AD-A110 899 LOGISTICS MANAGEMENT INST WASHINGTON DC F/G 15/5
THE SORTIE-GENERATION MODEL SYSTEM. VOLUME IV. SORTIE-GENERATIO--ETC(U)
SEP 81 M J KONVALINKA, J B ABELL MDA903-81-C-0166
UNCLASSIFIED LMI-ML102-VOL-4 NL

1 OF 3
AD
A110899

LEVEL II

THE SORTIE-GENERATION MODEL SYSTEM
VOLUME IV
SORTIE-GENERATION MODEL
PROGRAMMER'S MANUAL

September 1981

Michael J. Konvalinka
John B. Abell

Prepared pursuant to Department of Defense Contract MDA903-81-C-0166 (ML102). Views or conclusions contained in this document should not be interpreted as representing the official opinion or policy of the Department of Defense.

LOGISTICS MANAGEMENT INSTITUTE
4701 SANGAMORE ROAD
WASHINGTON, D.C. 20016

PREFACE

This volume is the fourth of six volumes that describe the LMI Sortie-Generation Model System. Volume I, Executive Summary, discusses the problem the system is designed to address and provides an overview of the principal parts of the system. Volume II, Sortie-Generation Model User's Guide, provides sufficient information to allow a user to run the Sortie-Generation Model (SGM). Volume III, Sortie-Generation Model Analyst's Manual, describes the mathematical structures, derivations, assumptions, limitations, and data sources of the SGM at a very detailed level. Volume IV, Sortie-Generation Model (SGM) Programmer's Manual, specifies the details of the computer programs, file structures, job control language, and operating environment of the SGM. Volume V describes the maintenance subsystem and explains the construction of the maintenance input file to the SGM. Volume VI describes the spares subsystem and shows a user how to build the spares file that is used by the SGM.

Potential users are cautioned that no volume is intended to provide, by itself, all of the information needed for a comprehensive understanding of the operation of the SGM.

ACKNOWLEDGMENTS

We are indebted to Professor Edward J. Ignall of Columbia University for his many imaginative contributions to the formulation of the Sortie-Generation Model; to Professor Louis W. Miller of the Wharton School, University of Pennsylvania, who first suggested the basic structure of the multiple-server queueing system incorporated in the model; to Professor John A. Muckstadt of Cornell University for his helpful suggestions early in the model's development; and to our colleague at LMI, Mr. David L. Goodwin, who contributed greatly to the software development, and to Ms. Susan F. Flynn, for her contributions to the material in Chapters 4 and 6 and her significant improvements in the SGM software design.

TABLE OF CONTENTS

LIST OF FIGURES

VOLUME IV

SORTIE-GENERATION MODEL PROGRAMMER'S MANUAL

1. MODEL DESCRIPTION

INTRODUCTION

The purpose of this chapter is to describe the basic logical structure of the Sortie-Generation Model (SGM). This description provides a useful framework for understanding the details of the computer implementation which are provided in the later chapters of this volume. For the reader's convenience, the remainder of this chapter is included here and may also be found in Volume II, Sortie-Generation Model User's Guide.

STATES AND PROCESSES

The SGM is a hybrid analytic/simulation model that estimates the expected maximal number of sorties that can be flown by a specified aircraft type in a wartime scenario. This estimate is based on aircraft characteristics, maintenance manpower and recoverable spares levels, and user inputs that describe the scenario of interest.

The SGM consists of a collection of aircraft <u>states</u>, <u>processes</u> that cause transitions between states, and <u>logic</u> that governs those processes. The SGM simulates the transition of aircraft between these states throughout a daily flying schedule that is specified by the user. The definitions of the states, the logic of the state transitions, and the interaction of these transitions with the flying schedule determine the basic structure of the SGM.

Aircraft States

There are five aircraft states in the SGM:

1) Mission-capable
2) Maintenance
3) Not mission-capable, supply (NMCS)
4) Combat loss
5) Reserve

These states are mutually exclusive and collectively exhaustive; i.e., every aircraft is in one and only one state. The states are described below.

Mission Capable. An aircraft is considered mission-capable if it is capable of flying a combat mission. It is not mission-capable if it is undergoing essential corrective maintenance, or is missing a mission-essential part. There is no explicit representation in the SGM of aircraft that are partially mission-capable.

Maintenance. An aircraft is in maintenance status if it requires unscheduled, on-aircraft repair that is essential to the performance of its mission. This repair may or may not be due to failure of a part; however, in this model, an aircraft is not allowed to enter maintenance until all needed parts have been obtained from supply or repair.

NMCS. An aircraft is not mission-capable, supply if the aircraft is missing an essential part. In the SGM, only mission-essential Line Replaceable Units (LRUs) can cause an aircraft to become NMCS.

Combat Loss. An aircraft is counted as a combat loss if it does not return from a sortie. Once an aircraft has been lost it can never be recovered. Battle-damaged aircraft that return from a sortie are not considered in this model.

Reserve. Reserve aircraft consist of mission-capable aircraft that are used to replace combat losses. The user specifies an initial number of aircraft that are held in reserve at the beginning of the scenario; these reserve aircraft replace combat losses at the end of each day, until all reserves have been exhausted. Aircraft are allowed to leave this reserve state, but no aircraft can enter it.

Processes - Transitions Between States

There are eight processes in the SGM which cause transitions between aircraft states:

1) Ground aborts
2) Breaks
3) Aircraft repairs
4) Parts demands
5) Parts repair
6) Cannibalization
7) Attrition
8) Commitment of reserves

Figure 1-1 depicts the relationships among the various states and processes.

EVENTS

The events that occur in the SGM are related to a flying schedule with user-specified characteristics. The flying schedule consists of a number of periods or cycles each of which is divided into three segments of lengths T_L, T_F, and T_W, respectively. During the last period, the T_W segment is replaced by an overnight recovery period. The user specifies the first and last take-off times of the day; the time, T_L, which is the average minimal length of time required to launch a sortie given a mission-capable aircraft; the sortie length, T_F; and the number of periods per day. The time, T_W, is then computed by the SGM program. The flying schedule is the same each day except for the number of aircraft to be flown each period, which the user can vary. A typical flying schedule is portrayed in Figure 1-2.

Figure 1-3 portrays two segments of a flying day; the flying period on the left is intended to be typical and the one on the right to be the last period of the day. The events that occur in the SGM are denoted by circled

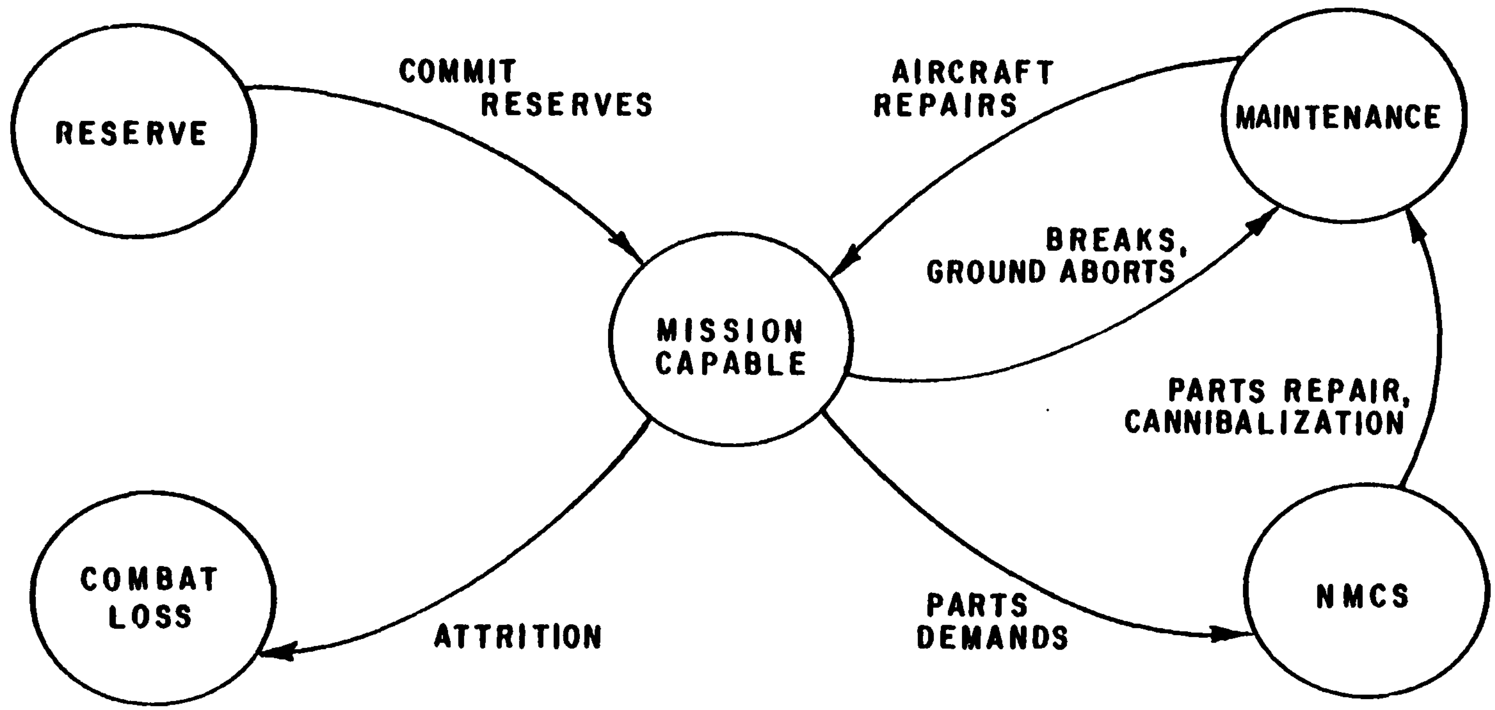
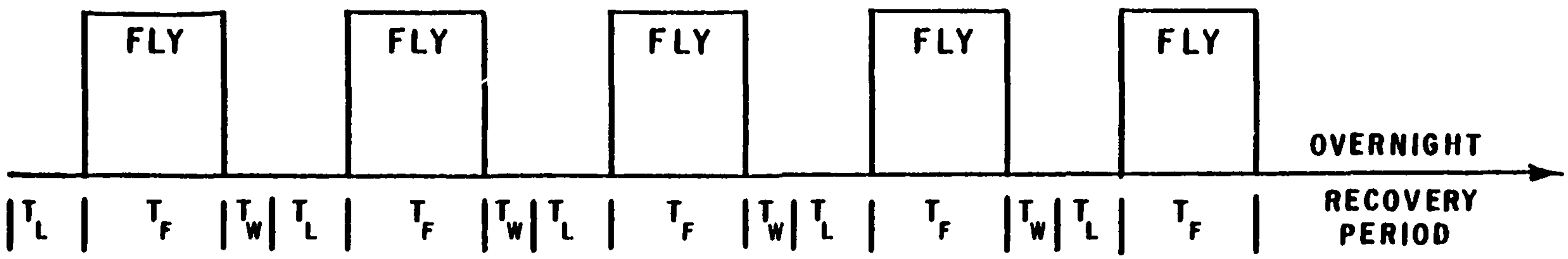


FIGURE 1-1
STATES AND PROCESSES



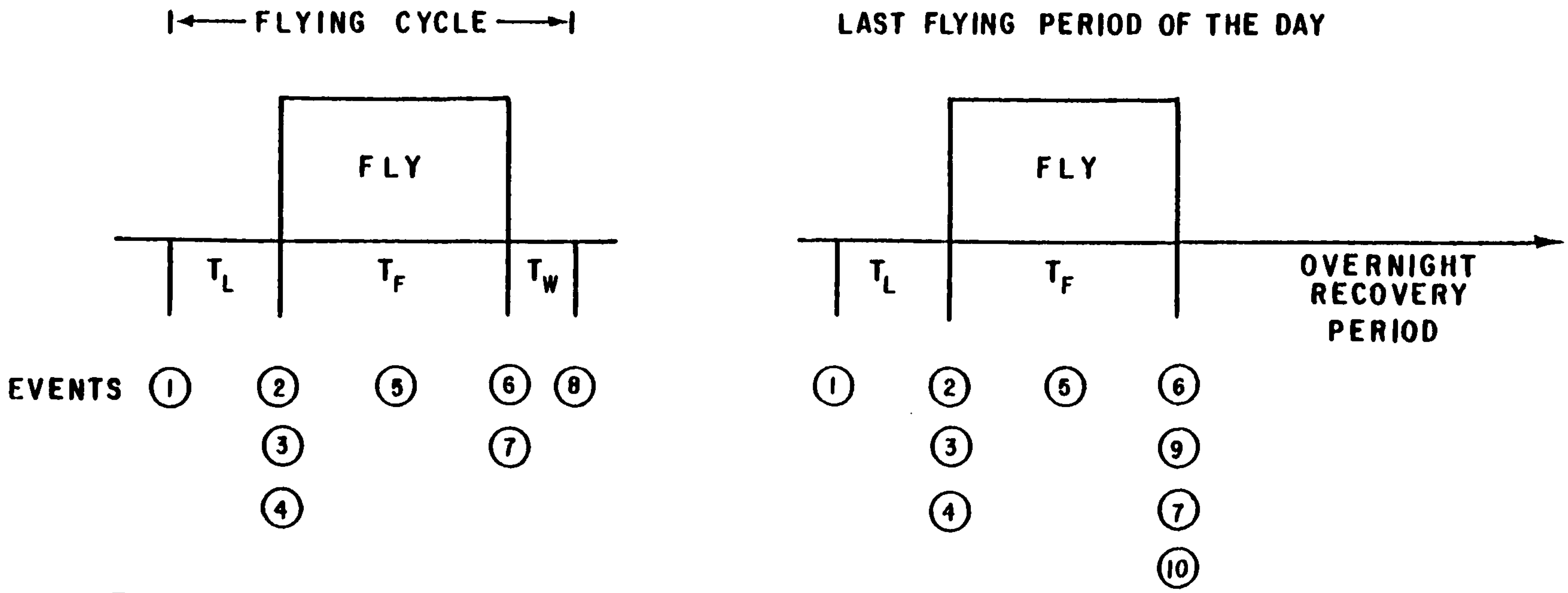
T_L = MINIMAL TIME TO LAUNCH GIVEN A MISSION-CAPABLE AIRCRAFT

T_F = SORTIE LENGTH

T_W = WAITING TIME (FIXED BY NUMBER OF PERIODS. FIRST AND LAST TAKEOFF TIMES, T_L AND T_F)

FIGURE 1-2

A TYPICAL FIVE-PERIOD FLYING DAY



T_L = MINIMAL TIME TO LAUNCH GIVEN A MISSION-CAPABLE AIRCRAFT

T_F = SORTIE LENGTH

T_W = WAITING TIME (FIXED BY NUMBER OF PERIODS, FIRST AND LAST TAKEOFF TIMES, T_L AND T_F)

FIGURE 1-3

SGM FLYING CYCLE

numbers placed under the figures at the appropriate positions on the time line. Each of those events is described here.

Event 1

All mission-capable aircraft are prepared for launch. Any aircraft that is not mission-capable at this time (i.e., T_L before takeoff) cannot be flown during this cycle because, by definition, T_L is the minimal time required to launch an aircraft that is mission-capable.

Event 2

Aircraft that are repaired during the period of length T_L leave maintenance and become mission-capable but are not available for flight during this cycle.

Event 3

All aircraft that were prepared for takeoff are subjected to the probability of ground abort. A ground abort is defined as an unsuccessful attempt by an aircrew to fly an aircraft. The aborted aircraft enter maintenance. No parts demands are generated by ground aborts.

Event 4

The remaining aircraft that were prepared for takeoff fly sorties.

Event 5

Each aircraft that flies is subjected to the probability of attrition and, for each combat loss, an aircraft is deducted from the current strength of the organization.

Event 6

Aircraft that are repaired during the period of length T_F leave maintenance and become mission-capable.

Event 7

Each aircraft returning from flight is subjected to the probability of break, i.e., the probability of requiring essential corrective maintenance prior to flying another combat mission. At the same time, parts demands are generated. Demands that can be filled from stock on-hand result in issues of that stock. Demands that cannot be filled from stock and cannot be satisfied by cannibalization from aircraft that are NMCS result in additional aircraft becoming NMCS.

Event 8

Aircraft that are repaired during the period of length T_W leave maintenance and become mission-capable.

Event 9

This event occurs only after the last flight of the day. It accounts for the parts repair process by subjecting each part in repair to the probability that the repair was completed during the preceding 24 hours. Remaining parts shortages are consolidated on as few aircraft as possible. If the consolidation results in fewer NMCS aircraft than before, the aircraft leaving NMCS status enter maintenance at this time.

Event 10

This event also occurs only after the last flight of the day. Combat losses may be replaced by available reserve aircraft, if the user so specifies. Any remaining reserve aircraft after losses have been replaced are committed according to user specification in the scenario input parameter. If reserves are to be used only as attrition fillers, then any remaining reserve aircraft are left in the reserve pool; thus, the UE for the scenario will never increase. If the user has selected the reserve augmentation mode, then all reserve aircraft will be committed when they become available; hence, the UE for the scenario may actually increase.

REPAIR PROCESS

The entry of an aircraft into maintenance results from a ground abort or a "break" during a sortie. In either case, following the ground abort or sortie, the aircraft is subjected to a sequence of random draws that determines the subset of work centers that will be involved in the maintenance on that aircraft. A work center is a set of maintenance personnel with a particular skill. Examples of work centers are the structural repair shop, the hydraulic shop, and the automatic flight control system shop.

In the construction of the maintenance data base that supports the SGM, the aircraft repair times for all work centers involved in the repair of the aircraft are measured from the time of the ground abort or landing of the aircraft. For each work center involved in the repair, a random draw is made from an exponential distribution of repair time for that work center. The mean of that distribution is the reciprocal of the service rate contained in the maintenance data base for the work center in question. All work centers involved in the repair are assumed to work on the aircraft simultaneously; thus, the recovery time of the aircraft is simply the longest of the repair times for all the work centers involved in the recovery of that particular aircraft.

In the SGM, once the aircraft leaves maintenance and becomes mission-capable again, it loses its identity and is counted simply as another aircraft in the mission-capable pool.

2. COMPUTER IMPLEMENTATION

INTRODUCTION

This chapter describes the computer program implementation of the logical structure discussed in Chapter 1. The purposes of this description are to provide a macro-level view of the model implementation, identify its current computer environment, and discuss the software design goals used in the development of the SGM routines. The micro-level information such as detailed routine descriptions, common definitions, error-message descriptions, etc., is presented in later chapters and appendices.

The first portion of the chapter provides a general overview of the model architecture. The processing flow and major components are identified and discussed. This overview should help a programmer understand how everything is tied together in the model and provide a starting point for detailed maintenance or modification of the model.

The next portion of the chapter provides a short description of the computer environment in which the SGM was developed and is currently operating. References to the appropriate computer manuals are provided in Appendix A.

The remainder of the chapter explains the design goals used in the development of the SGM routines. Appendix C contains detailed documentation of the individual routines as part of the program listings. Included with the discussio of the SGM routines is a description of the system utilities used in the SGM. The use of such utilities has been minimized to allow the SGM to be more easily moved to different computer environments. The description provides sufficient detail to enable a programmer to develop similar routines if they are unavailable on another system.

MODEL STRUCTURE

Figure 2-1 shows the basic structure of the SGM. This block diagram describes the processing flow between the major routines comprising the SGM. The SGM is a three-phase process: the initialization phase in which the scenario parameters, aircraft maintenance work centers, and spares data are initialized; the performance of the actual simulation; and finally the printing of the sortie results collected during the simulation. Descriptions of these phases follow.

Initialization

The INIT routine performs the initialization steps necessary to prepare the various variables, arrays, and tables needed for the simulation. As shown in Figure 2-2, initialization is a three-step process. First, the scenario input parameters are loaded and initialized by the INITSCN routine. This routine reads the scenario input file (file-01) created by the Set-Parameter Program, creates the temporary scratch file containing the parameters which are allowed to vary on a daily basis, and prints a scenario summary listing the parameters for this SGM run.

The second step initializes the information describing aircraft maintenance in the work centers. The INITWC routine loads the work-center input data (file-02), computes the break-rate probabilities needed for sampling work-center loading, and prints a summary of the work-center parameters for this SGM run.

Finally, the spare-parts variables are initialized by the INITPRT routine. The spares inputs are loaded (file-04), and statistics and probabilities needed for parts sampling are computed. Due to the large number of part types used with each SGM run, a listing of the spares inputs is not provided as part of the SGM output results.

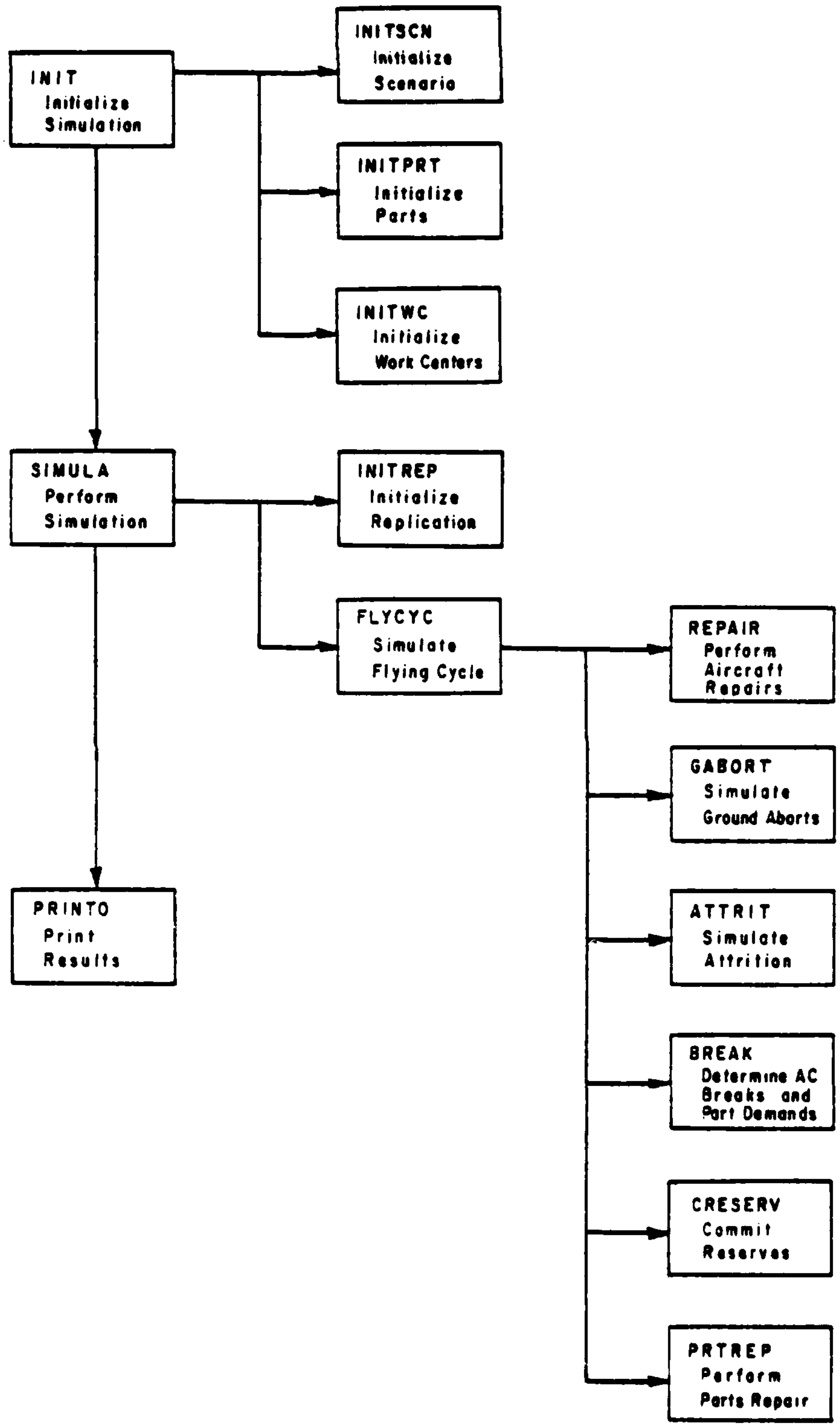


FIGURE 2-1. SGM BLOCK DIAGRAM

INIT

- INITSCN: Initialize Scenario parameters.
- INITWC: Initialize Work Center Data.
- INITPRT: Initialize Spares Data.

FIGURE 2-2. SGM INITIALIZATION

This initialization process is entirely distinct from the remainder of the SGM. Once the initialization process has been completed, INIT and its corresponding subroutines are no longer needed for any other phase of the simulation. Thus, these routines could be overlaid with the remainder of the model routines to conserve memory requirements. Since our typical flying scenarios run adequately within our computer memory restrictions, we do not currently use overlay techniques.

Simulation

The SIMULA routine performs the simulation phase of the SGM. This routine consists of repeated execution of the FLYCYC routine which represents the flying cycle described in Chapter 1. Figure 2-3 provides an outline of the basic structure of these routines. For each replication of the simulation, the specified number of flying days is simulated; a flying day consists of a sequence of identical flying cycles followed by an overnight period before the start of the next day (see Figure 1-1).

The flying cycle simulated in FLYCYC is the basic logical unit of the SGM. As shown in Figure 2-3, the subroutines comprising FLYCYC represent the various processes which cause transitions between aircraft states, e.g.,

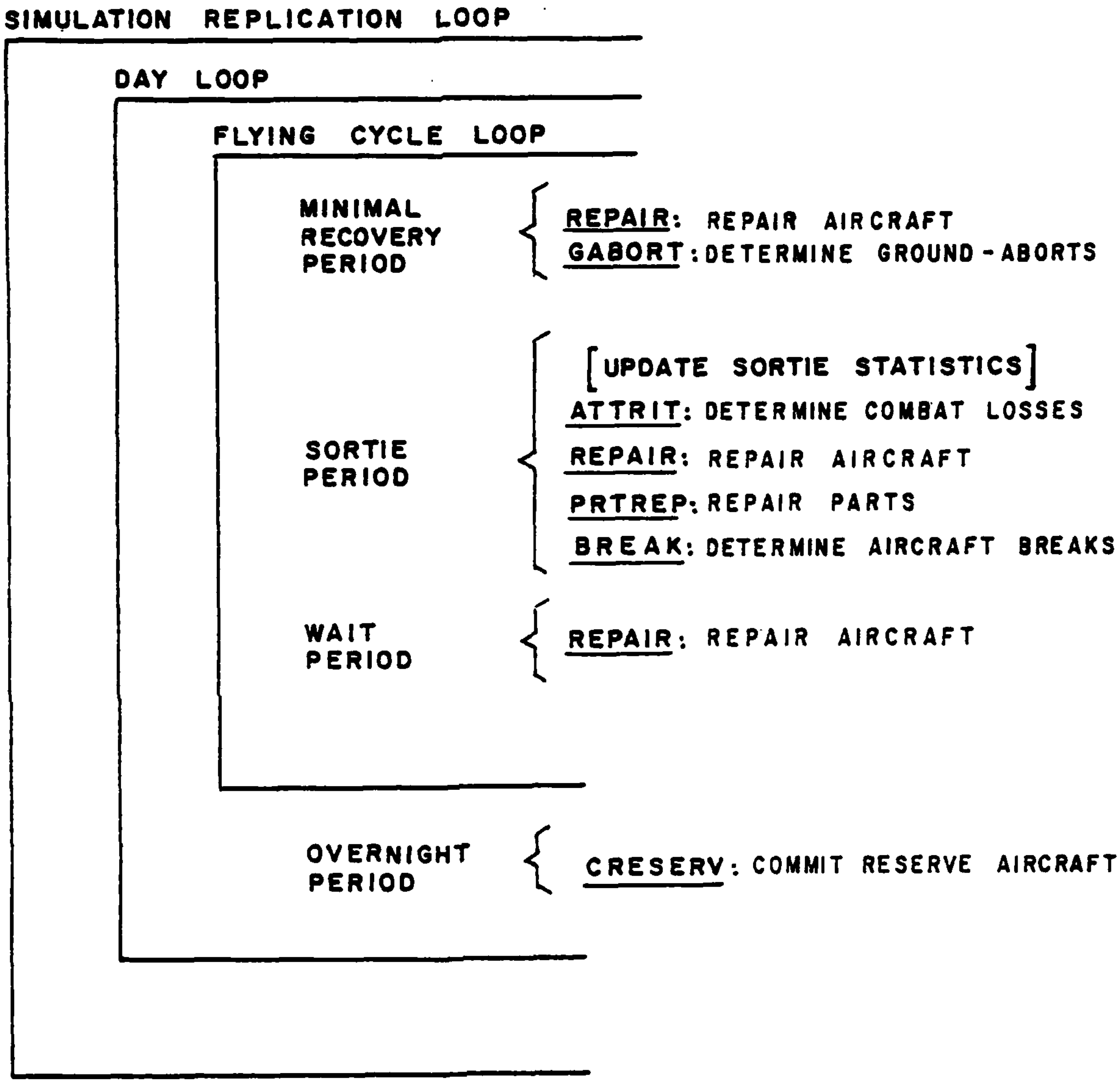


FIGURE 2-3. SIMULATION STRUCTURE

attrition, ground aborts, aircraft repair, etc. This implementation follows very closely the event description of a flying cycle shown in Figure 1-3.

Each flying cycle consists of three periods: a minimal recovery period, sortie period, and wait or overnight period. The results of the various aircraft processes are computed at the start or end of each period, and the number of aircraft in each of the states is updated accordingly. For example, aircraft repair (REPAIR routine) is performed at the end of every period to determine the number of aircraft repairs which have been made during that period. Any repairs would result in transfer of aircraft from the maintenance state to the mission-capable state.

Thus, the SGM is a time-stepped simulation. At the end of each period (periods may be of different lengths), the effects of processes on aircraft states are updated. The implementation of each aircraft process is represented by a major module of the SGM (except the BREAK routine which includes aircraft breaks, part demands and cannibalization). A detailed discussion of the implementation of each process is provided in the documentation for the corresponding module. This documentation is included in the program listings in Appendix C. The following paragraphs describe the interaction of these processes during each of the periods comprising a flying cycle.

<u>Minimal Recovery Period</u>. At the start of this period, the number of flyable aircraft is determined from the number of aircraft in the mission-capable state. The actual number which begin either a preflight or thruflight inspection to prepare for the next sortie is computed as the minimum of flyable aircraft and scheduled sorties.

At the end of the period, the number of aircraft repairs and ground aborts are determined by the REPAIR and GABORT routines respectively. REPAIR computes the number of aircraft repairs which have occurred during this period causing transfer of aircraft from the maintenance state to the mission-capable state. GABORT computes the number of ground aborts among the aircraft preparing for flight, causing the transfer of aircraft from mission-capable to maintenance. Although these routines represent simultaneous points in simulated time, the actual order of execution is important. Aircraft repairs are computed first to ensure that new ground aborts have no chance to be repaired in this period.

Sortie Period. All aircraft scheduled for the flying period, which did not ground-abort, are counted as having flown a sortie, and the various statistics for sortie results are updated. The ATTRIT routine is called next to determine the number of aircraft transferred from the mission-capable state to the combat-losses state. These attritted aircraft are still counted as having flown a sortie.

At the end of this period, the REPAIR, PRTREP, and BREAK routines cause transfer of aircraft between the maintenance, NORS, and mission-capable states. The order of execution here is very important. First, REPAIR computes aircraft repairs during the sortie period. These repaired aircraft are transferred from maintenance to mission-capable. Next PRTREP determines, for each part type, the number of parts resupplied since the last parts-repair calculation. In addition, as the number of backorders for each type is updated, a new number of NORS aircraft is computed assuming maximum cannibalization. Any decrease from the old NORS number is transferred from the NORS state to the mission-capable state. PRTREP is executed after REPAIR to ensure that these new aircraft entering maintenance do not have a chance to be repaired during the sortie period. PRTREP is actually executed only once each day on the last flying cycle of the day. Parts repair is an extremely time-consuming process because of the large number of part types modeled; hence, we approximate the parts repair process by only updating the parts resupplied once every 24 hours.

BREAK computes the number of aircraft breaks at the end of the sortie and determines the part demands resulting from these breaks. Any demands which cannot be filled, either from the on-hand stock or by maximum cannibalization, result in NORS aircraft. The remaining broken aircraft are transferred directly to the maintenance state. Since BREAK may cause new aircraft in maintenance and new parts in resupply, it must be called after REPAIR and PRTREP to ensure that these parts or aircraft are not allowed to be repaired instantaneously by these routines.

Wait Period. Aircraft repairs are determined by the REPAIR routine at the end of this period. These repaired aircraft are transferred to the mission-capable state and are immediately available to fly.

Overnight Period. The last flying cycle of the day has an overnight period instead of a wait period. Again, aircraft repairs are determined at the end of the period by the REPAIR routine. Also, the available reserve aircraft are committed at this point of the flying day by the CRESERV routine. Since the reserves arrive in a fully mission-capable state, the REPAIR computations are unaffected by CRESERV; hence, the order of execution here is unimportant.

Print Results

The final step of each SGM simulation run is performed by the PRINTO routine. It computes the sortie statistics, prepares data for sortie plots, and prints the results of the SGM run. Descriptions and samples of these results and plots are provided in Volume II, SGM User's Guide.

The sortie statistics printed by this routine consist of the average numbers of aircraft in each possible aircraft state. These data are collected at the beginning of each sortie period during each flying cycle throughout the simulation; PRINTO computes the averages and various cumulative totals and prints a sortie profile describing the simulation results for each flying cycle of each flying day.

This routine also creates a temporary data file (file-07) to pass sortie results to the Plot Program. This file contains, for each flying day, the total number of sorties flown and the average number of sorties flown per aircraft. These results are graphed by the Plot Program as part of the SGM output.

As with the initialization routines, the PRINTO routine is entirely distinct from the initialization and simulation portions of the SGM. Once the simulation has been completed those routines could be overlaid with the PRINTO routine.

COMPUTER ENVIRONMENT

The SGM has been developed on System C, an unclassified computer system located at the Pentagon and supported by the Air Force Data Services Center (AFDSC). This system operates on a Honeywell G-635 computer under the series 600/6000 GCOS Time-Sharing System. Access to the system is possible on remote terminals by a dial-up procedure.

The SGM has been written in the Honeywell 600/6000 FORTRAN programming language, the only version of FORTRAN available on the system. The run process has been designed so that the model may be run in either the remote-batch or time-sharing modes. There are advantages and disadvantages to both procedures. If System C is carrying a light load (i.e., only a few users are signed on), then a time sharing run is significantly faster; however, throughout the simulation the terminal cannot be used for any other purpose and it is not possible to direct the output elsewhere. Once a job has been submitted interactively to be run as a batch job, the user is free to make other runs, use the terminal for some other purpose, or even to log-off the computer.

For a detailed description of System C, the Time-Sharing System, and FORTRAN 600/6000, the user is referred to the Honeywell and AFDSC manuals referenced in Appendix A.

SGM ROUTINES

The SGM currently consists of a main program and 39 subroutines, functions, and block data subprograms. Figure 2-4 provides a list and short description of each of these FORTRAN routines. Complete program listings of all routines can be found in Appendix C of this volume.

These routines have been designed to be self-documenting. Extensive comments have been included with each routine to describe the purpose of the routine and define each of its input and output arguments. Definitions are also provided for all common variables referenced or modified by the routine.

All routines are written using Program Design Language (PDL), a software development technique for designing and documenting routines. With PDL, logical steps in the routine are expressed in "structured" English statements. The actual FORTRAN programming language statements are inserted immediately following the corresponding PDL statement. The use of PDL eliminates the need for flow charts; PDL designs are easier to produce, easier to change, and easier to read than flow chart forms. More detailed descriptions of PDL are provided in "PDL - A Tool For Software Design" and Software Documentation and Development Conventions, which are referenced in Appendix A of this volume.

Routines have also been designed using a top down, structured programming approach. Each routine is modular; most are less than one page and none are more than three pages in length.

ROUTINE	DESCRIPTION
MAIN	- MAIN PROGRAM FOR LMI SORTIE-GENERATION MODEL (SGM).
ALIAS	- INITIALIZE TABLES NEEDED FOR "ALIAS" SAMPLING METHOD.
ATTRIT	- SIMULATE ATTRITION PROCESS DURING A SORTIE PERIOD.
BLOCK DATA	- INITIALIZES COMMON TABLES FOR BIT MANIPULATIONS.
BREAK	- SIMULATE AIRCRAFT BREAKS AFTER A SORTIE.
CRESERV	- COMMIT RESERVE AIRCRAFT.
FLYCYC	- SIMULATE AIRCRAFT FLYING CYCLE.
GABORT	- SIMULATE AIRCRAFT GROUND-ABORT PROCESS.
INIT	- INITIALIZE SGM SIMULATION.
INITBO	- INITIALIZE PARTS IN RESUPPLY AT START OF SIMULATION.
INITPRT	- LOAD AND INITIALIZE SPARE-PARTS DATA.
INITREP	- INITIALIZE VARIABLES FOR A SIMULATION REPLICATION.
INITSCN	- READ AND INITIALIZES SCENARIO INPUTS.
INITWC	- LOAD AND INITIALIZE MAINTENANCE WORK CENTER DATA.
IPOISSON	- GENERATE RANDOM SAMPLE FROM A POISSON DISTRIBUTION.
LBITS	- MASK-OFF LEFTMOST 1-BITS IN A COMPUTER WORD.
MAKEPD	- CONVERTS PARTS DEMAND ARRAY INTO A PDF.
MNOM	- GENERATE MULTINOMIAL SAMPLE FOR PART DEMAND TYPE.
MUPDATE	- UPDATE MAINTENANCE AIRCRAFT-STATE BIT-VECTOR.
N1BITS	- COUNT NUMBER OF 1-BITS IN A COMPUTER WORD.
N1VECT	- COUNT NUMBER OF 1-BITS IN A BIT-VECTOR.
NBINOM	- GENERATE RANDOM SAMPLE FROM BINOMIAL DISTRIBUTION.
NDMNDS	- GENERATE SAMPLE OF TOTAL SORTIE PART DEMANDS.
NORSAC	- CALCULATE INITIAL NUMBER OF NORS AIRCRAFT.
NORSBK	- DETERMINES NORS AIRCRAFT FROM A SORTIE.
NREPS	- RANDOM SAMPLE OF AIRCRAFT REPAIRS IN A WORK CENTER.
PRINTO	- PRINT-OUT RESULTS OF THE SIMULATION RUN.
PRTREP	- SIMULATES PROCESS OF REPAIRING PARTS.
PSTAT	- CALCULATES STATISTICS FOR TOTAL PART DEMANDS.
REPAIR	- SIMULATES PROCESS OF WORK CENTER AIRCRAFT REPAIR.
SIMULA	- PERFORM SIMULATION REPLICATIONS.
SORTDS	- DESCENDING SORT OF A REAL ARRAY.
TBITSL	- TRANSFER 1-BITS FROM LEFT OF A BIT-VECTOR.
TBITSR	- TRANSFER 1-BITS FROM RIGHT OF A BIT-VECTOR.
UEUPDAT	- UPDATE UE-STRENGTH FOR SCENARIO.
WCDIST	- DETERMINE BREAK DISTRIBUTION INTO WORK CENTERS.
WCPROB	- INITIALIZE WORK-CENTER SEQUENTIAL BREAK PROBABILITIES.
WCREAD	- READ AND INITIALIZE WORK CENTER DATA.
XNORM	- DRAW RANDOM SAMPLE FROM A NORMAL DISTRIBUTION.
ZBITSL	- ZERO-OUT 1-BITS IN LEFTMOST PORTION OF A WORD.

FIGURE 2-4. SGM ROUTINES

SYSTEM ROUTINES

The SGM uses a number of system-supplied utility routines. A list of all such routines is shown in Figure 2-5, and the remainder of this section provides a description of each routine.

ROUTINE	DESCRIPTION
CONCAT	- MOVE CHARACTER STRING.
FCLOSE	- CLOSE-OUT A FILE.
MEMSIZ	- DETERMINE ALLOCATED MEMORY SIZE.
PTIME	- DETERMINE CPU PROCESSING TIME.
SPRAY	- INITIALIZE AN ARRAY WITH A CONSTANT.
UNIFM1	- GENERATE RANDOM NUMBER.
ZERO	- ZERO AN ARRAY.

FIGURE 2-5. SYSTEM UTILITY ROUTINES

CONCAT - Call CONCAT (A, N, B, M, L)

CONCAT is used to move a character substring of arbitrary length and position within a string. A is the string to be replaced and N is the initial character of A; characters are numbered, left to right, 1, 2, ...; B is the replacement string and M is the initial character of B; L is the number of characters to be replaced. This call causes the Nth through (N + L - 1)th character of A to be replaced with the Mth through (M + L - 1)th character of B.

FCLOSE - Call FCLOSE(U)

FCLOSE closes a file and releases the buffer assigned to that file. U is the logical file number of the file to be closed.

MEMSIZ - Call MEMSIZ(J)

This routine provides the capability of obtaining allocated memory. J, the return value of this call, is the number of 1024-word blocks currently allocated to this job.

PTIME - Call PTIME(A)

This routine provides the means of obtaining processor time. A, a real variable, is the processor time in hours.

SPRAY - Call SPRAY (Z, A1, N1, ... An, Nn)

SPRAY will place the value Z (real or integer constant or variable) into each of N1 consecutive locations starting at the first location in array A1. Argument pairs are limited to the maximum number of continuation cards. "A1" may be a real or integer array, but N1 must be an integer constant or variable.

UNIFM1 - R=UNIFM1(SEED)

UNIFM1 is a FORTRAN-compatible, assembly-language routine for calculating random numbers having a uniform (rectangular) distribution on the unit interval. The starting number SEED is used to initialize the calculations of the random number. Subsequent calls use the previously calculated random number in place of SEED.

ZERO - Call ZERO(A1, N1, ..., An, Nn)

ZERO will place a zero in each of N1 consecutive locations starting at the first location in array A1. Argument pairs are limited only by the maximum number of continuation cards permitted by the FORTRAN IV compiler. "A1" may be a real or integer array, but N1 must be an integer constant or variable.

3. COMMONS

INTRODUCTION

The SGM uses FORTRAN common blocks to pass values between subroutines. All key model information is stored in these commons and most of the memory requirements for an SGM run are determined by the size of the common arrays. As shown in Figure 3-1, the associated common variables and arrays are grouped logically into twelve labeled common blocks. The purposes of this chapter are to describe the various programming conventions followed in the use of these commons, to describe the parameter values which set the size of the common arrays, and finally to define each common variable and array.

PROGRAMMING CONVENTIONS

The following paragraphs describe the various programming conventions followed in the use of labeled commons for the SGM.

All common arrays are dimensioned using FORTRAN parameter values to provide flexibility in configuring the model for different scenarios. These parameters are described in detail in the next subsection.

Storage for all arrays used in the SGM is maintained in labeled common blocks; this allows the user to determine major core requirements by examining the dimensions of the arrays in these twelve common blocks.

A variable or array name is always the same in each occurrence of a common block. A common block appears in an SGM subroutine only if some variable or array in that block is referenced or modified by that subroutine. Figure 3-2 indicates the location of the common blocks throughout the SGM. For each common block it provides a list of those SGM routines which either reference or modify a variable in that common block. A routine is considered to reference a variable if it uses that variable but does not change its value

```
/ACSTATE/  -  AIRCRAFT BIT-VECTORS.
  COMMON /ACSTATE/ LENGTH, NACVC(MAXVEC), IFLYVC(MAXVEC),
                   MAINVC(MAXVEC), NORSVC(MAXVEC), LOSTVC(MAXVEC)

/ALIASC/   -  TABLES FOR PART-TYPE SAMPLING.
  COMMON /ALIASC/  FRACT(MAXPRT), IALIAS(MAXPRT), FPARTS

/BITS/     -  BIT MANIPULATION TABLES.
  COMMON /BITS/    MASKO,MASK(35), MLEFTO,MSKLFT(36),
                   IZCOUT,ICOUNT(63)

/DEMAND/   -  MEAN AND VARIANCE FOR TOTAL PART DEMANDS.
  COMMON /DEMAND/  ACMEAN, ACVAR, NPERAC

/INPUT/    -  FLYING SCENARIO PARAMETERS.
  COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
                   MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT

/PARTS/    -  PART CHARACTERISTICS.
  COMMON /PARTS/   NPARTS, IQPA(MAXPRT), NBACKO(MAXPRT),
                   BRPRATE(MAXPRT), DRPRATE(MAXPRT), INITSJ(MAXPRT),
                   RESUPP(MAXPRT), BNRTS(MAXPRT), NBASE(MAXPRT),
                   NDEPOT(MAXPRT)

/RSEED/    -  SEED FOR RANDOM NUMBER GENERATOR.
  COMMON /RSEED/   SEED

/STATS/    -  CUMULATIVE STATISTICS FOR SIMULATION RESULTS.
  COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
                   NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT

/TIME/     -  FLYING CYCLE TIMES AND SIMULATION PARAMETERS.
  COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
                   NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE

/WCBRK/    -  WORK CENTER BREAK RATES.
  COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
                   PBRKSEQ(2,MAXWC), INDXWC(MAXWC)

/WCINPUT/  -  WORK CENTER INPUTS.
  COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)

/WCMAINT/  -  AIRCRAFT WORK CENTER LISTS.
  COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
```

FIGURE 3-1. SGM COMMONS

COMMON BLOCK	MODIFYING-(M) AND REFERENCING-(R) ROUTINES	
/ACSTATE/	FLYCYC	(M)
	INITREP	(M)
	N1VECT	(R)
	TBITSL	(R)
	TBITSR	(R)
	UEUPDAT	(M)
	ZBITSL	(R)
/ALIASC/	INITPRT	(M)
	MNOM	(R)
/BITS/	BLOCK DATA	(M)
	LBITS	(R)
	MUPDATE	(R)
	N1BITS	(R)
	N1VECT	(R)
	UEUPDAT	(R)
/DEMAND/	INITPRT	(M)
	NDMNDS	(R)
/INPUT/	INIT	(M)
	INITREP	(M)
	INITSCN	(M)
	PRINTO	(R)
	SIMULA	(M)
/PARTS/	INITBO	(M)
	INITPRT	(M)
	INITREP	(M)
	NORSBK	(M)
	PRTREP	(M)
/RSEED/	INITBO	(M)
	INITSCN	(M)
	NBINOM	(M)
	NDMNDS	(M)
	NREPS	(M)
	WCDIST	(M)
/STATS/	FLYCYC	(M)
	INIT	(M)
	INITREP	(M)
	PRINTO	(R)
	SIMULA	(M)

FIGURE 3-2. SGM COMMON REFERENCES

/TIME/	FLYCYC	(R)
	INIT	(R)
	INITSCN	(M)
	PRINTO	(R)
	SIMULA	(M)
/WCBRK/	FLYCYC	(R)
	INIT	(R)
	INITREP	(R)
	INITSCN	(M)
	INITPRT	(M)
	SIMULA	(R)
/WCINPUT/	FLYCYC	(R)
	INITREP	(R)
	INITWC	(M)
	PRINTO	(R)
	WCDIST	(R)
/WCMAINT/	INITREP	(M)
	MUPDATE	(R)
	REPAIR	(M)
	WCDIST	(M)

(R) - A ROUTINE IS CONSIDERED TO REFERENCE A COMMON BLOCK IF IT USES A VARIABLE IN THAT BLOCK, BUT DOES NOT CHANGE ITS VALUE.

(M) - A ROUTINE IS CONSIDERED TO MODIFY A COMMON BLOCK IF IT CHANGES THE VALUE OF A VARIABLE IN THAT BLOCK.

FIGURE 3-2. SGM COMMON REFERENCES (CONT'D)

and is considered to modify a common block if it changes the value of a variable in the block.

All common variables are of type integer or real except for INFPART and INFMAN which are logical variables. The type of these other common variables is determined implicitly by the name, following standard FORTRAN conventions. If the first character of a variable name begins with any of the characters between I and N, it is an integer variable. If the first character is any other alphabetic character, it is a real variable.

A nonstandard FORTRAN technique for referencing the 0th word of an array is used in several common arrays. An extra word is placed before the beginning of the array. The use of ARRAY(0) actually references this extra word before ARRAY(1). Thus, the array is, in effect, indexed 0, 1, 2, . . . instead of 1, 2, . . . This technique may not work with other FORTRAN compilers.

PARAMETERS

The dimensions of all SGM common arrays are controlled with FORTRAN parameter values. Figure 3-3 provides a list and description of all SGM parameter values. These parameter values allow maximum flexibility in configuring the SGM to handle different flying scenarios. For example, if the user desires to increase the number of work center types the SGM can handle, the MAXWC parameter must be increased in all routines containing this parameter. This change can be made using a single command with the system text editor. If 30 work centers were needed rather than the current maximum of 25, the user would just change all occurrences of the character string "MAXWC=25" to "MAXWC=30" throughout the SGM source code, recompile the program, and run it. These parameters also allow the user to minimize the core requirements for any particular flying scenario.

The only limitation on increasing these various parameter values is that the overall system core limitations on programs must not be exceeded. The SGM as currently configured requires approximately 20K words of core. This configuration allows a maximum of 108 aircraft (MAXAC=108), 25 work centers (MAXWC=25), 304 LRU types (MAXPRT=304), and 30 flying days (MAXDAY=30).

The values of these particular parameters, MAXWC, MAXPRT, and MAXDAY, determine the major computer memory requirements of the model.

MAXAC	- MAXIMUM ALLOWABLE UE-STRENGTH (# AIRCRAFT)
MAXWC	- MAXIMUM ALLOWABLE NUMBER OF WORK CENTERS
MAXBIT	- NUMBER OF BITS IN A COMPUTER WORD ON THIS SYSTEM
MAXPRT	- MAXIMUM ALLOWABLE NUMBER OF PART-TYPES
MAXVEC	- MAXIMUM ALLOWABLE LENGTH (IN COMPUTER WORDS) OF AIRCRAFT BIT-VECTORS A BIT-VECTOR MUST BE AT LEAST "MAXAC" BITS LONG, PLUS AN EXTRA WORD TO STORE THE AIRCRAFT COUNT FOR THAT VECTOR HENCE, MAXVEC IS A FUNCTION OF MAXAC AND MAXBIT
MAXDAY	- MAXIMUM ALLOWABLE NUMBER OF FLYING DAYS
MAXCYC	- MAXIMUM ALLOWABLE NUMBER OF FLYING CYCLES PER DAY
MAXSTAT	- CURRENT NUMBER OF STATISTICS COLLECTED PER FLYING CYCLE PER DAY
LFLD	- LENGTH OF BIT-FIELD USED IN THE WORK-CENTER REPAIR LISTS THIS BIT-FIELD MUST BE LARGE ENOUGH TO HOLD (MAXAC-1), THE TAIL NUMBER OF THE LAST AIRCRAFT THUS, (2**LFLD) MUST BE GREATER THAN OR EQUAL TO MAXAC
NPERWRD	- NUMBER OF BIT-FIELDS PER COMPUTER WORD FOR THESE WORK-CENTER LISTSTHUS NPERWRD IS A FUNCTION OF LFLD AND MAXBIT
MXINWC	- LENGTH (IN COMPUTER WORDS) OF THE WORK-CENTER LISTS MXINWC IS COMPUTED SO THAT THE MAXIMUM ALLOWABLE NUMBER OF BIT FIELDS IN A WORK-CENTER LIST IS EQUAL TO MAXAC, THE MAXIMUM NUMBER OF AIRCRAFT
IFSCEN	- FILE NUMBER OF SCENARIO INPUT FILE
IFWC	- FILE NUMBER OF WORK CENTER INPUT FILE
IFPRT	- FILE NUMBER OF SPARES INPUT FILE

FIGURE 3-3. SGM PARAMETERS

COMMON DESCRIPTIONS

This section provides a description of each labeled common block used to pass values in the Sortie-Generation Model. Each subsection describes the purpose of the block followed by a definition of each associated variable or array. The commons are presented in alphabetical order by name.

/ACSTATE/ - Aircraft Bit-Vectors

This block contains the various aircraft-status bit-vectors. These vectors describe the state of the simulation at any point in time. Each bit in these vectors represents a unique aircraft, and an aircraft is marked as being in a particular state by setting the corresponding bit in that bit-vector to 1.

- LENGTH: length, in computer words, of the various aircraft-status bit-vectors. LENGTH is equal to the number of computer words necessary to hold a number of bits equal to NAC, the current UE-strength (or number of aircraft).

- NACVC(I): I=1, 2, . . ., LENGTH. This is a bit-vector with the first NAC bits set to 1, where NAC is the current number of aircraft. This vector represents the set of all possible aircraft and is used to initialize the mission-capable bit-vector, IFLYVC.

- IFLYVC(I): I=1, 2, . . ., LENGTH. Aircraft-status bit-vector indicating aircraft currently in the mission-capable state. A 1-bit indicates the corresponding aircraft is mission-capable and a 0-bit indicates not-mission-capable.

- MAINVC(I): I=1, 2, . . ., LENGTH. Aircraft-status bit-vector indicating aircraft currently undergoing maintenance in at least one work center. A 1-bit indicates the corresponding aircraft is in maintenance.

- NORSVC(I): I=1, 2, . . ., LENGTH. Aircraft-status bit-vector indicating aircraft currently NORS, i.e., waiting for some part. A 1-bit indicates the corresponding aircraft is NORS.

- LOSTVC(I): I=1, 2, . . ., LENGTH. Aircraft-status bit-vector indicating aircraft lost due to attrition. A 1-bit indicates that the corresponding aircraft is a combat loss.

/ALIASC/ - Tables For Part-Type Sampling

This block contains the tables needed for the Alias method of sampling from a discrete probability distribution. This method is used to determine the type of a given broken part. These tables are initialized by the ALIAS subroutine and remain fixed throughout the simulation.

- FRACT(I): I=1, 2, . . ., MAXPRT. This array is used initially to load the demands-per-flying-hour of the Ith part type in the INITPRT subroutine. These values are then modified by the MAKEPD subroutine to convert the demand values to a discrete probability distribution. Finally, the ALIAS subroutine converts these probabilities to fractional cutoff values for the Alias sampling method. The values remain fixed for the remainder of the simulation.

- IALIAS(I): I=1, 2, . . ., MAXPRT. Table of aliases needed for the Alias sampling method. This array is initialized in the ALIAS subroutine and remains fixed thereafter.

- FPARTS: floating-point value of the number of part-types being modeled, i.e., FPARTS=FLOAT(NPARTS). This variable is used to speed-up the sampling procedure: Rather than converting NPARTS to a real number each time the MNOM subroutine is called, the converted number is stored in this block once and used from then on.

/BITS/ - Bit Manipulation Tables

This block contains three sets of tables used for accessing bits and bit fields within a computer word. Note that the following programming technique is used in each of these tables: An extra word is placed before the beginning of each table. This extra word represents the 0th indexed word in the table. Thus, the table is actually indexed 0, 1, 2, ... This technique of referencing the 0th word of an array is not standard FORTRAN and may not work with other FORTRAN compilers. These tables remain fixed throughout the simulation.

- MASK(I): I=0, 1, ...,35. MASK is the bit accessing table used in the SGM. The bits in the computer word are numbered, left to right, 0, 1, 2, ...,35, and MASK(I) has a 1-bit in the Ith position and zeroes elsewhere. This table is used to mask-off the Ith bit in a computer word.

- MASKLFT(I): I=0, 1, ...,36. MSKLFT is used to mask-off the leftmost bits in a computer word. The first (leftmost) I bits of MSKLFT(I) are 1-bits and the remaining bits are zero. Thus, for example, MSKLFT(0) would be all 0s and MSKLFT(36) would be all 1s.

- ICOUNT(I): I=0, 1, ...,63. This is a table which is used to count the number of 1-bits in any given 6-bit field. In a 6-bit field, there are $2^6 = 64$ possible bit patterns -- the binary representations of the integers 0, 1, 2, ...,63. ICOUNT(I) contains the number of 1-bits in the binary representation of I, e.g., ICOUNT(3)=2. This table is used in counting the number of 1-bits representing aircraft in the various aircraft-status bit-vectors. This technique is much faster than a bit-by-bit count.

/DEMAND/ - Mean And Variance For Total Part Demands

This common block contains the various statistics describing the random variable representing the number of part demands per aircraft, given that the aircraft has broken upon returning from a sortie. These variables are used by the NDMNDS function to generate a random sample of the total number of part demands during a sortie. They are initialized in the PSTAT subroutine and remain fixed throughout the simulation.

- ACMEAN: expected value of the random variable described above.

- ACVAR: variance of total part demands per broken aircraft.

- NPERAC: Total number of installed parts per aircraft. This variable is used to ensure that a legitimate sample is generated for the total number of part demands during a sortie period. NPERAC is equal to the sum of the QPAs (quantity-per-aircraft) of all part types modeled in this simulation run.

/INPUT/ - Flying Scenario Parameters

This block contains the various user-specified parameters describing the flying scenario to be simulated. The parameters are initially set in the INITSCN subroutine; however, some of these values are reset at the start of each flying day of the simulation.

- INITUE: Initial UE-strength at the start of the simulation.

- NAC: Current UE-strength. If reserves are used only as attrition fillers, then NAC is always equal to INITUE; however, if reserves are assigned as they become available, then NAC may be greater than INITUE.

- PATTRIT: Probability that an aircraft does not return from a sortie due to combat attrition. This rate may be different for each day of the scenario.

- IRES: The number of aircraft in reserve that are available to augment the current UE of the scenario. As described previously, the user may specify whether these reserves are committed on the day they become available or are to be used only as attrition fillers to replace combat losses. The number of reserve aircraft arriving on the scene may be specified for each day of the scenario.

- RNMCM: Proportion of the possessed aircraft that are not-mission-capable-maintenance at the start of the flying scenario. For example, if the user specifies an initial NMCM rate of 0.3 with a UE of 72, the SGM will begin each simulation experiment with 22 aircraft undergoing maintenance. The remaining 50 aircraft will initially be either mission-capable or waiting for a recoverable spare part.

- INFPART: Logical variable indicating whether the infinite part assumption holds. If INFPART is true then there is never any shortage of parts; hence, no NORS aircraft.

- MAXFLY(I): I=1, 2,..., NCYCLES. Maximum number of aircraft to be scheduled on the Ith wave of this flying day. These values may also be different for each day of the scenario.

- INFMAN: Logical variable indicating whether infinite manpower is assumed for all work centers. If INFMAN is TRUE then the number of crews or servers for each work center is set equal to the maximum allowable number of aircraft.

- ISCALE: Parameter to set the maximum vertical scale on the sorties-per-day plot of the SGM results. For example, if the user wanted plots of a series of SGM runs, he would use this parameter to ensure that all the plots are on the same scale. If 0 is input, the scale is determined from the maximum sorties per day that occur in the SGM results.

- IAUGMNT: A variable, 0 or 1, indicating how reserve aircraft are to be committed. If IAUGMNT=1, all reserve aircraft are committed on the day they become available; if IAUGMNT=0, the reserves are used only as attrition fillers to replace combat losses.

/PARTS/ - Part Characteristics

This block contains the characteristics of the various part types being modeled.

- NPARTS: number of part types being modeled.
- IQPA(I): I=1, 2, ..., NPARTS. QPA (Quantity-Per-Aircraft) of Ith type.
- NBACKO(I): I=1, 2, ..., NPARTS. Number of backorders of Ith type. NBACKO(I) is defined as the number of parts in resupply minus the initial stock level. Thus, NBACKO(I) may be negative.
- BRPRATE(I): I=1, 2, ..., NPARTS. Base repair rate (in parts per day) for the Ith type. The base repair rate is defined as the inverse of the average base repair time.
- DRPRATE(I): I=1, 2, ..., NPARTS. Depot resupply rate (in parts per day) for the Ith type. The depot resupply rate is defined as the inverse of the average depot resupply time.
- INITSJ(I): I=1, 2,..., NPARTS. Initial stock level for Ith type.
- RESUPP(I): I=1, 2,...,NPARTS. Expected number of type-I parts in resupply at the start of the scenario. It is used as the mean of a Poisson distribution in generating a starting number in resupply for each simulation replication.
- BNRTS(I): I=1, 2, ...,NPARTS. Percentage of type-I demands which are not base repairable. A demand which is not base repairable may be condemned or repaired at the depot but in either case an order will be placed for depot resupply.
- NBASE(I): I=1, 2, ...,NPARTS. Number of type-I parts currently in base repair.
- NDEPOT(I): I=1, 2, ...,NPARTS. Number of type-I parts currently on-order from the depot.

/RSEED/ - Seed For Random Number Generator

This block contains the current seed for the random number generator used by this simulation. The seed is updated each time a random draw is made throughout this simulation. The seed is initialized in the INITSCN subroutine with an initial user-specified seed.

/STATS/ - Cumulative Statistics For Simulation Results

This block contains the various statistics produced by the simulation. These statistics consist of the average number of aircraft in the various states at the start of each sortie period for each flying day.

- EXPECT(I,J,K): Cumulative statistics array.
- NRESRV: Current number of aircraft in the reserve pool.
- IZDAY: Defined as ITOTRES(0); see below.
- ITOTRES(I): I=0,...,NUMDAY. Cumulative number of available reserve aircraft up to and including the Ith day. The 0th day represents the initial number of reserve aircraft. This array is used in computing on-the-scene aircraft for sorties/aircraft/day in the PRINTO routine.

/TIME/ - Flying Cycle Times and Simulation Parameters

This block contains the various times describing a flying cycle and also the dimensions of the simulations.

- PREFLITE: The minimal required time (in hours) between the landing of the aircraft and takeoff for the next sortie, provided that no corrective maintenance is required. It includes only the time required to taxi, park, chock, shut down, refuel, rearm, inspect, and launch.
- SORTLGTH: Fixed length of each sortie, in hours.
- WAITCYC: Number of hours between end of a sortie period during the day and the start of the minimal recovery period for the next sortie.
- TYMNITE: Number of hours between end of the last sortie period of the day and the start of the minimal recovery period for the first sortie of the next day.
- NSIM: Number of simulation replications to be performed.
- ISIM: Number designating current simulation replication. ISIM=1, 2, ..., NSIM.
- NUMDAY: Number of flying days to be simulated.
- IDAY: Number designating current day of the simulation. IDAY=1, 2,..., NUMDAY.
- NCYCLES: Number of flying cycles for the current flying day of the simulation.

- ICYCLE: Number designating the current flying cycle. ICYCLE=1, 2,..., NCYCLES.

/WCINPUT/ - Work Center Inputs

This common block contains the essential information from the maintenance manpower input file. The information is initialized in the INITWC subroutine and is never modified for the remainder of the program. It provides the basic information needed to simulate aircraft repair in the various work centers.

- NWC: Number of work centers being modeled. The work centers are numbered 1, 2,..., NWC.
- NCREWS(I): I=1,..., NWC. The number of servers or crews in the Ith work center. If infinite manpower is assumed, then NCREWS(I) is initialized to MAXAC, the maximum possible number of aircraft.
- SRATE(I): I=1,..., NWC. The service rate (in aircraft per hour) for the crews in the Ith work center.

/WCMAINT/ - Aircraft Work Center Lists

This block contains the list of aircraft currently undergoing maintenance in each work center. These lists are zeroed-out at the beginning of each simulation replication and aircraft are added and deleted from the lists as they break and are repaired. The length of these arrays is set as a function of the parameter values for LFLD, MAXAC, MAXBIT, MAXWC, NPERWRD, and MXINWC. This function is described in the definition of LISTRP below.

- INREPR(J): J=1,..., NWC. Number of aircraft currently undergoing maintenance in the Jth work center. Also indicates the number of aircraft tail numbers contained in the work center list, LISTRP(.,J).
- LISTRP(I,J): LISTRP(.,J) is a list of aircraft numbers indicating those aircraft requiring maintenance in the Jth work center (J=1, 2,..., NWC). This list contains exactly INREPR(J) aircraft numbers. To save space, these lists have been packed into bit-fields "LFLD" bits wide; hence, if "MAXBIT" is the length of a computer word on this system, then there are (MAXBIT/LFLD) bit-fields stored per word. The aircraft numbers stored in these bit-fields indicate a unique bit position in the various aircraft-status bit-vectors. The aircraft are numbered, left-to-right, 0, 1, 2,..., (MAXAC-1), where MAXAC is the maximum

possible number of aircraft. To get the Ith aircraft number in a work center list, the corresponding bit position and word index must be computed.

/WCBRK/ - Work Center Break Rates

This block contains the various break probabilities associated with maintenance and work center repair. These probabilities are initialized and remain fixed throughout the simulation.

- PACBRK: Probability that an aircraft returning from a sortie requires unscheduled maintenance in one or more work centers prior to further flight.

- PACGABT: Probability that an aircraft undergoes some failure immediately before takeoff, requiring unscheduled maintenance which renders it not-mission-capable. This rate may be different for each day of the scenario.

- PBRKWC(I): I=1,..., NWC. Probability that an aircraft returning from a sortie breaks into work center I. This array is a direct input from the maintenance manpower input file.

- PWCPROD: Product-formula overall work center break rate. This probability is computed from the individual work center break rates.

- PBRKSEQ(1,J); Probability that an aircraft breaks into the work center indicated by "INDXWC(J)" and does not break into any of the work centers -- INDXWC(J+1), INDXWC(J+2),..., INDXWC(NWC), given that the aircraft has broken into at least one of the work centers -- INDXWC(J), INDXWC(J+1),..., INDXWC(NWC). This implies that PBRKSEQ(1,NWC) must always equal 1.0.

- PBRKSEQ(2,J): Probability that an aircraft has broken into the work center indicated by 'INDXWC(J)', given that the aircraft has broken into at least one of the work centers indicated by -- INDXWC(J), INDXWC(J+1), ..., INDXWC(NWC).

- INDXWC(J): Indicates the index of the work center with the Jth largest break probability. Thus, INDXWC(1) indicates the work center with the largest break probability, and INDXWC(NWC) indicates the one with the smallest.

4. DATA FILES

INTRODUCTION

The Sortie-Generation Model (SGM) uses a variety of mass storage files for input data, temporary parameter storage, and output results. This chapter provides a general overview of the data flows in the SGM, followed by a brief description of each file used or generated by the SGM.

DATA FLOWS

The mass storage files used by the SGM are divided into three categories: input files, temporary scratch files, and output files. A flow chart of the various data files within the SGM is provided in Figure 4-1.

The SGM input files are produced by three different programs. The scenario input parameters are produced by the Set-Parameter Program in which the user interactively specifies the desired values for the simulation. The definitions of these scenario parameters are described in Volume II, SGM User's Guide. The work center and spares inputs are produced by the SGM Maintenance and Spares Subsystems, with detailed descriptions of these inputs provided in Volumes V and VI, respectively.

The SGM also uses two temporary files to conserve memory and provide sortie results to the Plot Program. The Varying Scenario Parameters File (File-03) is used to store the scenario parameters that are allowed to vary for each day of the flying scenario (e.g., attrition rate, available reserves). This file is initialized by the SGM at the start of the simulation, and the entire file is read for each replication of the simulation.

The Plot Data File (File-07) is the other temporary file produced by the SGM; sortie results are written to this file to be used as input to the Plot Program.

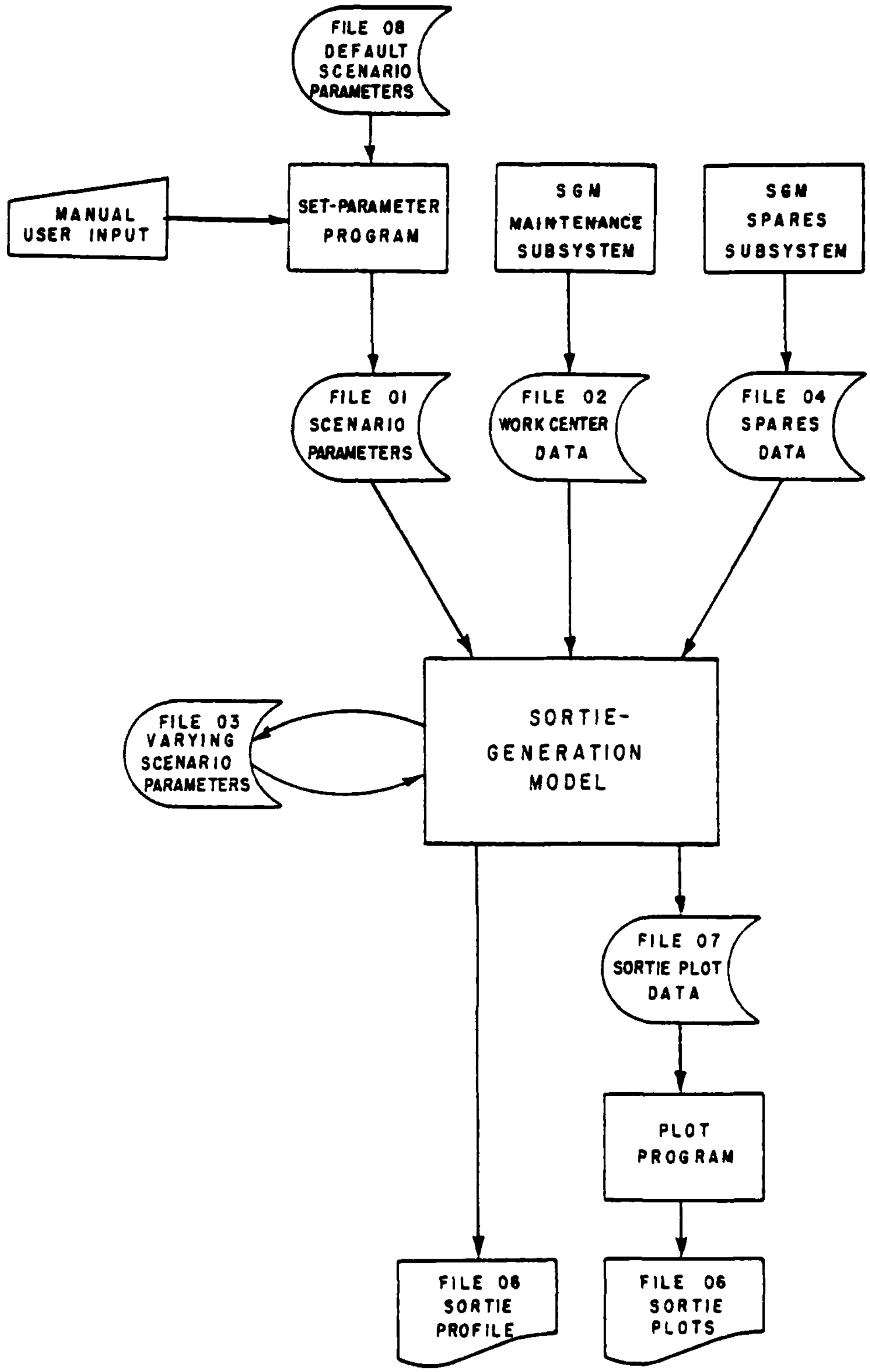


FIGURE 4-1. SGM DATA FILES

All output results are written to the File-06, the standard output print file for this computer system. The SGM prints a scenario summary and sortie profile, and the Plot Program prints two graphs of sortie results. Descriptions of these outputs are provided in Volume II, SGM User's Guide.

FILE DESCRIPTIONS

This section provides a brief description of each data file used or created by the SGM. Each file description includes the following information:

- Purpose of the file
- File format (e.g., sequential or random, permanent or temporary, media type, and approximate length). On this computer system, files are classified according to media type as shown in Figure 4-2. Also, files are measured in units of llinks, where 1 llink is approximately 320 computer words in length.
- Source of data file (i.e., routine or program which created this file)
- File destination (i.e., routine or program which reads information from this file)
- Updating instructions for file maintenance as appropriate

Media Code		File Type
0	-	Print-line image with no slew (BCD)
1	-	Binary record (e.g., FORTRAN binary record, COMDK etc.)
2	-	Hollerith card image (BCD)
3	-	Print-line image (BCD)
5	-	TSS ASCII file format
6	-	ASCII Standard System Format

FIGURE 4-2. SERIES 6000 FORTRAN MEDIA CODES

The SGM currently uses seven data files and the descriptions are listed according to the file unit number (note that unit -05 is not assigned to any file).

Scenario Parameters (FILE-01)

- Purpose: Contains the user-specified scenario parameters (e.g., attrition rates, sortie length) for an SGM run. Media code 0 was chosen for this file to allow the file to be read by a program operating in either batch or time-sharing mode. When the SGM is run in batch mode, a SELECTD control card must be used to request this file. This ensures that subsequent Set-Parameter runs do not alter this particular file before it is actually used in this SGM run.

- Format: Sequential linked permanent file, media 0, 1 llink in length.

- Source: An output file of the interactive Set-Parameter Program.

- Destination: Read by the SGM routine INITSCN.

- Updating: New scenario file is usually created for each SGM run.

Work Center Data (FILE-02)

- Purpose: Contains the work center data describing the maintenance at some specified base. Detailed information concerning the derivation and updating of this file is contained in Volume V, Maintenance Subsystem.

- Format: Sequential linked permanent file, media 3, 1 llink in length.

- Source: Produced by SGM Maintenance Subsystem.

- Destination: Read by the SGM routine WCREAD.

- Updating: Maintenance manpower files are generated for each base to be modeled. The files must be updated as new maintenance information is received from the appropriate base.

Varying Scenario Parameters (FILE-03)

- Purpose: Temporary file created to store the scenario parameters that are allowed to vary on a daily basis in the simulation. All of these data are also contained in the Scenario Parameter File (File-01); however, the information has been converted to binary format on the scratch file to save processing time in loading the scenario parameter values at the start of each simulation day. This approach eliminates the need for any arrays to store the parameter values for each day.

- Format: Linked temporary file, media 1, 2 llinks in length.

- Source: Initialized by the SGM routine INITSCN.

- Destination: Read by the SGM routine SIMULA.

- Updating: New scratch file automatically created and released with each SGM run.

Spares Data (FILE-04)

- Purpose: Contains the spare parts data for a specified base, aircraft type, and availability level. For detailed information on spares data, see Volume VI, Spares Subsystem.

- Format: Sequential linked permanent file, media 1, about 10 llinks in length (varies by base).

- Source: Produced by the SGM Spares Subsystem.

- Destination: Read by the SGM routine INITPRT.

- Updating: Must be updated periodically to reflect changes in the spares data.

Standard System Output File (FILE-06)

- Purpose: Standard system output file to which all SGM results, graphs, and error messages are written.

- Format: That for the standard system output.

- Source: The SGM PRINTO routine and the Plot Program generate the sortie results and graphs which are sent to this file. Many other SGM routines will send error messages to this file if an error is detected.

- Destination: Output is sent to a user-specified printer which may be either a time sharing terminal or a batch output device.

Sortie Plot Data (FILE-07)

- Purpose: Contains the sorties per aircraft and sorties per day results of an SGM run.

- Format: Linked temporary file, media 1, approximately 1 llink in length.

- Source: An output of the SGM Routine PRINTO.

- Destination: Provides the inputs for the Plot Program.

- Updating: A temporary file which is automatically created and released for each SGM run.

Default Scenario Parameters (FILE-08)

- Purpose: Contains default scenario parameter values for a specified aircraft type. These values are used to initialize the scenario for a Set-Parameter run. This default file allows the user to generate new scenarios from the base scenario with very little work.

- Format: Sequential linked permanent file, media 5, 1 llink in length.

- Source: Created manually by the user using the system text editor.

- Destination: Input to the interactive Set-Parameter Program.

- Updating: The user may update this file as often as desired to reflect new base scenarios.

5. ERROR MESSAGES

INTRODUCTION

In addition to the various computer system error checks, numerous error checks have been programmed into the Sortie-Generation Model (SGM). This chapter provides an explanation of the error messages that may result from these SGM checks. The extensive error detection capabilities of the Honeywell Series 6000 FORTRAN and resulting error messages are described in the Honeywell FORTRAN manuals referenced in Appendix A of this manual.

Figure 5-1 provides a list of all possible SGM error messages. The error checks resulting in these messages have been designed to detect many of the

```
$$$$$$$$ INITBO ERROR   - TOO MANY PARTS IN RESUPPLY
$$$$$$$$ INITPRT ERROR  - INVALID PART CHARACTERISTIC
$$$$$$$$ INITPRT ERROR  - TOO MANY LRU TYPES
$$$$$$$$ INITWC ERROR   - LFLD PARAMETER TOO SMALL
$$$$$$$$ IPOISSON ERROR - NEGATIVE MEAN
$$$$$$$$ LBITS ERROR    - TOO FEW 1-BITS TO MASK
$$$$$$$$ TBITSL ERROR   - TOO FEW 1-BITS TO TRANSFER
$$$$$$$$ TBITSR ERROR   - TOO FEW 1-BITS TO TRANSFER
$$$$$$$$ UEUPDAT ERROR  - UE OVERFLOW
$$$$$$$$ WCDIST ERROR   - SEQUENTIAL SAMPLING ERROR
$$$$$$$$ WCDIST ERROR   - INCONSISTENT BROKEN AIRCRAFT
$$$$$$$$ WCREAD ERROR   - TOO MANY WORK-CENTERS
$$$$$$$$ WCREAD ERROR   - INVALID WORK CENTER DATA
$$$$$$$$ XNORM ERROR    - NEGATIVE STANDARD DEVIATION
$$$$$$$$ ZBITSL ERROR   - NOT ENOUGH 1S TO ZERO
```

FIGURE 5-1. SGM ERROR MESSAGES

typical errors resulting from improperly formatted input files, inconsistent flying scenarios, or newly introduced routines which may not be completely bug-free. In many instances a tradeoff has been made in increasing the reliability of the SGM at the expense of computational speed; however, these checks have proved invaluable in detecting subtle logical design errors in the development of the model.

These SGM error checks have been designed to allow continued execution. An error message is printed, a reasonable patch is made, and control is returned to the calling routine. The purpose of this design decision to continue execution of the model, even though the results may no longer be valid, was to provide as much debug information as possible from each SGM run. Thus, this was with the hope that any additional independent errors might also be detected in the same SGM run. However, some errors may propagate further errors, leading to numerous error messages and even fatal system errors. This design decision also requires the user to check each SGM run carefully to ensure that no error messages have been printed, even though the run may have terminated successfully with reasonable results. All SGM error messages begin with the characters "$$$$$$$$", and the messages will always appear before the SGM sortie profile. Hence, this area should be carefully scanned after each run.

STANDARD FORMAT

All SGM error messages follow the standard format shown in Figure 5-2. The beginning of each message is double-spaced from the line preceding it, and lines containing any error message always begin with the characters "$$$$$$$$". The first words in each message identify the model routine which detected the error and generated the message. For example, in Figure 5-2, the sample message indicates that an error was detected in the WCREAD routine.

The remainder of the first line provides a short description of the particular error detected. Again, in our example, the description, "TOO MANY WORK-CENTERS IN THE INPUT FILE", indicates that the work center input file contains too many work center types for the current SGM configuration.

STANDARD SGM ERROR FORMAT

```
$$$$$$$$     "Routine name" ERROR - "error description"
$$$$$$$$          "additional variable values"
```

EXAMPLE

```
$$$$$$$$     WCREAD ERROR - TOO MANY WORK-CENTERS IN THE INPUT FILE
$$$$$$$$          ONLY THE FIRST  25 WORK-CENTERS WERE USED
$$$$$$$$          INCREASE -MAXWC- PARAMETER IF YOU WANT MORE WC-S
```

FIGURE 5-2. STANDARD ERROR MESSAGE FORMAT

The remaining lines of the error message provide values of various subroutine variables related to the source of the error. The error messages have been designed to print the values of all variables which may aid in determining either the exact cause of the error or the needed fix. The Figure 5-2 example provides the current value of the MAXWC parameter which sets the size of the various work center arrays. This parameter must be increased to handle additional work centers.

MESSAGE DESCRIPTIONS

This section provides a detailed description of each SGM error message. Each subsection provides a description of the error which caused the message, the possible causes of the error, suggested actions to correct the problem, and an explanation of the variable values printed with the message to aid in the debugging process.

INITBO Error - Too Many Parts in Resupply

This message indicates that a recoverable part type has more backorders (NBACKO(K)) at the beginning of a simulation replication than there are aircraft on-the-scene (NAC). The maximum number of allowable backorders is equal to the UE-aircraft strength times the QPA (quantity-per-aircraft) for that part-type. The characteristics of this part type are printed with the error message; the number of backorders is truncated at the maximum allowable, and execution of the simulations continues. The results are no longer valid, but continued execution may provide more debug information.

This error is extremely unlikely, but might occur if the spares file is incorrect, and an impossible value has been loaded for this part's initial resupply (RESUPP(K)). Another possibility is that an extremely low value for the number of aircraft (NAC) is being used. The user should examine the spares file carefully to ensure that it has no impossible values and also check to ensure that the flying scenario being used is consistent with this particular spares file.

INITPRT - Invalid Part Characteristic

This message indicates that some part in the spares input file has one or more invalid characteristics (e.g., a negative demand rate or initial stock level). This problem could be caused by a bug in the SPARES subsystem or by a spares-input file with an improper format. The NSN of this part along with its characteristics is printed with the error message to aid in the debug process.

This error is not fatal; execution of the simulation will continue. However, the part with the invalid characteristics will not be included in the simulation run.

INITPRT Error - Too Many LRU Types

This message indicates that the spares input file contains too many part types; the current size of the parameter, MAXPRT, which sets the size of the various part arrays, is too small. However, the model will load the first MAXPRT part types and perform the simulation run with just these types. This may still give valid results since the parts file is sorted in order of parts most likely to cause NORS aircraft. Hence, if only the last few part types were not loaded, the SGM's estimate of sortie-generation capability would probably not be affected.

To obtain a run with all the parts in the input file, the user should determine the number currently in the file, and reset the MAXPRT parameter accordingly. The current value of MAXPRT is printed with this error message.

INITWC - LFLD Parameter Too Small

This message indicates that the "LFLD" parameter has been set too small. This parameter defines the width of the bit-field used for storing aircraft tail numbers. Lists of aircraft tail numbers are used to indicate those aircraft which have broken into the various work centers. Thus, the length of the bit-field must be able to store the largest possible aircraft tail number. The aircraft are numbered 0, 1, 2,..., MAXAC-1 where MAXAC is the parameter indicating the maximum possible number of aircraft. LFLD and MAXAC must be consistent, and the formula to ensure this consistency is given by $MAXAC \leq 2^{LFLD}$. The current values of LFLD and MAXAC are printed with the message. The simulation does not terminate; however, the results are unreliable.

IPOISSON Error - Negative Mean

This message indicates that a negative mean has been passed to the IPOISSON routine. The value of the mean for a Poisson random variable must be a non-negative number. The value of this input mean, RMEAN, is also printed with the error message. The return value is set to zero and execution of the simulation continues; this is not a fatal error.

LBITS Error - Too Few 1-Bits to Mask

This message indicates that the specified number of 1-bits to mask (NBITS) is more than the number (IFOUND) actually contained in the given input word; thus, exactly NEXTRA 1-bits were not masked as requested. This indicates that some subroutine assumes there are more 1-bits in the input word than there actually are. The values of all three pertinent variables, NBITS, IFOUND, and NEXTRA are printed with the error message to aid in the debugging process. This error is not fatal; the subroutine will mask all the 1-bits it found; thus the output word will just be a copy of the input word. However, the results of the simulation are no longer reliable.

TBITSL Error - Too Few 1-Bits to Transfer

This message indicates that the specified number of 1-bits to transfer (NONES) is more than actually contained in the bit-vector from which the transfer is to be made (IFROM(1)); thus, exactly NLEFT 1-bits have not been transferred as requested. This message indicates that some subroutine assumes there are more 1-bits in the bit-vector than there actually are. The values of all three pertinent variables, NONES, IFROM(1), and NLEFT are printed with this error message to aid in the debugging process. This error is not fatal -- the subroutine will transfer all the 1's that are there, zero-out the IFROM vector, and continue execution. The results of the simulation are no longer reliable; however, continued execution may provide additional debug information.

TBITSR Error - Too Few 1-Bits to Transfer

This message indicates that the specified number of 1-bits to transfer (NONES) is more than actually contained in the bit-vector from which the transfer is to be made (IFROM(1)); thus, exactly NLEFT 1-bits have not been transferred as requested. This indicates that some subroutine thinks there are more 1-bits in the bit-vector than there actually are. The values of all three pertinent variables, NONES, IFROM(1), and NLEFT are printed with this error message to aid in the debugging process. This error is not fatal; the subroutine will transfer all the 1's that are there, zero-out the IFROM vector, and continue execution. The results of the simulation are no longer reliable; however, continued execution may provide additional debug information.

UEUPDAT Error - UE Overflow

This message indicates that the desired UE strength (NAC) is larger than the maximum permissible number of aircraft (MAXAC). The current values of these two parameters are printed with the error message and also a note explaining that the subroutine will truncate NAC to the current allowable maximum, MAXAC, and execution of the simulation continues. The results are no longer valid, but more debug information may be provided by further execution of the program.

This error typically occurs in one of the following ways. Either the initial UE strength is too large or enough reserve aircraft are augmented during the scenario to cause the UE to increase beyond MAXAC. In both cases, the user should increase the MAXAC parameter in all routines using that parameter to a value large enough to handle the desired UE.

WCDIST Error - Sequential Sampling Error

This message indicates that the sequential sampling process for determining the work centers an aircraft has broken into did not terminate properly. One possible cause of this error is that the last entry in the sequential sampling array is less than 1.0; hence, either a bug has been introduced into the initializing routine for this array (WCPROB) or the entry has been written over during the simulation. Another possibility is that the random number generator has given a number greater than 1.0. The value of this last sequential sampling entry, PBRKSEQ(1,NWC) and also the value of the random draw, RDRAW, is printed with this message to aid in determining the cause of the error. The simulation results are no longer valid; however, execution of the program will continue.

WCDIST - Inconsistent Broken Aircraft

This message indicates that the specified number of aircraft to break into work centers exceeds the actual number of aircraft contained in the specified input bit-vector; i.e., either the variable NBRKAC >IACVC(1), or the value of IACVC(1) is no longer consistent with the number of 1-bits contained in the bit-vector IACVC. The values of NBRKAC, IACVC(1), and the number of aircraft actually broken into work centers, NSELEC, are printed with this error message. Execution of the simulation continues; however, the results should no longer be considered valid.

WCREAD Error - Too Many Work Centers

This message indicates that the maintenance manpower input file contains too many work centers; the current size of the work center arrays, MAXWC, is too small. However, the model will load the first MAXWC work centers in the file and perform a simulation run with just these work centers. To obtain a run with all the work centers in the input file, the user should determine the number of work centers currently in the file, and reset the MAXWC parameter in all routines to this value.

WCREAD Error - Invalid Work Center Data

This message indicates that a work center in the maintenance input file has one or more invalid characteristics, e.g., a negative service rate or break rate greater than 1.0. This problem could be caused by a bug in the Maintenance Manpower Subsystem or an input file with improper format. The AFSC of this problem work center along with its characteristics is printed with the error message to aid in the debugging process. This error is not fatal; execution of the simulation will continue. However, this problem work center will be eliminated from the simulation run.

XNORM Error - Negative Standard Deviation

This message indicates that a negative standard deviation has been passed to the XNORM routine. The value of the standard deviation must always be non-negative. The value of this input standard deviation, STDEV, is also printed with this error message. The return value is set to zero and execution of the simulation continues. The simulation results are no longer reliable; however, continued execution may provide additional debug information.

ZBITSL Error - Not Enough 1's to Zero

Indicates that the specified number of 1-bits to zero-out (NONES) is greater than the number of 1-bits indicated by the first word (IARRAY(1)) of the input bit-vector; thus, exactly NLEFT 1-bits have not been zeroed out as requested. This message indicates that some subroutine thinks there are more 1-bits in the bit-vector than there actually are. The values of all three of the pertinent variables, NONES, IARRAY(1), and NLEFT are printed. This subroutine will zero-out all the 1-bits in the vector and return control to the calling subroutine; this error does not terminate the simulation, but the results are no longer valid.

6. RUN JOB CONTROL LANGUAGE (JCL)

INTRODUCTION

The JCL files for the SGM are of two types, an input deck of control card images, or a series of system level commands. These correspond to either submitting a batch job, or executing a time-sharing run, respectively. In both cases, the JCL files must be run via the LMI STARS SUBMIT Subsystem, an interactive program for submitting batch or time-sharing runs on System C, the current computer environment for the SGM. This SUBMIT Subsystem is described in the LMI STARS User's Guide referenced in Appendix A of this manual.

TIME-SHARING

Two time-sharing JCL files are used with the SGM: a command file for running the Set-Parameter Program to specify the scenario parameters for a simulation run and another command file for submitting an interactive SGM run. Figures 6-1 and 6-2 provide listings of these run command files. Line-by-line explanations are provided below. Detailed information about time-sharing commands is provided in the Honeywell time-sharing manuals referenced in Appendix A.

Set-Parameters JCL File

This section provides a brief explanation of each line of the JCL file to set the scenario parameters for an SGM run. The numbers correspond to those shown in Figure 6-1.

```
(1) REMO CLEARFILES
(2) TEMP 01
(3) GET 0S29/N232D/SGM/ZD.&AC-TYPE."08" =
(4) RUNY 0S29/N232D/SGM/HZDATA,R
(5) PERM 01:0S29/N232D/SGM/PARAMS
(6) REMO PARAMS:HZDATA:08
```

FIGURE 6-1. SCENARIO SET-PARAMETERS RUN

Explanation

1 - Remove all files from user's available file table (AFT).

2 - Create a temporary file-01 in which scenario parameters will be written.

3 - Attach default scenario parameters for the specified aircraft type as file-08.

4 - Execute the previously compiled and loaded FORTRAN program which allows the user to specify interactively the scenario parameters.

5 - Copy the temporary file-01 containing the new scenario parameters to the permanent file, PARAMS.

6 - Remove all accessed files from the user's AFT.

SGM Time-Sharing Run

This section provides a brief description of each line of the JCL file for performing an SGM time-sharing run. The numbers correspond to the line numbers in Figure 6-2.

```
(1) REMO CLEARFILES
(2) GET OS29/N232D/SGM/PARAMS"01",R
(3) GET OS29/N241D/CDEP/SGMINPT2/&MANPOWERBASE."02",R
(4) GET LA61A/SLAY/DATA/&AC-TYPE./&SPARESFILE."04",R
(5) TEMP 07;03
(6) RUNY OS29/N232D/SGM/CSGM,R
(7) RUNY OS29/N232D/SGM/CPLOT,R
(8) REMO CSGM;07;03;CPLOT;01;02;04
```

FIGURE 6-2. SGM TIME-SHARING RUN

Explanation

1 - Remove all files from user's available file table (AFT).

2 - Attach permanent file containing scenario parameters as file-01.

3 - Attach specified maintenance manpower input data as file-02.

4 - Attach specified spares input data as file-04

5 - Create a temporary file-07 for sortie plot results, and a temporary file-03 for daily scenario parameters.

6 - Execute the previously compiled SGM FORTRAN program.

7 - Execute the previously compiled Plot Program.

8 - Remove all accessed files from user's AFT.

BATCH

The interactive parameter-setting program can only be run in the time-sharing mode; however, the SGM run process has been designed so that the model may be run in either the remote-batch or time-sharing mode. The control cards for a batch SGM run are listed in Figure 6-3. Detailed information is provided in the Honeywell Control Cards Reference Manual listed in Appendix A of this manual.

SGM Batch Run

This section provides a brief description of each line of the control cards to perform an SGM batch run. The line numbers of the explanation correspond to those in Figure 6-3.

```
100##S,R(XL) :,8,16,58
110$:NOTE:** MIKE **  OS29/N232D/SGM/RSGMBTCH
120$:IDENT:OS2011N241D ,OS29UGOODWIN
130$:OPTION:FORTRAN,NOMAP
140$:SELECT:OS29/N232D/SGM/CSGM
150$:EXECUTE
160$:LIMITS:16,27K,,1K
170$:DATA:01,NCKSUM,COPY
180$:SELECTD:OS29/N232D/SGM/PARAMS
190$:ENDCOPY
200$:PRMFL:02,R,S,OS29/N241D/CDEP/SGMINPT2/&MANPOWERBASE.
210$:PRMFL:04,R,S,LA61A/SLAY/DATA/&AC-TYPE./&SPARESFILE.
220$:FILE:03,A3R
230$:FILE:07,A1S
240$:OPTION:FORTRAN,NOMAP
250$:SELECT:OS29/N232D/SGM/CPLOT
260$:EXECUTE
270$:LIMITS:1,13K,,2K
280$:FILE:07,A1R
290$:ENDJOB
```

FIGURE 6-3. SGM BATCH RUN

Explanation

100 - Set card format, disposition, tab character and settings.

110 - Comment card identifying user and program to be run.

120 - User and account number.

130 - Set standard options for loading FORTRAN program; however, do not provide a memory map listing.

140-160 - Execute previously compiled SGM FORTRAN program with a CPU time limit of 0.14 hours, a core limit of 27K words, and an output limit of 1024 lines.

170-190 - Attach permanent file containing scenario parameters as file-01. The SELECTD card is used because it copies the file at the time the job is submitted, freeing the parameters file for subsequent updating.

200 - Attach specified maintenance manpower input data as file-02.

210 - Attach specified spares input data as file-04.

220 - Create temporary scratch file-03 for daily scenario parameters.

230 - Create temporary file for SGM sortie results to be used as input for the Plot Program.

240 - Set standard options for loading FORTRAN Plot Program; no memory map is to be listed.

250-270 - Execute previously compiled FORTRAN Plot Program with specified CPU-time, core-size, and output limits.

280 - Access file-07 containing plot data from SGM run; release file after run.

290 - Marks end of control cards.

APPENDIX A

RELATED DOCUMENTS

This appendix lists documents which provide useful information in programming and maintaining the Sortie-Generation Model on System C. This list is organized into the following general categories:

- Honeywell manuals describing the GCOS Series 600/6000 computer system, languages, etc.
- Air Force Data Services Center (AFDSC) manuals describing System C operating procedures,
- LMI reports and manuals describing previously developed software used in conjunction with the SGM.
- Technical literature describing software development techniques, software standards, and computer algorithms used in the development of the Sortie-Generation Model.

HONEYWELL MANUALS

Language Processors

COBOL Reference Manual, BS08, August 1972.

COBOL User's Guide, BS09, June 1971.

FORTRAN, BJ67, March 1973.

Operating System

Control Cards Reference Manual, BS19, February 1973.

General Comprehensive Operating Supervisor (GCOS), BR43, October 1973.

GCOS Control Cards and Abort Codes Pocket Guide, BJ69, January 1973.

GCOS Time-Sharing System Pocket Guide, BS12, October 1974.

Service and Utility Routines, Including Generators

Bulk Media Conversion, BP30, August 1973.

General Loader, BN90, March 1972.

FORTRAN Subroutine Libraries Reference Manual, BR95, May 1973.

Service Routines, DA97, June 1973.

Sort/Merge Program (Generator), BN87, March 1972.

Trace and Debug Routines, DB20, September 1972.

Time Sharing Systems

GCOS Time-Sharing System General Information, BS01, July 1973.

GCOS Time-Sharing System Programmer's Reference Manual, BR39, November 1971.

Time-Sharing Applications Library, DA44, December 1976.

Time-Sharing FORTRAN, BR70, February 1973.

Time-Sharing Text Editor, BR40, June 1973.

Time-Sharing Terminal/Batch Interface Facility, BR99, January 1972.

AFDSC MANUALS

Air Force Data Services Center

Users Handbook, General Information and Procedures, Volume I, October 1979.

Users Handbook, AFDSC Project Management and Standards, Volume II, March 1979.

Users Handbook, GCOS Computer System, Volume III, November 1976.

Users Handbook, GCOS Remote Terminals, Volume IV, December 1975.

LMI REPORTS

Logistics Management Institute

LMI Availability System Levels of Indenture Model, (Task AF-605) November 1979.

LMI Availability System Overview, (Task AF605), August 1978.

Test of the Availability Model, (Task AF-605), August 1978.

The System That Automatically Runs Systems (STARS) Overview, (Task AF-605), August 1978.

STARS (The System That Automatically Runs Systems) System Guide, (Task AF-605), August 1978.

STARS User's Guide, (Task AF-605), August 1978.

STARS Analyst's Guide, (Task AF-605), August 1978.

An Efficient Optimization Procedure for Levels-of-Indenture Inventory Model, (Task AF-605 Working Note), February 1978.

A Method of Treating Common Recoverable Components in the LMI Essentiality Model, (Task 76-5 Working Note), March 1976.

A Model to Allocate Repair Dollars and Facilities Optimally, (Task 74-9), August 1974.

Test of a System Which Considers the Priority Allocation of Spare Recoverable Components, (Task 73-7), August 1973.

Measurements of Military Essentiality, (Task 72-3), August 1972.

TECHNICAL LITERATURE

Caine, Stephen H. and E. Kent Gordon, "PDL - A tool for software design" in Tutorial on Software Design Techniques, California: Institute of Electrical and Electronics Engineers Inc., pp 168-173, 1977.

Feller, William, An Introduction to Probability Theory and Its Applications, Volume I, New York: John Wiley and Sons, 1968.

Fishman, George S., Principles of Discrete Event Simulation, New York: John Wiley and Sons, 1978.

Gass, Saul I., Computer Science and Technology: Computer Model Documentation: A Review and an Approach, Washington, D.C.: National Bureau of Standards, February 1979.

Graves, Joseph S., "On the Storage and Handling of Binary Data using FORTRAN with Applications to Integer Programming" in Operations Research, Vol. 27 No. 3, pp. 534-547, May-June 1979.

Krecker, Dr. D. K. et al., Software Documentation and Development Conventions, (BDM/W-7-9-556-TR), The BDM Corporation, September 1979.

Kronmal, Richard A. and Arthur V. Peterson Jr., "On the Alias Method for Generating Random Variables from a Discrete Distribution" in The American Statistician, Vol. 33 No. 4, pp. 214-218, November 1979.

Naylor, Thomas H. et al., Computer Simulation Techniques, New York: John Wiley and Sons, 1966.

National Bureau of Standards, Computer Science and Technology, Computer Model Documentation Guide, (NBS Special Publication 500-73), January 1981.

APPENDIX B

SAMPLE MODEL RUN

This appendix provides a sample SGM run along with complete listings of all inputs used to produce these results. This run represents a typical F-4E maximal-effort flying scenario. The SGM inputs consist of a flying scenario description, a recoverable-spares file, and an aircraft maintenance file. Listings of the scenario parameters and the spares file are provided immediately after the SGM run results. The aircraft maintenance description is always provided as part of the actual SGM results. The aircraft maintenance file was produced by the SGM Maintenance Subsystem based on maintenance data collected from Seymour Johnson AFB. The spares file was produced by the SGM Spares Subsystem for a 72-UE wing located at Seymour Johnson AFB.

SGM RUN RESULTS

```
=RUNC OS29/N232D/SGM/RSGMTSS
ENTER MANPOWERBASE?
=SJWC
ENTER AC-TYPE      ?
=F4
ENTER SPARESFILE   ?
=SEYMORNF
```

```
*****************************************************************
***************************   SGM RUN   *************************
*****************************************************************
```

SIMULATION -   REPLICATIONS = 40   RANDOM NUMBER SEED = 12.3

AIRCRAFT -   UE = 72   RESERVES = 24   MAXIMUM LAUNCH-SIZE = 72

FLYING SCHEDULE -

DAYS	WAVES PER DAY	TAKEOFF TIMES FIRST	TAKEOFF TIMES LAST	MINIMAL TURNAROUND	SORTIE LENGTH	WAIT TIME	OVERNIGHT RECOVERY
30	5	0600	1824	1.40	1.70	0.00	8.50

RATES -

INITIAL NMCM RATE	ATTRITION	AIRCRAFT BREAK RATE	GROUND-ABORT
0.150	0.01	0.2000	0.0400

LRU TYPES - 262

****
********** AIRCRAFT MAINTENANCE **********
****

WC #	AFSC	BREAK RATE	TOTAL SERVERS	SERVICE RATE (ACFT/HOUR)
1	321X2	0.2878	27.77	0.1417
2	325X0	0.1515	15.06	0.1384
3	328R3	0.1062	31.07	0.1273
4	328X0	0.2010	18.36	0.1769
5	328X4	0.1506	9.57	0.2507
6	404X1	0.0225	12.00	0.1510
7	423E2	0.1699	9.95	0.0682
8	423E3	0.0608	8.31	0.1043
9	423X0	0.1188	12.28	0.1327
10	423X1	0.0793	6.57	0.1571
11	423X4	0.0836	11.91	0.1365
12	426X2	0.0508	10.81	0.1585
13	427R0	0.0379	4.56	0.3955
14	427X5	0.1633	14.89	0.2584
15	431E1	0.0335	10.73	0.0857
16	431X1	0.0527	131.49	0.5356
17	462X0	0.1641	7.27	0.5434

DAY	PER	SORTIES/ PERIOD	SORTIES/ DAY	SORTIES/ AC	NMCM	NMCS	CUM. LOSSES	RES. LEFT
1	1	55.4			10.0	4.4	0.	24.0
	2	45.2			19.0	5.7	0.4	
	3	38.6			23.9	7.1	0.9	
	4	34.9			26.0	8.3	1.3	
	5	33.2	207.2	2.91	26.5	9.4	1.6	
			207.2					
2	1	47.5			14.2	8.4	1.7	22.3
	2	40.0			19.5	9.7	2.4	
	3	35.9			23.2	10.7	2.8	
	4	32.3			24.6	12.1	3.1	
	5	30.7	186.5	2.63	25.3	13.0	3.6	
			393.7					
3	1	46.7			11.7	11.4	4.0	20.0
	2	37.9			18.8	12.9	4.6	
	3	32.5			22.9	14.2	5.2	
	4	30.3			24.0	15.2	5.5	
	5	28.3	175.6	2.48	24.4	16.2	5.8	
			569.3					
4	1	43.1			12.7	14.3	6.1	17.8
	2	36.1			18.5	15.6	6.6	
	3	31.5			21.9	16.7	6.9	
	4	29.2			23.5	17.7	7.1	
	5	28.2	168.0	2.35	22.9	18.7	7.4	
			737.3					
5	1	41.3			12.1	16.6	7.8	16.2
	2	34.9			17.7	17.8	8.1	
	3	30.1			21.2	18.9	8.5	
	4	27.3			22.3	20.0	8.8	
	5	25.3	159.1	2.23	23.4	20.8	9.1	
			896.4					
6	1	40.5			11.7	18.3	9.4	14.6
	2	33.9			16.7	19.5	9.8	
	3	28.5			21.0	20.6	10.1	
	4	27.3			21.2	21.4	10.3	
	5	25.8	156.2	2.19	21.3	22.5	10.5	
			1052.6					
7	1	38.2			12.0	20.1	10.8	13.2
	2	32.4			16.3	21.7	11.1	
	3	28.5			18.5	22.7	11.6	
	4	24.8			21.3	23.6	11.9	
	5	23.2	147.0	2.06	21.8	24.3	12.2	
			1199.6					
8	1	37.1			12.7	20.9	12.4	11.6
	2	31.6			16.7	21.9	12.9	
	3	27.1			20.0	22.9	13.1	
	4	24.3			21.6	23.8	13.5	
	5	23.4	143.5	2.02	21.8	24.7	13.8	

			1343.1					
9	1	36.8			12.6	21.1	14.1	10.0
	2	30.8			16.8	22.5	14.6	
	3	27.4			19.4	23.1	14.9	
	4	24.6			21.2	23.9	15.0	
	5	23.2	142.8	2.00	21.9	24.8	15.3	
			1485.9					
10	1	36.6			11.7	21.8	15.5	8.5
	2	30.4			16.8	23.2	15.7	
	3	27.4			19.2	24.0	15.9	
	4	25.0			20.3	25.2	16.1	
	5	23.5	143.0	2.00	20.8	25.8	16.3	
			1628.9					
11	1	35.9			12.4	22.1	16.6	7.5
	2	30.5			16.6	23.3	17.0	
	3	26.4			19.5	24.4	17.3	
	4	25.3			20.0	24.9	17.5	
	5	23.9	142.0	1.99	19.9	26.0	17.7	
			1770.8					
12	1	35.9			11.6	22.6	17.8	6.3
	2	30.2			16.6	23.7	18.2	
	3	26.9			18.0	25.2	18.5	
	4	24.5			19.3	26.2	18.7	
	5	23.6	141.0	1.98	19.5	26.8	18.8	
			1911.9					
13	1	36.4			10.6	23.3	19.2	5.0
	2	29.7			16.1	24.4	19.5	
	3	26.7			18.2	25.4	19.8	
	4	24.8			18.8	26.2	20.0	
	5	22.9	140.4	1.97	19.8	27.0	20.2	
			2052.3					
14	1	35.9			11.4	23.0	20.4	3.9
	2	29.8			15.9	24.2	20.6	
	3	25.9			19.3	25.1	20.9	
	4	24.3			19.3	26.0	21.2	
	5	22.8	138.8	1.95	20.2	26.9	21.4	
			2191.0					
15	1	35.4			11.5	23.1	21.5	3.0
	2	30.8			15.3	24.2	21.7	
	3	27.0			17.5	25.0	22.1	
	4	24.4			18.7	26.0	22.5	
	5	23.6	141.2	1.99	19.0	26.7	22.8	
			2332.3					
16	1	35.4			10.6	23.3	23.0	2.1
	2	29.9			15.4	24.3	23.2	
	3	26.3			18.0	25.3	23.4	
	4	23.6			19.7	25.9	23.8	
	5	22.5	137.7	1.95	20.2	26.4	24.0	
			2469.9					
17	1	34.6			10.6	23.9	24.3	1.5
	2	28.9			15.0	24.9	24.5	
	3	25.5			17.6	25.7	24.9	
	4	23.9			18.4	26.5	25.2	

	5	21.7	134.5	1.93	19.1	27.3	25.5	
			2604.4					
18	1	33.8			11.1	23.7	25.8	0.7
	2	28.6			14.7	24.8	26.0	
	3	25.0			17.5	25.6	26.3	
	4	23.1			18.4	26.5	26.4	
	5	21.9	132.3	1.91	18.7	27.3	26.6	
			2736.7					
19	1	32.9			10.7	23.8	26.8	0.5
	2	28.1			14.5	24.8	27.1	
	3	24.5			16.9	25.8	27.3	
	4	22.9			17.9	26.5	27.5	
	5	21.3	129.7	1.90	18.5	27.0	27.8	
			2866.4					
20	1	33.4			9.6	23.3	28.0	0.2
	2	27.6			14.4	24.4	28.4	
	3	24.5			16.5	25.1	28.7	
	4	23.0			17.3	25.8	28.8	
	5	21.3	129.8	1.93	18.0	26.5	29.1	
			2996.2					
21	1	32.2			9.5	23.5	29.3	0.1
	2	27.7			13.0	24.7	29.6	
	3	24.0			15.5	25.6	29.8	
	4	22.1			16.5	26.5	30.1	
	5	20.4	126.3	1.91	17.4	27.2	30.3	
			3122.6					
22	1	30.2			10.3	23.6	30.5	0.0
	2	25.6			14.0	24.7	30.8	
	3	23.2			15.9	25.3	30.9	
	4	21.1			17.0	26.0	31.1	
	5	19.7	119.8	1.84	17.2	26.7	31.3	
			3242.3					
23	1	29.9			9.7	23.6	31.5	0.
	2	25.2			13.3	24.8	31.7	
	3	23.0			14.9	25.4	32.0	
	4	21.1			15.6	26.0	32.3	
	5	18.9	118.1	1.84	17.0	26.6	32.6	
			3360.4					
24	1	29.3			9.1	23.5	32.8	0.
	2	24.5			13.2	24.3	33.1	
	3	21.8			15.0	25.0	33.3	
	4	20.5			15.5	25.8	33.4	
	5	18.8	115.0	1.83	15.9	26.5	33.7	
			3475.4					
25	1	28.6			9.2	23.0	33.8	0.
	2	24.1			13.2	23.8	34.0	
	3	20.5			15.5	25.0	34.2	
	4	19.7			15.5	25.8	34.4	
	5	19.1	112.0	1.81	15.0	26.5	34.7	
			3587.3					
26	1	29.0			8.0	23.0	34.9	0.
	2	24.0			11.7	24.2	35.1	
	3	20.7			14.1	25.1	35.3	

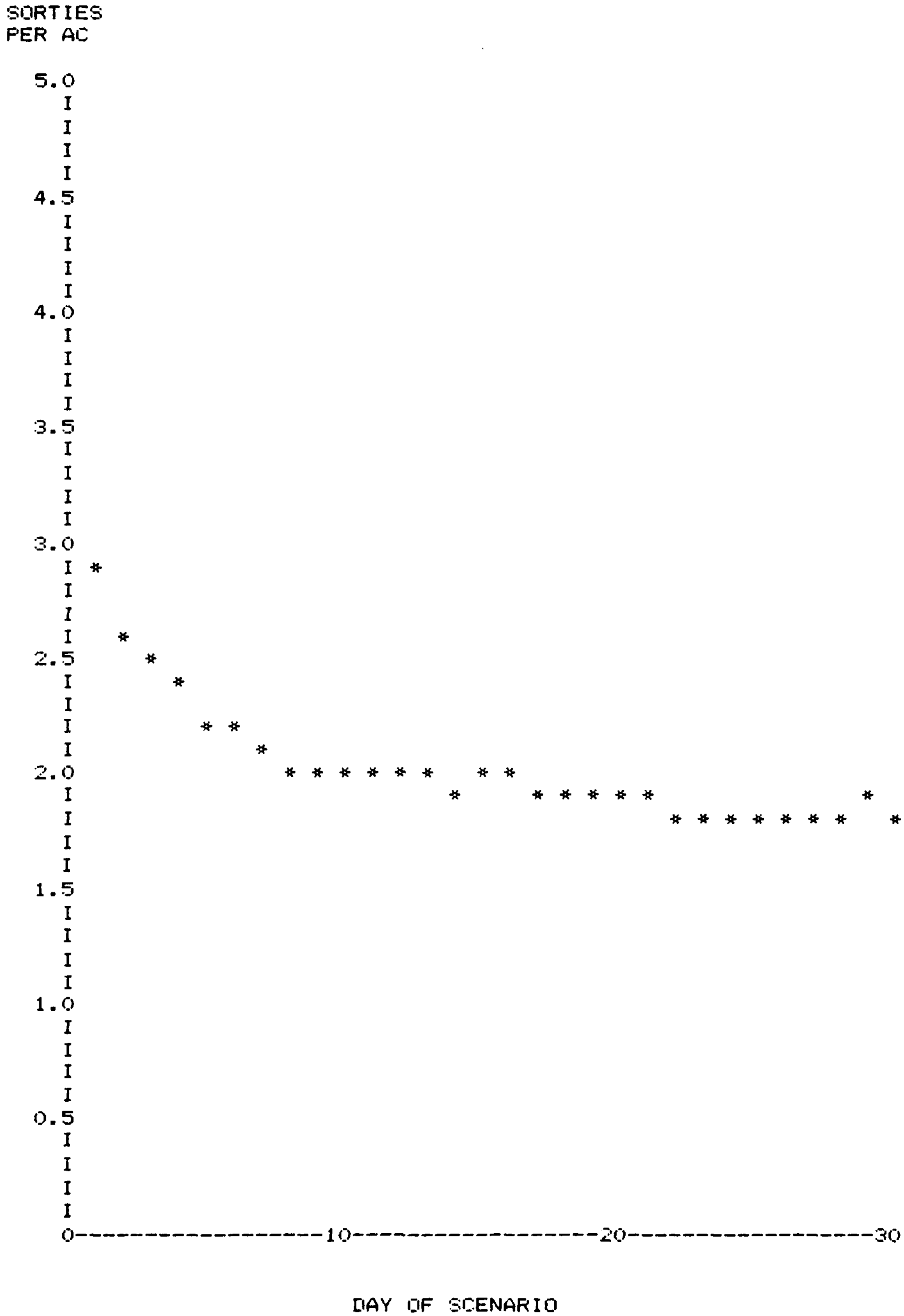
	4	18.6			15.3	25.5	35.6	
	5	17.6	109.9	1.81	15.7	26.1	35.8	
			3697.2					
27	1	27.8			8.4	22.7	35.9	0.
	2	23.0			12.4	23.7	36.2	
	3	20.5			14.1	24.3	36.3	
	4	18.2			15.2	25.2	36.5	
	5	17.3	106.9	1.79	15.6	25.8	36.7	
			3804.1					
28	1	27.8			8.3	22.3	36.8	0.
	2	23.9			10.8	23.2	37.0	
	3	20.2			13.7	24.0	37.1	
	4	18.6			14.9	24.6	37.3	
	5	17.4	107.9	1.83	15.3	25.0	37.6	
			3912.0					
29	1	27.1			8.2	21.8	37.8	0.
	2	23.2			11.3	22.4	38.2	
	3	20.4			12.9	23.3	38.5	
	4	18.6			13.9	24.0	38.7	
	5	17.5	106.8	1.86	14.0	24.7	39.0	
			4018.8					
30	1	26.5			7.9	21.3	39.1	0.
	2	22.0			10.8	22.4	39.4	
	3	20.0			12.5	23.0	39.6	
	4	18.1			13.6	23.5	39.8	
	5	17.2	103.7	1.84	13.7	24.4	39.9	
			4122.5					

TOTAL SORTIES FLOWN = 4122.5

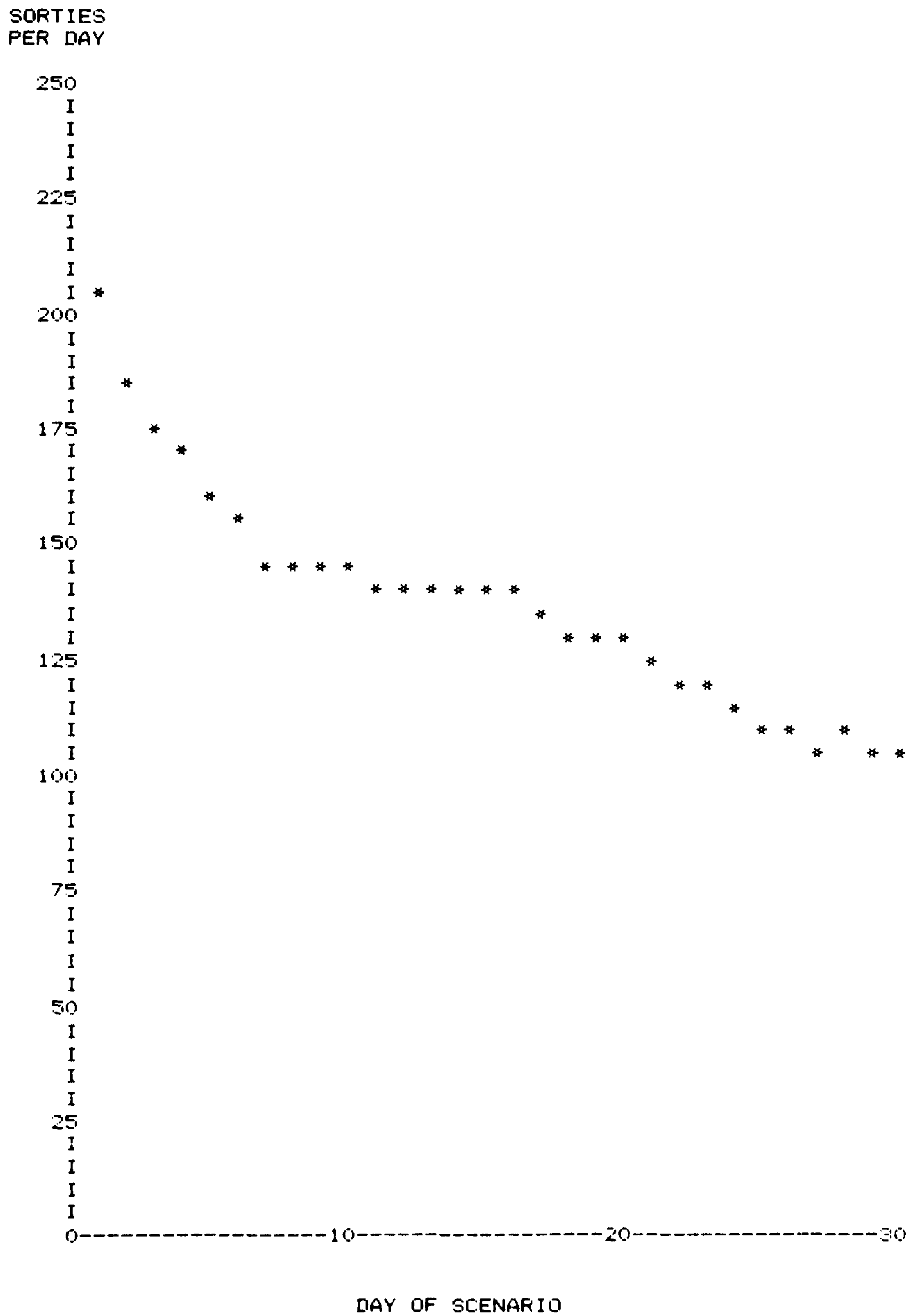
CPU TIME USED = 9.54 MIN

MEMORY USED = 20 K WORDS

2.91	2.63	2.48	2.35	2.23
2.19	2.06	2.02	2.00	2.00
1.99	1.98	1.97	1.95	1.99
1.95	1.93	1.91	1.90	1.93
1.91	1.84	1.84	1.83	1.81
1.81	1.79	1.83	1.86	1.84


SORTIES
PER AC
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
10
20
30
DAY OF SCENARIO

207	186	176	168	159
156	147	144	143	143
142	141	140	139	141
138	135	132	130	130
126	120	118	115	112
110	107	108	107	104


SORTIES
PER DAY
250
225
200
175
150
125
100
75
50
25
0
10
20
30
DAY OF SCENARIO

SCENARIO INPUT PARAMETERS

THE CURRENT VALUES OF THE SCENARIO INPUTS ARE :

INPUT CODE	SCENARIO ITEM			CURRENT VALUE
1	# SIMULATIONS		=	40
2	RANDOM NUMBER SEED		=	12.3
3	UE		=	72
4	AIRCRAFT BREAK RATE		=	.20
5	INITIAL NMCM RATE		=	.15
6	# DAYS		=	30
7	FIRST TAKEOFF TIME		=	0600
8	LAST TAKEOFF TIME		=	1824
9	SORTIE LENGTH (HRS)		=	1.7
10	MINIMAL RECOVERY TIME (HRS)		=	1.4
11	INFINITE MANPOWER (YES/NO)		=	NO
12	INFINITE SPARE PARTS (YES/NO)		=	NO
13	AUGMENT RESERVE AC (YES/NO)		=	NO
14	MAX SORTIES/DAY FOR PLOT(OR 0)		=	0

THE FOLLOWING ITEMS MAY VARY BY DAY(D) OR CYCLE/DAY(C/D)

15	ATTRITION RATE	(D)	=	.01
16	GROUND ABORT RATE	(D)	=	.04
17	# MASS LAUNCHES PER DAY	(D)	=	5
18	RESERVE AIRCRAFT	(D)	=	24
19	MAXIMUM LAUNCH-SIZE	(C/D)	=	72

RECOVERABLE SPARES FILE

NSN	REMOVAL RATE	QPA	FAP	INITIAL STOCK	INITIAL NO. IN RESUPPLY	BASE NRTS	RESUPPLY TIMES (DAYS) BASE	DEPOT
1430010454699BF	.01786	1	1.0	5	8.156	0.08	6.0	26.2
1430010387038BF	.01700	1	1.0	6	7.211	0.05	6.0	27.4
2620000884523	.01708	2	0.9	90	28.649	1.00	0.	18.0
5865001994210EW	.00109	4	1.0	1	1.405	0.90	6.0	31.0
1430010399244BF	.01285	1	0.3	1	1.744	0.08	6.0	27.7
6610004629837BF	.00404	1	1.0	6	3.091	0.85	5.0	14.7
1430010610350BF	.00607	1	1.0	3	2.226	0.	6.0	0.
1630004463778	.01676	2	1.0	33	9.226	0.06	5.0	14.0
1270010588980	.00483	1	1.0	4	0.937	0.45	6.0	11.5
5826010395000	.00600	1	0.3	1	1.373	0.30	9.0	22.1
5826010401785	.00692	1	0.3	2	1.650	0.40	8.0	21.0
1430010387055BF	.00484	1	1.0	3	1.742	0.	6.0	0.
5826010183511	.00168	2	1.0	9	3.878	0.60	4.0	34.6
1430002356325BF	.01023	1	0.9	6	2.282	0.09	3.0	17.1
2840008717414PL	.00084	2	1.0	2	2.431	0.76	6.0	28.6
5865000233292EW	.00200	1	1.0	0	0.005	1.00	0.	6.8
5865003713344EW	.00133	4	1.0	3	0.723	0.97	3.0	13.3
1270000641997	.00441	1	1.0	6	1.345	0.66	4.0	10.9
6115008681999EW	.00184	5	0.2	1	0.092	0.43	2.0	78.8
5865000999348EW	.00103	5	1.0	2	0.548	0.98	2.0	12.3
6610008144117BF	.00147	1	1.0	2	0.726	1.00	0.	12.7
1660000714255	.00513	1	1.0	19	4.565	0.47	5.0	34.0
5865000854945EW	.00059	1	1.0	0	0.005	1.00	0.	14.6
1560007883941BF	.00066	1	1.0	1	0.664	0.78	6.0	23.1
5865001627964EW	.00070	3	1.0	3	0.542	0.95	5.0	18.5
5865007598099EW	.00067	4	1.0	2	0.410	0.97	4.0	15.4
1270005562269	.01025	1	0.4	5	1.519	0.29	4.0	16.9
6115010267271EW	.00171	4	0.3	0	0.041	0.80	5.0	16.0
5821010668605	.00218	1	1.0	2	0.803	0.17	5.0	16.3
5865010481589EW	.00055	6	1.0	2	0.463	0.88	4.0	16.3
5865001887918EW	.00050	1	1.0	0	0.005	0.83	6.0	14.3
5865004095152EW	.00112	2	1.0	1	0.008	0.15	6.0	14.0
5826010395013	.00340	1	0.3	2	0.853	0.71	8.0	16.7
6610009988758BF	.00226	1	1.0	5	1.503	0.80	5.0	14.1
5865004376027EW	.00118	1	1.0	3	0.011	1.00	0.	16.0
4310010183040BF	.00157	1	1.0	4	1.182	0.96	7.0	15.1
1680004500573BF	.00090	3	1.0	5	0.712	1.00	0.	14.0
5865004764442EW	.00073	4	1.0	3	0.402	0.96	2.0	11.5
5826010403093	.00217	1	0.3	1	0.515	0.72	5.0	17.3
5865001559266EW	.00055	10	1.0	3	0.525	0.95	2.0	11.5
6605009940194	.01606	1	0.7	12	2.667	0.01	5.0	14.0
5865001350117EW	.00084	6	0.8	0	0.012	0.42	6.0	6.6
2620010579673	.02334	2	0.1	24	6.176	1.00	0.	21.0
5865003294045EW	.00063	2	0.6	0	0.023	0.85	1.0	15.0
5865000076945EW	.00055	4	1.0	2	0.252	0.97	4.0	11.8
1430010682150BF	.00137	2	1.0	2	0.059	1.00	0.	14.0
1430010384963BF	.00151	1	1.0	4	1.320	0.88	2.0	16.3
5865001350116EW	.00104	6	0.8	1	0.012	0.05	5.0	8.0
5865000076949EW	.00057	4	1.0	2	0.240	0.97	4.0	10.5
5865000094382EW	.00055	3	1.0	2	0.257	0.97	1.0	12.2

NSN	REMOVAL RATE	QPA	FAP	INITIAL STOCK	INITIAL NO. IN RESUPPLY	BASE NRTS	RESUPPLY TIMES (DAYS) BASE	DEPOT
5826010424054	.00388	1	0.3	2	0.696	0.88	6.0	10.7
5865008685177EW	.00081	2	0.6	1	0.027	0.86	1.0	14.0
1430004902978BF	.00379	1	1.0	16	4.636	0.94	4.0	20.9
5865008685230EW	.00077	4	0.8	0	0.012	0.13	3.0	13.2
5865001681504EW	.00097	2	0.6	0	0.013	0.21	4.0	7.7
6615010709243BF	.00399	1	0.3	1	0.378	0.04	5.0	20.5
2995006911224	.00179	2	1.0	9	2.204	0.82	6.0	14.0
5865010976255EW	.00069	2	1.0	5	5.387	1.00	0.	20.2
5865010211657EW	.00132	2	1.0	3	0.750	0.12	5.0	16.8
5865010149262EW	.00050	1	1.0	1	0.006	1.00	0.	13.0
1650010841569	.00408	2	1.0	18	3.083	0.57	5.0	14.0
1560008670561BF	.00073	2	1.0	3	0.858	0.87	6.0	11.7
5865004263144EW	.00100	4	1.0	6	2.315	0.24	6.0	12.6
5826010395015	.00113	1	0.3	1	0.319	0.77	6.0	18.9
1650001486506BF	.00120	2	1.0	6	0.713	1.00	0.	12.0
1430001326677BF	.00067	1	1.0	2	0.611	0.91	4.0	16.0
1650009243005BF	.00075	2	1.0	4	0.484	0.90	4.0	14.0
5826010419255	.00252	1	0.3	2	0.533	0.52	8.0	16.4
5865008685231EW	.00139	2	0.6	2	0.025	0.07	5.0	27.0
1270010428441	.00071	1	0.4	1	0.280	0.85	2.0	23.2
6615010546075BF	.00167	1	1.0	8	1.954	1.00	0.	22.0
1560007906873BF	.00059	1	1.0	1	0.238	0.76	7.0	8.8
1270003528728	.00100	1	0.4	1	0.272	0.86	3.0	15.4
5826010397621	.00051	1	1.0	2	0.497	0.49	5.0	33.3
1270003495219	.00099	1	0.4	1	0.276	0.83	3.0	15.9
1430010597789BF	.00064	1	1.0	1	0.286	0.10	6.0	20.5
1430000780463BF	.00422	1	1.0	4	1.027	0.08	3.0	15.0
1270003495873	.00090	1	0.4	1	0.236	0.93	3.0	15.5
1430001790011BF	.00053	1	0.1	0	0.018	0.86	6.0	10.5
6615004200406BF	.00051	3	1.0	4	0.641	1.00	0.	12.0
6610000109356BF	.00121	1	1.0	2	0.533	0.70	4.0	8.8
5826010419398	.00122	1	0.3	1	0.262	0.43	8.0	17.7
1430001444336BF	.00137	1	0.7	6	1.767	0.94	4.0	30.7
1270005429309	.00083	1	0.4	1	0.236	0.84	9.0	15.4
6610004001201BF	.00064	1	1.0	2	0.554	0.77	5.0	16.2
5865000139369EW	.00125	2	0.5	2	0.017	0.52	3.0	9.0
1430010533212BF	.00135	1	1.0	2	0.486	0.	6.0	0.
6610004335240	.00459	1	0.1	2	0.461	0.88	6.0	15.3
1270003482091	.00064	1	0.4	1	0.194	0.79	6.0	19.3
6610001812539	.00120	2	1.0	8	1.826	0.85	17.0	14.0
5826010419380	.00096	1	0.3	1	0.206	0.64	6.0	15.6
5865000139368EW	.00081	2	0.5	1	0.006	0.06	2.0	13.0
1270005518449	.00140	1	0.4	2	0.444	0.93	3.0	15.9
5826010419381	.00104	1	0.3	1	0.181	0.33	6.0	17.6
5895009190413	.00062	2	1.0	6	1.325	0.80	4.0	23.4
6610009250935	.00139	1	0.1	1	0.165	0.86	5.0	15.8
1430001946467BF	.00084	1	1.0	2	0.459	0.90	5.0	9.0
2840008846275PL	.00053	2	1.0	4	0.959	0.96	11.0	14.0
6610001337868	.00070	1	1.0	2	0.372	0.92	5.0	10.7
1560001430932BF	.00227	1	1.0	5	1.243	0.09	9.0	7.6

NSN	REMOVAL RATE	QPA	FAP	INITIAL STOCK	INITIAL NO. IN RESUPPLY	BASE NRTS	RESUPPLY TIMES (DAYS) BASE	DEPOT
5865010805675EW	.00333	1	1.0	7	2.664	0.29	4.0	20.0
2840010269455PL	.00083	2	1.0	6	1.475	0.97	5.0	14.0
1430005072655BF	.00721	1	0.3	3	0.606	0.06	4.0	17.3
6610009250934	.00739	1	0.1	4	0.897	0.78	10.0	15.5
1680010520816LS	.00103	2	1.0	6	1.153	0.97	9.0	11.0
1270003495215	.00097	1	0.4	1	0.125	0.16	4.0	19.6
1430000435192BF	.00056	1	1.0	1	0.125	0.08	3.0	14.5
1430001117990BF	.00144	1	1.0	4	0.951	0.60	4.0	15.3
1560000829118BF	.00052	1	1.0	1	0.101	0.04	3.0	21.4
1660001359566	.00185	1	1.0	7	1.420	0.97	6.0	16.0
1270010298391	.00058	1	0.4	1	0.098	0.75	3.0	10.3
1270003939141	.00061	1	0.4	1	0.088	0.15	5.0	23.8
6610010451020	.00119	1	1.0	4	0.817	0.80	7.0	14.1
2840010272393PL	.00082	2	1.0	6	1.320	0.84	8.0	14.0
1430001444333BF	.00426	1	0.1	1	0.086	0.12	4.0	25.7
2840006865740PL	.00052	2	1.0	4	0.868	0.96	8.0	12.0
1680007335768LS	.00050	4	1.0	7	0.515	0.73	4.0	12.0
5826010408428	.00094	1	0.3	1	0.083	0.70	6.0	4.7
6610001811750	.00058	1	1.0	2	0.294	0.83	7.0	10.9
6605009458168	.01023	1	0.7	13	2.595	0.38	4.0	14.0
5826010329930	.00195	1	1.0	4	1.033	0.15	6.0	14.2
1430001444284BF	.00088	1	1.0	10	2.610	0.84	6.0	56.4
1430005072644BF	.00656	1	1.0	7	1.530	0.08	3.0	14.1
1630002769849	.00180	2	1.0	8	1.325	0.28	7.0	10.0
5895003977851	.00243	1	0.3	2	0.311	0.02	8.0	54.6
1430001444319BF	.00056	1	1.0	3	0.615	0.95	1.0	18.8
1560009547752BF	.00055	2	1.0	6	1.023	0.83	4.0	21.0
2995006141130PL	.00069	2	1.0	5	0.945	0.97	10.0	11.0
6615010520423BF	.00065	1	1.0	3	0.519	1.00	0.	15.2
1660004463827	.00057	1	1.0	2	0.272	0.89	6.0	9.5
2915001338007PL	.00068	2	1.0	6	1.195	0.93	4.0	15.0
4320000586925HS	.00160	4	1.0	18	3.142	0.71	5.0	12.0
6115009031256BF	.00247	2	1.0	10	1.869	0.44	5.0	10.0
1430003592030BF	.00082	1	0.1	1	0.026	0.91	6.0	9.4
1270005518452	.00737	1	0.4	4	0.729	0.06	4.0	15.4
2915010887077PL	.00055	2	1.0	5	0.794	0.95	11.0	12.1
5826009941578	.00050	1	1.0	2	0.217	0.14	6.0	22.3
1650009243006BF	.00082	2	1.0	6	0.416	0.91	6.0	11.0
1270005518451	.00086	1	0.4	2	0.224	0.79	4.0	15.5
5826000897912	.00233	1	1.0	7	1.154	0.89	5.0	12.0
6605010787915	.00769	1	0.3	4	0.505	0.15	4.0	9.0
5826010329923	.00056	1	1.0	2	0.208	0.18	5.0	14.1
1560001430930BF	.00108	1	1.0	4	1.319	0.10	11.0	18.2
1430005315163BF	.00324	1	1.0	4	0.676	0.05	3.0	14.2
5895003977852	.00147	1	1.0	3	0.450	0.09	4.0	18.2
6610004001202BF	.00096	2	0.7	6	0.829	0.92	15.0	11.0
1430002989723BF	.00160	1	0.3	3	0.435	0.89	5.0	16.8
5826004449847	.00092	1	0.5	2	0.133	0.85	2.0	15.0
1650009995494BF	.00085	1	1.0	3	0.360	0.39	6.0	13.0
1270001487615	.00453	1	0.4	3	0.371	0.13	3.0	12.9

NSN	REMOVAL RATE	QPA	FAP	INITIAL STOCK	INITIAL NO. IN RESUPPLY	BASE NRTS	RESUPPLY TIMES (DAYS) BASE	DEPOT
1430005072656BF	.00993	1	1.0	15	3.482	0.26	3.0	14.0
5990002445715NT	.00234	1	0.6	5	0.943	0.95	3.0	11.9
1620009891992	.00077	1	1.0	3	0.352	0.99	5.0	9.2
6645008722128	.00061	1	1.0	3	0.389	0.92	5.0	12.2
6615006000969BF	.00065	1	1.0	4	0.606	1.00	0.	18.0
1630010266543	.00089	1	1.0	4	0.623	0.77	5.0	15.1
1270001185901	.00136	1	0.4	2	0.119	0.12	3.0	16.3
2995001598730	.00163	2	1.0	11	1.836	0.98	9.0	11.0
1430001834083BF	.00349	2	0.1	4	0.279	0.93	2.0	12.0
1430001444315BF	.00067	1	0.7	3	0.409	0.87	3.0	16.3
6610000657276BF	.00054	2	1.0	6	0.641	0.93	3.0	12.0
1095004538407	.00107	1	1.0	3	0.361	0.08	4.0	22.8
1430000600341BF	.00070	1	1.0	4	0.630	0.90	3.0	16.6
1270001095653	.00080	1	0.4	2	0.063	0.14	4.0	14.0
1430001117993BF	.00073	1	1.0	4	0.645	0.91	3.0	15.6
6680008800844BF	.00051	2	1.0	6	0.602	0.80	8.0	12.0
1270004752473	.00077	1	0.2	2	0.066	0.93	4.0	6.7
6760004051090	.00182	1	1.0	4	0.880	0.16	4.0	11.3
2935007892422	.00064	2	1.0	7	0.719	0.85	4.0	13.0
1430003934750BF	.00057	1	0.1	2	0.022	0.93	6.0	12.0
6615007202931	.00054	1	1.0	3	0.244	0.92	5.0	9.0
6610000863840	.00144	2	1.0	10	1.943	0.89	4.0	10.0
1430005957721BF	.00061	1	0.1	2	0.013	0.94	6.0	6.0
1270009160176	.00051	1	1.0	4	0.523	0.97	1.0	18.1
6615010520422BF	.00107	1	1.0	4	0.458	1.00	0.	8.0
6605008365333	.00847	1	0.7	10	1.605	0.11	4.0	22.0
1430005203506BF	.00438	2	0.1	5	0.229	0.88	3.0	8.0
1270009755895	.00053	1	1.0	3	0.220	0.91	3.0	8.0
6610009867628BF	.00130	2	1.0	11	1.549	0.93	5.0	12.0
5865010384616EW	.00069	2	1.0	10	8.402	1.00	0.	17.9
1430001940072BF	.00050	1	1.0	3	0.196	0.96	9.0	6.2
6605009497835	.00622	1	0.7	7	1.072	0.07	4.0	14.0
1270004767945	.00092	1	1.0	5	0.703	0.87	4.0	14.4
6685001159606BF	.00061	1	1.0	4	0.372	0.78	4.0	14.0
1660004959012BF	.00175	1	1.0	7	1.022	0.82	6.0	13.0
6105002620432BF	.00426	1	1.0	15	3.317	0.91	5.0	14.0
1270000238962	.00262	1	1.0	5	0.592	0.27	3.0	7.0
1430001444292BF	.00081	1	1.0	5	0.705	0.90	4.0	15.3
6620005538827	.00082	2	1.0	9	0.901	0.94	6.0	11.0
6615008699834	.00082	1	1.0	4	0.306	0.30	5.0	14.0
6340001165963BF	.00113	2	1.0	10	1.069	0.79	4.0	11.0
1270000041879	.00106	1	0.4	3	0.095	0.10	3.0	20.0
6610004809436BF	.00189	1	1.0	7	0.998	0.63	4.0	14.0
1430003980384BF	.00096	1	1.0	6	0.967	0.95	4.0	16.5
1660000893553	.00072	1	1.0	4	0.280	0.74	4.0	10.0
1270000238954	.00137	1	1.0	4	0.283	0.36	3.0	5.0
1430001458910BF	.00421	1	1.0	6	0.780	0.05	3.0	4.0
4810000893550TP	.00069	1	1.0	4	0.269	0.64	4.0	10.0
1270000238963	.00442	1	0.2	4	0.264	0.10	4.0	15.0
1430009328553BF	.00128	1	1.0	8	1.439	0.96	3.0	19.0

NSN	REMOVAL RATE	QPA	FAP	INITIAL STOCK	INITIAL NO. IN RESUPPLY	BASE NRTS	RESUPPLY TIMES (DAYS) BASE	DEPOT
1430001444407BF	.00102	1	1.0	5	0.589	0.92	2.0	10.0
1650003500992BF	.00118	1	1.0	6	0.744	0.88	4.0	13.0
1430001747045BF	.01938	1	0.1	4	0.267	0.06	3.0	21.0
1430008339603BF	.00058	1	1.0	5	0.591	0.99	8.0	16.7
1430002471537BF	.00075	1	1.0	5	0.574	0.88	3.0	14.0
6710002600300	.00063	1	1.0	4	0.262	0.35	3.0	17.0
5865010418257EW	.00095	2	1.0	9	1.099	0.44	6.0	9.0
1270004767946	.00207	1	1.0	5	0.469	0.19	3.0	9.0
6615003739254BF	.00080	1	1.0	5	0.416	1.00	0.	11.0
6610007998315	.00165	1	1.0	7	0.935	0.85	3.0	12.0
1660009091473	.00065	1	1.0	5	0.454	0.94	4.0	14.0
6610001506785	.00153	1	1.0	7	0.939	0.94	3.0	12.0
1650008369785BF	.00057	1	1.0	4	0.215	0.19	5.0	14.0
1430002193773BF	.00059	1	1.0	5	0.464	0.87	9.0	14.0
1430001747048BF	.01159	1	0.1	4	0.156	0.08	3.0	15.0
1430010039780BF	.00246	1	0.9	6	0.682	0.08	4.0	14.0
6110001871018BF	.00059	1	1.0	5	0.369	0.36	7.0	22.0
6615005905172BF	.00148	1	1.0	5	0.353	0.05	4.0	14.0
6610010347616	.00133	1	1.0	5	0.328	0.17	4.0	10.0
1430001330189BF	.00057	1	1.0	5	0.427	0.84	3.0	14.0
1650007906855BF	.00082	1	1.0	5	0.348	0.43	5.0	12.0
1430004100845BF	.00116	1	1.0	7	0.946	0.92	5.0	14.0
6610008831034	.00229	1	1.0	9	1.739	0.94	3.0	11.0
1280009338792NT	.00156	1	1.0	6	0.497	0.23	5.0	10.0
1270000231042	.00064	1	1.0	5	0.259	0.85	4.0	8.0
6605008365335	.01156	1	0.7	16	2.494	0.46	3.0	11.0
6110005717654BF	.00200	1	1.0	9	1.173	0.89	4.0	12.0
1430010039782BF	.00101	1	0.9	5	0.256	0.04	4.0	14.0
5865010169623EW	.00072	1	1.0	5	0.213	0.17	5.0	9.0
1430010039781BF	.00293	1	0.6	6	0.508	0.09	4.0	14.0
1660007935799	.00225	1	1.0	10	1.352	0.97	4.0	12.0
1660010215625	.00066	1	1.0	6	0.406	0.91	5.0	14.0
6610009539670	.00114	1	1.0	7	0.766	0.93	4.0	12.0
6685006845176	.00119	2	1.0	14	1.732	0.94	5.0	11.0
6605001113645	.00084	1	1.0	6	0.407	0.90	7.0	10.0
6610009942170	.00107	1	0.7	6	0.342	0.89	4.0	11.0
6605009876166	.00085	1	0.7	5	0.108	0.12	3.0	14.0
1280009338793NT	.00123	1	1.0	7	0.543	0.57	4.0	12.0
6110000978394BF	.00267	2	1.0	23	3.190	0.91	5.0	12.0
6610004546632BF	.00666	1	1.0	12	1.845	0.05	4.0	14.0
5865NC134683LEW	.00496	1	1.0	12	1.598	0.17	4.0	16.0
5841000738241	.00461	1	1.0	12	1.480	0.25	4.0	13.0
6615009825301	.00147	1	1.0	8	0.701	0.33	5.0	14.0
5895009190400	.00449	2	1.0	20	1.841	0.12	3.0	10.0
5895001688798	.00377	1	1.0	15	1.781	0.04	4.0	12.0
5895005205891	.00824	1	1.0	14	1.890	0.05	4.0	3.0
5831007825305	.00179	2	1.0	15	0.780	0.02	3.0	14.0
6615010159539BF	.00396	1	0.7	9	0.597	0.04	4.0	14.0
6610008451070	.00334	1	1.0	14	1.718	0.88	4.0	11.0
2920010139867YP	.00090	1	0.5	8	1.693	0.93	7.0	15.0

NSN	REMOVAL RATE	QPA	FAP	INITIAL STOCK	INITIAL NO. IN RESUPPLY	BASE NRTS	RESUPPLY TIMES (DAYS) BASE	DEPOT
5841000656743	.00935	1	1.0	18	2.173	0.09	4.0	14.0
6615000228011	.00271	1	1.0	11	0.946	0.25	4.0	11.0
6615000593851	.00608	1	1.0	26	4.642	0.92	4.0	14.0
5895008100140	.00909	1	1.0	19	2.309	0.06	4.0	17.0
6720001034963	.00066	1	0.5	10	0.234	0.19	5.0	40.0
6680006518045	.00332	1	1.0	18	2.510	0.94	6.0	11.0
5895009190410	.00163	2	0.1	22	0.069	0.11	2.0	22.0
5895007908764	.00558	1	1.0	18	1.397	0.08	4.0	16.0
5895008100189	.00973	1	1.0	24	2.854	0.08	4.0	15.0
2910010092822YP	.00164	1	0.5	16	4.726	0.96	7.0	15.3
1630008521432	.00068	2	1.0	58	0.741	0.02	10.0	28.0
6610009453112BF	.00352	1	1.0	42	20.708	0.95	3.0	13.0

APPENDIX C

PROGRAM LISTINGS

This appendix provides listings of the FORTRAN programs described below.

Sortie-Generation Model (SGM)

The SGM consists of a main program and 39 subroutines, functions, and block data subprograms. The main program is listed first. The subprograms, each beginning on a new page, are then listed in alphabetical order.

Scenario Input Program

This program provides an interactive interface to the SGM and is used to prepare a scenario parameter input file for an SGM run. The main program is listed first followed by the subprograms.

Plot Program

This program provides the graphs of SGM sortie results which appear in every SGM run. It produces graphs of the average sorties per aircraft per day and also total sorties flown per day.

SGM PROGRAM

```
C********** OS29/N232D/SGM/NEWSGM
C***************************************************************
C*** MAIN PROGRAM
C***************************************************************
C** MAIN       - MAIN PROGRAM FOR LMI SORTIE-GENERATION MODEL (SGM).
C***     THIS IS THE MAIN PROGRAM FOR THE LMI SORTIE GENERATION
C*** MODEL (SGM). A LIST OF ALL COMMON BLOCKS AND PARAMETER STATEMENTS
C*** USED IN THE MODEL IS PROVIDED AT THE BEGINNING OF THIS MAIN
C*** PROGRAM. THE PROCESSING SEQUENCE IS AS FOLLOWS - FIRST,
C*** ALL INPUTS ARE LOADED AND NECESSARY INITIALIZATION PERFORMED.
C*** THEN, THE ACTUAL SIMULATION IS PERFORMED, AND FINALLY THE RUN
C*** RESULTS ARE PRINTED TO THE STANDARD OUTPUT FILE.
C***
C*** INPUT FILES --
C***     01 - SCENARIO INPUT PARAMETERS
C***     02 - WORK CENTER INPUT DATA
C***     03 - SCRATCH FILE USED FOR DAILY SCENARIO PARAMETERS
C***     04 - SPARES INPUT DATA
C*** OUTPUT FILES --
C***     06 - STANDARD OUTPUT FILE (RUN RESULTS)
C***     07 - SORTIE RESULTS FOR PLOT PROGRAM
C*** PARAMETERS --
C***   MAXAC      - MAXIMUM ALLOWABLE UE-STRENGTH (# AIRCRAFT)
C***   MAXWC      - MAXIMUM ALLOWABLE NUMBER OF WORK CENTERS
C***   MAXBIT     - NUMBER OF BITS IN A COMPUTER WORD ON THIS SYSTEM
C***   MAXPRT     - MAXIMUM ALLOWABLE NUMBER OF PART-TYPES.
C***   MAXVEC     - MAXIMUM ALLOWABLE LENGTH (IN COMPUTER WORDS) OF
C***                AIRCRAFT BIT-VECTORS. A BIT-VECTOR MUST BE AT
C***                LEAST "MAXAC" BITS LONG, PLUS AN EXTRA WORD
C***                TO STORE THE AIRCRAFT COUNT FOR THAT VECTOR.
C***                HENCE, MAXVEC IS A FUNCTION OF MAXAC AND MAXBIT.
C***   MAXDAY     - MAXIMUM ALLOWABLE NUMBER OF FLYING DAYS.
C***   MAXCYC     - MAXIMUM ALLOWABLE NUMBER OF FLYING CYCLES PER DAY.
C***   MAXSTAT    - CURRENT NUMBER OF STATISTICS COLLECTED PER
C***                FLYING CYCLE PER DAY.
C***   LFLD       - LENGTH OF BIT-FIELD USED IN THE WORK-CENTER
C***                REPAIR LISTS. THIS BIT-FIELD MUST BE LARGE ENOUGH
C***                TO HOLD (MAXAC-1), THE TAIL NUMBER OF THE
C***                LAST AIRCRAFT. THUS, (2**LFLD) MUST BE GREATER
C***                THAN OR EQUAL TO MAXAC.
C***   NPERWRD    - NUMBER OF BIT-FIELDS PER COMPUTER WORD FOR THESE
C***                WORK-CENTER LISTS.THUS NPERWRD IS A FUNCTION
C***                OF LFLD AND MAXBIT.
C***   MXINWC     - LENGTH (IN COMPUTER WORDS) OF THE WORK-CENTER LISTS.
C***                MXINWC IS COMPUTED SO THAT THE MAXIMUM ALLOWABLE
C***                NUMBER OF BIT FIELDS IN A WORK-CENTER LIST IS
C***                EQUAL TO MAXAC, THE MAXIMUM NUMBER OF AIRCRAFT.
C***   IFSCEN     - FILE NUMBER OF SCENARIO INPUT FILE
C***   IFWC       - FILE NUMBER OF WORK CENTER INPUT FILE
C***   IFPRT      - FILE NUMBER OF SPARES INPUT FILE
C***************************************************************
C---
      PARAMETER  MAXAC=108,MAXWC=25,MAXBIT=36,MAXPRT=304,
```


PRECEDING PAGE BLANK-NOT FILMED

```
&                 MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      PARAMETER MAXDAY=30,MAXCYC=10,MAXSTAT=5
      PARAMETER IFSCEN=01, IFWC=02, IFPRT=04
      LOGICAL INFMAN,INFPART
C---
C---/ACSTATE/  - AIRCRAFT BIT-VECTORS.
      COMMON /ACSTATE/ LENGTH, NACVC(MAXVEC), IFLYVC(MAXVEC),
&                      MAINVC(MAXVEC), NORSVC(MAXVEC), LOSTVC(MAXVEC)
C---
C---/ALIASC/   - TABLES FOR PART-TYPE SAMPLING.
      COMMON /ALIASC/  FRACT(MAXPRT), IALIAS(MAXPRT), FPARTS
C---
C---/BITS/     - BIT MANIPULATION TABLES.
      COMMON /BITS/    MASK0,MASK(35), MLEFT0,MSKLFT(36),
&                      IZCOUT,ICOUNT(63)
C---
C---/DEMAND/   - MEAN AND VARIANCE FOR TOTAL PART DEMANDS.
      COMMON /DEMAND/  ACMEAN, ACVAR, NPERAC
C---
C---/INPUT/    - FLYING SCENARIO PARAMETERS.
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
C---
C---/PARTS/    - PART CHARACTERISTICS.
      COMMON /PARTS/   NPARTS, IQPA(MAXPRT), NBACK0(MAXPRT),
&                      BRPRATE(MAXPRT), DRPRATE(MAXPRT), INITSJ(MAXPRT),
&                      RESUPP(MAXPRT), BNRTS(MAXPRT), NBASE(MAXPRT),
&                      NDEPOT(MAXPRT)
C---
C---/RSEED/    - SEED FOR RANDOM NUMBER GENERATOR.
      COMMON /RSEED/   SEED
C---
C---/STATS/    - CUMULATIVE STATISTICS FOR SIMULATION RESULTS.
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                      NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
C---
C---/TIME/     - FLYING CYCLE TIMES AND SIMULATION PARAMETERS.
      COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
&                      NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE
C---
C---/WCBRK/    - WORK CENTER BREAK RATES.
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                      PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
C---
C---/WCINPUT/  - WORK CENTER INPUTS.
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
C---
C---/WCMAINT/  - AIRCRAFT WORK CENTER LISTS.
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
C---
C---  *COLLECT STARTING CPU-TIME AND CORE-MEMORY REQUIREMENT
       CALL PTIME(START)
```

```
      CALL MEMSIZ(KSIZE)
C---
C---  *LOAD AND INITIALIZE SCENARIO, WORK CENTER AND PARTS DATA
      CALL INIT(IFSCEN,IFWC,IFPRT)
C---
C---  *RUN THE ACTUAL SIMULATION
      CALL SIMULA
C---
C---  *PRINT-OUT THE RESULTS OF THE SIMULATION
      CALL PRINTO
C---
C---  *PRINT MEMORY AND CPU-TIME USED
      CALL PTIME(FINISH)
      WRITE(6,9001)(FINISH-START)*60.,KSIZE
C---
   STOP
 9001 FORMAT(//,"OCPU TIME USED =",F6.2," MIN",/,
&              "OMEMORY USED   =",I6," K WORDS")
   END
```

```
C***************************************************************
      SUBROUTINE ALIAS(N,FRACT,IALIAS)
C***************************************************************
C** ALIAS      - INITIALIZE TABLES NEEDED FOR "ALIAS" SAMPLING METHOD.
C***       ALIAS IS A FORTRAN SUBROUTINE WHICH INITIALIZES THE
C*** TABLES USED BY THE ALIAS METHOD FOR GENERATING RANDOM
C*** VARIABLES FROM A DISCRETE DISTRIBUTION. SEE - "ON THE
C*** ALIAS METHOD FOR GENERATING RANDOM VARIABLES FROM A DISCRETE
C*** DISTRIBUTION" IN THE AMERICAN STATISTICIAN, NOV 1979, VOL 33,
C*** NO 4, PP 214-218, FOR A DESCRIPTION OF THIS METHOD AND THE
C*** ALGORITHM USED IN THIS ROUTINE TO CREATE THE NECESSARY TABLES.
C*** TWO TABLES ARE NEEDED FOR THIS METHOD - A TABLE OF
C*** FRACTIONAL CUTOFF VALUES AND ANOTHER FOR THE CORRESPONDING
C*** ALIASES. THE PROCEDURE USED TO GENERATE THESE TABLES IS A
C*** SINGLE-PASS, LINKED-LIST PROCEDURE.
C***
C*** INPUT -
C***   N          - NUMBER OF MASS POINTS OF THE DISCRETE DISTRIBUTION
C***                WHICH IS BEING SAMPLED.
C*** INPUT/OUTPUT -
C***   FRACT(I)   - UPON INPUT, FRACT(I) IS THE PROBABILITY
C***                DISTRIBUTION OF A RANDOM VARIABLE, R.
C***                FRACT(I)=PROBABILITY( R = I), I=1,2,...,N .
C***                UPON OUTPUT FROM THIS SUBROUTINE, FRACT CONTAINS
C***                THE TABLE OF FRACTIONAL CUTOFF VALUES USED BY THE
C***                ALIAS METHOD.
C*** OUTPUT -
C***   IALIAS(I)  - TABLE OF ALIASES USED BY ALIAS METHOD. I=1,...,N
C***************************************************************
C---
      DIMENSION FRACT(N), IALIAS(N)
C---
C---      *INITIALIZE LIST HEADERS TO NO ENTRIES
          LHEAD = 0
          MHEAD = 0
C---
C---      *DO FOR(EACH POINT OF THE PROBABILITY DISTRIBUTION)
          FLOATN = FLOAT(N)
          DO 600  I=1,N
C---
C---         *INITIALIZE FRACTIONAL CUTOFF VALUE FOR THIS POINT
             FRACT(I) = FLOATN * FRACT(I)
C---
C---         *IF(THIS INDEX BELONGS IN THE "LESS" LIST, I.E. THOSE
C---              INDICES FOR WHICH FRACT(I) IS LESS THAN 1.0)THEN
             IF(FRACT(I).GE.1.0) GO TO 100
C---
C---            *ADD THIS INDEX TO HEAD OF "LESS" LIST
                IALIAS(I) = LHEAD
                LHEAD     = I
C---
C---         *ELSE (INDEX BELONGS IN "MORE" LIST)
             GO TO 200
```

```
 100        CONTINUE
C---
C---          *ADD INDEX TO HEAD OF "MORE" LIST
               IALIAS(I) = MHEAD
               MHEAD     = I
C---
C---        *END IF (WHICH LIST TEST)
 200         CONTINUE
C---
C---        *DO WHILE(BOTH LISTS ARE NOT EMPTY)
 300         CONTINUE
             IF(MHEAD.EQ.0) GO TO 500
             IF(LHEAD.EQ.0) GO TO 500
C---
C---          *REMOVE NEXT INDEX FROM "LESS" LIST
               LNEXT = LHEAD
               LHEAD = IALIAS(LHEAD)
C---
C---          *SET ALIAS FOR THIS INDEX TO NEXT ENTRY IN "MORE" LIST
               IALIAS(LNEXT) = MHEAD
C---
C---          *UPDATE CUTOFF VALUE FOR THIS "MORE" ENTRY
               FRACT(MHEAD) = FRACT(MHEAD) - (1.0-FRACT(LNEXT))
C---
C---          *IF(THIS INDEX NO LONGER BELONGS IN "MORE" LIST)THEN
               IF(FRACT(MHEAD).GE. 1.0) GO TO 400
C---
C---             *REMOVE INDEX FROM "MORE" LIST AND ADD IT TO "LESS"
                  LNEXT         = MHEAD
                  MHEAD         = IALIAS(MHEAD)
                  IALIAS(LNEXT) = LHEAD
                  LHEAD         = LNEXT
C---
C---          *END IF (SWITCH LISTS TEST)
 400            CONTINUE
C---
C---        *END DO (LISTS LOOP)
             GO TO 300
 500         CONTINUE
C---
C---      *END DO (INDEX LOOP)
 600       CONTINUE
C---
C---      *ADJUST FRACT(I) TO SAVE TIME IN MNOM SUBROUTINE.
           DO 700 I=1,N
              FRACT(I) = FRACT(I) + (I-1)
 700       CONTINUE
C---
      RETURN
      END
```

```
C*****************************************************************
      SUBROUTINE ATTRIT(PLOST,NTOFLY,IFLYVC,LOSTVC,NLOST)
C*****************************************************************
C++ ATTRIT     - SIMULATE ATTRITION PROCESS DURING A SORTIE PERIOD.
C***       ATTRIT IS A FORTRAN SUBROUTINE WHICH SIMULATES THE
C*** EFFECTS OF ATTRITION DURING A SORTIE. ATTRIT DRAWS A RANDOM
C*** SAMPLE FROM A BINOMIAL DISTRIBUTION BASED ON THE NUMBER OF
C*** AIRCRAFT FLYING THE SORTIE AND THE PROBABILITY OF ATTRITION
C*** GIVEN AN AIRCRAFT FLIES A SORTIE. ATTRIT THEN SELECTS THE
C*** RIGHTMOST AIRCRAFT FROM THE CURRENT FLYABLE AIRCRAFT VECTOR AS
C*** THE AIRCRAFT WHICH WERE ATTRITED. THE RIGHTMOST ONES ARE SELECTED
C*** TO SPEED UP COMPUTATION IN OTHER ROUTINES.
C***
C*** INPUT -
C***   PLOST      - PROBABILITY THAT AN AIRCRAFT ATTRITS GIVEN THAT IT
C***                FLIES A SORTIE.
C*** INPUT/OUTPUT -
C***   NTOFLY     - NO. OF A/C TO FLY THIS PERIOD.
C***   IFLYVC     - FLYABLE AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE STILL FLYABLE DURING THE CURRENT
C***                FLYING CYCLE. I.E. THOSE AIRCRAFT WHICH WERE FLYABLE
C***                AT THE START OF PREFLIGHT AND HAVE NOT GROUND-ABORTED,
C***                ATTRITED, OR BROKEN THUS FAR IN THE CYCLE.
C***                THE FIRST WORD, IFLYVC(1), CONTAINS THE TOTAL
C***                NUMBER OF AIRCRAFT STILL FLYABLE THUS FAR IN
C***                THE CURRENT FLYING CYCLE. THE REMAINDER OF THE
C***                ARRAY IS A BIT VECTOR WITH EACH BIT REPRESENTING
C***                AN AIRCRAFT. A 1-BIT INDICATES THE AIRCRAFT IS
C***                STILL FLYABLE. NOTE THAT IFLYVC(1) ALSO INDICATES
C***                THE NUMBER OF 1-BITS IN THIS BIT VECTOR.
C***   LOSTVC     - ATTRITED AIRCRAFT VECTOR. INDICATES THOSE AIR-
C***                CRAFT WHICH HAVE ATTRITED THUS FAR IN THE SIMULATION
C***                AND NOT BEEN REPLACED BY RESERVES. THE FIRST WORD,
C***                LOSTVC(1), CONTAINS THE TOTAL NUMBER OF AIRCRAFT
C***                WHICH HAVE BEEN LOST AND NOT REPLACED BY RESERVES.
C***                THE REMAINDER OF THE ARRAY IS A BIT VECTOR WITH
C***                EACH BIT REPRESENTING AN AIRCRAFT. A 1-BIT INDICATES
C***                THE AIRCRAFT HAS BEEN ATTRITTED. NOTE THAT
C***                LOSTVC(1) ALSO INDICATES THE NUMBER OF 1-BITS IN
C***                THIS BIT VECTOR.
C*** OUTPUT -
C***   NLOST      - NUMBER OF AIRCRAFT LOST ON THIS SORTIE
C*****************************************************************
C---
C---      *DETERMINE NUMBER OF ATTRITIONS BY SAMPLING FROM THE
C---                APPROPRIATE BINOMIAL DISTRIBUTION
          NLOST = NBINOM(PLOST,NTOFLY)
C---
C---      *IF(ANY AIRCRAFT WERE ATTRITED)THEN
          IF(NLOST .EQ. 0)  GO TO 1000
C---
C---         *REDUCE NO. OF A/C CAPABLE OF FLYING THIS PERIOD
             NTOFLY = NTOFLY - NLOST
```

```
C---
C---          *TRANSFER RIGHTMOST AIRCRAFT FROM FLYABLE AIRCRAFT
C---                VECTOR TO THE ATTRITED AIRCRAFT VECTOR
               CALL TBITSR(NLOST,IFLYVC,LOSTVC)
C---
C---        *END IF (ZERO ATTRITIONS TEST)
 1000        CONTINUE
C---
      RETURN
      END
```

AD-A110 899 LOGISTICS MANAGEMENT INST WASHINGTON DC F/G 15/5
THE SORTIE-GENERATION MODEL SYSTEM. VOLUME IV. SORTIE-GENERATIO--ETC(U)
SEP 81 M J KONVALINKA, J B ABELL MDA903-81-C-0166
UNCLASSIFIED LMI-ML102-VOL-4 NL

2 of 3
AD
A110899

```
C*****************************************************************
   BLOCK DATA
C*****************************************************************
C** BLOCK DATA - INITIALIZES COMMON TABLES FOR BIT MANIPULATIONS.
C***     THIS SUBPROGRAM INITIALIZES THE TABLES CONTAINED IN
C*** THE /BIT/ COMMON BLOCK.  THIS INITIALIZATION IS DONE
C*** DURING COMPILATION: THE SUBPROGRAM CONTAINS NO
C*** EXECUTABLE STATEMENTS. THE /BIT/ COMMON BLOCK CONTAINS
C*** THREE SETS OF TABLES WHICH ARE USED FOR ACCESSING BITS AND
C*** BIT FIELDS WITHIN A COMPUTER WORD. NOTE THAT THE FOLLOWING
C*** PROGRAMMING TECHNIQUE IS USED IN EACH OF THESE TABLES - AN
C*** EXTRA WORD IS PLACED BEFORE THE BEGINNING OF EACH TABLE. THIS
C*** EXTRA WORD REPRESENTS TABLE(0), I.E., THE 0TH INDEXED WORD IN
C*** THE TABLE. THUS , THE TABLE IS ACTUALLY INDEXED 0,1,2,...
C*** THIS TECHNIQUE OF REFERENCING THE 0TH WORD OF AN ARRAY IS
C*** NOT STANDARD FORTRAN AND MAY NOT WORK WITH OTHER FORTRAN COMPILERS.
C*** THESE TABLES REMAIN FIXED THROUGHOUT THE SIMULATION.
C***
C*** COMMON TABLES --
C***    MASK(I)    - I=0,1,...,35.  MASK IS THE BIT ACCESSING TABLE
C***                 USED IN THE SGM.  THE BITS IN THE COMPUTER WORD
C***                 ARE NUMBERED, LEFT TO RIGHT, 0,1,2,...,35,
C***                 AND MASK(I) HAS A 1-BIT IN THE ITH POSITION AND
C***                 ZEROES ELSEWHERE. THIS TABLE IS USED TO MASK-OFF
C***                 THE ITH BIT IN A COMPUTER WORD.
C***    MSKLFT(I)  - I=0,1,...,36 . MSKLFT IS USED TO MASK-OFF THE
C***                 LEFTMOST BITS IN A COMPUTER WORD. THE FIRST
C***                 (LEFTMOST) I BITS OF MSKLFT(I) ARE 1-BITS
C***                 AND THE REMAINING BITS ARE ZERO. THUS, FOR
C***                 EXAMPLE, MSKLFT(0) WOULD BE ALL 0S AND
C***                 MSKLFT(36) WOULD BE ALL 1S.
C***    ICOUNT(I)  - I=0,1,...,63.  THIS IS A TABLE WHICH IS USED TO
C***                 COUNT THE NUMBER OF 1-BITS IN ANY GIVEN 6-BIT
C***                 FIELD. IN A 6-BIT FIELD, THERE ARE 2**6 = 64
C***                 POSSIBLE BIT PATTERNS -- THE BINARY
C***                 REPRESENTATIONS OF THE INTEGERS 0,1,2,...63.
C***                 ICOUNT(I) CONTAINS THE NUMBER OF 1-BITS IN THE
C***                 BINARY REPRESENTATION OF I, E.G., ICOUNT(3)=2 .
C***                 THIS TABLE IS USEFUL IN COUNTING THE NUMBER OF
C***                 1-BITS REPRESENTING AIRCRAFT IN THE VARIOUS
C***                 AIRCRAFT-STATUS BIT-VECTORS. THIS TECHNIQUE
C***                 IS MUCH FASTER THAN A BIT-BY-BIT COUNT.
C*****************************************************************
C--
      COMMON /BITS/ MASK0,MASK(35), MLEFT0,MSKLFT(36),
     &                            IZCOUT,ICOUNT(63)
C--
      DATA MASK0/0400000000000 /
      DATA MASK/                 0200000000000, 0100000000000,
     &          040000000000 , 020000000000 , 010000000000 ,
     &          04000000000  , 02000000000  , 01000000000  ,
     &          0400000000   , 0200000000   , 0100000000   ,
     &          040000000    , 020000000    , 010000000    ,
```

```
&             04000000     , 02000000     , 01000000     ,
&             0400000      , 0200000      , 0100000      ,
&             040000       , 020000       , 010000       ,
&             04000        , 02000        , 01000        ,
&             0400         , 0200         , 0100         ,
&             040          , 020          , 010          ,
&             04           , 02           , 01           /
C---
      DATA MLEFT0/0/
      DATA MSKLFT/0400000000000, 0600000000000, 0700000000000,
&               0740000000000, 0760000000000, 0770000000000,
&               0774000000000, 0776000000000, 0777000000000,
&               0777400000000, 0777600000000, 0777700000000,
&               0777740000000, 0777760000000, 0777770000000,
&               0777774000000, 0777776000000, 0777777000000,
&               0777777400000, 0777777600000, 0777777700000,
&               0777777740000, 0777777760000, 0777777770000,
&               0777777774000, 0777777776000, 0777777777000,
&               0777777777400, 0777777777600, 0777777777700,
&               0777777777740, 0777777777760, 0777777777770,
&               0777777777774, 0777777777776, 0777777777777 /
C---
      DATA IZCOUT/0/
      DATA ICOUNT/    1,  1,  2,  1,  2,  2,  3,  1,  2,  2,
&                     3,  2,  3,  3,  4,  1,  2,  2,  3,  2,
&                     3,  3,  4,  2,  3,  3,  4,  3,  4,  4,
&                     5,  1,  2,  2,  3,  2,  3,  3,  4,  2,
&                     3,  3,  4,  3,  4,  4,  5,  2,  3,  3,
&                     4,  3,  4,  4,  5,  3,  4,  4,  5,  4,
&                     5,  5,  6  /
C---
     END
```

```
C*****************************************************************
      SUBROUTINE BREAK(PBREAK,PBRKSEQ,INDXWC,NTOFLY,IFLYVC,NORSVC)
C*****************************************************************
C++ BREAK      - SIMULATE AIRCRAFT BREAKS AFTER A SORTIE.
C***     BREAK IS A FORTRAN SUBROUTINE WHICH SIMULATES THE PROCESS
C*** OF AIRCRAFT BREAKING UPON RETURNING FROM A SORTIE. GIVEN A NUMBER
C*** OF FLYABLE AIRCRAFT AND THE OVERALL BREAK RATE, THIS ROUTINE FIRST
C*** DETERMINES THE NUMBER OF AIRCRAFT WHICH BROKE. IT THEN DETERMINES
C*** THE NUMBER AND DISTRIBUTION OF PARTS DEMANDS RESULTING FROM THESE
C*** BREAKS. THEN ASSUMING IMMEDIATE AND MAXIMUM CANNABILIZATION, THE
C*** NUMBER OF NORS AIRCRAFT AMONG THESE BROKEN AIRCRAFT IS DETERMINED.
C*** THOSE AIRCRAFT WHICH ARE NOT NORS ARE PROBABILISTICALLY BROKEN
C*** DIRECTLY INTO THE VARIOUS WORKCENTERS.
C***
C*** INPUT -
C***   PBREAK     - PROBABILITY THAT A FLYABLE AIRCRAFT BREAKS
C***                INTO AT LEAST ONE WORKCENTER UPON RETURNING
C***                FROM A SORTIE.
C***   PBRKSEQ    - 2-DIMENSIONAL ARRAY USED TO DETERMINE THE
C***                DISTRIBUTION OF ABORTS INTO THE VARIOUS
C***                WORKCENTERS.
C***   INDXWC     - AN INDEX ARRAY USED TO DETERMINE THE DISTRIBUTION
C***                OF BREAKS INTO THE VARIOUS WORK CENTERS.
C*** INPUT/OUTPUT -
C***   NTOFLY     - NO. OF A/C TO FLY THIS PERIOD.
C***   IFLYVC     - FLYABLE AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE STILL FLYABLE DURING THE CURRENT
C***                FLYING CYCLE. I.E. THOSE AIRCRAFT WHICH WERE FLYABLE
C***                AT THE START OF PREFLIGHT AND HAVE NOT GROUND-ABORTED,
C***                ATTRITED, OR BROKEN THUS FAR IN THE CYCLE.
C***                THE FIRST WORD, IFLYVC(1), CONTAINS THE TOTAL
C***                NUMBER OF AIRCRAFT STILL FLYABLE THUS FAR IN
C***                THE CURRENT FLYING CYCLE. THE REMAINDER OF THE
C***                ARRAY IS A BIT VECTOR WITH EACH BIT REPRESENTING
C***                AN AIRCRAFT. A 1-BIT INDICATES THE AIRCRAFT IS
C***                STILL FLYABLE. NOTE THAT IFLYVC(1) ALSO INDICATES
C***                THE NUMBER OF 1-BITS IN THIS BIT VECTOR.
C***   NORSVC     - NORS AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE NORS DUE TO UNAVAILABLE PARTS.
C***                THE FIRST WORD, NORSVC(1), CONTAINS THE TOTAL
C***                NUMBER OF 1-BITS IN THE NORS STATUS VECTOR.
C***                THE REMAINDER OF THE ARRAY IS A BIT STRING WITH
C***                EACH BIT REPRESENTING AN AIRCRAFT. A 1 INDICATES
C***                THE AIRCRAFT IS NORS. NOTE THAT NORSVC(1) ALSO
C***                INDICATES THE NUMBER OF 1-BITS IN THIS BIT STRING.
C*****************************************************************
C---
      DIMENSION NORSVC(1)
C---
C---        *IF(THERE ARE STILL ANY FLYABLE AIRCRAFT)THEN
             IF(NTOFLY.EQ.0) GO TO 4000
C---
C---           *DETERMINE NUMBER OF AIRCRAFT BREAKING INTO WORKCENTERS
```

```
C---              BY SAMPLING FROM THE APPROPRIATE BINOMIAL DISTRIBUTION
               NTOTBK = NBINOM(PBREAK,NTOFLY)
C---
C---         *IF(THERE ARE ANY BROKEN AIRCRAFT)THEN
               IF(NTOTBK.EQ.0) GO TO 3000
C---
C---            *REDUCE NO. OF A/C CAPABLE OF FLYING THIS PERIOD
                 NTOFLY = NTOFLY - NTOTBK
C---
C---            *DETERMINE NUMBER/DISTRIBUTION OF PARTS DEMANDS RESULTING
C---             FROM THESE BROKEN AIRCRAFT AND DETERMINE NEW NUMBER OF
C---             NORS AIRCRAFT AFTER IMMEDIATE AND MAXIMUM CANNABILIZATION
                 NEWNOR = NORSBK(NTOTBK,NORSVC(1))
                 NORDIF = NEWNOR - NORSVC(1)
C---
C---            *IF(NOT ALL THE BROKEN AIRCRAFT ARE NORS)THEN
                 IF(NORDIF.GE.NTOTBK) GO TO 1000
C---
C---               *BREAK THE LEFTMOST FLYABLE AIRCRAFT INTO MAINTENANCE
                    CALL WCDIST(NTOTBK-NORDIF,PBRKSEQ,INDXWC,IFLYVC)
C---
C---            *END IF (ALL NORS TEST)
 1000            CONTINUE
C---
C---            *IF(SOME OF THE BROKEN AIRCRAFT ARE NORS)THEN
                 IF(NORDIF.LE.0) GO TO 2000
C---
C---               *TRANSFER LEFTMOST AIRCRAFT FROM FLYABLE STATUS
C---                          VECTOR TO THE NORS STATUS VECTOR
                    CALL TBITSL(NORDIF,IFLYVC,NORSVC)
C---
C---            *END IF (NONZERO NORS TEST)
 2000            CONTINUE
C---
C---         *END IF (ZERO BREAKS TEST)
 3000          CONTINUE
C---
C---       *END IF (ZERO FLYABLE AIRCRAFT TEST)
 4000        CONTINUE
C---
  RETURN
  END
```

```
C*****************************************************************
   SUBROUTINE CRESERV(IAUGMNT,LOSTVC,NRESRV,NAC)
C*****************************************************************
C++ CRESERV    - COMMIT RESERVE AIRCRAFT.
C***    CRESERV IS A FORTRAN SUBROUTINE WHICH WILL ALLOCATE
C*** RESERVE AIRCRAFT TO REPLACE THOSE AIRCRAFT WHICH HAVE BEEN
C*** LOST DUE TO ATTRITION. IF ENOUGH RESERVES ARE LEFT, ALL LOSSES
C*** ARE REPLACED; HENCE THE ATTRITION VECTOR IS ZEROED OUT. IF
C*** THERE ARE NOT ENOUGH RESERVES TO COVER ALL THE LOSSES, THEN
C*** THE ATTRITIONS ON THE LEFT OF THE ATTRITION VECTOR ARE
C*** REPLACED. THIS ARBITRARY SELECTION WILL HELP TO SPEED UP THE
C*** SELECTION ROUTINES. NOTE THAT ALL RESERVES ARE ASSUMED TO
C*** BE FULLY MISSION CAPABLE (I.E. FLYABLE) WHEN COMMITTED.
C***
C*** INPUT --
C***   IAUGMNT    - FLAG INDICATING WHETHER RESERVES ARE TO BE USED
C***                ONLY AS ATTRITION FILLERS OR TO AUGMENT THE
C***                CURRENT UE-STRENGTH. IF IAUGMNT=0, RESERVES
C***                ARE USED ONLY TO REPLACE COMBAT LOSSES; HENCE
C***                NOT ALL RESERVES MAY BE COMMITTED WHEN THEY
C***                BECOME AVAILABLE. IF IAUGMNT=1, ALL RESERVES
C***                ARE COMMITTED IMMEDIATELY UPON BECOMING
C***                AVAILABLE. THIS FLAG IS A USER-SPECIFIED INPUT
C***                WHICH REMAINS FIXED THROUGHOUT THE SIMULATION.
C*** INPUT/OUTPUT --
C***   LOSTVC     - ATTRITED AIRCRAFT VECTOR. INDICATES THOSE AIR-
C***                CRAFT WHICH HAVE ATTRITED THUS FAR IN THE SIMULATION
C***                AND NOT BEEN REPLACED BY RESERVES. THE FIRST WORD,
C***                LOSTVC(1), CONTAINS THE TOTAL NUMBER OF AIRCRAFT
C***                WHICH HAVE BEEN LOST AND NOT REPLACED BY RESERVES.
C***                THE REMAINDER OF THE ARRAY IS A BIT VECTOR WITH
C***                EACH BIT REPRESENTING AN AIRCRAFT. A 1-BIT INDICATES
C***                THE AIRCRAFT HAS BEEN ATTRITTED. NOTE THAT
C***                LOSTVC(1) ALSO INDICATES THE NUMBER OF 1-BITS IN
C***                THIS BIT VECTOR.
C***   NRESRV     - NUMBER OF AIRCRAFT CURRENTLY IN RESERVE.
C***   NAC        - CURRENT UE-STRENGTH. IF THE AUGMENT FLAG
C***                IS SET, THE RESERVES WHICH REMAIN AFTER
C***                REPLACING COMBAT LOSSES WILL BE USED TO
C***                AUGMENT THIS CURRENT UE-STRENGTH. IF THE
C***                FLAG IS NOT SET OR THERE ARE NOT ENOUGH
C***                RESERVES TO COVER ALL THE COMBAT LOSSES, THEN
C***                NAC WILL REMAIN UNCHANGED.
C*****************************************************************
C---
     DIMENSION LOSTVC(1)
C---
C--- *REPLACE AS MANY AIRCRAFT LOSSES AS POSSIBLE
     NFILLS = MIN0(LOSTVC(1),NRESRV)
     CALL ZBITSL(NFILLS,LOSTVC)
     NRESRV = NRESRV - NFILLS
C---
C--- *IF(THERE ARE STILL REMAINING RESERVES AND EXCESS RESERVES
```

```
C---                              ARE TO BE AUGMENTED)THEN
      IF(NRESRV.LE.0)GO TO 100
      IF(IAUGMNT.EQ.0)GO TO 100
C---
C---     *INCREASE UE BY AUGMENTING REMAINING RESERVES
         NAC = NAC + NRESRV
         CALL UEUPDAT(NAC)
C---
C---     *SET REMAINING RESERVES TO NONE
         NRESRV = 0
C---
C--- *END IF (AUGMENTATION TEST)
  100 CONTINUE
C---
      RETURN
      END
```

```
C*****************************************************************
   SUBROUTINE FLYCYC
C*****************************************************************
C++ FLYCYC     - SIMULATE AIRCRAFT FLYING CYCLE.
C***     THIS ROUTINE IS THE BASIC LOGICAL STRUCTURE OF THE SGM
C*** SIMULATION. THE VARIOUS DISCRETE EVENTS WHICH CAN OCCUR,
C*** GROUND-ABORTS, AIRCRAFT REPAIRS, BREAKS, ETC., ARE
C*** STRUCTURED ACCORDING TO A USER-SPECIFIED FLYING CYCLE
C*** CONSISTING OF A MINIMAL-RECOVERY PERIOD, A SORTIE-PERIOD,
C*** AND EITHER A WAIT OR AN OVERNIGHT PERIOD. A SPECIFIED SEQUENCE
C*** OF THESE FLYING CYCLES COMPRISE A FLYING DAY, AND THE
C*** SIMULATION CONSISTS OF A SEQUENCE OF FLYING DAYS.
C***     THE NUMEROUS INPUTS AND OUTPUTS OF THIS ROUTINE ARE
C*** ALL CONTAINED IN COMMON BLOCKS. DEFINITIONS ARE PROVIDED
C*** IN THE VARIOUS ROUTINES CALLED BY FLYCYC AND WILL NOT
C*** BE REPEATED HERE.
C*****************************************************************
C---
      PARAMETER  MAXAC=108,MAXWC=25,MAXBIT=36,
&                 MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER MAXDAY=30,MAXCYC=10,MAXSTAT=5
      LOGICAL INFMAN,INFPART
      COMMON /ACSTATE/ LENGTH, NACVC(MAXVEC), IFLYVC(MAXVEC),
&                      MAINVC(MAXVEC), NORSVC(MAXVEC), LOSTVC(MAXVEC)
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                      NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
      COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
&                      NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                      PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
C---
C---  *UPDATE OVERALL MAINTENANCE BIT-VECTOR FOR THIS FLYING CYCLE
       CALL MUPDATE(NWC,MAINVC)
C---
C---  *COMPUTE NUMBER OF FULLY-MISSION-CAPABLE AIRCRAFT AND NUMBER
C---             OF AIRCRAFT TO SCHEDULE FOR NEXT SORTIE
C---   (A MISSION-CAPABLE AIRCRAFT IS DEFINED AS ONE NOT IN
C---    MAINTENANCE, NORS, OR COMBAT LOSS STATES)
       DO 100 L=2,LENGTH
          IFLYVC(L) = XOR (NACVC(L),LOSTVC(L),MAINVC(L),NORSVC(L))
  100  CONTINUE
       IFLYVC(1) = N1VECT (IFLYVC)
       NTOFLY = MIN0 (MAXFLY(ICYCLE),IFLYVC(1))
C---
C---  *AIRCRAFT-REPAIR EVENT -- DETERMINE AIRCRAFT REPAIRED IN
C---         EACH WORK-CENTER DURING MINIMAL-RECOVERY PERIOD
       CALL REPAIR(PREFLITE,NWC,NCREWS,SRATE)
C---
C---  *GROUND-ABORT EVENT -- DETERMINE NUMBER OF GROUND ABORTS
       CALL GABORT(PACGABT,PBRKSEQ,INDXWC,NTOFLY,IFLYVC)
```

```
 ---
C--  *STATISTICS EVENT -- UPDATE CUMULATIVE STATISTICS
C--       (STATISTICS ARE ALWAYS COLLECTED AT THE BEGINNING
C--       OF THE SORTIE PERIOD.  THE MAINTENANCE STAT
C--       REFLECTS ONLY THOSE AIRCRAFT IN MAINTENANCE AT THE
C--       START OF THE MINIMAL-RECOVERY PERIOD AND DOES
C--       NOT ACCOUNT FOR GROUND-ABORTS OR REPAIRS DURING
C--       THIS MINIMAL RECOVERY PERIOD)
      EXPECT(1,ICYCLE,IDAY) = EXPECT(1,ICYCLE,IDAY) + NTOFLY
      EXPECT(2,ICYCLE,IDAY) = EXPECT(2,ICYCLE,IDAY) + MAINVC(1)
      EXPECT(3,ICYCLE,IDAY) = EXPECT(3,ICYCLE,IDAY) + NORSVC(1)
      EXPECT(4,ICYCLE,IDAY) = EXPECT(4,ICYCLE,IDAY) + LOSSTOT
      EXPECT(5,ICYCLE,IDAY) = EXPECT(5,ICYCLE,IDAY) + NRESRV
C--
C--  *IF(THIS IS NOT THE LAST CYCLE OF THE LAST DAY)THEN
C--          (NONE OF THE FOLLOWING WORK WILL AFFECT
C--           THE OUTPUT RESULTS IF THIS IS THE LAST SORTIE)
      IF((ICYCLE.GE.NCYCLES).AND.(IDAY.GE.NUMDAY))GO TO 400
C--
C--    *ATTRITION EVENT -- DETERMINE NUMBER OF ATTRITED AIRCRAFT
        CALL ATTRIT(PATTRIT,NTOFLY,IFLYVC,LOSTVC,NLOST)
        LOSSTOT = LOSSTOT + NLOST
C--
C--    *REPAIR EVENT -- DETERMINE NUMBER OF AIRCRAFT REPAIRED IN EACH
C--         WORK CENTER DURING THE SORTIE PERIOD
        CALL REPAIR(SORTLGTH,NWC,NCREWS,SRATE)
C--
C--    *IF(IT IS NOT THE LAST SORTIE OF THE FLYING DAY)THEN
        IF(ICYCLE.EQ.NCYCLES) GO TO 200
C--
C--       *BREAK-EVENT -- DETERMINE AIRCRAFT BREAKS AND PART DEMANDS
           CALL BREAK(PACBRK,PBRKSEQ,INDXWC,NTOFLY,IFLYVC,NORSVC)
C--
C--       *AIRCRAFT-REPAIR EVENT --DETERMINE AIRCRAFT REPAIRED IN
C--            EACH WORK-CENTER DURING THE WAIT PERIOD
           CALL REPAIR(WAITCYC,NWC,NCREWS,SRATE)
C--
C--    *ELSE (THIS IS THE OVERNIGHT PERIOD)
        GO TO 300
  200   CONTINUE
C--
C--       *PARTS-REPAIR EVENT -- DETERMINE SPARE PARTS REPAIRED
C--                  IN THE LAST 24-HOUR PERIOD
           CALL PRTREP(24.,PBRKSEQ,INDXWC,NORSVC)
C--
C--       *BREAK-EVENT -- DETERMINE AIRCRAFT BREAKS AND PART DEMANDS
           CALL BREAK(PACBRK,PBRKSEQ,INDXWC,NTOFLY,IFLYVC,NORSVC)
C--
C--       *AIRCRAFT-REPAIR EVENT - DETERMINE OVERNIGHT AIRCRAFT REPAIRS
           CALL REPAIR(TYMNITE,NWC,NCREWS,SRATE)
C--
C--       *COMMIT-RESERVES EVENT -- BRING-IN AVAILABLE FULLY-MISSION-
C--            CAPABLE RESERVE AIRCRAFT TO REPLACE ANY COMBAT LOSSES
```

```
           CALL CRESERV(IAUGMNT,LOSTVC,NRESRV,NAC)
C---
C---     *END IF (OVERNIGHT PERIOD TEST)
  300     CONTINUE
C---
C---  *END IF(LAST-CYCLE-OF-LAST-DAY TEST)
  400  CONTINUE
C---
      RETURN
      END
```

```
C*****************************************************************
      SUBROUTINE GABORT(PABORT,PBRKSEQ,INDXWC,NTOFLY,IFLYVC)
C*****************************************************************
C++ GABORT     - SIMULATE AIRCRAFT GROUND-ABORT PROCESS.
C***    GABORT IS A FORTRAN SUBROUTINE WHICH SIMULATES THE PROCESS
C*** OF AIRCRAFT GROUND-ABORTING INTO WORKCENTERS AT THE END OF PREFLIGHT.
C*** GIVEN A NUMBER OF FLYABLE AIRCRAFT AND THE OVERALL GROUND-ABORT
C*** RATE, THIS ROUTINE CALCULATES THE TOTAL NUMBER OF GROUND-ABORTS,
C*** AND THEN DETERMINES WHICH WORKCENTERS THESE AIRCRAFT BROKE INTO.
C***
C*** INPUT -
C***   PABORT     - PROBABILITY THAT A FLYABLE AIRCRAFT GROUND-ABORTS
C***                INTO AT LEAST ONE WORKCENTER DURING PREFLIGHT.
C***   PBRKSEQ    - 2-DIMENSIONAL ARRAY USED TO DETERMINE THE
C***                DISTRIBUTION OF ABORTS INTO THE VARIOUS
C***                WORKCENTERS.
C***   INDXWC     - AN INDEX ARRAY USED TO DETERMINE THE DISTRIBUTION
C***                OF BREAKS INTO THE VARIOUS WORK CENTERS.
C*** INPUT/OUTPUT -
C***   NTOFLY     - NO. OF A/C TO FLY THIS PERIOD.
C***   IFLYVC     - FLYABLE AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE STILL FLYABLE DURING THE CURRENT
C***                FLYING CYCLE. I.E. THOSE AIRCRAFT WHICH WERE FLYABLE
C***                AT THE START OF PREFLIGHT AND HAVE NOT GROUND-ABORTED,
C***                ATTRITED, OR BROKEN THUS FAR IN THE CYCLE.
C***                THE FIRST WORD, IFLYVC(1), CONTAINS THE TOTAL
C***                NUMBER OF AIRCRAFT STILL FLYABLE THUS FAR IN
C***                THE CURRENT FLYING CYCLE. THE REMAINDER OF THE
C***                ARRAY IS A BIT VECTOR WITH EACH BIT REPRESENTING
C***                AN AIRCRAFT. A 1-BIT INDICATES THE AIRCRAFT IS
C***                STILL FLYABLE. NOTE THAT IFLYVC(1) ALSO INDICATES
C***                THE NUMBER OF 1-BITS IN THIS BIT VECTOR.
C*****************************************************************
C---
C---       *DETERMINE NUMBER OF AIRCRAFT GROUND-ABORTING INTO WORKCENTERS
C---            BY SAMPLING FROM THE APPROPRIATE BINOMIAL DISTRIBUTION
           NTOTBK = NBINOM(PABORT,NTOFLY)
C---
C---       *IF(THERE ARE ANY BROKEN AIRCRAFT)THEN
            IF(NTOTBK.EQ.0) GO TO 1000
C---
C---          *REDUCE NO. OF A/C CAPABLE OF FLYING THIS PERIOD
               NTOFLY = NTOFLY - NTOTBK
C---
C---          *BREAK THE LEFTMOST FLYABLE AIRCRAFT INTO MAINTENANCE
               CALL WCDIST(NTOTBK,PBRKSEQ,INDXWC,IFLYVC)
C---
C---       *END IF (ZERO BREAKS TEST)
 1000      CONTINUE
C---
      RETURN
      END
```

```
C*********************************************************************
   SUBROUTINE INIT(IFSCEN,IFWC,IFPRT)
C*********************************************************************
C++ INIT        - INITIALIZE SGM SIMULATION.
C***    THIS ROUTINE READS AND INITIALIZES THE VARIABLES FOR AN
C*** SGM RUN.  IT PERFORMS THE FOLLOWING SERIES OF STEPS -
C*** 1) LOAD AND SET THE VARIOUS PARAMETERS DESCRIBING THE RUN SCENARIO,
C*** 2) LOAD AIRCRAFT MAINTENANCE MANPOWER INPUTS, 3) LOAD
C*** SPARE PARTS INFORMATION, AND 4) SET MISCELLANEOUS PARAMETERS
C*** FOR MODEL USE.  EACH INPUT FILE IS CLOSED IMMEDIATELY AFTER
C*** ALL OF ITS INFORMATION HAS BEEN READ.
C***
C*** INPUTS --
C***   IFSCEN     - INPUT FILE CONTAINING SCENARIO PARAMETERS
C***   IFWC       - INPUT FILE CONTAINING MAINTENANCE MANPOWER INPUTS
C***   IFPRT      - INPUT FILE CONTAINING SPARE PARTS DATA
C*** COMMON INPUTS --
C***   INFPART    - LOGICAL FLAG INDICATING WHETHER INFINITE PARTS
C***                ASSUMPTION HOLDS.
C***   INFMAN     - LOGICAL FLAG INDICATING WHETHER INFINITE MANPOWER
C***                ASSUMPTION IS BEING MADE.
C***   SORTLGTH   - LENGTH (IN HOURS) OF EACH SORTIE.
C***   PACBRK     - AIRCRAFT BREAK RATE.
C***   NAC        - CURRENT UE STRENGTH.
C*** COMMON OUTPUT --
C***   EXPECT(I,J,K) - CUMULATIVE STATISTICS ARRAY.
C*********************************************************************
C---
      PARAMETER MAXCYC=10, MAXSTAT=5, MAXDAY=30, MAXWC=25
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                      NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
      COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
&                      NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                      PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
C---
C---  *LOAD AND SET SCENARIO INPUT PARAMETERS
       CALL INITSCN(IFSCEN)
C---
C---  *LOAD AND INITIALIZE SPARE-PARTS DATA
       CALL INITPRT(IFPRT,INFPART,SORTLGTH,PACBRK)
C---
C---  *READ AIRCRAFT MAINTENANCE MANPOWER INPUTS
       CALL INITWC(IFWC,INFMAN,NAC)
C---
C---  *ZERO-OUT CUMULATIVE-STATISTICS ARRAY
       CALL ZERO(EXPECT,MAXSTAT*MAXCYC*MAXDAY)
C---
   RETURN
   END
```

```
C*****************************************************************
      SUBROUTINE INITBO(NAC)
C*****************************************************************
C++ INITBO     - INITIALIZE PARTS IN RESUPPLY AT START OF SIMULATION.
C***      INITBO INITIALIZES THE NUMBER OF BACKORDERS
C*** FOR EACH PART TYPE AT THE START OF EACH SIMULATION
C*** REPLICATION. FOR EACH PART TYPE, A RANDOM SAMPLE IS DRAWN
C*** FROM THE APPROPRIATE POISSON DISTRIBUTION TO DETERMINE THE
C*** NUMBER IN RESUPPLY AND THE NUMBER OF BACKORDERS IS COMPUTED
C*** USING THIS RESUPPLY NUMBER AND THE INITIAL STOCK
C*** LEVEL. THE MEAN OF THE POISSON FOR EACH PART IS THE
C*** PIPELINE FOR EACH TYPE. "NAC" INDICATES NUMBER OF ON-HAND
C*** AIRCRAFT AT THE START OF THE SIMULATION.
C***
C*** INPUTS --
C***   NAC         - CURRENT UE-STRENGTH.
C*** COMMON INPUTS --
C***   NPARTS      - NUMBER OF PART-TYPES BEING MODELED.
C***   RESUPP(K)   - (K=1,..,NPARTS) EXPECTED NUMBER OF TYPE-K PARTS
C***                 IN RESUPPLY AT THE START OF THE SCENARIO. USED
C***                 AS THE MEAN OF A POISSON DISTRIBUTION TO GENERATE
C***                 A SAMPLE OF TYPE-K PARTS INITIALLY IN RESUPPLY.
C***   INITSJ(K)   - (K=1,...,NPARTS) INITIAL BASE STOCK LEVEL OF KTH
C***                 PART TYPE.
C***   IQPA(K)     - (K=1,...,NPARTS) QPA OF KTH PART TYPE.
C***   BNRTS(K)    - (K=1,...,NPARTS) BASE-NOT-REPAIRED-THIS-STATION
C***                 RATE. INDICATES PROPORTION OF TYPE-K FAILURES
C***                 WHICH ARE REPAIRED AT THE BASE.
C*** COMMON OUTPUTS --
C***   NBACKO(K)   - (K=1,...,NPARTS) NUMBER OF BACKORDERS FOR KTH
C***                 PART-TYPE. BACKORDERS ARE DEFINED AS
C***                 (# IN RESUPPLY)-(INITIAL STOCK LEVEL)
C***   NBASE(K)    - (K=1,...,NPARTS) NUMBER OF TYPE-K PARTS IN BASE
C***                 RESUPPLY.
C***   NDEPOT(K)   - (K=1,...,NPARTS) NUMBER OF TYPE-K PARTS IN DEPOT
C***                 RESUPPLY.
C*****************************************************************
C---
      PARAMETER MAXPRT=304
      COMMON/RSEED/ SEED
      COMMON /PARTS/ NPARTS,IQPA(MAXPRT),NBACKO(MAXPRT),
&        BRPRATE(MAXPRT),DRPRATE(MAXPRT),INITSJ(MAXPRT),RESUPP(MAXPRT),
&        BNRTS(MAXPRT),NBASE(MAXPRT),NDEPOT(MAXPRT)
C---
C---      *DO FOR(EACH PART TYPE)
          DO 200 K=1,NPARTS
C---
C---         *DRAW SAMPLE FROM POISSON DISTRIBUTION FOR NUMBER OF
C---                          PARTS IN RESUPPLY
             NRESUPP=IPOISSON(RESUPP(K),SEED)
C---
C---         *COMPUTE INITIAL BACKORDERS
             NBACKO(K)=NRESUPP - INITSJ(K)
```

```
C---
C---          *IF (IF BACKORDERS GREATER THAN PARTS ON-HAND)
               IF(NBACKO(K).LE.NAC*IQPA(K)) GOTO 100
C---
C---             *PRINT WARNING MESSAGE AND TRUNCATE NBACKO(K)
                  WRITE(6,9001)
&                 K,NBACKO(K),NAC,IQPA(K),NRESUPP,INITSJ(K),RESUPP(K)
                  NBACKO(K)=NAC*IQPA(K)
                  NRESUPP=NBACKO(K)+INITSJ(K)
C---
C---          *END IF(TRUNCATE BACKORDERS AT MAXIMUM AVAILABLE)
 100           CONTINUE
C---
C---          *ALLOCATE THESE PARTS BETWEEN BASE AND DEPOT RESUPPLY
               RBASE=0.0
               IF(BRPRATE(K).GT.0.0) RBASE=(1-BNRTS(K))/BRPRATE(K)
               RDEPOT=0.0
               IF(DRPRATE(K).GT.0.0) RDEPOT=BNRTS(K)/DRPRATE(K)
               NDEP=NBINOM(RDEPOT/(RBASE+RDEPOT),NRESUPP)
               NDEPOT(K)=NDEPOT(K)+NDEP
               NBASE(K)=NBASE(K)+NRESUPP-NDEP
C---
C---       *END DO (PARTS LOOP)
  200       CONTINUE
C---
   RETURN
 9001 FORMAT("0$$$$$$$$ INITBO ERROR - TOO MANY PARTS IN RESUPPLY",/,
& " $$$$$$$$     K=",I3,"  NBACKO(K)=",I5,"  NAC=",I3,"  IQPA(K)=",I3,
& /," $$$$$$$$      NRESUPP, INITSJ(K), RESUPP(K) = ",2I5,F10.3)
   END
```

```
C******************************************************************
   SUBROUTINE INITPRT(IFILE,INFPART,SORTLGTH,PACBRK)
C******************************************************************
C++ INITPRT    - LOAD AND INITIALIZE SPARE-PARTS DATA.
C***     THIS ROUTINE LOADS THE SPARE-PARTS INPUT DATA AND
C*** INITIALIZES THE STATISTICS AND TABLES NEEDED FOR
C*** SAMPLING TOTAL PART DEMANDS AND ALSO DETERMINING PART-TYPE
C*** FOR A GIVEN BROKEN PART.
C******************************************************************
C---
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER MAXPRT=304
      LOGICAL INFPART
      CHARACTER CNSN*18
      COMMON /PARTS/ NPARTS,IQPA(MAXPRT),NBACKO(MAXPRT),
&        BRPRATE(MAXPRT),DRPRATE(MAXPRT),INITSJ(MAXPRT),RESUPP(MAXPRT),
&        BNRTS(MAXPRT),NBASE(MAXPRT),NDEPOT(MAXPRT)
      COMMON /ALIASC/  FRACT(MAXPRT),IALIAS(MAXPRT),FPARTS
      COMMON /DEMAND/  ACMEAN, ACVAR, NPERAC
C---
C---  *IF(INFINITE PARTS ARE NOT ASSUMED, I.E. NORS AIRCRAFT ARE
C---       TO BE MODELED)THEN
       IF(INFPART)GO TO 900
C---
C---     *READ-IN PARTS DATA AND PERFORM ERROR CHECKS
         NPARTS=1
 100     CONTINUE
         READ(IFILE,END=200) CNSN,FRACT(NPARTS),IQPA(NPARTS),FAP,
&           INITSJ(NPARTS),RESUPP(NPARTS),BNRTS(NPARTS),BDAYS,DDAYS
            IF((FRACT(NPARTS).GT.0.0).AND.(FRACT(NPARTS).LE.1.0))
&                                                GO TO 50
            IF(IQPA(NPARTS).GT.0)GO TO 50
            IF((FAP.GT.0.0).AND.(FAP.LE.1.0))GO TO 50
            IF(INITSJ(NPARTS).GE.0)GO TO 50
            IF(RESUPP(NPARTS).GE.0.0)GO TO 50
            IF((BNRTS(NPARTS).GE.0.0).AND.(BNRTS(NPARTS).LE.1.0))
&                                 GO TO 50
            IF(BDAYS.GE.0.0)GO TO 50
            IF(DDAYS.GE.0.0)GO TO 50
               WRITE(6,9003) CNSN,FRACT(NPARTS),IQPA(NPARTS),FAP,
&              INITSJ(NPARTS),RESUPP(NPARTS),BNRTS(NPARTS),BDAYS,DDAYS
               GO TO 100
 50         CONTINUE
            BRPRATE(NPARTS)=0.0
            IF(BDAYS.GT.0.0) BRPRATE(NPARTS)=1.0/(24.0*BDAYS)
            DRPRATE(NPARTS)=0.0
            IF(DDAYS.GT.0.0) DRPRATE(NPARTS)=1.0/(24.0*DDAYS)
            FRACT(NPARTS)=FRACT(NPARTS)*FAP*SORTLGTH
            IF(FRACT(NPARTS).LE.0.0)GO TO 100
            NPARTS=NPARTS+1
         IF(NPARTS.LE.MAXPRT)GO TO 100
            READ(IFILE,END=200)CNSN
            WRITE(6,9004)MAXPRT
```

```
 200    CONTINUE
        NPARTS=NPARTS-1
        WRITE(6,9005)NPARTS
C---
C---    *CLOSE-OUT SPARES INPUT FILE
        CALL FCLOSE(IFILE)
C---
C---    *COMPUTE MEAN AND VARIANCE OF RANDOM VARIABLE - TOTAL
C---         -PART-DEMANDS PER BROKEN AIRCRAFT
        CALL PSTAT(PACBRK,NPARTS,IQPA,FRACT,ACMEAN,ACVAR,NPERAC)
C---
C---    *CONVERT PART-DEMANDS-PER-FLYING-HOUR TO A PDF
        CALL MAKEPD(NPARTS,IQPA,FRACT)
C---
C---    *SET-UP TABLES NEEDED FOR ALIAS METHOD OF SAMPLING PART-DEMANDS
        CALL ALIAS(NPARTS,FRACT,IALIAS)
        FPARTS = FLOAT(NPARTS)
C---
C--- *ELSE (INFINITE PARTS ASSUMED)
     GO TO 950
 900 CONTINUE
C---
C---    *PRINT MESSAGE INDICATING INFINITE SPARE PARTS ASSUMPTION
        WRITE(6,9002)
C---
C--- *END IF (INFINITE SPARE-PARTS TEST)
 950 CONTINUE
C---
  RETURN
 9002 FORMAT(1H0,7X,"INFINITE SPARE PARTS ASSUMED FOR THIS SGM RUN, ",
&    /,"I.E., NO AIRCRAFT EVER WAITS FOR A SPARE PART.")
 9003 FORMAT("0$$$$$$$$ INITPRT ERROR - INVALID PART CHARACTERISTIC",
&  /,       " $$$$$$$$     NPARTS, CNSN = ",I3,1X,A18,
&  /,       " $$$$$$$$     FRACT, IQPA, FAP = ",F6.4,I4,F5.2,
&  /,       " $$$$$$$$     INITSJ, RESUPP, BNRTS = ",I4,2F8.3,
&  /,       " $$$$$$$$     BDAYS, DDAYS  = ",2F10.2)
 9004 FORMAT("0$$$$$$$$ INITPRT ERROR - TOO MANY LRU TYPES",/,
&  " $$$$$$$$     MAXPRT = ",I5)
 9005 FORMAT(1H0,3X,"LRU TYPES - ",I4)
  END
```

```
C*************************************************************
   SUBROUTINE INITREP
C*************************************************************
C++ INITREP    - INITIALIZE VARIABLES FOR A SIMULATION REPLICATION.
C***     THIS ROUTINE PERFORMS THE LENGTHY INITIALIZATION NEEDED
C*** EACH SIMULATION REPLICATION OF THE SGM. THIS PROCESS
C*** IS ORGANIZED IN THE FOLLOWING MANNER. FIRST, MISCELLANEOUS
C*** OPERATIONS ARE PERFORMED, THEN SPARES INITIALIZATION, AND
C*** FINALLY, WORK CENTER INITIALIZATION. THE NUMEROUS COMMON OUTPUTS
C*** OF THIS ROUTINE ARE NOT DEFINED HERE, BUT THE
C*** OPERATIONS BEING PERFORMED SHOULD BE CLEAR FROM THE PROGRAM-
C*** DESIGN LANGUAGE (PDL) CORRESPONDING TO EACH OPERATION.
C*************************************************************
C---
      PARAMETER  MAXAC=108,MAXWC=25,MAXBIT=36,MAXPRT=304,
&                MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      PARAMETER MAXDAY=30,MAXCYC=10,MAXSTAT=5
      COMMON /ACSTATE/ LENGTH, NACVC(MAXVEC), IFLYVC(MAXVEC),
&                   MAINVC(MAXVEC), NORSVC(MAXVEC), LOSTVC(MAXVEC)
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                   MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      COMMON /PARTS/   NPARTS, IQPA(MAXPRT), NBACKO(MAXPRT),
&                   BRPRATE(MAXPRT), DRPRATE(MAXPRT), INITSJ(MAXPRT),
&                   RESUPP(MAXPRT), BNRTS(MAXPRT), NBASE(MAXPRT),
&                   NDEPOT(MAXPRT)
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                   NRESRV, IZDAY,ITOTRES(MAXDAY),LOSSTOT
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                   PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
      LOGICAL INFPART
C---
C---  *INITIALIZE RESERVE AIRCRAFT COUNTS
       CALL ZERO(ITOTRES,MAXDAY)
       IZDAY=0
       NRESRV=0
C---
C---  *RESET INITIAL UE FOR THIS REPLICATION
       NAC = INITUE
       CALL UEUPDAT(NAC)
C---
C---  *INITIALIZE CUMULATIVE AIRCRAFT LOSSES TO NONE
       LOSSTOT = 0
C---
C---  *CLEAR AIRCRAFT-STATUS BIT-VECTORS
       CALL ZERO(LOSTVC,MAXVEC,
&               NORSVC,MAXVEC,
&               INREPR,MAXWC)
C---
C---  *SET FIRST NAC BITS OF THE FLYING BIT-VECTOR TO FLYABLE
       DO 100 I=1,LENGTH
```

```
 100     IFLYVC(I)=NACVC(I)
C---
C---  *IF(INFINITE PARTS NOT ASSUMED)THEN
      IF(INFPART)GO TO 200
C---
C---     *CLEAR BASE AND DEPOT RESUPPLY COUNTS
         CALL ZERO(NBASE,MAXPRT,NDEPOT,MAXPRT)
C---
C---     *CALCULATE INITIAL BACKORDERS FOR EACH PART-TYPE
         CALL INITBO(NAC)
C---
C---     *INITIALIZE NUMBER/DISTRIBUTION OF NORS AIRCRAFT
         NORS = NORSAC(NPARTS,IQPA,NBACKO)
         CALL TBITSL(NORS,IFLYVC,NORSVC)
C---
C---  *END IF (INFINITE SPARES TEST)
 200  CONTINUE
C---
C---  *INITIALIZE NUMBER/DISTRIBUTION OF AIRCRAFT IN MAINTENANCE
      NBRKAC = INT(RNMCM*FLOAT(NAC))
      CALL WCDIST(NBRKAC,PBRKSEQ,INDXWC,IFLYVC)
C---
   RETURN
   END
```

```
C****************************************************************
   SUBROUTINE INITSCN(IFSCEN)
C****************************************************************
C** INITSCN   - READ AND INITIALIZES SCENARIO INPUTS.
C***      THIS ROUTINE LOADS THE SCENARIO PARAMETERS SPECIFIED
C*** BY THE USER. IT ALSO PREPARES THE SCRATCH FILE (FILE 03)
C*** WHICH IS USED TO WRITE A COPY OF THOSE SCENARIO
C*** PARAMETERS WHICH ARE ALLOWED TO VARY ON A DAILY BASIS
C*** THROUGHOUT THE SCENARIO. A LIST OF VALUES IS WRITTEN TO THIS
C*** SCRATCH FILE FOR EACH SIMULATION DAY. THEN, WHEN EACH
C*** FLYING DAY BEGINS (FOR EACH SIMULATION REPLICATION), THE
C*** PARAMETER VALUES FOR THAT DAY ARE LOADED.
C****************************************************************
C---
      PARAMETER MAXWC=25, MAXVARY=5, MAXCYC=10
      CHARACTER*4 FTOTYM,LTOTYM
      CHARACTER*20 CHSEED
      CHARACTER*80 NEXTLINE
      LOGICAL VARY(MAXVARY),VARYSW,INFPART,INFMAN
      COMMON /RSEED/   SEED
      COMMON /TIME/ PREFLITE,SORTLGTH,WAITCYC,
&       TYMNITE,NSIM,ISIM,NUMDAY,IDAY,NCYCLES,ICYCLE
      COMMON /INPUT/  INITUE,NAC,PATTRIT,IRES,RNMCM,INFPART,
&         MAXFLY(MAXCYC),INFMAN,ISCALE,IAUGMNT
      COMMON /WCBRK/ PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                   PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
C---
C--- *READ STORED INPUT
 5    FORMAT(V)
      READ(IFSCEN,5) (VARY(I),I=1,MAXVARY)
     READ(IFSCEN,5) CHSEED,FTOTYM,LTOTYM,INFMAN,INFPART,NSIM,NAC
&    ,PACBRK,RNMCM,NUMDAY,PREFLITE,SORTLGTH
     READ(IFSCEN,5) ISCALE
     NEXTLINE=' '
     INITUE=NAC
     WRITE(06,9000) NSIM,CHSEED,NAC
     VARYSW=.F.
     READ(IFSCEN,5) PATTRIT,PACGABT,NCYCLES,IRES,WAITCYC,TYMNITE
     READ(IFSCEN,5) (MAXFLY(I),I=1,NCYCLES)
     IF (VARY(4)) GOTO 100
         WRITE(06,9008) IRES
         GOTO 200
 100  CONTINUE
         WRITE(06,9007)
         CALL CONCAT(NEXTLINE,28,'VARY BY DAY',1,11)
         VARYSW=.T.
 200  CONTINUE
     IF (VARY(5)) GOTO 300
         WRITE(06,9010) MAXFLY(1)
         GOTO 400
 300  CONTINUE
         WRITE(06,9009)
         VARYSW=.T.
```

```
         CALL CONCAT(NEXTLINE,45,'VARIES BY CYCLE/DAY',1,19)
 400  CONTINUE
        WRITE(06,9011) NEXTLINE
        IF (VARY(3)) GOTO 500
           WRITE(06,9012) NUMDAY,NCYCLES,FTOTYM,LTOTYM,PREFLITE,SORTLGTH,
&          WAITCYC,TYMNITE
           GOTO 600
 500  CONTINUE
         WRITE(06,9013) NUMDAY,FTOTYM,LTOTYM,PREFLITE,SORTLGTH
         VARYSW=.T.
 600  CONTINUE
      WRITE(06,9014) RNMCM,PACBRK
      NEXTLINE=' '
      IF (VARY(1)) GOTO 700
         WRITE(06,9015) PATTRIT
         GOTO 800
 700  CONTINUE
         WRITE(06,9016)
         VARYSW=.T.
         CALL CONCAT(NEXTLINE,19,'BY DAY',1,6)
 800  CONTINUE
      IF (VARY(2)) GOTO 900
         WRITE(06,9017) PACGABT,NEXTLINE
         GOTO 1000
 900  CONTINUE
         CALL CONCAT(NEXTLINE,46,'BY DAY',1,6)
         WRITE(06,9018) NEXTLINE
         VARYSW=.T.
 1000  CONTINUE
       DO 1200 IDAY=1,NUMDAY
         WRITE(03) PATTRIT,PACGABT,NCYCLES,IRES,WAITCYC,TYMNITE
         WRITE(03) (MAXFLY(J),J=1,NCYCLES)
             IF (.NOT.VARYSW) GOTO 1100
                WRITE(06,9001) 'DAY =',IDAY
                IF (VARY(1)) WRITE(06,9002) PATTRIT
                IF (VARY(2)) WRITE(06,9003) PACGABT
                IF (VARY(3)) WRITE(06,9004) NCYCLES,WAITCYC,TYMNITE
                IF (VARY(4)) WRITE(06,9005) IRES
                IF (VARY(5)) WRITE(06,9006) 'CYCLE',(J,J=1,NCYCLES)
                IF (VARY(5)) WRITE(06,9006) 'MAX-FLY',
&                  (MAXFLY(J),J=1,NCYCLES)
 1100    CONTINUE
        IF (IDAY.EQ.NUMDAY) GOTO 1200
            READ(IFSCEN,5) PATTRIT,PACGABT,NCYCLES,IRES,WAITCYC,TYMNITE
            READ(IFSCEN,5) (MAXFLY(J),J=1,NCYCLES)
 1200  CONTINUE
C---
C---  *CONVERT USER SEED TO A REAL NUMBER
        DECODE(CHSEED,5)SEED
C---
C---  *CLOSE SCENARIO INPUT FILE
C---   CALL FCLOSE(IFSCEN)
C---
```

```
  RETURN
 9001   FORMAT('0',A5,I3)
 9002   FORMAT(' ATTRITION RATE =',F6.4)
 9003   FORMAT(' GROUND-ABORT RATE =',F6.4)
 9004   FORMAT(' WAVES PER DAY =',I3/' WAIT TIME =',F4.2,
&          /' OVERNITE RECOVERY =',F5.2)
 9005   FORMAT(' RESERVES =',I3)
 9006   FORMAT(' ',A7,10(2X,I3))
C
 9000  FORMAT('1'///4X'****************************************',
&       '***************************',/,4X'**********************'
&       ,'*****  SGM RUN  ****************************',/,4X
&       '**********************************************************',
&       '*******',///,4X
&       'SIMULATION -   REPLICATIONS =',I4,5X,
&       'RANDOM NUMBER SEED = ',A8/'0',3X,'AIRCRAFT -',3X,
&       'UE =',I3)
 9007 FORMAT('+',28X,'RESERVES')
 9008 FORMAT('+',27X,'RESERVES =',I3)
 9009 FORMAT('+',44X,'MAXIMUM LAUNCH-SIZE')
 9010 FORMAT('+',44X,'MAXIMUM LAUNCH-SIZE =',I3)
 9011 FORMAT(' ',A80/'0',3X,'FLYING SCHEDULE -'/'0',10X,'WAVES',3X,
&       'TAKEOFF',
&       ' TIMES',6X,'MINIMAL',3X,'SORTIE',2X,'WAIT',2X,'OVERNIGHT'/
&       4X,'DAYS',2X,'PER DAY',3X,'FIRST',3X,'LAST',4X,
&       'TURNAROUND',2X,'LENGTH',2X,'TIME',2X,'RECOVERY')
 9012 FORMAT('0',3X,I3,5X,I2,6X,A4,4X,A4,6X,F5.2,5X,F5.2,3X,F4.2,
&       3X,F5.2)
 9013 FORMAT('0',3X,I3,4X,'VARY',5X,A4,4X,A4,6X,F5.2,5X,F5.2,2X,
&       'VARIES',2X,'VARIES'/' ',9X,'BY DAY',39X,'BY DAY',2X,'BY DAY')
 9014  FORMAT('0',3X,'RATES -'/'0',6X,'INITIAL',17X,'AIRCRAFT'/
&       6X,'NMCM RATE',3X,'ATTRITION',3X,'BREAK RATE',3X,
&       'GROUND-ABORT'/'0',7X,F5.3,7X,13X,F6.4,)
 9015 FORMAT('+',17X,F6.2)
 9016 FORMAT('+',18X,'VARIES')
 9017 FORMAT('+',43X,F6.4/' ',A80)
 9018 FORMAT('+',45X,'VARIES'/' ',A80)
  END
```

```
**************************************************
      SUBROUTINE INITWC(IFILE,INFMAN,NAC)
C*******************************************************************
C++ INITWC     - LOAD AND INITIALIZE MAINTENANCE WORK CENTER DATA.
C***      THIS ROUTINE INITIALIZES THE INFORMATION NEEDED
C*** FOR MODELING THE AIRCRAFT MAINTENANCE WORK-CENTERS.
C*** IT READS THE MAINTENANCE MANPOWER INPUT FILE, PRINTS
C*** A LISTING OF THESE INPUTS, AND COMPUTES ADJUSTED BREAK-
C*** RATE ARRAYS FOR WORK-CENTER BREAKS
C***
C*** INPUTS --
C***      IFILE   - UNIT NUMBER OF THE INPUT FILE FROM
C***                WHICH THE WORK-CENTER INPUTS ARE READ.
C***                THIS FILE IS CLOSED-OUT AFTER THE INPUTS
C***                ARE READ
C***      INFMAN  - LOGICAL VARIABLE INDICATING WHETHER INFINITE
C***                MANPOWER IS ASSUMED FOR ALL WORK-CENTERS. IF
C***                INFMAN=TRUE THEN NUMBER OF SERVERS FOR EACH WC
C***                IS SET EQUAL TO THE MAXIMUM ALLOWABLE NUMBER
C***                OF AIRCRAFT.
C***      NAC     - UE (UNIT EQUIPMENT); NUMBER OF AIRCRAFT
C***                POSSESSED BY THE BASE OF INTEREST.
C*** COMMON INPUTS --
C***      PACBRK  - USER-INPUT AIRCRAFT BREAK-RATE.  PROBABILITY
C***                THAT AN AIRCRAFT RETURNING FROM A SORTIE REQUIRES
C***                UNSCHEDULED MAINTENANCE IN AT LEAST 1 WORK-CENTER.
C***      PACGABT - PROBABILITY THAT AN AIRCRAFT GROUND-ABORTS DURING
C***                THE PRE-TAKEOFF PERIOD.
C*** COMMON OUTPUTS --
C***      NWC     - NUMBER OF WORK-CENTERS TO BE MODELED
C***      NCREWS  - NUMBER-OF-SERVERS ARRAY. NCREWS(I) IS THE
C***                NUMBER OF SERVERS IN THE ITH WORK-CENTER.
C***      SRATE   - SERVICE-RATES ARRAY. SRATE(I) IS THE SERVICE-
C***                RATE (IN AIRCRAFT PER HOUR) OF THE SERVERS
C***                FOR THE ITH WORK-CENTER.
C***      PBRKSEQ - WORK-CENTER BREAK ARRAYS FOR THE SEQUENTIAL
C***                SAMPLING PROCESS OF DETERMINING WHICH WORKCENTERS
C***                AIRCRAFT BREAK INTO.
C***      INDXWC  - SORTED ARRAY OF WORK-CENTER INDICES USED WITH
C***                SEQUENTIAL-SAMPLING PROCESS.
C*******************************************************************
C--
      PARAMETER MAXWC=25, MAXAC=108, LFLD=7
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
      COMMON /WCBRK/ PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
     &               PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
      LOGICAL INFMAN
C--
C--  *READ AND ECHO-PRINT MAINTENANCE MANPOWER INPUT FILE
      CALL WCREAD(IFILE,MAXWC,NWC,PBRKWC,NCREWS,SRATE)
C--
C--  *IF INFINITE MANPOWER ASSUMED -- RESET NUMBER OF SERVERS
C--   PER SHIFT TO MAX NUMBER OF AIRCRAFT; THUS NO AIRCRAFT
```

```
C---   WILL EVER WAIT FOR A SERVER
       IF(INFMAN) CALL SPRAY(MAXAC,NCREWS,NWC)
       IF(INFMAN)WRITE(6,9001)
C---
C--- *INITIALIZE WORK-CENTER BREAK ARRAYS FOR GROUND-ABORTS AND
C---   BREAKS; NOTE THAT THE SAME ARRAYS ARE CURRENTLY USED FOR BOTH
       CALL WCPROB(NWC,PBRKWC,PBRKSEQ,INDXWC,PWCPROD)
C---
C--- *PERFORM ERROR CHECK TO ENSURE LFLD PARAMETER LARGE ENOUGH
C---          SO THAT THE BIT-FIELD CAN STORE THE MAX AC #
       IF(MAXAC.GT.2**LFLD)WRITE(6,9002)LFLD,MAXAC
C---
   RETURN
 9001 FORMAT(1H0,7X,"INFINITE MANPOWER ASSUMED FOR THIS SGM RUN -",/,
&   7X,"I.E., THERE ARE NEVER ANY AIRCRAFT QUEUES IN MAINTENANCE.")
 9002 FORMAT("0$$$$$$$$ INITWC ERROR - LFLD PARAMETER TOO SMALL",/,
&          " $$$$$$$$     LFLD, MAXAC = ",2I5)
   END
```

```
C*****************************************************************
   INTEGER FUNCTION IPOISSON(RMEAN,SEED)
C*****************************************************************
C++ IPOISSON   - GENERATE RANDOM SAMPLE FROM A POISSON DISTRIBUTION.
C***      THIS ROUTINE GENERATES A RANDOM SAMPLE FROM A
C*** POISSON DISTRIBUTION WITH A GIVEN MEAN. THE EXPONENTIAL-DRAW
C*** METHOD IS USED FOR DISTRIBUTIONS WITH SMALL MEANS, AND
C*** A NORMAL APPROXIMATION IS USED FOR LARGER MEANS ( >20).
C***
C*** INPUT --
C***   RMEAN      - MEAN OF POISSON DISTRIBUTION FROM WHICH SAMPLE
C***                IS TO BE GENERATED.
C*** INPUT/OUTPUT --
C***   SEED       - SEED OF RANDOM NUMBER GENERATOR.
C*****************************************************************
C---
C---      *IF(INPUT PARAMETER IS A LEGITIMATE MEAN FOR A POISSON)
           IF(RMEAN.LT.0.0)GO TO 400
C---
C---         *IF(MEAN IS NOT TOO LARGE)
              IF(RMEAN .GT. 20.0) GO TO 200
C---
C---            *USE EXPONENTIAL DRAW METHOD FOR POISSON SAMPLE
                 IPOISSON=-1
                 PROD=1.0
                 TEST=EXP(-RMEAN)
 100             CONTINUE
                    IPOISSON=IPOISSON+1
                    PROD=PROD*UNIFM1(SEED)
                 IF(PROD.GE.TEST)GO TO 100
C---
C---         *ELSE (LARGE MEAN)
              GO TO 300
 200          CONTINUE
C---
C---            *USE NORMAL APPROXIMATION TO POISSON
                 IPOISSON=MAX0(0, INT(XNORM(RMEAN,SQRT(RMEAN),SEED)+.5))
C---
C---         *END IF (SIZE OF MEAN TEST)
 300          CONTINUE
C---
C---      *ELSE (MEAN IS LESS THAN ZERO)
           GO TO 500
 400       CONTINUE
C---
C---         *SET RETURN VALUE TO ZERO AND PRINT ERROR MESSAGE
              IPOISSON = 0
              WRITE(6,9001)RMEAN
C---
C---      *END IF (LEGITIMATE MEAN TEST)
 500       CONTINUE
C---
   RETURN
```

```
 9001 FORMAT("0$$$$$$$$ IPOISSON ERROR - NEGATIVE MEAN ",/,
&           " $$$$$$$$     RMEAN = ",F10.5)
   END
```

```
C*****************************************************************
   INTEGER FUNCTION LBITS(IWORD,NBITS)
C*****************************************************************
C++ LBITS      - MASK-OFF LEFTMOST 1-BITS IN A COMPUTER WORD.
C***      LBITS IS A FORTRAN FUNCTION WHICH WILL SELECT A GIVEN
C*** NUMBER OF 1-BITS FROM THE LEFTMOST PORTION OF A GIVEN INPUT
C*** WORD. LBITS RETURNS A WORD CONSISTING OF THESE SELECTED
C*** 1-BITS WITH 0'S EVERYWHERE ELSE. NOTE THAT THE INPUT
C*** WORD SHOULD CONTAIN AT LEAST AS MANY 1-BITS AS THE NUMBER TO BE
C*** SELECTED, 'NBITS'.
C***      THIS ROUTINE IS SPECIFIC TO A COMPUTER WITH 36-BIT WORDS
C*** SINCE IT WORKS BY EXTRACTING 6-BIT FIELDS.
C***
C*** INPUTS --
C***   IWORD      - WORD FROM WHICH THE 1-BITS ARE TO BE SELECTED.
C***                THIS WORD SHOULD CONTAIN AT LEAST AS MANY 1-BITS
C***                AS REQUESTED. IF MORE THAN THAT ARE REQUESTED,
C***                THIS ROUTINE WILL RETURN AN EXACT COPY OF THE INPUT.
C***   NBITS      - NUMBER OF 1-BITS TO BE SELECTED FROM 'IWORD'.
C*** OUTPUT --
C***   LBITS      - A COPY OF THE PORTION OF THE INPUT WORD CONTAINING
C***                THE SPECIFIED NUMBER OF LEFTMOST 1-BITS.
C*** COMMON TABLES USED --
C***   ICOUNT(I)  - NUMBER OF 1-BITS IN THE BINARY REPRESENTATION OF THE
C***                INDEX I.   I=0,1,2,...,63
C***   MSKLFT(I)  - MASK FOR WHICH THE LEFTMOST I-BITS ARE 1-BITS,
C***                AND THE REMAINDER OF THE WORD IS ZERO.  I=1,2,3,...,36
C***   MASK(I)    - CONTAINS A 1 IN THE ITH BIT (COUNTING FROM THE LEFT)
C***                AND 0'S EVERYWHERE ELSE.  I=0,1,2,...,35
C*****************************************************************
C---
      PARAMETER  MAXBIT=36,LFIELD=6
      COMMON /BITS/ MASK0,MASK(35), MLEFT0,MSKLFT(36),
&                                 IZCOUT,ICOUNT(63)
C---
C---        *IF(NO BITS ARE REQUESTED)THEN
             IF(NBITS.GT.0) GO TO 1000
C---
C---           *RETURN A VECTOR OF ALL ZEROES
                LBITS = 0
C---
C---        *ELSE (SELECT LEFTMOST BITS)
             GO TO 5000
 1000        CONTINUE
C---
C---          **SEARCHING FROM LEFT TO RIGHT, FIND THE 6-BIT FIELD
C---            CONTAINING THE LAST BIT TO BE SELECTED
C---
C---             *INITIALIZE DO
                  IBIT   = 0
                  IFOUND = 0
C---
C---             *DO UNTIL(APPROPRIATE 6-BIT FIELD IS FOUND)
```

```
 2000            CONTINUE
C---
C---                *UPDATE NUMBER OF 1-BITS FOUND SO FAR
                     IFOUND = IFOUND + ICOUNT(FLD(IBIT,LFIELD,IWORD))
C---
C---                *UPDATE FIELD COUNTER
                     IBIT = IBIT + LFIELD
C---
C---              *END DO (FIELD LOOP)
                   IF((IFOUND.LT.NBITS) .AND. (IBIT.LT.MAXBIT)) GO TO 2000
C---
C---            *COMPUTE NUMBER OF EXTRA 1-BITS INCLUDED
                 NEXTRA = IFOUND - NBITS
C---
C---            *PERFORM ERROR CHECK TO ENSURE PROPER NUMBER OF 1S FOUND
                 IF(NEXTRA.LT.0)WRITE(6,9001)NBITS,IFOUND,NEXTRA
C---
C---            *DO WHILE(THERE ARE EXTRA 1'S TO ELIMINATE)
 3000            CONTINUE
                 IF(NEXTRA.LE.0) GO TO 4000
C---
C---              *DECREMENT BIT COUNTER
                   IBIT = IBIT - 1
C---
C---              *DECREMENT EXTRA-BIT COUNTER IF THIS BIT IS A 1
                   IF(AND(IWORD,MASK(IBIT)).NE.0) NEXTRA=NEXTRA-1
C---
C---            *END DO (EXTRA 1'S LOOP)
                 GO TO 3000
 4000            CONTINUE
C---
C---            *RETURN PORTION OF INPUT UP TO, BUT NOT INCLUDING
C---                                                  THIS LAST BIT
                 LBITS = AND(IWORD,MSKLFT(IBIT))
C---
C---          *END IF (ZERO BITS REQUESTED TEST)
 5000          CONTINUE
C---
   RETURN
 9001 FORMAT("0$$$$$$$$ LBITS ERROR - TOO FEW 1-BITS TO MASK",/,
&           " $$$$$$$$     NBITS, IFOUND, NEXTRA = ",3I5)
   END
```

```
C*****************************************************************
   SUBROUTINE MAKEPD (N,IQPA,DEMAND)
C*****************************************************************
C++ MAKEPD     - CONVERTS PARTS DEMAND ARRAY INTO A PDF.
C***      THE ALIAS METHOD REQUIRES A LEGITIMATE PROBABILITY
C***      DISTRIBUTION AS AN INPUT. THIS ROUTINE TAKES THE QPA
C***      AND PROB[DEMAND] FIGURES FOR EACH PART, AND FORMS A PDF
C***      FROM THEIR PRODUCTS. THERE ARE N PART TYPES.
C*****************************************************************
C---
      DIMENSION IQPA(N),DEMAND(N)
C---
      SUM = 0.0
      DO 30 K=1,N
         DEMAND(K) = DEMAND(K) * FLOAT(IQPA(K))
         SUM = SUM + DEMAND(K)
   30 CONTINUE
      RECIP = 1.0 / SUM
      SUM = 0.0
      DO 60 K=1,N-1
         DEMAND(K) = DEMAND(K) * RECIP
         SUM = SUM + DEMAND(K)
   60 CONTINUE
      DEMAND(N) = 1.0 - SUM
C---
   RETURN
   END
```

```
C*************************************************************************
   INTEGER FUNCTION MNOM (DUMMY)
C*************************************************************************
C++ MNOM       - GENERATE MULTINOMIAL SAMPLE FOR PART DEMAND TYPE.
C***    THIS FUNCTION GENERATES A MULTINOMIAL SAMPLE INDICATING
C***    WHICH PART TYPE HAS BROKEN.  IT USES TWO TABLES CREATED
C***    PREVIOUSLY BY SUBROUTINE 'ALIAS'.
C***
C*** COMMON INPUTS -
C***       FPARTS    - FLOATING-POINT VALUE OF NUMBER OF PART TYPES, N.
C***       FRACT(I)  - TABLE OF FRACTIONAL CUTOFF VALUES USED BY THE
C***                   ALIAS METHOD.    I=1,2,...,N
C***       IALIAS(I) - TABLE OF ALIASES USED BY ALIAS METHOD. I=1,...,N
C*** OUTPUT -
C***       MNOM      - INDEX INDICATING TYPE OF PART WHICH HAS BROKEN.
C***                   MNOM=1,2,...,N  (NUMBER OF PART TYPES)
*********************************************************************
C---
      PARAMETER MAXPRT=304
      COMMON /RSEED/ SEED
      COMMON /ALIASC/ FRACT(MAXPRT),IALIAS(MAXPRT),FPARTS
C---
C---  *MAKE 'U' A UNIFORM (0,N) RANDOM REAL NUMBER
       U = FPARTS * UNIFM1(SEED)
C---  *
C---  *MAKE 'IU' A UNIFORM RANDOM INTEGER (1,N)
       IU = IFIX(U) + 1
C---
C---  *IF NECESSARY, REPLACE 'IU' BY ITS ALIAS
       IF (U .GT. FRACT(IU))  IU = IALIAS(IU)
C---
C---  RESULT IS RETURNED AS 'MNOM'
       MNOM = IU
C---
   RETURN
   END
```

```
C*****************************************************************
   SUBROUTINE MUPDATE(NWC,MAINVC)
C*****************************************************************
C++ MUPDATE    - UPDATE MAINTENANCE AIRCRAFT-STATE BIT-VECTOR.
C***   MUPDATE UPDATES THE OVERALL MAINTENANCE BIT-VECTOR. THIS
C*** UPDATING PROCESS CONSISTS OF GOING THROUGH EACH WORK-CENTER
C*** LIST OF AIRCRAFT AND MARKING THE CORRESPONDING BIT-POSITION IN
C*** THE MAINTENANCE BIT-VECTOR FOR ANY AIRCRAFT IN SUCH A LIST; THUS,
C*** ANY AIRCRAFT IN AT LEAST ONE WORK-CENTER LIST WILL BE MARKED AS
C*** BEING IN MAINTENANCE STATUS.
C***    THE MAINTENANCE BIT-VECTOR IS UPDATED AT THE BEGINNING OF
C*** EACH FLYING CYCLE.  IT IS NOT MAINTAINED DURING THE FLYING CYCLE,
C*** SINCE IT IS ONLY NEEDED AT THE START OF PREFLIGHT TO DETERMINE
C*** THOSE AIRCRAFT WHICH ARE NOT MISSION-CAPABLE BECAUSE THEY ARE
C*** IN AT LEAST ONE WORK-CENTER. IT IS MUCH FASTER TO UPDATE
C*** ONCE EACH FLYING CYCLE RATHER THAN UPDATING IT EACH TIME A
C*** WORK-CENTER BREAK OR REPAIR OCCURS.
C***
C*** INPUTS -
C***    NWC        - NUMBER OF WORK-CENTERS BEING SIMULATED
C*** COMMON INPUTS -
C***    MASK(I)    - CONTAINS A 1 IN THE ITH BIT (COUNTING FROM LEFT)
C***                 AND ZEROES EVERYWHERE ELSE. I=0,...,35
C***    LENGTH     - LENGTH (IN WORDS) OF AIRCRAFT BIT-VECTORS
C***    INREPR(J)  - NUMBER OF AIRCRAFT IN WORKCENTER-J.
C***    LISTRP(I,J) - LISTRP( . ,J) IS A LIST OF AIRCRAFT NUMBERS
C***                 INDICATING THOSE AIRCRAFT REQUIRING MAINTENANCE IN
C***                 THE JTH WORK-CENTER (J=1,2,...,NWC). THIS LIST
C***                 CONTAINS EXACTLY INREPR(J) AIRCRAFT NUMBERS. TO SAVE
C***                 SPACE, THESE LISTS HAVE BEEN PACKED INTO BIT-FIELDS
C***                 INSTEAD OF WORDS. EACH NUMBER IS STORED IN A BIT-FIELD
C***                 "LFLD" BITS WIDE; HENCE, IF "MAXBIT" IS THE LENGTH
C***                 OF A COMPUTER WORD ON THIS SYSTEM, THEN THERE ARE
C***                 (MAXBIT/LFLD) BIT-FIELDS STORED PER WORD. THE AIRCRAFT
C***                 NUMBERS STORED IN THESE BIT-FIELDS INDICATE A UNIQUE
C***                 BIT-POSITION IN THE VARIOUS AIRCRAFT-STATUS BIT-
C***                 VECTORS. THE AIRCRAFT ARE NUMBERED,LEFT-TO-RIGHT,
C***                 0,1,2,...,(MAXAC-1) . TO GET THE ITH AIRCRAFT NUMBER
C***                 IN A WORK-CENTER LIST, THE CORRESPONDING
C***                 BIT-POSITION AND WORD-INDEX MUST BE COMPUTED.
C*** OUTPUTS -
C***    MAINVC     - MAINTENANCE AIRCRAFT-STATUS BIT-VECTOR. EACH BIT
C***                 REPRESENTS AN AIRCRAFT. A 1 INDICATES THE
C***                 CORRESPONDING AIRCRAFT IS BEING REPAIRED IN AT
C***                 LEAST ONE WORK-CENTER, AND 0 INDICATES THE
C***                 AIRCRAFT IS NOT CURRENTLY IN MAINTENANCE.
C*****************************************************************
C---
      PARAMETER MAXWC=25
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
      COMMON /BITS/ MASK0,MASK(35),MLEFT0,MSKLFT(36),
```

```
&                    IZCOUT, ICOUNT(63)
      DIMENSION MAINVC(MAXVEC)
C---
C---  *INITIALIZE MAINTENANCE BIT-VECTOR TO NO AIRCRAFT
       CALL ZERO(MAINVC,MAXVEC)
C---
C---  *DO FOR(EACH WORK-CENTER)
       IF(NWC.EQ.0) GO TO 400
       DO 300 J=1,NWC
C---
C---     *DO WHILE(STILL AIRCRAFT IN THIS WORK-CENTER MAINTENANCE LIST)
          NUM = INREPR(J)
          IF(NUM.EQ.0) GO TO 200
          DO 100 I=1,NUM
C---
C---        *GET NEXT AIRCRAFT NUMBER ON LIST FROM APPROPIATE BIT-FIELD
             IAC = FLD(MOD(I-1,NPERWRD)*LFLD,LFLD,
&                          LISTRP(1+(I-1)/NPERWRD,J))
C---
C---        *COMPUTE WORD AND BIT POSITIONS INDICATED BY THIS AC #
             IWORD = 2 + IAC/MAXBIT
             IBIT  = MOD(IAC,MAXBIT)
C---
C---        *MARK CORRESPONDING POSITION IN AIRCRAFT MAINT BIT-VECTOR
             MAINVC(IWORD) = OR(MAINVC(IWORD), MASK(IBIT))
C---
C---     *END DO (WORK-CENTER LIST LOOP)
  100     CONTINUE
  200     CONTINUE
C---
C---  *END DO (WORK-CENTER LOOP)
  300  CONTINUE
  400  CONTINUE
C---
C---  *COMPUTE TOTAL AIRCRAFT IN MAINTENANCE BY COUNTING 1-BITS IN
C---                   AIRCRAFT MAINTENANCE VECTOR
       MAINVC(1) = N1VECT(MAINVC)
C---
    RETURN
    END
```

```
C*****************************************************************
   INTEGER FUNCTION N1BITS(IWORD)
C*****************************************************************
C++ N1BITS     - COUNT NUMBER OF 1-BITS IN A COMPUTER WORD.
C***      N1BITS IS A FORTRAN SUBROUTINE WHICH WILL RETURN THE
C*** NUMBER OF 1-BITS IN A GIVEN WORD.  THIS ROUTINE IS SPECIFIC
C*** TO A COMPUTER WITH 36-BIT WORDS, SINCE IT WORKS BY EXTRACTING
C*** 6-BIT FIELDS FROM THE WORD. IT USES EACH 6-BIT FIELD EXTRACTED
C*** AS AN INDEX INTO A TABLE, AND THE ENTRIES IN THE TABLE CONTAIN THE
C*** CORRESPONDING NUMBER OF 1-BITS FOR THAT INDEX.
C***
C*** INPUT --
C***   IWORD      - WORD FOR WHICH THE 1-BITS ARE TO BE COUNTED
C*** OUTPUT --
C***   N1BITS     - NUMBER OF 1-BITS IN THE GIVEN INPUT WORD. HENCE,
C***                N1BITS RETURNS AN INTEGER BETWEEN 0 AND 36.
C*** TABLE USED -
C***   ICOUNT(I) - NUMBER OF 1-BITS IN THE BINARY REPRESENTATION OF THE
C***                INDEX I.  I=1,...,63
C*****************************************************************
C--
     COMMON /BITS/    MASK0,MASK(35), MLEFT0,  MSKLFT(36),
&                                IZCOUT,ICOUNT(63)
C--
          N1BITS = ICOUNT( FLD( 0,6,IWORD) )
&                 + ICOUNT( FLD( 6,6,IWORD) )
&                 + ICOUNT( FLD(12,6,IWORD) )
&                 + ICOUNT( FLD(18,6,IWORD) )
&                 + ICOUNT( FLD(24,6,IWORD) )
&                 + ICOUNT( FLD(30,6,IWORD) )
C--
   RETURN
   END
```

```
C*****************************************************************
   INTEGER FUNCTION N1VECT(IARRAY)
C*****************************************************************
C** N1VECT     - COUNT NUMBER OF 1-BITS IN A BIT-VECTOR.
C***     N1VECT IS A FORTRAN SUBROUTINE WHICH WILL RETURN THE
C*** NUMBER OF 1-BITS IN THE WORDS COMPRISING A GIVEN INPUT ARRAY,
C*** NOT INCLUDING THE FIRST WORD. THIS ROUTINE IS USED TO COUNT
C*** THE NUMBER OF 1-BITS IN THE VARIOUS AIRCRAFT-STATUS
C*** BIT-VECTORS.
C***     THIS ROUTINE IS SPECIFIC TO A 36-BIT-WORD COMPUTER, SINCE IT
C*** WORKS BY EXTRACTING 6-BIT FIELDS FROM THE ARRAY WORDS. IT USES
C*** EACH 6-BIT FIELD EXTRACTED AS AN INDEX INTO A TABLE, AND THE
C*** ENTRIES IN THE TABLE CONTAIN THE CORRESPONDING NUMBER OF 1-BITS
C*** FOR THAT INDEX.
C***
C*** INPUT -
C***   IARRAY    - ARRAY FOR WHICH THE 1-BITS ARE TO BE COUNTED.
C*** COMMON TABLE USED -
C***   ICOUNT(I) - NUMBER OF 1-BITS IN THE BINARY REPRESENTATION OF THE
C***               INDEX I.  I=1,...,63
C*** OUTPUT -
C***   N1VECT    - NUMBER OF 1-BITS IN THE GIVEN INPUT ARRAY,
C***               EXCLUDING THE FIRST WORD.
C*****************************************************************
C---
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /BITS/ MASK0,MASK(35), MLEFT0,MSKLFT(36),
&                                 IZCOUT,ICOUNT(63)
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
&                MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION IARRAY(1)
C---
         N1VECT = 0
         DO 1000  I=2,LENGTH
            IWORD  = IARRAY(I)
            N1VECT = N1VECT + ICOUNT( FLD( 0,6,IWORD) )
&                           + ICOUNT( FLD( 6,6,IWORD) )
&                           + ICOUNT( FLD(12,6,IWORD) )
&                           + ICOUNT( FLD(18,6,IWORD) )
&                           + ICOUNT( FLD(24,6,IWORD) )
&                           + ICOUNT( FLD(30,6,IWORD) )
 1000    CONTINUE
C---
   RETURN
   END
```

```
C*****************************************************************
   INTEGER FUNCTION NBINOM(PBINOM,NTRYS)
C*****************************************************************
C++ NBINOM     - GENERATE RANDOM SAMPLE FROM BINOMIAL DISTRIBUTION.
C***       NBINOM GENERATES A RANDOM SAMPLE FROM A BINOMIAL DISTRIBUTION
C*** WITH THE GIVEN INPUT CHARACTERISTICS. THIS ROUTINE USES A
C*** COMBINATION OF TWO METHODS TO GENERATE THIS SAMPLE. FOR
C*** BINOMIALS WITH RELATIVELY SMALL NUMBERS OF TRIALS, THE
C*** STRAIGHTFORWARD BERNOULLI TRIALS METHOD IS USED. FOR LARGER
C*** VALUES, THE INVERSE TRANSFORM METHOD IS USED.
C***       NOTE THAT THE NUMBER OF FAILURES IN A BINOMIAL SAMPLE IS
C*** THE COMPLEMENT OF THE NUMBER OF SUCCESSES IN THAT DRAW. HENCE
C*** THIS ROUTINE WILL SAMPLE FROM THE COMPLEMENTARY BINOMIAL
C*** DISTRIBUTION OF FAILURES WHEN THE PROBABILITY OF SUCCESS IS
C*** GREATER THAN .5 .
C***
C*** INPUTS --
C***   PBINOM    - PROBABILITY CHARACTERISTIC OF THE BINOMIAL.
C***               PBINOM ALSO EQUALS THE PROBABILITY THAT THE
C***               BERNOULLI VARIABLE UNDERLYING THIS BINOMIAL EQUALS 1.
C***   NTRYS     - NUMBER OF BERNOULLI TRIALS CHARACTERIZING THIS
C***               BINOMIAL.
C*****************************************************************
C---
      COMMON /RSEED/ SEED
C---
C---  *INITIALIZE SAMPLE TO NO SUCCESSES
      NBINOM = 0
C---
C---  *IF(THIS IS NOT A SPECIAL DISTRIBUTION TO BE HANDLED SEPERATELY)
      IF((PBINOM.LE.0.0) .OR. (PBINOM.GE. 1.0)
&                             .OR. (NTRYS.LE.4)) GO TO 3000
C---
C---     *DRAW RANDOM SAMPLE FROM UNIFORM (0,1) DISTRIBUTION
         RDRAW = UNIFM1(SEED)
C---
C---     *DETERMINE WHETHER TO SAMPLE SUCCESSES OR FAILURES
         PFAIL = AMAX1(PBINOM,1.0-PBINOM)
         PSUCC = 1.0 - PFAIL
C---
C---     *COMPUTE QUICK APPROXIMATION TO PROB(0 SUCCESSES)
         PROB = 1.0 - FLOAT(NTRYS)*PSUCC
         IF(RDRAW.LE.PROB) GO TO 2000
C---
C---     *COMPUTE EXACT PROBABILITY OF NO SUCCESSES
         PROB = PFAIL**NTRYS
C---
C---     *IF(RANDOM DRAW DOES NOT FALL WITHIN THIS PORTION OF THE CDF)
         IF(RDRAW.LE.PROB) GO TO 2000
C---
C---        *INITIALIZE LOOP TO FIND APPROPRIATE PLACE IN THE CDF
            RATIO = PSUCC/PFAIL
            NPLUS1 = NTRYS + 1
```

```
          CDF = PROB
C---
         *DO UNTIL(APPROPRIATE CDF INDEX IS FOUND)
 1000     CONTINUE
C---
C---        *UPDATE SAMPLE COUNTER
             NBINOM = NBINOM + 1
C---
C---        *COMPUTE NEXT ENTRY IN CUMULATIVE DISTRIBUTION FUNCTION
             PROB = (FLOAT(NPLUS1-NBINOM)/FLOAT(NBINOM))*RATIO*PROB
             CDF = CDF + PROB
C---
C---      *END DO (CDF LOOP)
           IF((RDRAW.GT.CDF) .AND. (NBINOM.LT.NTRYS)) GO TO 1000
C---
C---    *END IF (0 SUCCESSES TEST)
 2000    CONTINUE
C---
C---    *COMPLEMENT RESULT IF FAILURES WERE SAMPLED
         IF(PBINOM.GT. .5) NBINOM = NTRYS - NBINOM
C---
C---  *ELSE (SPECIAL CASES)
      GO TO 7000
 3000    CONTINUE
C---
C---    *IF(THIS IS A DEGENRATIVE DISTRIBUTION)THEN
         IF((PBINOM.GT.0).AND.(PBINOM.LT.1.0).AND.(NTRYS.GT.0))
&                                        GO TO 4000
C---
C---      *SAMPLE FROM DISTRIBUTION (IF PBINOM=0, OR NTRYS=0
C---                            THEN WE ARE DONE)
           IF(PBINOM.GE.1.0) NBINOM = NTRYS
C---
C---    *ELSE (USE BERNOULLI TRIAL METHOD)
         GO TO 6000
 4000    CONTINUE
C---
C---      *PERFORM APPROPRIATE NUMBER OF BERNOULLI TRIALS
           DO 5000 I=1,NTRYS
              IF(UNIFM1(SEED).LE.PBINOM) NBINOM = NBINOM + 1
 5000      CONTINUE
C---
C---    *END IF (DEGENERATIVE DISTRIBUTION TEST)
 6000    CONTINUE
C---
C---  *END IF (SPECIAL CASES TEST)
 7000    CONTINUE
C---
      RETURN
      END
```

```
C*****************************************************************
   INTEGER FUNCTION NDMNDS(NBRKAC)
C*****************************************************************
C++ NDMNDS     - GENERATE SAMPLE OF TOTAL SORTIE PART DEMANDS.
C***      NDMNDS IS A FORTRAN FUNCTION WHICH GENERATES A
C*** SAMPLE NUMBER OF PARTS DEMANDS ON A SORTIE, GIVEN THE TOTAL
C*** NUMBER OF AIRCRAFT WHICH BROKE ON THAT SORTIE. THE
C*** PROBABILITY DISTRIBUTION OF TOTAL PARTS DEMANDS IS APPROXIMATED
C*** USING EITHER A NORMAL DISTRIBUTION (IF MEAN IS LARGE ENOUGH TO
C*** APPLY THE CENTRAL LIMIT THEOREM) OR A POISSON DISTRIBUTION.
C***
C*** INPUTS --
C***   NBRKAC     - NUMBER OF AIRCRAFT WHICH BROKE ON THE SORTIE
C*** COMMON INPUTS --
C***   ACMEAN     - EXPECTED VALUE OF THE RANDOM VARIABLE REPRESENTING
C***                THE NUMBER OF PARTS DEMANDS PER AIRCRAFT, GIVEN
C***                THAT THE AIRCRAFT HAS BROKEN UPON RETURNING FROM
C***                A SORTIE.
C***   ACVAR      - VARIANCE OF TOTAL PARTS DEMAND PER BROKEN AIRCRAFT
C***   NPERAC     - TOTAL NUMBER OF PARTS PER AIRCRAFT. THIS IS USED TO
C***                ENSURE THAT A LEGITIMATE SAMPLE IS GENERATED.
C*****************************************************************
C---
      PARAMETER CUTOFF=0.0
      COMMON /RSEED/  SEED
      COMMON /DEMAND/ ACMEAN, ACVAR, NPERAC
C---
C---       *INITIALIZE SAMPLE TO NO PARTS DEMANDS
            NDMNDS = 0
C---
C---       *IF(THERE WERE ANY BROKEN AIRCRAFT)THEN
            IF(NBRKAC.EQ.0) GO TO 300
C---
C---          *COMPUTE MEAN OF DISTRIBUTION OF TOTAL DEMANDS
C---           CORRESPONDING TO NUMBER OF BROKEN AIRCRAFT
               FLTAC  = FLOAT(NBRKAC)
               BMEAN  = FLTAC * ACMEAN
C---
C---          *IF(EXPECTED TOTAL DEMANDS IS SMALL)THEN
               IF(BMEAN.GT.CUTOFF) GO TO 100
C---
C---             *USE POISSON APPROXIMATION
                  NDMNDS = IPOISSON(BMEAN,SEED)
C---
C---          *ELSE
               GO TO 200
  100          CONTINUE
C---
C---             *USE NORMAL APPROXIMATION
                  BSTDEV = SQRT(FLTAC*ACVAR)
                  NDMNDS = MAX0(0,INT(XNORM(BMEAN,BSTDEV,SEED)+.5))
C---
C---          *END IF (APPROXIMATION TYPE TEST)
```

```
  200         CONTINUE
C---
C---         *ENSURE THAT A FEASIBLE ANSWER HAS BEEN GENERATED
              NDMNDS = MINO(NDMNDS,NPERAC*NBRKAC)
C---
C---       *END IF (ZERO BROKEN AC TEST)
  300      CONTINUE
C---
   RETURN
   END
```

```
C***********************************************************************
   INTEGER FUNCTION NORSAC(NPARTS,IQPA,NBACKO)
C***********************************************************************
C++ NORSAC     - CALCULATE INITIAL NUMBER OF NORS AIRCRAFT.
C***      NORSAC IS A FORTRAN FUNCTION WHICH CALCULATES THE CURRENT
C*** NUMBER OF NORS AIRCRAFT - ASSUMING PERFECT CANNIBALIZATION.
C***
C*** INPUT -
C***   NPARTS     - TOTAL NUMBER OF PART TYPES.
C***   IQPA(K)    - NUMBER OF TYPE-K PARTS INSTALLED ON EACH AIRCRAFT.
C***   NBACKO(K)  - NUMBER OF BACKORDERS FOR PARTS OF TYPE-K.IF
C***                NBACKO(K) IS POSITIVE, THEN UNFULFILLED REQUESTS
C***                FOR PARTS OF THIS TYPE HAVE BEEN MADE. IF IT IS
C***                NEGATIVE, THEN NBACKO(K) INDICATES THE NUMBER OF
C***                OF PARTS ON-THE-SHELF.
C*** OUTPUT -
C***   NORSAC     - CURRENT NUMBER OF NORS AIRCRAFT BASED ON THE GIVEN
C***                BACKORDER AND QPA INFORMATION AND ASSUMING PERFECT
C***                CANNABILIZATION.
C***********************************************************************
C---
      DIMENSION NBACKO(NPARTS), IQPA(NPARTS)
C---
C---        *INITIALIZE NUMBER OF NORS AIRCRAFT TO NONE
             NORSAC = 0
C---
C---        *DO FOR(EACH PART TYPE)
             DO 2000 K=1,NPARTS
C---
C---           *IF(THESE PARTS CAUSE THE MAX NUMBER OF NORS THUS FAR)
                IF(NORSAC*IQPA(K) .GE. NBACKO(K)) GO TO 1000
C---
C---              *UPDATE NUMBER OF NORS AIRCRAFT
                   NORSAC=INT(FLOAT(NBACKO(K))/FLOAT(IQPA(K)) + .999)
C---
C---           *END IF (NEW NORS MAXIMUM TEST)
 1000            CONTINUE
C---
C---        *END DO (PARTS LOOP)
 2000        CONTINUE
C---
   RETURN
   END
```

```
C***************************************************************
   INTEGER FUNCTION NORSBK(NBRKAC,NOROLD)
C***************************************************************
C++ NORSBK     - DETERMINES NORS AIRCRAFT FROM A SORTIE.
C***      NORSBK IS A FORTRAN FUNCTION WHICH CALCULATES THE NUMBER
C*** OF NORS AIRCRAFT RESULTING FROM A SORTIE WITH A SPECIFIED NUMBER
C*** OF BROKEN AIRCRAFT -- ASSUMING IMMEDIATE AND MAXIMUM
C*** CANNABILIZATION OF PARTS. NORSBK DETERMINES THE TOTAL
C*** NUMBER AND DISTRIBUTION OF THE PARTS DEMANDS RESULTING FROM THIS
C*** SORTIE. IT UPDATES FOR EACH PART TYPE DEMANDED, THE
C*** NUMBER OF PARTS ON-THE-SHELF, BACKORDERED, AND IN RESUPPLY.
C***
C*** INPUTS --
C***   NBRKAC     - NUMBER OF AIRCRAFT WHICH BROKE DURING THE SORTIE
C***   NOROLD     - NUMBER OF NORS AIRCRAFT BEFORE THIS LATEST SORTIE.
C*** COMMON INPUTS --
C***   IQPA(K)    - NUMBER OF TYPE-K PARTS INSTALLED ON EACH AIRCRAFT.
C***   INFPART    - LOGICAL FLAG INDICATING WHETHER THE INFINITE PARTS
C***                ASSUMPTION HOLDS. IF INFPART IS TRUE THEN THERE
C***                IS NEVER ANY SHORTAGE OF PARTS; HENCE, NO NORS AC.
C*** COMMON INPUTS/OUTPUTS -
C***   NBACKO(K)  - NUMBER OF BACKORDERS FOR PARTS OF TYPE-K.IF
C***                NBACKO(K) IS POSITIVE, THEN UNFULFILLED REQUESTS
C***                FOR PARTS OF THIS TYPE HAVE BEEN MADE. IF IT IS
C***                NEGATIVE, THEN NBACKO(K) INDICATES THE NUMBER
C***                OF PARTS ON-THE-SHELF.
C*** OUTPUT -
C***   NORSBK      - NUMBER OF NORS AIRCRAFT AT THE END OF THIS SORTIE
C***                 ASSUMING MAXIMUM AND IMMEDIATE CANNABILIZATION.
C***************************************************************
C---
      PARAMETER MAXPRT=304, MAXCYC=10
      COMMON /PARTS/ NPARTS,IQPA(MAXPRT),NBACKO(MAXPRT),
&        BRPRATE(MAXPRT),DRPRATE(MAXPRT),INITSJ(MAXPRT),RESUPP(MAXPRT),
&        BNRTS(MAXPRT),NBASE(MAXPRT),NDEPOT(MAXPRT)
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      LOGICAL INFPART
C---
C---        *INITIALIZE NEW NORS TO OLD NUMBER OF NORS AIRCRAFT
             NORSBK = NOROLD
C---
C---        *IF(THERE ARE ANY BROKEN AIRCRAFT AND
C---            INFINITE PARTS NOT ASSUMED)THEN
             IF(NBRKAC.EQ.0) GO TO 5000
             IF(INFPART) GO TO 5000
C---
C---           *DETERMINE TOTAL NUMBER OF PARTS DEMANDS FROM THESE BROKEN AC
                NDEMS = NDMNDS(NBRKAC)
C---
C---           *IF(ANY PARTS WERE DEMANDED)THEN
                IF(NDEMS.EQ.0) GO TO 4000
C---
```

```
C---            *DO FOR(EACH PART DEMAND)
                 DO 3000 I=1,NDEMS
C---
C---              *DETERMINE PART-TYPE FOR THIS DEMAND BY SAMPLING
C---               FROM A MULTNOMIAL DISTRIBUTION
                   KTYPE = MNOM()
C---
C---              *UPDATE BACKORDERS FOR THIS PART TYPE
                   NBACKO(KTYPE) = NBACKO(KTYPE) + 1
C---
C---              *IF(AN UNFULFILLED DEMAND HAS OCCURRED)THEN
                   IF(NBACKO(KTYPE).LE.0) GO TO 2000
C---
C---                *IF(THIS DEMAND CAUSES A NEW NORS AC)THEN
                     IF(NORSBK*IQPA(KTYPE).GE.NBACKO(KTYPE))
&                      GO TO 1000
C---
C---                  *INCREMENT NUMBER OF NORS AIRCRAFT
                       NORSBK = NORSBK + 1
                      IF((NORSBK-NOROLD).GE.NBRKAC) GOTO 5000
C---
C---                *END IF (NEW NORS AIRCRAFT TEST)
 1000                CONTINUE
C---
C---              *END IF (UNFULFILLED DEMAND TEST)
 2000              CONTINUE
C---
C---            *END DO (DEMAND LOOP)
 3000            CONTINUE
C---
C---          *END IF (ZERO DEMANDS TEST)
 4000          CONTINUE
C---
C---        *END IF (NO BROKEN AC OR INFINITE PARTS TEST)
 5000        CONTINUE
C---
   RETURN
   END
```

```
C****************************************************************
   INTEGER FUNCTION NREPS(TIMET,NREPJ,NCRWSJ,SRATEJ)
C****************************************************************
C++ NREPS      - RANDOM SAMPLE OF AIRCRAFT REPAIRS IN A WORK CENTER.
C***     NREPS IS A FORTRAN FUNCTION WHICH RETURNS A SAMPLE NUMBER
C*** OF AIRCRAFT REPAIRED IN A WORKCENTER, BASED ON THE LENGTH OF THE
C*** REPAIR PERIOD, NUMBER OF SERVERS, REPAIR RATE FOR EACH SERVER,
C*** AND THE NUMBER OF AIRCRAFT IN THE WORKCENTER AT THE START OF
C*** THE REPAIR PERIOD. IT IS ASSUMED THAT NO NEW AIRCRAFT ARRIVE
C*** DURING THE REPAIR PERIOD, AND REPAIR TIMES ARE EXPONENTIALLY
C*** DISTRIBUTED, WITH THE SAME DISTRIBUTION APPYING TO EACH SERVER
C*** INDEPENDENTLY.
C***
C*** INPUTS -
C***   TIMET      - LENGTH (IN HOURS) OF THE REPAIR PERIOD.
C***   NREPJ      - NUMBER OF AIRCRAFT IN THE WORKCENTER AT THE
C***                START OF THE REPAIR PERIOD.
C***   NCRWSJ     - NUMBER OF REPAIR CREWS (SERVERS) FOR THIS
C***                WORKCENTER.
C***   SRATEJ     - REPAIR RATE (AIRCRAFT/HOUR) FOR EACH CREW IN
C***                THIS WORKCENTER.
C*** COMMON INPUT/OUTPUT --
C***   SEED       - SEED FOR RANDOM NUMBER GENERATOR.
C*** OUTPUT -
C***   NREPS      - NUMBER OF AIRCRAFT REPAIRED IN THIS WORKCENTER
C***                DURING THE REPAIR PERIOD.  NREPS=0,1,2,...,NREPJ
C****************************************************************
C---
      COMMON /RSEED/SEED
C---
C---        *IF(NUMBER OF AC IN REPAIR IS LESS THAN THE NUMBER OF CREWS)
            IF(NREPJ.GT.NCRWSJ) GO TO 1000
C---
C---           *DETERMINE NUMBER OF AIRCRAFT REPAIRED BY SAMPLING
C---                     FROM THE APPROPRIATE BINOMIAL DISTRIBUTION
               NREPS = NREPJ - NBINOM( EXP(-SRATEJ*TIMET) , NREPJ )
C---
C---        *ELSE (MORE AIRCRAFT THAN CREWS)
            GO TO 4000
 1000       CONTINUE
C---
C---           *INITIALIZE VARIABLES
               NREPS  = 0
               CUMP   = 1.0
               MAXREP = NREPJ - NCRWSJ + 1
               CWRATE = FLOAT(NCRWSJ)*SRATEJ
               EXPTYM = EXP(-CWRATE*TIMET)
C---
C---           *DO UNTIL(A SERVER BECOMES IDLE OR THE NEXT AIRCRAFT
C---                  DEPARTURE TIME EXCEEDS LENGTH OF REPAIR PERIOD)
 2000          CONTINUE
C---
C---              *GENERATE AND ACCUMULATE NEXT AIRCRAFT DEPARTURE FROM
```

```
C--                                           THIS WORKCENTER
          CUMP = CUMP * UNIFM1(SEED)
C--
C--      *EXIT LOOP, IF REPAIR TIMES EXCEED TIME INTERVAL LENGTH
          IF(CUMP .LT. EXPTYM) GO TO 3000
C--
C--      *INCREMENT NUMBER OF AIRCRAFT REPAIRED
          NREPS = NREPS + 1
C--
C--     *END DO (REPAIRED AIRCRAFT LOOP)
        IF(NREPS.LT.MAXREP) GO TO 2000
C--
C--     **A SERVER HAS JUST BECOME IDLE, PERFORM A BINOMIAL DRAW
C--       TO DETERMINE HOW MANY MORE AC ARE REPAIRED
C--
C--      *COMPUTE TIME LEFT IN THE INTERVAL
C--         (LENGTH OF REPAIR PERIOD) - (TIME OF LAST REPAIR)
          TLEFT = TIMET + ALOG(CUMP)/CWRATE
C--
C--      *COMPUTE PROBABILITY AN AIRCRAFT IS NOT REPAIRED
C--       IN THE REMAINDER OF THE INTERVAL
          PNOREP = EXP(-SRATEJ*TLEFT)
C--
C--      *GENERATE A BINOMIAL DRAW TO DETERMINE NUMBER OF
C--                  REMAINING AIRCRAFT WHICH ARE NOT REPAIRED
          NOTREP=NBINOM(PNOREP,NCRWSJ-1)
C--
C--      *COMPUTE TOTAL AIRCRAFT REPAIRED DURING PERIOD
          NREPS=NREPJ-NOTREP
C--
C--
C--     *EXIT FROM DO LOOP
 3000    CONTINUE
C--
C--    *END IF (MORE AIRCRAFT THAN CREWS TEST)
 4000   CONTINUE
C--
   RETURN
   END
```

```
C*****************************************************************
   SUBROUTINE PRINTO
C*****************************************************************
C++ PRINTO     - PRINT-OUT RESULTS OF THE SIMULATION RUN.
C***      THIS ROUTINE PRINTS THE RESULTS OF THE SGM SIMULATION.
C*** THESE RESULTS CONSIST OF THE AVERAGE NUMBERS OF AIRCRAFT IN
C*** THE VARIOUS POSSIBLE AIRCRAFT STATES AT THE START OF EACH
C*** SORTIE PERIOD FOR EACH FLYING DAY OF THE SCENARIO. THE AVERAGE
C*** SORTIES PER AIRCRAFT PER DAY IS COMPUTED AND ALSO PRINTED.
C*** THESE SORTIES PER AIRCRAFT PER DAY FIGURES AND THE TOTAL SORTIE
C*** PRODUCTION PER DAY ARE PRINTED TO A SCRATCH FILE (FILE 07) TO BE
C*** USED AS INPUT FOR SORTIE PLOTS.
C*****************************************************************
C---
      PARAMETER  MAXAC=108,MAXWC=25,MAXBIT=36,MAXPRT=304,
&                   MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER MAXDAY=30,MAXCYC=10,MAXSTAT=5
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                       NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
      COMMON /TIME/ PREFLITE,SORTLGTH,WAITCYC,
&       TYMNITE,NSIM,ISIM,NUMDAY,IDAY,NCYCLES,ICYCLE
      COMMON /INPUT/  INITUE,NAC,PATTRIT,IRES,RNMCM,INFPART,
&           MAXFLY(MAXCYC),INFMAN,ISCALE,IAUGMNT
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
C---
C---  *PRINT EXPECTED NUMBER OF AVAILABLE AIRCRAFT FOR EACH SORTIE PERIOD
       WRITE (6,9005)
       TOTFLY = 0.0
       WRITE (7) FLOAT(ISCALE),NUMDAY
       FSIM = FLOAT(NSIM)
       DO 6000 J=1,NUMDAY
          SORTYDAY = EXPECT(1,1,J)
          OFFSCENE = EXPECT(4,1,J)+EXPECT(5,1,J)
          WRITE(6,9008) J,1,(EXPECT(M,1,J)/FSIM,M=1,5)
          DO 5000 I=2,NCYCLES-1
             WRITE (6,9006) I,(EXPECT(M,I,J)/FSIM,M=1,4)
             SORTYDAY = SORTYDAY + EXPECT(1,I,J)
             OFFSCENE = OFFSCENE + EXPECT(4,I,J) + EXPECT(5,I,J)
 5000     CONTINUE
          I = NCYCLES
          SORTYDAY = (SORTYDAY + EXPECT(1,I,J))/FSIM
          OFFSCENE = (OFFSCENE + EXPECT(4,I,J) + EXPECT(5,I,J))/FSIM
          AONSCENE = NCYCLES*(INITUE+ITOTRES(J)) - OFFSCENE
          SORTYAC=NCYCLES*SORTYDAY/AONSCENE
          WRITE (6,9009) I,EXPECT(1,I,J)/FSIM,SORTYDAY,
&            SORTYAC,(EXPECT(M,I,J)/FSIM,M=2,4)
          TOTFLY = TOTFLY + SORTYDAY
C---
C---     *WRITE THE MEAN SORTIES PER DAY (FOR EACH DAY) TO A FILE THAT
C---         COULD BE USED BY 'CALLPLT2' TO PRODUCE A PLOT.
          WRITE (7) SORTYAC, SORTYDAY
          WRITE(6,9003) TOTFLY
 6000  CONTINUE
```

```
      WRITE (6,9010) TOTFLY
C---
   RETURN
 9002 FORMAT(V)
 9003 FORMAT (F26.1)
 9005 FORMAT ('1'///10X,'SORTIES/  SORTIES/  SORTIES/',21X,
&         'CUM.   RES.'/
&         ' DAY  PER  PERIOD',5X,'DAY',8X,'AC',4X,
&             '   NMCM     NMCS   LOSSES  LEFT'//)
 9006 FORMAT (' ',2X,I5,F9.1,F29.1,F9.1,F8.1)
 9008 FORMAT (' ',I2,I5,F9.1,F29.1,F9.1,F8.1,F8.1)
 9009 FORMAT (' ',2X,I5,F9.1,F9.1,F11.2,2F9.1,2F8.1)
 9010 FORMAT (/// TOTAL SORTIES FLOWN =',F11.1)
   END
```

```
C*****************************************************************
      SUBROUTINE PRTREP(RTIME,PBRKSEQ,INDXWC,NORSVC)
C*****************************************************************
C** PRTREP     - SIMULATES PROCESS OF REPAIRING PARTS.
C***      PRTREP IS A FORTRAN SUBROUTINE WHICH SIMULATES THE PROCESS
C*** OF REPAIRING PARTS. PARTS REPAIR IS ASSUMED TO BE
C*** EXPONENTIAL WITH AN INFINITE NUMBER OF SERVERS, I.E. NO PART
C*** EVER HAS TO WAIT TO BEGIN SERVICE. IN ADDITION TO REPAIRING
C*** PARTS, THIS FUNLTION CALCULATES THE NEW NUMBER OF NORS AIRCRAFT
C*** REMAINING AFTER REPAIR OF THESE PARTS. IF ANY PREVIOUSLY NORS
C*** AIRCRAFT ARE READY TO GO INTO MAINTENANCE, THIS ROUTINE
C*** WILL DISTRIBUTE THEM PROBABILISTICALLY AMONG THE VARIOUS
C*** WORKCENTERS.
C***
C*** INPUT -
C***   RTIME      - AVERAGE TIME (IN HOURS) THESE PARTS HAVE BEEN
C***                IN REPAIR SINCE THE LAST TIME PARTS REPAIR WAS
C***                SIMULATED.
C***   PBRKSEQ    - 2-DIMENSIONAL ARRAY USED TO DETERMINE THE
C***                DISTRIBUTION OF ABORTS INTO THE VARIOUS
C***                WORKCENTERS.
C*** INPUT/OUTPUT -
C***   NORSVC     - NORS AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE NORS DUE TO UNAVAILABLE PARTS.
C***                THE FIRST WORD, NORSVC(1), CONTAINS THE TOTAL
C***                NUMBER OF 1-BITS IN THE NORS STATUS VECTOR.
C***                ARRAY IS A BIT VECTOR WITH EACH BIT REPRESENTING
C***                AN AIRCRAFT. A 1-BIT INDICATES THE AIRCRAFT IS
C***                STILL FLYABLE. NOTE THAT IFLYVC(1) ALSO INDICATES
C***                THE NUMBER OF 1-BITS IN THIS BIT VECTOR.
C*** COMMON INPUT -
C***   INFPART    - LOGICAL FLAG INDICATING WHETHER THE INFINITE PARTS
C***                ASSUMPTION HOLDS. IF INFPART IS TRUE THEN THERE
C***                IS NEVER ANY SHORTAGE OF PARTS; HENCE, NO
C***                NORS AC.
C***   NPARTS     - NUMBER OF PART TYPES BEING MODELED.
C***   IQPA(K)    - NUMBER OF TYPE-K PARTS INSTALLED ON EACH AIRCRAFT.
C***   RPRATE(K)  - REPAIR RATE (PARTS/HOUR) FOR TYPE-K PARTS
C***   INITSJ(K)  - INITIAL BASE STOCK LEVEL FOR TYPE-K PARTS.
C*** COMMON INPUT/OUTPUT -
C***   NBACKO(K)  - NUMBER OF BACKORDERS FOR PARTS OF TYPE-K.IF
C***                NBACKO(K) IS POS'TIVE, THEN UNFULFILLED REQUESTS
C***                FOR PARTS OF THIS TYPE HAVE BEEN MADE. IF IT IS
C***                NEGATIVE, THEN NBACKO(K) INDICATES THE NUMBER OF
C***                OF PARTS ON-THE-SHELF.
C*****************************************************************
C---
      PARAMETER MAXPRT=304, MAXCYC=10
      COMMON /PARTS/ NPARTS,IQPA(MAXPRT),NBACKO(MAXPRT),
     &    BRPRATE(MAXPRT),DRPRATE(MAXPRT),INITSJ(MAXPRT),RESUPP(MAXPRT),
     &    BNRTS(MAXPRT),NBASE(MAXPRT),NDEPOT(MAXPRT)
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
     &                 MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
```

```
      DIMENSION NORSVC(1)
      LOGICAL INFPART
C---
C---    *INITIALIZE NEW NUMBER OF NORS AIRCRAFT TO NONE
        NEWNOR = 0
C---
C---    *IF(INFINITE PARTS NOT ASSUMED)THEN
        IF(INFPART) GO TO 5000
C---
C---      *DO FOR(EACH PART TYPE)
          DO 3000 K=1,NPARTS
C---
C---        *IF(THERE ARE ANY OF THIS PART IN REPAIR)THEN
            INSHPK = NBACKO(K) + INITSJ(K)
            IF(INSHPK.LE.0) GO TO 2000
C---          *DETERMINE NUMBER OF THESE WHICH ARE NEW DEMANDS
              NEW = INSHPK - (NDEPOT(K) + NBASE(K))
C---
C---          *PERFORM BINOMIAL DRAW TO DETERMINE BASE/DEPOT SPLIT
              NEWDEP=NBINOM(BNRTS(K),NEW)
              NDEPOT(K)=NDEPOT(K)+NEWDEP
              NBASE(K)=NBASE(K)+(NEW-NEWDEP)
C---
C---          *COMPUTE PROBABILITY OF REPAIR
              PDEP = 1.0 - EXP(-DRPRATE(K)*RTIME)
              PBSE = 1.0 - EXP(-BRPRATE(K)*RTIME)
C---
C---          *DETERMINE NUMBER OF PARTS REPAIRED BY SAMPLING
C---           FROM THE APPROPRIATE BINOMIAL DISTRIBUTION
              NUMDEP = NBINOM(PDEP,NDEPOT(K))
              NUMBSE = NBINOM(PBSE,NBASE(K))
C---
C---          *UPDATE NUMBER IN-SHOP AND BACKORDERED
              NBACKO(K) = NBACKO(K) - (NUMDEP + NUMBSE)
              NDEPOT(K) = NDEPOT(K) - NUMDEP
              NBASE(K) = NBASE(K) - NUMBSE
C---
C---          *IF(THESE PARTS CAUSE THE MAX NUMBER OF NORS THUS FAR)
              IF(NEWNOR*IQPA(K) .GE. NBACKO(K)) GO TO 1000
C---
C---            *UPDATE NUMBER OF NORS AIRCRAFT
                NEWNOR=INT(FLOAT(NBACKO(K))/FLOAT(IQPA(K)) + .999)
C---
C---          *END IF (NEW NORS MAXIMUM TEST)
 1000         CONTINUE
C---
C---        *END IF (NO PARTS IN REPAIR TEST)
 2000       CONTINUE
C---
C---      *END DO (PARTS LOOP)
 3000     CONTINUE
C---
C---      *IF(ANY PREVIOUSLY NORS AIRCRAFT ARE READY FOR REPAIR)THEN
```

```
          NORDIF = NORSVC(1) - NEWNOR
          IF(NORDIF.LE.0) GO TO 4000
C---
C---        *SELECT LEFTMOST NORS AIRCRAFT TO ENTER MAINTENANCE
             CALL WCDIST(NORDIF,PBRKSEQ,INDXWC,NORSVC)
C---
C---       *END IF (NEW NONNORS AC TEST)
 4000      CONTINUE
C---
C---    *END IF (INFINITE PARTS TEST)
 5000    CONTINUE
C---
   RETURN
   END
```

```
C*****************************************************************
      SUBROUTINE PSTAT(PBREAK,NPARTS,IQPA,DEMAND,
     &                                        ACMEAN,ACVAR,NPERAC)
C*****************************************************************
C++ PSTAT      - CALCULATES STATISTICS FOR TOTAL PART DEMANDS.
C***       PSTAT IS A FORTRAN ROUTINE WHICH CALCULATES THE MEAN
C*** AND VARIANCE OF THE RANDOM VARIABLE REPRESENTING THE TOTAL
C*** NUMBER OF PART DEMANDS FROM AN AIRCRAFT WHICH HAS BROKEN UPON
C*** RETURNING FROM A SORTIE.
C***
C*** INPUTS -
C***   PBREAK     - PROBABILITY THAT AN AIRCRAFT BREAKS UPON RETURNING
C***                FROM A SORTIE.
C***   NPARTS     - NUMBER OF PART TYPES BEING MODELED FOR THIS TYPE
C***                OF AIRCRAFT.
C***   IQPA(K)    - QUANTITY PER AIRCRAFT FOR TYPE-K PARTS.
C***   DEMAND(K)  - PROBABILITY THAT A GIVEN TYPE-K PART WILL BE
C***                DEMANDED BY AN AIRCRAFT RETURNING FROM A SORTIE.
C***  OUTPUTS -
C***   ACMEAN     - MEAN OF THE RANDOM VARIABLE REPRESENTING NUMBER
C***                OF PART DEMANDS PER BROKEN AIRCRAFT.
C***   ACVAR      - VARIANCE OF TOTAL PART DEMANDS PER BROKEN AIRCRAFT.
C***   NPERAC     - TOTAL NUMBER OF PARTS PER AIRCRAFT. THIS IS USED
C***                TO ENSURE THAT A LEGITIMATE SAMPLE IS GENERATED.
C*****************************************************************
C---
       DIMENSION IQPA(NPARTS), DEMAND(NPARTS)
C---
C---  *INITIALIZE STATISTICS
        ACMEAN = 0.0
        ACVAR  = 0.0
        NPERAC = 0
C---
C---  *DO FOR(EACH PART TYPE)
        DO 1000  K=1,NPARTS
C---
C---     *ACCUMULATE STATISTICS
           PRTYPE = IQPA(K) * DEMAND(K)
           ACMEAN = ACMEAN + PRTYPE
           ACVAR  = ACVAR + PRTYPE*(PBREAK-DEMAND(K))
           NPERAC = NPERAC + IQPA(K)
C---
C---  *END DO (PART LOOP)
 1000  CONTINUE
C---
C---  *COMPLETE MEAN/VARIANCE COMPUTATIONS
        ACMEAN = ACMEAN/PBREAK
        ACVAR  = ACVAR/(PBREAK*PBREAK)
C---
      RETURN
      END
```

```
C*************************************************************
   SUBROUTINE REPAIR(TIMET,NWC,NCREWS,SRATE)
C*************************************************************
C++ REPAIR     - SIMULATES PROCESS OF WORK CENTER AIRCRAFT REPAIR.
C***      REPAIR IS A FORTRAN ROUTINE WHICH SIMULATES THE
C*** PROCESS OF REPAIRING AIRCRAFT IN ALL WORKCENTERS DURING A
C*** SPECIFIED TIME PERIOD. A NUMBER OF REPAIRED AIRCRAFT IS GENERATED
C*** FOR EACH WORKCENTER, BY SAMPLING FROM THE APPROPRIATE PROBABILITY
C*** DISTRIBUTIONS. THIS ROUTINE SIMPLY UPDATES THE TOTAL NO. OF
C*** AIRCRAFT IN REPAIR IN EACH WORKCENTER; IT DOES NOT CONCERN
C*** ITSELF WITH WHICH PARTICULAR AIRCRAFT IN A W/C ARE
C*** BEING REPAIRED, NOR THE TOTAL NO. OF AIRCRAFT IN MAINTENANCE.
C*** IMPLICITLY, IT IS ASSUMED THAT WE ARE REPAIRING THE RIGHTMOST
C*** AIRCRAFT ON A LIST OF AIRCRAFT THAT NEED WORK IN A GIVEN W/C,
C*** AND THAT IF AIRCRAFT ARE PLACED ON THAT LIST (IN ROUTINE
C*** 'WCDIST') IN A RANDOM ORDER, THEN THIS METHOD OF REPAIR IS
C*** ALSO RANDOM; I.E., IT DOESN'T FAVOR LOW-NUMBERED A/C, OR
C*** HIGH-NUMBERED A/C, OR RECENTLY-BROKEN A/C, ETC.
C***
C*** INPUT --
C***   TIMET      - LENGTH (IN HOURS) OF THE REPAIR PERIOD.
C***   NWC        - TOTAL NUMBER OF WORKCENTERS.
C***   NCREWS     - NUMBER OF CREWS IN WORKCENTER-J.
C***   SRATE(J)   - REPAIR RATE (AIRCRAFT/HOUR) FOR EACH CREW IN
C***                WORKCENTER-J.
C*** COMMON INPUTS/OUTPUTS --
C***   INREPR(J)  - NO. OF A/C IN MAINTENANCE IN W/C J.
C*************************************************************
C---
      PARAMETER MAXWC=25,MAXBIT=36,MAXAC=108,MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER  MAXDAY=30
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      DIMENSION NCREWS(NWC), SRATE(NWC)
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC),INREPR(MAXWC)
C---
C---        *DO FOR(EACH WORKCENTER)
             DO 400  J=1,NWC
C---
C---           *IF(THERE ARE ANY AIRCRAFT IN REPAIR IN THIS WORKCENTER)
                NACINJ = INREPR(J)
                IF(NACINJ.EQ.0) GO TO  200
C---
C---              *GENERATE SAMPLE NUMBER OF AIRCRAFT REPAIRED
                   NFIXED = NREPS(TIMET,NACINJ,NCREWS(J),SRATE(J))
C---
C---              *UPDATE NO. OF A/C IN THIS W/C
                   INREPR(J) = NACINJ - NFIXED
C---
C---           *END IF (ZERO AIRCRAFT IN REPAIR TEST)
  200           CONTINUE
C---
C---        *END DO (WORKCENTER LOOP)
  400        CONTINUE
```

```
C---
      RETURN
      END
```

```
C*****************************************************************
      SUBROUTINE SIMULA
C*****************************************************************
C** SIMULA     - PERFORM SIMULATION REPLICATIONS.
C***      THIS ROUTINE PROVIDES THE BASIC STRUCTURE OF THE SGM
C*** SIMULATION. THE LOOPS WHICH CONTROL THE SIMULATION REPLICATIONS,
C*** THE FLYING DAYS FOR EACH REPLICATION, AND THE FLYING CYCLES
C*** FOR EACH FLYING DAY ARE CONTAINED IN THIS ROUTINE. THE
C*** ROUTINE BASICALLY JUST EXECUTES A SPECIFIED NUMBER OF
C*** FLYING CYCLES EACH FLYING DAY.
C***    THE ROUTINE READS A SUBSET OF SCENARIO PARAMETERS FOR EACH
C*** FLYING DAY OF THE SCENARIO FROM A SCRATCH FILE (03) WHICH
C*** WAS INITIALIZED IN THE INITSCN SUBROUTINE. THIS APPROACH ALLOWS
C*** THESE PARAMETERS TO VARY ON A DAILY BASIS DURING THE
C*** SIMULATION.
C***
C*** COMMON INPUTS --
C***    NSIM       - NUMBER OF SIMULATION REPLICATIONS TO BE PERFORMED
C***    NUMDAY     - NUMBER OF FLYING DAYS TO SIMULATE
C***    NCYCLES    - NUMBER OF FLYING CYCLES FOR EACH DAY. THIS PARAMETER
C***                 IS READ FROM THE SCRATCH FILE EACH SIMULATION DAY
C*** COMMON OUTPUTS --
C***     ISIM      - (ISIM=1,...,NSIM) CURRENT REPLICATION NUMBER
C***     IDAY      - (IDAY=1,..,NUMDAY) CURRENT FLYING DAY
C***     ICYCLE    - (ICYCLE=1,...,NCYCLES) CURRENT FLYING CYCLE
C***     NRESRV    - CURRENT NUMBER OF AIRCRAFT IN THE RESERVE POOL
C***     ITOTRES(I)- (I=0,...,NUMDAY) CUMULATIVE NUMBER OF AVAILABLE
C***                 RESERVE AIRCRAFT UP TO AND INCLUDING THE ITH DAY.
C***                 THE 0TH DAY REPRESENTS THE INITIAL NUMBER OF RESERVE
C***                 AIRCRAFT. THIS ARRAY IS USED IN COMPUTING ON-THE-
C***                 SCENE AIRCRAFT FOR SORTIES/AIRCRAFT/DAY IN THE
C***                 PRINTO ROUTINE.
C***     IRES      - NUMBER OF RESERVE AIRCRAFT WHICH BECOME AVAILABLE
C***                 ON THIS FLYING DAY.
C***     PATTRIT,PACGABT,WAITCYC,TYMNITE,MAXFLY(I)
C***               - SCENARIO PARAMETERS WHICH ARE ALLOWED TO VARY
C***                 ON A DAILY BASIS. THEY ARE LOADED IN THIS
C***                 ROUTINE AND SUPPLIED TO THE FLYCYC ROUTINE
C*****************************************************************
C---
      PARAMETER MAXWC=25, MAXDAY=30, MAXCYC=10, MAXSTAT=5
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
     &                 MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
     &                 NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
      COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
     &                 NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
     &                 PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
C---
C---  *DO FOR(EACH SIMULATION REALIZATION)
       DO 300 ISIM=1,NSIM
C---
```

```
C---    *INITIALIZE VARIABLES AT START OF EACH REPLICATION
         CALL INITREP
C---
C---    *REWIND DAILY SCENARIO PARAMETERS FILE
         REWIND 03
C---
C---    *DO FOR(EACH FLYING DAY)
         DO 200 IDAY=1,NUMDAY
C---
C---       *READ DAILY SCENARIO PARAMETERS
            READ(03) PATTRIT,PACGABT,NCYCLES,IRES,WAITCYC,TYMNITE
            READ(03) (MAXFLY(J),J=1,NCYCLES)
C---
C---       *UPDATE RESERVE AC POOL WITH NEW AVAILABLE RESERVES
            NRESRV = NRESRV + IRES
            ITOTRES(IDAY) = ITOTRES(IDAY-1) + IRES
C---
C---       *DO FOR(EACH FLYING CYCLE)
            DO 100 ICYCLE=1,NCYCLES
C---
C---          *SIMULATE A FLYING CYCLE
               CALL FLYCYC
C---
C---       *END DO (CYCLE LOOP)
  100       CONTINUE
C---
C---    *END DO (DAY LOOP)
  200    CONTINUE
C---
C--- *END DO (REPLICATION LOOP)
  300 CONTINUE
C---
   RETURN
   END
```

```
C****************************************************************
      SUBROUTINE SORTDS(NRECS,RKEYS,INDEX)
C****************************************************************
C++ SORTDS     - DESCENDING SORT OF A REAL ARRAY.
C***      THIS ROUTINE RETURNS A SORTED LIST OF INDICES ACCORDING
C*** TO THE GIVEN ARRAY OF FLOATING-POINT KEYS.  THIS IS A
C*** DESCENDING SORT (I.E. LARGEST TO SMALLEST) USING THE
C*** SIMPLE EXCHANGE SORT TECHNIQUE.
C***
C*** INPUTS --
C***       NRECS    - NUMBER OF ENTRIES OR RECORDS IN THE INPUT AND
C***                  OUTPUT ARRAYS.
C***       RKEYS    - ARRAY OF FLOATING-POINT KEYS WHOSE CORRESPONDING
C***                  INDICES ARE TO BE SORTED.
C*** OUTPUTS --
C***       INDEX    - ARRAY OF INDICES, 1,...,NRECS , SORTED IN DESCENDING
C***                  ORDER ACCORDING TO THE CORRESPONDING RKEY ENTRY.
C***                  THUS, RKEY( INDEX(1) ) IS THE LARGEST ENTRY, AND
C***                  RKEY( INDEX(NRECS) ) IS THE SMALLEST.
C****************************************************************
C---
      DIMENSION RKEYS(NRECS), INDEX(NRECS)
C---
C---  *INITIALIZE INDEX ARRAY TO NATURAL ORDER
       DO 100 I=1,NRECS
          INDEX(I)=I
  100  CONTINUE
C---
C---  *IF(THERE IS MORE THAN 1 RECORD)THEN
       IF(NRECS.LE.1)GO TO 500
C---
C---     *DO UNTIL(NO INTERCHANGES OCCUR)
          LIMIT=NRECS-1
  200     CONTINUE
C---
C---        *INITIALIZE INTERCHANGE FLAG TO NONE
             LASTX=0
C---
C---        *DO FOR(EACH CNSECUTIVE PAIR OF RECORDS)
             DO 400 J=1,LIMIT
C---
C---           *IF(INDICES OF THESE RECORDS SHOULD BE INTERCHANGED)
                IFIRST = INDEX(J)
                ISEC   = INDEX(J+1)
                IF(RKEYS(IFIRST).GE.RKEYS(ISEC)) GO TO 300
C---
C---              *INTERCHANGE INDICES AND MARK LAST EXCHANGE LOCATION
                   INDEX(J)   = ISEC
                   INDEX(J+1) = IFIRST
                   LASTX = J
C---
C---           *END IF (INTERCHANGE TEST)
  300           CONTINUE
```

```
C---
C---        *END DO (RECORD PAIR LOOP)
  400        CONTINUE
C---
C---     *END DO (INTERCHANGE PASS LOOP)
         LIMIT = LASTX
         IF(LASTX.NE.0) GO TO 200
C---
C---  *END IF (SINGLE RECORD TEST)
  500  CONTINUE
C---
   RETURN
   END
```

```
C*****************************************************************
      SUBROUTINE TBITSL(NONES,IFROM,ITO)
C*****************************************************************
C++ TBITSL     - TRANSFER 1-BITS FROM LEFT OF A BIT-VECTOR.
C***      TBITSL IS A FORTRAN SUBROUTINE WHICH WILL TRANSFER
C*** A SPECIFIED NUMBER OF 1-BITS IN THE LEFTMOST PORTION OF
C*** A BIT-VECTOR, 'IFROM', TO THE CORRESPONDING POSITIONS
C*** OF A BIT-VECTOR, 'ITO'. THE TRANSFERRED BITS ARE THEN
C*** ZEROED OUT IN 'IFROM'. THESE BIT-VECTORS ARE KEPT IN
C*** ARRAYS, ORGANIZED IN THE FOLLOWING MANNER - THE FIRST WORD
C*** OF THE ARRAY CONTAINS THE CURRENT NUMBER OF 1-BITS IN THE
C*** BIT-VECTOR, AND THE REMAINING WORDS OF THE ARRAY CONTAIN THE
C*** ACTUAL BIT-VECTOR.
C***
C*** INPUT -
C***   NONES      - NUMBER OF 1-BITS TO BE TRANSFERRED. NOTE THAT
C***                NONES MUST NOT BE GREATER THAN THE NUMBER OF
C***                1S IN THE BIT STRING, 'IFROM.
C*** INPUT/OUTPUT -
C***   IFROM      - ARRAY CONTAINING THE BIT-VECTOR FROM WHICH
C***                THE LEFTMOST 1-BIT POSITIONS ARE TO BE
C***                TRANSFERRED. NOTE THAT THESE LEFTMOST
C***                1-BITS ARE ZEROED OUT IN 'IFROM' UPON
C***                COMPLETION OF THIS ROUTINE. IFROM(1) IS
C***                A COUNTER WHICH INDICATES THE CURRENT
C***                NUMBER OF 1-BITS IN THE BIT-VECTOR. THE
C***                ACTUAL BIT-VECTOR IS THE CONSECUTIVE BITS
C***                CONTAINED IN THE WORDS IFROM(2)-IFROM(LENGTH)
C***    ITO       - ARRAY CONTAINING THE BIT-VECTOR TO WHICH
C***                THE LEFTMOST 1-BITS IN 'IFROM' ARE TO BE
C***                TRANSFERRED. THE RIGHTMOST POSITIONS IN ITO
C***                REMAIN AS BEFORE. ITO(1) IS A COUNTER WHICH
C***                INDICATES THE CURRENT NUMBER OF 1-BITS IN
C***                THE BIT-VECTOR. THE ACTUAL BIT VECTOR IS THE
C***                CONSECUTIVE BITS CONTAINED IN THE WORDS -
C***                        ITO(2) - ITO(LENGTH)
C*** COMMON INPUT -
C***   LENGTH     - LENGTH (IN COMPUTER WORDS) OF THE ARRAYS
C***                CONTAINING THE VARIOUS BIT-VECTORS; IFLYVC, ETC
C*****************************************************************
C---
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
     &                 MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION IFROM(1), ITO(1)
C---
C---      *INITIALIZE TRANSFER LOOP
          NLEFT  = NONES
          INDEX  = 2
C---
C---      *DO WHILE(ALL APPROPRIATE WORDS HAVE NOT BEEN MODIFIED)
 1000     CONTINUE
          IF(NLEFT.LE.0)      GO TO 4000
```

```
         IF(INDEX.GT.LENGTH) GO TO 4000
C---
C---       *COUNT NUMBER OF 1S IN NEXT WORD
            NXT1S  = N1BITS( IFROM(INDEX) )
C---
C---       *IF(NOT ALL 1-BITS IN THIS WORD SHOULD BE TRANSFERRED)
            IF(NXT1S.LE.NLEFT) GO TO 2000
C---
C---         *TRANSFER THE PROPER NUMBER OF 1S
              LTRANS       = LBITS( IFROM(INDEX) ,NLEFT)
              ITO(INDEX)   = OR(ITO(INDEX),LTRANS)
              IFROM(INDEX) = XOR( IFROM(INDEX) ,LTRANS)
C---
C---       *ELSE (ALL 1-BITS IN THIS WORD ARE TO BE TRANSFERRED)
            GO TO 3000
 2000       CONTINUE
C---
C---         *TRANSFER THE ENTIRE WORD
              ITO(INDEX)   = OR(ITO(INDEX),IFROM(INDEX))
              IFROM(INDEX) = 0
C---
C---       *END IF (ALL 1S TEST)
 3000       CONTINUE
C---
C---       *UPDATE NUMBER OF 1S LEFT TO TRANSFER
            NLEFT = NLEFT - NXT1S
C---
C---       *INCREMENT INDEX FOR NEXT WORD OF BIT-VECTOR
            INDEX = INDEX + 1
C---
C---     *END DO (WORD LOOP)
          GO TO 1000
 4000     CONTINUE
C---
C---     *UPDATE 1S COUNTER FOR THESE BIT-VECTORS
          ITO(1)   = ITO(1)   + NONES
          IFROM(1) = IFROM(1) - NONES
C---
C---     *PERFORM ERROR CHECK TO ENSURE APPROPRIATE NUMBER
C---                                  OF 1-BITS WAS TRANSFERRED
          IF(NLEFT.GT.0) WRITE(6,9001)NONES,IFROM(1),NLEFT
C---
   RETURN
 9001 FORMAT("0$$$$$$$$ TBITSL ERROR - TOO FEW 1-BITS TO TRANSFER",
&   /,       " $$$$$$$$     NONES, IFROM(1), NLEFT = ",3I5)
   END
```

```
C****************************************************************
      SUBROUTINE TBITSR(NONES,IFROM,ITO)
C****************************************************************
C++ TBITSR     - TRANSFER 1-BITS FROM RIGHT OF A BIT-VECTOR.
C***       TBITSR IS A FORTRAN SUBROUTINE WHICH WILL TRANSFER
C*** A SPECIFIED NUMBER OF 1-BITS IN THE RIGHTMOST PORTION OF
C*** A BIT-VECTOR, 'IFROM', TO THE CORRESPONDING POSITIONS
C*** OF A BIT-VECTOR, 'ITO'. THE TRANSFERRED BITS ARE THEN
C*** ZEROED OUT IN 'IFROM'. THESE BIT-VECTORS ARE KEPT IN
C*** ARRAYS, ORGANIZED IN THE FOLLOWING MANNER - THE FIRST WORD
C*** OF THE ARRAY CONTAINS THE CURRENT NUMBER OF 1-BITS IN THE
C*** BIT-VECTOR, AND THE REMAINING WORDS OF THE ARRAY CONTAIN THE
C*** ACTUAL BIT-VECTOR.
C***
C*** INPUT -
C***   NONES      - NUMBER OF 1-BITS TO BE TRANSFERRED. NOTE THAT
C***                NONES MUST NOT BE GREATER THAN THE NUMBER OF
C***                1'S IN THE BIT STRING, 'IFROM.
C*** INPUT/OUTPUT -
C***   IFROM      - ARRAY CONTAINING THE BIT-VECTOR FROM WHICH
C***                THE RIGHTMOST 1-BIT POSITIONS ARE TO BE
C***                TRANSFERRED. NOTE THAT THESE RIGHTMOST
C***                1-BITS ARE ZEROED OUT IN 'IFROM' UPON
C***                COMPLETION OF THIS ROUTINE. IFROM(1) IS
C***                A COUNTER WHICH INDICATES THE CURRENT
C***                NUMBER OF 1-BITS IN THE BIT-VECTOR. THE
C***                ACTUAL BIT-VECTOR IS THE CONSECUTIVE BITS
C***                CONTAINED IN THE WORDS IFROM(2)-IFROM(LENGTH)
C***    ITO       - ARRAY CONTAINING THE BIT-VECTOR TO WHICH
C***                THE RIGHTMOST 1-BITS IN 'IFROM' ARE TO BE
C***                TRANSFERRED. THE LEFTMOST POSITIONS IN ITO
C***                REMAIN AS BEFORE. ITO(1) IS A COUNTER WHICH
C***                INDICATES THE CURRENT NUMBER OF 1-BITS IN
C***                THE BIT-VECTOR. THE ACTUAL BIT VECTOR IS THE
C***                CONSECUTIVE BITS CONTAINED IN THE WORDS
C***                       ITO(2) - ITO(LENGTH)
C*** COMMON INPUT -
C***   LENGTH     - LENGTH (IN COMPUTER WORDS) OF THE ARRAYS
C***                CONTAINING THE VARIOUS BIT-VECTORS; IFLYVC, ETC
C****************************************************************
C---
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
     &                MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION IFROM(1), ITO(1)
C---
C---      *INITIALIZE TRANSFER LOOP
          NLEFT  = NONES
          INDEX  = LENGTH
C---
C---      *DO WHILE(ALL APPROPRIATE WORDS HAVE NOT BEEN MODIFIED)
 1000     CONTINUE
          IF(NLEFT.LE.0)      GO TO 4000
```

```
          IF(INDEX .LE. 1) GO TO 4000
C---
C---        *COUNT NUMBER OF 1'S IN NEXT WORD
             NXT1S  = N1BITS( IFROM(INDEX) )
C---
C---        *IF(NOT ALL 1-BITS IN THIS WORD SHOULD BE TRANSFERRED)
             IF(NXT1S.LE.NLEFT) GO TO 2000
C---
C---          *TRANSFER THE PROPER NUMBER OF 1'S
               NTRANS       = LBITS( IFROM(INDEX) ,NXT1S-NLEFT)
               ITO(INDEX)   = OR(ITO(INDEX),XOR(IFROM(INDEX),NTRANS))
               IFROM(INDEX) = NTRANS
C---
C---        *ELSE (ALL 1-BITS IN THIS WORD ARE TO BE TRANSFERRED)
             GO TO 3000
 2000        CONTINUE
C---
C---          *TRANSFER THE ENTIRE WORD
               ITO(INDEX)   =  OR (ITO(INDEX),IFROM(INDEX))
               IFROM(INDEX) = 0
C---
C---        *END IF (ALL 1'S TEST)
 3000        CONTINUE
C---
C---        *UPDATE NUMBER OF 1'S LEFT TO TRANSFER
             NLEFT = NLEFT - NXT1S
C---
C---        *DECREMENT INDEX FOR NEXT WORD OF BIT-VECTOR
             INDEX = INDEX - 1
C---
C---      *END DO (WORD LOOP)
           GO TO 1000
 4000      CONTINUE
C---
C---      *UPDATE 1'S COUNTER FOR THESE BIT-VECTORS
           ITO(1)   = ITO(1)   + NONES
           IFROM(1) = IFROM(1) - NONES
C---
C---      *PERFORM ERROR CHECK TO ENSURE APPROPRIATE NUMBER
C---                                  OF 1-BITS WAS TRANSFERRED
           IF(NLEFT.GT.0) WRITE(6,9001)NONES,IFROM(1),NLEFT
C---
   RETURN
 9001 FORMAT("0$$$$$$$$ TBITSR ERROR - TOO FEW 1-BITS TO TRANSFER",
&   /,        " $$$$$$$$     NONES, IFROM(1), NLEFT = ",3I5)
   END
```

```
C*****************************************************************
      SUBROUTINE UEUPDAT(NAC)
C*****************************************************************
C++ UEUPDAT    - UPDATE UE-STRENGTH FOR SCENARIO.
C***      THIS ROUTINE IS USED TO UPDATE THE UE-STRENGTH OF THE
C*** SCENARIO. THE UE-STRENGTH IS INITIALIZED AT THE START OF
C*** THE FLYING SCENARIO AND NORMALLY DOES NOT CHANGE THROUGHOUT
C*** THE SIMULATION. HOWEVER, IF THE USER HAS SELECTED THE
C*** AUGMENTATION MODE FOR RESERVE AIRCRAFT, AND ENOUGH RESERVES
C*** ARE AVAILABLE TO MORE THAN REPLACE COMBAT LOSSES, THEN THE
C*** EXCESS RESERVES ARE USED TO INCREASE THE CURRENT UE-STRENGTH.
C***      INCREASING THE UE-STRENGTH REQUIRES RECALCULATION
C*** OF THE LENGTH OF THE AIRCRAFT-STATUS BIT-VECTORS, AND ALSO
C*** REINITIALIZATION OF THE TOTAL-AIRCRAFT-POPULATION VECTOR.
C***
C*** INPUT --
C***   NAC        - NEW UPDATED UE-STRENGTH.
C***   MAXAC      - PARAMETER INDICATING MAXIMUM ALLOWABLE
C***                AIRCRAFT IN THE CURRENT SGM CONFIGURATION.
C***   MAXBIT     - NUMBER OF BITS PER COMPUTER WORD.
C*** COMMON OUTPUT --
C***   LENGTH     - LENGTH (IN COMPUTER WORDS) OF THE AIRCRAFT
C***                BIT-VECTORS. LENGTH MUST BE LARGE ENOUGH SO THAT
C***                THE BIT-VECTORS ARE AT LEAST "NAC" BITS
C***                LONG AT "MAXBIT" BITS PER COMPUTER WORD.
C***   NAC(I)     - (I=1,2,...,LENGTH) BIT-VECTOR WITH FIRST "NAC"
C***                BITS SET TO 1. THIS BIT-VECTOR INDICATES THE
C***                TOTAL AIRCRAFT POPULATION FOR THE SCENARIO.
C*****************************************************************
C--
      PARAMETER MAXBIT=36, MAXAC=108, MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /BITS/MASK0,MASK(35), MLEFT0,MSKLFT(36),IZCOUT,
     &           ICOUNT(63)
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
     &                 MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
C--
C-- *PERFORM ERROR CHECK TO ENSURE NO UE-OVERFLOW
      IF(NAC.LE.MAXAC)GO TO 100
         WRITE(6,9001)NAC,MAXAC
         NAC = MAXAC
  100 CONTINUE
C--
C-- *RECOMPUTE LENGTH OF AIRCRAFT-STATUS BIT-VECTORS
      LENGTH = 2 + (NAC-1)/MAXBIT
C--
C-- *INITIALIZE TOTAL-AIRCRAFT-POPULATION BIT-VECTOR
      CALL SPRAY(MSKLFT(36),NACVC(2),LENGTH-1)
      NACVC(LENGTH)=MSKLFT(MOD(NAC-1,MAXBIT))+1
      NACVC(1)=NAC
C--
      RETURN
 9001 FORMAT("0$$$$$$$$ UEUPDAT ERROR - UE OVERFLOW",/,
     &       " $$$$$$$$     NAC, MAXAC = ",2I5,/,
```

```
&          " $$$$$$$$     NAC WILL BE TRUNCATED TO MAXAC")
   END
```

```
C*****************************************************************
   SUBROUTINE WCDIST(NBRKAC,PBRKSEQ,INDXWC,IACVC)
C*****************************************************************
C++ WCDIST     - DETERMINE BREAK DISTRIBUTION INTO WORK CENTERS.
C***      WCDIST IS A FORTRAN SUBROUTINE WHICH SIMULATES THE PROCESS
C*** OF AIRCRAFT BREAKING INTO WORKCENTERS. GIVEN A NUMBER OF AIRCRAFT
C*** WHICH HAVE BROKEN INTO AT LEAST ONE WORKCENTER - 1,2,...,NWC, THIS
C*** ROUTINE DETERMINES (BY SIMULATION) WHICH PARTICULAR WORKCENTERS
C*** THE AIRCRAFT BROKE INTO.
C***
C*** INPUTS --
C***   IACVC      - BIT VECTOR INDICATING AIRCRAFT FROM WHICH BROKEN
C***                AIRCRAFT ARE TO BE SELECTED. A 1-BIT
C***                INDICATES AN AIRCRAFT WHICH IS A CANDIDATE FOR
C***                ONE OF THE BROKEN AIRCRAFT. THIS ROUTINE ARBITRARILY
C***                SELECTS THE LEFTMOST 1-BITS AS THOSE AIRCRAFT WHICH
C***                WILL BREAK INTO MAINTENANCE. NORMALLY, IACVC
C***                IS 'IFLYVC' OR 'NORSVC'.
C***   NBRKAC     - NUMBER OF BROKEN AIRCRAFT WHICH ARE TO BE BROKEN
C***                INTO THE DIFFERENT WORKCENTERS. THE LEFTMOST
C***                'NBRKAC' 1-BITS IN 'IACVC' ARE SELECTED AS THE
C***                AIRCRAFT WHICH BROKE.
C*** COMMON INPUTS --
C***   NWC        - TOTAL NUMBER OF WORKCENTERS
C***   MASK(I)    - CONTAINS A 1 IN THE ITH BIT (COUNTING FROM THE LEFT)
C***                AND ZEROES EVERYWHERE ELSE. I=0,...,35
C*** COMMON INPUTS/OUTPUTS --
C***   INREPR(J)  - NUMBER OF AIRCRAFT IN WORKCENTER-J.
C***    LISTRP(I,J) - LISTRP( . ,J) IS A LIST OF AIRCRAFT NUMBERS
C***                 INDICATING THOSE AIRCRAFT REQUIRING MAINTENANCE IN
C***                 THE JTH WORK-CENTER (J=1,2,...,NWC). THIS LIST
C***                 CONTAINS EXACTLY INREPR(J) AIRCRAFT NUMBERS. TO SAVE
C***                 SPACE, THESE LISTS HAVE BEEN PACKED INTO BIT-FIELDS
C***                 INSTEAD OF WORDS. EACH NUMBER IS STORED IN A BIT-FIELD
C***                 "LFLD" BITS WIDE; HENCE, IF "MAXBIT" IS THE LENGTH
C***                 OF A COMPUTER WORD ON THIS SYSTEM, THEN THERE ARE
C***                 (MAXBIT/LFLD) BIT-FIELDS STORED PER WORD. THE AIRCRAFT
C***                 NUMBERS STORED IN THESE BIT-FIELDS INDICATE A UNIQUE
C***                 BIT-POSITION IN THE VARIOUS AIRCRAFT-STATUS BIT-
C***                 VECTORS. THE AIRCRAFT ARE NUMBERED,LEFT-TO-RIGHT,
C***                 0,1,2,...,(MAXAC-1) . TO GET THE ITH AIRCRAFT NUMBER
C***                 IN A WORK-CENTER LIST, THE CORRESPONDING
C***                 BIT-POSITION AND WORD-INDEX MUST BE COMPUTED.
C*****************************************************************
C---
      PARAMETER MAXWC=25,MAXBIT=36,MAXAC=108,MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      COMMON /RSEED/   SEED
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
      COMMON /BITS/ MASK0,MASK(35), MLEFT0,MSKLFT(36),
     &                             IZCOUT,ICOUNT(63)
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
```

```
&                     MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION PBRKSEQ(2,MAXWC), INDXWC(1), IACVC(1)
C---
C---      *IF(THERE ARE ANY BROKEN AIRCRAFT)THEN
           IF(NBRKAC.EQ.0) GO TO  700
C---
C---         *INITIALIZE NUMBER OF SELECTED AIRCRAFT TO NONE
              NSELEC = 0
C---
C---         *DO FOR(EACH WORD OF THE INPUT AIRCRAFT VECTOR)
              DO 500 IWORD = 2,LENGTH
C---
C---            *INITIALIZE DO
                 IACBIT = IACVC(IWORD)
                 IBRKVC = 0
C---
C---            *DO FOR(EACH BIT OF THIS BIT-VECTOR WORD)
                 DO 400  IBIT = 1,MAXBIT
C---
C---               *IF(THIS BIT INDICATES AN ELIGIBLE AIRCRAFT)THEN
                    MASKAC = MASK(IBIT-1)
                    IF(AND(IACBIT,MASKAC) .EQ. 0) GO TO 300
C---
C---                  *SELECT THIS AIRCRAFT AS BROKEN
                       IBRKVC = OR(IBRKVC,MASKAC)
                       NSELEC = NSELEC + 1
C---
C---                  *COMPUTE AIRCAFT #
                       IAC = (IWORD-2)*MAXBIT + (IBIT-1)
C---
C---                  *DO FOR(EACH WORKCENTER)
                       DO  100  J = 1,NWC
C---
C---                     *DRAW RANDOM SAMPLE FROM UNIFORM (0,1)
                          RDRAW = UNIFM1(SEED)
C---
C---                     *CONTINUE LOOP WITH NEXT WORKCENTER IF
C---                       DRAW INDICATES NO BREAK INTO THIS WC
                          IF(RDRAW .GT. PBRKSEQ(2,J)) GO TO 100
C---
C---                     *UPDATE NUMBER/DISTRIBUTION FOR THIS WC
                          JREAL = INDXWC(J)
                          NTOREP = INREPR(JREAL) + 1
C---
C---                     *MAKE 'IRAND' A RANDOM INTEGER BETWEEN 1 AND
C---                        THE NO. OF A/C THAT WILL BE IN THIS W/C
                          IRAND = INT(UNIFM1(SEED)*FLOAT(NTOREP)) + 1
C---
C---                     *MOVE THE A/C CURRENTLY AT SPOT 'IRAND' IN THE
C---                        LIST TO THE RIGHTMOST SPOT.
                          FLD(MOD(NTOREP-1,NPERWRD)*LFLD,LFLD,
&                              LISTRP(1+(NTOREP-1)/NPERWRD,JREAL))
&                           = FLD(MOD(IRAND-1,NPERWRD)*LFLD,LFLD,
```

```
&                           LISTRP(1+(IRAND-1)/NPERWRD,JREAL))
C---
C---                   *INSERT THE SELECTED A/C INTO SPOT 'IRAND'
                        FLD(MOD(IRAND-1,NPERWRD)*LFLD,LFLD,
&                            LISTRP(1+(IRAND-1)/NPERWRD,JREAL))
&                        = IAC
C---
C---                   *INCREMENT THE NO. OF A/C IN THIS W/C
                        INREPR(JREAL) = INREPR(JREAL) + 1
C---
C---                   *EXIT WORKCENTER LOOP IF DRAW ALSO INDICATES
C---                    NO BREAKS INTO REMAINING WORKCENTERS
                        IF(RDRAW .LE. PBRKSEQ(1,J)) GO TO 200
C---
C---                 *END DO (WORKCENTER LOOP)
  100                 CONTINUE
C---
C---              ***** ERROR ***** IF THIS STATEMENT IS REACHED,
C---                                THEN EITHER PBRKSEQ(1,NWC) DOES
C---                                NOT EQUAL 1.0, OR THE
C---                                RANDOM DRAW IS GREATER THAN 1.0
                   WRITE(6,9001)PBRKSEQ(1,NWC),RDRAW
C---
C---                 *EXIT DO (WORKCENTER LOOP) - THIS IS THE
C---                  NORMAL EXIT FROM THE WORKCENTER LOOP
  200                 CONTINUE
C---
C---                 *IF ALL THE BROKEN AIRCRAFT HAVE BEEN SELECTED
C---                  THEN ALL LOOPS ARE TERMINATED
                      IF(NSELEC .GE. NBRKAC) GO TO 600
C---
C---              *END IF (FLYABLE AIRCRAFT TEST)
  300              CONTINUE
C---
C---           *END DO (BIT LOOP)
  400           CONTINUE
C---
C---           *UPDATE INPUT BIT-VECTOR
                IACVC(IWORD)  = XOR(IACVC(IWORD),IBRKVC)
C---
C---        *END DO (WORD LOOP)
  500        CONTINUE
C---
C---              ***** ERROR ***** IF THIS STATEMENT IS REACHED,
C---                                THEN NOT ENOUGH BROKEN AC WERE FOUND
                   WRITE(6,9002)NBRKAC,IACVC(1),NSELEC
C---
C---        *EXIT - ALL BROKEN AC HAVE BEEN SELECTED
  600        CONTINUE
C---
C---        *UPDATE LAST WORD OF INPUT VECTOR
             IACVC(IWORD)  = XOR(IACVC(IWORD),IBRKVC)
C---
```

```
C--         *UPDATE NUMBER OF AIRCRAFT IN INPUT VECTOR
             IACVC(1)  = IACVC(1)  - NBRKAC
C--
C--       *END IF (ZERO BREAKS TEST)
  700       CONTINUE
C--
   RETURN
 9001 FORMAT("0$$$$$$$$ WCDIST ERROR - SEQUENTIAL SAMPLING ERROR",/,
&           " $$$$$$$$     PBRKSEQ(1,NWC), RDRAW = ",2F10.4)
 9002 FORMAT("0$$$$$$$$ WCDIST ERROR - INCONSISTENT BROKEN AIRCRAFT",
&            " $$$$$$$$     NBRKAC, IACVC(1), NSELEC = ",3I5)
   END
```

```
C*****************************************************************
      SUBROUTINE WCPROB(NWC,PBRKWC,PBRKSEQ,INDXWC,PWCPROD)
C*****************************************************************
C++ WCPROB     - INITIALIZE WORK-CENTER SEQUENTIAL BREAK PROBABILITIES.
C***        THIS ROUTINE CALCULATES THE PROBABILITIES NECESSARY TO
C***  SIMULATE THE DISTRIBUTION OF AIRCRAFT BREAKS INTO THE VARIOUS
C***  WORKCENTERS.
C***
C*** INPUTS -
C***      NWC          - NUMBER OF WORKCENTERS BEING MODELED.
C***      PBRKWC(J)    - PROBABILITY THAT AN AIRCRAFT WILL BREAK INTO
C***                     WORKCENTER-J. NOTE THAT THIS BREAK MAY BE DUE
C***                     TO SORTIE BREAKS OR GROUND-ABORTS, DEPENDING
C***                     ON HOW THIS ROUTINE IS CALLED.
C***  OUTPUTS -
C***      PBRKSEQ(1,J)- PROBABILITY THAT AN AIRCRAFT BREAKS INTO THE
C***                     WORKCENTER INDICATED BY 'INDXWC(J)', AND DOES
C***                     NOT BREAK INTO ANY OF THE WORKCENTERS -
C***                       INDXWC(J+1),INDXWC(J+2), ...., INDXWC(NWC)
C***                     GIVEN THAT THE AIRCRAFT HAS BROKEN INTO AT LEAST
C***                     ONE OF THE WORKCENTERS -
C***                       INDXWC(J), INDXWC(J+1), ...., INDXWC(NWC).
C***                     THUS, PBRKSEQ(1,(NWC) MUST EQUAL 1.0 .
C***      PBRKSEQ(2,J)- PROBABILITY THAT AN AIRCRAFT HAS BROKEN INTO THE
C***                     WORKCENTER INDICATED BY 'INDXWC(J)', GIVEN THAT
C***                     THE AIRCRAFT HAS BROKEN INTO AT LEAST ONE OF THE
C***                     WORKCENTERS INDICATED BY -
C***                       INDXWC(J), INDXWC(J+1), ...., INDXWC(NWC).
C***      INDXWC(J)    - INDICATES THE INDEX OF THE WORKCENTER WITH THE
C***                     JTH LARGEST BREAK PROBABILITY. THUS, INDXWC(1)
C***                     INDICATES THE WORKCENTER WITH THE LARGEST
C***                     BREAK PROBABILITY, INDXWC(NWC) INDICATES THE
C***                     ONE WITH THE SMALLEST, ETC. .
C***      PWCPROD      - PRODUCT-FORMULA OVERALL WORK-CENTER BREAK-RATE.
C***                     COMPUTED FROM THE INDIVIDUAL WC BREAK-RATES.
C*****************************************************************
C---
      DIMENSION PBRKWC(NWC), PBRKSEQ(2,NWC), INDXWC(NWC)
C---
C---  *COMPUTE SORTED ARRAY OF WORK-CENTER INDICES ACCORDING TO
C---   LARGEST-TO-SMALLEST BREAK-RATE.
       CALL SORTDS(NWC+0,PBRKWC,INDXWC)
C---
C---  *INITIALIZE END-POINT PROBABILITIES
       PBRKSEQ(1,NWC) = 1.0
       PBRKSEQ(2,NWC) = 1.0
C---
C---    *DO UNTIL(ALL PROBABILITIES HAVE BEEN CALCULATED)
         J = NWC - 1
         POLD = 1.0 - PBRKWC(INDXWC(NWC))
 100     CONTINUE
C---
C---       *COMPUTE NEXT PROBABILITIES
```

```
            PROB  = PBRKWC(INDXWC(J))
            PNEW  = POLD * (1.0 - PROB)
            PBRKSEQ(2,J) = PROB/(1.0 - PNEW)
            PBRKSEQ(1,J) = PBRKSEQ(2,J) * POLD
            POLD   = PNEW
C---
C---     *END DO (WC LOOP)
         J = J - 1
         IF(J.GT.0) GO TO 100
C---
C---  *SAVE PRODUCT-FORMULA OVERALL WORK-CENTER BREAK-RATE
       PWCPROD = 1.0 - POLD
C---
   RETURN
   END
```

```
C*****************************************************************
      SUBROUTINE WCREAD(IFILE,MAXWC,NWC,PBRKWC,NCREWS,SRATE)
C*****************************************************************
C++ WCREAD      - READ AND INITIALIZE WORK CENTER DATA.
C***      WCREAD READS WORK-CENTER DATA FROM THE MAINTENANCE
C*** MANPOWER INPUT FILE. THIS DATA IS ASSUMED TO BE ON UNIT
C*** "IFILE", ONE FREE-FORMAT RECORD PER WORK-CENTER.
C*** THIS ROUTINE RETURNS THE NUMBER OF WORK-CENTERS LOADED,"NWC";
C*** THE BREAK-RATE ARRAY, "PBRKWC"; THE SERVICE-RATE ARRAY, "SRATE";
C*** AND THE SERVERS ARRAY, "NCREWS".
C***       THE SERVERS ARRAY, "NCREWS", REPRESENTS THE NUMBER OF CREWS
C*** AVAILABLE PER 12-HOUR SHIFT.  THE INPUT FILE CONTAINS THE
C*** TOTAL NUMBER OF CREWS AVAILABLE FOR THE PARTICULAR WORK-CENTER.
C*** THE 12-HOUR SHIFT NUMBER IS COMPUTED BY DIVIDING THIS AVAILABLE-
C*** SERVERS IN HALF AND ROUNDING TO THE NEAREST INTEGER NUMBER.
C*** IN ADDITION, IT IS ASSUMED THERE IS ALWAYS AT LEAST ONE CREW
C*** PER SHIFT.
C***      THE AFSC DESCRIBING THE WORK-CENTER, AND THE TOTAL NUMBER OF
C*** SERVERS ARE ECHO-PRINTED, BUT ARE NOT SAVED FOR FUTURE USE.
C***      THE MAXIMUM NUMBER OF WORK-CENTERS WHICH CAN BE LOADED IS
C*** SPECIFIED BY THE "MAXWC" INPUT PARAMETER. IF MORE THAN
C*** THIS NUMBER IS READ, AN ERROR MESSAGE IS PRINTED, AND THE SGM
C*** RUN CONTINUES WITH ONLY THE FIRST "MAXWC" WORK-CENTERS.
C*** INPUTS --
C***       IFILE    - INPUT FILE NUMBER FROM WHICH MAINTENANCE MANPOWER
C***                  INPUT DATA IS TO BE READ
C***       MAXWC    - MAXIMUM NUMBER OF WORK-CENTERS WHICH CAN BE LOADED
C*** OUTPUTS --
C***       NWC      - NUMBER OF WORK-CENTERS LOADED
C***       PBRKWC   - ARRAY OF WORK-CENTER BREAK RATES
C***       NCREWS   - ARRAY OF WORK-CENTER CREW NUMBERS
C***       SRATE    - ARRAY OF WORK-CENTER SERVICE RATES
C*****************************************************************
C---
      DIMENSION PBRKWC(MAXWC), NCREWS(MAXWC), SRATE(MAXWC)
      CHARACTER CAFSC*5
C---
C---  *PRINT HEADER FOR ECHO-CHECK OF INPUT DATA
       WRITE(6,9001)
C---
C---  *INITIALIZE NUMBER OF WORK CENTERS
       NWC=1
C---
C---  *DO UNTIL(NO MORE WORK CENTERS TO LOAD)
  100  CONTINUE
C---
C---     *READ NEXT WORK-CENTER RECORD
          READ(IFILE,9000,END=200)
&                   CAFSC,PBRKWC(NWC),SERVERS,SRATE(NWC)
C---
C---     *PERFORM ERROR CHECK ON INPUT DATA
          IF((PBRKWC(NWC).GT.0.0).AND.(PBRKWC(NWC).LE.1.0))
&                                  GO TO 150
```

```
          IF(SERVERS.GE.0.0)GO TO 150
          IF(SRATE(NWC).GE.0.0)GO TO 150
            WRITE(6,9004)CAFSC,PBRKWC(NWC),SERVERS,SRATE(NWC)
            GO TO 100
  150     CONTINUE
C---
C---     *COMPUTE INTEGER-NUMBER OF CREWS PER 12-HOUR SHIFT
          NCREWS(NWC)=MAX0(1,INT(SERVERS*.5 + .5))
C---
C---     *ECHO-PRINT WORK-CENTER INFORMATION
          WRITE(6,9002)NWC,CAFSC,PBRKWC(NWC),SERVERS,SRATE(NWC)
C---
C---     *INCREMENT WORK-CENTER INDEX
          NWC=NWC+1
C---
C---  *END DO (WORK-CENTER LOOP)
        IF(NWC.LE.MAXWC)GO TO 100
C---     *PRINT ERROR MESSAGE IF STILL MORE DATA ON THE FILE
          READ(IFILE,9000,END=200)PBRKWC(NWC),SERVERS,SRATE(NWC)
          WRITE(6,9003)MAXWC
  200   CONTINUE
C---
C---  *ADJUST NUMBER OF WORK-CENTERS TO ACCOUNT FOR EOF READ
        NWC=NWC-1
C---
C---  *CLOSE-OUT MANPOWER INPUT FILE
        CALL FCLOSE(IFILE)
C---
     RETURN
 9000 FORMAT(1X,A5,1X,F9.4,1X,F9.2,1X,F9.4)
 9001 FORMAT("1",//,7X,"******************************************",/,
&                 ,7X,"**********  AIRCRAFT MAINTENANCE  **********",/,
&                 ,7X,"******************************************",//,
&   21X," BREAK",1X,"  TOTAL",3X,"SERVICE RATE",/,
&   7X,"WC #",2X," AFSC",3X," RATE",2X," SERVERS",2X,"(ACFT/HOUR)",//)
 9002 FORMAT(7X,I3,3X,A5,3X,F6.4,1X,F7.2,3X,F9.4)
 9003 FORMAT("0$$$$$$$$ WCREAD ERROR - TOO MANY WORK-CENTERS ",
&   "IN THE INPUT FILE",/,
&   " $$$$$$$$     ONLY THE FIRST ",I3," WORK-CENTERS WERE USED;",/,
&   " $$$$$$$$     INCREASE -MAXWC- PARAMETER IF YOU WANT MORE WC-S")
 9004 FORMAT("0$$$$$$$$ WCREAD ERROR - INVALID WORK CENTER DATA",/,
&           " $$$$$$$$     PBRKWC, SERVERS, SRATE = ",3F8.3)
     END
```

```
C***************************************************************
   REAL FUNCTION XNORM (XMEAN,STDEV,SEED)
C***************************************************************
C** XNORM      - DRAW RANDOM SAMPLE FROM A NORMAL DISTRIBUTION.
C***   THIS IS A REAL-VALUED FORTRAN FUNCTION WHICH GENERATES
C*** A RANDOM SAMPLE ACCORDING TO A NORMAL PROBABILITY
C*** WITH THE GIVEN INPUT MEAN AND STANDARD DEVIATION.
C*** THE TECHNIQUE IS TO APPROXIMATE A NORMAL DISTRIBUTION USING
C*** THE CENTRAL LIMIT THEOREM. 12 INDEPENDENT SAMPLES ARE
C*** DRAWN FROM A UNIFORM(0,1) DISTRIBUTION AND THEN ADDED.
C*** THE RESULT IS APPROXIMATELY NORMALLY DISTRIBUTED WITH MEAN 6
C*** AND STANDARD DEVIATION 1. THE SAMPLE IS THEN TRANSLATED TO
C*** OBTAIN A SAMPLE FROM A DISTRIBUTION WITH THE GIVEN
C*** INPUT MEAN AND STANDARD DEVIATION.
C***
C*** INPUTS --
C***   XMEAN    - MEAN OF THE NORMAL DISTRIBUTION FROM WHICH
C***              THE SAMPLE IS TO BE GENERATED.
C***   STDEV    - STANDARD DEVIATION OF NORMAL DISTRIBUTION FROM
C***              WHICH SAMPLE IS TO BE GENERATED. IF THIS VALUE
C***              IS NEGATIVE, AN ERROR MESSAGE IS PRINTED.
C*** INPUT/OUTPUT --
C***   SEED     - CURRENT SEED OF RANDOM NUMBER GENERATOR.
C*** OUTPUT --
C***   XNORM    - RANDOM SAMPLE FROM A NORMAL DISTRIBUTION WITH
C***              GIVEN MEAN AND STANDARD DEVIATION.
C***************************************************************
C---
C---  *IF(STANDARD DEVIATION IS LEGITIMATE)THEN
      IF(STDEV.LT.0.0)GO TO 100
C---
C---     *DRAW SAMPLES FROM 12 UNIFORM (0,1) DISTRIBUTIONS AND
C---            ADJUST THE MEAN OF THIS RANDOM SAMPLE TO ZERO
         XNORM=UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)
&            +UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)
&            +UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)-6.
C---
C---     *CONVERT TO A SAMPLE FROM DISTRIBUTION WITH APPROPRIATE
C---                        MEAN AND STANDARD DEVIATION.
         XNORM = XMEAN + STDEV*XNORM
C---
C---  *ELSE (NEGATIVE STANDARD DEVIATION)
      GO TO 200
  100 CONTINUE
C---
C---     *SET RETURN VALUE TO ZERO AND PRINT ERROR MESSAGE
         XNORM = 0.0
         WRITE(6,9001)STDEV
C---
C---  *END IF (STD DEV TEST)
  200 CONTINUE
C---
   RETURN
```

```
 9001 FORMAT("0$$$$$$$$$ XNORM ERROR - NEGATIVE STANDARD DEVIATION",/,
&           " $$$$$$$$$     STDEV = ",F10.5)
      END
```

```
C*****************************************************************
   SUBROUTINE ZBITSL(NONES,IARRAY)
C*****************************************************************
C** ZBITSL     - ZERO-OUT 1-BITS IN LEFTMOST PORTION OF A WORD.
C***      ZBITSL IS A FORTRAN SUBROUTINE WHICH WILL ZERO-OUT
C*** A SPECIFIED NUMBER OF 1-BITS IN THE LEFTMOST PORTION OF
C*** A BIT-VECTOR. THIS BIT-VECTOR IS KEPT IN AN ARRAY,
C*** ORGANIZED IN THE FOLLOWING FASHION - THE FIRST WORD OF
C*** THE ARRAY CONTAINS THE CURRENT NUMBER OF 1-BITS IN THE
C*** BIT-VECTOR, AND THE REMAINING WORDS OF THE ARRAY CONTAIN
C*** THE ACTUAL BIT-VECTOR. THIS ROUTINE ZEROES OUT THE PROPER
C*** 1-BITS, AND UPDATES THE 1-BIT COUNTER IN THE FIRST WORD
C*** OF THE ARRAY.
C***
C*** INPUT -
C***   NONES      - NUMBER OF 1-BITS TO BE ZEROED-OUT. NOTE THAT
C***                NONES MUST BE .LE. NUMBER OF 1S IN THE BIT-
C***                VECTOR.
C*** INPUT/OUTPUT -
C***   IARRAY     - ARRAY CONTAINING THE BIT-VECTOR TO BE
C***                MODIFIED. IARRAY(1) IS A COUNTER WHICH INDICATES
C***                THE CURRENT NUMBER OF 1-BITS IN THE BIT-VECTOR.
C***                THE ACTUAL BIT-VECTOR IS THE CONSECUTIVE BITS
C***                CONTAINED IN THE WORDS IARRAY(2)-IARRAY(LENGTH)
C*** COMMON INPUT -
C***   LENGTH     - LENGTH (IN COMPUTER WORDS) OF THE ARRAYS
C***                CONTAINING THE VARIOUS BIT-VECTORS; IFLYVC, ETC
C*****************************************************************
C---
      PARAMETER MAXAC=108, MAXBIT=36, MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
&                   MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION IARRAY(1)
C---
C---       *INITIALIZE DO
            NLEFT  = NONES
            INDEX  = 2
C---
C---       *DO WHILE(ALL APPROPRIATE WORDS HAVE NOT BEEN MODIFIED)
 1000       CONTINUE
            IF(NLEFT.LE.0)      GO TO 4000
            IF(INDEX.GT.LENGTH) GO TO 4000
C---
C---          *COUNT NUMBER OF 1S IN NEXT WORD
               NXT1S  = N1BITS( IARRAY(INDEX) )
C---
C---          *IF(NOT ALL 1-BITS IN THIS WORD SHOULD BE ZEROED)
               IF(NXT1S.LE.NLEFT) GO TO 2000
C---
C---             *ZERO-OUT THE APPROPRIATE NUMBER OF 1S
                  IARRAY(INDEX) = XOR( IARRAY(INDEX) ,
&                                   LBITS(IARRAY(INDEX),NLEFT) )
C---
```

```
C--          *ELSE (ALL 1-BITS IN THIS WORD ARE TO BE ZEROED)
              GO TO 3000
 2000         CONTINUE
C--
C--             *ZERO-OUT THE ENTIRE WORD
                 IARRAY(INDEX) = 0
C--
C--          *END IF (ALL 1S TEST)
 3000         CONTINUE
C--
C--          *UPDATE NUMBER OF 1S LEFT TO ZERO-OUT
              NLEFT = NLEFT - NXT1S
C--
C--          *INCREMENT INDEX FOR NEXT WORD OF BIT-VECTOR
              INDEX = INDEX + 1
C--
C--       *END DO (WORD LOOP)
           GO TO 1000
 4000      CONTINUE
C--
C--       *UPDATE 1S COUNTER FOR THIS BIT VECTOR
           IARRAY(1) = IARRAY(1) - NONES
C--
C--       *PERFORM ERROR CHECK TO ENSURE APPROPRIATE NUMBER
C--        OF 1-BITS WAS ZEROED-OUT
           IF(NLEFT.GT.0) WRITE(6,9001)NONES,IARRAY(1),NLEFT
C--
   RETURN
 9001 FORMAT("0$$$$$$$$ ZBITSL ERROR - NOT ENOUGH 1S TO ZERO",/,
&   " $$$$$$$$     NONES,IARRAY(1),NLEFT = ",3I5)
   END
```

*

SCENARIO INPUT PROGRAM

```
C OS29/N232D/SGM/ZDATA
C THIS PROGRAM PROVIDES AN INTERACTIVE INTERFACE TO THE SORTIE
C GENERATION MODEL, BY ALLOWING THE USER TO MODIFY THE INPUT
C ITEMS IN THE NAMELIST ZDATA.
      PARAMETER MAXA=14,MAXCYC=10,MAXAVARY=4,MAXCVARY=1,
    &  MAXVARY=MAXAVARY+MAXCVARY,MAXPAR=MAXVARY+MAXA,MAXDAY=30
      CHARACTER*25 RESPONCE
      CHARACTER*2 DAY
      CHARACTER*20 ITEM
      CHARACTER*6 A(MAXPAR),INFMAN,NONORS,NSIM,SEED,UE,MAXFLY,RES,
     & ATTRIT,ANYBRK,ANYGA,RNMCM,NUMDAY,NCYCLE,FTOTYM,LTOTYM,
     & PREFLT,SRTLTH,SCALE,IAUGMT,AVARY(MAXDAY,MAXAVARY),
     &  CVARY(MAXDAY,MAXCYC,MAXCVARY)
      LOGICAL VARY(MAXVARY)
      COMMON /EDATA/ DAY,RESPONCE
      COMMON /VDATA/ ITEM
      COMMON /VARYSW/ VARY
      COMMON /GLOBAL/ICYC,IDAYS
      COMMON/INZDATA/INFMAN,NONORS,NSIM,SEED,UE,MAXFLY,RES,ATTRIT,
     &   ANYBRK,ANYGA,RNMCM,NUMDAY,NCYCLE,FTOTYM,LTOTYM,
     &   PREFLT,SRTLTH,SCALE,IAUGMT
      NAMELIST/ZDATA/INFMAN,NONORS,NSIM,UE,MAXFLY,RES,ATTRIT,
     &   ANYBRK,ANYGA,RNMCM,NUMDAY,NCYCLE,FTOTYM,LTOTYM,PREFLT,
     &   SRTLTH,SCALE,IAUGMT
      EQUIVALENCE (A(1),INFMAN)
      DIMENSION LIST(MAXPAR),LISTOUT(MAXA),LISTVARY(MAXVARY)
C LIST IS USED AS AN INDEX IN THE ARRAYS WHICH CORRESPOND TO
C THE ZDATA ITEMS. IT DETERMINES  THE ORDER IN WHICH THE
C PARAMETERS WILL BE LISTED. IT IS DEFINED AS FOLLOWS;
C     LIST(I) = # WHICH INDICATES WHAT PARAMETER IS TO BE
C               ITH IN THE DATA-CODE ORDER.
C THE NUMBERS ASSOCIATED WITH THE PARAMETERS ARE AS FOLLOWS;
C    1 : INFMAN , 2 : NONORS , 3 : NSIM , 4 : SEED
C    5 : UE , 6 : MAXFLY , 7 : RES , 8 : ATTRIT
C    9 : ANYBRK , 10 : ANYGA , 11 : RNMCM , 12 : NUMDAY
C    13 : NCYCLE , 14 : FTOTYM , 15 : LTOTYM , 16 : PREFLT
C    17 : SRTLTH , 18 : SCALE , 19 : IAUGMT
      DATA VARY/.F.,.F.,.F.,.F.,.F./
      DATA LISTOUT/4,14,15,1,2,3,5,9,11,12,16,17,18,19/
      DATA LISTVARY/1,2,3,4,5/
      DATA LIST/3,4,5,9,11,12,14,15,17,16,1,2,19,18,8,10,13,7,6/
      READ(08,ZDATA)
      CALL FCLOSE (8)
      ENCODE(DAY,1) MAXDAY
 1    FORMAT(I2)
      CALL SPRAY(ATTRIT,AVARY(1,1),MAXDAY)
      CALL SPRAY(ANYGA,AVARY(1,2),MAXDAY)
      CALL SPRAY(NCYCLE,AVARY(1,3),MAXDAY)
      CALL SPRAY(MAXFLY,CVARY(1,1,1),MAXDAY*MAXCYC)
      CALL SPRAY('0     ',AVARY(1,4),MAXDAY)
      CALL SPRAY(RES,AVARY(1,4),1)
      DECODE(NCYCLE,5) ICYC
      DECODE(NUMDAY,5) IDAYS
```

PRECEDING PAGE BLANK-NOT FILMED

```
      CALL FPARAM(1,80)
      NUMVARY=0
      PRINT ,' ENTER RANDOM NUMBER SEED'
      READ ,A(4)
      PRINT ,' '
      PRINT ,' CODE - FUNCTION'
      PRINT ,'    1   SET PARAMETERS FOR SGM RUN'
      PRINT ,'    2   LIST CURRENT SCENARIO'
      PRINT ,'    3   CHANGE SCENARIO'
 100  CONTINUE
      PRINT ,' '
      PRINT ,' ENTER FUNCTION CODE (1-SET/2-LIST/3-CHANGE)'
      CALL ANYERR(IERR)
      CALL FXOPT(32,1,1,0)
 200  CONTINUE
      IERR=0
      READ ,ICODE
      IF(IERR.EQ.32) GOTO 250
      IF ((ICODE .GE. 1) .AND. (ICODE .LE. 3)) GO TO 300
 250  CONTINUE
        PRINT ,' FUNCTION CODE MUST BE IN RANGE 1-3, PLEASE REENTER'
        GO TO 200
 300  CONTINUE
      CALL FXOPT(32,0,0,0)
      GO TO (600,500,400) ,ICODE
 400  CONTINUE
      CALL CHANGE(CVARY,AVARY,LISTVARY,NUMVARY,A,LIST)
      GO TO 100
 500  CONTINUE
      CALL LISTA(CVARY,AVARY,LISTVARY,NUMVARY,A,LIST)
      GO TO 100
 600  CONTINUE
      PRINT ,' '
      PRINT ,' TO BEGIN SIMULATION USE EITHER,'
      PRINT ,' '
      PRINT ,'    RUNC OS29/N232D/SGM/RSGMTSS - FOR TIME SHARING RUN'
      PRINT ,' OR,'
      PRINT ,'    RUN OS29/N232D/SGM/RSGMBTCH - FOR BATCH RUN.'
      IF ((A(1) .EQ. 'Y') .OR. (A(1) .EQ. 'YES')) GO TO 700
         A(1)='F'
         GO TO 800
 700  CONTINUE
         A(1)='T'
 800  CONTINUE
      IF ((A(2) .EQ. 'Y') .OR. (A(2) .EQ. 'YES')) GO TO 900
         A(2)='F'
         GO TO 1000
 900  CONTINUE
        A(2)='T'
 1000 CONTINUE
      IF ((A(19).EQ.'Y').OR.(A(19).EQ.'YES')) GOTO 1050
         A(19)='0      '
         GOTO 1075
```

```
1050 CONTINUE
        A(19)='1     '
1075 CONTINUE
     CALL FMEDIA(01,0)
     WRITE(01,5) (VARY(I),I=1,MAXVARY)
     WRITE(01,5) ((A(LISTOUT(I)),' '),I=1,MAXA-2)
     WRITE(01,5) A(LISTOUT(MAXA-1)),A(LISTOUT(MAXA))
5    FORMAT(V)
     DECODE(FTOTYM,5) IFTOTYM
     DECODE(LTOTYM,5) ILTOTYM
     DECODE(PREFLT,5) XPREFLT
     DECODE(SRTLTH,5) XSRTLTH
     DO 1100 J=1,IDAYS
        DECODE(AVARY(J,3),5) INCYCLE
        CALL SETTIME(IFTOTYM,ILTOTYM,XPREFLT,XSRTLTH,WAITCYC,TYMNITE,
    &    INCYCLE)
        WRITE(01,5) (AVARY(J,I),I=1,MAXAVARY),WAITCYC,TYMNITE
        WRITE(01,5) ((CVARY(J,K,I),K=1,INCYCLE),I=1,MAXCVARY)
1100 CONTINUE
     STOP
     END
```

```
C*************************************************************
C*****   LISTA IS A FORTRAN SUBROUTINE WHICH LISTS THE CURRENT
C***** VALUES OF THE PARAMETERS ALONG WITH THEIR CODES.
C*************************************************************
      SUBROUTINE LISTA(CVARY,AVARY,LISTVARY,NUMVARY,A,LIST)
      PARAMETER MAXA=14,MAXAVARY=4,MAXCVARY=1,MAXDAY=30,
     &   MAXCYC=10,MAXVARY=MAXAVARY+MAXCVARY,MAXPAR=MAXA+MAXVARY
      CHARACTER*6 A(MAXPAR),AVARY(MAXDAY,MAXAVARY),
     & CVARY(MAXDAY,MAXCYC,MAXCVARY),NUMDAY
      CHARACTER*33 CNAME(MAXPAR)
      INTEGER LIST(MAXPAR),LISTVARY(MAXVARY)
      COMMON /GLOBAL/ ICYC,IDAYS
      DATA CNAME/'   INFINITE MANPOWER (YES/NO)    '
     &      ,'   INFINITE SPARE PARTS (YES/NO)'
     &      ,'   # SIMULATIONS                 '
     &      ,'   RANDOM NUMBER SEED            '
     &      ,'   UE                            '
     &      ,'   MAXIMUM LAUNCH-SIZE    (C/D) '
     &      ,'   RESERVE AIRCRAFT        (D)  '
     &      ,'   ATTRITION RATE          (D)  '
     &      ,'   AIRCRAFT BREAK RATE           '
     &      ,'   GROUND ABORT RATE       (D)  '
     &      ,'   INITIAL NMCM RATE             '
     &      ,'   # DAYS                        '
     &      ,'   # MASS LAUNCHES PER DAY (D)  '
     &      ,'   FIRST TAKEOFF TIME            '
     &      ,'   LAST TAKEOFF TIME             '
     &      ,'   MINIMAL RECOVERY TIME (HRS)  '
     &      ,'   SORTIE LENGTH (HRS)           '
     &      ,'   MAX SORTIES/DAY FOR PLOT(OR 0) '
     &      ,'   AUGMENT RESERVE AC (YES/NO)  '/
      PRINT ,' '
      PRINT ,'THE CURRENT VALUES OF THE SCENARIO INPUTS ARE :'
      PRINT ,' '
      PRINT ,' INPUT          SCENARIO ITEM          CURRENT'
      PRINT ,' CODE                                   VALUE'
      PRINT ,' '
      PRINT10, ((I,CNAME(LIST(I)),A(LIST(I))),I=1,MAXA)
 10   FORMAT(3X,I2,A33,' = ',A7)
      NOTVARY=MAXVARY-NUMVARY
      PRINT ,' '
      PRINT ,'THE FOLLOWING ITEMS MAY VARY BY DAY(D) OR CYCLE/DAY(C/D)'
      PRINT ,' '
     PRINT10,(MAXA+LISTVARY(I),CNAME(LIST(MAXA+LISTVARY(I))),
     & A(LIST(MAXA+LISTVARY(I)))),I=1,NOTVARY)
     IF(NUMVARY.LE.0) GOTO 99
     ENTRY LISTONE(CVARY,AVARY,LIST,LISTVARY,NOTVARY)
     ITIMES=(IDAYS+9)/10
     JMIN=1
     DO 900 I=1,ITIMES
         JMAX=MIN0(JMIN+9,IDAYS)
         PRINT20, (J,J=JMIN,JMAX)
 20      FORMAT('0','DAY ',I2,2X,9I7)
```

```
      DO 700 K=NOTVARY+1,MAXVARY
          GOTO (400,400,400,400,500,500), LISTVARY(K)
400       CONTINUE
         PRINT30,MAXA+LISTVARY(K),CNAME(LIST(LISTVARY(K)+MAXA)),
   &     (AVARY(J,LISTVARY(K)),J=JMIN,JMAX)
30       FORMAT('0',2X,I2,A33/6X,10A7)
       GOTO 700
500    CONTINUE
      PRINT40,MAXA+LISTVARY(K),CNAME(LIST(LISTVARY(K)+MAXA))
      DO 600 L=1,ICYC
        PRINT50,L,((CVARY(J,L,LISTVARY(K)-MAXAVARY)),
   &      J=JMIN,JMAX)
40    FORMAT('0',2X,I2,A33/'CYCLE')
50    FORMAT(2X,I2,2X,10(1X,A6))
600   CONTINUE
700   CONTINUE
     JMIN=JMIN+10
900  CONTINUE
99   CONTINUE
     RETURN
     END
```

```
C***************************************************************
C*****   CHANGE IS A FORTRAN SUBROUTINE WHICH IS USED TO CHANGE
C***** THE VALUES OF THE PARAMETERS. THE USER MAY CONTINUE
C***** ENTERING PARAMETER CODES AND THEIR VALUES AS LONG AS IS
C***** DESIRED, EXITING FROM THE ROUTINE WHEN A PARAMETER CODE OF
C***** 0 IS ENTERED.
C***************************************************************
      SUBROUTINE CHANGE(CVARY,AVARY,LISTVARY,NUMVARY,A,LIST)
      PARAMETER MAXCYC=10,MAXDAY=30,MAXA=14,MAXAVARY=4,
    &  MAXCVARY=1,MAXVARY=MAXAVARY+MAXCVARY,MAXPAR=MAXA+MAXVARY
      CHARACTER*20 ITEM
      CHARACTER*6 CVARY(MAXDAY,MAXCYC,MAXCVARY),A(MAXPAR),
     & AVARY(MAXDAY,MAXAVARY),NUMDAY,VALUE
      LOGICAL VSWITCH,VARY(MAXVARY)
      DIMENSION ONEVARY(MAXVARY),LIST(MAXPAR),LISTVARY(MAXVARY)
      EXTERNAL VALUERR
      COMMON /VDATA/ ITEM
      COMMON /VARYSW/ VARY
      COMMON /GLOBAL/ ICYC,IDAYS
      PRINT ,' '
      PRINT ,' IF CHANGES ARE DESIRED ENTER THE INPUT CODE OF THE ITEM'
      PRINT ,' TO BE ALTERED, ELSE ENTER 0'
 100  CONTINUE
      CALL FXOPT(32,1,1,1)
      CALL ANYERR(IERR)
      CALL FXALT(VALUERR)
     IERR=0
      VSWITCH=.F.
      READ ,ITEM
5     FORMAT(V)
      DECODE(ITEM,5) ICODE
      CALL FXOPT(32,0,0,0)
      IF(IERR.NE.32) GOTO 200
         VSWITCH=.T.
         DECODE(ITEM,5) ICODE,VALUE
200    CONTINUE
      IF ((ICODE .GE. 0) .AND. (ICODE .LE. MAXPAR)) GO TO 300
          PRINT10,' INPUT CODE MUST BE IN RA    . ',MAXPAR,
    &          ', PLEASE REENTER'
10        FORMAT(A31,I2,A16)
          GO TO 100
300  CONTINUE
      IF (ICODE .EQ. 0) GO TO 99
      IF (ICODE.LE.MAXA) GOTO 700
        IF ((ICODE-MAXA).LE.MAXAVARY) GOTO 500
           ICVCODE=ICODE-MAXA-MAXAVARY
           IF (.NOT.VSWITCH) GOTO 400
               A(LIST(ICODE))=VALUE
              CALL CHECK(ICODE,A(LIST(ICODE)),LIST(ICODE))
              CALL SPRAY(A(LIST(ICODE)),CVARY(1,1,ICVCODE),
        &   MAXDAY*MAXCYC)
              GOTO 850
400        CONTINUE
```

```
        CALL READVARY(CVARY,AVARY,LIST,ICODE-MAXA,NUMVARY,LISTVARY,
   &       A(LIST(ICODE)))
        GOTO 850
500  CONTINUE
     IF (.NOT.VSWITCH) GOTO 600
       A(LIST(ICODE))=VALUE
       CALL CHECK(ICODE,A(LIST(ICODE)),LIST(ICODE))
       CALL SPRAY(A(LIST(ICODE)),AVARY(1,ICODE-MAXA),MAXDAY)
       IF (ICODE-MAXA.EQ.4)
   &       CALL SPRAY('0      ',AVARY(2,4),MAXDAY-1)
       GOTO 850
600  CONTINUE
     CALL READVARY(CVARY,AVARY,LIST,ICODE-MAXA,NUMVARY,LISTVARY,
   &     A(LIST(ICODE)))
     GOTO 850
700  CONTINUE
     IF (VSWITCH) GOTO 800
       PRINT ,' '
       PRINT ,' ENTER NEW VALUE OF ITEM, OLD VALUE=',A(LIST(ICODE))
       READ ,VALUE
800  CONTINUE
     A(LIST(ICODE))=VALUE
     CALL CHECK(ICODE,A(LIST(ICODE)),LIST(ICODE))
850  CONTINUE
     PRINT ,' '
     PRINT ,' ENTER NEXT INPUT CODE OR 0 IF FINISHED'
     GOTO (100,100,100,100,100,100,100,100,100,100,
  &  100,900,1000,100,100,100,100,100,100) ,LIST(ICODE)
900  CONTINUE
     DECODE(A(12),5) IDAYS
     GOTO 100
1000 CONTINUE
     DECODE(A(13),5) I
     IF (I.GT.ICYC) ICYC=I
     GO TO 100
99    CONTINUE
     RETURN
     END
```

```
C***************************************************************
C*****  CHECK IS A FORTRAN SUBROUTINE WHICH INSURES THAT THE
C***** NEW VALUE ENTERED FOR A PARAMETER IS LEGITAMATE.
C***************************************************************
      SUBROUTINE CHECK(ICODE,ITEM,LICODE)
      PARAMETER MAXDAY=30,MAXCYC=10,MAXAC=108
      PARAMETER MAXPAR=19
      CHARACTER*20 ITEM
      CHARACTER*25 MESSAGE(MAXPAR)
      DIMENSION MAX(MAXPAR)
      DATA MESSAGE/'INFINITE MANPOWER','INFINITE SPARES',' ',' ',
     &     ' UE > MAXAC',' MAXFLY > MAXAC',' ',' ATTRIT > 1',
     &     ' BREAK RATE > 1',' GABORT RATE > 1',
     &     ' INITIAL READINESS > 1',' # DAYS > MAXDAY',
     &     ' # LAUNCHES > MAXCYC',
     &     ' FIRST T-0 TIME > 2400',' LAST T-0 TIME > 2400',
     &     ' SORTIE LENGTH > 24.0',' PREFLIGHT TIME > 24.0',' ',
     &     ' AUGMENT RESERVE '/
      DATA MAX/0,0,0,0,MAXAC,MAXAC,0,1,1,1,1,MAXDAY,MAXCYC,2400,2400,
     &   24,24,0,0/
      GO TO (100,100,99,99,300,300,99,300,300,300,300,300
     & ,300,300,300,300,300,99,100) ,LICODE
 100  CONTINUE
      CALL YORN(MESSAGE(LICODE),ITEM,ICODE)
      GO TO 99
 300  CONTINUE
      CALL ILTJ(MESSAGE(LICODE),ITEM,ICODE,MAX(LICODE))
 99   CONTINUE
      RETURN
      END
```

```
C*************************************************************
C*****    ILTJ IS A FORTRAN SUBROUTINE WHICH ENSURES THAT A
C***** PARAMETER VALUE IS WITHIN LEGAL BOUNDS, 0 => VALUE => MAX,
C***** BY PROMPTING THE USER FOR A NEW VALUE IF IT IS OUT OF BOUNDS.
C*************************************************************
      SUBROUTINE ILTJ(MESSAGE,ITEM,ICODE,MAX)
      CHARACTER*20 ITEM
      CHARACTER*25 MESSAGE
      REAL NUM
      CALL ANYERR(IERR)
      CALL FXOPT(32,1,1,0)
 100  CONTINUE
      IERR=0
      DECODE(ITEM,20) NUM
      IF (IERR.NE. 32) GOTO 150
         PRINT ,' INPUT ITEM MUST BE NUMERIC'
         GOTO 300
 150  CONTINUE
 20   FORMAT(V)
      IF (NUM .GE. 0.0) GO TO 200
         PRINT ,' INPUT ITEM ',ICODE,' MUST BE > 0'
         GO TO 300
 200  CONTINUE
      IF (NUM .LE. MAX) GO TO 99
        PRINT ,MESSAGE
        PRINT ,ITEM,'>',MAX
 300    CONTINUE
        PRINT ,' PLEASE REENTER INPUT ITEM ',ICODE
        READ ,ITEM
        PRINT ,' '
       GO TO 100
 99   CONTINUE
      CALL FXOPT(32,0,0,0)
      RETURN
      END
```

```
C*************************************************************
C***** YORN IS A FORTRAN SUBROUTINE WHICH ENSURES THAT ITEM IS
C***** EITHER YES(/Y) OR NO(/N), BY PROMPTING THE USER FOR A NEW
C***** VALUE IF IT IS INCORRECT, UNTIL IT IS.
C*************************************************************
      SUBROUTINE YORN(MESSAGE,ITEM,ICODE)
      CHARACTER*6 ITEM
      CHARACTER*25 MESSAGE
 10   CONTINUE
      IF ((ITEM .EQ. 'Y') .OR. (ITEM .EQ. 'N') .OR.
     &    (ITEM .EQ. 'YES') .OR. (ITEM .EQ. 'NO')) GO TO 99
        PRINT ,MESSAGE,'  MUST BE YES OR NO, PLEASE REENTER'
        READ ,ITEM
        GO TO 10
 99  CONTINUE
     RETURN
     END
```

```
C****************************************************************
C*****     SEGMENT (SETTIME - SET WAIT CYCLE & OVERNIGHT TIMES)
      SUBROUTINE SETTIME(IFTOTYM,ILTOTYM,XPREFLT,XSRTLTH,WAITCYC,
     &  TYMNITE,INCYCLE)
C*****   SETTIME IS A FORTRAN ROUTINE USED TO CALCULATE THE
C***** WAIT CYCLE AND OVERNIGHT TIMES FOR SORTIES GIVEN THE
C***** INITIAL AND LAST TAKEOFF TIMES, THE PREFLIGHT TIME, THE
C***** LENGTH OF A SORTIE , AND THE NUMBER OF CYCLES PER DAY.
C****************************************************************
        DECMIN = ABS(DECHR(ILTOTYM)-DECHR(IFTOTYM))
        WAITCYC = DECMIN/FLOAT(INCYCLE-1)-(XPREFLT+XSRTLTH)
        IF (WAITCYC.LT.0.0) STOP 'WAITCYC < 0'
        TYMNITE = 24.0 - (XPREFLT+XSRTLTH+DECMIN)
        RETURN
        END
```

```
C**********************************************************
C*****     FUNCTION (DECHR - DECIMAL HOUR EQUIV. OF MILITARY TIME)
C*****    DECHR IS A FORTRAN REAL FUNCTION WHICH RETURNS THE DECIMAL
C***** HOUR EQUIVALENT OF AN HOUR SPECIFIED IN MILITARY TERMS.
C**********************************************************
      REAL FUNCTION DECHR(MILTIME)
      REAL XMIN
      INTEGER IHR
      IHR = INT(MILTIME/100)
      XMIN = ((FLOAT(MILTIME)/100.0) - FLOAT(IHR))/.6
      DECHR = FLOAT(IHR) + XMIN
      RETURN
      END
```

```
C*************************************************************
C*****   READVARY IS A FORTRAN SUBROUTINE WHICH IS USED TO READ
C***** THE VALUES OF A PARAMETER WHICH MAY VARY BY DAY OR
C***** CYCLE/DAY. THESE NEED TO BE HANDLED SEPARATELY WHETHER THE
C***** USER WANTS TO VARY THEIR VALUES OR NOT AS AN ARRAY IS
C***** USED AND MUST BE FILLED.
C*************************************************************
      SUBROUTINE READVARY(CVARY,AVARY,LIST,ICODE,NUMVARY,LISTVARY,ITEM)
     PARAMETER MAXCYC=10,MAXDAY=30,MAXA=14,MAXAVARY=4,MAXCVARY=1,
    & MAXVARY=MAXAVARY+MAXCVARY,MAXPAR=MAXA+MAXVARY
      LOGICAL VARY(MAXVARY),EXIT,FORM
      CHARACTER*6 CVARY(MAXDAY,MAXCYC,MAXCVARY),AVARY(MAXDAY,MAXAVARY)
     &   ,XVALUE(MAXCYC),ANSWER,ITEM
      CHARACTER*25 RESPONCE
      CHARACTER*64 TEXT(30)
      REAL X(MAXCYC)
      INTEGER ONEVARY(MAXVARY),LIST(MAXPAR),LISTVARY(MAXVARY)
     &   ,ILIST(MAXVARY)
      EXTERNAL MULTIERR,DAYSERR
      COMMON /VARYSW/ VARY
      COMMON /GLOBAL/ ICYC,IDAYS
      COMMON /EDATA/ DAY,RESPONCE
      DATA ONEVARY/0,0,0,0,0/
      DATA ILIST/1,2,3,4,5/
      DATA ONEVARY/0,0,0,0,0/
      DATA TEXT/'           ***  SYNTAX RULES  ***',' ',
     & 'VALUES MAY BE ENTERED IN THE FOLLOWING WAYS;',
     & ' ','1) STARTING AT DAY 1, ONE DAY AT A TIME.',
     & '  EX. =.01','      =.02','     ...   NUMBER OF DAY TIMES',
     & '      =.02',' ','2) ONE VALUE FOR MULTIPLE DAYS.',
     & '  EX. =.01*5  WILL ENTER .01 FOR THE NEXT 5 DAYS.',
     & '  EX. =.01** WILL ENTER .01 FOR THE REST OF THE DAYS.',
     & ' ','3) ONE VALUE FOR SPECIFIC DAYS.',
     & '  EX. =.01(5-20) WILL ENTER THE VALUE .01 FOR DAYS 5 THRU 20.',
     & ' ','IF THE ITEM VARIES BY CYCLE PER DAY IT MAY BE ENTERED',
     & 'AS ABOVE, OR BY WAVE WITH THE NUMBER OF CYCLES SPECIFIED.',
     & '  EX. =5#12;12;12;24;0 WILL ENTER THE VALUE 12 FOR WAVES',
     & '      1 THRU 3, 24 FOR WAVE 4, AND 0 FOR WAVE 5, FOR THE ',
     & '      CURRENT DAY.',
     & '  EX. =5#12;12;12;24;0(5-20) SAME AS ABOVE EXCEPT VALUES',
     & '      ARE ENTERED FOR DAYS 5 THRU 20.',' ',
     & 'OPTIONS:','=SYN - PRINT THE SYNTAX RULES.',
     & '=LIST - LIST THE CURRENT VALUES OF THE ITEM BEING ALTERED.',
     & '=DAY - PRINT THE CURRENT DAY A VALUE IS BEING ENTERED FOR.',
     & '=STOP - ALLOWS USER TO STOP ENTERING VALUES.'/
      ISTART=1
      ILIM=1
      PRINT10,' DO YOU WANT PARAMETER ',MAXA+ICODE,
     & ' TO VARY BY DAY ?'
 10   FORMAT(' '/' ',A23,I2,A18)
      READ ,ANSWER
      CALL YORN('            YOUR RESPONCE',ANSWER)
       IF ((ANSWER.EQ.'Y').OR.(ANSWER.EQ.'YES')) GOTO 100
```

```
        PRINT ,' '
        PRINT ,' ENTER NEW VALUE OF ITEM, OLD VALUE=',ITEM
        READ ,RESPONCE
        LENGTH=MAXDAY
        VARY(ICODE)=.F.
         IF (ICODE.NE.4) GOTO 75
              LENGTH=1
              EXIT=.T.
              CALL SPRAY('0       ',AVARY(2,4),MAXDAY-1)
75       CONTINUE
        GOTO 700
100  CONTINUE
     IF (VARY(ICODE)) GOTO 150
       VARY(ICODE)=.T.
       NEXT=MAXVARY-NUMVARY
       TEMP=LISTVARY(NEXT)
       LISTVARY(NEXT)=ICODE
       LISTVARY(ILIST(ICODE))=TEMP
       TEMP=ILIST(ICODE)
       ILIST(ICODE)=NEXT
       ILIST(NEXT)=TEMP
       NUMVARY=NUMVARY+1
150  CONTINUE
      CALL ANYERR(IERR)
      CALL FXOPT(32,1,1,1)
      CALL FXOPT(67,1,1,1)
      EXIT=.F.
5     FORMAT(V)
      PRINT ,' '
      PRINT ,' WOULD YOU LIKE THE SYNTAX RULES EXPLAINED ?'
      READ ,ANSWER
      CALL YORN('            YOUR RESPONCE',ANSWER)
      IF ((ANSWER.EQ.'Y').OR.(ANSWER.EQ.'YES')) PRINT15,
     & (TEXT(I),I=1,30)
15    FORMAT(' ',A63)
      PRINT ,' '
      PRINT20,' ENTER VALUES FOR ITEM ',ICODE+MAXA
20    FORMAT(A25,I3)
200   CONTINUE
      READ ,RESPONCE
      IF (RESPONCE.EQ.'STOP') GOTO 900
      IF (RESPONCE.NE.'SYN') GOTO 300
        PRINT ,TEXT
        GOTO 200
300   CONTINUE
      IF (RESPONCE.NE.'LIST') GOTO 400
         ONEVARY(MAXVARY)=ICODE
         CALL LISTONE(CVARY,AVARY,LIST,ONEVARY,MAXVARY-1)
         GOTO 200
400   CONTINUE
      IF (RESPONCE.NE.'DAY') GOTO 450
         PRINT30,ISTART
30       FORMAT(' DAY = ',I2)
```

```
      GOTO 200
450  CONTINUE
     IERR=0
     LENGTH=1
     IF (ICODE.LE.MAXAVARY) GOTO 600
        FORM=.F.
        DO 500 J=2,4
           INDEX=J
           IF (KOMPCH(RESPONCE,INDEX,'#',1).NE.0) GOTO 500
                CALL CONCAT(RESPONCE,INDEX,',',1)
                DECODE(RESPONCE,5) ILIM,RESPONCE
                CALL REMOSEMI(RESPONCE,ILIM,FORM)
                IF (ILIM.LE.MAXCYC) GOTO 500
                   PRINT40,ILIM,MAXCYC
40                 FORMAT('0',' # CYCLES > MAXCYC'/I4,' >',I4/
    &              'PLEASE REENTER')
                   GOTO 200
500     CONTINUE
     IF (.NOT.FORM) GOTO 600
        PRINT ,' FORMAT ERROR IN INPUT VALUES, PLEASE REENTER'
       GOTO 200
600  CONTINUE
     CALL FXALT(MULTIERR)
     DECODE(RESPONCE,5) (X(I),I=1,ILIM)
     IF ((IERR.NE.32).AND.(IERR.NE.67)) GOTO 700
     CALL FXALT(DAYSERR)
     IERR=0
     DECODE(RESPONCE,5) (X(I),I=1,ILIM),LENGTH
     IF ((IERR.NE.32).AND.(IERR.NE.67)) GOTO 700
     IERR=0
     DECODE(RESPONCE,5) (X(I),I=1,ILIM),JSTART,LENGTH
     IF (IERR.NE.32) GOTO 650
         PRINT ,'FORMAT ERROR IN INPUT VALUES, PLEASE REENTER'
         GOTO 200
650  CONTINUE
     IF((JSTART.GT.0).AND.(JSTART.LE.LENGTH).AND.(LENGTH.LE.MAXDAY))
    & GOTO 675
        PRINT ,'ILLEGAL DAYS SPECIFICATION, PLEASE REENTER'
        GOTO 200
675  CONTINUE
     ISTART=JSTART
     LENGTH=LENGTH-ISTART+1
     EXIT=.T.
700  CONTINUE
     IF (LENGTH.GT.0) GOTO 715
        PRINT ,'ILLEGAL LENGTH SPECIFICATION, PLEASE REENTER'
        GOTO 200
715  CONTINUE
     LENGTH=MINO(LENGTH,IDAYS-ISTART+1)
     DECODE(RESPONCE,5) (XVALUE(I),I=1,ILIM)
     GOTO (725,725,725,725,750) ,ICODE
725  CONTINUE
     CALL CHECK(MAXA+ICODE,XVALUE(1),LIST(MAXA+ICODE))
```

```
      CALL SPRAY(XVALUE(1),AVARY(ISTART,ICODE),LENGTH)
      GOTO 800
750   CONTINUE
      ICVCODE=ICODE-MAXAVARY
      DO 775 I=1,ILIM
         CALL CHECK(MAXA+ICODE,XVALUE(I),LIST(MAXA+ICODE))
         CALL SPRAY(XVALUE(I),CVARY(ISTART,I,ICVCODE),LENGTH)
775   CONTINUE
      DO 785 I=ILIM+1,MAXCYC
         CALL SPRAY(XVALUE(ILIM),CVARY(ISTART,I,ICVCODE),LENGTH)
785   CONTINUE
800   CONTINUE
      ISTART=ISTART+LENGTH
      IF (ISTART.GT.IDAYS) EXIT=.T.
      ITEM=XVALUE(1)
     IF (.NOT.EXIT) GOTO 200
900   CONTINUE
      CALL FXOPT(67,0,0,0)
      RETURN
      END
```

```
C*****************************************************************
C*****   VALUERR IS A FORTRAN SUBROUTINE USED AS AN ALTERNATE
C***** ERROR PROCEDURE TO TEST IF A PARAMETER VALUE HAS BEEN ENTER
C***** ALONG WITH ITS CODE (I.E. 15;.01). IF SO IT CORRECTS
C***** THE PROBLEM.
C*****************************************************************
      SUBROUTINE VALUERR
      COMMON /VDATA/ ITEM
      DO 200 I=2,20
         INDEX=I
         IF (KOMPCH(ITEM,INDEX,';',1).NE.0) GOTO 100
         CALL CONCAT(ITEM,INDEX,',',1)
         GOTO 300
 100  CONTINUE
 200 CONTINUE
 300 CONTINUE
     RETURN
     END
```

```
C************************************************************
C*****   MULTIERR IS A FORTRAN SUBROUTINE USED AS AN ALTERNATE
C***** EROR PROCEDURE TO TEST IF THE VALUE OF A PARAMETER WHICH
C***** MAY VARY HAS BEEN SPECIFIED FOR MORE THAN ONE DAY
C***** (I.E. .01*4). IF SO IT CORRECTS THE CHARCTER STRING.
C************************************************************
      SUBROUTINE MULTIERR
      COMMON /EDATA/ DAY,RESPONCE
       DO 200 I=2,25
          INDEX=I
          IF (KOMPCH(RESPONCE,INDEX,'*',1).NE.0) GOTO 100
          CALL CONCAT(RESPONCE,INDEX,',',1)
          IF (KOMPCH(RESPONCE,INDEX+1,'*',1).EQ.0)
     &         CALL CONCAT(RESPONCE,INDEX+1,DAY,1,2)
          GOTO 300
 100   CONTINUE
 200  CONTINUE
 300  CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*****   DAYSERR IS A FORTRAN SUBROUTINE USED AS AN ALTERNATE
C***** ERROR PROCEDURE TO TEST IF THE VALUE OF A PARAMETER WHICH
C***** MAY VARY HAS BEEN ENTERED FOR SPECIFIC DAYS (I.E. 5(5-15)).
C***** IF SO IT CORRECTS THE CHARACTER STRING.
C*****************************************************************
      SUBROUTINE DAYSERR
      CHARACTER*1 CHR(3)
      COMMON /EDATA/ DAY,RESPONCE
      DATA CHR/'(','-',')'/
      ICHR=1
      DO 200 I=2,25
         INDEX=I
         IF (KOMPCH(RESPONCE,INDEX,CHR(ICHR),1).NE.0) GOTO 100
         CALL CONCAT(RESPONCE,INDEX,',',1)
         ICHR=ICHR+1
 100  CONTINUE
 200 CONTINUE
     RETURN
     END
```

```
C**********************************************************
C*****   REMOSEMI IS A FORTRAN SUBROUTINE WHICH REPLACES THE ';'S
C***** IN RESPONCE WITH ','S. ';'S ARE PART OF THE PROPER FORMAT FOR
C***** ENTERING PARAMETER VALUES WHICH VARY BY CYCLE/DAY.
C**********************************************************
      SUBROUTINE REMOSEMI(RESPONCE,ILIM,FORM)
       CHARACTER*25 RESPONCE
      LOGICAL FORM
       NUM=0
       DO 200 I=2,25
          INDEX=I
          IF(KOMPCH(RESPONCE,INDEX,';',1).NE.0) GOTO 100
               NUM=NUM+1
               CALL CONCAT(RESPONCE,INDEX,',',1)
 100   CONTINUE
 200   CONTINUE
       IF (NUM.NE.ILIM-1) FORM=.T.
      RETURN
      END
```

PLOT PROGRAM

```
C*** OS29/N232D/SGM/CALLPLOT
      PARAMETER MAXDAY=30
      DIMENSION ARR1(MAXDAY),ARR2(MAXDAY)
      LOGICAL DEFAULT/.T./
       REWIND 7
       READ(7) SCALE,NUMDAY
       IF (NUMDAY .LE. MAXDAY) GO TO 10
          WRITE(6,5) NUMDAY,MAXDAY
 5        FORMAT(//'****ERROR IN CALLPLOT NUMDAY > MAXDAY'/
     &         ' NUMDAY = 'I5,' MAXDAY = ',I3)
          GO TO 99
 10    CONTINUE
       IF (SCALE .EQ. 0.0) DEFAULT=.F.
        DO 20 I=1,NUMDAY
        READ (7) ARR2(I),ARR1(I)
        IF (ARR2(I) .LE. 10.0) GO TO 12
           WRITE(06,6) IFIX(ARR2(I)),10
 12     CONTINUE
        IF (ARR1(I) .LE. SCALE) GO TO 20
           IF (DEFAULT) GO TO 15
             SCALE=ARR1(I)
             GOTO 20
 15        CONTINUE
           ARR1(I)=SCALE
           WRITE(06,6) IFIX(ARR1(I)),IFIX(SCALE)
 6         FORMAT(//'***',I4,' TOO LARGE TRUNCATED TO',I4)
 20     CONTINUE
        SCALE=FLOAT(IFIX((SCALE+49)/50))
        WRITE(6,35) (ARR2(I),I=1,NUMDAY)
  35    FORMAT ("1"///(5F8.2))
        WRITE (6,49)
 49     FORMAT ('1'////'   SORTIES'/'   PER AC'//)
 50     FORMAT ('1'////'   SORTIES'/'   PER DAY'//)
        CALL PLOT (ARR2,NUMDAY,0.1,0,"(F5.1)")
        WRITE (6,30) (IFIX(ARR1(I)+.5),I=1,NUMDAY)
  30    FORMAT ('1'///(5I8))
        WRITE (6,50)
        CALL PLOT (ARR1,NUMDAY,SCALE,1,"(I5)")
  99    CONTINUE
        STOP
        END
```

PRECEDING PAGE BLANK-NOT FILMED

```
*
*
      SUBROUTINE PLOT (ARR,NUMDAY,SCALE,ITYPE,FORM)
      PARAMETER MAXDAY=30,MAXA=MAXDAY*51
      DIMENSION ARR(MAXDAY)
      CHARACTER*10 FORM
      CHARACTER A*1(MAXDAY,51)
      CHARACTER YNUM*5(51)
      DATA A/MAXA*' '/
      DATA YNUM/50*'    I','     '/
*
      IF (ITYPE .EQ. 1) GO TO 10
      DO 15 I=5,50,5
         ENCODE(YNUM(I),FORM) I*SCALE
   15 CONTINUE
      GO TO 30
   10 CONTINUE
      DO 35 I=5,50,5
         ENCODE (YNUM(I),FORM) IFIX(SCALE*I)
   35 CONTINUE
   30 CONTINUE
*
      DO 20 J=1,NUMDAY
       K=IFIX(ARR(J)/SCALE+.5)
   20 A(J,K) = '*'
*
   45   CONTINUE
      DO 60 I=1,50
      WRITE (6,70) YNUM(51-I),(A(J,51-I),J=1,NUMDAY)
   70 FORMAT (3X,A5,30(1X,A1))
      DO 50 J=1,NUMDAY
         A(J,51-I)=' '
   50 CONTINUE
   60 CONTINUE
      WRITE (6,80)
   80 FORMAT (7X,'0-------------------10-------------------20',
     &          '-------------------30'///,
     &          28X,'DAY OF SCENARIO')
      RETURN
      END

*
```

APPENDIX D

FEDERAL INFORMATION PROCESSING STANDARD SOFTWARE SUMMARY

01. Summary date: Yr. 81 | Mo. 09 | Day 21

02. Summary prepared by (Name and Phone): Michael J. Konvalinka (301) 229-1000

03. Summary action: New X | Replacement __ | Deletion __

Previous Internal Software ID: ________

04. Software date: Yr. 81 | Mo. 09 | Day 21

05. Software title: The Sortie-Generation Model System Volume IV, Sortie-Generation Model Programmer's Manual

06. Short title:

07. Internal Software ID: None

08. Software type:
- X Automated Data System
- __ Computer Program
- __ Subroutine/Module

09. Processing mode:
- __ Interactive
- __ Batch
- X Combination

10. Application area

General:
- __ Computer Systems Support/Utility
- __ Scientific/Engineering
- __ Bibliographic/Textual
- __ Management/Business
- __ Process Control
- X Other

Specific: Logistics Capability Assessment

11. Submitting organization and address:
Logistics Management Institute
4701 Sangamore Road
P. O. Box 9489
Washington, D.C. 20016

12. Technical contact(s) and phone:
Mr. John B. Abell
Mr. Michael J. Konvalinka
(301)229-1000 AV 287-2779

13. Narrative

The Sortie-Generation Model System provides the capability for relating aircraft spares and maintenance manpower levels to the maximal sortie-generation capability of tactical air forces over time.

14. Keywords

Readiness; Resource Allocation; Sortie Generation Capability; Logistics Capability Assessment

15. Computer manuf'r and model: Honeywell G-635

16. Computer operating system: GCOS

17. Programming language(s): Cobol 600, Fortran 600/GMAP

18. Number of source program statements: 15000

19. Computer memory requirements: 49k words, 36 bits each

20. Tape drives: 4

21. Disk/Drum units: 1 Disk, 2 million words

22. Terminals: 1 time sharing

23. Other operational requirements:

24. Software availability: Available X | Limited __ | In-house only __

25. Documentation availability: Available X | Inadequate __ | In-house only __

26. FOR SUBMITTING ORGANIZATION USE

STANDARD FORM 185
JULY 1974
U.S. DEPT. COMMERCE NBS
FIPS PUB. 30

AD-A110 899 LOGISTICS MANAGEMENT INST WASHINGTON DC F/G 15/5
THE SORTIE-GENERATION MODEL SYSTEM. VOLUME IV. SORTIE-GENERATIO--ETC(U)
SEP 81 M J KONVALINKA, J B ABELL MDA903-81-C-0166
UNCLASSIFIED LMI-ML102-VOL-4 NL

3 3

END
DATE
FILMED
3 82
DTIC

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A110848

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

The Sortie-Generation Model System
Volume IV
Sortie-Generation Model Programmer's Manual

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

LMI Task ML102

7. AUTHOR(s)

John B. Abell, Michael J. Konvalinka

8. CONTRACT OR GRANT NUMBER(s)

MDA903-81-C-0166

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Logistics Management Institute
4701 Sangamore Road
P.O. Box 9489
Washington, D.C. 20016

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Assistant Secretary of Defense
(Manpower, Reserve Affairs, & Logistics)
The Pentagon, Washington, D.C.

12. REPORT DATE

September 1981

13. NUMBER OF PAGES

[redacted]

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)

"A" Approved for public release

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

Unlimited

18. SUPPLEMENTARY NOTES

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

Readiness; Resource Allocation; Sortie Generation Capability;
Logistics Capability Assessment

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

The Sortie-Generation Model System provides the capability for relating aircraft spares and maintenance manpower levels to the maximal sortie-generation capability of tactical air forces over time.

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

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)