M84 COMPARE
@ELT,L G6GECTEST.G6COMPARE/INTERPS
@ASG,A H73M80R16X2.
@USE T1.,H73M80R16X2.
@ASG,A H7INT3M81.
@USE T2.,H7INT3M81.
@ASG,A H7INT3M82.
@USE T3.,H7INT3M82.
@ASG,A H73M82R16X2.
@USE T4.,H73M82R16X2.
@ASG,A H7INT3M83.
@USE T5.,H7INT3M83.
@ASG,A H73M84R16X2.
@USE T6.,H73M84R16X2.
@DATA,L T1.
@END
@DATA,L T2.
@END
@DATA,L T3.
@END
@DATA,L T4.
@END
@DATA,L T5.
@END
@DATA,L T6.
@END
@XQT G6ROLLUP.COMPARE
   1  T1.
   2  T2.
   3  T3.
   4  T4.
   5  T5.
   6  T6.
   0  DONE
  A B
1 1  7
4 1 15  1 12  1 16  1 17
3 1 24  1 25  1 23
4 1 51  1 53  1 56  1 57
@EOF
```

Figure H-5. Second Example of Runstream for Execution of the AFP Division Compare Reporter

H-5. PROGRAM

a. The AFP Division Compare Reporter does not perform any computation on AFP final combat potentials. The Reporter reads six complete sets of potentials and simply displays subsets of potentials in different combinations. Although the Reporter reads and stores both weapon and division potentials, the current version of the Reporter does nothing more with the division potentials (COPs).

b. CMPARE is the current version of the source text for the main program of the AFP Division Compare Reporter. Local to CMPARE are subroutines ZERO, GETREC, and GETFIL. The Reporter's other subroutines are BLDCIP, BLDSCR, BLDSC1, BLDSC2, DASH1, DASH2, DOREP, and HEAD. The source texts for these latter subroutines are maintained as separate elements within the AFPSYS library.

c. Figure H-6 portrays the basic logic of the AFP Division Compare Reporter.

d. The source texts for main program CMPARE and subroutines ZERO, GETREC, and GETFIL are listed in Figures H-7 through H-10.

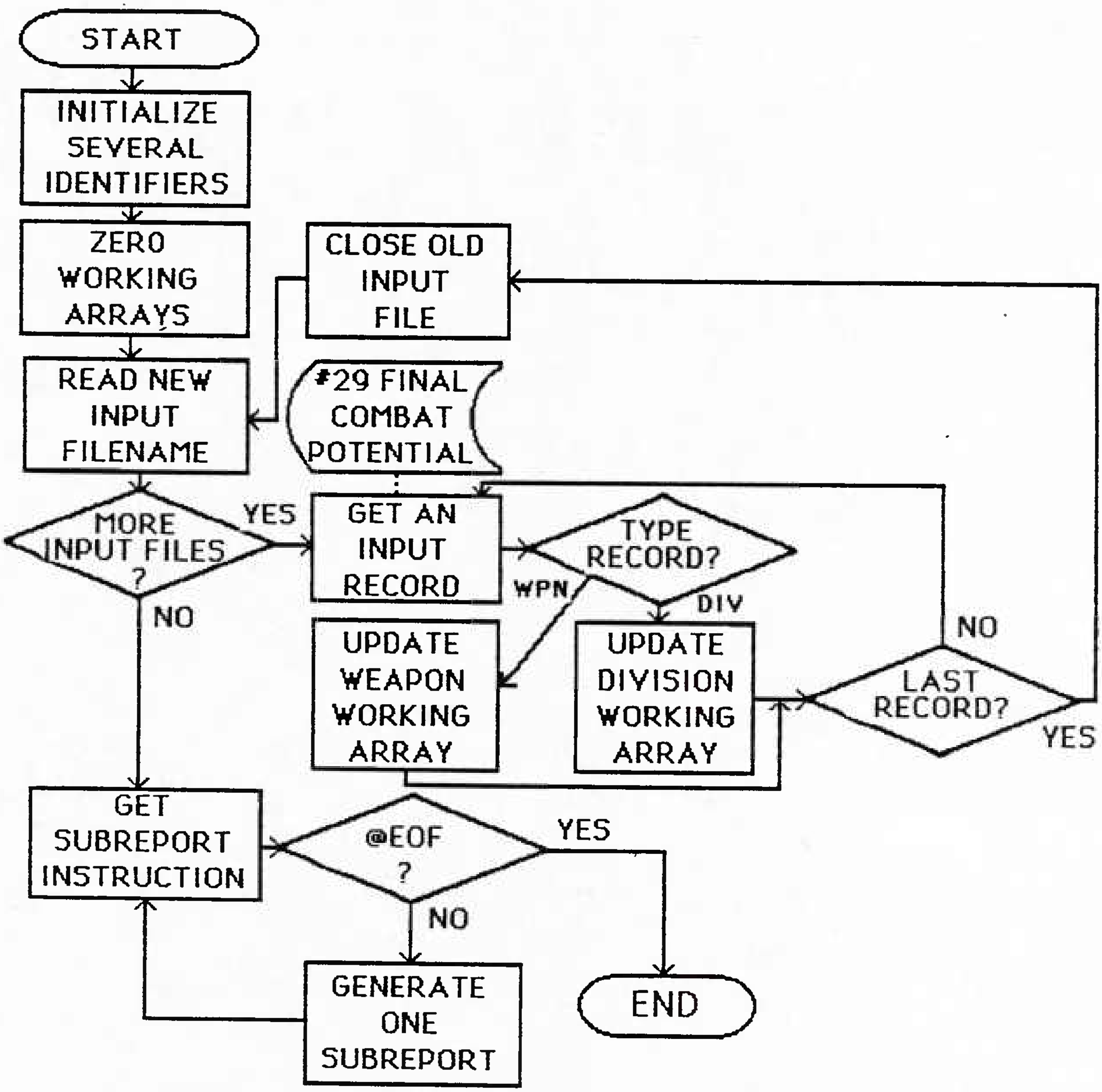


Figure H-6. Flow Diagram of the Basic Logic of the AFP Division Compare Reporter

```
      C        COMPARE CIPS & SCORES AMONG WEAPONS, SEEDS, & SERIES
      C        G.E.C. -- 13 OCT 83
              COMMON/WPNDAT/WVAL(5,4,60,2,6),COP(5,2,2,6)
               DIMENSION ISCNTW(6,2),
             1 ISCNTS(12),X(5),ISCNTR(6,2),ISCNTT(12),JRECS(12)
              EQUIVALENCE (ISCNTW,ISCNTS),(ISCNTR,ISCNTT)
              COMMON/AWRK/IWRK,IDUM,KTHTR,JTPD,JVIS,JPOS,JDAY
              COMMON/GLOBAL/IBFOR,IRFOR,JCASE
              COMMON/NAME/FNAME
              CHARACTER*50 INSTR
              CHARACTER*20 FNAME,ONAME,DONE
              DATA ISCNTW/
             1 110,130,150,170,111,151,
             2 120,140,160,180,121,161/
              DATA ISCNTR/
             1 110,130,150,170,111,151,
             2 120,140,160,180,121,161/
              DATA JRECS/
             1 1,2,3,4,1,2,
             2 1,2,3,4,1,2/
      C
              JVIS=0
              JPOS=0
              JDAY=0
      C       LOGICAL DESTINATION OF ROLLUP OUTPUT
              IWRK=30
              IPRNT=6
      C       LOGICAL SOURCE OF TARTY FILES
              ISOR=29
              DONE='DONE'
      C
              CALL ZERO(WVAL,14400)
              CALL ZERO(COP,120)
              GOTO 95
          90  CLOSE(ISOR)
          95  CALL GETFIL(FNAME,KFOR)
              IF(FNAME.EQ.DONE) GOTO 2000
              ONAME=FNAME
              WRITE(6,3) FNAME
         3    FORMAT(1X,A20)
              INSTR='@USE 29,'//ONAME//'.'
              CALL FACSF(INSTR)
      C
        100   CALL GETREC(ISCNT,IWPN,X,ISOR)
              DO 200 I=1,12
              IF(ISCNT.NE.ISCNTT(I)) GOTO 200
              IS=I
              GOTO 300
        200   CONTINUE
              GOTO 90
```

Figure H-7. Source Listing of the Main Program CMPARE of the AFP Division Compare Reporter (page 1 of 2 pages)

```
C     2 FORMAT(/' UNRECOGNIZABLE RECORD TYPE=',I5/)
C       IERR1=IERR1+1
C       IF(IERR1.LT.10) GOTO 100
C       STOP 'IERR1'
C
  300 IREC=JRECS(IS)
      IF(IS.LT.7) THEN
       ISIDE=1
      ELSE
       ISIDE=2
      ENDIF
C
      IF((IS.EQ.5).OR.(IS.EQ.6).OR.(IS.EQ.11).OR.(IS.EQ.12)) GOTO 500
C
C     HERE IF WEAPON RECORD READ
C
      DO 450 I=1,5
      WVAL(I,IREC,IWPN,ISIDE,KFOR)=X(I)
  450 CONTINUE
      GOTO 100
C
C     HERE IF COP RECORD READ
C
  500 DO 550 I=1,5
      COP(I,IREC,ISIDE,KFOR)=X(I)
  550 CONTINUE
C     ISCNT = 161 IS LAST RECORD OF AN ENVIRONMENT
      IF(ISCNT.EQ.161) GOTO 90
      GOTO 100
C
C      DO COMPARATIVE REPORTS
C
 2000 CALL DOREP
C
      STOP 'DONE'
C
```

Figure H-7. Source Listing of the Main Program CMPARE of the AFP Division Compare Reporter (page 2 of 2 pages)

(a) **WVAL(5,4,60,2,6).** Final combat potentials of weapons are read from the six input files into this array. Combat potentials are then extracted from this array, without modification, for insertion in subreports. An array element WVAL(I,J,K,L,M) is indexed--

1. I=1 to 5 for the components of five-valued form of combat potentials: personnel, light armored vehicles, heavy armored vehicles, aircraft, and the weighted rolled-up scalar.

2. J=1 to 4 for the four kinds of weapon potentials: unmodulated score, unmodulated CIP, modulated score, and modulated CIP.

3. K=1 to 60 for the 60 different weapon types.

4. L=1 to 2 for the two sides: Blue and Red.

5. M=1 to 6 for the six input files.

(b) **COP(5,2,2,6).** Final combat potentials of divisions (COPs) are read from the six input files into this array. In the current version of the Reporter, nothing more is done with the COPs. An array element COP(I,J,K,L) is indexed--

1. I=1 to 5 as in (1)(a) 1. above.

2. J=1 to 2 for the two kinds of COPs: unmodulated COP and modulated COP.

3. K=1 to 2 for the two sides: Blue and Red.

4. L=1 to 6 for the six input files.

(2) Three small arrays are used to store needed data.

(a) **ISCNTR(6,2) and ISCNTW(6,2).** The identifiers of record types within final potential input and output (the current version of the Reporter does not write to any file to be saved) files are stored in these arrays. Because the Reporter works only with final potentials, the same identifiers are stored in both arrays, admittedly redundantly. An array element ISCNTR(I,J) or ISCNTW(I,J) is indexed--

1. I=1 to 6 for the input (or in the future output) record type: unmodulated score, unmodulated CIP, modulated score, modulated CIP, unmodulated COP, and modulated COP.

2. J=1 to 2 for side: Blue and Red.

(b) **JRECS(12).** The array stores the identifiers linking the input record types to the indices of the kinds of potentials being processed. There are 12 input record types: 110, 130, 150, 170, 111, 151, 120, 140, 160 180, 121, 161. The first through fourth and seventh through tenth JRECS() values provide the indices to the unmodulated score, unmodulated CIP, modulated score, and modulated CIP positions within the weapon-related working array WVAL() described above. The fifth, sixth, eleventh, and twelfth JRECS() values provide the indices to the unmodulated COP and modulated COP positions within the division related working array COP() described above.

e. The CMPARE source listing in Figure H-7 includes some intralinear comments. The following paragraphs provide some additional commentary.

(1) Lines 3-11 in Figure H-7 provide the needed declarative statements, mostly for the arrays described above. Some scalar character variables are also declared.

(2) Lines 12-20 initialize reference arrays with identifiers and indices needed for correct processing of the final combat potentials from the six input files.

(3) Line 22 begins the executable statements. Lines 22-30 initialize several variables. JVIS, JPOS, and JDAY, by convention, are zero-valued identifiers in final combat potential records. If there were any Reporter output to a file for storage, it would be output to unit 30. Unit 6 is the system standard print file. Unit 29 is the source of the six successively read input files of final combat potentials. "DONE" is an end-of-input files marker.

(4) Lines 32 and 33 zero the working arrays.

(5) Line 25. The first time through there is no need to close any input file on unit 29.

(6) Line 36. Gets the name of an input file to be read and the index of the number of the file relative to the other input files. The name may be the end-of-files marker, "DONE".

(7) Line 37. Checks to determine whether all the input files have been read. If so, jump to begin generation of comparative subreports.

(8) Line 41. Concatenates an EXEC level instruction.

(9) Line 43. And calls FACSF to attach the physical file to unit 29.

(10) Line 44. Gets a record from the current input file.

(11) Lines 45-49. Determines what type of record was just read.

(12) Line 50. Here only if record was not identifiable. ISCNT was set to 999 by GETREC at the end-of-file. So GOTO 90 to close the current file.

(13) Lines 56-63. Line 56 sets the index of the kind of final combat potential supplied in the input record. Lines 57-61 set the side index for Blue (1) or Red (2). Line 63 checks whether the record applies to an entire division, and, if so, transfers control to statement 500 for division processing.

(14) Lines 65-70. This section is for the storing of the final combat potentials of a weapon type within the working array WVAL(). Lines 67-69 loop over the components of five-valued combat potential. Line 70 transfers control to retrieve another input record.

(15) Lines 74-79. This section is for the storing of the final combat potentials of a division within the working array COP(). Lines 74-76 loop over the components of five-valued combat potential. A record of type 161 is the last record within an input file. In line 78, the record type is checked, and, if its type is 161, control is transferred to get another input file name. If the record is not type 161, line 79 transfers control to get another final combat potential record from the current input file.

(16) Line 83 calls DOREP for the generation of comparative subreports.

(17) Line 85 provides normal termination of the AFP Division Compare Reporter.

f. Subroutine ZERO. Figure H-8 provides the source listing of subroutine ZERO. The subroutine simply fills a real array with zeros.

(1) Argument X is the address of the array.

(2) Argument N is the length of the array.

```
      SUBROUTINE ZERO(X,N)
      DIMENSION X(N)
      DO 100 I=1,N
      X(I)=0.0
  100 CONTINUE
      RETURN
      END
C
```

Figure H-8. Source Listing of Subprogram ZERO of the AFP Division Compare Reporter

g. Subroutine GETREC. Figure H-9 provides the source listing of subroutine GETRAC. Subroutine GETREC reads a record from a final combat potentials file. Several of the fields read from a record are not used in subsequent processing.

```
      SUBROUTINE GETREC(ISCNT,IWPN,X,ISOR)
      DIMENSION X(5)
      COMMON/NAME/FNAME
      CHARACTER*20 FNAME
      READ(ISOR,1,END=100) ISCNT,KTHTP,JTPD,JVIS,JPOS,JDAY,
     1 IWPN,X(1),X(2),X(3),X(4),X(5),IBFOR,IRFOR,JCASE
      IR=IR+1
      RETURN
    1 FORMAT(I5,A3,I4,3I3,I5,5F10.3,2I6,I5)
C
  100 WRITE(6,2) IR, FNAME
      ISCNT=999
      RETURN
    2 FORMAT (' LAST RECORD READ = ', I6,2X,A20)
      END
C
```

Figure H-9. Source Listing of Subprogram GETREC of the AFP Division Compare Reporter

(1) Argument ISCNT is the record type identifier and is returned by the subroutine. All types should be greater than 100. At an end-of-file, ISCNT is set to 999.

(2) Argument IWPN is the identifier of the weapon type and is returned by the subroutine.

(3) Argument X is a five-element real array which receives the five-valued final potential from a record and is returned by the subroutine.

(4) Argument ISOR is passed to the subroutine as the number of the unit from which input is to be read.

h. Subroutine GETFIL. Figure H-10 provides the source listing of subroutine GETFIL. Subroutine GETFIL reads and inputs file name and the order of the file in the sequence of six input files.

(1) Argument FNAME returns the input file name or the end-of-file names mark, "DONE."

(2) Argument IFOR is the number of the file in the sequence of six input files.

```
      SUBROUTINE GETFIL(FNAME,IFOR)
      CHARACTER*20 FNAME
      READ(5,1) IFOR,FNAME
      RETURN
    1 FORMAT(I3,2X,A20)
      END
```

Figure H-10. Source Listing of Subprogram GETFIL of the AFP Division Compare Reporter

i. Subroutine BLDCIP. Figure H-11 provides the source listing of subroutine BLDCIP. Subroutine BLDCIP outputs a single "CIP line" within a comparative subreport. BLDCIP, on a single call, outputs an identifier of the kind of CIP (unmodulated or modulated) and two five-valued sets of CIPs, one set for each of the two divisions being compared. BLDCIP is called six times for each weapon type within a single subreport. BLDCIP is called from subroutine DOREP.

```
      SUBROUTINE BLDCIP(LINE,X,Y)
      DIMENSION X(5),Y(5)
      CHARACTER*7 LINE
      WRITE(6,1) LINE,(X(I),I=1,5),(Y(I),I=1,5)
      RETURN
    1 FORMAT(18X,': ',A7,' :',5(F8.3,' :'),':',5(F8.3,' :'))
      END
```

Figure H-11. Source Listing of Subprogram BLDCIP of the AFP Division Compare Reporter

(1) Argument LINE is a 7-character identifier of the kind of CIP: "U CIP " or "M CIP " for unmodulated or modulated CIPs, respectively.

(2) Argument X is the address of the five-element real array containing the five-valued CIP corresponding to the current weapon type within the first division of the compared pair.

(3) Argument Y is the address of the five-element real array containing the five-valued CIP corresponding to the current weapon type within the second division of the compared pair.

j. Subroutine BLDSCR. Figure H-12 provides the source listing of subroutine BLDSCR. Subroutine BLDSCR outputs a single "modulated score line" within a comparative subreport. BLDSCR, on a single call, outputs an identifier of the kind of score (modulated) and two five-valued sets of scores, one set for each of the two divisions being compared. BLDSCR is called three times for each weapon type within a single subreport. BLDSCR is called from subroutine DOREP.

```
      SUBROUTINE BLDSCR(LINE,X,Y)
      DIMENSION X(5),Y(5)
      CHARACTER*7 LINE
      WRITE(6,1) LINE,(X(I),I=1,5),(Y(I),I=1,5)
      RETURN
    1 FORMAT(18X,': ',A7,' :',5(F8.0,' :'),':',5(F8.0,' :'))
      END
```

Figure H-12. Source Listing of Subprogram BLDSCR of the AFP Division Compare Report

(1) Argument LINE is a seven-character identifier of the kind of score: always "M SCORE" in the current version of the Reporter.

(2) Argument X is the address of the five-element real array containing the five-valued modulated score corresponding to the current weapon type within the first division of the compared pair.

(3) Argument Y is the address of the five-element real array containing the five-valued modulated score corresponding to the current weapon type within the second division of the compared pair.

k. Subroutine BLDSC1. Figure H-13 provides the source listing of subroutine BLDSC1. Subroutine BLDSC1 outputs a single "unmodulated score line" within a comparative subreport. BLDSC1, on a single call, outputs identifiers of the weapon type and side, a six-character weapon nomenclature, an identifier of the kind of score (unmodulated), and two five-valued sets of scores, one set for each of the two divisions being compared. Because the weapon and side identifiers and the weapon nomenclature are output for a weapon type only once per subreport, BLDSC1 is called only once for each weapon type within a single subreport. BLDSC1 is called from subroutine DOREP.

```
      SUBROUTINE BLDSC1(IW,NAME,LINE,X,Y,IS)
      DIMENSION X(5),Y(5)
      CHARACTER*6 NAME
      CHARACTER*7 LINE
C
      WRITE(6,1) IW,IS,NAME,LINE,(X(I),I=1,5),(Y(I),I=1,5)
      RETURN
C
    1 FORMAT(I3,'/',I1,1X,A6,' 1 & 4: ',A7,' :',
     1 5(F8.0,' :'),' :',5(F8.0,' :'))
      END
```

Figure H-13. Source Listing of Subprogram BLDSC1 of the AFP Division Compare Reporter

(1) Argument IW is the identifier of weapon type.

(2) Argument NAME is a six-character nomenclature for the weapon type.

(3) Argument LINE is a seven-character identifier of the kind of score: always "U SCORE" in the current version of the Reporter.

(4) Argument X is the address of the five-element real array containing the five-valued unmodulated score corresponding to the current weapon type within the first division of the compared pair.

(5) Argument Y is the address of the five-element real array containing the five-valued unmodulated score corresponding to the current weapon type within the second division of the compared pair.

(6) Argument IS is the index of the side: "1" for Blue, "2" for Red.

l. Subroutine BLDSC2. Figure H-14 provides the source listing of subroutine BLDSC2. Subroutine BLDSC2 outputs a single "unmodulated score line" within a comparative subreport. BLDSC2, on a single call, outputs identifiers of the divisions, an identifier of the kind of score (unmodulated), and two five-valued sets of scores, one set for each of the two divisions being compared. BLDSC2 is called twice for each weapon type within a single subreport. BLDSC2 is called from subroutine DOREP.

(1) Argument SUBTAB is a five-character of identifier of the section of the subreport: "2 & 5" or "3 & 6".

(2) Argument LINE is a seven-character identifier of the kind of score: always "U SCORE" in the current version of the Reporter.

(3) Argument X is the address of the five-element real array containing the five-valued unmodulated score corresponding to the current weapon type within the first division of the compared pair.

(4) Argument Y is the address of the five-element real array containing the five-valued unmodulate score corresponding to the current weapon type within the second division of the compared pair.

```
      SUBROUTINE BLDSC2(SUBTAB,LINE,X,Y)
      DIMENSION X(5),Y(5)
      CHARACTER*6 SUBTAB
      CHARACTER*7 LINE
      WRITE(6,1) SUBTAB,LINE,(X(I),I=1,5),(Y(I),I=1,5)
      RETURN
    1 FORMAT(13X,A5,': ',A7,' :',5(F8.0,' :'),':'5(F8.0,' :'))
      END
```

Figure H-14. Source Listing of Subprogram BLDSC2 of the AFP Division Compare Reporter

m. Subroutine DASH1. Figure H-15 provides the source listing of subroutine DASH1. Subroutine DASH1 outputs a line of 129 dashes as part of a subreport. DASH1 is called from subroutiine DOREP.

```
      SUBROUTINE DASH1
      WRITE(6,1)
      RETURN
    1 FORMAT(1X,129(1H-))
      END
```

Figure H-15. Source Listing of Subprogram DASH1 of the AFP Division Compare Reporter

n. Subroutine DASH2. Figure H-16 provides the source listing of subroutine DASH2. Subroutine DASH2 outputs a short line of 119 dashes as part of a subreport. DASH2 is called from subroutine DOREP.

```
      SUBROUTINE DASH2
      WRITE(6,1)
      RETURN
    1 FORMAT(11X,119(1H-))
      END
```

Figure H-16. Source Listing of Subprogram DASH2 of the AFP Division Reporter

o. Subroutine DOREP. Figure H-17 provides the source listing of subroutine DOREP. Subroutine DOREP is called by the main program of the Reporter to output as many subreports as directed by operator input within the Reporter runstream. The working array WVAL(), in common with the main program, contains the combat potentials that are extracted selectively for output within subreports.

(1) DOREP depends on two scratch arrays: ISS() and IWN(). A subreport may contain up to 10 weapon types.

(a) ISS(10). Array ISS() stores the side identifiers of the weapon types to be included within a directed subreport.

(b) IWN(10). Array IWN() stores the weapon identifiers of the weapon types to be included within a directed subreport.

(2) A reference array, NOMEN(60,2), is provided within DOREP for the storage and reference of six-character weapon nomenclatures. However, in the current version of DOREP, all nomenclatures have been left blank.

(3) The following comments apply to lines within Figure H-17.

(a) Lines 6-9 declare arrays and two character variables.

(b) Line 11 initializes the weapon nomenclatures to blank.

(c) Line 13 reads the two one-character symbols identifying the columns of subreports.

(d) Line 18 reads the directive for a subreport.

<u>1</u>. IWT specifies how many weapons are to be included within the subreport. IWT should not exceed 10; DOREP does not check for legal values of IWT! An "@EOF" image within the runstream indicates that no more subreports are required; control transfers to statement 1000 for return to the main program.

```
      SUBROUTINE DOREP
C        14 OCT 83 -- G.E.C.
C        READS REPORTER INSTRUCTIONS UNTIL @EOF
C        TO BUILD AND OUTPUT TYPE WEAPON REPORTS
C
      COMMON/WPNDAT/WVAL(5,4,60,2,6),COP(5,2,2,6)
      DIMENSION ISS(10),IWN(10)
      CHARACTER*1 SER1,SER2
      CHARACTER*6 NOMEN(60,2)
C
      DATA NOMEN/120*'      '/
C
      READ(5,1) SER1,SER2
    1 FORMAT(1X,A1,1X,A1)
C
C        GET PARAMETERS FOR A SUBREPORT
C
  100 READ(5,2,END=1000) IWT,(ISS(I),IWN(I),I=1,IWT)
    2 FORMAT()
C
C        SKIP TO TOP OF NEW PAGE
C
      WRITE(6,4)
    4 FORMAT(1H1)
C
      WRITE(6,3) SER1,SER2
    3 FORMAT(50X,A1,'-SERIES',20X,'::',20X,A1,'-SERIES'/
     1 50X,8H--------,20X,'::',20X,8H--------)
C
      DO 200 I=1,IWT
      IW=IWN(I)
      IS=ISS(I)
      CALL HEAD
C
      CALL DASH1
      CALL BLDSC1(IW,NOMEN(IW,IS),'U SCORE',
     1 WVAL(1,1,IW,IS,1),WVAL(1,1,IW,IS,4),IS)
      CALL BLDCIP('U CIP   ',WVAL(1,2,IW,IS,1),WVAL(1,2,IW,IS,4))
      CALL BLDSCR('M SCORE',WVAL(1,3,IW,IS,1),WVAL(1,3,IW,IS,4))
      CALL BLDCIP('M CIP   ',WVAL(1,4,IW,IS,1),WVAL(1,4,IW,IS,4))
      CALL DASH2
      CALL BLDSC2('2 & 5','U SCORE',WVAL(1,1,IW,IS,2),
     1 WVAL(1,1,IW,IS,5))
      CALL BLDCIP('U CIP   ',WVAL(1,2,IW,IS,2),WVAL(1,2,IW,IS,5))
      CALL BLDSCR('M SCORE',WVAL(1,3,IW,IS,2),WVAL(1,3,IW,IS,5))
      CALL BLDCIP('M CIP   ',WVAL(1,4,IW,IS,2),WVAL(1,4,IW,IS,5))
      CALL DASH2
      CALL BLDSC2('3 & 6 ','U SCORE',WVAL(1,1,IW,IS,3),
     1 WVAL(1,1,IW,IS,6))
      CALL BLDCIP('U CIP   ',WVAL(1,2,IW,IS,3),WVAL(1,2,IW,IS,6))
      CALL BLDSCR('M SCORE',WVAL(1,3,IW,IS,3),WVAL(1,3,IW,IS,6))
      CALL BLDCIP('M CIP   ',WVAL(1,4,IW,IS,3),WVAL(1,4,IW,IS,6))
      CALL DASH1
      WRITE(6,2)
  200 CONTINUE
C
      GOTO 100
C
 1000 RETURN
C
      END
```

Figure H-17. Source Listing of Subprogram DOREP of the AFP Division Compare Reporter

2. ISS(I) specifies the side of the Ith weapon.

3. IWN(I) specifies the type of the Ith weapon.

(e) Line 23 ejects to new page.

(f) Line 26 outputs a subreport heading including one-symbol identifiers of the two divisions being compared.

(g) Lines 30 and 55 are the outer limits of the loop over the weapon types to be included within the current subreport.

(h) Lines 31 and 32 set weapon and side scalars for repeated reference within the loop.

(i) Line 33 calls for the output of standard column headings.

(j) Lines 35-53 build a one-weapon section of a subreport line-by-line by means of calls to special one-liner subroutines. Some of the subroutines are called more than once, some with different arguments each time. The most important calls are those that send addresses of five-element subarrays within WVAL() for output as five-valued combat potentials. The indexing of WVAL() as arguments within the call statements "picks" the corrent kind of potential, weapon, side, and original input file.

(k) Line 57 transfers control to statement 100 for another subreport directive, if any.

p. Subroutine HEAD. Subroutine HEAD is called from subroutine DOREP to output the standard column headings within a subreport. Figure H-18 provides the source listing of subroutine Head.

```
      SUBROUTINE HEAD
      WRITE(6,1)
      RETURN
    1 FORMAT(' ITEM',23X,
     1 2(':   PERS    :   LVEH    :   HVEH    :   ACFT    :   SCALAR :'))
      END
```

Figure H-18. Source Listing of Subprogram HEAD of the AFP Division Compare Reporter

q. MAP Element CMPMAP. Figure H-19 provides a listing of the MAP element for collection of the program elements of the Division Compare Reporter.

```
@MAP    ,G6ROLLUP.COMPARE
   NOT  TPF$.
   IN   G6ROLLUP.CMPARE
   IN   G6ROLLUP.DOREP
   IN   G6ROLLUP.DASH1
   IN   G6ROLLUP.DASH2
   IN   G6ROLLUP.HEAD
   IN   G6ROLLUP.BLDSC1
   IN   G6ROLLUP.BLDCIP
   IN   G6ROLLUP.BLDSCR
   IN   G6ROLLUP.BLDSC2
@EOF
```

Figure H-19. Listing of the MAP Element for Collection of the Program Elements of the AFP Division Compare Reporter

APPENDIX I

THE AFP INTERPOLATION MODULE

Section I. OVERVIEW

I-1. PURPOSE. The AFP Interpolation Module is designed to save time in the determination of final combat potentials for an intermediate Blue equipment inventory "between" two other inventories whose combat potentials are already known. The principal notion of "between" arises for a division undergoing modernization. Such a division begins with a baseline or old inventory, progresses through intermediate inventories, and thus evolves toward a target or final inventory. In the real world, the target inventory may itself be changed long before it is attained. In the AFP world, it is assumed that the target inventory is a final goal. Intermediate inventories must not contain equipment types not found in either the baseline or target inventories.

I-2. FEATURES. Toward accomplishment of the above primary design objective, the AFP Interpolation Module is designed to:

a. Accept files or elements of final combat potentials for baseline and target inventories.

b. Accept combat weapon inventories for baseline, target, and intermediate divisions. (The module processes only one intermediate inventory in a single execution.)

c. Accept the normal input to the AFP CS/CSS Module.

d. Generate estimates of net CS/CSS moduli averaged over all target categories and combat postures.

e. Interpolate both unmodulated and modulated CIPs for all Blue weapon types included in the intermediate inventory. The user may select one of two methods described below.

f. Combine the interpolated CIPs with intermediate weapon quantities to generate interpolated unmodulated and modulated Blue weapon scores.

g. Sum the interpolated Blue weapon scores to produce interpolated unmodulated and modulated Blue COPs.

h. Generate a standard formatted final combat potentials file for the intermediate Blue inventory.

I-3. RELATION TO COMPLETE SYSTEM. The relation of the AFP Interpolation Module to the AFP System in general is portrayed in Figure I-1. There the module is highlighted by being enclosed in an oval. What is not obvious in Figure I-1 is that, given final combat potentials for the related DIV A (baseline) and DIV B (target) divisions, as shown in block 25 of the figure, only the Interpolation Module need be executed in order to estimate the final combat potentials of DIV X (the intermediate division). The other modules need not be executed. Hence, interpolation eliminates at least 16 executions of the Combat and CBT/CS/CSS Merge Modules. Of course, the full system can be executed as usual to estimate the final combat potentials of an intermediate division. Occasional exercise of the full system is recommended as a check on the performance of the Interpolation Module. After all, the Interpolation Module does not perform all the functions of the full AFP System.

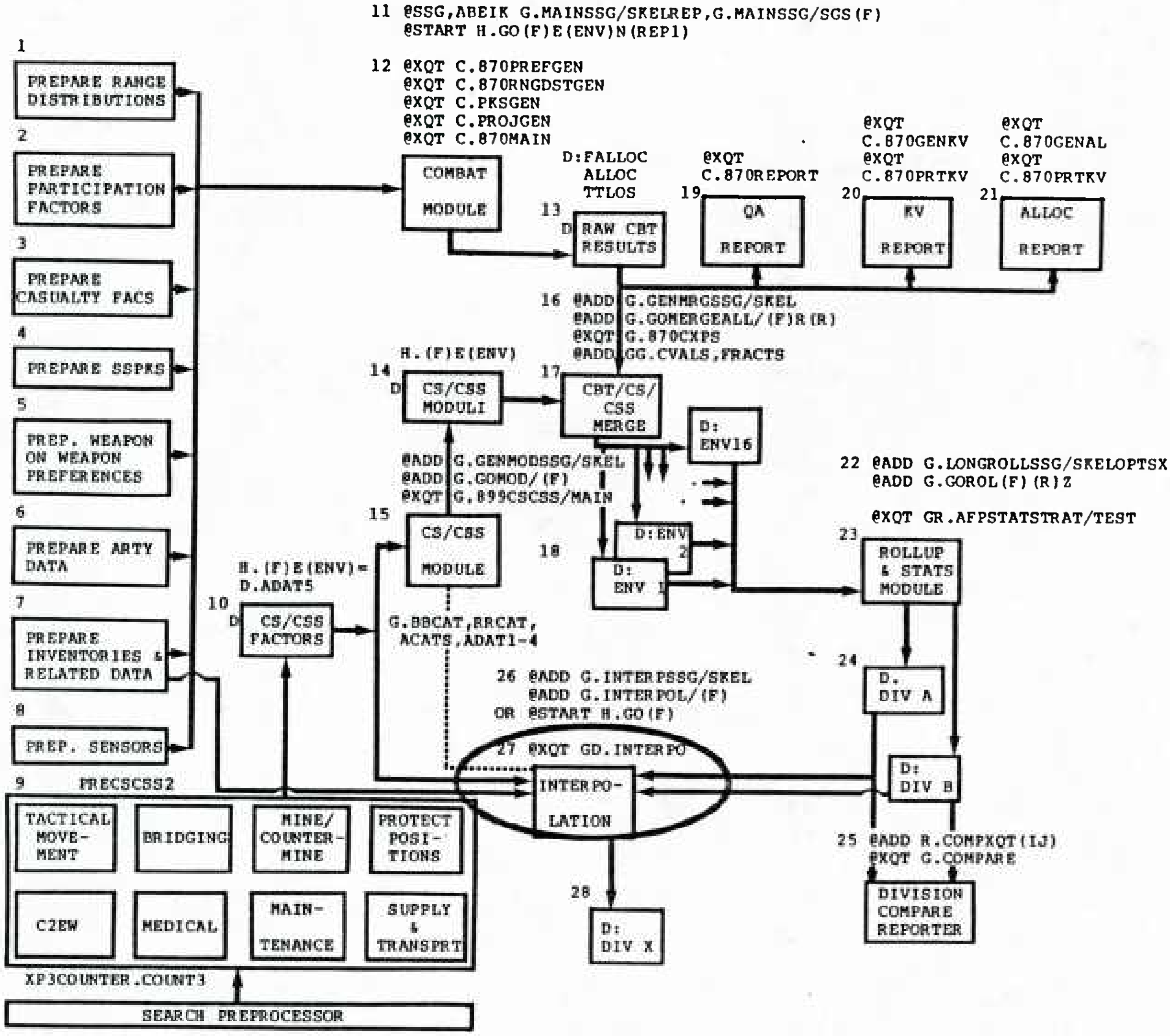


Figure I-1. Relation of the AFP Interpolation Module to the AFP System in General

Section II. INPUT

I-4. PRINCIPLE. Almost all input to the AFP Interpolation Module is identical in format and content to input required by other AFP modules. Data pertaining to the baseline and target divisions are identical in both format and content to the input supplied to other AFP modules in the generation of those divisions' final combat potentials. Data pertaining to an intermediate division are in the standard format but do include new values. The inventory and CS/CSS factors for the intermediate division are both new in content. Interpolation does depend on a special scalar value, the inventory phase parameter. The inventory phase parameter is intended to express the fraction of "time" elapsed from baseline to intermediate inventory dates in relation to the total time between baseline and target inventory dates.

I-5. INPUT TYPES. The AFP Interpolation Module "expects" input data in the following order. Many of the files may be created at run time as copies of source data elements.

a. The inventory phase parameter (runstream). The number and indices of Blue nondivisional weapons.

b. Contents of the file (#17) specifying the weapon categories to which Blue weapon types belong. The file is described as BBCAT in Appendix E on the APF CS/CSS Module. The file is read in the Interpolation Module by subprogram GETCAT.

c. Contents of the file (#18) specifying the weapon categories to which Red weapon types belong. The file is described as RRCAT in Appendix E on the AFP CS/CSS Module. The file is read in the Interpolation Module by subprogram GETCAT.

d. Contents of the file (#11) specifying whether Blue and Red weapon types are affected by CS/CSS factors in general (Y/N for each type). The file is described as ADAT1 in Appendix E on the AFP CS/CSS Module. The file is read in the Interpolation Module by subprogram GETDAT.

e. Contents of the file (#12) specifying for each Blue and Red weapon type and each CS/CSS function whether the corresponding measures and countermeasures apply (Y/N for each measure/countermeasure for each function for each weapon type by Blue and Red). The file is described as ADAT2 in Appendix E on the AFP CS/CSS Module. The file is read in the Interpolation Module by subprogram GETDAT.

f. Contents of the file (#13) specifying for each CS/CSS function and each combat environment whether the Blue and Red countermeasures apply (Y/N for each measure/countermeasure for each function for each combat environment by Blue and Red). The file is described as ADAT3 in Appendix E on the AFP CS/CSS Module. The file is read in the Interpolation Module by subprogram GETDAT.

g. Contents of the file (#14) specifying the relative weight ascribed to each CS/CSS function in each combat environment (numerical weight: 0.00 - 1.25, for example). The file is described as ADAT4 in Appendix E on the AFP CS/CSS Module. The file is read in the Interpolation Module by subprogram GETDAT.

h. Name of the file containing the raw CS/CSS factors for the baseline division. The name is read in the Interpolation Module by subprogram USENAM.

i. Pointers (16) specifying for each combat environment the set of CS/CSS factors to be applied. Contents of the file (#29) named in h above and specifying the CS/CSS factors by CS/CSS function (possibly for different combat environments) for Blue and Red measures and countermeasures for the baseline division. The file is described as ADAT5 in Appendix E on the AFP CS/CSS Module. The file is read in the Interpolation Module by subprogram GETFAC.

j. Name of the file containing the raw CS/CSS factors for the target (final) division. The name is read in the Interpolation Module by subprogram USENAM.

k. Pointers (16) specifying for each combat environment the set of CS/CSS factors to be applied. Contents of the file (#29) named in j above and specifying the CS/CSS factors by CS/CSS function (possibly for different combat environments) for Blue and Red measures and countermeasures for the target division. The file is described as ADAT5 in Appendix E on the AFP CS/CSS Module. The file is read in the Interpolation Module by subprogram GETFAC.

l. Name of the file containing the raw CS/CSS factors for the intermediate division. The name is read in the Interpolation Module by subprogram USENAM.

m. Pointers (16) specifying for each combat environment the set of CS/CSS factors to be applied. Contents of the file (#29) named in l above and specifying the CS/CSS factors by CS/CSS function (possibly for different combat environments) for Blue and Red measures and countermeasures for the intermediate division. The file is described as ADAT5 in Appendix E on the AFP CS/CSS Module. The file is read in the Interpolation Module by subprogram GETFAC.

n. Contents of the file (#16) specifying for each CS/CSS function and each combination of Blue weapon categories (12) and Red categories (12) whether the Blue and Red measures and countermeasures apply (Y/N for each combination of function, Blue category, Red category, and Blue and Red measures and countermeasures). The file is described as ACATS in Appendix E on the APF CS/CSS Module. The file is read in the Interpolation Module by subprogram GTCATV.

o. The numerical weights (summing to 1.0) ascribed to the 16 combat environments for weighted summation of partial combat potentials to yield final combat potentials. The weights are read from the runstream by subprogram ENVWTS of the Interpolation Module.

p. Name of the file containing the inventory for the baseline division. The name is read in the Interpolation Module by subprogram USENAM.

q. Contents of the file (#29) named in p above and specifying the quantities of Blue weapons by type in the baseline division inventory. The file is read as G6INVB in the Interpolation Module by subprogram GETINV.

r. Name of the file containing the inventory for the target (final) division. The name is read in the Interpolation Module by subprogram USENAM.

s. Contents of the file (#29) named in r above and specifying the quantities of Blue weapons by type in the target (final) division inventory. The file is read as G6INVT in the Interpolation Module by subprogram GETINV.

t. Name of the file containing the inventory for the intermediate division. The name is read in the Interpolation Module by subprogram USENAM.

u. Contents of the file (#29) named in t above and specifying the quantities of Blue weapons by type in the intermediate division inventory. The file is read as G6INVI in the Interpolation Module by subprogram GETINV.

v. Name of the file containing the final combat potentials of the baseline division. The name is read from the runstream by subprogram GETFIL called by subprogram GETROL of the Interpolation Module.

w. Partial contents of the file (#29) named in v above and specifying the final combat potentials (unmodulated and modulated scores, CIPs, and COPs) of the baseline and opposing divisions. Only the Blue scores are needed from this file. The file is described (paragraph B-2i) and illustrated (Figure B-5) in the AFP product section of Appendix B. The file is read in the Interpolation Module by subprogram GETREC called by subprogram GETROL. Example extract records are shown in Figure I-2. Because combat potentials may not or may have been generated, including target values within the "personnel," light vehicle, heavy vehicle, and aircraft elements, consistency with paragraph y should be maintained.

x. Name of the file containing the final combat potentials of the target (final) division. The name is read from the runstream by subprogram GETFIL called by subprogram GETROL of the Interpolation Module.

y. Partial contents of the file (#29) named in x above and specifying the final combat potentials (unmodulated and modulated scores, CIPs, and COPs) of the target (final) and opposing divisions. Only the Blue scores are needed from this file. The file is described (paragraph B-2i) and illustrated (Figure B-5) in the AFP product section of Appendix B. The file is read in the Interpolation Module by subprogram GETREC called by subprogram GETROL. Example extract records are shown in Figure I-2. Because combat potentials may not or may have been generated, including target values within the "personnel," light vehicle, heavy vehicle, and aircraft elements, consistency with paragraph x should be maintained.

	FIELD								
	1	2	3	4	5	6	7	8	9
25.	110	E 1 0 0 0	16	991.096	67.506	90.345	.000	21.586	1 1 1
26.	130	E 1 0 0 0	16	4.737	.327	.436	.000	.104	1 1 1
27.	150	E 1 0 0 0	16	990.053	67.287	90.312	.000	21.555	1 1 1
28.	170	E 1 0 0 0	16	4.779	.325	.435	.000	.104	1 1 1
29.	110	E 1 0 0 0	17	524.247	35.921	45.623	.000	11.156	1 1 1
30.	130	E 1 0 0 0	17	4.628	.318	.400	.000	.098	1 1 1
31.	150	E 1 0 0 0	17	523.781	35.855	45.722	.000	11.161	1 1 1
32.	170	E 1 0 0 0	17	4.621	.317	.401	.000	.098	1 1 1
33.	110	E 1 0 0 0	20	2669.735	157.594	180.532	3.791	51.092	1 1 1
34.	130	E 1 0 0 0	20	7.992	.473	.538	.011	.153	1 1 1
35.	150	E 1 0 0 0	20	2673.525	157.388	180.920	4.131	51.470	1 1 1
36.	170	E 1 0 0 0	20	8.000	.472	.539	.012	.154	1 1 1
37.	110	E 1 0 0 0	26	105.556	4.346	.000	8.111	8.841	1 1 1
38.	130	E 1 0 0 0	26	3.175	.131	.000	.244	.265	1 1 1
39.	150	E 1 0 0 0	26	107.007	4.320	.000	8.807	9.538	1 1 1
40.	170	E 1 0 0 0	26	3.218	.130	.000	.264	.286	1 1 1
					*				
					*				
					*				
169.	120	E 1 0 0 0	51	3.160	.074	.000	.000	.016	1 1 1
170.	140	E 1 0 0 0	51	.129	.003	.000	.000	.001	1 1 1
171.	160	E 1 0 0 0	51	3.329	.079	.000	.000	.017	1 1 1
172.	180	E 1 0 0 0	51	.136	.003	.000	.000	.001	1 1 1
173.	120	E 1 0 0 0	52	45.216	7.383	.013	.000	.902	1 1 1
174.	140	E 1 0 0 0	52	.741	.121	.000	.000	.015	1 1 1
175.	160	E 1 0 0 0	52	47.456	7.748	.013	.000	.947	1 1 1
176.	180	E 1 0 0 0	52	.778	.127	.000	.000	.015	1 1 1
177.	120	E 1 0 0 0	56	19.989	2.611	.047	.000	.335	1 1 1
178.	140	E 1 0 0 0	56	.169	.022	.000	.000	.003	1 1 1
179.	160	E 1 0 0 0	56	20.952	2.749	.049	.000	.352	1 1 1
180.	180	E 1 0 0 0	56	.178	.023	.000	.000	.003	1 1 1
					*				
					*				
					*				
189.	111	E 1 0 0 0	0	11916.452	897.693	489.232	103.601	293.738	1 1 1
190.	151	E 1 0 0 0	0	11919.645	894.350	488.403	112.592	302.274	1 1 1
191.	121	E 1 0 0 0	0	898.871	184.892	12.760	89.643	112.274	1 1 1
192.	161	E 1 0 0 0	0	975.283	193.200	13.275	106.670	131.446	1 1 1

Figure I-2. Example Extract Records from File of Final Combat Potentials Output from the AFP Rollup and Stats Module as Input to the AFP Interpolation Module

Section III. OUTPUT

I-6. OUTPUT EXAMPLE. The product of the Interpolation Module is a file and listing of final Blue combat potentials for the intermediate division. Typical records of such a file are illustrated in Figure I-3. The file is in standard AFP System combat potential output format as described in paragraph B-2i and also illustrated in Figure B-5 of Appendix B. However, the files producible by the Interpolation Module do not contain Red combat potentials. In Figure I-3, the first four elements of combat potentials have not been weighted by target values.

	FIELD								
	1	2	3	4	5	6	7	8	9
17.	10	E 1 1 1 1	16	1191.458	84.925	113.229	.000	26.959	1 1 1
18.	30	E 1 1 1 1	16	5.784	.412	.550	.000	.131	1 1 1
19.	50	E 1 1 1 1	16	1197.489	85.355	113.802	.000	27.096	1 1 1
20.	70	E 1 1 1 1	16	5.813	.414	.552	.000	.132	1 1 1
21.	10	E 1 1 1 1	17	644.042	47.542	48.500	.000	13.075	1 1 1
22.	30	E 1 1 1 1	17	5.776	.426	.435	.000	.117	1 1 1
23.	50	E 1 1 1 1	17	647.302	47.782	48.746	.000	13.141	1 1 1
24.	70	E 1 1 1 1	17	5.805	.429	.437	.000	.118	1 1 1
25.	10	E 1 1 1 1	20	3252.010	234.271	345.906	5.000	83.696	1 1 1
26.	30	E 1 1 1 1	20	9.855	.710	1.048	.015	.254	1 1 1
27.	50	E 1 1 1 1	20	3272.702	235.457	347.657	5.448	84.554	1 1 1
28.	70	E 1 1 1 1	20	9.917	.714	1.054	.017	.256	1 1 1
29.	10	E 1 1 1 1	26	118.500	6.625	.000	8.000	9.006	1 1 1
30.	30	E 1 1 1 1	26	3.703	.207	.000	.250	.281	1 1 1
31.	50	E 1 1 1 1	26	120.453	6.659	.000	8.717	9.732	1 1 1
32.	70	E 1 1 1 1	26	3.764	.208	.000	.272	.304	1 1 1
					*				
					*				
					*				
157.	20	E 1 1 1 1	51	1.706	.000	.000	.000	.004	1 1 1
158.	40	E 1 1 1 1	51	.059	.000	.000	.000	.000	1 1 1
159.	60	E 1 1 1 1	51	1.780	.000	.000	.000	.005	1 1 1
160.	80	E 1 1 1 1	51	.061	.000	.000	.000	.000	1 1 1
161.	20	E 1 1 1 1	52	51.161	6.395	.000	.000	.810	1 1 1
162.	40	E 1 1 1 1	52	.839	.105	.000	.000	.013	1 1 1
163.	60	E 1 1 1 1	52	53.377	6.672	.000	.000	.845	1 1 1
164.	80	E 1 1 1 1	52	.875	.109	.000	.000	.014	1 1 1
165.	20	E 1 1 1 1	56	21.396	1.297	.000	.000	.192	1 1 1
166.	40	E 1 1 1 1	56	.181	.011	.000	.000	.002	1 1 1
167.	60	E 1 1 1 1	56	22.322	1.353	.000	.000	.201	1 1 1
168.	80	E 1 1 1 1	56	.189	.011	.000	.000	.002	1 1 1
					*				
					*				
					*				
177.	11	E 1 1 1 1	0	16413.689	1193.812	704.285	112.000	373.358	1 1 1
178.	51	E 1 1 1 1	0	16494.709	1197.939	705.834	122.042	384.262	1 1 1
179.	21	E 1 1 1 1	0	1052.358	180.632	13.746	125.153	148.850	1 1 1
180.	61	E 1 1 1 1	0	1144.605	187.630	14.106	150.793	175.517	1 1 1

Figure I-3. Example Extract Records from File of Interpolated Final Combat Potentials as Output by the AFP Interpolation Module for an Intermediate Division (Blue only)

Section IV. RUNSTREAM

I-7. RUNSTREAM INTRODUCTION. This section describes an example SSG program for generating AFP Interpolation Module runstreams and provides some examples of generated runstreams. Familiarity with the UNIVAC Symstream language and SSG processor is assumed.

I-8. INTENT. Runstream generation is intended to simplify several possible problems in correctly applying the Interpolation Module. As should be evident from the preceding section on INPUT, the Interpolation Module requires a variety of data about three different divisions: the baseline, target, and intermediate divisions. In a typical application, the intermediate division may be of interest over several different years with variation in inventory and CS/CSS factors from year to year. The Interpolation Module must be executed separately for each intermediate year. However, the runstream generator, in a single execution, may generate the runstreams for all the intermediate years. Some data are common to the baseline, target, and all intermediate divisions. Some data are needed only for the baseline and target divisions. The module requires that some files be preassigned to logical units throughout a single module run. On the other hand, the module successively assigns some different files to the same logical unit during module execution. The runstream generator is intended to simplify the task of keeping the file assignments "straight."

I-9. RUNSTREAM AND INPUT LINK. Figure I-4 serves as a bridge among the Interpolation Module's input requirements, runstream generation, and later program descriptions. To a large extent, Figure I-4 is a tabular recapitulation of paragraph I-5 in the paragraph on input.

a. Field 1 identifies a program element requiring data identified somewhere in the module's runstream.

b. Field 2 provides a generalized identifier of the data set (e.g., part of a file name) or an example value.

c. Field 3 identifies the "expected" FORTRAN unit.

d. Field 4 includes FORTRAN record format specifier, as appropriate.

e. Field 5 identifies receiving arrays or variables.

FIELD

1 Program or Sub-program	2 Element, File, or Example	3 Unit	4 Record Format	5 Receiving Array or Variable
DOFAX	0.2	5	()	T
GETCAT	BBCAT	17	(3X,10I3)	LBCAT(60)
GETCAT	RRCAR	18	(3X,10I3)	LRCAT(60)
GETDAT	ADAT1	11	(5X,10A1)	BW(60),RW(60)
	ADAT2	12	(5X,9(4A1,1X))	UFUN(9,60)...TFUN(9,60)
	ADAT3	13	(5X,9(4A1,1X))	EUFUN(9,16)...ETFUN(9,16)
	ADAT4	14	(5X,9F6.4)	A(9,16)
USENAM	'BADAT5'	5	(A20)	FNAME
GETFAC	' 1 1...'	5	()	INDX(I,1) I=1,16
	BADAT5	29	(3X,I2,4F8.4)	J,U(I,J,1)...T(I,J,1); I=1,9
USENAM	'TADAT5'	5	(A20)	FNAME
GETFAC	' 1 1...'	5	()	INDX(I,2) I=1,16
	TADAT5	29	(3X,I2,4F8.4)	J,U(I,J,2)...T(I,J,2); I=1,9
USENAME	'IADAT5'	5	(A20)	FNAME
GETFAC	' 1 1...'	5	()	INDX(I,3) I=1,16
	IADAT5	29	(3X,I2,4F8.4)	J,U(I,J,3)...T(I,J,3); I=1,9

(NOTE: Sets of U(I,J,K)...T(I,J,K) need not be input for all values of J; indeed, only one J-set may be sufficient--then all corresponding pointers in INDX(J,K) for given K should point to the same J-set.)

GTCATV	ACATS	16	(4X,12(1X,4A1))	UCAT(9,12,12)...TCAT(9,12,12)
ENVWTS	0.091	5	()	EW(16)
USENAM	'INVB'	5	(A20)	FNAME
GETINV	INVB	29	()	QTY(60,J) J=1
USENAM	'INVT'	5	(A20)	FNAME
GETINV	INVT	29	()	QTY(60,J) J=2
USENAM	'INVI'	5	(A20)	FNAME
GETINV	INVI	29	()	QTY(60,J) J=3
GETROL				
GETFIL	'ROLb'	5	(A20)	FNAME
GETREC	ROLb	29	(...5F10.3...)	WVAL(5,60,K,4) K=1
GETFIL	'ROLt'	5	(A20)	FNAME
GETREC	ROLt	29	(...5F10.3...)	WVAL(5,60,K,4) K=2
GETFIL	'DONE'	5	(A20)	FNAME

Figure I-4. Summary of the Input Files and Records Required During Execution of the Interpolation Module

I-10. EXAMPLE OF SSG RUNSTREAM GENERATOR. Figure I-5 displays an example SSG program for generating a set of Interpolation Module runstreams. The example is set up for a specific application: division HM in 80 as the baseline, division JM in 84 as the target, and 13 divisions as the intermediates in each of years 80 through 84. Execution of the example program generates 13 runstreams, one for each intermediate division. Production versions of the generator and runstreams must be classified if classified read/write keys are included. No keys are included in any examples shown.

a. SGS Section. The order of SGSs is not important; their contents are critical.

(1) The SGS "PHI" specifies the phase parameter values for the five "intermediate" years. There should be as many values of PHI as there are intermediate years. PHI values always affect computation of modulated combat potentials. Depending on the setting of SGS VARY (see (29) below) PHI value may also affect computation of unmodulated combat potentials. A PHI value is intended to represent the corresponding fraction of time between base and target years.

(2) The SGS "ALLDIV" specifies the symbols identifying the divisions to be "interpolated." The example contains 13 division symbols for generation of 13 runstreams. More or less symbols are permitted.

(3) The SGS "IFORCE" specifies the symbols identifying the years for which interpolations are to be performed for each division identified in ALLDIV above.

(4) The SGS "KTHTR" specifies the symbol (here bbE) to be inserted in output records as the theater identifier. The same theater identifier applies to all intermediate divisions.

(5) The SGS "JTPD" specifies the symbols (here bb80 through bb84) to be inserted in output records as the time period (year) identifier.

(6) The SGS "DIVNUM" specifies the symbols to be inserted in output records as the identifiers (possibly TPSN) of the Blue divisions. The identifiers are assumed to remain constant over all intermediate time periods.

(7) The SGS "IRFOR" specifies the symbol (here bb1000) to be inserted in output records as the threat division identifier.

(8) The SGS "JCASE" provides case identifiers (here b8032 through b8432) to be inserted in output records.

(9) The SGS "BFORCE" specifies the symbol (here HM80) identifying the baseline division in many of the input files/elements. A separate identifier is provided for CS/CSS factors.

```
@SSG,BK
SGS
PHI ''0.05''  ''0.25''  ''0.50''  ''0.75''  ''0.95''
ALLDIV 3M 1C 1A 2A 3A 49A 50A 1M 4M 5M 8M 24M 40M
IFORCE 80 81 82 83 84
KTHTR ''  E''
JTPD ''  80''  ''  81''  ''  82''  ''  83''  ''  84''
DIVNUM  ''  2101''  ''  2201''  ''  2401''  ''  2301''  ''  3101'' ;
  ''  3201''  ''  3301''  ''  3401''  ''  3501''  ''  3601'' ;
  ''  3701''  ''  3801''  ''  3901''
IRFOR ''  1000''
JCASE ''  8032''  ''  8132''  ''  8232''  ''  8332''  ''  8432''
BFORCE HM80
TFORCE JM84
BFORCEC HMC0
TFORCEC JMC0
BFORCES H80
TFORCES J84
ROLNAME R16X10 R16X10
ENVWTS ''0.126 0.126 0.042 0.126''
ENVWTS ''0.054 0.054 0.036 0.036''
ENVWTS ''0.128 0.032 0.096 0.064''
ENVWTS ''0.016 0.040 0.016 0.008''
NEWOUT Y
BNODIV 41
USEFILES N N Y Y Y Y Y
SNAME HM80 JM84 80 81 82 83 84
ENAME R16X10 R16X10
CSFILE BADAT5 TADAT5 IADAT5
LOCALCS  N       N       N
CSPTR '' 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4''  .BASE
CSPTR '' 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4''  .TARGET
CSPTR '' 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4''  .INTERMEDIATE
FIL ADAT1 ADAT2 ADAT3 ADAT4 ACATS BBCAT RRCAT
NUM 11 12 13 14 16 17 18
ELTLIST Y
START N
VARY TRUE
@EOF
SKEL
*INCREMENT IDX TO [ALLDIV,1]
*CREATE SGS: IDIV [ALLDIV,1,IDX,1]
*CREATE SGS: IBFOR [DIVNUM,1,IDX,1]
*IF [START,1,1,1] = Y
*BRKPT,K H7RUNS.GO[IDIV,1,1,1]
*EDIT ON
#RUN,/TPR D03C[IDIV,1,1,1,0,1,2],H3899T2277D,UNCLASSIFIED,600,1000&
  . COOPER 762-8786
*ELSE
*BRKPT,K 30GECTEST.INTERPOL/[IDIV,1,1,1]
```

Figure I-5. Example of SSG Program for the Generation of Runstreams for Execution of the AFP Interpolation Module (page 1 of 6 pages)

```
*END
#HDG UNCLASSIFIED AFP INTERPOL [BFORCE,1,1,1] [IDIV,1,1,1] [TFORCE,1,1,
*IF [ELTLIST,1,1,1] = Y
#ELT,L 30GECTEST.INTERPSSG/SKEL
*IF [START,1,1,1] = Y
#ELT,L H7RUNS.GO[IDIV,1,1,1]
*ELSE
#ELT,L 30GECTEST.INTERPOL/[IDIV,1,1,1]
*END
*END
*INCREMENT A TO [FIL,1]
#ASG,T [NUM,1,A,1].,///100
#ED 30GECTEST.[FIL,1,A,1],[NUM,1,A,1].
*LOOP     .A
*IF [LOCALCS,1,1,1] = Y
#ASG,T    [CSFILE,1,1,1].,///100   . BASE CSCSS FACTORS
#DATA,IL   [CSFILE,1,1,1].
 1  1  1.0      1.0      1.0      1.0
 2  1  1.0      1.0      1.0      1.0
 3  1  1.0      1.0      1.0      1.0
 4  1  1.0      1.0      1.0      1.0
 5  1  1.0      1.0      1.0      1.0
 6  1  1.0      1.0      1.0      1.0
 7  1  1.0      1.0      1.0      1.0
 8  1  1.0      1.0      1.0      1.0
 9  1  1.0      1.0      1.0      1.0
 1  2  1.0      1.0      1.0      1.0
 2  2  1.0      1.0      1.0      1.0
 3  2  1.0      1.0      1.0      1.0
 4  2  1.0      1.0      1.0      1.0
 5  2  1.0      1.0      1.0      1.0
 6  2  1.0      1.0      1.0      1.0
 7  2  1.0      1.0      1.0      1.0
 8  2  1.0      1.0      1.0      1.0
 9  2  1.0      1.0      1.0      1.0
 1  3  1.0      1.0      1.0      1.0
 2  3  1.0      1.0      1.0      1.0
 3  3  1.0      1.0      1.0      1.0
 4  3  1.0      1.0      1.0      1.0
 5  3  1.0      1.0      1.0      1.0
 6  3  1.0      1.0      1.0      1.0
 7  3  1.0      1.0      1.0      1.0
 8  3  1.0      1.0      1.0      1.0
 9  3  1.0      1.0      1.0      1.0
 1  4  1.0      1.0      1.0      1.0
 2  4  1.0      1.0      1.0      1.0
 3  4  1.0      1.0      1.0      1.0
 4  4  1.0      1.0      1.0      1.0
 5  4  1.0      1.0      1.0      1.0
 6  4  1.0      1.0      1.0      1.0
```

Figure I-5. Example of SSG Program for the Generation of Runstreams for Execution of the AFP Interpolation Module (page 2 of 6 pages)

```
 7   4   1.0        1.0        1.0        1.0
 8   4   1.0        1.0        1.0        1.0
 9   4   1.0        1.0        1.0        1.0
#END
*ELSE
#ASG,T [CSFILE,1,1,1].           . BASE CSCSS FACTORS
#ED,I [CSFILE,1,1,1].
#EDIT
*INCREMENT IX FROM 1 TO 4
ADD H7CSCSSI.[BFORCEC,1,1,1]E0[*IX]
*LOOP      .IX
EXI
*END
*IF [LOCALCS,1,2,1] = Y
#ASG,T    [CSFILE,1,2,1].,///100   . TARGET CSCSS FACTORS
#DATA,IL   [CSFILE,1,2,1].
 1   1   1.0        1.0        1.0        1.0
 2   1   1.0        1.0        1.0        1.0
 3   1   1.0        1.0        1.0        1.0
 4   1   1.0        1.0        1.0        1.0
 5   1   1.0        1.0        1.0        1.0
 6   1   1.0        1.0        1.0        1.0
 7   1   1.0        1.0        1.0        1.0
 8   1   1.0        1.0        1.0        1.0
 9   1   1.0        1.0        1.0        1.0
 1   2   1.0        1.0        1.0        1.0
 2   2   1.0        1.0        1.0        1.0
 3   2   1.0        1.0        1.0        1.0
 4   2   1.0        1.0        1.0        1.0
 5   2   1.0        1.0        1.0        1.0
 6   2   1.0        1.0        1.0        1.0
 7   2   1.0        1.0        1.0        1.0
 8   2   1.0        1.0        1.0        1.0
 9   2   1.0        1.0        1.0        1.0
 1   3   1.0        1.0        1.0        1.0
 2   3   1.0        1.0        1.0        1.0
 3   3   1.0        1.0        1.0        1.0
 4   3   1.0        1.0        1.0        1.0
 5   3   1.0        1.0        1.0        1.0
 6   3   1.0        1.0        1.0        1.0
 7   3   1.0        1.0        1.0        1.0
 8   3   1.0        1.0        1.0        1.0
 9   3   1.0        1.0        1.0        1.0
 1   4   1.0        1.0        1.0        1.0
 2   4   1.0        1.0        1.0        1.0
 3   4   1.0        1.0        1.0        1.0
 4   4   1.0        1.0        1.0        1.0
 5   4   1.0        1.0        1.0        1.0
 6   4   1.0        1.0        1.0        1.0
 7   4   1.0        1.0        1.0        1.0
```

Figure I-5. Example of SSG Program for the Generation of Runstreams for Execution of the AFP Interpolation Module (page 3 of 6 pages)

```
 8  4  1.0     1.0     1.0     1.0
 9  4  1.0     1.0     1.0     1.0
#END
*ELSE
#ASG,T [CSFILE,1,2,1].        . TARGET CSCSS FACTORS
#ED,I [CSFILE,1,2,1].
#EDIT
*INCREMENT IX FROM 1 TO 4
ADD H7CSCSSI.[TFORCEC,1,1,1]E0[*IX]
*LOOP     .IX
EXI
*END
#ASG,A    H7BASEDATA.           . INVENTOPIES
#ASG,T    30INVB.,///4000
#ASG,T    30INVT.,///4000
#ASG,T    30INVI.,///4000
*IF [USEFILES,1,1,1] = Y
#ASG,A     H7[BFORCES,1,1,1][ROLNAME,1,1,1].  . BASE ROLLUP
#USE B.,H7[BFORCES,1,1,1][ROLNAME,1,1,1].
*ELSE
#ASG,T B.
#ASG,A H7[SNAME,1,1,1].
#ED H7[SNAME,1,1,1].[ENAME,1,1,1],B.    . BASE ROLLUP
*END
*IF [USEFILES,1,2,1] = Y
#ASG,A H7[TFORCES,1,1,1][ROLNAME,1,2,1].    . TARGET ROLLUP
#USE T.,H7[TFORCES,1,1,1][ROLNAME,1,2,1].
*ELSE
#ASG,T T.
#ASG,A H7[SNAME,1,2,1].
#ED H7[SNAME,1,2,1].[ENAME,1,2,1],T.    . TARGET ROLLUP
*END
#ED   H7BASEDATA.[BFORCE,1,1,1]E01,30INVB.
#ED   H7BASEDATA.[TFORCE,1,1,1]E01,30INVT.
#ASG,T    [CSFILE,1,3,1].,///100  . INTERMEDIATE CSCSS FACTORS
*INCREMENT FO TO [IFORCE,1]
*EDIT ON
#HDG UNCLASSIFIED AFP INTERPOL [BFORCE,1,1,1] [IDIV,1,1,1]&
[IFORCE,1,FO,1] [TFORCE,1,1,1]
#ERS 30INVI.
#ED   H7BASEDATA.[IDIV,1,1,1][IFORCE,1,FO,1]E01,30INVI.
#ERS [CSFILE,1,3,1].
*IF [LOCALCS,1,3,1] = Y
#DATA,IL   [CSFILE,1,3,1].
 1  1  1.0     1.0     1.0     1.0
 2  1  1.0     1.0     1.0     1.0
 3  1  1.0     1.0     1.0     1.0
 4  1  1.0     1.0     1.0     1.0
 5  1  1.0     1.0     1.0     1.0
 6  1  1.0     1.0     1.0     1.0
```

Figure I-5. Example of SSG Program for the Generation of Runstreams for Execution of the AFP Interpolation Module (page 4 of 6 pages)

```
        7   1   1.0        1.0        1.0        1.0
        8   1   1.0        1.0        1.0        1.0
        9   1   1.0        1.0        1.0        1.0
        1   2   1.0        1.0        1.0        1.0
        2   2   1.0        1.0        1.0        1.0
        3   2   1.0        1.0        1.0        1.0
        4   2   1.0        1.0        1.0        1.0
        5   2   1.0        1.0        1.0        1.0
        6   2   1.0        1.0        1.0        1.0
        7   2   1.0        1.0        1.0        1.0
        8   2   1.0        1.0        1.0        1.0
        9   2   1.0        1.0        1.0        1.0
        1   3   1.0        1.0        1.0        1.0
        2   3   1.0        1.0        1.0        1.0
        3   3   1.0        1.0        1.0        1.0
        4   3   1.0        1.0        1.0        1.0
        5   3   1.0        1.0        1.0        1.0
        6   3   1.0        1.0        1.0        1.0
        7   3   1.0        1.0        1.0        1.0
        8   3   1.0        1.0        1.0        1.0
        9   3   1.0        1.0        1.0        1.0
        1   4   1.0        1.0        1.0        1.0
        2   4   1.0        1.0        1.0        1.0
        3   4   1.0        1.0        1.0        1.0
        4   4   1.0        1.0        1.0        1.0
        5   4   1.0        1.0        1.0        1.0
        6   4   1.0        1.0        1.0        1.0
        7   4   1.0        1.0        1.0        1.0
        8   4   1.0        1.0        1.0        1.0
        9   4   1.0        1.0        1.0        1.0
#END
*ELSE
#ED,I [CSFILE,1,3,1].
#EDIT
*INCREMENT IX FROM 1 TO 4
ADD H7CSCSSI.[IDIV,1,1,1][IFORCE,1,FO,1]EO[*IX]
*LOOP      .IX
EXI
*END
*IF [USEFILES,1,FO+2,1] = Y
*IF [NEWOUT,1,1,1] = Y
#DELETE,C H7INT[IDIV,1,1,1][IFORCE,1,FO,1].
#ASG,UP   H7INT[IDIV,1,1,1][IFORCE,1,FO,1].    . INTERP OUTPUT FILE
#KEEP,C   H7INT[IDIV,1,1,1][IFORCE,1,FO,1].
*ELSE
#ASG,A H7INT[IDIV,1,1,1][IFORCE,1,FO,1].    . INTERP OUTPUT FILE
*END
#USE  30.,H7INT[IDIV,1,1,1][IFORCE,1,FO,1].
*ELSE
#ASG,T 30.     .INTERP TEMP OUTPUT FILE
```

Figure I-5. Example of SSG Program for the Generation of Runstreams for Execution of the AFP Interpolation Module (page 5 of 6 pages)

```
#ASG,A H7[IDIV,1,1,1][SNAME,1,FO+2,1].
*END
#ERS  30.
#XQT   30DOFAX.INTERPO
[PHI,1,FO,1]  [VARY,1,1,1]
*EDIT ON
[BNODIV,1]&
*INCREMENT IBNO TO [BNODIV,1]
 [BNODIV,1,IBNO,1]&
*LOOP    .IBNO
*EDIT OFF
*INCREMENT CSFIL TO 3
[CSFILE,1,CSFIL,1].
[CSPTR,CSFIL,1,1]
*LOOP    .CSFIL
*INCREMENT EW TO [ENVWTS]
[ENVWTS,EW,1,1]
*LOOP    .EW
30INVB.
30INVT.
30INVI.
*. IMAGE GIVING INDICES IN CBT POT OUTPUT RECORDS
*EDIT ON
[KTHTR,1,1,1]&
[JTPD,1,FO,1]&
  0  0  C&
  [IBFOR,1,1,1]&
[IRFOR,1,1,1]&
[JCASE,1,FO,1]
B.
T.
DONE
*IF [USEFILES,1,FO+2,1] = N
#ED 30.,H7[IDIV,1,1,1][SNAME,1,FO+2,1].INT
*END
#DATA,L 30.
#END
#FREE 30.
*LOOP    .FO
#DATA,L B.
#END
#DATA,L T.
#END
*INCREMENT A TO [NUM,1]
#FREE [NUM,1,A,1].
*LOOP    .A
#FREE B.
#FREE T.
*IF [START,1,1,1] = Y
#FIN
*END
*REMOVE SGS IDIV
*REMOVE SGS IBFOR
*LOOP    .IDX
@EOF
@EOF
```

Figure I-5. Example of SSG Program for the Generation of Runstreams for Execution of the AFP Interpolation Module (page 6 of 6 pages)

(10) The SGS "TFORCE" specifies the symbol (here JM84) identifying the target division in many of the input files/elements. A separate identifier is provided for CS/CSS factors.

(11) The SGS "BFORCEC" specifies the symbol (here HM00) identifying the baseline division in CS/CSS factor input. The symbol may be the same as for BFORCE if so defined at CS/CSS factor generation time.

(12) The SGS "TFORCEC" specifies the symbol (here JM00) identifying the target division in CS/CSS factor input. The symbol may be the same as for TFORCE if so defined at CS/CSS factor generation time.

(13) The SGS "BFORCES" specifies the symbol (here H80) identifying the baseline division in final combat potential input file. The name is critical only if the corresponding USEFILES symbol (defined below) is Y(es. The symbol may be the same as for TFORCE is so defined at rollup time.

(14) The SGS "TFORCES" specifies the symbol (here J84) identifying the target division in final combat potential input file. The name is critical only if the corresponding USEFILES symbol (defined below) is Y(es. The symbol may be the same as for TFORCE is so defined at rollup time.

(15) The SGS "ROLNAME" specifies the symbols identifying parts of the names of final combat potential input files for the baseline and target divisions. (Here both symbols are R16x2, suggesting that the final potentials of both baseline and target divisions were determined by rollups over all 16 combat environments and two replications.) ROLNAMEs are used only if the corresponding USEFILES setting is Y(es.

(16) The SGS "ENVWTS" specifies the 16 weights (summing to 1.0) to be used in "rolling up" over all 16 combat environments.

(17) The SGS "NEWOUT" specifies whether new files must be opened for receiving the results of interpolations. Y(es or N(o specifies whether the output file is a new one.

(18) The SGS "BNODIV" specifies which Blue weapon types (here only 41) are to be reqarded as nondivisional. The effect is to exclude nondivisional weapons from COP computation.

(19) The SGS "USEFILES" specifies whether combat potential data are to be treated as files Y(es or elements N(o. The symbols correspond to baseline, target, and intermediate divisions.

(20) The SGS "SNAME" specifies parts of the names of the files containing combat potential elements. The symbols are critical only if the corresponding USEFILES are set to N(o.

(21) The SGS "ENAME" specifies the element names of combat potentials of the baseline and target divisions. The symbols are critical only if the corresponding USEFILES are set to N(o.

(22) The SGS "CSFILE" specifies names of temporary files for CS/CSS factors for baseline, target, and intermediate divisions.

(23) The SGS "LOCALCS" specifies whether CS/CSS factors are to be used as defined within the generator (here all 1.0) or as defined in other sources. Y(es specifies use of the data in the generator; N(o specifies an external source. Specifications are set separately for baseline, target, and intermediate divisions. Only one switch is set for the intermediate divisions; i.e., it is assumed that the same data strategy (though not necessarily the same data) apply to all the intermediate divisions.

(24) The SGS "CSPTR" specify the combat environment sets of CS/CSS factors to be used in each of the 16 combat environments for each of the baseline, target, and intermediate divisions. It is assumed that the same specification applies to all intermediate divisions. The SGSs correspond to baseline, target, and intermediate divisions, respectively. The CS/CSS factor strategy of the combat module is to read one or more sets of factors for each division and then to use the sets by combat environment in accord with CSPTR.

(25) The SGS "FIL" specifies the names of elements containing data used in the calculation of CS/CSS moduli.

(26) The SGS "NUM" specifies the temporary files into which the elements specified in FIL are copied respectively.

(27) The SGS "ELTLIST" specifies whether the SSG program and corresponding runstream are to be listed in run output. Y(es produces the listings. Note that Y(es leads to printing the SSG program current at the time of runstream excution; that version may not be the one current at the time of runstream generation.

(28) The SGS "START" specifies whether runstreams are to be generated as START elements Y(es or simple ADD elements N(o. START elements begin with @RUN and end with @FIN images.

(29) The SGS "VARY" specifies whether the phase parameter PHI is to affect the computation of intermediate unmodulated combat potentials.

b. SKEL Section. The SKEL section of Figure I-5 creates names of files and elements and control images in accord with the above SGS definitions. Generated runstream elements are saved for careful inspection prior to their execution. The SKEL section is hardly the most general program imaginable. Special AFP cases or conditions may require modification of both SGS and SKEL. However, many special, one-time variations may be handled most easily by editing a standard generated runstream without change to the SSG program. Note that the example program is based on the assumption that all versions of the intermediate division have data named consistently. On the other hand, the program already includes SGSs permitting some variation in naming; e.g., HM80, HM00, H80, were all necessary during system development and test because analysts exercised nonstandard initiatives in naming files and elements. (It is a law of nature that analysts follow naming conventions no more than half the time.) Changes to the limits in lines 40 and 185 permit generation of runstreams for any consecutive subsets of divisions and years.

I-11. GENERATED RUNSTREAM I. Figure I-6 displays the runstream generated as a START element for the division 3M in accord with the example SSG program shown in Figure I-5. The example shown generates 12 other runstreams during single execution.

I-12. GENERATED RUNSTREAM II. Figure I-7 displays a runstream generated as an ADD element for the division 24M in accord with the example SSG program shown in Figure I-5 but with "START N". That example (with "START N") would generate 12 other ADD elements as well.

```
@RUN,/TPR DOG63M,H3899T2277D,UNCLASSIFIED,600,1000   .
@HDG UNCLASSIFIED AFP INTERPOL HM80 3M JM84
@ELT,L G6GECTEST.INTERPSSG/SKEL
@ELT,L H7RUNS.G03M
@ASG,T 11.,///100
@ED G6GECTEST.ADAT1,11.
@ASG,T 12.,///100
@ED G6GECTEST.ADAT2,12.
@ASG,T 13.,///100
@ED G6GECTEST.ADAT3,13.
@ASG,T 14.,///100
@ED G6GECTEST.ADAT4,14.
@ASG,T 16.,///100
@ED G6GECTEST.ACATS,16.
@ASG,T 17.,///100
@ED G6GECTEST.BBCAT,17.
@ASG,T 18.,///100
@ED G6GECTEST.RRCAT,18.
@ASG,T BADAT5.        . BASE CSCSS FACTORS
@ED,I BADAT5.
@EDIT
ADD H7CSCSSI.HM00E01
ADD H7CSCSSI.HM00E02
ADD H7CSCSSI.HM00E03
ADD H7CSCSSI.HM00E04
EXI
@ASG,T TADAT5.       . TARGET CSCSS FACTORS
@ED,I TADAT5.
@EDIT
ADD H7CSCSSI.JM00E01
ADD H7CSCSSI.JM00E02
ADD H7CSCSSI.JM00E03
ADD H7CSCSSI.JM00E04
EXI
@ASG,A   H7BASEDATA.          . INVENTORIES
@ASG,T   G6INVB.,///4000
@ASG,T   G6INVT.,///4000
@ASG,T   G6INVI.,///4000
@ASG,T B.
@ASG,A H7HM80.
@ED H7HM80.R16X10,B.     . BASE ROLLUP
@ASG,T T.
@ASG,A H7JM84.
@ED H7JM84.R16X10,T.     . TARGET ROLLUP
@ED  H7BASEDATA.HM80E01,G6INVB.
@ED  H7BASEDATA.JM84E01,G6INVT.
@ASG,T   IADAT5.,///100   . INTERMEDIATE CSCSS FACTORS
@HDG UNCLASSIFIED AFP INTERPOL HM80 3M80 JM84
@ERS G6INVI.
@ED  H7BASEDATA.3M80E01,G6INVI.
```

Figure I-6. First Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module
(page 1 of 5 pages)

```
@ERS IADAT5.
@ED,I IADAT5.
@EDIT
ADD H7CSCSSI.3M80E01
ADD H7CSCSSI.3M80E02
ADD H7CSCSST.3M80E03
ADD H7CSCSSI.3M80E04
EXI
@DELETE,C H7INT3M80.
@ASG,UP   H7INT3M80.     . INTERP OUTPUT FILE
@KEEP,C   H7INT3M80.
@USE  30.,H7INT3M80.
@ERS  30.
@XGT   G6DOFAX.INTERPO
0.05  TRUE
1 41
BADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
TADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
IADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
0.126 0.126 0.042 0.126
0.054 0.054 0.036 0.036
0.126 0.032 0.096 0.064
0.016 0.040 0.016 0.008
G6INVB.
G6INVT.
G6INVI.
  E  80  0  0  0  2101  1000 8032
B.
T.
DONE
@DATA,L 30.
@END
@FREE 30.
@HDG UNCLASSIFIED AFP INTERPOL HY80 3M81 JM84
@ERS G6INVI.
@ED  H7BASEDATA.3M81E01,G6INVI.
@ERS IADAT5.
@ED,I IADAT5.
@EDIT
ADD H7CSCSSI.3M81E01
ADD H7CSCSSI.3M81E02
ADD H7CSCSSI.3M81E03
ADD H7CSCSSI.3M81E04
EXI
@DELETE,C H7INT3M81.
@ASG,UP   H7INT3M81.     . INTERP OUTPUT FILE
@KEEP,C   H7INT3M81.
```

Figure I-6. First Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module (page 2 of 5 pages)

```
@USE   30.,H7INT3M81.
@ERS   30.
@XGT    G6DOFAY.INTERPO
0.25   TRUE
1  41
BADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
TADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
IADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
0.126 0.126 0.042 0.126
0.054 0.054 0.036 0.036
0.128 0.032 0.096 0.064
0.016 0.040 0.016 0.008
G6INVB.
G6INVT.
G6INVI.
  E  81   0   0   0   2101   1000 8132
B.
T.
DONE
@DATA,L 30.
@END
@FREE 30.
@HDG UNCLASSIFIED AFP INTERPOL HM80 3M82 JM84
@ERS G6INVI.
@ED  H7BASEDATA.3M82E01,G6INVI.
@ERS IADAT5.
@ED,I IADAT5.
@EDIT
ADD H7CSCSSI.3M82E01
ADD H7CSCSSI.3M82E02
ADD H7CSCSSI.3M82E03
ADD H7CSCSSI.3M82E04
EXI
@DELETE,C H7INT3M82.
@ASG,UP  H7INT3M82.     . INTERP OUTPUT FILE
@KEEP,C  H7INT3M82.
@USE   30.,H7INT3M82.
@ERS   30.
@XGT    G6DOFAY.INTERPO
0.50   TRUE
1  41
BADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
TADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
IADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
```

Figure I-6. First Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module
page 3 of 5 pages)

```
0.126 0.126 0.042 0.126
0.054 0.054 0.036 0.036
0.128 0.032 0.096 0.064
0.016 0.040 0.016 0.008
G6INVB.
G6INVT.
G6INVI.
  E  82  0  0  0  2101  1000 8232
B.
T.
DONE
@DATA,L 30.
@END
@FREE 30.
@HDG UNCLASSIFIED AFP INTERPOL HM30 3M83 JM84
@ERS G6INVI.
@ED  H7BASEDATA.3M83E01,G6INVI.
@ERS IADAT5.
@ED,I IADAT5.
@EDIT
ADD H7CSCSSI.3M83F01
ADD H7CSCSSI.3M83F02
ADD H7CSCSSI.3M83F03
ADD H7CSCSSI.3M83F04
EXI
@DELETE,C H7INT3M83.
@ASG,UP  H7INT3M83.     . INTERP OUTPUT FILE
@KEEP,C  H7INT3M83.
@USE  30.,H7INT3M83.
@ERS  30.
@XQT   G6DOFAX.INTERPO
0.75  TRUE
1  41
BADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
TADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
IADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
0.126 0.126 0.042 0.126
0.054 0.054 0.036 0.036
0.128 0.032 0.096 0.064
0.016 0.040 0.016 0.008
G6INVB.
G6INVT.
G6INVI.
  E  83  0  0  0  2101  1000 8372
B.
T.
DONE
```

Figure I-6. First Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module (page 4 of 5 pages)

```
@DATA,L 30.
@END
@FREE 30.
@HDG UNCLASSIFIED AFP INTERPOL HM80 3M84 JM84
@ERS G6INVI.
@ED  H7BASEDATA.3M84E01,G6INVI.
@ERS IADAT5.
@ED,I IADAT5.
@EDIT
ADD H7CSCSSI.3M84E01
ADD H7CSCSSI.3M84E02
ADD H7CSCSSI.3M84E03
ADD H7CSCSSI.3M84E04
EXI
@DELETE,C H7INT3M84.
@ASG,UP   H7INT3M84.     . INTERP OUTPUT FILE
@KEEP,C   H7INT3M84.
@USE  30.,H7INT3M84.
@ERS  30.
@XQT   G6DOFAX.INTERPO
0.95  TRUE
1 41
BADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
TADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
IADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
0.126 0.126 0.042 0.126
0.054 0.054 0.036 0.036
0.128 0.032 0.096 0.064
0.016 0.040 0.016 0.006
G6INVB.
G6INVT.
G6INVI.
  E  84   0   0   0  2101  1000 8432
B.
T.
DONE
@DATA,L 30.
@END
@FREE 30.
@DATA,L B.
@END
@DATA,L T.
@END
@FREE 11.
@FREE 12.
@FREE 13.
@FREE 14.
@FREE 16.
@FREE 17.
@FREE 18.
@FREE B.
@FREE T.
@FIN
```

Figure I-6. First Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module (page 5 of 5 pages)

```
@HDG UNCLASSIFIED AFP INTERPOL HM80 24M JM84
@ELT,L 30GECTEST.INTERPSSG/SKEL
@ELT,L 30GECTEST.INTERPOL/24M
@ASG,T 11.,///100
@ED 30GECTEST.ADAT1,11.
@ASG,T 12.,///100
@ED 30GECTEST.ADAT2,12.
@ASG,T 13.,///100
@ED 30GECTEST.ADAT3,13.
@ASG,T 14.,///100
@ED 30GECTEST.ADAT4,14.
@ASG,T 16.,///100
@ED 30GECTEST.ACATS,16.
@ASG,T 17.,///100
@ED 30GECTEST.BBCAT,17.
@ASG,T 18.,///100
@ED 30GECTEST.RRCAT,18.
@ASG,T BADAT5.          . BASE CSCSS FACTORS
@ED,I BADAT5.
@EDIT
ADD H7CSCSSI.HM00E01
ADD H7CSCSSI.HM00E02
ADD H7CSCSSI.HM00E03
ADD H7CSCSSI.HM00E04
EXI
@ASG,T TADAT5.          . TARGET CSCSS FACTORS
@ED,I TADAT5.
@EDIT
ADD H7CSCSSI.JM00E01
ADD H7CSCSSI.JM00E02
ADD H7CSCSSI.JM00E03
ADD H7CSCSSI.JM00E04
EXI
@ASG,A    H7BASEDATA.              . INVENTORIES
@ASG,T    30INVB.,///4000
@ASG,T    30INVT.,///4000
@ASG,T    30INVI.,///4000
@ASG,T B.
@ASG,A H7HM80.
@ED H7HM80.R16X10,B.     . BASE ROLLUP
@ASG,T T.
@ASG,A H7JM84.
@ED H7JM84.R16X10,T.     . TARGET ROLLUP
@ED  H7BASEDATA.HM80E01,30INVB.
@ED  H7BASEDATA.JM84E01,30INVT.
@ASG,T    IADAT5.,///100   . INTERMEDIATE CSCSS FACTORS
@HDG UNCLASSIFIED AFP INTERPOL HM80 24M80 JM84
@ERS 30INVI.
@ED  H7BASEDATA.24M80E01,30INVI.
@ERS IADAT5.
```

Figure I-7. Second Example of an SSG-generated Runstream for Execution of the AFP Interpolation Model (page 1 of 5 pages)

```
@ED,I  IADAT5.
@EDIT
ADD  H7CSCSSI.24M80E01
ADD  H7CSCSSI.24M80E02
ADD  H7CSCSSI.24M80E03
ADD  H7CSCSSI.24M80E04
EXI
@DELETE,C  H7INT24M80.
@ASG,UP   H7INT24M80.     . INTERP OUTPUT FILE
@KEEP,C   H7INT24M80.
@USE   30.,H7INT24M80.
@ERS   30.
@XQT    30DOFAX.INTERPO
0.05   TRUE
1  41
BADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
TADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
IADAT5.
 1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
0.126  0.126  0.042  0.126
0.054  0.054  0.036  0.036
0.126  0.032  0.096  0.064
0.016  0.040  0.016  0.008
30INVB.
30INVT.
30INVI.
  E   80   0   0   0   3801   1000  8032
B.
T.
DONE
@DATA,L  30.
@END
@FREE  30.
@HDG  UNCLASSIFIED  AFP  INTERPOL  HM80  24M81  JM84
@ERS  30INVI.
@ED   H7BASEDATA.24M81E01,30INVI.
@ERS  IADAT5.
@ED,I  IADAT5.
@EDIT
ADD  H7CSCSSI.24M81E01
ADD  H7CSCSSI.24M81E02
ADD  H7CSCSSI.24M81E03
ADD  H7CSCSSI.24M81E04
EXI
@DELETE,C  H7INT24M81.
@ASG,UP   H7INT24M81.     . INTERP OUTPUT FILE
@KEEP,C   H7INT24M81.
@USE   30.,H7INT24M81.
```

Figure I-7. Second Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module (page 2 of 5 pages)

```
@ERS   30.
@XQT    30DOFAX.INTERPO
0.25   TRUE
1  41
BADAT5.
  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
TADAT5.
  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
IADAT5.
  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
0.126 0.126 0.042 0.126
0.054 0.054 0.036 0.036
0.128 0.032 0.096 0.064
0.016 0.040 0.016 0.008
30INVB.
30INVT.
30INVI.
   E   81   0   0   0   3801   1000 9132
B.
T.
DONE
@DATA,L 30.
@END
@FREE 30.
@HDG UNCLASSIFIED AFP INTERPOL HM80 24M82 JM84
@ERS 30INVI.
@ED   H7BASEDATA.24M82E01,30INVI.
@ERS IADAT5.
@ED,I IADAT5.
@EDIT
ADD H7CSCSSI.24M82E01
ADD H7CSCSSI.24M82E02
ADD H7CSCSSI.24M82E03
ADD H7CSCSSI.24M82E04
EXI
@DELETE,C H7INT24M82.
@ASG,UP   H7INT24M82.       . INTERP OUTPUT FILE
@KEEP,C   H7INT24M82.
@USE   30.,H7INT24M82.
@ERS   30.
@XQT    30DOFAX.INTERPO
0.50   TRUE
1  41
BADAT5.
  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
TADAT5.
  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
IADAT5.
  1  2  3  4  1  2  3  4  1  2  3  4  1  2  3  4
0.126 0.126 0.042 0.126
```

Figure I-7. Second Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module (page 3 of 5 pages)

```
0.054 0.054 0.036 0.036
0.128 0.032 0.096 0.064
0.016 0.040 0.016 0.008
30INVB.
30INVT.
30INVI.
  E  82  0  0  0  3801  1000 8232
B.
T.
DONE
@DATA,L 30.
@END
@FREE 30.
@HDG UNCLASSIFIED AFP INTERPOL HMED 24M83 JM84
@ERS 30INVI.
@ED  H7BASEDATA.24M83E01,30INVI.
@ERS IADAT5.
@ED,I IADAT5.
@EDIT
ADD H7CSCSSI.24M83E01
ADD H7CSCSSI.24M83E02
ADD H7CSCSSI.24M83E03
ADD H7CSCSSI.24M83E04
EXI
@DELETE,C H7INT24M83.
@ASG,UP  H7INT24M83.     . INTERP OUTPUT FILE
@KEEP,C  H7INT24M83.
@USE   30.,H7INT24M83.
@ERS   30.
@XQT    30DCFAX.INTERPO
0.75  TRUE
1 41
BADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
TADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
IADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
0.126 0.126 0.042 0.126
0.054 0.054 0.036 0.036
0.128 0.032 0.096 0.064
0.016 0.040 0.016 0.008
30INVB.
30INVT.
30INVI.
  E  83  0  0  0  3801  1000 8332
B.
T.
DONE
@DATA,L 30.
```

Figure I-7. Second Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module (page 4 of 5 pages)

```
@END
@FREE 30.
@HDG UNCLASSIFIED AFP INTERPOL HMED 24M84 JM84
@ERS 30INVI.
@FD  H7BASEDATA.24M84E01,30INVI.
@ERS IADAT5.
@FD,I IADAT5.
@EDIT
ADD H7CSCSSI.24M84E01
ADD H7CSCSSI.24M84E02
ADD H7CSCSSI.24M84E03
ADD H7CSCSSI.24M84E04
EXI
@DELETE,C H7INT24M84.
@ASG,UP  H7INT24M84.     . INTERP OUTPUT FILE
@KEEP,C  H7INT24M84.
@USE  30.,H7INT24M84.
@ERS  30.
@XQT   30DOFAX.INTERPO
0.95  TRUE
1 41
BADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
TADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
IADAT5.
 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
0.126 0.126 0.042 0.126
0.054 0.054 0.036 0.036
0.128 0.032 0.096 0.064
0.016 0.040 0.016 0.008
30INVB.
30INVT.
30INVI.
  E   84   0   0   0   3801   1000 8432
B.
T.
DONE
@DATA,L 30.
@END
@FREE 30.
@DATA,L B.
@END
@DATA,L T.
@END
@FREE 11.
@FREE 12.
@FREE 13.
@FREE 14.
@FREE 16.
@FREE 17.
@FREE 18.
@FREE B.
@FREE T.
```

Figure I-7. Second Example of an SSG-generated Runstream for Execution of the AFP Interpolation Module (page 5 of 5 pages)

Section V. PROGRAM

I-13. MOTIVATION. The AFP Interpolation Module implements logic for the interpolation of combat potentials of divisions with inventories "between" two bounding inventories whose combat potentials are already known. In its current form, the Interpolation Module estimates combat potentials in something less than 1/100 the computer time required to determine potentials from scratch. In absolute terms, the times are 10 minutes for interpolation and over 40 hours for full system execution with 10 Combat Module replications, per combat environment. The increase in speed is accompanied by some loss in accuracy.

I-14. PROGRAM LOGIC. Figure I-8 displays the basic logical flow of the Interpolation Module. The flow diagram does not reveal the interpolation technique, which is imbedded in the subprograms GETINV through XMCIP and, to a degree, in OUTX, the output routine. The algorithm is explained in paragraph I-16. But before coming to grips with the algorithm, the operator/programer should pause to consider that much more than a single quantity must be estimated. The Interpolation Module must estimate the unmodulated and modulated scores and CIPs for up to 60 Blue weapon types. Inasmuch as scores are merely the products of equipment quantities and corresponding CIPs, it is only the CIPs that need be interpolated in the usual sense of the word. Once CIPs have been estimated, simple multiplication yields scores. And once scores have been determined, summing the unmodulated and modulated scores yields the unmodulated and modulated COPs, respectively. The unmodulated and modulated CIPs are determined differently. Because the explanation of the algorithm involves references to Interpolation Module program arrays, the module's reference and working arrays are defined in the next paragraph, before detailed description of interpolation.

a. The key to the interpolation method is recognition of the need to determine net CS/CSS moduli for the three inventories. This is a simple matter for the baseline and target inventories. For them, net moduli are defined simply as the ratios of modulated to unmodulated scores from the files of final combat potentials. Those files are available from the AFP Rollup and Stats Module. For a specific weapon type, let the net moduli be represented by M(B) and M(T) for the baseline (B) and target (T) inventories, respectively. Needed are corresponding M(I) for the intermediate (I) force. But, of course, no file of combat potentials already exists for the intermediate force; it is that file which is to be generated by the Interpolation Module. However, data are already available to construct so-called "synthetic moduli" for all three forces.

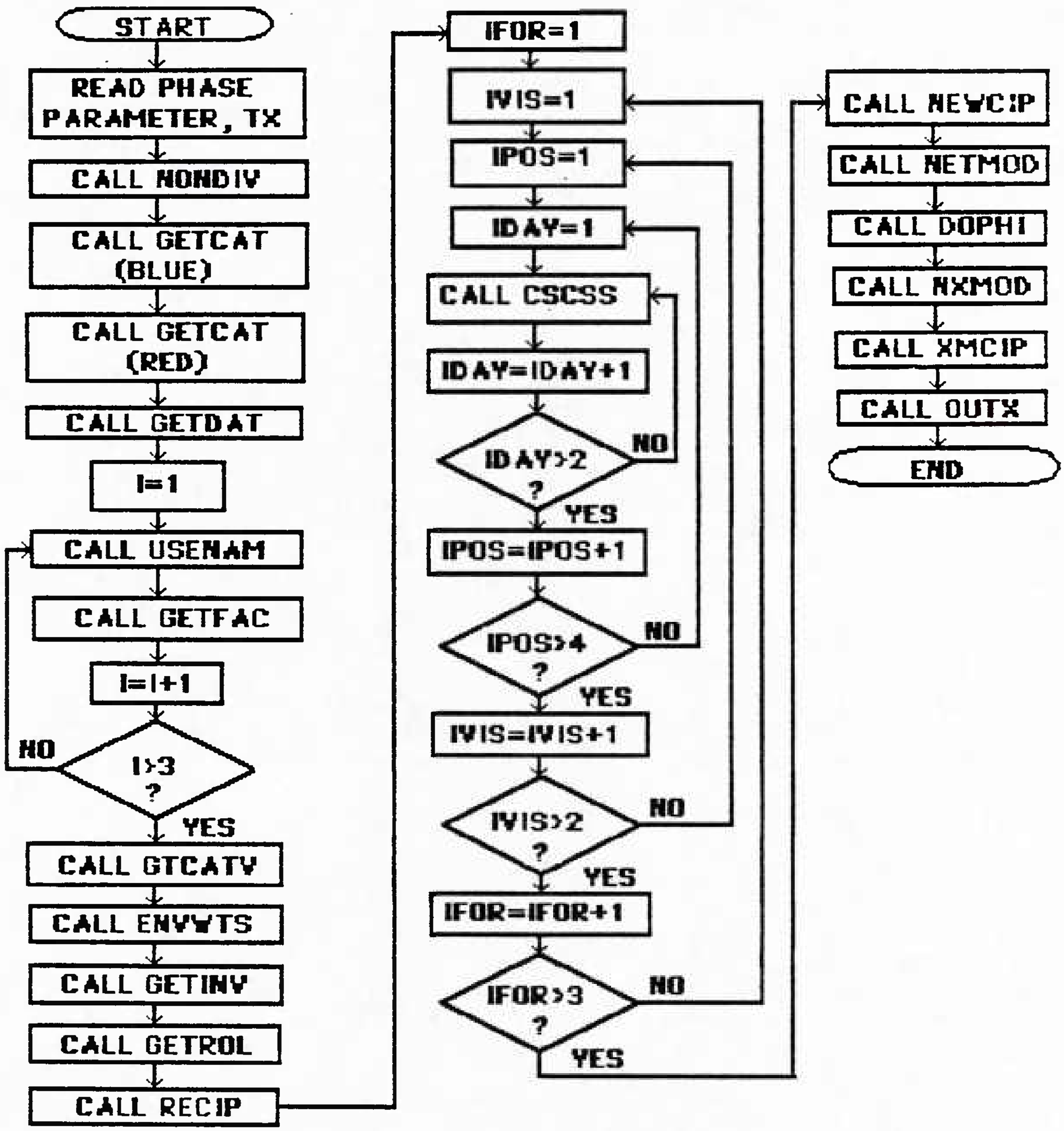


Figure I-8. Flow Diagram of the Basic Logic of the AFP Interpolation Module

b. An implied table has the form:

	Baseline	Intermediate	Target
Net modulus	M(B)	?	M(T)
Synthetic modulus	S(B)	S(I)	S(T)

where only the "?" for M(I) is unknown. It is assumed that the following relation holds:

$$M(I)/S(I) = (1-p) * M(B)/S(B) + p * M(T)/S(T)$$

hence, M(I) is determined as

$$M(I) = S(I) * ((1-p) * M(B)/S(B) + p * M(T)/S(T))$$

where "p" is a so-called "phase parameter" reflecting whether the intermediate division is "closer" to the baseline or target division. The parameter should lie on the range $0.0 <= p <= 1.0$. The algorithm described in paragraph I-16 below implements this notion of determining M(I) in terms of M(B), S(B), M(T), S(T), and p for all components of modulated scores and CIPs for all Blue weapon types.

I-15. PROGRAM ARRAYS. The principal arrays defined and used in the Interpolation Module are referenced in the paragraph I-16 description of the interpolation algorithm.

a. QTY(60,3). The array QTY() stores the weapon inventories of the baseline, target, and intermediate divisions for reference. Once stored, the inventories are not modified. An array element QTY(I,J) is indexed:

(1) I=1 to 60 for the 60 Blue weapon types.

(2) J=1 to 3 for the three divisions: baseline, target, and intermediate.

b. RSCORE(5,60,2). The array RSCORE() stores the five-valued final unmodulated scores for weapons of the baseline and target divisions. Once stored, the scores are not modified. The array RSCORE() is sometimes treated as the first subarray of the larger array WVAL(5,60,2,4). An array element RSCORE(K,J,K) is indexed:

(1) I=1 to 5 for the five components of unmodulated scores: personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar.

(2) J=1 to 60 for the 60 Blue weapon types.

(3) K=1 to 2 for the baseline and target divisions.

c. XSCORE(5,60,2). The array XSCORE() stores the five-valued final modulated scores for weapons of the baseline and target divisions. Once stored, the scores are not modified. The array XSCORE() is sometimes treated as the third subarray of the larger array WVAL(5,60,2,4). An array element XSCORE(I,J,K) is indexed:

(1) I=1 to 5 for the five components of modulated scores: personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar.

(2) J=1 to 60 for the 60 Blue weapon types.

(3) K=1 to 2 for the baseline and target divisions.

d. UCIP(5,60,2). The array UCIP() stores the five-valued final unmodulated CIPs for weapons of the baseline and target divisions. Although the CIPs are available directly from the files of final combat potentials for the baseline and target divisions, the values in those files are the result of truncation of usually more precise values. The Interpolation Module restores some of that precision by dividing the corresponding unmodulated score by the inventory quantities. The array UCIP() is sometimes treated as the second subarray of the larget array WVAL(5,60,2,4). An array element UCIP(I,J,K) is indexed:

(1) I=1 to 5 for the five components of unmodulated CIPs: personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar.

(2) J=1 to 60 for the 60 Blue weapon types.

(3) K=1 to 2 for the baseline and target divisions.

e. XCIP(5,60,2). The array XCIP() stores the five-valued final modulated CIPs for weapons of the baseline and target divisions. Although the CIPs are available directly from the files of final combat potentials for the baseline and target divisions, the values in those files are the result of truncation of usually more precise values. The Interpolation Module restores some of that precision by dividing the corresponding unmodulated score by the inventory quantities. The array XCIP() is sometimes treated as the fourth subarray of the larget array WVAL(5,60,2,4). An array element XCIP(I,J,K) is indexed:

(1) I=1 to 5 for the five components of unmodulated CIPs: personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar.

(2) J=1 to 60 for the 60 Blue weapon types.

(3) K=1 to 2 for the baseline and target divisions.

f. XNMOD(5,60,3). (This array is also sometimes identified as XMOD(5,60,3).) The array XNMOD() stores derived values of what are called the "net CS/CSS moduli." These special versions of CS/CSS moduli are generated during the interpolation process for the baseline, target, and intermediate divisions. The net moduli are generated first for the baseline and target divisions. These and other factors are then used to generate net moduli for the intermediate division. An array element XNMOD(I,J,K) is indexed:

(1) I=1 to 5 for the five components of combat potentials: personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar.

(2) J=1 to 60 for the 60 Blue weapon types.

(3) K=1 to 2 for the baseline, target, and intermediate divisions.

g. FAX(12,60,3). The array FAX() stores derived sample values of CS/CSS moduli for the baseline, target, and intermediate divisions. The values are derived within the Interpolation Module by the standard AFP CS/CSS approach for each combat posture and then (weighted) summed over posture. The values are samples in the sense that only one Red weapon per target category is considered. An array element FAX(I,J,K) is indexed:

(1) I=1 to 12 for the 12 Red target categories.

(2) J=1 to 60 for the 60 Blue weapon types.

(3) K=1 to 3 for the baseline, target, and intermediate divisions.

h. PHI(60,3). The array PHI() stores derived "synthetic CS/CSS moduli" for the baseline, target, and intermediate divisions. The values are simply the arithmetic means of elements in array FAX() averaged over the 12 Red target categories. Hence, an element of array PHI() is a special CS/CSS modulus already averaged over targets and combat postures. An array element PHI(I,J) is indexed:

(1) I=1 to 60 for the 60 Blue weapon types.

(2) J=1 to 3 for baseline, target, and intermediate divisions.

i. XUCIP(5,60). The array XUCIP() stores derived, interpolated estimates of unmodulated CIPs for the intermediate division. An array element XUCIP(I,J) is indexed:

(1) I=1 to 5 for the five components of unmodulated CIPs: personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar.

(2) J=1 to 60 for the 60 Blue weapon types of the intermediate division.

j. VCIP(5,60). The array VCIP() stores derived, interpolated estimates of modulated CIPs for the intermediate division. An array element VCIP(I,J) is indexed:

(1) I=1 to 5 for the five components of modulated CIPs: personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar.

(2) J=1 to 60 for the 60 Blue weapon types of the intermediate division.

k. EWT(16). The array EWT() stores combat environmental weights. The Interpolation Module reads the 16 environmental weights. An array element EWT(I) is indexed by combat environment.

l. INDX(16,3). The array INDX() stores pointers specifying the sets of CS/CSS factors to be used by combat environment and by division. An array element INDX(I,J) is indexed:

(1) I=1 to 16 for the combat environments.

(2) J=1 to 3 for the baseline, target, and intermediate divisions.

m. The subprogram CSCSS within the Interpolation Module is very nearly identical to a subprogram within the AFP System's CS/CSS Module. Apart from the arrays FAX() and EWT() introduced above, subprogram CSCSS uses the same arrays already identified and defined in Appendix E on the CS/CSS process in particular.

n. Note that the personnel through aircraft elements of combat potentials may include target values if so generated by the AFP CBT/CS/CSS Merge Module. If so, the "personnel" element then also may contain some other weapon values.

I-16. KEY TO INTERPOLATION ALGORITHM. Figure I-9 provides the principal track for describing the methods used to interpolate unmodulated and modulated CIPs. Once the new CIPs have been obtained, the unmodulated and modulated scores can be obtained by simple multiplication in steps not represented in Figure I-9. The description involves references to Interpolation Module program arrays (defined in paragraph I-15 immediately above) and special simplified symbols (in Figure I-9) representing arbitrary elements within those arrays.

a. The unmodulated scores and CIPs and the modulated scores and CIPs are already known for the baseline and target divisions for all weapon types. These are symbolized for the Ith component of combat potentials and Jth Blue weapon type in Figure I-9 by:

(1) Unmodulated score--BB = RSCORE(I,J,1) for the baseline division and LLL = RSCORE(I,J,2) for the target division.

(2) Unmodulated CIP--DDD = UCIP(I,J,1) for the baseline division and NNN = UCIP(I,J,2) for the target division.

(3) Modulated score--CCC = XSCORE(I,J,1) for the baseline division and MMM = XSCORE(I,J,2) for the target division.

(4) Modulated CIP--EEE = XCIP(I,J,1) for the baseline division and OOO = XCIP(I,J,2) for the target division.

AFP CIP INTERPOLATION

		BASELINE INVENTORY						INTERMEDIATE INVENTORY				TARGET INVENTORY					
STEP	SUB-ROUTINE	AAA= QTY (J,1)	BBB= RSCORE (I,J,1)	CCC= XSCORE (I,J,1)	DDD= UCIP (I,J,1)	EEE= XCIP (I,J,1)	FFF= XNMOD (I,J,1)	GGG= QTY (J,3)	HHH= XUCIP (I,J)	III= XNMOD (I,J,3)	JJJ= VCIP (I,J)	KKK= QTY (J,2)	LLL= RSCORE (I,J,2)	MMM= XSCORE (I,J,2)	NNN= UCIP (I,J,2)	OOO= XCIP (I,J,2)	PPP= XNMOD (I,J,2)
1	GETINV	AAA						GGG				KKK					
2	GETROL		BBB	CCC									LLL	MMM			
3	RECIP				DDD= BBB/AAA	EEE= CCC/AAA									NNN= LLL/KKK	OOO= MMM/KKK	
4	CSCSS						FAX (K,J,1)			FAX (K,J,3)							FAX (K,J,2)
5	NEWCIP (SEE NOTE#1)								HHH= (AAA* DDD + KKK* NNN)/ (AAA+ KKK)								
6	NETMOD						FFF= CCC/BBB										PPP= MMM/LLL
7	DOPHI						PHI(J, 1)=SUM(FAX(K,J ,1),K)/ 12			PHI(J, 3)=SUM(FAX(K,J ,3),K)/ 12							PHI(J, 2)=SUM(FAX(K,J ,2),K)/ 12
8	NXMOD									III= PHI(J,3)*((1-T)*FFF/ PHI(J,1)+T*PPP /PHI(J, 2))							
9	XMCIP										JJJ= III* HHH						

I-TH COMPONENT OF COMBAT POTENTIAL (PERSONNEL, LIGHT ARMORED VEHICLES, HEAVY ARMORED VEHICLES, AIRCRAFT, SCALAR).
J-TH BLUE WEAPON TYPE.
K-TH RED WEAPON CATEGORY (OF 12).

FAX MODULI BY RED WEAPON CATEGORY BY BLUE WEAPON TYPE; WEIGHTED SUM OVER FOUR COMBAT POSTURES.
PHI MODULI AVERAGED OVER RED WEAPON CATEGORIES.
T INVENTORY PHASE PARAMETER.
1,2,3 BASELINE, TARGET (FINAL), AND INTERMEDIATE INVENTORIES.

NOTE#1 OR HHH=(BBB+LLL)/(AAA+KKK)
BOTH SHOWN FOR PROGRAM VARIABLE VARY=.FALSE.
IF VARY=.TRUE., THEN
HHH=((1-T)*AAA*DDD+T*KKK*NNN)/((1-T)*AAA+T*KKK)

Figure I-9. Tabular Summary of the Steps in Computation of Interpolated Unmodulated and Modulated CIPs within the AFP Interpolation Module

b. It remains for the Interpolation Module to determine the counterparts of the quantities in paragraph a for the intermediate division. There is some freedom in the order in which steps may be performed. The order of steps shown in Figure I-9 is convenient but arbitrary in some respects.

(1) Step 1 in Figure I-9 corresponds to the call of subprogram GETINV in order to read and store the inventories of the baseline, target, and intermediate weapon inventories in array QTY() symbolized by the general elements AAA = QTY(J,1), KKK = QTY(J,2), and GGG = QTY(J,3), respectively.

(2) Step 2 in Figure I-9 corresponds to the call of subprogram GETROL in order to read and store the unmodulated and modulated scores for the baseline and target divisions in arrays RSCORE() and XSCORE() symbolized by the general elements BBB, CCC, LLL, and MMM, respectively.

(3) Step 3 in Figure I-9 corresponds to the call of subprogram RECIP in order to compute and store values of the unmodulated and modulated CIPs for the baseline and target divisions in arrays UCIP() and XCIP() symbolized by the general elements DDD, EEE, NNN, and OOO, respectively. Although values are available directly from the original combat potential files for the baseline and target divisions, the subprogram RECIP generates higher precision elements. In each case, a CIP is computed simply as the result of dividing a score by the inventory of the corresponding weapon type. Thus, DDD = BBB / AAA, for example. At this stage, enough information is already in hand for computation of the unmodulated CIPs for the intermediate division; however, this step is not performed until step 5 below.

(4) Step 4 in Figure I-9 corresponds to calls of subprogram CSCSS in order to compute sample CS/CSS moduli for the baseline, target, and intermediate divisions. The moduli are samples in the sense that they are computed for all Blue weapons types but only in relation to a single target type from each of the 12 Red weapon categories. Moduli are computed for each of the 16 combat environments and summed in accord with the usual combat environmental weights. The results of this step are stored in array FAX() for use later in step 7.

(5) Step 5 in Figure I-9 corresponds to the call (with argument VARY =,FALSE.) of program NEWCIP in order to compute the unmodulated CIPs for the intermediate division. Results are stored in array XUCIP() for later output. An element of the array XUCIP() is symbolized by HHH in Figure I-9. Note that the HHH are one of the desired classes of final combat potentials sought. The formula for the HHH is straightforward:

$$HHH = (AAA * DDD + KKK * NNN) / (AAA + KKK)$$

It is characteristic of the AFP System that the same weapon type may have different CIPs in different divisions. Thus, in general, the unmodulated CIPs DDD and NNN may be unequal. The problem is to choose unmodulated CIPs HHH "between" those values for the intermediate division. The given formula assigns the HHH in proportion to the quantities of the corresponding weapon types in each of the baseline and target inventories. Note that if the CIPs are the same in the baseline and target divisions, the same value is assigned to the intermediate division, regardless of the numbers of weapons of that type. If NEWCIP is called with VARY=.TRUE, HHH = ((1-PHI) * AAA * DDD + PHI * KKK * NNN) / ((1 - PHI) * AAA + PHI * KKK), this latter option permits unmodulated combat potentials to vary with PHI (the phase of the intermediate year) unless DDD + NNN.

(6) Step 6 in Figure I-9 corresponds to the call of subprogram NETMOD in order to compute and store what are called the "net CS/CSS moduli" implicit in the original results for the baseline and target divisions. In general, the modulated scores and CIPs for the baseline and target divisions are the results of summation over 16 combat environments and over many engagements; hence, those results involve the application of possibly very many CS/CSS moduli for each weapon type. A measure of the net effect of all the summation is given by the ratio of a modulated score to and unmodulated score (or for that matter, the ratio of a modulated CIP to an unmodulated CIP). The net CS/CSs moduli are stored in array XNMOD(), whose elements are symbolized in Figure I-9 by FFF and PPP for the baseline and target divisions, respectively. Notice that the net moduli depend on results of both the CS/CSS Module and the Combat Module. A net modulus for the baseline division, for example, is then simply

FFF = CCC / BBB, or

(net modulus) = (modulated score) / (unmodulated score)

which, as already noted, can also be determined as

(net modulus) = (modulated CIP) / (unmodulated CIP)

The idea here is to estimate the net moduli for the baseline and target divisions and, in the following steps, to estimate net moduli for the intermediate division, and then apply those net moduli as multipliers of the already derived unmodulated CIPs to generate modulated CIPs...

(modulated CIP) = (net modulus) * (unmodulated CIP)

but this is jumping all the way to step 9 below. Before that...

(7) Step 7 in Figure I-9 corresponds to the call of subprogram DOPHI in order to rollup the previously determined sample CS/CSS moduli over the 12 Red target categories. The results of summation of elements of array FAX() are stored in array PHI() for the baseline, target, and intermediate divisions. Notice that the elements of array PHI() do not depend on results of the Combat Module.

(8) Step 8 in Figure I-9 corresponds to the call of subprogram NXMOD in order to estimate and store the net CS/CSS moduli of the intermediate division. Results are stored in array XNMOD() with elements corresponding to the intermediate division symbolized by III in Figure I-9. In terms of the foregoing notation and symbols,

III - PHI(J,3) * ((1-T) * FFF / PHI(J,1) + T * PPP / PHI(J,2))

Note that PHI() for all the divisions were determined without reference to the Combat Module. FFF and PPP were determined for the baseline and target divisions by reference to both the CS/CSS and Combat Modules. It is assumed that the missing reference to the Combat Module in the case of the intermediate division can be "filled in" as being in proportion to corresponding known ratios FFF/PHI(J,1) and PPP/PHI(J,2). Here, an "inventory phase parameter" T is introduced. As applied in the above expression, T is simply a convexity parameter on the interval 0.00 - 1.0. It is intended to reflect how far in "inventory time" the intermediate division has progressed from baseline to target inventory. Currently the Interpolation Module applies the same value of T to all weapon types. Certainly the definition of T can be generalized to depend on weapon type in general and inventory in particular; i.e., it may be made a function of AAA, GGG, and KKK, if necessary, to make interpolation more accurate.

(9) Step 9 in Figure I-9 corresponds to the call of subprogram XMCIP in order to estimate and store final modulated CIPs for the intermediate division. The results, of course, depend on the foregoing steps. The modulated CIPs are stored in array VCIP(), whose elements are symbolized by JJJ in Figure I-9. The computation is implied to multiply the previously estimated unmodulated CIPs by the just determined net CS/CSS moduli.

JJJ = III * HHH

(modulated CIP) = (net modulus) * (unmodulated CIP)

At the close of step 9, both the unmodulated and modulated CIPs of the intermediate division have been estimated. It remains to determine the unmodulated and modulated scores as well as unmodulated and modulated COPs. These steps are not shown in Figure I-9. The steps are performed within OUTX, the output subprogram. The scores are generated by multiplying CIPs by the corresponding weapon counts from the previously stored inventory data.

(score) = (quantity) * (CIP)

The COPs are generated by summation of the scores.

I-17. SOURCE LISTINGS. The source listings for the main and subprograms of the Interpolation Module appear in Figures I-10 through I-34. The source listings include some intralinear comments. The following paragraphs provide additional commentary.

I-18. MAIN PROGRAM DOFAX. Figure I-10 presents the source listing of the program element DOFAX, the main program of the Interpolation Module.

```
C     7 OCT 83 -- G. E. C.
C     PROGRAM TO INTERPOLATE CIPS, UNMOD & MOD, FOR AN INVENTORY
C     'BETWEEN' A BASELINE (BASE) AND AN END DATE (TARGET) INVENTORY
C       BASE--->INTERMEDIATE--->TARGET
C
C
        PARAMETER M=60,N=60,NFUNS=9,NENV=16,NCATS=12
C
        COMMON/FAX3/FAX(12,60,3),EWT(16)
        COMMON/FORCES/QTY(60,3)
        COMMON/RCLLBT/RSCORE(5,60,2),UCIP(5,60,2),XSCORE(5,60,2),
       1 XCIP(5,60,2)
        COMMON/NEWVAL/XUCIP(5,60),XMOD(5,60,3),PHI(60,3),VCIP(5,60)
        COMMON/FACTOR/U(9,16,3),V(9,16,3),S(9,16,3),T(9,16,3),
       1 INDX(16,3)
        COMMON/AWPK/IWRK,ISCNT,ITHTRX,ITPDX,IVISXX,IPCSXX,IDAYXX
C
        CHARACTER*1 BW(M),RW(N),UFUN(NFUNS,M),VFUN(NFUNS,M),
       1 SFUN(NFUNS,N),TFUN(NFUNS,N),EUFUN(NFUNS,NENV),
       2 EVFUN(NFUNS,NENV),ESFUN(NFUNS,NENV),ETFUN(NFUNS,NENV),
       3 UCAT(NFUNS,NCATS,NCATS),VCAT(NFUNS,NCATS,NCATS),
       4 SCAT(NFUNS,NCATS,NCATS),TCAT(NFUNS,NCATS,NCATS)
        CHARACTER*3 ITHTRX
C
        DIMENSION A(NFUNS,NENV),LBCAT(M),LRCAT(N)
        LOGICAL VARY
        DATA NPOS,NDAY/4,2/
C
C            ARRAYS
C            ------
C
C     FAX(I,J,K)  BLUE CSCSS MODULI COMPUTED IN USUAL WAY FOR:
C      I=1,12   EVERY 5TH RED WPN TYPE
C      J=1,60   EACH BLUE WPN TYPE
C      K=1,3    BASE, TARGET, & INTER FORCES
C     EWT(I)         ENVIRONMENT WEIGHTS ROLLED UP BY POSTURE
C      I=1,4      POSTURE: D.I., D.L., DELAY, ATK
C     QTY(I,J)      WEAPON INVENTORIES
C      I=1,60 WEAPON TYPE
C      J=1,3  BASE, TARGET, & INTER FORCES
C     RSCORE(I,J,K)   RAW SCORES
C     UCIP(I,J,K)    UNMODULATED CIPS
C     XSCORE(I,J,K)  MODULATED SCORES
C     XCIP(I,J,K)    MODULATED CIPS
C
C
C
C     WVAL(I,J,K,L)    EQUIVALENT TO ABOVE FOUR ARRAYS
C      I=1,5    PERS, LVEH, HVEH, ACFT, SCLR
C      J=1,60   WEAPON TYPE
```

Figure I-10. Source Listing of the Main Program of the AFP Interpolation Module (page 1 of 4 pages)

```
C      K=1,2   BASE, TARGET, FORCE
C      L=1,4   RSCORE, UCIP, SCORE, XCIP
C     XUCIP(I,J)  INTERPOLATED UNMODULATED CIPS FOR INTER FORCE
C      I=1,5   PERS, LVEH, HVEH, ACFT, SCLR
C      J=1,60 WEAPON TYPE
C     XNMOD(I,J,K) NET MODULI
C       I=1,5 PERS, LVEH, HVEH, ACFT, SCLR
C       J=1,60 WEAPON TYPE
C       K=1,3 BASE, TARGET, INTER-FORCE
C      PHI(I,J)   SYNTHETIC MODULI
C       I=1,60 WEAPON TYPE
C       J=1,3 BASE, TARGET, & INTER FORCES
C      VCIPS(5,60)  INTERPOLATED MODULATE CIPS FOR INTER-FORCE
C       I=1,5  PERS, LVEH, HVEH, ACFT, SCLR
C       J=1,60 WEAPON TYPE
C
C
C       READ THE PHASE PARAMETER & WHETHER NEWCIPS SHOULD DEPEND
C         ON TX OR BE CONSTANT BETWEEN BASE AND TARGET YEARS
        READ(5,1) TX,VARY
    1   FORMAT()
        WRITE(6,2) TX,VARY
     2 FORMAT(/' TIME PHASE PARAMETER=',F4.2,' NEWCIP VARY=',L10)
        CALL NONDIV
        INFILE=17
        CALL GETCAT(LBCAT,M,'BBCAT-FILE',INFILE)
        INFILE=18
        CALL GETCAT(LRCAT,N,'RRCAT-FILE',INFILE)
C
        CALL GETDAT(M,N,NFUNS,NENV,UFUN,VFUN,SFUN,TFUN,
       1 EUFUN,EVFUN,ESFUN,ETFUN,BW,RW,A)
C
C      GET THE ORIGINAL CS/CSS FACTORS FOR EACH INVENTORY
C
        DO 10 I=1,3
        CALL USENAM
        CALL GETFAC(U(1,1,I),V(1,1,I),S(1,1,I),T(1,1,I),NFUNS,
       1 INDX(1,I))
    10 CLOSE(29)
C
        CALL GTCATV(UCAT,VCAT,SCAT,TCAT,NFUNS,NCAT)
C
        CALL ENVWTS
C
        IWRK= 25
C
C     GET BASE, INTERMEDIATE, & TARGET INVENTORIES
C
        CALL GETINV
        WRITE(6,101)
```

Figure I-10. Source Listing of the Main Program of the AFP Interpolation Module (page 2 of 4 pages)

```
  101 FORMAT(' GETINV COMPLETE')
C
C  GET BASE, & TARGET ROLLED UP BLUE SCORES & CIPS
C
      CALL GETROL
      WRITE(6,102)
  102 FORMAT(' GETROL COMPLETE')
C
C        CORRECT CIPS FOR EARLIER ROUNDOFFS.
C
      CALL RECIP
      WRITE(6,103)
  103 FORMAT(' RECIP COMPLETE')
C
C        GENERATE MEAN MODULI
C
      DO 60 IFOR=1,3
      DO 50 IVIS=1,2
      IVISX=IVIS
      DO 40 IPOS=1,4
      IPOSX=IPOS
      JPOS=4
      IF(IPOSX.EQ.4) JPOS=1
      DO 30 IDAY=1,2
      IDAYX=IDAY
      CALL CSCSS(M,N,NFUNS,IVIS,IPOS,IDAY,NENV,UFUN,VFUN,SFUN,
     1 TFUN, EUFUN,EVFUN,ESFUN,ETFUN,BW,RW,NPOS,NDAY,A,JPOS,
     2 LBCAT,LRCAT,UCAT,VCAT,SCAT,TCAT,NCATS,IFOR)
      WRITE(6,104) IFOR,IVIS,IPOS,IDAY
  104 FORMAT(' COMPLETED CSCSS FOR: IFOR=',I1,' IVIS=',I1,
     1 ' IPOS=',I1,' IDAY=',I1)
   30 CONTINUE
   40 CONTINUE
   50 CONTINUE
   60 CONTINUE
C
C        COMPUTE UNMODULATED CIPS FOR INTER-FORCE
C
      CALL NEWCIP(TX,VARY)
      WRITE(6,105)
  105 FORMAT(' NEWCIP COMPLETE')
C
C        COMPUTE 'NET MODULI' FOR BASE AND TARGET FORCES
C
      CALL NETMOD
      WRITE(6,106)
  106 FORMAT(' NETMOD COMPLETE')
C
C        COMPUTE 'SYNTHETIC MODULI' FOR ALL THREE FORCES
C
```

Figure I-10. Source Listing of the Main Program of the AFP Interpolation Module (page 3 of 4 pages)

```
      CALL DOPHI
      WRITE(6,107)
  107 FORMAT(' DOPHI COMPLETE')
C
C         ESTIMATE THE 'NET MODULI' FOR INTER-FORCE
C           XNMOD(IV,IW,3)
C
      CALL NXMOD(TX)
      WRITE(6,108)
  108 FORMAT(' NXMOD COMPLETE')
C
C         MODULATE THE INTERFORCE UCIPS
C           VCIP(IV,IW)=XNMOD(IV,IW,3)*XUCIP(IV,IW)
C
      CALL XMCIP
      WRITE(6,109)
  109 FORMAT(' XMCIP COMPLETE')
C
C         GENERATE INTER-FORCE TARTY-LIKE REPORT
C
      CALL OUTX
      WRITE(6,110)
  110 FORMAT(' OUTX COMPLETE')
C
      CALL DMPMOD
      WRITE(6,111)
  111 FORMAT(' DMPMOD COMPLETE')
C
      STOP 'DONE'
      END
```

Figure I-10. Source Listing of the Main Program of the AFP Interpolation Module (page 4 of 4 pages)

a. Lines 7-25 declare the reference, working, and scratch arrays of the main program.

b. Line 70 obtains the inventory phase parameter and the value of VARY from the runstream. Line 74 calls subprogram NONDIV to read the indices of Blue nondivisional weapons for exclusion from COP computations.

c. Lines 75-91 obtain data that support the CS/CSS moduli computation as described in Appendix E on the CS/CSS process. There is one exception here in the Interpolation Module. Lines 85-89 obtain three sets of CS/CSS factors, one each for the baseline, target, and intermediate divisions. In a single execution of the CS/CSS Module described in Appendix E, only one set of CS/CSS factors is input.

d. Line 93 calls ENVWTS to obtain a set of 16 combat environmental weights. It is assumed that the same set of 16 weights is applicable to all three divisions.

e. Line 99 calls GETINV to obtain the three inventories corresponding to the baseline, target, and intermediate divisions.

f. Line 105 calls GETROL to obtain the previously generated final combat potentials of the baseline and target divisions.

g. Line 111 calls RECIP to determine more precise values of CIPs than read by GETROL. GETROL reads CIPs as formatted in the final combat potentials files. More precise CIPs may be estimated by dividing the weapon scores by the weapon quantities.

h. Lines 117 and 135 define the limits of a quadruply-nested loop structure for the determination of sample CS/CSS moduli. The outer loop is over the baseline, target, and intermediate divisions. This looping over three different divisions differs from the practice of the CS/CSS Module itself where only one of these would be examined in a single execution. Another diference within the Interpolation Module is attention only to the 12 Red weapon (target) categories instead of to the full 60 weapon types as in the CS/CSS Module. The loops over visibility and day/night are degenerate. It is only the combat posture loop that is nontrivial among combat environmental indices.

i. Line 139 calls NEWCIP to compute the unmodulated CIPs of the intermediate division in one of two ways depending on the value of VARY.

j. Line 145 calls NETMOD to compute the net CS/CSS moduli for the baseline and target divisions.

k. Line 151 calls DOPHI to sum the CS/CSS Module over the Red weapon (target) categories, thereby producing the so-called "synthetic moduli." Synthetic moduli are generated for baseline, target, and intermediate divisions.

l. Line 158 calls NXMOD to estimate the net CS/CSS for the intermediate division from the net moduli of the baseline and target divisions, from the synthetic moduli of all three divisions, and from the inventory phase parameter.

m. Line 165 calls XMCIP to estimate the modulated CIPs for the intermediate division as products of the net moduli and the unmodulated CIPs.

n. Line 171 calls OUTX to complete the computation of final combat potentials for the intermediate division and to output the complete results. Both unmodulated and modulated CIPs are already known upon entry to OUTX. OUTX multiplies the CIPs by inventory quantities to determine the unmodulated and modulated scores. OUTX sums the scores in order to determine the COPs. OUTX writes Blue combat potentials of the intermediate division to unit 30.

o. Line 175 calls DMPMOD to dump the array of net CS/CSS Module for checking purposes.

p. Line 179 terminates normal execution of the Interpolation Module.

I-19. SUBPROGRAM CSCSS. Figure I-11 presents the source listing of subprogram CSCSS of the Interpolation Module. This subroutine is substantially the same as the one employed in the CS/CSS Module and as described in Appendix E. The main differences peculiar to the version here are:

a. Line 14 declares arrays FAX() and ENT(). FAX() stores moduli for each of the 12 Red weapon (target) categories. The Interpolation Module saves sample moduli in FAX() rather than outputting them to a file as is the case in the CS/CSS Module. EWT() provides the environmental weights by combat posture. The Interpolation Module, within a single execution, rolls up moduli by posture.

b. Within a single Interpolation Module execution, CSCSS is required to generate moduli for three divisions.

c. Line 66 limits the Red weapon loop to the first weapon in each of the usual 12 target categories.

d. Red moduli are not computed or saved.

e. Line 101 applies a combat posture weight to the most recently computed modulus and updates the corresponding element of array FAX().

```
      SUBROUTINE CSCSS(M,N,NFUNS,IVIS,IPOS,IDAY,NENV,
     *UFUN,VFUN,SFUN,TFUN,EUFUN,EVFUN,ESFUN,ETFUN,
     *BW,RW,NPOS,NDAY,A,JPOS,LBCAT,LRCAT,
     *UCAT,VCAT,SCAT,TCAT,NC,IFOR)
C
      PARAMETER NBSTEP=1, NRSTEP=1
C
      CHARACTER*1 BW(M),RW(N),UFUN(NFUNS,M),VFUN(NFUNS,M),
     *SFUN(NFUNS,N),TFUN(NFUNS,N),EUFUN(NFUNS,NENV),EVFUN(NFUNS,NENV),
     *ESFUN(NFUNS,NENV),ETFUN(NFUNS,NENV),NO,
     *UCAT(NFUNS,NC,NC),VCAT(NFUNS,NC,NC),SCAT(NFUNS,NC,NC),
     *TCAT(NFUNS,NC,NC)
C
      COMMON/FAX3/FAX(12,60,3),EWT(16)
      DIMENSION A(NFUNS,NENV),LBCAT(M),LRCAT(N)
C
      NO='N'
C
C     A U,V,S, OR T ELEMENT IS BUILT AS THE DIAGONAL OF A HYPER-
C     PARALLELOPIPED FOR THE L-TH ENVIRONMENT:
C        ELE = SQRT(SUM(A(K,L)*F(K)^2;K)/SUM(A(K,L);K))
C
C     IN CURRENT FORM, DOES NOT DEPEND ON FORCE MASSES OR RATIO!!!
C
C     ARRAYS & THINGS:
C        A(K,L)            WT OF K-TH FUNCT IN L-TH ENVIRONMENT.
C        BF                VLUE NET FACTOR OVER ALL FUNCTS. L ENVIR.
C        RF                RED NET FACTOR OVER ALL FUNCTS. L ENVIR.
C        BW(I)             SW, WHETHER BLUE TYPE I AFFECTED.
C        RW(J)             SW, WHETHER RED TYPE J AFFECTED.
C        UFUN(K,I)         SW, WHETHER K FUNCT AFFECTS B-TYPE I U'S.
C        VFUN(K,I)         SW, WHETHER K FUNCT AFFECTS B-TYPE I V'S.
C        SFUN(K,J)         SW, WHETHER K FUNCT AFFECTS R-TYPE J S'S.
C        TFUN(K,J)         SW, WHETHER K FUNCT AFFECTS R-TYPE J T'S.
C        EUFUN(K,L)        SW, WHETHER K FUNCT & L ENVIR AFFECT U'S.
C        EVFUN(K,L)        SW, WHETHER K FUNCT & L ENVIR AFFECT V'S.
C        ESFUN(K,L)        SW, WHETHER K FUNCT & L ENVIR AFFECT S'S.
C        ETFUN(K,L)        SW, WHETHER K FUNCT & L ENVIR AFFECT T'S.
C        UCAT(K,J,I)       SW, WHETHER K FUNCTION, R-TYPE J, AND B-TYPE I
C                                 AFFECT U'S.
C        VCAT(K,J,I)       SW, WHETHER K FUNCTION, R-TYPE J, AND B-TYPE I
C                                 AFFECT V'S.
C        SCAT(K,J,I)       SW, WHETHER K FUNCTION, R-TYPE J, AND B-TYPE I
C                                 AFFECT S'S.
C        TCAT(K,J,I)       SW, WHETHER K FUNCTION, R-TYPE J, AND B-TYPE I
C                                 AFFECT T'S.
C
C     COMPUTE INDEX OF ENVIRONMENT
C
      LENV=(IVIS-1)*NPOS*NDAY+(IDAY-1)*NPOS+IPOS
```

Figure I-11. Source Listing of the Subprogram CSCSS of the AFP Interpolation Module (page 1 of 2 pages)

```
      LENVJ=(IVIS-1)*NPOS*NDAY+(IDAY-1)*NPOS+JPOS
C
C     COMPUTE NORM
C
      AC=0.0
      AD=0.0
      DO 50 JF=1,NFUNS
      AC=AC+A(JF,LENV)
      AD=AD+A(JF,LENVJ)
   50 CONTINUE
C
      DO 1000 IB=1,M
      IF (BW(IB).EQ.NO) GO TO 1000
      IBCAT=LBCAT(IB)
      IRC=0
      DO 900 IR=1,N,5
      BF=1.0
      IRC=IRC+1
      RF=1.0
      IF (RW(IR).EQ.NO) GO TO 900
      IRCAT=LRCAT(IR)
      RR=0.0
      BB=0.0
      DO 800 JF=1,NFUNS
      U=1.0
      IF(UFUN(JF,IB).EQ.NO) GO TO 710
      IF(EUFUN(JF,LENV).EQ.NO) GO TO 710
      IF(UCAT(JF,IRCAT,IBCAT).EQ.NO) GO TO 710
      U=FU(JF,IFOR,LENV)
  710 V=1.0
      IF(VFUN(JF,IB).EQ.NO) GO TO 720
      IF(EVFUN(JF,LENV).EQ.NO) GO TO 720
      IF(VCAT(JF,IRCAT,IBCAT).EQ.NO) GO TO 720
      V=FV(JF,IFOR,LENV)
  720 S=1.0
      IF(SFUN(JF,IR).EQ.NO) GO TO 730
      IF(ESFUN(JF,LENV).EQ.NO) GO TO 730
      IF(SCAT(JF,IRCAT,IBCAT).EQ.NO) GO TO 730
      S=FS(JF,IFOR,LENV)
  730 T=1.0
      IF(TFUN(JF,IR).EQ.NO) GO TO 740
      IF(ETFUN(JF,LENV).EQ.NO) GO TO 740
      IF(TCAT(JF,IRCAT,IBCAT).EQ.NO) GO TO 740
      T=FT(JF,IFOR,LENV)
  740 CONTINUE
      UVST=(U*V)/(S*T)
      UVST=UVST*UVST
      BB=BB+A(JF,LENV)*UVST
  800 CONTINUE
      BF=SQRT(BB/AC)
      FAX(IRC,IB,IFOR)=FAX(IRC,IB,IFOR)+EWT(LENV)*BF
  900 CONTINUE
 1000 CONTINUE
      RETURN
      END
```

Figure I-11. Source Listing of the Subprogram CSCSS of the AFP Interpolation Module (page 2 of 2 pages)

I-20. SUBPROGRAM DOPHI. Figure I-12 presents the source listing of subprogram DOPHI of the Interpolation Module. DOPHI simply arithmetically averages the sample CS/CSS moduli over Red weapon (target) categories in array FAX() and stores the means in array PHI(). The function is performed for each division (lines 10-18 loop), each Blue weapon type (lines 11-17 loop), and each target category (lines 13-15 loop).

```
 1            SUBROUTINE DOPHI
 2            COMMON/FORCES/QTY(60,3)
 3            COMMON/POLLBT/RSCORE(5,60,2),UCIP(5,60,2),XSCORE(5,60,2),
 4           1XCIP(5,60,2)
 5            COMMON/NEWVAL/XUCIP(5,60),XNMOD(5,60,3),
 6           1PHI(60,3),VCIP(5,60)
 7            COMMON/FAX3/FAX(12,60,3),EWT(16)
 8            Z=1.0/12.0
 9            CALL ZERO(PHI,180)
10            DO 300 IFOR=1,3
11             DO 200 IB=1,60
12            P=0.0
13            DO 100 IC=1,12
14            P=P+FAX(IC,IB,IFOR)
15      100   CONTINUE
16            PHI(IB,IFOR)=P*Z
17      200   CONTINUE
18      300   CONTINUE
19            RETURN
20            END
```

Figure I-12. Source Listing of the Subprogram DOPHI of the AFP Interpolation Module

I-21. SUBPROGRAM ENVWTS. Figure I-13 presents the source listing of subprogram ENVWTS of the Interpolation Module. ENVWTS reads a complete set of 16 combat environment weights (line 3).

```
1            SUBROUTINE ENVWTS
2            COMMON/FAX3/FAX(2160),EWT(16)
3            READ(5,1) (EWT(I),I=1,16)
4      1     FORMAT()
5            RETURN
6            END
```

Figure I-13. Source Listing of Subprogram ENVWTS of the AFP Interpolation Module

I-22. FUNCTION SUBPROGRAMS FS, FT, FU, AND FV. Figures I-14, I-15, I-16, and I-17 present the source listings of subprograms FS, FT, FU, and FV of the Interpolation Module. These functions return CS/CSS factors corresponding to Red measure, Red countermeasure, Blue measure, and Blue countermeasure, respectively. The first argument, JZ, is simply the index of the CS/CSS function of current interest. The functions are generalizations of those employed in the CS/CSS Module. The generalizations add the formal arguments IFOR and IE within the function calls. The argument IFOR is necessary because the Interpolation Module must discriminate among baseline, target, and intermediate divisions. The argument IE is necessary because the Interpolation Module must match each combat environment with an environmentally correct set of CS/CSS factors by means of the pointer references in lines 3. The reference arrays U(), V(), S(), and T() are three-dimensional in the Interpolation Module but only one-dimensional in the CS/CSS Module.

```
1          FUNCTION FS(JZ,IFOR,IE)
2          COMMON/FACTOR/U(9,16,3),V(9,16,3),S(9,16,3),T(9,16,3),INDX(16,3)
3          IEX=INDX(IE,IFOR)
4          FS=S(JZ,IEX,IFOR)
5          RETURN
6          END
```

Figure I-14. Source Listing of Subprogram FS of the AFP Interpolation Module

```
1          FUNCTION FT(JZ,IFOR,IE)
2          COMMON/FACTOR/U(9,16,3),V(9,16,3),S(9,16,3),T(9,16,3),INDX(16,3)
3          IEX=INDX(IE,IFOR)
4          FT=T(JZ,IEX,IFOR)
5          RETURN
6          END
```

Figure I-15. Source Listing of Subprogram FT of the AFP Interpolation Module

```
1          FUNCTION FU(JZ,IFOR,IE)
2          COMMON/FACTOR/U(9,16,3),V(9,16,3),S(9,16,3),T(9,16,3),INDX(16,3)
3          IEX=INDX(IE,IFOR)
4          FU=U(JZ,IEX,IFOR)
5          RETURN
6          END
```

Figure I-16. Source Listing of Subprogram FU of the AFP Interpolation Module

```
      FUNCTION FV(JZ,IFOR,IE)
      COMMON/FACTOR/U(9,16,3),V(9,16,3),S(9,16,3),T(9,16,3),INDX(16,3)
      IEX=INDX(IE,IFOR)
      FV=V(JZ,IEX,IFOR)
      RETURN
      END
```

Figure I-17. Source Listing of Subprogram FV of the AFP Interpolation Module

I-23. SUBPROGRAM GETCAT. Figure I-18 presents the source listing of subprogram GETCAT of the Interpolation Module. The subroutine is taken from the CS/CSS Module. GETCAT reads and stores an array of indices indicating the weapon categories to which weapon types belong.

```
      SUBROUTINE GETCAT(LCAT,N,FNAME,INFILE)
      CHARACTER RDERR*80,FNAME*10
      DIMENSION LCAT(N)
      READ(INFILE,100,ERR=110) (LCAT(I),I=1,N)
  100 FORMAT(3X,10I3)
      GO TO 120
  110 READ(0,115) RDERR
  115 FORMAT(A80)
      PRINT*, 'ERR IN READING ', FNAME, '  RECORD= ', RDERR
      STOP
  120 PRINT*, 'AT END - ', FNAME
      RETURN
      END
```

Figure I-18. Source Listing of Subprogram GETCAT of the AFP Interpolation Module

a. Argument LCAT is the address of the array in which category indices are to be stored.

b. Argument N is the length of array LCAT.

c. Argument FNAME is the name of the source file.

d. Argument INFILE is the number of the unit on which the source file is located.

I-24. SUBPROGRAM GETDAT. Figure I-19 presents the source listing of subprogram GETDAT of the Interpolation Module. The subroutine is taken from the CS/CSS Module. GETDAT reads and stores four kinds of data needed in the CS/CSS moduli generation process. GETDAT is described in Appendix E.

```
      SUBROUTINE GETDAT(M,N,NFUNS,NENV,UFUN,VFUN,
     *SFUN,TFUN,EUFUN,EVFUN,ESFUN,ETFUN,BW,RW,A)
C
      CHARACTER RDERR*80,FNAME*10
C
      CHARACTER*1 BW(M), RW(N), UFUN(NFUNS,M), VFUN(NFUNS,M),
     *SFUN(NFUNS,NENV), TFUN(NFUNS,NENV), EUFUN(NFUNS,NENV),
     *EVFUN(NFUNS,NENV), ESFUN(NFUNS,NENV), ETFUN(NFUNS,NENV)
C
      DIMENSION A(NFUNS,NENV)
C
      FNAME='ADAT1 FILE'
      READ(11,100,ERR=310) (BW(I),I=1,M)
      READ(11,100,ERR=310) (RW(I),I=1,N)
  100 FORMAT(5X,10A1)
      PRINT*, 'AT END - ', FNAME
C
      FNAME='ADAT2 FILE'
      DO 150 I=1,M
      READ(12,200,ERR=310) (UFUN(K,I),VFUN(K,I),SFUN(K,I),
     *TFUN(K,I),K=1,NFUNS)
  150 CONTINUE
  200 FORMAT(5X,9(4A1,1X))
      PRINT*, 'AT END - ', FNAME
C
      FNAME='ADAT3 FILE'
      DO 250 L=1,NENV
      READ(13,200,ERR=310) (EUFUN(K,L),EVFUN(K,L),ESFUN(K,L),
     *ETFUN(K,L),K=1,NFUNS)
  250 CONTINUE
      PRINT*, 'AT END - ', FNAME
C
      FNAME='ADAT4 FILE'
      READ(14,300,ERR=310) ((A(K,L),K=1,NFUNS),L=1,NENV)
  300 FORMAT(5X,9F8.4)
      GO TO 320
  310 READ(0,315) RDERR
  315 FORMAT(A80)
      PRINT*, 'ERR IN READING ', FNAME, '  RECORD= ', RDERR
      STOP
  320 PRINT*, 'AT END - ', FNAME
      RETURN
      END
```

Figure I-19. Source Listing of Subroutine GETDAT of the AFP Interpolation Module

I-25. SUBPROGRAM GETFAC. Figure I-20 presents the source listing of subprogram GETFAC of the Interpolation Module. The subroutine is generalized from the CS/CSS Module. The original GETFAC is described in Appendix E. It is used differently in the Interpolation Module. The two-dimensional arrays U(), V(), S(), and T() "seen" by GETFAC within the Interpolation Module are only subarrays within larger, three-dimensional arrays. The added dimensions corresponds to the 16 combat environments and to the baseline, target, and intermediate divisions. GETFAC is called separately for each of these divisions. On each call the arguments U, V, S, and T are given different addresses corresponding to the different divisions.

```
      SUBROUTINE GETFAC(U,V,S,T,NFUNS,INDX)
      CHARACTER*80 RDERR
      DIMENSION U(NFUNS,16),V(NFUNS,16),S(NFUNS,16),
     1 T(NFUNS,16),INDX(16)
C
      READ(5,2) (INDX(I),I=1,16)
    2 FORMAT()
   90 READ(29,100,END=120,ERR=110) (IE,U(I,IE),V(I,IE),S(I,IE),
     1 T(I,IE),I=1,NFUNS)
  100 FORMAT(3X,I2,4F8.4)
      GO TO 90
  110 READ(0,115) RDERR
  115 FORMAT(A80)
      PRINT*, 'ERR IN READING ADATS FILE', '  RECORD= ', RDERR
      STOP
  120 PRINT*, 'AT END - ADATS FILE'
      RETURN
      END
```

Figure I-20. Source Listing of Subroutine GETFAC of the AFP Interpolation Module

I-26. SUBPROGRAM GETINV. Figure I-21 presents the source listing of subprogram GETINV of the Interpolation Module. GETINV reads and stores the inventories of the baseline, target, and intermediate divisions from files in the format accepted by the Combat Module. The three files are attached to unit 29 in succession dynamically within GETINV.

a. Lines 8 and 25 define the limits of the loop over baseline, target, and intermediate divisions.

b. Line 9 reads the name of the file corresponding to the division of interest.

c. Line 11 concatenates the file name within a string @USE statement.

d. Line 12 calls system routine FACSF to execute the @USE statement.

e. Lines 14 through 16 read the first 13 records simply to bypass them because they do not contain inventory information.

f. Lines 19-23 read 60 records containing the inventory quantities for the Blue weapon types. The desired quantity is the first entry of a record.

```
      SUBROUTINE GETINV
      COMMON/FORCES/QTY(60,3),IFOR
      CHARACTER*50 INSTR
      CHARACTER*30 FNAME
C  LOGICAL SOURCE OF INVENTORY FILES
      ISOR=29
C  LOOP OVER BASE, TARGET, AND INTERMEDIATE INVENTORIES
      DO 1000 IFOR=1,3
      READ(5,1) FNAME
    1 FORMAT(A30)
      INSTR='@USE 29,'//FNAME
      CALL FACSF(INSTR)
      JFOR=IFOR
      DO 100 I=1,13
      READ(ISOR,2)
  100 CONTINUE
    2 FORMAT(1X)
C
      DO 900 IB=1,60
      IBX=IB
      READ(ISOR,3,END=2000,ERR=3000)QTY(IB,IFOR)
    3 FORMAT()
  900 CONTINUE
      CLOSE(ISOR)
 1000 CONTINUE
      RETURN
C
 2000 WRITE(6,4) IBX,FNAME
    4 FORMAT(' PREMATURE END AT WPN ',I2,' IN FILE ',A30)
      STOP 'INV END'
C
 3000 WRITE(6,5) IBX,FNAME
    5 FORMAT(' ERR AT WPN ',I2,' IN FILE ',A30)
      STOP 'INV ERR'
C
      END
```

Figure I-21. Source Listing of Subprogram GETINV of the AFP Interpolation Module

I-27. SUBPROGRAM GETROL. Figure I-22 presents the source listing of subprogram GETROL of the Interpolation Module. GETROL reads and stores the combat potentials of the baseline and target divisions. Combat potentials are stored in array WVAL(5,60,2,4), which is the superarray of arrays RSCORE(5,60,2), UCIP(5,60,2), XSCORE(5,60,2), and XCIP(5,60,2). The latter store unmodulated scores, unmodulated CIPs, modulated scores, and modulated CIPs, respectively. The CIPs are later replaced by more precise values by subprogram RECIP.

```
      SUBROUTINE GETROL
C      RETRIEVE BASE AND TARGET FORCE ENVIRONMENTAL ROLLUPS
C    G. E. C.  6 OCT 83
      DIMENSION   X(5),ISCNTR(6,2),ISCNTT(12),JRECS(12)
      EQUIVALENCE (ISCNTR,ISCNTT)
      COMMON/AWRK/IWRK,IDUM,KTHTR,JTPD,JVIS,JPOS,JDAY
      COMMON/GLOBAL/IBFOR,IRFOR,JCASE
      COMMON/NAME/FNAME
      COMMON/ROLLBT/WVAL(5,60,2,4)
      CHARACTER*50 INSTR
      CHARACTER*20 FNAME;ONAME,DONE
      CHARACTER*3 KTHTR
      DATA ISCNTR/
     1 110,130,150,170,111,151,
     2 120,140,160,180,121,161/
      DATA JRECS/
     1 1,2,3,4,1,2,
     2 1,2,3,4,1,2/
C     READ OUTPUT RECORD IDENTIFIERS
      READ(5,4) KTHTR,JTPD,JVIS,JPOS,JDAY,IBFOR,IRFOR,JCASE
    4 FORMAT(A3,I4,3I3,2I6,I5)
C
      IPRNT=6
C     LOGICAL SOURCE OF ROLLUP FILES
      ISOR=29
      DONE='DONE'
C
      CALL ZERO(WVAL,2400)
    1 FORMAT()
      ISIDE=0
C
      GOTO 95
   90 CLOSE(ISOR)
   95 CALL GETFIL(FNAME)
      IF(FNAME.EQ.DONE) RETURN
      ISIDE=ISIDE+1

      ONAME=FNAME
      WRITE(6,3) FNAME
  3   FORMAT(1X,A20)
      INSTR='@USE 29,'//ONAME//'.'
      CALL FACSF(INSTR)
C
  100 CALL GETREC(ISCNT,IWPN,X,ISOR)
      DO 200 I=1,12
      IF(ISCNT.NE.ISCNTT(I)) GOTO 200
      IS=I
      GOTO 300
  200 CONTINUE
      WRITE(IPRNT,2) ISCNT
    2 FORMAT(/' UNRECOGNIZABLE RECORD TYPE=',I5/)
      IERR1=IERR1+1
      IF(IERR1.LT.10) GOTO 100
      STOP 'IERR1'
C
  300 IREC=JRECS(IS)
      IF(IS.GT.4) GOTO 90
C
C
C     HERE IF WEAPON RECORD READ
      DO 450 I=1,5
      WVAL(I,IWPN,ISIDE,IREC)=X(I)
  450 CONTINUE
      GOTO 100
C
C
```

Figure I-22. Source Listing of Subprogram GETROL of the AFP Interpolation Module

a. Lines 4-12 declare the necessary arrays and variables.

b. Lines 13-15 initialize ISCNTR() with the indices of the record types contained in the files read by GETROL. Lines 16-18 initialize JRECS() with the indices of the types of combat potentials corresponding to the record types read by GETROL.

c. Lines 20-21 input identifiers to be included in final combat potential output records.

d. Lines 23-26 initialize several variables.

e. Line 28 calls ZERO to initialize the storage array WVAL().

f. Line 34 calls GETFIL for the name of the file to be read.

g. Line 35 checks to see if no more files need be read.

h. Line 41 concatenates the file name within string @USE instruction.

i. Line 42 calls system routine FACSF to attach the file of interest to unit 29.

j. Line 44 calls GETREC to read one record from the file of interest.

k. Lines 45-49 identify the type of record.

l. Line 56 sets the index of the combat potential type from the identified record type.

m. Line 57 checks whether the record just recorded is beyond the first four types. If so, all information of interest has already been read from the file; so go back to line 33 to close the file.

n. Lines 61-63 store the five components of combat potential from the record just read in the appropriate five-vector within array WVAL().

o. Line 64 transfers control back to line 44 to read another record.

I-28. SUBPROGRAM GETREC. Figure I-23 presents the source listing of subprogram GETREC of the Interpolation Module. GETREC reads a record from a file containing final combat potentials. GETREC returns the index of the record type via argument ISCNT, the index of the weapon type via argument IWPN, and the five components of combat potential via the five-vector X().

```
      SUBROUTINE GETREC(ISCNT,IWPN,X,ISOR)
      DIMENSION X(5)
      COMMON/NAME/FNAME
      CHARACTER FNAME*20,KTHTRX*3
      READ(ISOR,1,END=100) ISCNT,KTHTRX,JTPDX,JVISX,JPOSX,JDAYX,
     1 IWPN,X(1),X(2),X(3),X(4),X(5),IBFORX,IRFORX,JCASEX
      IR=IR+1
      RETURN
    1 FORMAT(I5,A3,I4,3I3,I5,5F10.3,2I6,I5)
C
  100 WRITE(6,2) IR, FNAME
      STOP 'IERR2'
    2 FORMAT (' LAST RECORD READ = ', I6,2X,A20)
      END
C
```

Figure I-23. Source Listing of Subprogram GETREC of the AFP Interpolation Module

I-29. SUBPROGRAM GETFIL. Figure I-24 presents the source listing of subprogram GETFIL of the Interpolation Module. GETFIL reads a file name from the runstream and returns the name via the argument FNAME.

```
      SUBROUTINE GETFIL(FNAME)
      CHARACTER*20 FNAME
      READ(5,1)FNAME
      RETURN
    1 FORMAT(A20)
      END
```

Figure I-24. Source Listing of Subprogram GETFIL of the AFP Interpolation Module

I-30. SUBPROGRAM GTCATV. Figure I-25 presents the source listing of subprogram GTCATV of the Interpolation Module. GTCATV reads and stores Y/N values specifying the CS/CSS functions by Blue and Red weapon categories that may be included or excluded during generation of CS/CSS moduli. The subroutine is taken from the CS/CSS Module.

```
      SUBROUTINE GTCATV(UCAT,VCAT,SCAT,TCAT,NFUNS,NC)
C
      CHARACTER*1 UCAT(NFUNS,NC,NC),VCAT(NFUNS,NC,NC),
     *SCAT(NFUNS,NC,NC),TCAT(NFUNS,NC,NC)
C
      CHARACTER*80 RDERR
C
C
      DO 60 K=1,12
      DO 50 I=1,9
      READ(16,100,ERR=110) (UCAT(I,J,K),VCAT(I,J,K),
     *SCAT(I,J,K),TCAT(I,J,K),J=1,12)
   50 CONTINUE
   60 CONTINUE
  100 FORMAT(4X,12(1X,4A1))
      GO TO 120
  110 READ(0,115)RDERR
  115 FORMAT(A80)
      PRINT*, 'ERR IN READING ACATS FILE', '  RECORD= ', RDERR
      STOP
  120 PRINT*, 'AT END - ACATS FILE'
      RETURN
      END
```

Figure I-25. Source Listing of Subprogram GTCATV of the AFP Interpolation Module

I-31. SUBPROGRAM NETMOD. Figure I-26 presents the source listing of subprogram NETMOD of the Interpolation Module. NETMOD computes the ratios of modulated to unmodulated scores for baseline and intermediate divisions and stores the results in array XNMOD(). The ratios are the "net CS/CSS moduli" derived from the final combat potentials.

```
      SUBROUTINE NETMOD
      COMMON/FORCES/QTY(3,60)
      COMMON/ROLLPT/RSCORE(5,60,2),UCIP(5,60,2),XSCORE(5,60,2),
     1 XCIP(5,60,2)
      COMMON/NEWVAL/XUCIP(5,60),XNMOD(5,60,3),PHI(60,3),VCIP(5,60)
      DO 300 IFOR=1,2
      DO 200 IB=1,60
      DO 100 IV=1,5
      R=RSCORE(IV,IB,IFOR)
      IF(R.LE.0.0) GOTO 100
      XNMOD(IV,IB,IFOR)=XSCORE(IV,IB,IFOR)/R
  100 CONTINUE
  200 CONTINUE
  300 CONTINUE
      RETURN
      END
```

Figure I-26. Source Listing of Subprogram NETMOD of the AFP Interpolation Module

I-32. SUBPROGRAM NEWCIP. Figure I-27 presents the source listing of subprogram NEWCIP of the Interpolation Module. NEWCIP computes unmodulated CIPs for the intermediate division and stores the results in array XUCIP(). The CIPs are derived from the corresponding CIPs of the baseline and target divisions and their inventories. If VARY=.FALSE., a weapon has the same unmodulated combat potential in any intermediate division for given base and target divisions. VARY+FALSE. corresponds to original "standard" AFP practice. If VARY=.TRUE., a weapon's unmodulated combat potential may vary with the phase parameter, TX, if its potentials differ between base and target divisions.

```
      SUBROUTINE NEWCIP(TX,VARY)
      COMMON/FORCES/QTY(60,3)
      COMMON/ROLLBT/RSCORE(5,60,2),UCIP(5,60,2),XSCORE(5,60,2),
     1XCIP(5,60,2)
      COMMON/NEWVAL/XUCIP(5,60),XNMOD(5,60,3),
     1PHI(60,3),VCIP(5,60)
      LOGICAL VARY
      IF(VARY) GOTO 300
      DO 200 IB=1,60
      QH=QTY(IB,1)
      QJ=QTY(IB,2)
      Q=QH+QJ
      IF(Q.LE.0.0) GOTO 200
      QQ=1.0/Q
      DO 100 IV=1,5
      XUCIP(IV,IB)=(QH*UCIP(IV,IB,1)+QJ*UCIP(IV,IB,2))*QQ
  100 CONTINUE
  200 CONTINUE
      RETURN
  300 TX1=1.0-TX
      DO 500 IB=1,60
      QH=QTY(IB,1)
      QJ=QTY(IB,2)
      Q=QH+QJ
      IF(Q.LE.0.0) GOTO 500
      QQB=TX1*QH
      QQT=TX*QJ
      QQ=1.0/(QQB+QQT)
      DO 400 IV=1,5
      XUCIP(IV,IB)=(QQB*UCIP(IV,IB,1)+
     1   QQT*UCIP(IV,IB,2))*QQ
  400 CONTINUE
  500 CONTINUE
      RETURN
      END
```

Figure I-27. Source Listing of Subprogram NEWCIP of the AFP Interpolation Module

I-33. SUBPROGRAM NXMOD. Figure I-28 presents the source listing of subprogram NXMOD of the Interpolation Module. NXMOD computes and stores "net CS/CSS moduli" for the intermediate division. The net moduli for the intermediate division are computed from the net moduli of the baseline and target divisions and from the synthetic moduli of all three divisions.

```
1           SUBROUTINE NXMOD(T)
2           COMMON/FORCES/QTY(60,3)
3           COMMON/NEWVAL/XUCIP(5,60),XNMOD(5,60,3),PHI(60,3),VCIP(5,60)
4     C
5           DO 1000 IW=1,60
6           Q1=QTY(IW,1)
7           Q2=QTY(IW,2)
8           IF((Q1.EQ.0.0).AND.(Q2.EQ.0.0)) GOTO 1000
9           P1=PHI(IW,1)
10          P2=PHI(IW,2)
11          P3=PHI(IW,3)
12          IF((Q1.GT.0.0).AND.(Q2.GT.0.0)) THEN
13          DO 100 IV=1,5
14          XNMOD(IV,IW,3)=P3*((1.0-T)*XNMOD(IV,IW,1)/P1 +
15         1   T*XNMOD(IV,IW,2)/P2)
16      100 CONTINUE
17          ELSE
18           IF(Q1.GT.0.0) THEN
19            DO 200 IV=1,5
20            XNMOD(IV,IW,3)=P3*XNMOD(IV,IW,1)/P1
21      200   CONTINUE
22           ELSE
23            DO 300 IV=1,5
24            XNMOD(IV,IW,3)=P3*XNMOD(IV,IW,2)/P2
25      300   CONTINUE
26           ENDIF
27          ENDIF
28    C
29     1000 CONTINUE
30    C
31          RETURN
32    C
33          END
```

Figure I-28. Source Listing of Subprogram NXMOD of the AFP Interpolation Module

a. Lines 5 and 29 define the bounds of a loop over the Blue weapon types.

b. Lines 6 and 7 set the quantities of weapon type EW in the baseline and target divisions into scratch variables Q1 and Q2, respectively.

c. Line 8 checks whether the weapon type IW is present in either baseline or target division. If not, the type is skipped.

d. Lines 9-11 set scratch variables P1, P2, and P3 to the synthetic moduli of the weapon type IW in the baseline, target, and intermediate divisions, respectively.

e. Computation of the net modulus depends on whether the weapon type of interest is present in both baseline and target divisions or in just one of them.

(1) Line 12 checks for the weapon being in both divisions. If so, then the formula in lines 14 and 15 is applied to all five components of combat potential within the loop bounded by lines 13 and 16.

(2) If the weapon type is present only in the baseline division, the formula in line 20 is applied for all five components of combat potential within the loop bounded by lines 19 and 21.

(3) If the weapon type is present only in the target division, the formula in line 24 is applied for all five components of combat potential within the loop bounded by lines 23 and 25.

I-34. SUBPROGRAM OUTREX. Figure I-29 presents the source listing of subprogram OUTREX of the Interpolation Module. OUTREX is called by subprogram OUTX to output a single record (for one type of combat potential) of the file of final combat potentials of the intermediate division.

```
      SUBROUTINE OUTREX(IREC,IWPN,X)
      DIMENSION X(5)
      COMMON/AWRK/IWORK,IDUM,KTHTR,JTPD,JVIS,JPOS,JDAY
      COMMON/GLOBAL/IBFOR,IRFOR,JCASE
      CHARACTER*3 KTHTR
C
      WRITE(30,1) IREC,KTHTR,JTPD,JVIS,JPOS,JDAY,
     1 IWPN,X(1),X(2),X(3),X(4),X(5),IBFOR,IRFOR,JCASE
      RETURN
C
    1 FORMAT(I5,A3,I4,3I3,I5,5F10.3,2I6,I5)
C
      END
```

Figure I-29. Source Listing of Subprogram OUTREX of the AFP Interpolation Module

a. Argument IREC is the index of the record type to be output.

b. Argument IWPN is the index of the weapon type.

c. Argument X is the address of the five-vector containing the components to be output.

I-35. SUBPROGRAM OUTX. Figure I-30 presents the source listing of subprogram OUTX of the Interpolation Module. OUTX serves several purposes. OUTX outputs the final combat potentials of the intermediate division. When OUTX is called from the main program, the unmodulated and modulated CIPs have already been determined. OUTX multiplies the CIPs by the inventory quantities in order to compute the corresponding unmodulated and modulated scores. OUTX also accumulates the scores in order to determine the unmodulated and modulated COPs.

```
      SUBROUTINE OUTX
      COMMON/NEWVAL/XUCIP(5,60),XNMOD(5,60,3),PHI(60,3),VCIP(5,60)
      COMMON/FORCES/QTY(60,3)
      COMMON/NODIV/BNDW(60)
      LOGICAL BNDW
      DIMENSION BCOPS(5,2),X(5)
C
C       BLUE REC TYPES/110,130,150,170,111,151/
C
      CALL ZERO(BCOPS,10)
C
      DO 1000 IW=1,60
C
      Q=QTY(IW,3)
      IF(Q.LE.0.0) GOTO 1000
      DO 100 IV=1,5
      IF(XUCIP(IV,IW).GT.0.0) GOTO 200
  100 CONTINUE
      GOTO 1000
C
C       UNMODULATED SCORE
  200 DO 300 IV=1,5
      X(IV)=Q*XUCIP(IV,IW)
      IF(BNDW(IW)) GOTO 300
      BCOPS(IV,1)=BCOPS(IV,1)+X(IV)
  300 CONTINUE
      CALL OUTREX(110,IW,X)
C       UNMODULATED CIP
      CALL OUTREX(130,IW,XUCIP(1,IW))
C       MODULATED SCORE
      DO 400 IV=1,5
      X(IV)=Q*VCIP(IV,IW)
      IF(BNDW(IW)) GOTO 400
      BCOPS(IV,2)=BCOPS(IV,2)+X(IV)
  400 CONTINUE
      CALL OUTREX(150,IW,X)
C       MODULATED CIP
      CALL OUTREX(170,IW,VCIP(1,IW))
C
 1000 CONTINUE
C
C       UNMODULATED BCOP
C
      CALL OUTREX(111,0,BCOPS(1,1))
C
C       MODULATED BCOP
C
      CALL OUTREX(151,0,BCOPS(1,2))
C
      RETURN
```

Figure I-30. Source Listing of Subprogram OUTX of the AFP Interpolation Module

a. Line 10 calls subprogram ZERO to initialize array BCOPS() to 0.0.

b. Lines 12 and 40 define the bounds of a loop over Blue weapon types.

c. Line 14 sets scratch variable Q to the quantity of weapon type IW in the intermediate division.

d. If there are no weapons of type IW in the intermediate division, line 15 transfers control to the end of the weapon type loop, thereby skipping the current weapon type.

e. Lines 16-18 check whether the current weapon type possesses at least one nonzero component of its unmodulated CIP. If not, line 16 transfers control to skip the current weapon type.

f. Lines 22-26 put the unmodulated score in scratch vector X() and, if the weapon is divisional, add the score to the unmodulated COP.

g. Line 27 calls OUTREX to output the unmodulated score for weapon type IW.

h. Line 29 calls OUTREX to output the unmodulated CIP for weapon type IW.

i. Lines 31-35 put the modulated score in scratch vector X() and, if the weapon is divisional, add the score to the modulated COP.

j. Line 36 calls OUTREX to output the modulated score for weapon type IW.

k. Line 38 calls OUTREX to output the modulated CIP for weapon type IW.

l. Line 44 calls OUTREX to output the unmodulated COP.

m. Line 48 calls OUTREX to output the modulated COP.

I-36. SUBPROGRAM RECIP. Figure I-31 presents the source listing of subprogram RECIP of the Interpolation Module. RECIP computes and stores values of unmodulated and modulated CIPs for the baseline and target divisions. RECIP provides more precise values of CIPs than are available directly from the files of final combat potentials. RECIP simply divides the scores read from the combat potential files by the corresponding weapon quantities.

```
      SUBROUTINE RECIP
C       DIVIDE SCORES BY QUANTITIES FOR CLEANER ORIGINAL CIPS.
      COMMON/ROLLBT/WVAL(5,60,2,4)
      COMMON/FORCES/QTY(60,3)
C
      DO 1000 IFOR=1,2
      DO 900 IW=1,60
      Q=QTY(IW,IFOR)
      IF(Q.LE.0.0) GOTO 900
      DO 800 IVAL=1,5
      WVAL(IVAL,IW,IFOR,2)=WVAL(IVAL,IW,IFOR,1)/Q
      WVAL(IVAL,IW,IFOR,4)=WVAL(IVAL,IW,IFOR,3)/Q
  800 CONTINUE
  900 CONTINUE
 1000 CONTINUE
C
      RETURN
C
      END
```

Figure I-31. Source Listing of Subprogram RECIP of the AFP Interpolation Module

I-37. SUBPROGRAM USENAM. Figure I-32 presents the source listing of subprogram USENAM of the Interpolation Module. USENAM reads a file name from the runstream, builds an @USE instruction containing the name, and then calls system routine FACSF to attach the file to unit 29.

```
      SUBROUTINE USENAM
      CHARACTER*30 FNAME
      CHARACTER*50 INSTR
      READ(5,1) FNAME
    1 FORMAT(A20)
      INSTR='@USE 29,'//FNAME
      CALL FACSF(INSTR)
      RETURN
      END
```

Figure I-32. Source Listing of Subprogram USENAM to the AFP Interpolation Module

I-38. SUBPROGRAM XMCIP. Figure I-33 presents the source listing of subprogram XMCIP of the Interpolation Module. XMCIP performs the last step in computation of modulated CIPs of the intermediate division. Modulated CIPs are stored in array VCIP(). The subprogram simply multiplies the previously determined net CS/CSS moduli and unmodulated CIPs to yield modulated CIPs. XMCIP is structured as a doubly-nested loop. The outer loop runs over the Blue weapon types. The inner loop runs over the five components--personnel, light armored vehicles, heavy armored vehicles, aircraft, and scalar.

```
      SUBROUTINE XMCIP
      COMMON/NEWVAL/XUCIP(5,60),XNMOD(5,60,3),PHI(60,3),
     1 VCIP(5,60)
C
C
      DO 100 IW=1,60
      DO 90 IV=1,5
      VCIP(IV,IW)=XNMOD(IV,IW,3)*XUCIP(IV,IW)
   90 CONTINUE
  100 CONTINUE
      RETURN
      END
```

Figure I-33. Source Listing of Subprogram XMCIP of the AFP Interpolation Module

I-39. SUBPROGRAM ZERO. Figure I-34 presents the source listing of subprogram ZERO of the Interpolation Module. ZERO initializes a real array to 0.0. Argument X is the address of the array to be zeroed, and argument N is the length of the array.

```
      SUBROUTINE ZERO(X,N)
      DIMENSION X(N)
      DO 100 I=1,N
      X(I)= 0.0
  100 CONTINUE
      RETURN
      END
```

Figure I-34. Source Listing of Subprogram ZERO of the AFP Interpolation Module

I-40. SUBPROGRAM DMPMOD. Figure I-35 presents the source listing of subprogram DMPMOD of the Interpolation Module. DMPMOD lists the contents of the array XNMOD() for postrun inspection. In theory, the derived net moduli of the intermediate division should lie "between" the net moduli of the baseline and target divisions.

```
      SUBROUTINE DMPMOD
C       G.E. COOPER -- 3/6/84
      COMMON/NEWVAL/XUCIP(5,60),XNMOD(5,60,3),PHI(60,3),VCIP(5,60)
C
      DO 1000 IW=1,60
      DO 900 ID=1,3
      WRITE(6,1) ID,IW,PHI(IW,ID),(XNMOD(K,IW,ID),K=1,5)
  900 CONTINUE
      WRITE(6,2)
 1000 CONTINUE
      RETURN
    1 FORMAT(' DIV=',I1,' WPN=',I2,' PHI=',F6.3,' NMOD=',5F8.3)
    2 FORMAT(' ')
      END
```

Figure I-35. Source Listing of Subprogram DMPMOD of the AFP Interpolation Module

I-41. SUBPROGRAM NONDIV. Figure I-36 presents the source listing of subprogram NONDIV of the Interpolation Module. NONDIV reads a list of Blue nondivisional asset indices and sets elements of the local vector BNDW() accordingly. Values stored in BNDW() assure that only divisional assets are included in the computation of unmodulated and modulated COPs.

```
      SUBROUTINE NONDIV
C       G.E. COOPER -- 4/11/84
      DIMENSION NBNDX(60)
      COMMON/NODIV/BNDW(60)
      LOGICAL BNDW
C
      DO 100 I=1,60
      BNDW(I)=.FALSE.
  100 CONTINUE
C
      READ(5,1) NBNDIV,(NBNDX(I),I=1,NBNDIV)
    1 FORMAT()
C
      DO 200 I=1,NBNDIV
      BNDW(NBNDX(I))=.TRUE.
  200 CONTINUE
C
      RETURN
      END
```

Figure I-36. Source Listing of Subprogram NONDIV of the AFP Interpolation Module

I-42. MAP ELEMENT MAPDOFAX. Figure I-37 presents a listing of the MAP element for collection of the program elements of the Interpolation Module.

```
@MAP ,30DOFAX.INTERPO
IN 30DOFAX.DOFAX
IN 30DOFAX.CSCSS
IN 30DOFAX.DOPHI
IN 30DOFAX.ENVWTS
IN 30DOFAX.FS
IN 30DOFAX.FT
IN 30DOFAX.FU
IN 30DOFAX.FV
IN 30DOFAX.GETCAT
IN 30DOFAX.GETDAT
IN 30DOFAX.GETFAC
IN 30DOFAX.GETINV
IN 30DOFAX.GETROL
IN 30DOFAX.GTCATV
IN 30DOFAX.NETMOD
IN 30DOFAX.NEWCIP
IN 30DOFAX.NXMOD
IN 30DOFAX.OUTREX
IN 30DOFAX.OUTX
IN 30DOFAX.RECIP
IN 30DOFAX.USENAM
IN 30DOFAX.XMCIP
IN 30DOFAX.ZERO
IN 30DOFAX.DMPMOD
IN 30DOFAX.NONDIV
END
```

Figure I-37. Listing of the MAP Element for Collection of the Program Elements of the AFP Interpolation Module

APPENDIX J

KEY TO AFP OUTPUT REPORTS

J-1. OVERVIEW. The AFP System consists of many processes: computer programs, runstream generators, runstreams, and input, intermediate, and output data. Among all processes and programs, AFP draws somewhat arbitrary distinctions between major and minor modules. This short appendix provides a key to many available input, intermediate, and results data reports. Figure J-1 provides the standard view of the AFP System as displayed in many other appendices of this document. In some cases, a "report" may not be more than a straightforward listing for record.

J-2. REPORT KEY. Table J-1 lists AFP report-like output and the locations of the principal corresponding descriptive material within this documentation. Many descriptions include material on the included data. Any one preprocessor may consist of more than one computer program.

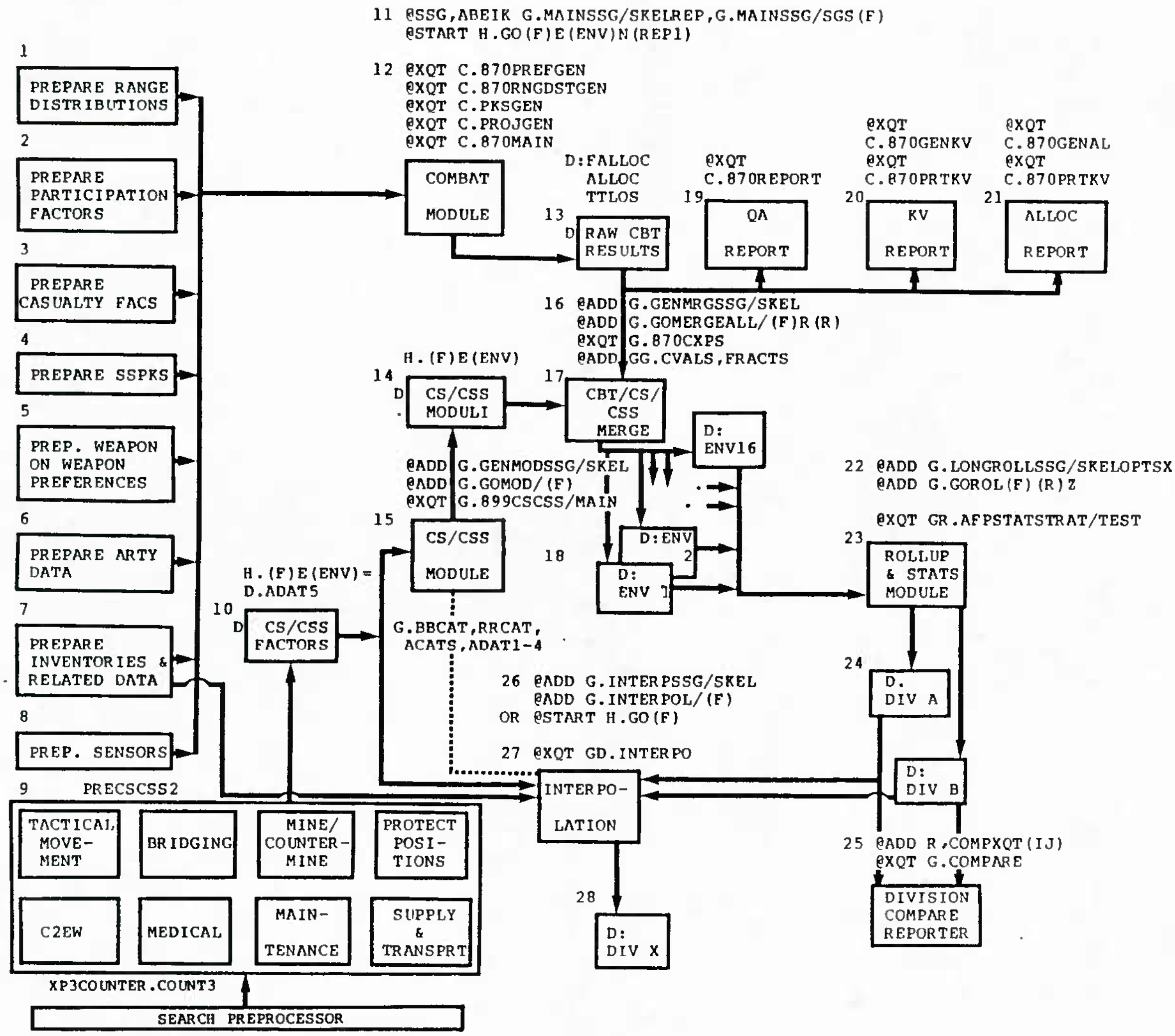


Figure J-1. The Relation Among Principal Data, Modules, Other Components, and Products of the Analysis of Force Potential (AFP) System

Table J-1. Key to AFP Output Record Copy and Report Examples and Descriptions (Page 1 of 2 pages)

Record copy or report	Block # in Fig J-1	Related process	Location of descriptions and/or examples
Basedata	6-8	Combat preproc	Annex I to Appendix D Annex VIII to Appendix D
Weapon-on-weapon preferences	5	Combat preproc	Annex II to Appendix D
Range distribution	1	Combat preproc	Annex III to Appendix D
Casualty factors	3	Combat preproc	Annex IV to Appendix D
Participation factors and engagement characteristics	2	Combat preproc	Appendix V to Appendix D
SSPKs	4	Combat preproc	Annex VI to Appendix D
Allocation scoreboard	21	Combat preproc	Appendix D, paragraph D-4c
Killer/victim scoreboard	20	Combat preproc	Appendix D, paragraph D-4d
Quality assurance report (QAREP)	19	Combat	Appendix D, paragraph D-4e
CS/CSS input	9	CS/CSS preproc	Annex I to Appendix E
CS/CSS factors	10	CS/CSS preproc	Annex I to Appendix E Annex II to Appendix E, Figure E-II-9
Special CS/CSS module input	15	CS/CSS Module	Annex II to Appendix E, Figures E-II-2 to E-II-8
CS/CSS moduli	14	CS/CSS Module	Annex II to Appendix E, Figure E-II-10

Table J-1. Key to AFP Output Record Copy and Report Examples and Descriptions (Page 2 of 2 pages)

Record copy or report	Block # in Fig J-1	Related process	Location of descriptions and/or examples
Special Merge Module input (CVALS and FRACTS)	16	CBT/CS/ CSS Merge Module	Appendix F, Figures F-3 and F-4
Raw combat report	17	CBT/CS/ CSS/Merge Module	Appendix F, Figure F-5
Partial combat potentials	18	CBT/CS/ CSS/Merge Module	Appendix B, Figure B-7 Appendix F, Figure F-6
Final combat potentials	24	Rollup & Stats Module	Appendix B, Figure B-6 Appendix G, Figure G-3
Statistical reports	24	Rollup & Stats Module	Appendix G, Figures G-4 and G-5
Division comparison	25	Division Compare Reporter	Appendix H, Figure H-3
Interpolated final combat potentials	28	Interpo-lation Module	Appendix I, Figure I-3

APPENDIX K

KEY TO AFP RUNSTREAM GENERATORS

K-1. OVERVIEW. The AFP System consists of many processes: computer programs, runstream generators, runstreams, and input, intermediate, and output data. Among all processes and programs, AFP draws somewhat arbitrary distinctions between major and minor modules. This short appendix provides a key to the AFP Systems runstream generators. Figure K-1 provides the standard view of the AFP System as displayed in many other appendices of this document. Lines in Figure K-1 with names containing the string "SSG" refer to report generators written in the UNIVAC Symstream language.

K-2. RUNSTREAM GENERATOR KEY. Table K-1 lists AFP runstream generators and the locations of the principal corresponding descriptive material within this documentation. Most descriptions include not only material on the generators but also examples of generated runstreams. As noted in many of the descriptions, the generators must be regarded as generic in the sense use of the generators may require modification of more than the SGSs. For example, many of the generators contain embedded user IDs. In the past, it has been the practice to change such IDs globally via the UNIVAC system editor rather than to define and redefine SGSs.

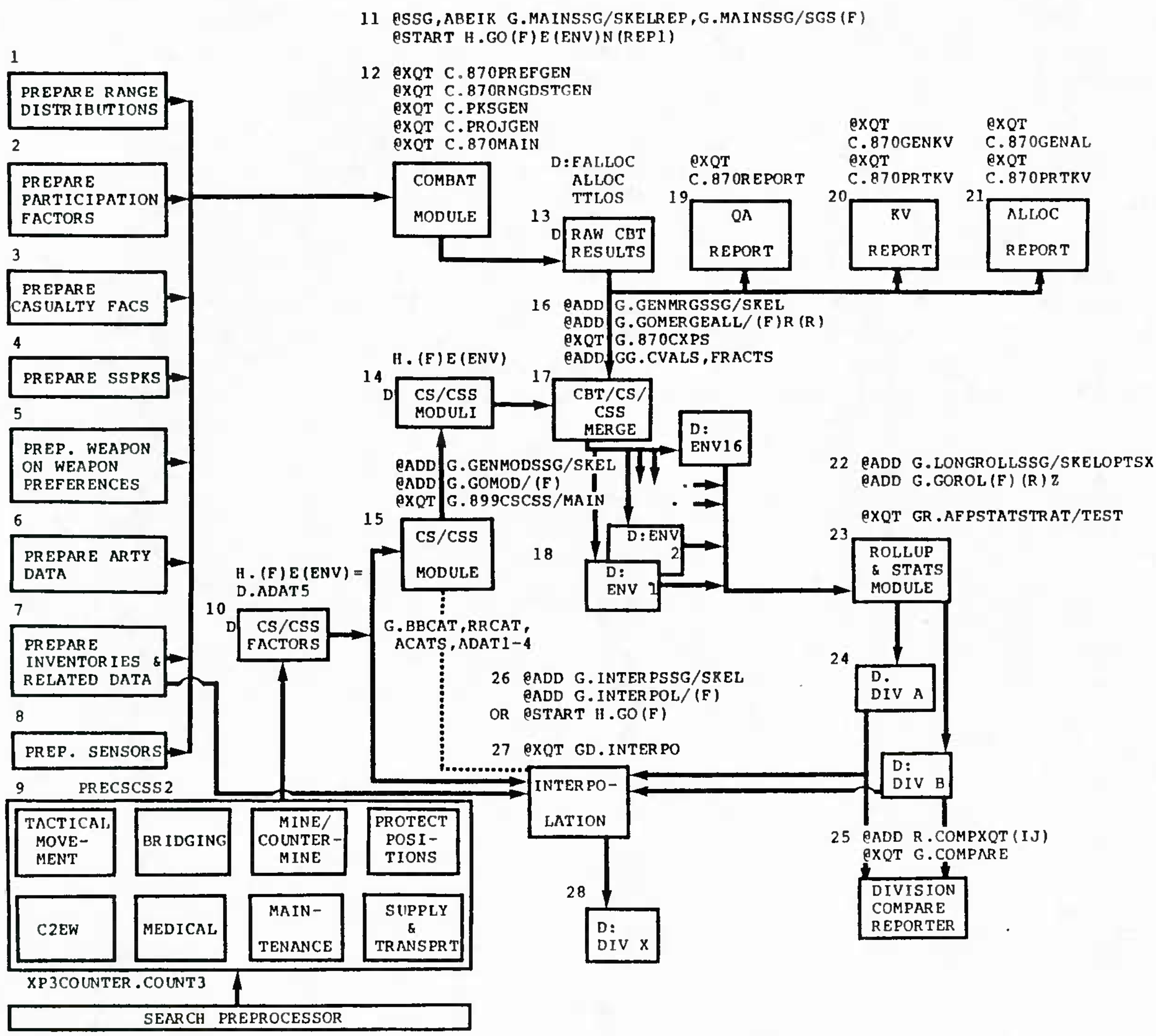


Figure K-1. The Relation Among Principal Data, Modules, Other Components, and Products of the Analysis of Force (AFP) System

Table K-1. Key to AFP Runstream Generator Descriptions and Examples

Runstream	Block # in Fig K-1	Related process	Location of descriptions and/or examples
Prepare combat module input (many)	1-8	Combat preproc	Annexes I-VI to Appendix D
Prepare CS/CSS input	9	CS/CSS preproc	Annex I to Appendix E
GENMRGSSG/SKEL	16	CS/CSS	Annex II to Appendix E
MAINSSG/SKELREP MAINSSG/SGS	11	Combat Module	Appendix D, Paragraph D-4
GENMRGSSG/SKEL	16	CBT/CS/ CSS Merge Module	Appendix F, Paragraph F-4
LONGROLLSSG/SKELOPTSX data	22	Rollup & Stats Module	Appendix G, Section IV
Example only	25	Division Compare Reporter	Appendix H, Paragraph H-4
INTERPSSG/SKEL	26	Interpo- lation Module	Appendix I, Section IV

APPENDIX L

DISTRIBUTION

Addressee	No of copies
Deputy Chief of Staff for Operations and Plans Headquarters, Department of the Army ATTN: DAMO-ZD Washington, DC 20310	1
Deputy Chief of Staff for Operations and Plans Headquarters, Department of the Army ATTN: DAMO-ODR Washington, DC 20310	1
Director of Net Assessment OSD-NA Room 3A930 The Pentagon Washington, DC 20310	1
Office of Director of Program Analysis and Evaluation OSD Room 3E836 The Pentagon Washington, DC 20310	1
Deputy Under Secretary of the Army (Operations Research) ATTN: Mr. Lester Washington, DC 20310	1
Chief of Staff, Army ATTN: DACS-DMO Washington, DC 20310	1

Addressee	No of copies
Director US Army TRADOC Systems Analysis Activity White Sands Missile Range, NM 88002	2
Director US Army Materiel Systems Analysis Activity Aberdeen Proving Ground, MD 21005	2
Director US Army Ballistic Research Laboratories Building 305 Aberdeen Proving Ground, MD 21005	2
Commander US Army Combat Developments Experimentation Command Fort Ord, California 93941	1
Director TRADOC Operations Research Activity White Sands Missile Range, NM 88002	2
Commander Combined Arms Combat Development Activity Fort Leavenworth, KS 66027	2
Commander US Army Operational Test and Evaluation Agency 5600 Columbia Pike ATTN: TRADOC LNO Falls Church, VA 22041	1
Commander Foreign Science and Technology Center 227th Street NE Charlottesville, VA 22901	1

Addressee	No of copies
Director Defense Nuclear Agency ATTN: LASS 6801 Telegraph Road Alexandria, VA 20305	1
Commander Army Research Institute 5001 Eisenhower Avenue Alexandria, VA 22333	1
Commander US Army Logistics Evaluation Agency New Cumberland Army Depot New Cumberland, PA 17070	1
Director Defense Logistics Studies Information Exchange US Army Logistics Management Center Fort Lee, VA 23801	1
Defense Technical Information Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22314	2
US Army Nuclear and Chemical Agency 7500 Backlick Road, Bldg #2073 ATTN: Library Springfield, VA 22150	1
The Pentagon Library (Army Studies Section) ATTN: ANRAL-RS The Pentagon Washington, DC 20310	2

Addressee	No of copies
Organization of the Joint Chiefs of Staff ATTN: Dr. William G. Lese, Jr. Room 1D940 The Pentagon Washington, DC 20310	1
Commandant US Army War College Carlisle Barracks, PA 17013	2
Commandant US Army War College ATTN: Director, Strategic Studies Institute Carlisle Barracks, PA 17013	1
Commandant US Air War College Maxwell Air Force Base, AL 36112	1
Commandant US Navy War College Newport, RI 02840	1
Commandant Industrial College of the Armed Forces Fort McNair Washington, DC 20319	1
President National Defense University Fort McNair Washington, DC 20319	1
Commandant National War College Fort McNair Washington, DC 20319	1

Addressee	No of copies
Commandant Armed Forces Staff College Norfolk, VA 23511	1
Commandant US Army Command and General Staff College Fort Leavenworth, KS 66027	1
Commandant US Army Command and General Staff College ATTN: Department of Combat Development, Force Development Fort Leavenworth, KS 66027	1
Commandant United States Military Academy ATTN: Mathematics Department West Point, NY 10996	1
Commandant United States Military Academy ATTN: Engineering Department West Point, NY 10996	1
Superintendent Naval Postgraduate School Monterey, CA 93943	1
Defense Systems Management School Concepts, Studies, and Simulations Building 202 Fort Belvoir, VA 22060	1
Commander/Director US Army Engineer Studies Center Casey Building, No. 2594 Fort Belvoir, VA 22060	1

Addressee	No of copies
Commander-in-Chief US Army, Europe & Seventh Army ATTN: AEAGX-OR (Mr. Dwarkin) APO New York 09403	1
Commander US Army Training and Doctrine Command ATTN: ATCD-AU Fort Monroe, VA 23651	1
Assistant Chief of Staff for Studies and Analyses US Air Force Room 1E388, Pentagon Washington, DC 20330	1
Headquarters, US Air Force Office of Worldwide Management of Studies & Analyses ATTN: AF/SAL The Pentagon Washington, DC 20330	1
AFTEC/OAN ATTN: Modelling Branch Kirtland Air Force Base, NM 87117	1
Commander Tactical Air Command TAC Analysis Group Langley Air Force Base, VA 23665	1
Joint Studies Group (TAC) ATTN: OSS Langley Air Force Base, VA 23665	1

Addressee	No of copies
Center for Wargaming Naval War College ATTN: Naval Warfare Gaming System Code 33 Newport, RI 02840	1
President Center for Naval Analyses 2000 North Beauregard Street Alexandria, VA 22311	1
Naval Research Laboratory ATTN: Code 5704 (R. M. Anderson) 4555 Overlook Avenue Washington, DC 20375	1
Commander Naval Air Systems Command ATTN: Systems Analysis Division (AIR-503) Washington, DC 20361	1
Marine Corps Operations Analysis Group Center for Naval Analyses 2000 North Beauregard Street P. O. Box 11280 Alexandria, VA 22311	1
Internal Distribution	
CAA Technical Library	2

GLOSSARY

Allocation. The term has two uses within AFP:

a. The AFP process of determining how many of each weapon type are to engage how many of each opposing weapon type. The process depends on input engagement preferences of weapon types for the opponent's weapon types and on the starting and surviving quantities of weapons. Allocation occurs at the beginning of each AFP day.

b. The AFP process of determining how many conflicts occur at each range and in each environment. (Current AFP practice is to limit each run of the AFP Combat Module to a single combat environment. Hence, allocation to environment is trivial.) This allocation process follows the one described above. In-between weapons are assigned to generalized duels at very close to the average odds generated by the first allocation process. The duels are distributed to ranges and environments. All the duels at the same range in a single environment comprise a conflict.

Assignments. For each combination of opposing types, the AFP Combat Module maintains a cumulative count of the numbers of times weapons are involved in the specific type-on-type conflicts. A weapon is counted as many times as it is assigned to conflicts. For example, a weapon that survives two conflicts and is killed in its third conflict involving the same type opponent is counted three times and contributes "3" to the tally of assignments in that type-on-type matchup. The tallies of assignments are usually reported only if Combat Module diagnostics are "turned on."

BAPD (Blue Attack Against Prepared Defense). One of the standard AFP combat postures. The Red to Blue division ratio in BAPD is 1:3.

CIP. See Combat Item Potential.

Class. Within AFP, if the word class appears alone, it usually refers to a weapon class. A weapon class is generic, e.g., the tank class, the rotary wing aircraft class. Weapons of a specific model belong to the same weapon type (see type).

Combat Item Potential (CIP). In strict terms, CIP refers only to those AFP measures of individual item combat potential that include results over all 16 combat environments with CS/CSS modulation. A CIP consists of five components: personnel, light armor, heavy armor, aircraft, and scalar. The scalar is a target-value-weighted sum of other components. The first four components may be produced with or without weighting by target values. If target value weighting is applied to the "personnel" component, that component then includes some weighted weapons, usually small arms, as well. The AFP system generates intermediate results in the same format as strictly defined CIPs. These intermediate values are usually referred to as "partial CIPs." The strictly defined CIPs, in these terms, are "final CIPs." A CIP is an estimate of a weapon's potential achievement during the first half of its lifetime. The estimate of achievement is strictly relative to the context of the analysis, i.e., relative to the quantities and qualities of friendly and opposing weapons in the specified combat environments at specified environmental weights.

Combat Item Score. For a type item, the product of that item's CIP, and the starting number of that item in a division.

Combat Organization Potential (COP). The sum of all the combat item scores of a division. In accord with the convention described for Combat Item Potentials (CIPs), COP refers to division potential weighted across all 16 combat environments and with CS/CSS modulation. Similarly, the distinction between partial and final COPs is made.

Combined Preference and Participation Factor. For indirect fire weapons, a single factor which is the product of the preference and participation factors separately input and defined for direct fire weapons. (See Preference Factor and Participation Factor.)

Conflict. A collection of pure type-on-type direct fire duels at a specific range and environment. Only one weapon type from each side participates in the duels of a conflict. In current AFP practice, an AFP conflict lasts 2.01 minutes. There are four successive conflicts in an AFP day. The survivors of one conflict may participate in the succeeding conflict on the same day. Weapons cannot enter another kind of conflict on an AFP day. That is, once assigned to an opposing type weapon, range, and environment, a weapon (as long as it survives) spends the day so engaged in all conflicts. The logical structure of the AFP Combat Module is such that it loops over all the duels in a conflict for the same shot cycle (see Shot Cycle) before proceeding to the next shot cycle for all those same duels.

Conflict Duration. Current AFP practice is for all conflicts (and, hence, duels) to last 2.01 minutes. The 0.01 minute is to avoid possible ambiguity with weapons with refire times that might fall exactly on the 2-minute mark. Direct fire may cease in less than 2.01 minutes if a weapon's targets are depleted. However, the "duel" continues in order that survivors of the direct fire exchanges remain liable to attrition from continuing indirect fire.

COP. See Combat Organization Potential.

Counterbattery Fire. The AFP Combat Module makes a distinction between two roles of indirect fire weapons. Indirect fire against ordinary direct fire weapons is treated as an add-on to the otherwise pure direct fire type-on-type engagements, conflicts, and duels. In those cases, the indirect fire weapons may inflict damage, but they do not receive fire from the direct fire weapons or from other indirect fire weapons. Indirect fire weapons can only suffer attrition when they, too, are treated as direct fire weapons. Counterbattery fire, just in AFP terms, is represented as direct fire engagements, conflicts, and duels between indirect fire weapons. The main difference here is that the indirect fire weapons are not subject to direct fire detection logic. Counterbattery fire logic is complicated inasmuch as the other indirect fire weapons assigned to fire at direct fire weapons may also fire at those counterbattery weapons treated as direct fire weapons.

CS/CSS Modulation. The process performed by the CBT/CS/CSS Merge Module by which the CS/CSS moduli generated by the CS/CSS Module is applied to or merged with results of the Combat Module. The results of type-on-type engagements represented by the Combat Module are multiplied, after the fact, by moduli corresponding to the specific type-on-type matchups.

CS/CSS Moduli. Multipliers produced by the AFP CS/CSS Module for application to results of the AFP Combat Module. The moduli reflect weapon-type-on-weapon-type adjustment factors reflecting independent analyses of opposing divisions' combat support and combat service support functions. In current AFP practice involving up to 60 friendly and up to 60 opposing weapon types, the CS/CSS Module generates 7,200 moduli, many of which possess identical values.

Day. In AFP terms, an abstraction consisting of a given number of successive conflicts. In current AFP practice, a day consists of four conflicts, and a Combat Module run consists of 2 days. The allocations of initial or surviving weapons occur at the beginning of AFP days.

Deep Range. In current AFP practice, the farthest of the six ranges represented in direct fire engagements within the AFP Combat Module are reserved for targets of indirect fire. Equipment in the deep range may include that belonging to headquarters and command posts, moving forward, or remaining in reserve.

Detection. The process by which a potential direct fire shooter determines a target. The detection routines in the AFP Combat Module were adapted from the CARMONETTE versions and apply logic developed at the Night Vision Laboratory and Combined Arms Center. A weapon not credited with detecting a target by the referenced detection routines may return fire after a specified number of shots have been fired by its opponent(s). Indirect fire weapons are assumed to have been assigned targets prior to the first conflicts represented within the AFP Combat Module.

Detection Time. The term, as used within AFP, is partly a misnomer. The Combat Module's detection routines return a "detection time," which may be very large to indicate nondetection. A first shot is fired by a direct fire weapon when its so-called detection time has elapsed. Hence, detection time is time to first shot in the special world of AFP. Furthermore, AFP does not represent the flight time of projectiles. Hence, detection time is also the time to first round impact to first impact from the beginning of the detection sequence.

Direct Fire. Most direct fire represented within the AFP Combat Module conforms to the usual notion of fire along line of sight. However, AFP applies its direct fire logic (less detection) to indirect, counterbattery fire.

Duel. The smallest weapon-on-weapon interaction represented within the AFP Combat Module is a duel. However, AFP generalizes the dictionary one-on-one definition of a duel to include one-on-N, where N can be one or more. In all AFP duels, at least one side possesses just one weapon in a duel. The sides in a duel are separated by a fixed range. A weapon within a duel may only fire upon its opponent in that same duel.

Engagement. In the hierarchy of combat interaction, AFP uses the term engagement to mean the interactions between direct fire weapons of a single type on each side. One result of the AFP allocation process of the first kind is to decompose the inventories of both sides into engagements. For example, one engagement may consist of m of M total DRAGONs versus n of N total T-62 tanks. The AFP allocation process of the second kind distributes the m-on-n engagement into duels, distributes the duels by range and environment (only one environment in current AFP practice), and then groups all duels at the same range and environment into a conflict. (See also Duel and Conflict.) In principle, opposing inventories of 60 different weapon types on each side may be composed into 60 x 60 = 3,600 engagements which may be distributed into many thousand of duels and then regrouped into smaller numbers of conflicts.

Environment (often prefixed as combat environment). The condition under which a conflict within the AFP Combat Module is assumed to occur. In current AFP practice, all conflicts within a single Combat Module run occur under the same combat environment. In principle, the Combat Module permits multiple environments within a single run; however, other AFP modules cannot correctly process the results of multienvironment Combat Module runs. Combat environment combines three features:

a. Posture (currently four: Red Attack against Prepared Defense (RAPD), STATIC, Red Attack against DElaying defense (RADE), and Blue Attack against Prepared Defense (BAPD).

b. Visibility (currently two: clear and degraded).

c. Brightness (currently two: day and night).

Combinations of these features yields 4 x 2 x 2 = 16 distinct combat environments. All 16 environments are included in standard AFP application.

Environmental Site. The logic of the AFP Combat Module permits combat to be represented for different combat environments at different sites. Indeed, the representation of different environments within a single Combat Module run requires different sites. Although input to the Combat Module permit designation of multiple sites, current AFP practice is to represent only one environment and only one site per run. Exercise of the multisite, multienvironment option in the Combat Module is discouraged because other AFP System modules do not include the same option.

External Loss. If nonzero external loss factors are input to the AFP Combat Module, the corresponding fractions of otherwise available weapons will be debited at the beginning of each AFP day. "External" loss factors provide a way to degrade weapon availabilities as a result of causes "external" to the AFP Combat Module.

Fractional Lifetime. AFP indices of combat potential are estimates of the potential achievements of weapons or organizations at some specified fractional loss of many types of weapons. In current AFP practice, that fraction is set to one-half. Hence, for those weapons subject to fractional lifetime criterion, AFP potentials are "half-life" potentials.

Half-life Potential. See Fractional Lifetime.

Indirect Fire. Indirect fire within the AFP Combat Module very nearly conforms to the usual notion of area fire on direct fire and weaponless targets. AFP introduces a special wrinkle by permitting indirect fire to fall on the special AFP counterbattery versions of direct fire engagements.

Interpolation. AFP interpolation is a special process performed by the AFP Interpolation Module. As currently implemented, interpolation permits Blue CIPs and COPs to be estimated for divisions with inventories "between" base and target divisions whose final combat potentials are already known. Interpolation is of the order of 100 times as fast as application of the full AFP process. Among other things, the "between" inventory must not include any weapon type not belonging to the base or target division.

Modulated. Term applied to components of combat potentials indicating that CS/CSS moduli have been applied. (See CS/CSS moduli.)

Odds Class. A result of the AFP Combat Module's allocation process of the first kind is the distribution of weapon types against opposing weapon types. If M weapons of a type are allocated against N weapons of an opposing type, the overall weapon ratio is M:N. The Combat Module decomposes the M:N into AFP duels (always only one of at least one weapon type in each duel) at odds of q:1 and (q+1):1. The odds classes of the duels are q:1 and (q+1):1. There are never more than two odds classes. There may be a single odds class. Note that the effect is to produce one or two odds classes as close as possible to the average odds M:N given that each duel include no more than one of the weapon types.

Participation. The AFP Combat Module accepts participation factors among its input. Such factors should lie on the closed interval 0.0 to 1.0. The factors provide a way for the Combat Module to withhold some otherwise available weapons on each AFP day. Weapons may fail to participate for any of many reasons not represented in the Combat Module. Unlike external losses, nonparticipating weapons are not considered permanent losses.

Preference. Perhaps the most novel notion applied within AFP is that of weapon engagement preferences. Every weapon type has a preference for engaging each of the opposing weapon types. A preference factor must lie on the interval 0.0 to 1.0. For any one weapon type, the sum of its preference factors for all opposing weapon types must never exceed 1.0. Preferences apply only to direct fire weapons. Inasmuch as AFP represents counterbattery fire as special direct fire, indirect fire weapons may have nonzero preference factors for one another; and those factors for any one weapon may sum to less than 1.0 permitting the remaining weapons of the type to engage in ordinary indirect fire. A zero preference need not prevent a weapon from engaging an opposing type if the opposing weapon has a nonzero preference for the weapon in question. Indeed, AFP modifies opposing preferences by adjusting them one-half the way toward their common mean. Preferences are applied within the AFP allocation process of the first kind. Final allocation depends not only on the modified preference factors but also on the opposing inventories. Weapons will not be allocated against preferred weapons that do not exist.

RADE (Red Attack against DElaying defense). One of the standard AFP postures. The Red to Blue division ratio in RADE is 4:1.

Range. The AFP Combat Module distributes direct fire duels over six ranges. In current AFP practice, the standard ranges are at 250, 500, 1,000, 1,500, 2,500 meters, and at "deep" range.

Range Distribution. AFP Combat Module input include range distributions for opposing pairs of weapon types. A fraction is specified for each of the six AFP ranges. The fractions across the six ranges should total 1.0. Duels are distributed in accord with the input fractions. Only whole duels are distributed. One consequence of distributing only whole duels is that fewer than six duels cannot possibly "occupy" all six ranges, The first duel is assigned to the range with the highest fraction. The next duel is assigned to the next highest fraction. The range distribution is performed for each odds class (never more than two odds classes) independently.

RAPD (Red Attach against Prepared Defense). One of the standard AFP combat postures. The Red to Blue division ratio in RAPD is 3:1.

Refire Time. The mean refire times of weapons are specified by input to the AFP Combat Module. Indirect weapon refire is not considered subject to randomness. Direct fire weapons are assumed to possess refire times lognormally distributed with the input means and with standard deviations equal to the means.

Score. See Combat Item Score.

Shot Cycle. The AFP Combat Module does not "keep time" in the sense of usual combat models or simulations. Within the Combat Module before a duel begins, the Module calculates how many shots would be fired within the alloted time given the numbers of weapons and their refire times. When a duel begins, the Combat Module counts shots instead of time. In general, the underlying time increments from shot cycle to shot cycle will be unequal. The inequality may arise because opposing sides may be firing weapons with different refire times and because refire times are drawn as lognormally distributed variates.

Stage. See Shot Cycle. For most purposes, an AFP stage may be considered a synonym for an AFP shot cycle. However, a subtle distinction can be drawn. The AFP Combat Module loops over all the duels of a conflict for the same shot cycle in each duel before processing the next shot cycle in any duel. In these terms, a shot cycle appears to be a feature of a duel, and a stage appears as the collection of the same shot cycle across duels.

STATIC. One of the standard AFP combat postures. The Red to Blue division ratio in STATIC is 1:1.

Supertroop. This possibly misleading term has nothing to do with the quality of troops. AFP adopted the term to represent aggregation of weapons, usually small arms. Representation of individual small arms had led to prohibitively long Combat Module execution times. Experiments with riflemen treated as groups of 10 were successful. Such grouping came to be known as "supertrooping," and the term stuck. In all cases to date, supertrooping has been by 10s. Note that SSPKs need not be adjusted when supertrooped weapons face supertrooped weapons. However, SSPks must be modified when supertrooped weapons face nonsupertrooped opponents. The appropriate PCAS entry for a 10-fold supertrooped weapon should be 10 times the nonsupertrooped value.

Surge. As used with AFP, the word surge refers to changes in casualty levels with posture. The medical combat service support measure within the AFP CS/CSS Module depends, in part, on casualty rate. With the STATIC posture considered a base case with "surge factor" of 1.0, surge factors for others' postures are determined by taking the ratio an opposing division's personnel combat potential component per opponent's division to the corresponding potential component in STATIC posture. For example, a Blue RAPD surge factor would be given by:

RAPD Blue Surge =

((RAPD Red Personnel Potential)*3)/((STATIC Red Personnel Potential)*1)

(The factor "3" in the numerator is there because the Red to Blue division ratio in RAPD posture is 3:1. Hence, the potential of each of three Red divisions may be inflicted against the single Blue division. The factor "1" in the denominator is there because the division ratio in STATIC posture is 1:1.)

Survivor Reallocation. If survivors remain after a conflict ends, and if the day's maximum number of conflicts have not been reached, the survivors are regrouped into duels, and the next conflict is processed. The new duels are against the same opposing weapon type and at the same range.

Type. If the word type is used alone within AFP, it usually refers to a weapon type. The fundamental direct fire interaction represented within the AFP Combat Module is between two opposing weapon types. In AFP terms, M60A1 and M60A3 tanks are different weapon types. (Tanks of different types belong to a weapon "class." (See class.)

Unmodulated. Term applied to components of combat potentials indicating that CS/CSS moduli have not been applied. See CS/CSS Moduli.

ANALYSIS OF FORCE POTENTIAL SYSTEM (AFPSYS)	STUDY SUMMARY CAA-D-84-14

THE REASON FOR PERFORMING THE STUDY is primarily widespread dissatisfaction with previous combat potential estimation methods that do not give enough attention to influences noted below in the study objectives.

THE PRINCIPAL FINDINGS during AFP System development and implementation and as evidenced by illustrative examples in the Operator's and Programer's Guide to the AFP System and by the parallel MICAF Study application are:

(1) All modules, submodules, and special processors of the AFP System for estimating the static combat potential of equipment and organizations have been tested and perform as designed.

(2) AFP estimates of static combat potentials depend on input to the AFP System and are sensitive to opposing sides' weapon characteristics, weapon quantities, type-on-type engagement preferences, environmental conditions, and combat support and combat service support levels.

(3) Full application of the AFP System is labor, data, and computer intensive.

THE MAIN ASSUMPTIONS for purposes of estimating static combat potentials:

(1) The large-scale battlefield may be decomposed into separate firepower-counterfirepower, combat support, and combat service support processes. These processes may be analyzed largely independently. Their separate results may be combined afterward to yield estimates of combat potentials.

(2) Total division firepower-counterfirepower processes may be decomposed into pure weapon type on pure weapon type engagements. The engagements may be further decomposed into still smaller matchups in which at least one weapon opposes one or more weapons. Only indirect, area fire weapons may impinge on the interaction of otherwise pure type-on-type "duels." The usual techniques of dynamic modeling and simulation need not be applied except to the independent duels of relatively short duration.

(3) Movement and maneuver need not be represented within the firepower-counterfirepower process. Tactical mobility may be treated adequately within the combat support and combat service support processes. Duels are distributed to fixed ranges.

THE PRINCIPAL LIMITATIONS

(1) Like all static indicators, AFP combat potentials may be inappropriate bases for estimating prolonged, fluid combat.

(2) Because AFP combat potentials depend on weighted averages for 16 distinct combat environments, the potentials may not be useful estimators for differently weighted or different environments. For example, interest in just one of the combat environments implies a vastly different weighting: just one 1.0 and 15 0.0's.

(3) AFP combat potentials are estimates of achievement for the very special circumstance in which one's own weapons are 50 percent attrited. (This is why AFP combat potentials are often called "half-life potentials.") In general, the potentials do not correspond to any one common moment in projected real time because different weapon types do not reach 50 percent survival at the same instant.

(4) In its current implementation, the AFP System does not represent suppression nor the effects of echelons above division (other than some nondivisional artillery and some fixed wing aircraft).

THE SCOPE OF THE STUDY included development and implementation of the AFP System and parallel support of the MICAF Study. The Operator's and Programer's Guide to the AFP System provides a wealth of information needed in maintaining and applying the AFP System. Some applications of the AFP System have been made in support of other studies. In particular, the MICAF I and II Studies depended heavily on AFP, and AFP "results" may be found in the MICAF I and II reports.

THE STUDY OBJECTIVES are to develop and demonstrate (via the parallel MICAF application study) a new method for estimating the static combat potential of equipment and organizations. That method is to depend more directly on quantitative data, full division inventories of opposing equipment, combat support, combat service support, and wider range of combat environments than in previous approaches.

THE BASIC APPROACH of AFP is to begin with a highly stylized abstraction of the battlefield, decompose the battlefield into separate processes, provide extensive input data to drive those processes, and then operate a system of specially developed computer programs which replicate estimates of kills and losses for 16 different combat environments, project those estimates to half-lives, modify the estimates in accord with support levels, and roll up everything into final estimates of combat potential.

THE STUDY SPONSOR is the Director, CAA.

THE STUDY EFFORT was directed by Mr. Gerald E. Cooper, Strategy, Concepts and Plans Directorate. All directorates contributed.

COMMENTS AND QUESTIONS may be directed to US Army Concepts Analysis Agency, ATTN: Assistant Director for Requirements and Resources, 8120 Woodmont Avenue, Bethesda, MD 20814-2797