GASP IIP

21

A MODIFIED VERSION OF GASP IIA

by

RAJAGOPALAN KRISHNASWAMY

B.E. (Mechanical Engineering)

University of Madras, India, 1970

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1973

Approved by:

L. E. Grosh

Major Professor

ACKNOWLEDGEMENTS

The author is deeply indebted to Dr. Louis E. Grosh for his valuable suggestions and constructive criticisms throughout the course of this work.

The author also wishes to thank Dr. Doris L. Grosh for her help in statistical analyses and Mr. John J. Devore for his assistance in computing.

Thankful appreciation is extended to Mrs. Bruce Peil for preparing the typescript.

THE FOLLOWING PAGES ARE BADLY SPECKLED DUE TO BEING POOR QUALITY PHOTOCOPIES.

THIS IS AS RECEIVED FROM CUSTOMER.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER PAGE

CHAPTER 1

INTRODUCTION

1.1 Systems Simulation

Simulation is a process in which a model of a system is analyzed to gain more insight into the system's behavior and performance characteristics. The model used should reflect all the important characteristics of the system being studied. The model describing a system can assume a variety of forms. It can be a physical model (model for a gas turbine or an airplane), an analog model, a symbolic model built with mathematical equations; or the model used can be a logical model, represented by a computer program. Computer simulation is essentially working with logical models of this kind. The magnitude and complexities of the problems encountered in modern industrial and technological areas have prompted the development and use of computer simulation.

Mihram (3, p. 214) mentions that all simulation models can be classified as either dynamic or static. Dynamic models exhibit properties that change with time. In dynamic models, time is considered to be one of the important state variables, whereas in static models time is not taken into consideration at all. A second scheme reported by Mihram classifies the models as stochastic and deterministic. A stochastic model mimics the random behavior of a system simulated. But in a deterministic model the random behavior of a system is not represented. In this work we shall primarily deal with dynamic models of the stochastic variety. When digital computers are used to simulate dynamic systems the continuous flow of time can be approximated

in two ways. The two methods as illustrated by Emshoff and Sisson (2, p. 194) are shown in Figure 1.1. In the first method, known as fixed-time interval simulation, time is advanced in discrete, equal intervals. In the second method time is broken into discrete intervals that represent the time between interactions among different elements of a system. These interactions result in a change in the state of the system and as seen in Figure 1.1, the time intervals between such state changes are generally unequal. The simulation thus performed is called a next-event simulation. It is seen in Figure 1.1 that in fixed-time interval simulation, in many time intervals, there are no state changes. This may lead to inefficient time-advance computations. Further, in this method some loss of information about the system's behavior is possible due to the uncertainty about the point in time at which the state of the system changes within each time interval. But this method is better suited than next-event simulation to systems wherein one or more of the state variables change continuously and can not be approximated by discrete state changes. In this work we restrict our discussion to next-event simulation.

Elements of a system interact within the system's boundary. Mihram (3, p. 215) contends that there are no hard and fast rules to precisely define the boundaries of any system. He adds that the systems analyst must use his judgement and draw a conceptual boundary for the system considered, such that the interactions between the elements within the boundary do not significantly affect the elements outside. The components of a system thus isolated can be classified as static and temporal. The static components are entities, their attributes and the relationships between the entities. Entities are nothing but the objects and elements present in the system.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

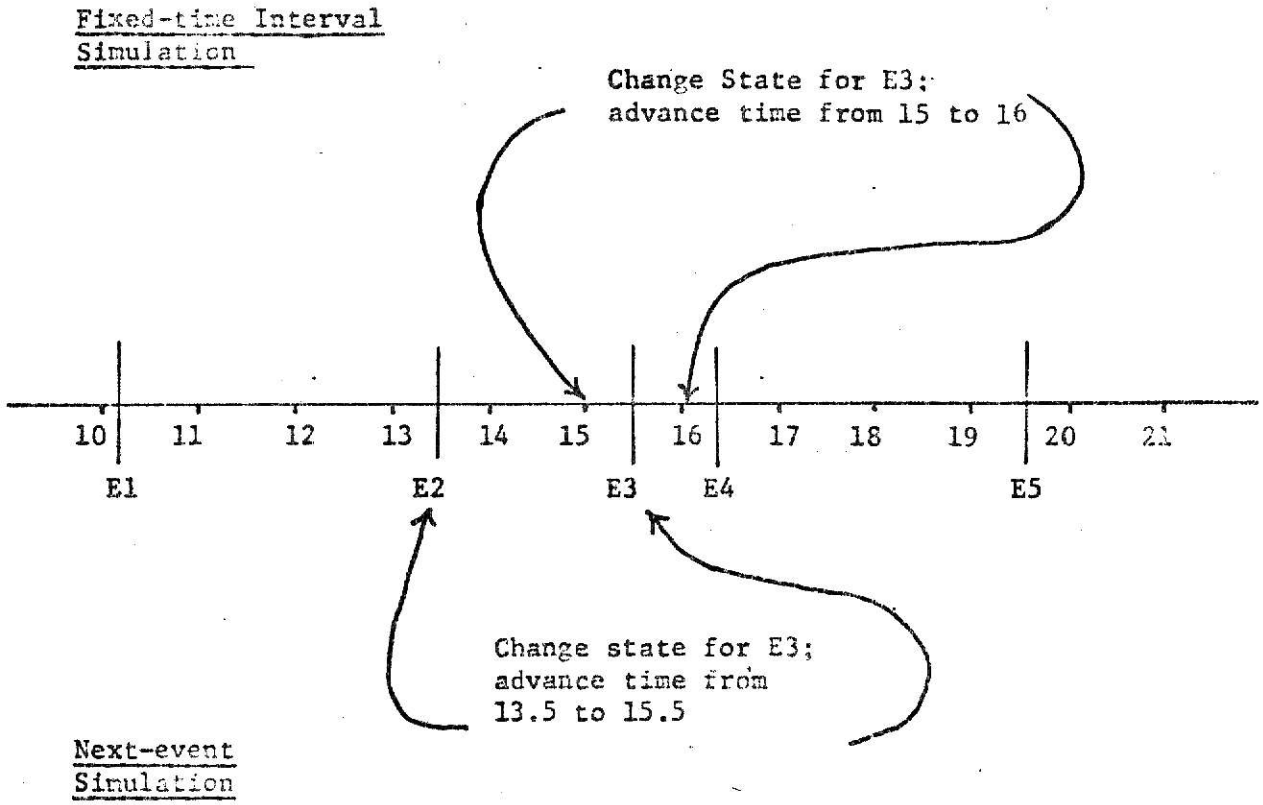


Figure 1.1 Time-advance Methods for Fixed-time Interval and Next-event Simulation.

Entities are characterized by their attributes. The state of an entity changes with time and its attributes indicate how much and to what states the entities have changed. The temporal components of a system, known as activities define the behavior of the system. Mihram (3, p. 218) lists the effects of activities in a system as:

1. To alter any of the attributes of one or more entities
2. To change the relationships between the entities
3. To transform system properties by changing the number of entities within the system.

Activities in general, begin with an initial event, last for a while and then are ended by a terminal event. These two types of events are classified as endogenous events. Another event type known as exogenous events refer to those events that are produced inside the system's boundary by elements that are outside. Exogenous events may be used to introduce new entities into the system. These entities which enter the system after the simulation has begun may depart before its completion. Such entities are called temporary entities as opposed to permanent entities which remain within the system's boundary throughout the period of simulation.

Information about entities and events can be stored in vectors or matrices established by the FORTRAN data declaration statements. Pritsker and Kiviat (1, p. 14) point out that the key to next-event simulation lies in organizing system events so that the order of execution within the computer corresponds to the order in which they would occur in the real system. Hence, in next-event simulation the relationships between events are of prime importance. The structure that maintains the relationships between entities or events is known as a file. The FORTRAN arrays that

accommodate all the files used in a simulation, are called the filing arrays. The term entry is used to describe both entities and events. An entry is a member of a file. A collection of entries ordered by a common attribute constitutes a file. The name cell is used to refer to a unit storage location. In the ensuing discussions, unless mentioned otherwise, all unit storage locations should be considered to be full words i.e. four bytes long.

1.2 GASP, GASP II and GASP IIA

1.2.1 GASP, A Next-Event Simulation Language

GASP-General Activity Simulation Program- is one of the oldest next-event simulation languages available. It was initially developed at U. S. Steel (8). It is FORTRAN based and consists of twenty-three subprograms which provide the analyst a framework to build a simulation model and execute it. The names and functions of these subprograms as illustrated by Pritsker and Kiviat (1, p.29) are given in Table 1.1. The standard manner of using GASP consists of precompiling all the GASP subprograms into a program library data set. Whenever the user runs a simulation problem, the user written subprograms are compiled and linked to the GASP data set. GASP is modular and hence it is easily fitted into small and medium sized computers. Since it is FORTRAN based it needs no special compiler and can be learned easily. Other simulation languages such as GPSS and SIMSCRIPT need special compilers to operate. GASP bears a strong resemblance to SIMSCRIPT. Programs written in GASP can be easily converted to SIMSCRIPT and vice versa. GASP II which originated at Arizona State University is an outgrowth of the original GASP (1).

Function	Subprograms
GASP Executive	SUBROUTINE GASP(NSET)
Initialization	SUBROUTINE DATAN(NSET)
Information Storage and	SUBROUTINE SET(JQ,NSET)
Retrival	SUBROUTINE FILEM(JQ,NSET)
	SUBROUTINE RMOVE(KCOL,JQ,NSET)
	SUBROUTINE FIND(XVAL,MCODE,JQ,JATT,KCOL,NSET)
Data Collection	SUBROUTEIN COLCT(X,N,NSET)
	SUBROUTINE TMST(X,T,N,NSET)
	SUBROUTINE HISTO(X1,A,W,N)
Statistical Computations	SUBROUTINE PRNTQ(JQ,MSET)
and Reporting	SUBROUTINE SUMRY(NSET)
Monitoring and Error	SUBROUTINE MONTR(NSET)
Reporting	SUBROUTINE ERROR(J,NSET)
Random Deviate Generators	SUBROUTINE DRAND(ISEED,RNUM)
	FUNCTION UNFRM(A,B)
	FUNCTION RNORM(J)
	FUNCTION RLOGN(J)
	FUNCTION ERLGN(J)
	SUBROUTINE NPOSN(J,NPSSN)
Other Support Routines	FUNCTION SUMQ(JATT,JQ,NSET)
	FUNCTION PRODQ(JATT,JQ,NSET)

Table 1.1 Functional Breakdown of the GASP II Subprograms.

1.2.2 GASP II Filing Array

GASP II uses a two-dimensional array NSET as the filing array. In FORTRAN two-dimensional arrays are stored by columns, as one-dimensional arrays. Starting from the first column, all the columns in the two-dimensional array are placed one after another to obtain an equivalent one-dimensional array. Figure 1.2 shows the correspondence between a two-dimensional array and an equivalant one-dimensional array. In a two-dimensional array that has seven rows, the cell (I, J) will be in position $(J-1)*M+I$, in its equivalant one-dimensional representation. It is seen in Figure 1.2 that the cell (3,2) occupies position (2-1)*5+3 = 8 in the one-dimensional array. It should be noted that the number of columns in a two-dimensional array does not play any role in locating a cell of the two-dimensional array in its one-dimensional equivalant. In GASP II, the variables MXX and ID represent the number of columns and rows, respectively, in NSET. Since the value of ID is not needed to locate a cell in NSET, in all the precompiled subprograms, the filing array is dimensioned as NSET (MXX, 1) by means of a DIMENSION statement. NSET is passed as an argument to all subprograms that use the filing array. The size of NSET is specified by giving a value to ID in the user written MAIN program as DIMENSION (MXX, ID). Thus, the user is able to vary the size of the filing array for particular problems. However by specifying the value of MXX in all the precompiled subprograms, a limitation is put on the number of attributes an entry can have.

Each column in the filing array accommodates an entry. The entries in a file are related and have one or more common attributes. One of these attributes determine the ordering of entries in that file. This attribute is called the 'ranking attribute'. Once this 'ranking attribute' is established

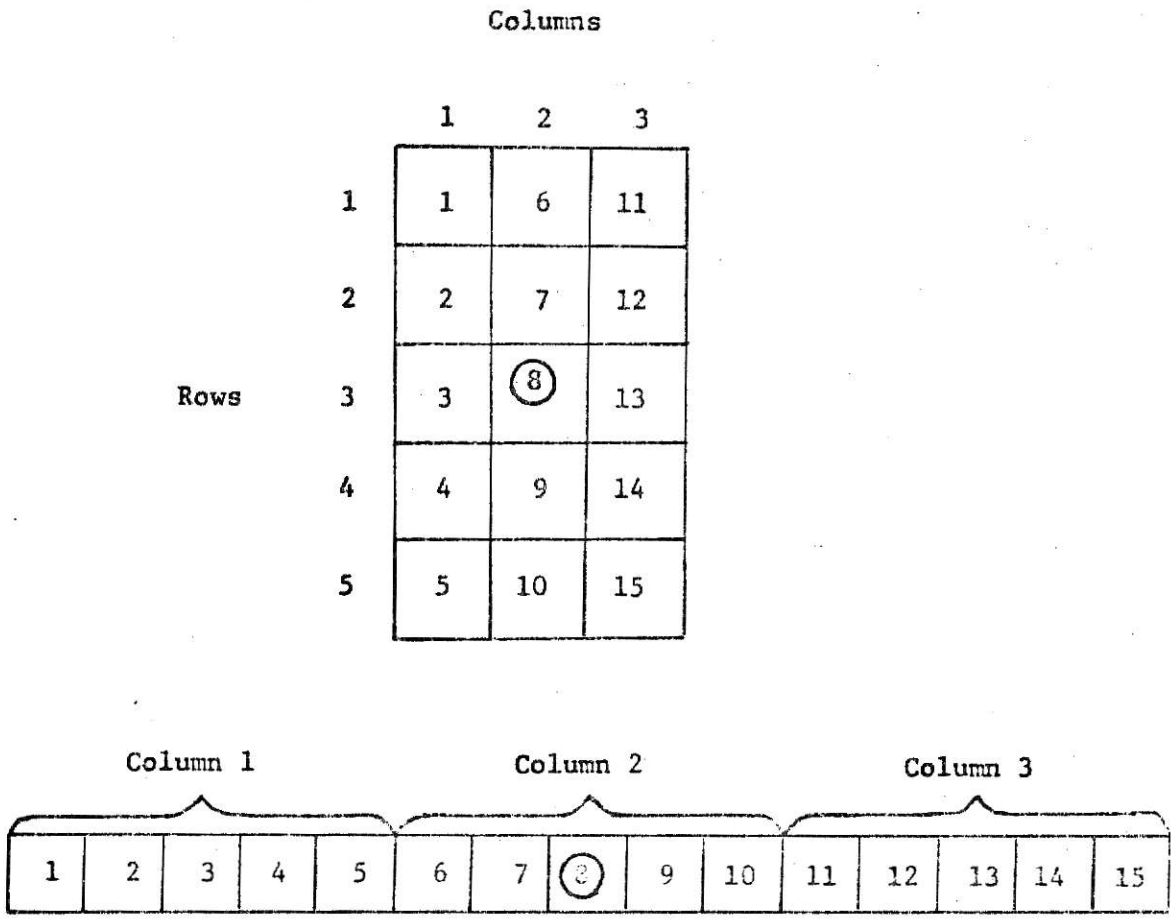


Figure 1.2 Two-dimensional Array and an Equivalent One-dimensional Array.

and the ranking is performed, every entry in the list, except the first and the last, should have a successor and a predecessor. In GASP II, two sets of pointers called 'successor pointers' and 'predecessor pointers' are used to give the column numbers of the successor and the predecessor of an entry. The first entry (MFE) in the file does not have a predecessor. Hence, the first entry in each file is given a standard code 9999 as its predecessor pointer value. Similarly, the last entry (MLE) of a file does not have a successor and hence assumes a standard code 7777 as its successor pointer value. One other standard code 8888 is used to denote the last unused column in the filing array. The last two rows in the filing array are used to store the successor and predecessor pointers. Those two rows are called successor row (MX) and predecessor row (MXX) respectively. Figure 1.3 shows entries in NSET, linked by successor and predecessor pointers.

1.2.3 GASP II Arrays

In addition to the filing array NSET, Gasp II uses a numberof other one-dimensional and two-dimensional arrays to carry out different data management chores involved in a simulation problem. According to their functions these arrays can be classified as:

1. Control arrays for data management
2. Statistical storage arrays
3. Information storage arrays

These arrays appear in a blank common block which is used by most of the subprograms in GASP II. It is very important for the successful use of GASP that this common block is properly set up in all subprograms needing access to GASP arrays. For all the arrays that appear in COMMON, suitable dimensions should be specified in the precompiled subprograms. A brief

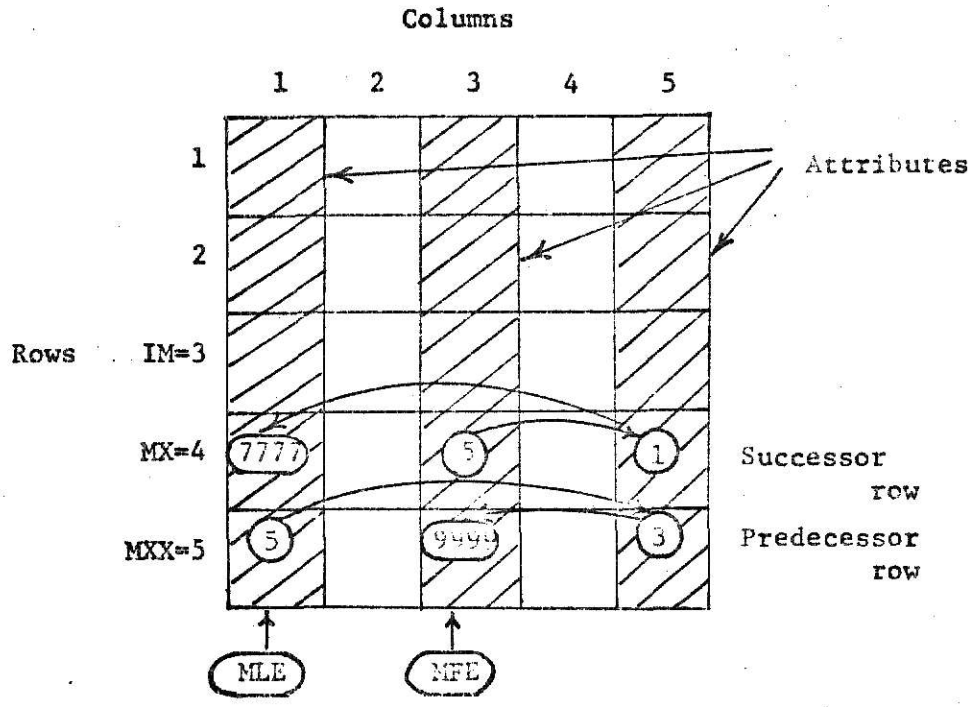


Figure 1.3 Entries in NSET Linked by Pointers.

discussion on the functional breakdown of GASP arrays has been given below.

1.2.3 (a) Control Arrays for Data Management

Besides the filing array, eleven other one-dimensional arrays are used in GASP II as control arrays for data management. These arrays can again be subclassified according to their functions:

1. A one-dimensional buffer array for transferring data to and from the filing array.
2. Five one-dimensional arrays to collect file statistics. These arrays resemble the arrays SUMA and SSUMA in their functions and are used exclusively to collect file statistics.
3. Five one-dimensional arrays for file management.

These arrays have been listed in Table 1.2 with an indication of functions performed by each group of arrays. Specifying the dimension for the array ATRIB in the precompiled subprograms will limit the number of attributes in an entry. Similarly, specifying the dimensions for other control arrays will limit the number of files that can be used in a simulation problem to some prespecified value.

1.2.3 (b) Statistical Storage and Information Storage Arrays

In GASP II, a one-dimensional array and three two-dimensional arrays are used to collect statistics and furnish histograms for the different variables used in a simulation problem. Specifying the dimensions of the arrays in the precompiled subprograms limits the number of statistics that may be estimated by the simulation.

Two other arrays, a one-dimensional and a two-dimensional array are used in GASP II for information storage. For a detailed description of

NO.	FUNCTIONS	ARRAYS
1	Filing array	NSET (MXX,ID)
2	Control Arrays for Data Management:	
	Buffer array	ATRIB (IM)
	Arrays for file management	INN (NOQ)
		KRANK (NOQ)
		MLE (NOQ)
		MFE (NOQ)
		MLC (NOQ)
	Arrays for file statistics	NQ(NOQ)
		QTIME (NOQ)
		ENQ (NOQ)
		VNQ (NOQ)
		MAXNQ (NOQ)
3	Statistical Storage Arrays:	
	To prepare histograms	NCELS (NHIST)
		JCELS (NHIST,MXC)
	To collect point estimates	SUMA (NCLCT,J)
	To collect time weighted statistics	SSUMA (NSTAT,J)
4	Information Storage Arrays:	
	To furnish parameters to stochastic generators; other general purpose use	PARAM (NPRMS,4)
	To store a name	NAME (N)

Table 1.2 Functional Breakdown of GASP II Arrays

all the GASP II arrays, the reader should refer to 'Simulation with GASP II' by Pritsker and Kiviat (1). Figure 1.4 shows a functional breakdown of GASP II arrays.

1.2.4 GASP IIA Filing Arrays

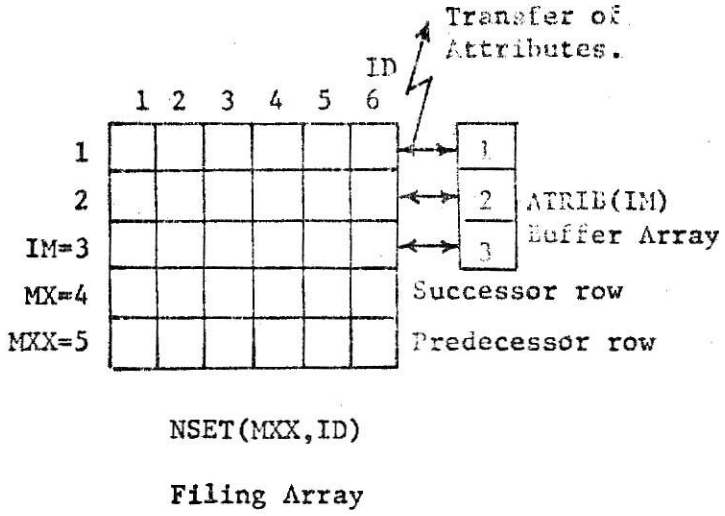
The filing array NSET used in GASP II is an integer array, whereas the buffer array ATRIB from which the attributes are transferred to the filing array, is a real array. During the transfer from ATRIB to NSET the attributes are changed from real to integer. This requires a scale factor to be specified by the user to minimize truncation errors (1, p.67). However, the truncation errors arising out of this scaling operation cannot be completely eliminated. Further, storing real numbers in an integer array will limit the magnitude of the numbers stored. Efforts were made to remove these difficulties in GASP IIA, an extended version of GASP II (1, p. 257). Instead of a single two-dimensional array NSET, GASP IIA uses two one-dimensional arrays, NSET for storing integer attributes and QSET for storing real attributes. This leads to the creation of one more buffer array called JTRIB for integer numbers. In GASP IIA the number of integer and real attributes in an entry are specified through the variables IM and IMM respectively. Figure 1.5 (a) and 1.5 (b) show the GASP IIA filing arrays and their relation to GASP II filing array. Even though NSET and QSET are one-dimensional arrays, for the sake of illustration they have been shown as two-dimensional arrays in Figure 1.5 (a). QSET has been shown as a two-dimensional array with five columns (ID = 5) and two rows (IMM = 2). Similarly, NSET has been represented as a two-dimensional array with five columns and three rows (MXX = 3).

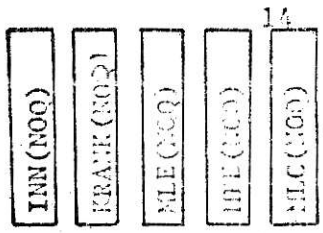
ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

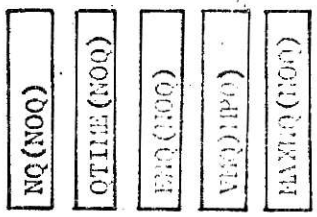
THIS IS THE BEST COPY AVAILABLE

I. Data Storage Arrays

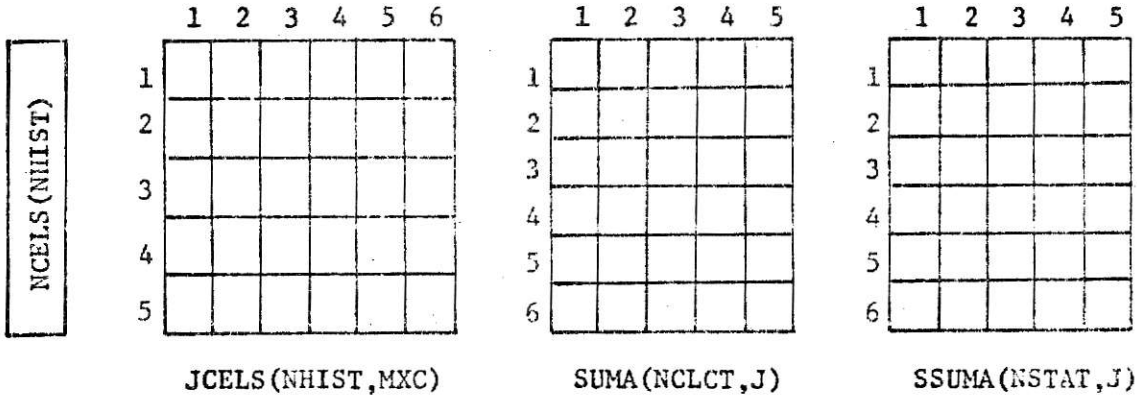




File Management Control Arrays



II. Statistical Storage Arrays



III. Information Storage Arrays

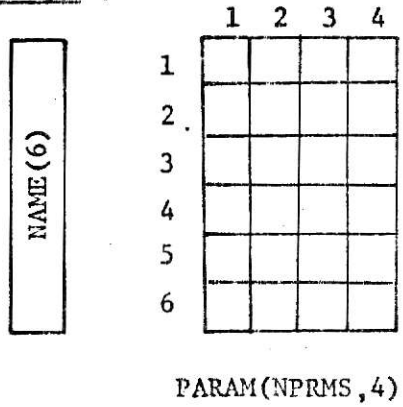


Figure 1.4 Functional Breakdown of GASP II Arrays.

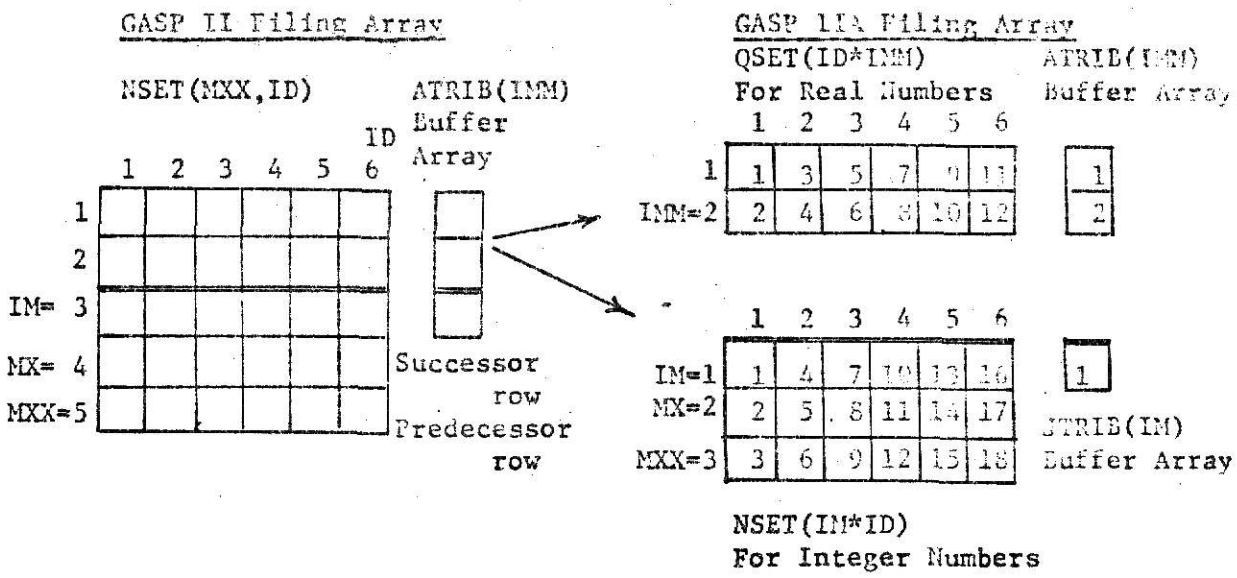


Figure 1.5 (a) Relationship Between GASP II and GASP IIA Filing Arrays.

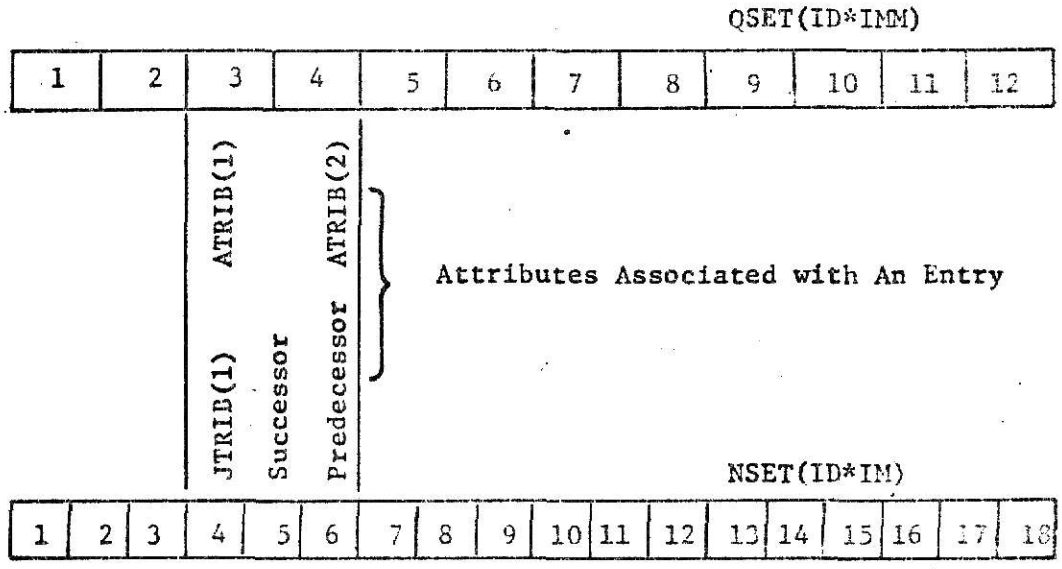


Figure 1.5 (b) GASP IIA Filing Arrays NSET and QSET.

When a particular column in QSET is used to store the real attributes of an entry the corresponding column in NSET is used to store integer attributes and the two pointers for that entry. In GASP IIA, before starting a simulation run, the value of IMM is determined such that the columns in QSET shown in Figure 1.5 (a) are big enough to accommodate the entry with the largest number of real attributes. Similarly, the value of IM should be determined such that the columns in NSET are large enough to accommodate the entry with the largest number of integer attributes plus the pointers of that entry.

To illustrate the determination of IM and IMM in GASP IIA, consider the following example. Assume three files; the first file, which is always the event file will handle three different events. The number of real and integer attributes for each event in the first file and for each entry in the second and third files are:

Example 1.1

File 1	Real Attributes	Integer Attributes
Event 1	1	1
Event 2	6	2
Event 3	2	2
File 2	1	6
File 3	2	2

Scanning the column for real attributes, it is seen that Event 2 in File 1 has the largest number of real attributes. Hence, the value of IMM is fixed as 6. Similarly, the entries in File 2 require the largest number of integer attributes. The value of IM is fixed as 6. Further, for

every entry filed, two additional cells are required in NSET to store the successor and the predecessor pointers. Hence, the value of MXX will be 8. Assume, the simulation will require at most 50 entries. Then, the value of ID is fixed as 50. Once the values of IM, IMM, and ID are determined, the sizes of NSET and QSET can be determined as:

Number of cells in QSET = IMM * ID
= 6 x 50
= 300 cells

Number of cells in NSET = MXX * ID
= 8 x 50
= 400 cells

The combination of any one column in NSET and the corresponding column in QSET can be viewed as a unit filing area. Once the values of IMM and IM are fixed, the length of the unit filing area is also fixed. Hence, the filing area in GASP IIA is essentially made up of a finite number of fixed length unit filing areas.

In all the precompiled GASP IIA subprograms, NSET and QSET are given unit dimensions as:

```
DIMENSION NSET (1), QSET (1)
```

To all the subprograms that use the filing arrays, NSET and QSET are passed as arguments. The sizes of the filing arrays are specified through the DIMENSION statement in the user written MAIN program. This arrangement allows the user to specify the filing array sizes according to the needs of the individual problems.

1.3 Statement of Objectives and Indications On Areas for Improvements

1.3.1 Statement of Objectives

The primary objectives that have motivated this work can be briefly stated as:

1. To eliminate the dimensional restrictions of precompiled arrays.
2. To improve the utilization of GASP IIA arrays by restructuring and modifications.
3. To obtain faster execution by eliminating redundant coding, adding new features and modifying existing features.
4. To improve the effectiveness of GASP IIA by incorporating new facilities.

Having thus stated the objectives, we shall now indicate the areas where modifications can be made and improvements can be effected to fulfill these objectives.

1.3.2 Difficulties Due to Dimensioning GASP IIA Arrays in Precompiled Subprograms

Specifying the dimensions of GASP IIA arrays in the precompiled subprograms will, in most cases, result in two types of problems; one due to 'over dimensioning' and another due to 'under dimensioning'.

Problems due to 'over dimensioning' arise, when the storage requirements of a particular problem are not big enough to use the GASP IIA arrays to their full capacity. This results in poor utilization of these arrays. Problems due to 'under dimensioning' arise when the storage space requirement in a particular problem exceed the amount of storage space reserved by specifying the dimensions of precompiled GASP IIA arrays. The GASP IIA arrays act as a built-in constraint, because specifying their dimensions in the

precompiled subprograms puts upper limits to the values of many GASP variables. Once specified, the dimensions of the precompiled arrays are not changeable without recompiling most of the GASP subprograms. GASP IIA has been used for research work at Kansas State University. Experience has shown that GASP needs to be recompiled at least once each year to modify the dimensions of one or more of the GASP arrays. In Chapter 2 ways and means have been suggested to overcome the problems due to 'over dimensioning' and 'under dimensioning'.

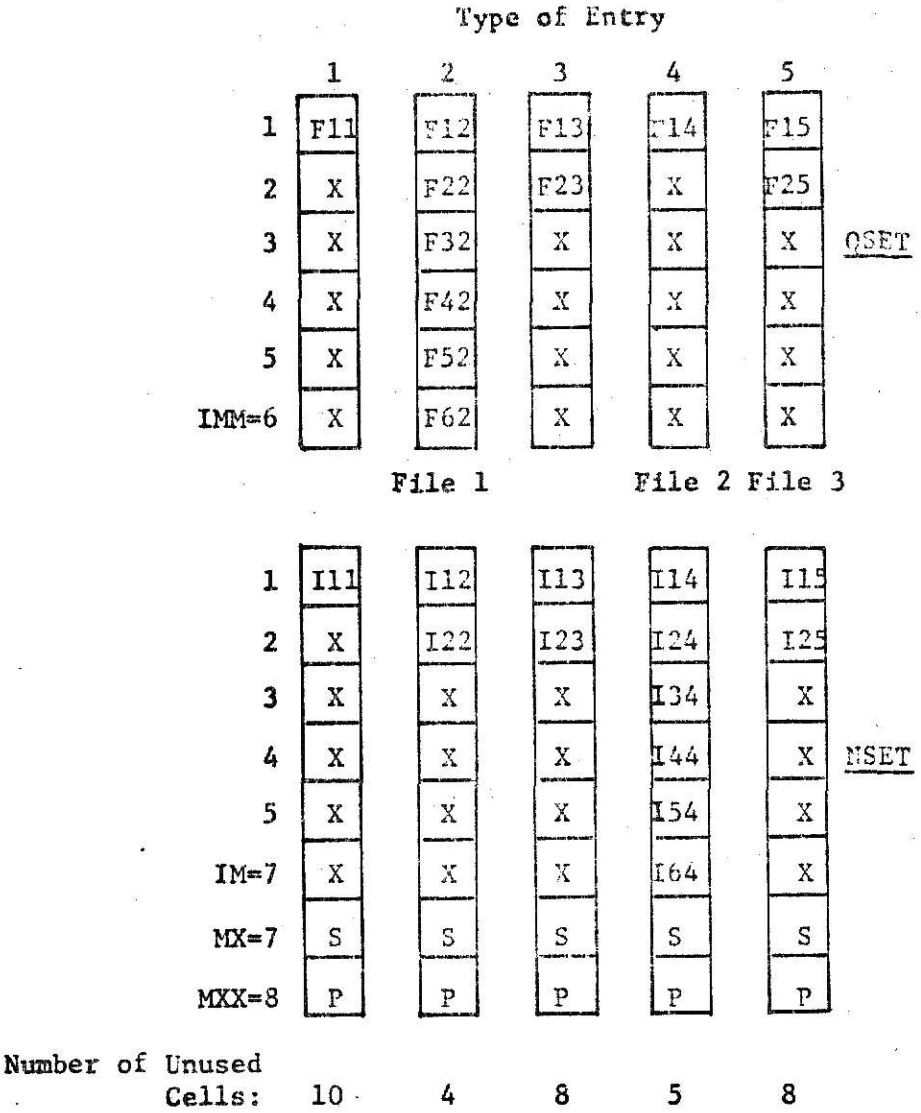
1.3.3 Poor Storage Space Utilization in the Filing Area

In GASP IIA when an entry is filed, if the number of floating point attributes in that entry is less than the IMM value specified, then the remaining cells in that column of QSET are not utilized. Similar wastage occurs in NSET when the number of fixed point attributes in the entry filed in less than IM.

In Example 1.1, five different types of entries are involved. The three events in File 1 use three different types of entries. File 2 and File 3 use two other types of entries. The type of an entry depends on what the attributes of that entry represent. Figure 1.6 shows the number of unused cells for each type of entry. A new method for storing the attributes has been outlined in Chapter 2. This method gives a better utilization of storage space in the filing area.

1.3.4 Unnecessary Movement of Entries to and from Buffer Arrays

In GASP IIA whenever an entry is to be filed the real attributes are first placed in the buffer array ATRIB and the integer attributes are placed in the buffer array JTRIB. Then the user calls the subroutine FILEM which inserts the attributes placed in ATRIB into a column in QSET and the



LEGEND

FMN - M^{th} floating point attribute of the N^{th} type entry.

IMN - M^{th} fixed point attribute of the N^{th} type entry.

X - Space allocated but not used.

Figure 1.6 Utilization of Storage Space in GASP IIA

attributes placed in JTRIB into a corresponding column in NSET. Similarly, whenever the user needs the attributes of an entry for further manipulations he calls the subroutine RMOVE which transfers the attributes of that entry from QSET to ATRIB and NSET to JTRIB. The attributes thus placed in ATRIB and JTRIB are available to the user. The movement of entries to and from buffer arrays are unnecessary and time consuming. A modified data storage and retrieval procedure in which the user communicates directly with the filing array without buffer arrays acting as intermediaries has been proposed in Chapter 3. This method should help to achieve faster execution.

1.3.5 Some More Proposals for Improvements

In GASP IIA whenever a file printout is requested the subroutine MONTR prints out the entire filing area, both the used and unused part of it. This involves a large amount of I/O operations. Also, it is difficult to sort out the contents of the individual files from this printout. Hence, a modified method has been proposed in Chapter 3 with an aim to reduce the amount of I/O operations involved and get more relevant information from the file printouts. Another proposed extension mentioned in Chapter 3 is the selective monitoring of events.

The addition of new random number generators has been discussed in Chapter 4. Every stochastic generator in GASP IIA uses the random number generator. The random number generator is at the heart of any stochastic simulation. Hence, if the random number generator is improved, it will improve the performance of all stochastic generators in general. The provision of a separate generator for antithetic random numbers has also been discussed. A proposal for a new RESET subroutine has also been made in Chapter 4. The results of this work has been discussed in Chapter 5.

Extensive restructuring of GASP IIA is required to fulfill our objectives. This involves conceptual changes in the working of many subprograms and a great amount of recoding. We have named the new, restructured GASP as GASP IIP; 'P' for pointers.

CHAPTER 2

RESTRUCTING THE GASP ARRAYS

2.1 Eliminating the Constraints Imposed by Precompiled Arrays

In GASP IIA, except for the filing arrays NSET and QSET, all the other arrays appear in a blank COMMON block used by most of the GASP subprograms. These arrays appearing in the COMMON block should be suitably dimensioned in the precompiled subprograms. This creates problems due to 'over dimensioning' and 'under dimensioning'. 'Over dimensioning' results in poor storage space utilization. 'Under dimensioning' makes the GASP arrays act as built-in constraints and necessitates the recompilation of GASP subprograms whenever the dimensions of one or more of the GASP arrays have to be altered.

We propose to overcome these problems by using a set of pointers and a single one-dimensional array that serves the purpose of all the GASP IIA arrays. By using the pointers we shall arrange all the GASP arrays, one after another, into a single one-dimensional array. This will be explained in Section 2.2. For the present let us assume this is possible. Before we can accommodate all the GASP IIA arrays into a one-dimensional array we should convert the two-dimensional arrays in GASP IIA into one-dimensional arrays. Here we should note that some of the GASP arrays are integer arrays and some are real number arrays. So, the GASP IIP array should be able to accommodate real numbers as well as integer numbers. To enable this we shall give two different names to the GASP IIP array; an integer name NSET and a real name QSET. We can accomplish this by declaring two arrays NSET and

QSET in a DIMENSION statement in the main program and then making them equivalent as:

MAIN

```
DIMENSION NSET(N),QSET(N)
EQUIVALENCE (NSET(1),QSET(1))
```

where N is the desired size of the GASP IIP array. But, we would like to have NSET and QSET in COMMON, so that they need not be passed as arguments to other subprograms. The FORTRAN IV compiler does not allow two variables specified in a COMMON statement to be made equivalent. We can get around this difficulty by specifying the size of the GASP IIP array in the user written MAIN program as indicated below:

MAIN

```
DIMENSION QSET(N)
COMMON VAR1,VAR2,.....,VARN,NSET(1)
EQUIVALENCE (NSET(1),QSET(1))
```

The above procedure pulls QSET into the GASP IIP blank COMMON block and makes it equivalent to NSET. The GASP IIP array of length specified in the DIMENSION statement is set up at the end of the COMMON block. In all the precompiled GASP subprograms, NSET and QSET are given unit dimensions as:

GASP subprograms

```
DIMENSION QSET(1)
COMMON VAR1,VAR2,.....,VARN,NSET(1)
EQUIVALENCE (NSET(1),QSET(1))
```

This procedure has three distinct features:

1. The exact dimension of the GASP IIP array need not be specified in the precompiled subprograms.
2. NSET and QSET are placed in COMMON. Hence, they need not be passed as arguments to the called subprograms.
3. NSET and QSET are equivalent. Hence the GASP array can be referred to either as NSET or as QSET i.e. the same array can handle both integer and real numbers. Thus we have eliminated the constraints imposed by precompiled GASP arrays.

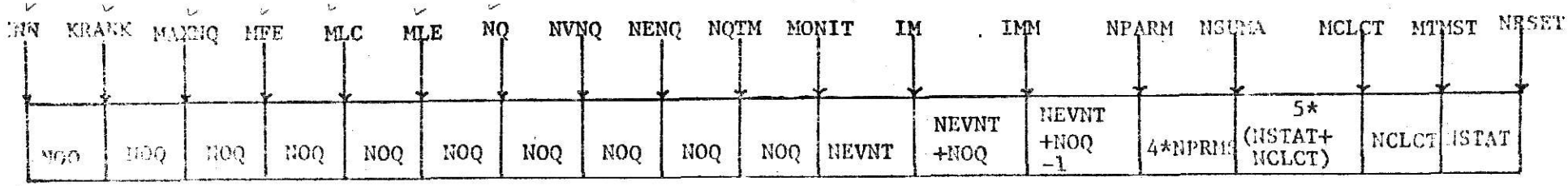
2.2 The Concept of Pointers and Elimination of Problems Due to 'Over Dimensioning'

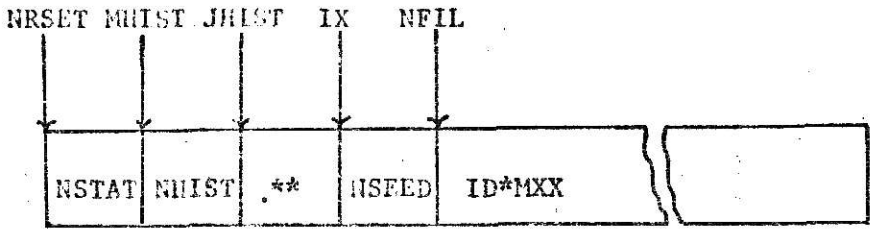
GASP IIP used a set of twenty-two pointers which divide the GASP IIP array into twenty-two segments of different lengths as shown in Figure 2.1. A pointer indicates the beginning of a segment. The value of a pointer is equal to the length of the GASP IIP array previous to that pointer. Hence,

Value of N^{th} pointer=Value of $(N-1)^{th}$ pointer+Length of $(N-1)^{th}$ segment

It can be seen from Figure 2.1 that the value of the pointer INN will always be zero. The last segment of the GASP IIP array, indicated by the pointer NFIL is used as the filing area. Each of the twenty-two segments in the GASP IIP array can be looked upon as being equal to a one-dimensional array. Table 2.1 shows the relationship between GASP IIA one-dimensional arrays and their corresponding segments in the GASP IIP array. For every one-dimensional array in GASP IIA, a pointer has been provided in GASP IIP. Using these pointers the GASP IIA one-dimensional arrays have been arranged one after another in the GASP IIP array.

NOTE: The value of a pointer is equal to the length of the array in front of that pointer. Hence, INN=0,KRANK=NOQ etc.





GASP IIP Filing Array

$$** \quad NHIST + \sum_{I=1}^{NHIST} NSET(JHIST+I)+NHIST*2$$

Figure 2.1 GASP IIP Array

GASP IIA	GASP IIP	
INN(JQ)	NSET(INN+JQ)	
KRANK(JQ)	NSET(KRANK+JQ)	
MAXNQ(JQ)	NSET(MAXNQ+JQ)	
MFE(JQ)	NSET(MFE+JQ)	1≤JQ≤NOQ
MLC(JQ)	NSET(MLC+JQ)	
MLE(JQ)	NSET(MLE+JQ)	
NQ(JQ)	NSET(NQ+JQ)	
VNQ(JQ)	QSET(NVNQ+JQ)	
ENQ(JQ)	QSET(NENQ+JQ)	
NQTM(JQ)	NSET(NQTM+JQ)	
NCELS(N)	NSET(JHIST+JQ)	1≤N≤NHIST
IX(I)	NSET(IX+I)	1≤I≤NSEED

Table 2.1 Correspondence between GASP IIA One-dimensional Arrays and Segments of GASP IIP Array.

The two-dimensional arrays in GASP IIA - SUMA, SSUMA, PARAM, JCELS - have been converted into one-dimensional arrays before being accommodated into the GASP IIP array. The arrays SUMA and SSUMA have been made to occupy the same segment, the beginning of which is indicated by the pointer NSUMA. First the values of SUMA or the statistics collected by the subroutine COLCT are stored and then the values of SSUMA or the statistics collected by the TMST subroutine. Similarly, GASP IIA arrays NCELS and JCELS have been accomodated into the same segment using the pointer JHIST. The array PARAM has been fitted into the GASP IIP array using the pointer NPARM. Table 2.2 shows the relation between the two-dimensional arrays of GASP IIA and their corresponding segments in the GASP IIP array.

Apart from the segments shown in Table 2.1 and 2.2, seven other segments are used to store the values of variables newly introduced in GASP IIP. Table 2.3 gives the names of the pointers associated with the segments and illustrates how the lengths of these segments can be calculated. The total length of the GASP IIP array can be obtained by the following expression.

$$\text{Total length} = (12*NOQ)+(3*NEVNT)+(4*NPRMS)+(6*NCLCT)+(7*NSTAT) +(4*NHIST) + \left(\sum_{N=1}^{NHIST} NSET(JHIST+N)\right)+NSEED+(ID*MXX)-2$$

The values of the variables appearing in the above expression will normally be known before a simulation run is made. Hence an estimate of the GASP IIP array length can be made. According to this estimate the size of QSET is specified through a DIMENSION statement in the user written MAIN program. For every simulation problem the values of the pointers which determine the lengths of the segments are calculated at execute time to satisfy the specific storage space requirements of that problem. As a result,

GASP IIA	INDX	GASP IIA
SUMA(N,J) 1≤N≤NCLCT 1≤J≤5	(N-1)*5+J	QSET(NSUMA+INDX)
SSUMA(N,J) 1≤N≤NSTAT 1≤J≤5	(NCLCT+N-1)*5+J	QSET(NSUMA+INDX)
JCELS(N,IC) 1≤N≤NHIST 1≤IC≤(NCELS(N) 2)	NHIST $+\sum_{I=1}^{(N-1)}$ NSET(JHIST+I) + (N-1)*2+IC	NSET(JHIST+INDX)
PARAM(I,J) 1≤I≤NPRMS 1≤J≤4	(I-1)*4+J	QSET(NPARM+INDX)

Table 2.2 Correspondence between GASP IIA Two-dimensional Arrays and Segments of GASP IIP Array

Number	Name of Pointer	Length of Pointer	DATAX Card Type
1	INN	NOQ	2
2	KRANK	NOQ	2
3	MAXNQ	NOQ	2
4	MFE	NOQ	2
5	MLC	NOQ	2
6	MLE	NOQ	2
7	NQ	NOQ	2
8	NVNQ	NOQ	2
9	NENQ	NOQ	2
10	NQTM	NOQ	2
11	NEVNT	NOQ	2
12	IM	NEVNT+NOQ-1	2
13	IMM	NEVNT+NOQ-1	2
14	NPARM	4*NPRMS	2
15	NSUMA	(NCLCT+NSTAT)*5	2
16	MCLCT	NCLCT	2
17	MTMST	NSTAT	2
18	NRSET	NSTAT	2
19	MHIST	NHIST	2
20	JHIST	$\text{NHIST}+\sum_{N=1}^{\text{NHIST}} \text{NSET(JHIST+N)}+\text{NHIST}*2$	4
21	IX	NSEED	8
22	NFIL	ID*MXX	2

Table 2.3 Pointers in GASP IIP and Lengths of Segments Associated with Them.

in all segments except the last one we achieve 100% storage space utilization. Thus the pointer system adopted in GASP IIP eliminates the poor storage space utilization that results from 'over dimensioning' the precompiled GASP arrays. The GASP IIP array is more compact and enables more economical use of available storage space than the GASP IIA arrays.

2.3 Better Utilization of the Filing Area

In example 1.1 we indicated how in GASP IIA each type of entry under-utilized its allotted space in NSET and QSET. In GASP IIP the names NSET and QSET refer to the same one-dimensional array. As a result, every cell in the array can be used to store either a real or an integer number.

The last segment in the GASP IIP array, indicated by the pointer NFIL, is used as the filing area. This segment has been shown as a two-dimensional array in Figure 2.2. We shall refer to this as the GASP IIP filing array. Each column in the filing array represent a unit filing area. Here also as in GASP IIA, the length of the unit filing area is fixed once the value of MXX is specified. While filing an entry first the integer attributes are stored and then the real attributes. This type of attribute arrangement is possible in GASP IIP, only because every cell in the GASP IIP array can accommodate either a real or an integer number. The successor and the predecessor pointers are stored in the last two cells in every column. We shall once again use the values of Example 1.1 to illustrate how the value of MXX is determined in GASP IIP.

	1	2	3	4	ID 5	
1	1	6	11	16	21	
2	2	7	12	17	22	NSET or OSET
3	3	8	13	18	23	
4	4	9	14	19	24	Successor Pointers
MXX=5	5	10	15	20	25	Predecessor Pointers

Figure 2.2 GASP IIP Filing Area Represented as a Two-dimensional Array.

Example 2.2

File Number	A Number of Real Attributes	B Number of Integer Attributes	C Total Number of Attributes
File 1			
Event 1	1	1	2
Event 2	6	2	8
Event 3	2	2	4
File 2	1	6	7
File 3	2	2	4

In GASP IIP the value of MXX is determined by the entry having the maximum number of attributes. It is seen from column A that Event 2 has the maximum number of attributes (8). Hence, the value of MXX is fixed as 10, with two extra cells in each column provided for pointers. As in Example 1.1, if ID is given a value of 50, then,

Number of cells in NSET or QSET = MXX * ID

= 10 x 50

= 500 cells.

To accommodate the same number of entries, GASP IIP needs a filing array of 500 cells as compared to 400 cells for NSET and 300 cells for QSET in GASP IIA. Hence, in this particular example, the filing method followed in GASP IIP, enables a reduction of 200 cells in the filing area. Figure 2.3 shows how each type of entry is stored in the GASP IIP filing area.

	Type of Entry 1	2	3	4	ID 5
1	I11	I12	I13	I14	I15
2	F11	I22	I23	I24	I25
3	X	F12	F13	I34	F15
4	X	F22	F23	I44	F25
5	X	F32	X	I56	X
6	X	F42	X	I64	X
7	X	F52	X	F14	X
8	X	F62	X	X	X
9	S	S	S	S	S
MXX=10	P	P	P	P	P

LEGEND

FMN - M^{th} real attribute of N^{th} type entry.

IMN - M^{th} integer attribute of N^{th} type entry.

X - Unused cell

Number of Unused Cells Per Entry	6	0	4	1	4	----- GASP IIP
	10	4	8	5	8	----- GASP IIA(See Example 1.1)

Length of unit filing area:

GASP IIA ---- 14 cells

GASP IIP ---- 10 cells

Figure 2.3 Number of Unused Cells per Entry in GASP IIP

It is seen in Figure 2.3 that for each type of entry filed, the number of unused cells per entry is always more in GASP IIA by 4 cells than in GASP IIP. These 4 cells represent the difference in the lengths of GASP IIA and GASP IIP unit filing areas. The length of the unit filing area in GASP IIA is determined by the entry having the maximum number of real attributes and by the entry having the maximum number of integer attributes. The length of the unit filing area will be the same in both GASP IIA and GASP IIP, only when the same type of entry happens to have the maximum number of integer as well as real attributes. In all other cases the length of the unit filing area of GASP IIP will be less than that of GASP IIA. Hence, in most cases, the number of unused cells and also the filing area requirement will be less in GASP IIP as compared to GASP IIA.

2.4 Some Incidental Benefits

2.4.1 Conversion of Two-dimensional Arrays into One-dimensional Arrays

Earlier in this Chapter we mentioned about converting the GASP IIA two-dimensional arrays into one-dimensional arrays. This, besides enabling us to set up the pointer system, will also result in faster execution. As Larson (7) points out, subscript computations at object time are expensive for multi-dimensional arrays. Further, it is not advisable to use a vector when a scalar will do or to use an N-dimensional array when an N-1 dimensional array will do. Test programs have been run to determine the difference in execution time using a one-dimensional array in the place of a two-dimensional array. These test programs are shown in Appendix A.1.

In this example the subroutine COLCT has been used along with a random number generator called DRAND. The subroutine COLCT is called five thousand

times to compute point estimates for the random numbers generated by DRAND. In the first case COLCT uses a two-dimensional array SUMA to compute point estimates. In the second case, SUMA has been used as a one-dimensional array. The INTIME subroutine has been used to time the different sections of the programs. In each case by subtracting the time taken for random number generation from the total execution time, the time taken for computing the point estimate alone is computed. The results have been shown in Table 2.4. A reduction in execution time, in excess of 12% is obtained when a one-dimensional array is used instead of a two-dimensional array.

2.4.2 Placing NSET and QSET in COMMON

In GASP IIA, the filing arrays NSET and QSET appear in DIMENSION statements and carry unit dimensions in all the precompiled subprograms. Their sizes are specified through the DIMENSION statement in the user written MAIN program. Most of the GASP IIA subprograms need access to NSET and QSET and to these subprograms NSET and QSET are passed as arguments. In GASP IIP, the names NSET and QSET refer to the same one-dimensional array which has been accommodated at the end of the blank COMMON block. Since in GASP IIP NSET and QSET are in COMMON, they need not be passed as arguments. This will reduce the length of the parameter lists for many subprograms and yield simpler source codings. Table 2.5 shows the reduction in the length of theparameter lists for a number of subprograms. The number of stars appearing after a subprogram's name indicate its frequency of use in a normal simulation run. The subprograms with two stars are the ones most frequently used.

	Dimension of SUMA	
	Single	Double
Total Time	509	555
DRAND Time	184	184
Hence,		
COLCT Time	325	371

Saving in Execution Time = 12.4%

Table 2.4 Time Comparison with One-dimensional and Two-dimensional Arrays.

NOTE: All times are in hundredth of a second.

Names of Subprograms		Percentage Reduction in Lengths of Parameters Lists
GASP	**	
DATAN	*	
MONTR		1.00%
SUMRY	*	
EVNTS	**	
ERROR		
FILEM	**	66.6%
PRNTQ	*	
SET	**	
COLCT	*	50.0%
RMOVE	**	
LOCAT		
PRODQ		40.0%
SUMQ		
TMST	*	
FINDN		25.0%
FINDQ		
Stochastic Generators		0%

Table 2.5 Reduction in the Length of Parameter Lists When NSET and QSET Are Placed in COMMON.

CHAPTER 3

ELIMINATION OF BUFFER ARRAYS AND IMPROVED FACILITIES FOR MONITORING AND FILE PRINTOUTS

3.1 Elimination of Buffer Arrays

3.1.1 Changes in Some Subprograms

In GASP IIA whenever an entry is to be filed, the user first places the integer attributes of that entry in JTRIB and the real attributes in ATRIB and calls the subroutine FILEM. This subroutine transfers the attributes from ATRIB to QSET and JTRIB to NSET and then calls the subroutine SET to adjust the pointers. Similarly, when an entry is to be retrieved from the filing array subroutine RMOVE is called which transfers the attributes of that entry from QSET to ATRIB and NSET to JTRIB. The values thus placed in ATRIB and JTRIB are available to the user for further manipulations. After this transfer RMOVE calls the subroutine SET which adjust the pointers and reinitiatizes to zeroes the cells that contained the attributes. This kind of entry movement is unnecessary and increases the execution time. In GASP IIP buffer arrays are not used. Whenever an entry is to be filed the attributes are placed directly into the filing array. Also whenever needed the user directly access the attributes of an entry stored in the filing array. The elimination of buffer arrays has induced changes in the subprograms GASP, DATAN AND SET. The procedures adopted in GASP IIP for filing and retrieving of entries are described in Section 3.2.

3.1.2 Subroutine GASP in GASP IIP

As in GASP IIA, the subroutine GASP is the master control routine. A

great deal of restructuring has been done in this subprogram to support the additional features incorporated in GASP IIP. This subprogram has been entirely recoded. Figure 3.1 shows a detailed flow chart for the subroutine GASP as it appears in GASP IIP. This subroutine, as before, starts the simulation, selects events, sequences time, produces intermediate simulation results and initiates the print out of the final output when a simulation run is completed. In addition to doing these things a little differently than its counterpart in GASP IIA, this subprogram performs some additional functions.

GASP first calls the subroutine DATAX to initialize all the GASP variables and to read in the initial file entries. DATAX at this time calls the subroutine SET which sets up the pointers for the filing array. After this is done, GASP using the ENTRY PRNFL in subroutine PRNTQ prints out all the files. GASP then starts and conducts the simulation by obtaining the values of current time TNOW and current event JEVNT from the first entry MFE, in File 1 which is always the event file. Since the use of buffer arrays has been eliminated in GASP IIP, the values of TNOW and JEVNT are obtained directly from the filing area and not through ATRIB and JTRIB as in GASP IIA.

Next, the switch 'JMNIT' is tested to see if monitoring is to be done or not. If JMNIT > 0 then NSET (MONIT + JEVNT) is tested to see if this particular event is to be monitored. If this event is scheduled to be monitored - i.e. NSET (MONIT + JEVNT) > 0 - the subroutine MONTR is called. On the other hand if JMNIT = 0, or if the event is not scheduled to be monitored - i.e. NSET (MONIT + JEVNT) = 0 - then, no monitoring takes place. After this GASP tests the event code to see if this event

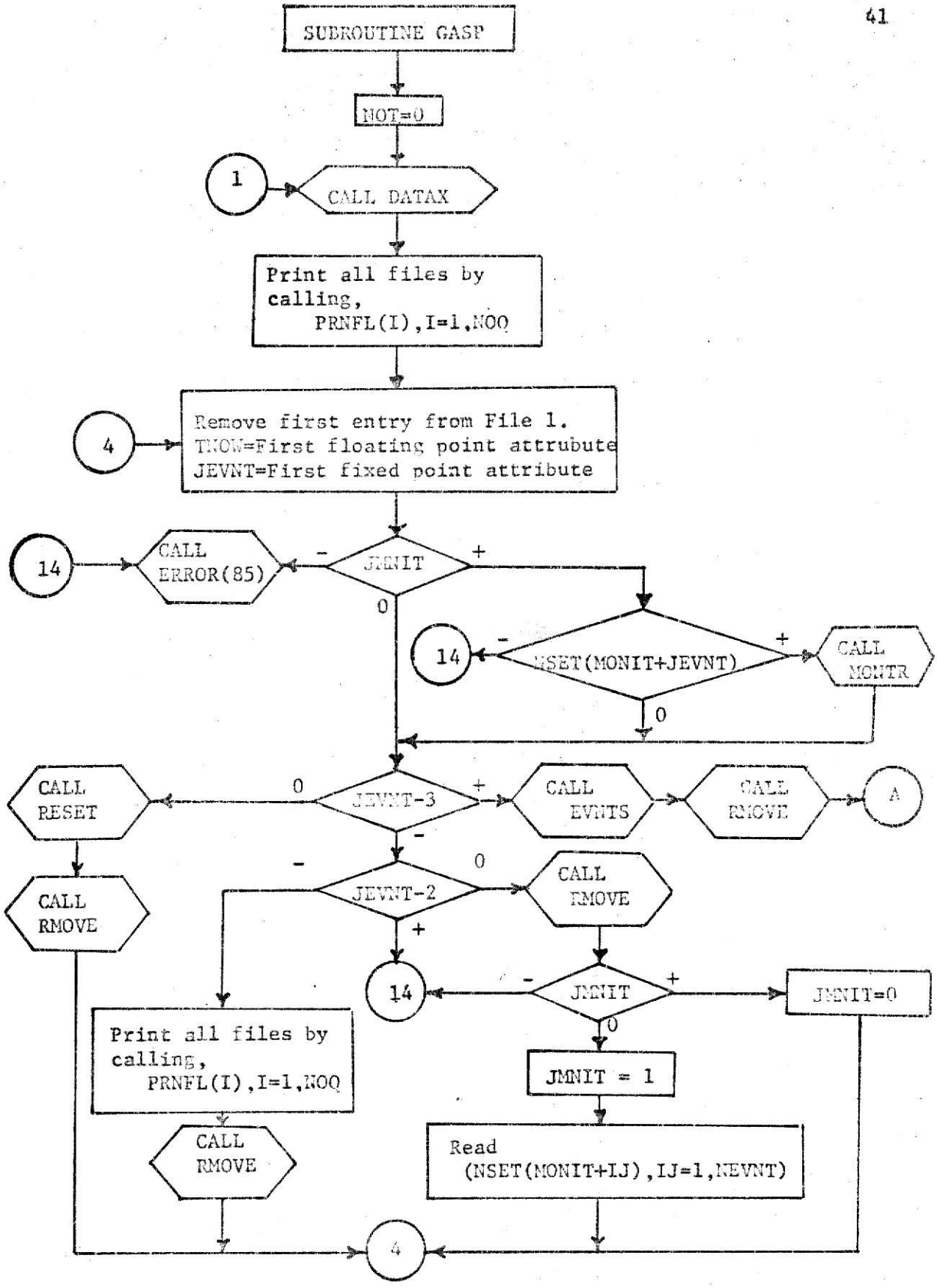
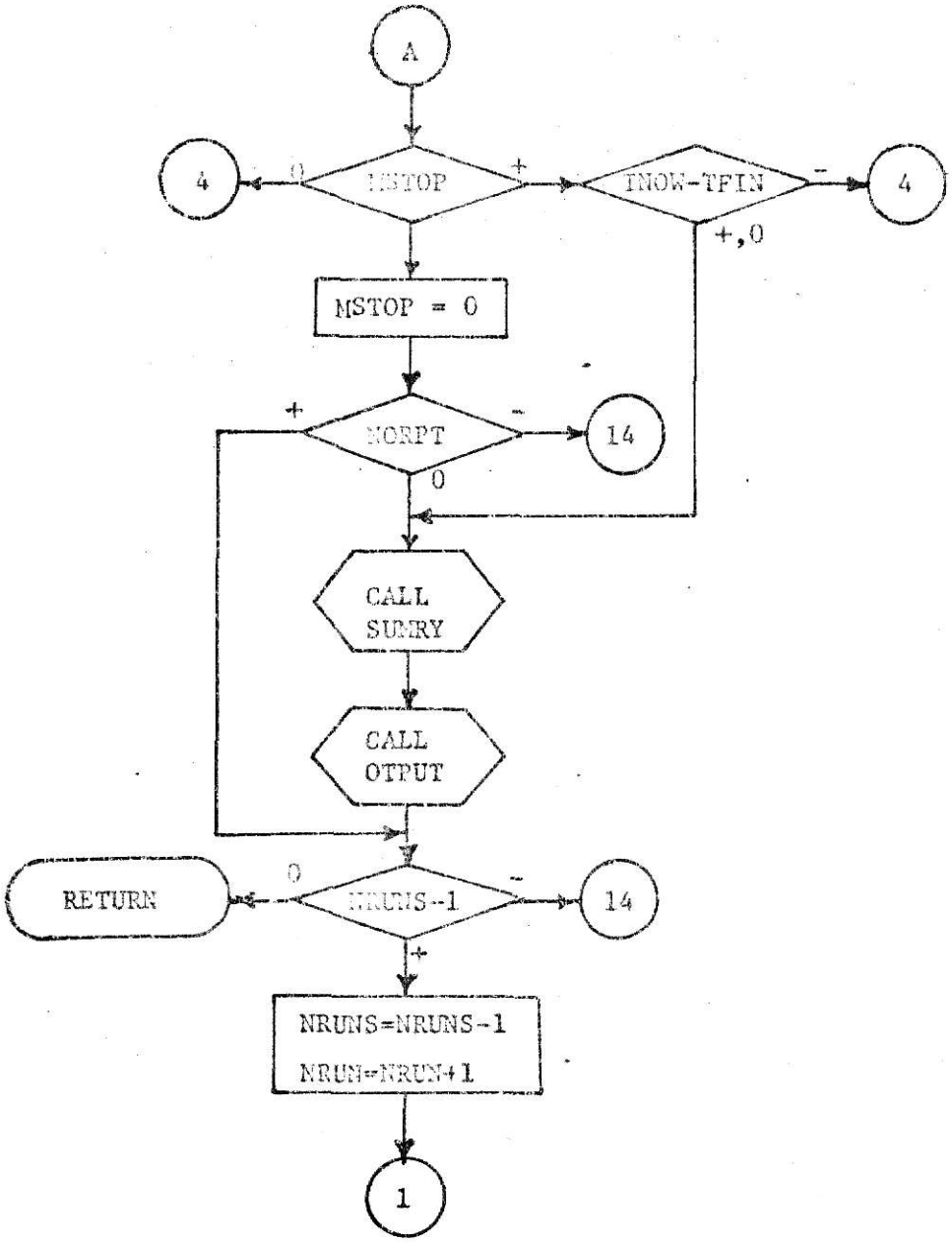


Figure 3.1 Flow Diagram for Subroutine GASP in GASP IIP


A
4
0
MSTOP
+
INOW-TFIN
-
4
+,0
MSTOP = 0
+
NORPT
-
14
0
CALL
SUMRY
CALL
OTPUT
RETURN
0
NRUNS-1
-
14
+
NRUNS=NRUNS-1
NRUN=NRUN+1
1

is one of the three standard events used in GASP IIP. In GASP IIP event codes 1, 2, and 3 denote standard events. All the user written events should be given event codes 4 and above. If the event code is 1, printouts of all the files are given using the ENTRY PRNFL in subroutine PRNTQ. If the event code is 2, the switch JMNIT is flipped. If JMNIT > 0 it is made equal to zero. If JMNIT = 0, then it is made equal to 1 and values of NSET (MONIT + IJ) are read in for all events. If the event code is 3, then the RESET subroutine is called to reset the required set of variables. Detailed information on resetting has been given in Chapter 4. If the event code is greater than 3, the subroutine EVNTS is called. In GASP IIP the event code is not transferred to the subroutine EVNTS as an argument as in GASP IIA. The subroutine EVNTS in GASP IIP, obtains the current event code JEVNT from the GASP blank COMMON. After this point the subroutine GASP in GASP IIP and its counterpart in GASP IIA function in the same manner.

3.1.3 Subroutine DATAX

When a data set comprising the load modules of all the GASP IIP subprograms was created, each member was given a name. For most of the members the member name and the subprogram name were one and the same. The name DATAN was used by one other member in the FORTRAN IV library for the IBM SYSTEM/360 Operating System, which was concatanated to the GASP IIP data set. Since the data set of the FORTRAN IV library appeared before GASP IIP data set in the concatanated data sets, whenever a reference to the name DATAN was made, the subprogram from the FORTRAN IV library was brought in which disrupted the simulation. To avoid this confusion the name of the subprogram DATAN in GASP IIA was changed to DATAX in GASP IIP.

This subprogram has been considerably modified and expanded. Almost double the number of source statements has been used in DATAX as compared to the subprogram DATAN in GASP IIA. The basic functions of this subroutine still remain the same. Three additional card types have been introduced in DATAX bringing the total to eleven.

Card type 1 is the same in both GASP IIP and GASP IIA. Card type 2 is a little different in GASP IIP. Three of the variables - IM, IMM, and MXC - appearing in GASP IIA card type 2 have been omitted in GASP IIP. IM and IMM have been replaced by MXX in GASP IIP. MXX specifies the length of a unit filing area or column in the GASP IIP filing array. Another variable MXC which gives the largest number of cells that can be used in a histogram, is not used in GASP IIP. The variable NEVNT which specifies the total number of events involved in a particular simulation run has been introduced in GASP IIP and is initialized through data card type 2 in DATAX.

After card type 2, the values of pointers INN, KRANK, MAXNQ, MFE, MLC, MLE, NQ, NVNQ, NENQ, NQTM, MONIT, IM, IMM, NPARM, NSUMA, MCLCT, MTMST, NRSET, MHIST, and JHIST are calculated as described in Section 2.2.

Data card type 3 has been added to support the filing and information retrieval procedures followed in GASP IIP. Every event in File 1 uses a different type of entry and also every file other than File 1 has a different type of entry. Hence, the total number of different entry types in a simulation run is given by the expression (NEVNT + NOQ - 1). Through data card type 3 the number of real and integer attributes for each entry type are specified. The segment following the pointer IM in GASP IIP array is (NEVNT + NOQ - 1) locations long and stores the number of integer attributes in each entry type. Similarly, the segment following the pointer IMM stores the number of real attributes in each entry type.

GASP IIP data card types 4, 5, 6, 7, and 8 are equivalent to the data card types 3, 4, 5, 6, and 7 in GASP IIA. After data card type 8 the values of pointers IX and NFIL are calculated.

Through data card type 9 the seed values for the random number generators are read in. In GASP IIA this was done after data card type 7. Creating a new data card type at this point in the program will provide facilities to manipulate just the seeds and not the values of other variables that precede the seeds.

Data card type 10 has been newly introduced in GASP IIP. This card type is used to specify the resetting times of the variables that use the subroutines COLCT, TMST, and HISTO. These values are used by the subroutine RESET. The variables QSET (MCLCT + I), I = 1, NCLCT, QSET (MTMST + I), I = 1, NSTAT and QSET (MHIST + I), I = 1, NHIST are initialized through this card type.

After data card type 10, the values of all pointers are printed. Then if it is the beginning of a simulation run, the value of MFA is set equal to 1.

Data card type 11, is used to read in the initial entries. At least two data cards will be used to file one entry. If the entry is going into File 1, the first data card carries the file number and the event code. If the entry is going into any other file it is enought that the first data card carries just the file number. The first data card is used to tell the program the number of integer and real attributes in the entry to be read, so that the attributes can be referred to with appropriate names i.e., integer numbers will be referred to as NSET and real point numbers will be referred to as QSET. If the entry belongs to File 1, then the event code

is used to determine the number of integer and real attributes in the entry filed. For all files other than File 1, the entry type is determined by the file number. The attributes are read in using G - FORMAT. The second card starts the string of attributes for the entry filed. The attribute field width is ten columns and if there are more than eight attributes in an entry, additional cards may be added as needed. The integer attributes are read first and then the real attributes. If the entry is going into File 1, the event code is punched as the first attribute in the second data card. Once the number of integer and real attributes in the entry to be filed have been determined, the attributes are read directly into the filing array. After each entry has been read into the filing array, the pointers are adjusted using the ENTRY FILEM in the subroutine SET.

After all the entries have been read, the statistical storage arrays are initialized as in GASP IIA. Figure 3.2 illustrates the GASP IIP input data sheet.

3.1.4 Subroutine SET in GASP IIP

In this subprogram in addition to the normal entry point (SET) two other nonstandard entry points (FILEM, RMOVE) have been provided. In the beginning of a simulation run the subroutine DATAX calls the subroutine SET to set up the pointers in the filing array and to initialize some GASP variables and some segments in the GASP IIP array. The use of two other entry points in this subprogram has eliminated the need for the control variable INIT that was previously used in the subprogram SET of GASP IIA. In GASP IIP the entire filing array is not initialized in the beginning of a simulation run. The GASP IIP filing array can be considered to be made up of a finite number of unit filing areas or columns. The last two

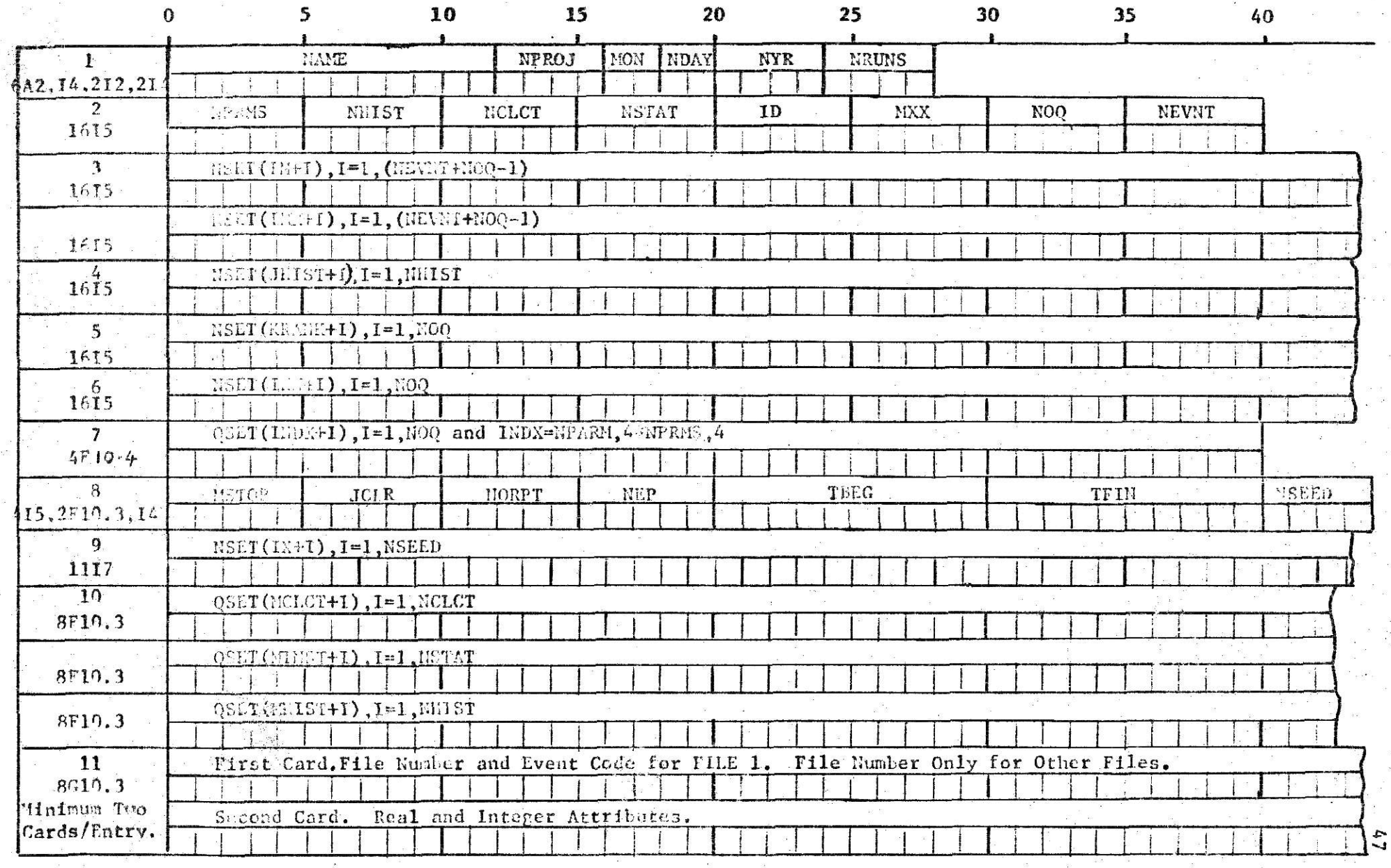


Figure 3.2 GASP IIP Data Sheet.

locations in each column is used to accommodate the 'successor' and 'predecessor' pointers. In GASP IIP, in the beginning of a simulation run only these pointers are initialized by the subroutine SET and the rest of the filing array is not initialized to zeroes. Whenever an entry is removed only the pointers at the end of the column that accommodated the entry are adjusted. The rest of the locations in that column are not set to zero. This is possible in GASP IIP because the types of entries are classified and the number of integer and real attributes in each type of entry is specified. Since the use of buffer arrays ATRIB and JTRIB, has been eliminated in GASP IIP, most of the functions of the subroutines FILEM and RMOVE have also been eliminated. This has made it possible to make FILEM and RMOVE as two nonstandard entry points in the subroutine SET.

3.2 Filing and Retrieving of Entries in GASP IIP

3.2.1 Role Played by FUNCTION LOCAT (NCODE, K, JATT)

This subprogram enables the user to relate a column number to a cell and attribute number. The first argument NCODE is used to specify code numbers. Unlike the function LOCAT in GASP IIA which uses a set of four code numbers, two for QSET and two for NSET, the function LOCAT in GASP IIP uses only two code numbers, since, NSET and QSET refer to the same one-dimensional array.

If NCODE = 1, the function returns the cell number, for the column and attribute numbers specified through K and JATT respectively. If NCODE = 2, it returns the column number, for the cell and attribute numbers specified through K and JATT. Hence, either a cell number or a column number is specified through K depending upon the value of NCODE. The statement INDX = LOCAT (1, 10, 5) gives the cell number the fifth attribute in the tenth column of the filing array. Similarly, the statement KCOL = LOCAT (2, 10, 5) is

used to obtain the column number of an entry, given that the fifth attribute of that entry is in the tenth cell.

In GASP IIP when an entry is to be filed, the user places the attributes of that entry directly in the first available column MFA. To do this he needs the number of the first cell in MFA which he obtains as:

INDX = LOCAT (1, MFA, 1)

In this manner, the function LOCAT is extensively used in the GASP IIP user written programs for filing entries.

3.2.2 An Example with Snapshots of the Filing Array

The procedure followed for filing and retriving of entries are explained here with snapshots of the GASP IIP filing array. The values of some of the GASP IIP variables used in this example are as follow:

Variables Initialised in DATAX and Used for the Filing and Retrieval of Entries

1. Number of files: NOQ = 2
2. Maximum number of entries in file: ID = 5
3. Length of a unit filing area: MXX = 5
4. Number of events: NEVNT = 2
5. Attributes in each type of entry:

Entry type: I	1	2	3
Real attributes: NSET (IR + I):	2	1	1
Integer attributes: NSET (IMM + I):	1	1	2

6. First file is LVF (Low Value First) NSET (INN + 1) = 1
   based on first real attribute NSET (KRANK + 1) = 1
   Second file is HVF (High Value First) NSET (INN + 2) = 2
   based on first integer attribute NSET (KRANK + 2) = 101

Variables Initialized in Subroutine SET

7. Standard flags:

   End of file indicator: KOL = 7777

   End of filing array indicator: KOF = 8888

   Beginning of file indicator: KLE = 9999

8. Maximum size of filing array: MXX * ID = 5 x 5

   = 25

(NOTE: In GASP IIA filing array requirement to handle this problem will be:

1. Floating point attributes/entry: IMM = 2
2. Fixed point attributes/entry: IM = 2
3. Predecessor pointer row: MXX = IM + 2 = 4
4. Maximum size of QSET: IDM * ID = 10
5. Maximum size of NSET: MXX * ID = 20

Hence, the total filing area requirement in GASP IIA is 30 cells as compared to 25 cells in GASP IIP)

Figure 3.3 (a) is a snapshot of the filing array immediately after initialization. It should be noted that only the pointers have been set in the last two rows. The rest of the filing array has not been initialized and carries undefined quantities. Column 1 is the first available column for filing an entry (MFA = 1). Since, this column has no predecessor, its predecessor pointer would normally be 9999. For debugging purposes this value is taken as zero immediately after initialization. Thus, if the filing array does not contain at least one entry with the flag value 9999, it indicates that no entries were ever inserted into the filing array.

Next an entry is filed into File 1 with an event code 1. As specified before this entry has two integer attributes and one real attribute. The

COLUMNS

ROWS	ID 1	2	3	4	5	
1	U	U	U	U	U	
2	U	U	U	U	U	
3	U	U	U	U	U	
4	2	3	4	5	8888	Successor Row
MXX=5	0	1	2	3	4	Predecessor Row

U = Undefined Values

OPERATION:

Initialization.

CALL SET

MFA = 1

File 1	MFE(1) = 0	MLE(1) = 0	NQ(1) = 0
File 2	MFE(2) = 0	MLE(2) = 0	NQ(2) = 0

Figure 3.3 (a) Sanpshot of the GASP IIP Filing Array after Initialization

COLUMNS

Rows	ID 1	2	3	4	5	
1	1	U	U	U	U	
2	2	U	U	U	U	
3	1.5	U	U	U	U	
4	7777	3	4	5	8888	Successor Row
MXX=5	9999	9999	2	3	5	Predecessor Row

OPERATION:

Insert an Entry into File 1.

```
INDX=LOCAT(1,MFA,1)
NSET(INDX)=1
NSET(INDX+1)=2
QSET(INDX+2)=1.5
CALL FILEM(1)
```

MFA = 2

File 1	MFE(1) = 1	MLE(1) = 1	NQ(1) = 1
File 2	MFE(2) = 0	MLE(2) = 0	NQ(2) = 0

Figure 3.3 (b) Snapshot of the GASP IIP Filing Array after Insertion of an Entry Into File 1.

statement INDX = LOCAT (1, MFA, 1) is used to obtain the number of the first cell in the first available column MFA. As seen in Figure 3.3 (b) first the fixed point attributes are stored and then the floating point attributes. Since, this is the only entry in File 1 it has no successor or predecessor. Hence, its successor and predecessor pointer values are 7777 and 9999 respectively. The first as well as the last entry in File 1 is in column 1. Hence MFE (1) = 1, MLE (1) = 1, and NQ (1) = 1, and the first available column is column 2 i.e. MFA = 2.

Next an entry belonging to File 2 is inserted in column 2 as shown in Figure 3.3 (c). This entry has one integer and two real attributes. Since this is the only entry in File 2 it has no successor or predecessor.

Figure 3.3 (d) shows the insertion of another entry into File 1. With event code 2. This entry has only two attributes and does not use the third cell in column 3. The entries in File 1, which is a LVF file, are ranked according to the first real attribute. Since the entry in column 3 has a ranking attribute 0.5 as compared to the ranking attribute value 1.5 of entry in column 1, the entry in column 3 becomes the first entry in File 1 and the entry in column 3 becomes its successor.

A second entry is inserted into File 2 and it occupies column 4 as shown in Figure 3.3 (e). File 2 is a HVf file ranked according to the first integer attribute. Since the ranking attribute of the entry in column 4 has a higher value than that of the entry in column 2, the entry in column 4 becomes the first entry in File 2.

Figure 3.3 (f) shows the last column of the filing array being occupied by an entry belonging to File 1. Since its ranking attribute has a value of 0.6 which is in between the ranking attribute values of the entries in columns 1 and 3, the new entry becomes the middle entry in File 1.

COLUMNS

ROWS	ID 1	2	3	4	5	
1	1	1	U	U	U	
2	2	2.5	U	U	U	
3	1.5	3.5	U	U	U	
4	7777	7777	4	5	8888	Successor Row
MXX=5	9999	9999	9999	3	4	Predecessor Row

OPERATION:

Insert an Entry into File 2.

```
INDX=LOCAT(1,MFA,1)
NSET(INDX)=1
QSET(INDX+1)=2.5
QSET(INDX+2)=3.5
CALL FILEM(2)
```

MFA=3

File 1	MFE(1)=1	MLE(1)=1	NQ(1)=1
File 2	MFE(2)=2	MLE(2)=2	NQ(2)=1

Figure 3.3 (c) Snapshot of the GASP IIP Filing Array after Insertion of an Entry Into File 2.

COLUMNS

ROWS	ID 1	2	3	4	5	
1	1	1	2	U	U	
2	2	2.5	0.5	U	U	
3	1.5	3.5	U	U	U	
4	7777	7777	1	5	8888	Successor Row
MXX=5	3	9999	9999	9999	4	Predecessor Row

OPERATION:

Insert an Entry into File 1.

```
INDX=LOCAT(1,MFA,1)
NSET(INDX)=2
QSET(INDX+1)=0.5
CALL FILEM(1)
```

MFA=4

File 1	MFE(1) = 3	MLE(1) = 1	NQ(1) = 2
File 2	MFE(2) = 2	MLE(2) = 2	NQ(1) = 1

Figure 3.3 (d) Snapshot of the GASP IIP Filing Array After the Insertion of a Second Entry Into File 1 (LVF).

COLUMNS

ROWS	ID 1	2	3	4	5	
1	1	1	2	2	U	
2	2	2.5	0.5	1.0	U	
3	1.5	3.5	U	2.0	U	
4	7777	7777	1	2	3888	Successor Row
MXX=5	3	4	9999	9999	9999	Predecessor Row

OPERATION:

Insert an Entry Into File 2.

INDXE=LOCAT(1,MFA,1)

NSET(INDX)=2

QSET(INDX+1)=1.0

QSET(INDX+2)=2.0

CALL FILEM(2)

MFA=5

File 1	MFE(1)=3	MLE(1)=1	NQ(1)=2
File 2	MFE(2)=4	MLE(2)=2	NQ(2)=2

Figure 3.3 (e) Snapshot of the GASP IIP Filing Array After Insertion of a Second Entry Into File 2 (HVF).

COLUMNS

ROWS	ID 1	2	3	4	5	
1	1	1	2	2	2	
2	2	2.5	0.5	1.0	0.6	
3	1.5	3.5	U	2.0	U	
4	7777	7777	5	2	1	Successor Row
MXX=5	5	4	9999	9999	3	Predecessor Row

OPERATION:

Insert an Entry Into File 1.

INDX=LOCAT(1,MFA,1)

NSET(INDX)=2

QSET(INDX+1)=0.6

CALL FILEM(1)

MFA=8888

File 1	MFE(1)=3	MLE(1)=1	NQ(1)=3
File 2	MFE(2)=4	MLE(2)=2	NQ(2)=2

Figure 3.3 (f) Snapshot of the GASP IIP Filing Array After Insertion of an Entry Between Two Entries. No More Columns Available for Storing Entries.

Figure 3.3 (g) shows the removal of the entry in column 4. This entry was the first of the two entries in File 2. Now, the entry in column 2 becomes the first as well as the last entry in File 2 and its pointer values are modified accordingly. It should be noted that when the entry is removed from column 4 only the pointers are modified and the other cells in that column are not reinitialized to zeroes.

Next an entry belonging to File 1 is inserted into the column just vacated. It should be noted in Figure 3.3 (h), that the two attributes of this entry occupy the first two cells of column 4. The number 2.0 in the third cell (circled element in Figure 3.3 (h)) is leftover from the previous entry and does not belong to the entry that currently occupies column 4.

3.3 Improved Facilities for Monitoring

Selective Monitoring. The subroute MONTR in GASP IIP has been considerably modified and it no longer gives a printout of the entire filing array as it did in GASP IIA. In GASP IIP subroutine MONTR is used only to monitor selected events between two given time points during a simulation run. Changes have been made in subroutine GASP for monitoring selected events whenever needed. In GASP IIA once the monitoring is triggered all events will be monitored which in many cases may not be necessary. The switch JMNIT controls the monitoring of events. Only when JMNIT is 'ON' (i.e. JMNIT is positive) the monitoring takes place. An event filed with a standard event code 2 flips the JMNIT switch i.e., if JMNIT is 'ON' it is switched 'OFF' and if it is 'OFF' it is switched 'ON'. Whenever JMNIT is switched on information has to be supplied to the program as to which events are to be monitored. This is specified through a data

COLUMNS

ID

ROWS		1	2	3	4	5	
	1	1	1	2	2	2	
	2	2	2.5	0.5	3.0	0.6	
	3	1.5	3.5	U	2.0	U	
	4	7777	7777	5	8888	1	Successor Row
MXX=5		5	9999	9999	9999	3	Predecessor Row

OPERATION:

Remove an Entry From File 1.

CALL RMOVE (4,2)

MFA=4

File 1	MFE(1)=3	MLE(1)=1	NQ(1)=3
File 2	MFE(2)=2	MLE(2)=2	NQ(2)=1

Figure 3.3 (g) Snapshot of GASP IIP Filing Array After an Entry Has Been Removed From File 1.

COLUMNS

ID

ROWS		1	2	3	4	5	
	1	1	1	2	2	2	
	2	2	2.5	0.5	2.5	0.6	
	3	1.5	3.5	U	2.0	U	
	4	4	7777	5	7777	1	Successor Row
MXX=5		5	9999	9999	1	3	Predecessor Row

OPERATION:

INDX=LOCAT(1,MFA,1)

NSET(INDX)=2

QSET(INDX+1)=2.5

CALL FILEM(1)

MFA=8888

File 1	MFE(1)=3	MLE(1)=4	NQ(1)=4
File 2	MFE(2)=2	MLE(2)=2	NQ(1)=1

Figure 3.3 (h) Snapshot of GASP IIP Filing Array After Insertion of a Fourth Entry Into File 1. Leftover Attribute in the Middle of the Entry.

card. Starting from the first column, as many columns as there are events - including the standard events 1, 2, and 3 - , are filled up either with a '1' or with a '0'. The variable NEVNT denotes the number of events used in a simulation run. A '1' in the first column will cause the event with an event code 1 to be monitored. Similarly, a '0' in the second column will supress the monitoring of the event with event code 2. The information thus supplied through a data card will be stored in a segment of the GASP IIP array, indicated by the pointer MONIT. If it is desired to start the monitoring of events at time T1, an event with code 2 should be schedules take place at that time. A data card that indicates the events to be monitored should be supplied at this time. If the monitoring is to be stopped at time T2 (T2 > T1), then, another event with event code 2 and event time T2 should be filed.

Difference Between GASP IIA and GASP IIP Monitoring. The monitoring procedure followed in GASP IIA does not facilitate selective monitoring of events. In GASP IIA, monitoring of an event takes place after that event has been executed. The information printed out by the subroutine MONTR consists of the event code and event time for the current event and all the attributes of the next event. Let us assume that a simulation deals with three types of events E1, E2, and E3 and between the time points T1 and T2 these events occur as shown in Figure 3.4. If we decide not to monitor events of type E2, then the monitoring method followed in GASP IIA will results in loss of information about events that immediately follow an E2 event. In the above example information about events that have been circled in Figure 3.4 will be lost. To avoid this the subroutine MONTR in GASP IIP prints out all the attributes of the current event and not those of the

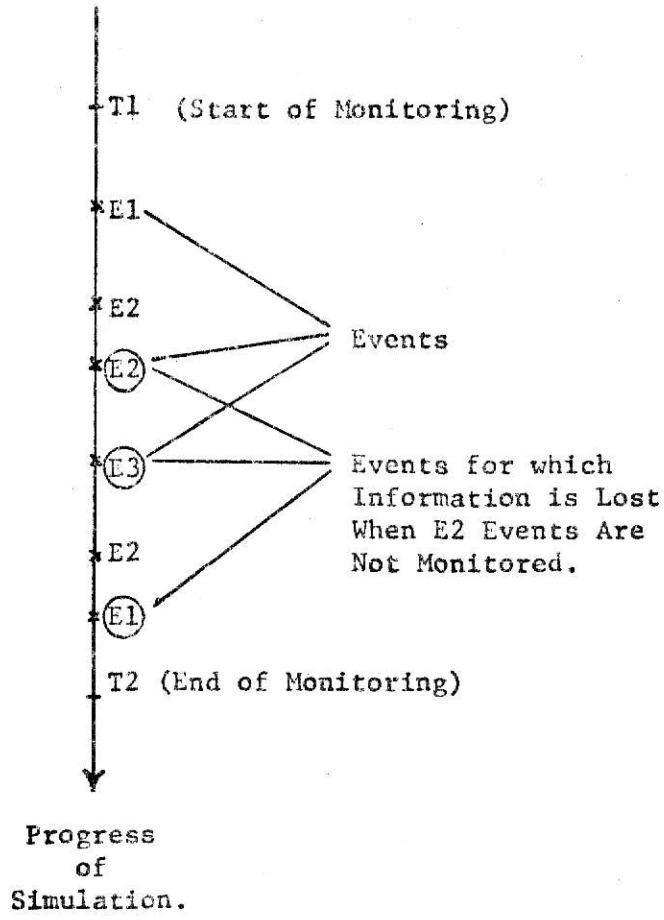


Figure 3.4 Effect of Selective Monitoring In GASP IIA.

next event. Further, in GASP IIP an event is monitored before its execution so as to facilitate effective debugging in case something goes wrong during the execution of that event.

Since the subroutine MONTR in GASP IIP is not used anymore to obtain the printouts of the filing array, its present size is about half of its previous size in GASP IIA. The ENTRY PRNFL in the subroutine PRNTQ is used to obtain file printouts whenever needed.

3.4 Improved Facilities for File Printouts

Need for Improvements. In GASP IIA after the filing array has been set up and initialized using subroutines SET and DATAN, the entire filing array is dumped out. Whenever a file dumpout is requested, subroutine MONTR dumps out the entire filing array, both the used and unused part of it. This type of printout involves large amount of I/O, especially when the filing array used is large. It is not easy to sort out the entries of individual files from this kind of printout.

GASP IIP File Printouts. In GASP IIP, if a file printout is needed at time T1, then an event with event code 1 and event time T1 is filed. Whenever an event with the standard event code 1 is encountered, GASP IIP gives printouts of individual files using the ENTRY PRNFL in subroutine PRNTQ. In this method, only that part of the filing array which is currently being used, is printed out. When an entry is dumped, only that part of the column which contains values relevant to that entry, is printed. In Figure 3.3 (h), columns 1, 3, 4, and 5 contain entries belonging to File 1. Let us assume that File 1 is being printed using ENTRY PRNFL in subroutine PRNTQ. For the entry in column 1, the values in all five cells of that column will be printed. But for the entries in columns 3, 4, and 5 the values in the third cell in each of these column will not be printed out,

because these values do not belong to the entries in those columns. In GASP IIP, this kind of selective printout is made possible by the information read in, through card type 3 in subroutine DATAX, stating the number of integer and real attributes for each type of entry. This modified file printout procedure followed in GASP IIP should reduce the I/O operations considerably.

CHAPTER 4

IMPROVED FACILITIES TO AID STOCHASTIC SIMULATION

4.1 New Facilities for Resetting

4.1.1 Transient and Steady States

Before we start collecting observations generated by a simulator we should know whether we are going to study the transient or the steady state behavior of the system. In simulation, most often the starting conditions will be atypical of the system's eventual operating conditions. The gradual change from the starting to the operating conditions is known as the transient state. Emshoff and Sisson (2, p. 190) define a system as having reached steady state if successive observations of the system's performance are statistically indistinguishable. If (x_i) is one of the state variables of interest in the simulated system, then when the steady state conditions prevail, $E(x_i)$ will be independent of time. Many simulation models are built and operated under the implicit assumption that they are capable of steady state behavior. In such cases if the starting conditions are atypical of the operating conditions, the model will exhibit transient behavior in the beginning as shown in Figure 4.1. If the objective is to study the system's steady state bahavior, the effects of transients on the data generated by the simulator must be removed; otherwise the conclusions drawn from the analysis of such data will be biased.

Emshoff and Sisson (2, p. 191) list two methods commonly used to remove the effects of transients. One method is to use long simulation runs so that the transient data is insignificant relative to the steady state data. This method is expensive, requiring long simulation runs.

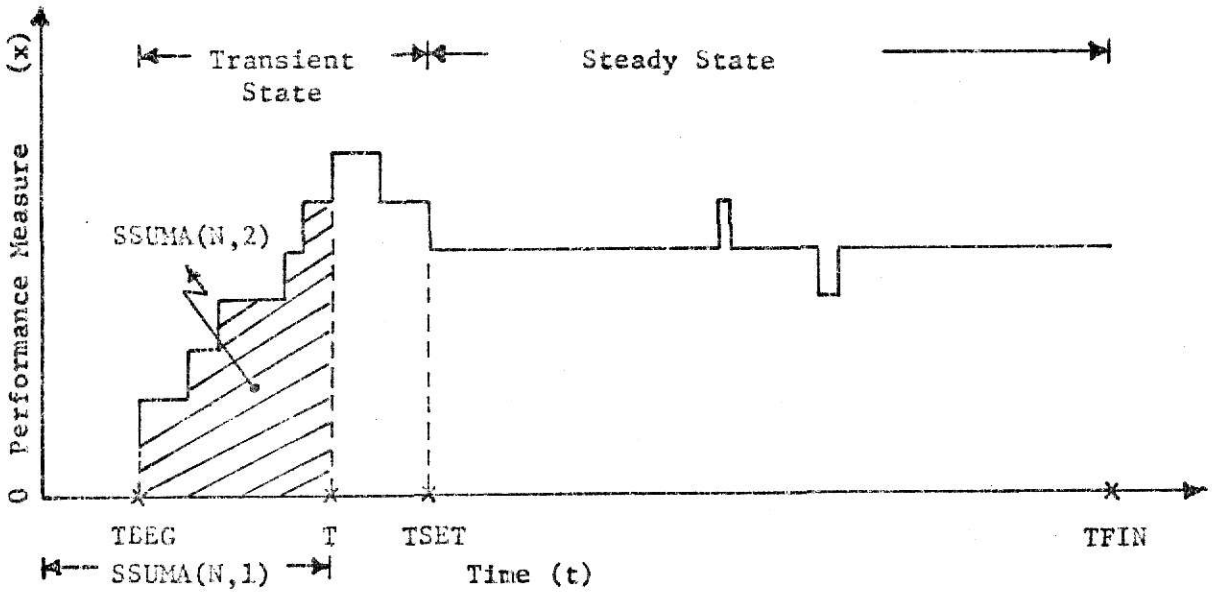


Figure 4.1 Transient and Steady States and Calculation of Time Weighted Statistics.

The second method is to run the simulator until steady state conditions are reached, then clear all statistical accumulations, but leave the state of the simulated system as it is. The conditions established at the end of the transient period are an estimate of the steady state operating conditions. Taking these as the initial conditions the simulation run is continued. This is called resetting.

4.1.2 Resetting in GASP IIA

Time Weighted Statistics. In GASP IIA the subroutine TMST is used to calculate time weighted statistics. Suppose X is one of the performance measures in the simulated system behaving as shown in Figure 4.1 over the period of simulation. We shall assume that the simulation starting time TBEG > 0 and that the variable X uses TMST with the variable code N. Suppose with in the time between TBEG and T, the variable X has assumed n different values X_1, X_2, X_3, $\cdots$, X_n. Whenever the variable X changes its state, TMST is called. The time weighted average of X can be calculated as:

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i \cdot \Delta t_i}{\sum_{i=1}^{n} \Delta t_i}$$

Here, Δt_i is the time interval for which the variable X has the value X_i. In TMST the time weighted summation $\sum_{i=1}^{n} X_i \cdot \Delta t_i$ is given by the variable SSUMA (N,2). This is equal to the shaded area in Figure 4.1. The summation of time intervals is indirectly calculated through SSUMA (N,1). At the start of the simulation SSUMA (N,1) is set equal to TBEG in subroutine DATAX. Its value is calculated by adding the time increments as:

$$\text{SSUMA } (N,1) = \text{TBEG} + \sum_{i=1}^{n} \Delta t_i$$

To get the time weighted average, the shaded area SSUMA (N, 2) in Figure 4.1 should be divided by the corresponding abscissa (SSUMA (N,1) - TBEG). Dividing SSUMA (N,2) just by SSUMA (N,1) as indicated by Pritsker and Kiviat (1, p. 318) will give a biased estimate for the mean if TBEG > 0. Similar correction should be applied while calculating the standard deviation also.

Resetting Operations. In GASP IIA, one way of resetting is to file a reset event to occur at the reset time TSET. When this event occurs the user calls the subroutine DATAN with NEP = 9. This clears the statistical accumulations and reinitializes the statistical storage arrays. Resetting is equivalent to starting a new simulation run at time TSET, with the state of the simulator at the end of the transient state providing the starting conditions. Hence, when the simulator is reset, SSUMA (N,1) is set equal to TSET. From now on to get time weighted statistics we should use (SSUMA (N,1) - TSET).

Need for New Facilities. In GASP IIA, there are no built-in facilities for these resetting operations. The user must provide all coding needed. Further, in GASP IIA all the variables are reset at the same time regardless of our needs. We would like to have the flexibility to reset only a selected set of variables. Also, we should be able to reset different variables at different times. To enable this and also to relieve the user from the task of providing the reset coding, we have incorporated new facilities in GASP IIP.

4.1.3 Resetting in GASP IIP

Reset Time and Reset Event. In GASP IIP, card type 10 in the subroutine DATAX is used to read in the reset times for all variables. The segments in the GASP IIP array indicated by the pointers MCLCT, MTMST and MHIST are used to store the reset times for the variables that use the subprograms COLCT, TMST and HISTO respectively. Suppose the variables to be reset can be grouped into three sets such that one set is to be reset at time T1, another set at T2, and the third set at T3, where T1, T2 and T3 > 0. Then three different reset events should be scheduled at event times T1, T2 and T3. In GASP IIP reset events are given the standard event code 3. Whenever an event with code 3 comes up the subroutine GASP calls the subroutine RESET. This subroutine has been added in GASP IIP. The reset event should have three integer attributes and one real attribute. The first integer attribute, as usual, is the event code and the real attribute is the event time. If the second attribute is greater than zero, then at the reset-time the subroutine RESET calls DATAX with the NEP value specified through the third integer attribute.

Subroutine Reset. A detailed flow chart for the RESET subroutine is shown in Figure 4.2. If RESET is called at time TNOW, it resets all the variables that are scheduled to be reset at that time. First, the reset-times of the variables that use the subroutines COLCT are tested and all the variables whose reset-times are equal to TNOW are reset. Next the variables using TMST and then the variables using HISTO are tested and reset as specified. The segment in GASP IIP array indicated by the pointer NRSET is used to assist the resetting of variables that use TMST. The cells in this segment are initialized to TBEG in the subroutine SET at the start of the

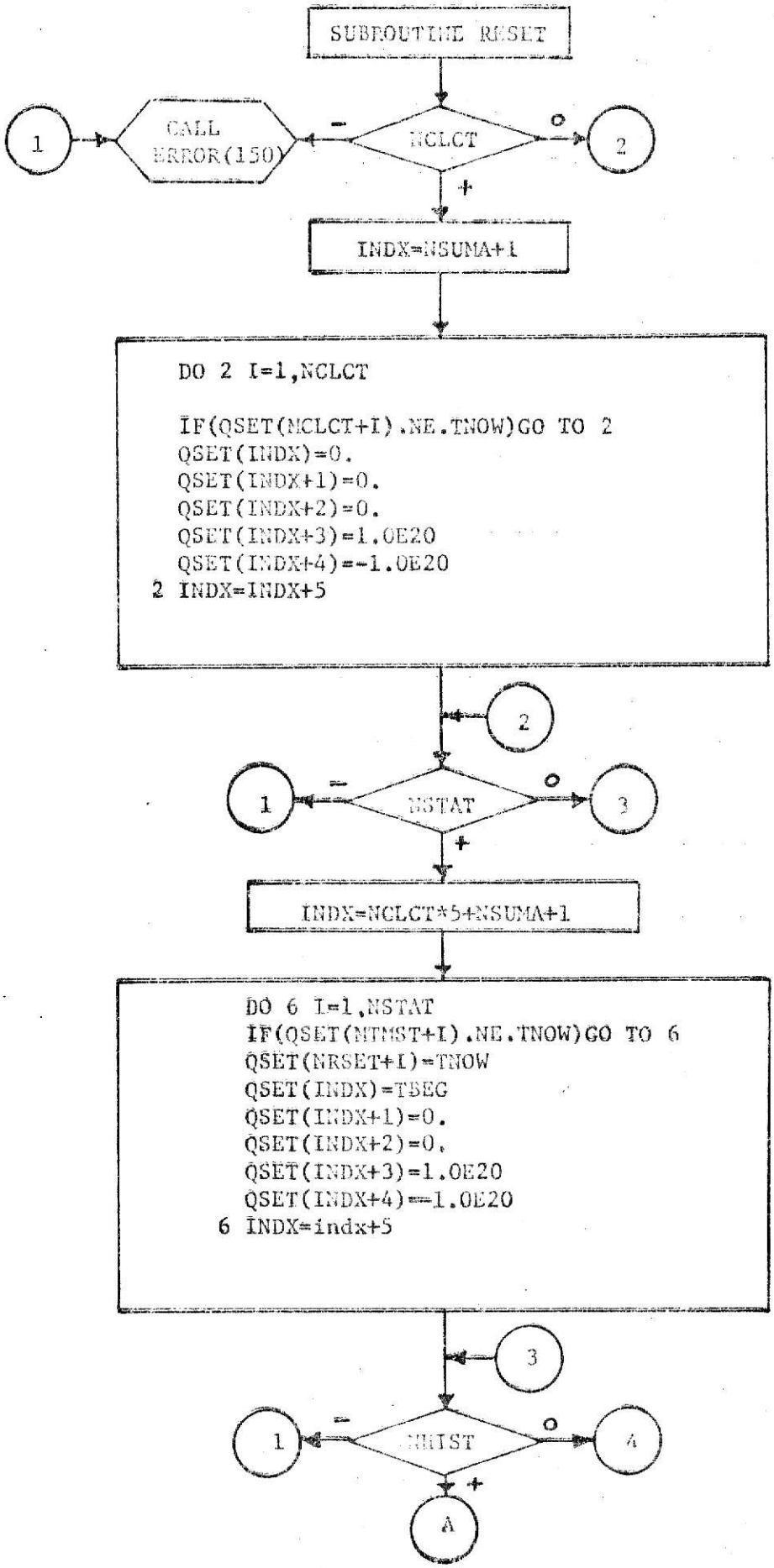
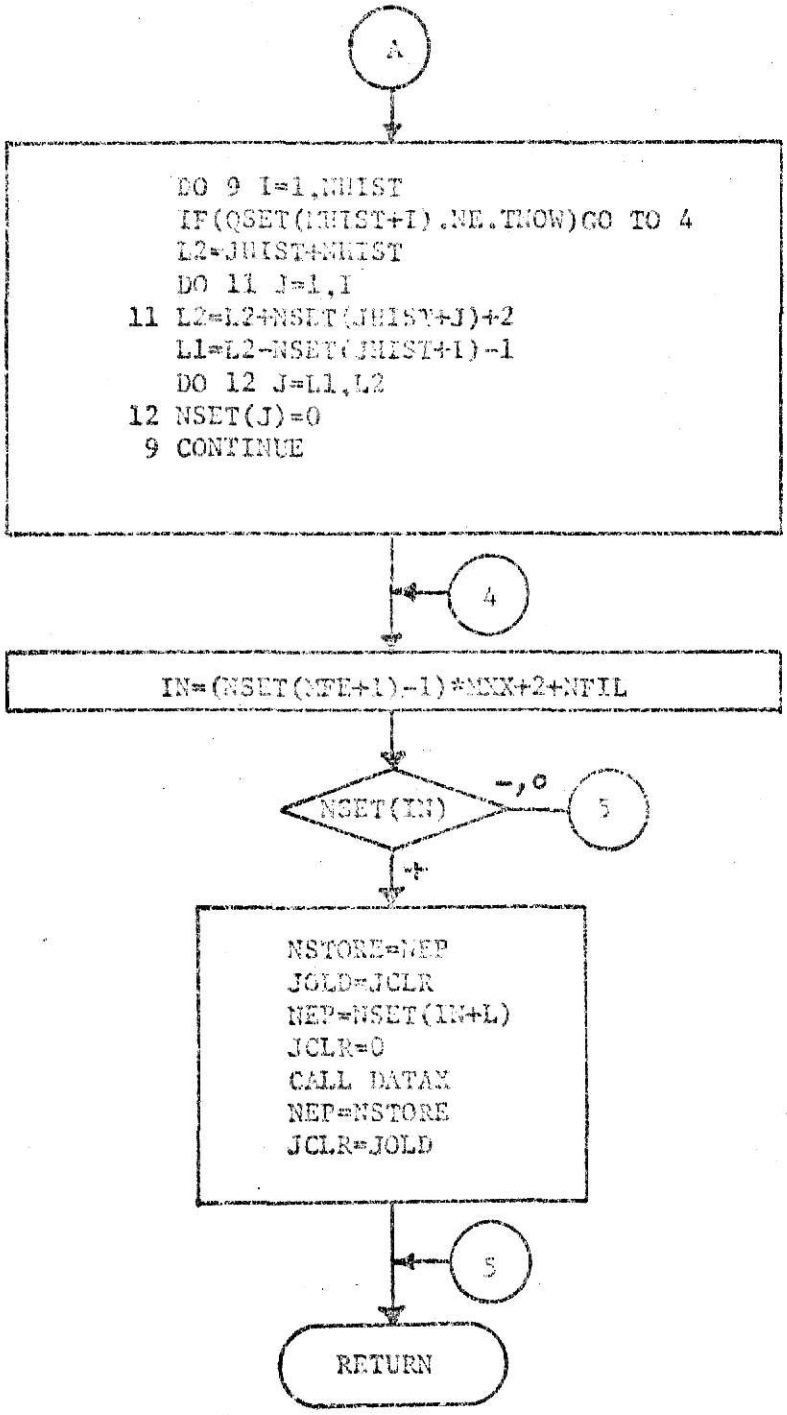


Figure 4.2 Flow Chart of Subroutine RESET.


A
DO 9 I=1,NHIST
IF(QSET(JHIST+I).NE.THOW)GO TO 4
L2=JHIST+NHIST
DO 11 J=1,I
11 L2=L2+NSET(JHIST+J)+2
L1=L2-NSET(JHIST+1)-1
DO 12 J=L1,L2
12 NSET(J)=0
9 CONTINUE
4
IN=(NSET(NFIL+1)-1)*MXX+2+NFIL
NSET(IN)
-,0
5
+
NSTORE=NEP
JOLD=JCLR
NEP=NSET(IN+L)
JCLR=0
CALL DATAX
NEP=NSTORE
JCLR=JOLD
5
RETURN

simulation. When the variable with code 1 is reset QSET (NRSET + 1) is set equal to the reset time of that variable in RESET. Towards the end of this section we shall explain how this value is to be used while calculating time weighted statistics. After the resetting operations are over the second integer attribute of the current reset event is tested. If it is greater than zero, the current values of JCLR and NEP are temporarily stored in JOLD and NSTORE. JCLR assumes a new value of zero, and NEP assumes the value specified through the third integer attribute of the reset event and DATAX is called. This facility has been provided to read in new data in the middle of a simulation run. On return from DATAX, the GASP variables JCLR and NEP are restored to their old values.

Corrections to be Applied. In GASP IIA while computing time weighted statistics for a variable with code 1, the value of SSUMA (I,1) should be corrected as (SSUMA (I,1) - TBEG) before resetting and (SSUMA (I,1) - TSET) after resetting. In GASP IIP this is taken care of by using

(QSET (INDX) - QSET (NRSET + I))

where, INDX = NSUMA + (NCLCT + I - 1) * 5 + 1. QSET (INDX) is the GASP IIP equivalent to SSUMA (I,1). The value of QSET (NRSET + I) will be TBEG before the variable is reset. When the variable is reset QSET (NRSET + I) is set equal to the reset time of that variable in the subroutine RESET. This arrangement enables the required corrections to be applied automatically whether a variable is reset or not.

4.2 New Psuedo-Random Number Generators

4.2.1 Role of the Pseudo-Random Number Generator

The random number generator, FUNCTION DRAND is used either directly or indirectly by every stochastic generator in GASP IIA, as shown below:

CALL

1. UNFRM → DRAND
2. RNORM → DRAND
3. RLOGN → RNORM → DRAND
4. ERLGN → DRAND
5. NPOSN → DRAND
   RNORM → DRAND

The arrows indicate the calling sequence. For instance, the FUNCTION NPOSN calls DRAND directly once and again indirectly through the FUNCTION RNORM. Since the random number generator is the focal point in a stochastic simulation, it should be efficient and produce statistically sound random numbers. In GASP IIA, DRAND is used to generate regular, as well as antithetic random number.

4.2.2 Antithetic Random Numbers

Assume X_i and Y_i, i=1, n represent the performance measures of a system under two different sets of operating conditions. The estimates of the system's average performance under the two different conditions are

$$\hat{\mu}_X = \sum_{i=1}^{n} X_i/n \quad \text{and} \quad \hat{\mu}_Y = \sum_{i=1}^{n} Y_i/n \qquad \text{- - - - - - - EQ 4.1}$$

We shall assume that X_i and Y_i are two sets of independent variables. Then the confidence we can place on the above estimates are indicated by the variances

$$\hat{\sigma}^2_{\hat{\mu}_X} = \frac{\sigma^2_X}{n} \quad \text{and} \quad \hat{\sigma}^2_{\hat{\mu}_Y} = \frac{\sigma^2_Y}{n} \qquad \text{- - - - - - - EQ 4.2}$$

We can compare the system's performance under the two different conditions by comparing the estimates of the average performance, $\hat{\mu}_X$ and $\hat{\mu}_Y$. For this comparison to be meaningful, the variances shown in EQ 4.2 should be small. One obvious way to reduce the sizes of the variances is to increase the sample size n. This means longer simulation runs and more computing expenses. Another method is to replicate the runs and introduce correlation between runs. Suppose we replicate the simulation runs twice with a correlation ρ between replications and observe the performance measures X_i and X_i'. We shall take a sample of size n/2 from each replication. We shall also assume that the performance measures within each replication are independent and have the same variance σ^2.

The estimate of the mean performance for both replications is

$$\hat{\mu} = \sum_{i=1}^{n/2} (X_i + X_i')/n \qquad \text{------- EQ 4.3 (a)}$$

and the variance estimate for this mean is

$$\hat{\sigma}^2_{\hat{\mu}} = (\sigma^2/n)\ (1 + \rho) \qquad \text{------- EQ 4.3 (b)}$$

We can see from EQ 4.3 (b) that by introducing a negative correlation between replications we can get a smaller variance estimate for the mean of the combined observations $(X_1 + X_1')\ (X_2 + X_2')\ \ldots\ldots (X_{\frac{n}{2}} + X'_{\frac{n}{2}})$. The variables produced by introducing negative correlation between pairs are known as antithetic variables. The method commonly used to produce a pair of antithetic simulations is to use a set of regular random numbers

R(0.0 < R < 1.0) to generate probablistic events in one run and to use the corresponding set of antithetic numbers (1.0-R) for the equivalent events in the second run (Emshoff and Sisson (2, p. 197)).

4.2.3 GASP IIA Pseudo-Random Number Generators

Facilities Available in GASP IIA. In GASP IIA modified by Grosh (5), the array IX is used to store the random number seeds. Suppose IX(I) is the seed used for random number generation. If I is even, then the one's compliment of the number generated is taken as the new random number. If I is odd the number is used as it is. Hence, everytime DRAND is called the subscript I is tested to see whether it is odd or even. To avoid this kind of testing we have provided separate generators in GASP IIP for regular and antithetic random numbers. In GASP IIA to make two consecutive simulation runs antithetic, the seeds that are stored in the odd locations of the array IX in the first run should be in even locations in the second run and vice versa. This requires changes in the source codings between replications. In GASP IIP we have added new facilities that enables the user to produce a pair of antithetic simulations without changing the source codings between replications.

Faster Generators. Our search for a faster generator led us to investigate a BAL generator proposed by Lewis, Goodman, and Miller (6). These generators use the full capacity of the 32-bit registers of the IBM SYSTEM/360 computers. The sequence of numbers (N_i) produced by them is based on the equation

$$N_{i+1} = N_i \quad A \pmod{p}$$

Where p is the prime $2^{31}-1$ and A is a positive primitive root of p. The value of A used in GASP IIP generators is 7^5. The integer random numbers generated are converted to real numbers as:

$$R_i = (N_i/2^{31}) + 2^{30}$$

This operation converts the binary number in a 32-bit register to a positive floating point number between 0 and 1.

Designed to Help Antithetic Simulation. To facilitate the simultaneous use of the regular as well as the antithetic simulators and to make two simulations antithetic without changing the source statements we have provided four random number generators with different member names. The member names are the names of the load modules that constitute the data set. Table 4.1 gives the member and subprogram names of these generators. One pair of generators has essentially the same source codings but different subprogram names and generates regular random numbers (R). The other pair with the same source codings but different subprogram names generates antithetic random numbers (1-R). Out of these four generators we can choose a maximum of two at a time by specifying their member names in a INCLUDE statement as:

```
//LKED. SYSIN DD *
 INCLUDE SYSLIB (NAME1, NAME2)
/*
```

The JCL statements given above shall be used right after the GASP data set has been linked to the user written programs. Care should be taken so that two generators with the same subprogram name are not chosen. Suppose

	Function	Subprogram Name	Member (or Load Module) Name
1.	To generate regular random members (R).	DRAND	DREG
2.	- do -	ARAND	AREG
3.	To generate antithetic random numbers (1-R).	DRAND	DANTI
4.	- do -	ARAND	AANTI

Table 4.1 Member (or Load Module) and Subprogram Names of GASP IIP Random Number Generators.

in the first run we have chosen DREG and AREG for use. To make the second run antithetic to the first we should choose DANTI and AANTI for the second run.

Some More Design Characteristics. The RAL generators in GASP IIP have access to the blank COMMON. The segment of the GASP IIP array indicated by the pointer IX is used to store the seeds for the random number generators. The random number generators can be invoked by statements such as:

R1 = DRAND (I)

If DREG is the name specified in the INCLUDE statement, then DRAND refers to the regular random number generator. The subscript I refers to the I^{th} cell in the GASP IIP segment that contains the seed to generate R1.

EXTERNAL Statements. Since we have two names for the generators, DRAND and ARAND, we must tell the stochastic generators which one to use. To do this the name of the generator (DRAND or ARAND) to be used is passed as an argument to the stochastic generators. The subprogram names that are passed as arguments should be declared EXTERNAL in the calling programs. All the stochastic generators have been modified to conform to this arrangement. The use of the EXTERNAL statement is illustrated in Appendix A.2. The subprogram names DRAND and ARAND have been declared EXTERNAL in the MAIN program. These names are received by the subroutine NPOSN through the argument ANAME. ANAME has been declared EXTERNAL in NPOSN since it is passed as an argument to RNORM.

4.2.4 Comparison of Speed and Statistical Soundness

Test programs have been run to compare the speed of the pseudo-random number generators used in GASP IIA and GASP IIP. These test programs are shown in Appendix A.3. Do loops have been used to call these

generators one hundred thousand times and the INTIME subroutine has been used to measure the time taken to do this. The time taken to execute the DO statement alone has been determined. The differences indicate the time taken for the random number generation by each generator. The results from these test runs are shown in Table 4.2. The saving in execution time is in excess of 50% with the BAL generator.

We have generated two sets of random numbers using the GASP IIA and the GASP IIP generators. EAch set consists of ten thousand numbers. These numbers have been tested using a one-dimensional chi-square test for agreement with the theoretical distribution. The theoretical distribution is assumed to be uniform over the interval 0.0 to 1.0. We have also made a serial test (two-dimensional chi-square) to test the degree of randomness between successive numbers. In both of these tests the significance level α, has been taken as 0.05.

Suppose we generate a set of N random numbers $r_1, r_2, r_3, \cdots, r_n$. We shall divide the range (0,1) of these variaties into M equal subintervals. The expected number of random numbers in each subinterval is (N/M). Let f_j, $j = 1, 2, \cdots, M$, denote the observed count of pseudorandom numbers r_i ($i = 1, 2, \cdots, N$) in the subinterval

$$(j-1)/M \leq r_i < (j/M)$$

Then,

$$\chi_1^2 = (M/N) \sum_{j=1}^{M} (f_j - N/M)^2, \text{ can}$$

be shown to have approximately a chi-square distribution with (M-1) degrees of freedom for a sequence of truely random numbers (4, p. 58). The calculated

Generators	Total time	Do loop time	Time for random generation
BAL	2387	289	2098
FORTRAN	5141	263	4878

$$\text{Savings in execution time} = \frac{4878-2098}{4878} \times 100$$

$$= 57.1\%$$

Table 4.2 Comparison of Speed

NOTE: All times are in hundredths of a second.

chi-square value is then compared with the theoretical value obtained from the table for the given degrees of freedom and significance level. If the calculated value is less than the theoretical value, we accept the null hypothesis (H_o) that the sequence of numbers generated are a random sample from their parent distribution.

The two-dimensional chi-square is usually applied to pairs of random numbers where the random numbers are taken as the coordinates of a point in a unit square divided into, say M^2 cells (4, p. 58). First the one-dimensional chi-square test is applied to the sequence of generated numbers and a chi-square value (χ^2) is calculated. Let, f_{jk} = a set of random numbers r_i ($i = 1, 2, \cdots, N-1$), that satisfies the inequalities

$$(j-1)/M \leq r_i < j/M \text{ and } (k-1)/M < r_{i+1} < K/M$$

where, $J = 1, 2, 3, \cdots, M$.

Then

$$\chi_2^2 = \frac{M^2}{N-1} \sum_{j=1}^{M} \sum_{k=1}^{M} (f_{jk} - \frac{N-1}{M^2})^2 .$$

It can be shown that $(\chi_2^2 - \chi_1^2)$ has an approximate chi-square distribution with $(M^2 - M)$ degrees of freedom (4, p. 59). If the chi-square value calculated is less than the theoretical value we accept the null hypothesis (H_o) that the numbers generated are random and that there is no serial correlation between successive numbers. The program used for the chi-square tests is shown in Appendix A.4. The results of these tests are presented in Table 4.3. The random numbers generated by both GASP IIA and GASP IIP generators passed the tests for the significance level 0.05.

Description	Significance level (α)	Degrees of freedom	χ^2 Theoretical	χ^2 Calculated	
				FORTRAN GEN	BAL GEN
One-dimensional Chi-square test	0.05	99	123.8	112.54	111.36
Two-dimensional Chi-square test	0.05	380	425	346.51	420.89

Table 4.3 Chi-square Tests.

4.3 Miscellaneous Changes

The subprograms FINDN, FINDQ, SUMQ, and PRODQ have been modified to conform to the pointer system and the attributes storing method followed in GASP IIP. Except for these modifications these subprograms function much the same way as their counterparts in GASP IIA. The source listing of these subprograms is included in Appendix B. The subroutine ERROR in GASP IIP calls the ENTRY PRNFL in subroutine PRNTQ to produce file printouts. The subroutine PRNTQ prints its own error messages to avoid enless loops in case of an error in PRNTQ. Table 4.4 lists the error codes and the subprograms in which they are raised.

Error Code	Subprogram in Which Error Occurred
20	PRODQ
	SUMQ
40	SET
50	FINDN
55	FINDQ
60	COLCT
70	TMST
80, 85	GASP
90, 95	PRNTQ
100	DATAX
110	SET (RMOVE)
120	SUMRY
130	MONTR
87, 140, 145	LOCAT
150	RESET

Table 4.4 Error Codes for GASP IIP Subprograms.

NOTE: Codes 90, 95, 120 and 130 are printed out by the subprograms in which they are reaised and not by subroutine ERROR.

CHAPTER 5

COMPARISON OF GASP IIA AND GASP IIP

5.1 Sample Simulation Problem

5.1.1 Objectives of the Simulation

We have written a jobshop simulation using GASP IIA and GASP IIP. This was one of the assigned problems in the simulation course offered by Grosh (5). Our primary objective here is not to study the characteristics of the simulated system, but to write a typical simulation that exercises the different sections in GASP IIA and GASP IIP so that we can compare

1. Programing effort involved in writing the simulation
2. Core space requirements
3. Speed of execution.

We have used, as much as possible, equivalent codings in both the cases to make the comparison meaningful. This sample problem will also serve to illustrate how to write a simulation using GASP IIP.

We have used FORTRAN IV G-Level and H-Level compilers to determine the execution speed and core space requirements in each case. For all our work we have used the IBM SYSTEM 360/50 Computer at Kansas State University.

5.1.2 Problem Defined

The simulation deals with a jobshop involving three machines. Orders are released to the shop every morning, the number of orders released has a Poisson distribution with a mean of 11 orders per day. Every job has three operations. A transition matrix (Table 5.1) is used to determine the

Present Machine (i) \ Next Machine (j)	1	2	3
New Order 0	0.4	0.5	0.1
1	0.0	0.6	0.4
2	0.3	0.0	0.7
3	0.5	0.5	0.0

P_{ij} = The probability that the next operation is on machine j given that the last one was on machine i.

Table 5.1 Transition Matrix

sequencing of the machines. This matrix gives the probability p_{ij} that the job currently on i^{th} machine will be processed next on j^{th} machine. The operation time has a Log Normal distribution with a mean of 2 hours and a standard deviation 0.5 hour. Each job is due 72 hours after entering the system. It is assumed, that in the beginning of the simulation there are no jobs in the system and that all machines are idle.

5.1.3 Statistics Gathered

We shall collect statistics on the slack time (= Due Date--Time of Completion) characteristics when the jobs are served on a FIFO (First In First Out) basis. We shall also collect statistics on machine utilizations and queue lengths. These statistics can be used to study the steady state and transient state characteristics of the system. We shall also collect data for histograms of job arrivals and operation times. These histograms can be used to study the resemblance between the generated and the theoretical distributions. Table 5.2 indicates how the GASP subprograms are used for this purpose.

5.1.4 Procedure Described

File Management. We shall use four files as indicated in Table 5.3. All files are ranked according to the first real attribute with the low values first.

Preparation of Data Cards. Figures 5.1 and 5.2 show the data sheets for this simulation in GASP IIA and GASP IIP. The cumulative probabilities calculated from the transition matrix and the parameters of the Poisson and the Log Normal distributions are read in through data card 6 in GASP IIA and data card 7 in GASP IIP. We have followed the procedure indicated by

Point Estimates and Histograms

Name of the Variable	Code Number in COLCT	Code Number in HISTO
Number of orders released per day (XJOBS)	1	1
Operation time (OPTM)	2	2
Slack time (=Due date - time of completion)	3	3

Time Weighted Statistics

Name of the Variable			Code Number
Jobs in queue	1	Given by entries in Files 2, 3 and 4	1
Jobs in queue	2		2
Jobs in queue	3		3
Status of machine	1	(BUS (1))	4
Status of machine	2	(BUS (2))	5
Status of machine	3	(BUS (3))	6
Number of jobs in the system			7

Table 5.2 Statistics Collected for the Sample Problem.

File No.	Event or Entity	Integer Attributes	Real Attributes
1	Arrival of JOB	1. Event code	1. Event time
	End of Operation	1. Event code	1. Event time
		2. Operation number	2. Due date
		3. Machine number	
2, 3, 4	Operations waiting for machines 1, 2 and 3	1. Dummy	1. Time of joining the queue
		2. Operation number	2. Due date
		3. Machine number	3. Operation time

Table 5.3 Files, Entities, Events and Attributes in the Sample Problem

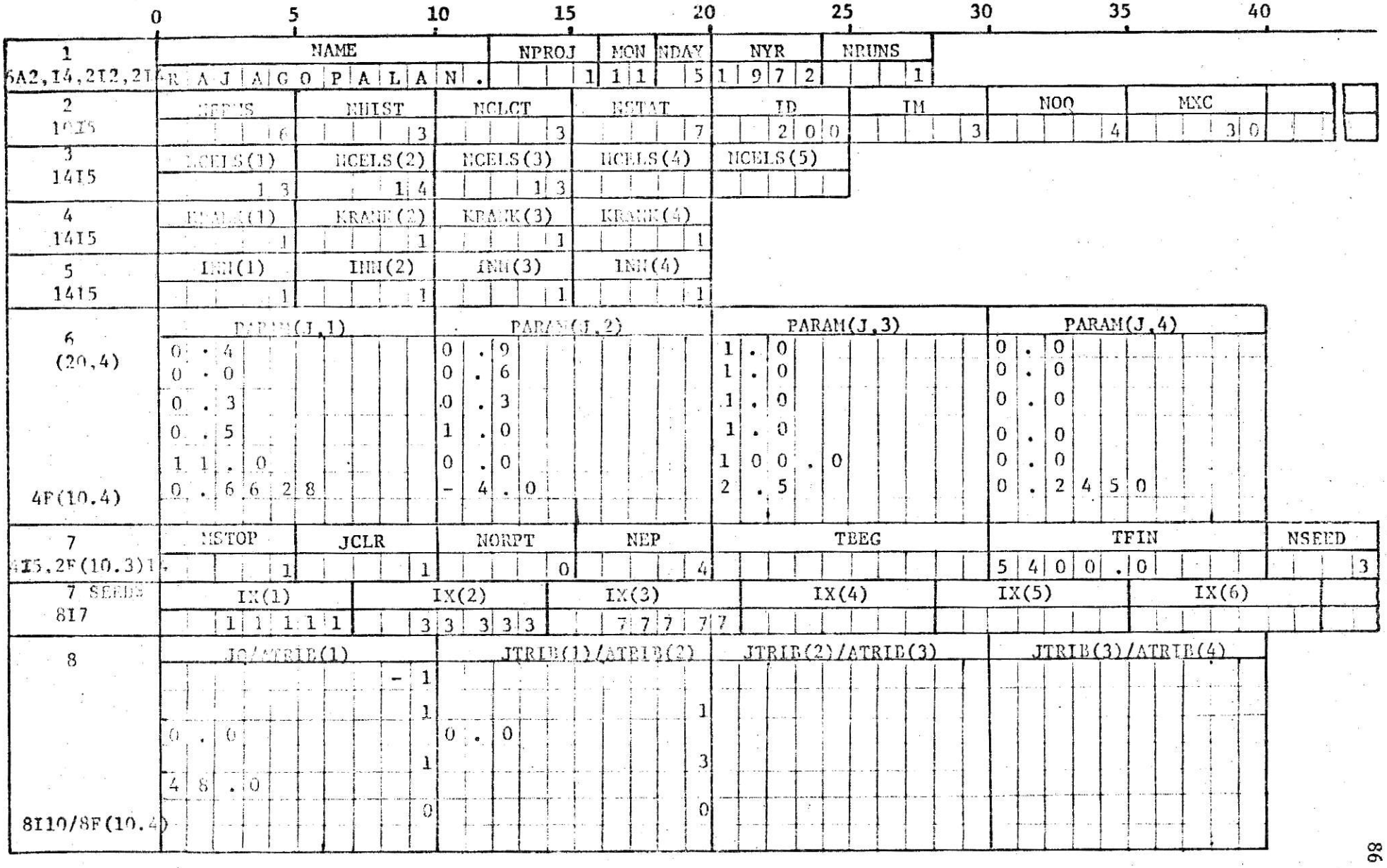


Figure 5.1 GASP IIA Data Sheet for Job Shop Simulation

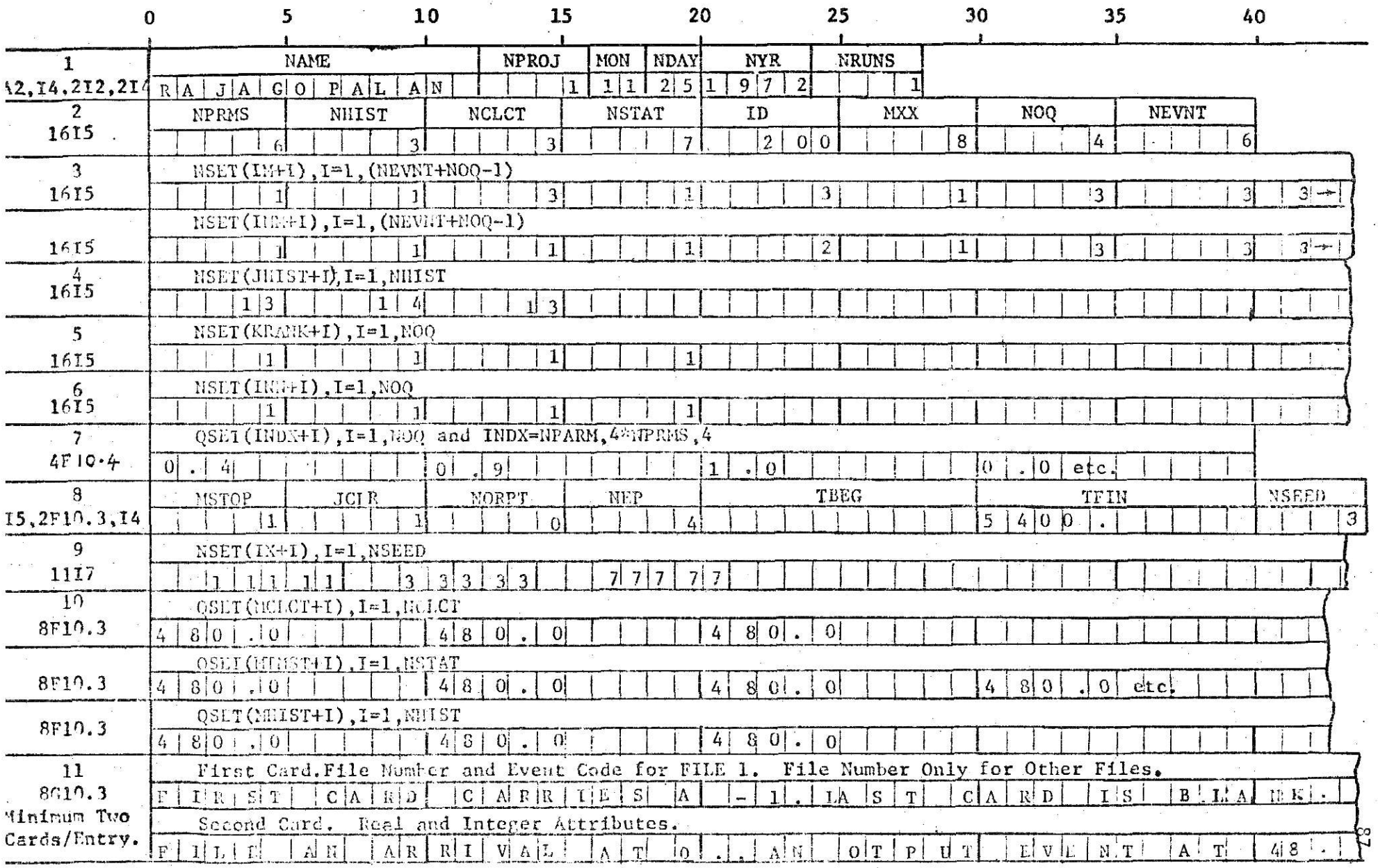


Figure 5.2 GASP IIP Data Sheet for Job Shop Simulation.

Pritsker and Kiviat (1, p. 98) to determine the parameters to be supplied for generating Log Normal variates. Suppose X is a normally distributed variable, then the variable Q given by the relation $Q = e^X$ is said to have a Log Normal distribution. The mean and the variance of Q and Y are related as:

$$\sigma_X^2 = \ln \left[\frac{\sigma_Q^2}{\mu_Q^2} + 1 \right]$$

From the data given,

$$\sigma_Q = 0.5 \quad \text{or} \quad \sigma_X = 0.245$$

$$\mu_Q = 2 \quad \text{or} \quad \mu_X = 0.6628$$

Assume,

Min Q = 1/60 hour OR Min X = -4

Max Q = 12 hours OR Max X = 2.5

μ, Min X, Max X, and σ_X are the parameters supplied to GASP for generating Log Normal varieties.

Dimensioning NSET and OSET. It is seen from Table 5.3 that in this example, the same entry type (jobs in the queues) has the maximum number of integer as well as real attributes; three in each category. Hence, in GASP IIA,

IM = 3, IMM = 3, and MXX = 5.

Similarly, in GASP IIP, MXX = 8. Hence the length of the unit filing areas is the same and equal to 8 in both the cases. Assume that the filing area will have to accommodate at most 200 entries at a time. Hence, ID = 200.

size of QSET = ID * LMM

= 600 cells

size of NSET = MXX * ID

= 1000 cells

In GASP IIP,

size of the filing area only = ID * MXX

= 1600 cells.

From Section 2.2,

Array space required = (12 * NOQ) + (3 * NEVNT) + (4 * NPRMS) + (6 * NCLCT) + (7 * NSTAT) + (4 * NHIST) + $\sum_{N=1}^{NHIST}$ (JHIST + N) + NSEED + (ID * MXX) -2

= (12 * 4) + (3 * 6) + (4 * 6) + (6 * 3) + (7 * 7) + (4 * 3) + (13 + 14 + 13) + 3 + (200 * 8) - 2

= 1810 cells

5.1.5 User Written Programs

A MAIN program and four subprograms have been written for this jobshop simulation. The functions of these user written programs are:

MAIN PROGRAM.

1. Set array dimensions.
2. Initialize non-GASP variables.
3. Call GASP.

EVNTS.

1. Call events according to event code

APRVL.

1. File the next arrival event.

2. Generate new arrivals. Generate machine number and operation time for each arrival and file it in the proper queue.
3. Test every machine. If a machine is idle and if there are jobs in its queue file an end of operation event for that machine.

ENDOP.

1. If the final operation is over compute slack time and exclude the job from the system. Go to Step 3.
2. If all operations are not over generate the next machine number and operation time. File the job in the proper queue. Test the new machine. If it is busy go to step 3, otherwise make the machine busy and file an end of operation event.
3. For the machine that just finished an operation if there are no jobs waiting in the queue make the machine idle, otherwise file an end of operation.

OUTPUT.

1. File the next output event.
2. Collect time statistics on the status of the machine.
3. Compute and print out the required intermediate results.

Source listings of the user written programs and the results have been given in Appendix A.5 and A.6.

5.2 Programming Effort Involved in GASP IIA and GASP IIP.

Any comparison of the programming effort involved in GASP IIA and GASP IIP is bound to be subjective. In the author's opinion, the various

additions, modifications and extensions discussed in Chapters 2, 3, and 4 have made simulation with GASP simpler in some areas and more complex in other areas. The blank COMMON block used in GASP IIP is more compact and easier to handle. The various GASP subprograms carry shorter or no argument lists. The provision of the RESET subroutine has considerably reduced the user's effort in resetting the different variables. It is easier to run an antithetic simulation in GASP IIP than in GASP IIA.

On the other hand the pointer system adopted in GASP IIP requires the computation and use of more complex subscripts. Further, to support the additional features incorporated in GASP IIP, the user should fill out two extra data cards (card types 3 and 10).

In GASP, whenever the entry in a column is removed, that column becomes the first available column NFA. This plus a thoughtful attribute arrangement for the different events and entities will help to reduce the work involved in the filing and retriving of entries. This is brought to light in Table 5.3, in the attribute arrangement of the end of operation events and the jobs in the queues. In these two entry types, common attributes occupy identical positions. Hence, in our example after a job has been removed from the queue, to file an end of operation for that job, we just insert two more attributes (event code and event time) into the column removed and call the ENTRY FILEM. Thus, in most cases a careful structuring and coding of the user written program will help to reduce the effort involved in using GASP for systems simulation.

5.3 Comparison of Core Space Requirements

5.3.1 Core Space Requirements of GASP IIA and GASP IIP

Table 5.4 compares the sizes of GASP IIA and GASP IIP subprograms. When compiled in G-Level the total size of the GASP IIP subprograms is

	GASP IIP		GASP IIA
SUBPROGRAM	G-Level	H-Level	G-Level
GASP	1400	956	1266
DATAX	7268	5884	4056
SET	4526	3084	3930
FILEN	-	-	740
RMOVE	-	-	730
LOCAT	646	410	758
MONTR	964	804	1482
HISTO	846	576	738
COLCT	552	398	728
TMST	590	420	746
DRAND:			486
DREG	104	104	-
DANTI	112	112	-
ARAND:			
AREG	112	112	-
AANTI	104	104	-
UNFRM	404	286	386
RNORM	716	536	672
RLOGN	396	308	366
NPOSN	918	740	914
ERLGN	778	566	730
PRNTQ	2050	1668	1614
SUMRY	2414	1994	2242
RESET	1198	914	-
ERROR	1378	1050	1638
SUMQ	1014	652	850
PRODQ	1026	660	862
FINDN	1026	636	1096
FINDQ	1228	760	1194
	31770	24734	28224

Table 5.4 Sizes of Subprograms in GASP IIA and GASP IIP

(All sizes in bytes)

3546 bytes (12.5%) greater than that of GASP IIA subprograms. Almost all of this difference is accounted for by the subprogram DATAX. Further, GASP IIA uses only one random number generator. Whereas GASP IIP uses four random number generators. But for these differences the core space requirements of GASP IIA and GASP IIP will be the same. However, merely comparing the numbers without taking into account the greater performance potentials of GASP IIP can be misleading. When GASP IIP is compiled in H-Level the total size comes to be 24734 bytes, thus resulting in 22% saving in core space over GASP IIP compiled in G-Level.

5.3.2 Core Space Requirements of the Sample Problem

Filing Arrays. In Section 2.3 we indicated that the attribute storage method adopted in GASP IIP will, in most cases, result in smaller filing area requirements. However, in our example, since the lengths of the unit filing areas are the same in both GASP IIA and GASP IIP, the filing area requirements to accommodate equal number of entries are also equal. For the jobshop simulation we use 1600 bytes of filing area in both cases. In general the space saving obtainable in the filing area with GASP IIP will vary from problem to problem, this saving can be significant in problems requiring large filing areas.

Other GASP-Arrays and Variables. Other precompiled arrays—control arrays for data management, statistical and information storage arrays—and the GASP variables has taken up 4640 bytes of core space. Since this is more than the requirements of the simulation, under utilization results. GASP IIP uses 1076 bytes for this purpose which is utilized 100%.

Breakdown of the Total Space Requirements. Table 5.5 shows the breakdown of the core space used in three test runs. For all the three

	GASP IIA	GASP IIP	
	Compiled in G--Level	Compiled in G-Level	Compiled in H-Level
Blank COMMON	4,640	7,476	7,476
Named COMMON	44	44	44
User Written Program	12,252	5,254	5,254
GASP Subprogram	22,348	25,966	19,846
IBM Routines	22,380	22,380	21,978
Total	61,664	61,120	54,598

Table 5.5 Core Space Breakdown in Three Test Runs

(All sizes in bytes)

test runs the user written routines were compiled in G-Level. The blank COMMON in GASP IIP includes the filing area. Whereas in GASP IIA the filing arrays appear in DIMENSION statements. This is one of the reasons for the user written programs in GASP IIA to be more than twice the size of GASP IIP subprograms. We can see from Table 5.5 that even with all the added facilities the total space requirements of GASP IIP, for this jobshop simulation is a little less than that of GASP IIA. When GASP IIP is compiled in H-Level we get a 10% space saving over GASP IIP compiled in G-Level.

5.4 Comparison of Execution Times

To compare the execution times we have simulated the jobshop using both GASP IIA and GASP IIP, for the same length of simulation time (5,400 hours). Table 5.6 shows that in this example the total number of jobs handled and completed are about the same in GASP IIA and GASP IIP.

We made five different runs as shown in Table 5.7. The first three runs were made in class A (56 K fast core and 72 K slow core). Due to the structural difference between the GASP IIA and GASP IIP arrays, the manner in which they use the fast and the slow core within the specified partition differ. This is brought to light in Table 5.7 by the percentage figures which indicate the proportion in which the fast core is available to the arrays in GASP IIA and GASP IIP.

When both GASP IIP and GASP IIA are compiled in G-Level and run in class A we obtain 17% saving in execution time with GASP IIP. In the third run, GASP IIP has been compiled in H-Level and the simulation run has been made in G-Level, class A. This is the fastest of all the five runs. In this run we get a 34% reduction in execution time over the second run and 20.5%

	GASP IIA	GASP IIP
Jobs completed	2273	2315
Back log	174	169
Jobs entering shop	2447	2484

Percentage difference:

Jobs completed = (42/2273) x 100

= 1.85%

Jobs entering shop = (37/2247) x 100

= 1.51%

Table 5.6 Comparison of Jobs Handled in the Sample System.

	RUNS	CORE BREAKDOWN: FILING ARRAYS	OTHER ARRAYS AND GASP VARIABLES	EXECUTION TIME IN HOURS	PERCENTAGE SAVING IN TIME	
1.	GASP IIP compiled in G-Level. Simulation run in G-Level, Class A	25% in fast core	100% in fast core	0.039	$\frac{0.008}{.047} \times 100$ = 17%	$\frac{.008}{.039} \times 100$ = 20.5%
2.	GASP IIA compiled in G-Level. Simulation run in G-Level, Class A	100% in fast core	5% in fast core	0.047	$\frac{0.016}{0.047} \times 100$ = 34%	
3.	GASP IIP compiled in G-Level. Simulation run in G-Level, Class A	100% in fast core	100% in fast core	0.031		
4.	GASP IIP compiled in G-Level. Simulation run in G-Level, Class B	100% in slow core	100% in slow core	0.115	$\frac{0.020}{0.135} \times 100$ =14.8%	
5.	GASP IIA compiled in G-Level. Simulation run in G-Level, Class B	100% in slow core	100% in slow core	0.135		

Table 5.7 Comparison of Execution Times.

reduction over the first run. The difference between the first and the third run can be attributed to the optimization done by the H-Level compiler and to the fact that in the third run the whole program is executed in fast core. The last two runs have been made in class B (128 K slow core) to eliminate the effect of the core speed between runs, so that the difference in execution times can be entirely attributed to the speed of the programs alone. It is seen from Table 5.7 that GASP IIP gives 14.8% reduction in execution time over GASP IIA when both are used entirely in slow core.

5.5 Conclusion

The test runs show that for medium size simulation problems, the proper mode of operation to suit the computing environment at Kansas State University will be, to compile GASP IIP in H-Level and run the simulation in G-Level. For making the test runs we have compiled GASP IIP and created a data set comprising the load modules of all the subprograms. The load module of every subprogram has been given a separate member name so that we have the ability to manipulate individual load modules. To execute a simulation the user written programs are compiled and linked to the GASP IIP data set.

Thus, through the restructuring of GASP arrays and the various modifications and extensions discussed in Chapters 2, 3, and 4 we have successfully eliminated the problems caused by the precompiled arrays. We have also added numerous new facilities and have made the existing facilities more effective. Even with all the added facilities and greater performance potentials, the core space requirements of GASP IIP is comparable to that of GASP IIA. GASP IIP is faster than GASP IIA and gives significant saving in execution time.

We have developed GASP IIP using the IBM SYSTEM 360/50 computer at Kansas State University. In all our work we have used the IBM FORTRAN IV G-Level and H-Level compilers. Some of the features used in GASP IIP such as, provision of multiple entry points in a subprogram and calling BAL subprograms from FORTRAN routines, are IBM extensions to American National Standard (ANS) FORTRAN. Further the BAL generators used in GASP IIP have been designed to use the full capacity of the 32-bits registers available in IBM SYSTEM 360 computers. To implement GASP IIP on other installations, these generators should be recoded to fit the word size available and the IBM extensions used in GASP IIP should be suitably modified. The IBM FORTRAN IV features not found in ANS FORTRAN have been listed in IBM FORTRAN IV language manual (Order No. GC28-6515-8). Further the user is cautioned to check if his compiler allows the type of COMMON implementations adopted in GASP IIP.

REFERENCES

1. Pritsker, A. Alan B., and Kiviat, Philip J. Simulation with GASP II. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.

2. Emshoff, James R., and Sisson, Roger L. Design and use of computer simulation models. The MacMillan Company. Collier-MacMillan Ltd., London, 1971.

3. Mihram, Arthur G. Simulation Statistical Foundations and Methodology. Academic Press, New York and London, 1972.

4. Naylor, Thomas H., et.al., Computer Simulation Techniques. John Wiley & Sons, Inc., New York.London.Sydney, 1966.

5. Grosh, Louis E., Assignments in Industrial Systems Simulation,Line No. 550-865, 1972, Spring, Kansas State University.

6. Lewis, P. A. W., Goodman, A. S., and Miller, J. M. A preudo-random number generator for the system/360: IBM System Journal, 1969: NO. 2., 136-146.

7. Larson, Chris. The Efficient Use of FORTRAN: DATAMATION, 1971, August 1, 24-31.

8. Kiviat, P. J. GASP-A General Activity Simulation Program. Monroeville, Pa.: Applied Research Laboratory, United States Steel Corporation, July 8, 1963.

APPENDIX A

TEST PROGRAMS

```
C*************************************************************
C     TO DETERMINE THE SAVING IN EXECUTION TIME WHEN A ONE-
C     DIMENSIONAL ARRAY IS USED INSTEAD OF A TWO-DIMENSIONAL
C     ARRAY
C*************************************************************
C
C     TEST RUN WITH TWO-DIMENSIONAL ARRAY
C
      COMMON SUMA(2,5)
      IX=11111
      NCLCT=2
      DO 1 I=1,NCLCT
      DO 2 J=1,3
    2 SUMA(I,J)=0.
      SUMA(I,4)=100.
    1 SUMA(I,5)=-100.
C
C     TIME FOR GENERATING RANDOM VARIABLES AND COMPUTING
C     POINT ESTIMATES
C
      CALL INTIME(I1)
      DO 3 K=1,5000
      X=DRAND(IX)
      CALL COLCT(X,2)
    3 CONTINUE
      CALL INTIME(I2)
      IX=11111
C
C     TIME FOR GENERATING RANDOM VARIATES ONLY
C
      CALL INTIME(I4)
      DO 5 I=1,5000
      X=DRAND(IX)
    5 CONTINUE
      CALL INTIME(I5)
      WRITE(6,4)((SUMA(I,J),J=1,5),I=1,NCLCT)
    4 FORMAT(' ',5F15.7)
      I3=(I2-I1)
      I6=(I5-I4)
      WRITE(6,6)I3,I6
    6 FORMAT(' ',2I10)
      STOP
      END


      SUBROUTINE COLCT(X,N)
      COMMON SUMA(2,5)
C
C     COMPUTE POINT ESTIMATES USING TWO-DIMENSIONAL ARRAY
C
      SUMA(N,1)=SUMA(N,1)+X
      SUMA(N,2)=SUMA(N,2)+X*X
      SUMA(N,3)=SUMA(N,3)+1.0
      SUMA(N,4)=AMIN1(SUMA(N,4),X)
      SUMA(N,5)=AMAX1(SUMA(N,5),X)
      RETURN
      END
```

Appendix A.1 Comparison of Execution Time.
One-dimensional Vs. Two-dimensional Array.

```
C
C     TEST RUN WITH A ONE-DIMENSIONAL ARRAY
C
      COMMON SUMA(10)
      IX=11111
      NCLCT=2
      INDX=1
      DO 1 I=1,NCLCT
      SUMA(INDX)=0.0
      SUMA(INDX+1)=0.0
      SUMA(INDX+2)=0.0
      SUMA(INDX+3)=100.0
      SUMA(INDX+4)=-100.0
    1 INDX=INDX+5
C
C     TIME FOR GENERATING RANDOM VARIABLES AND COMPUTING
C     POINT ESTIMATES
C
      CALL INTIME(I1)
      DO 3 K=1,5000
      X=DRAND(IX)
      CALL COLCT(X,2)
    3 CONTINUE
      CALL INTIME(I2)
      IX=11111
C
C     TIME FOR GENERATING RANDOM VARIATES ONLY
C
      CALL INTIME(I4)
      DO 5 I=1,5000
      X=DRAND(IX)
    5 CONTINUE
      CALL INTIME(I5)
      L2=5*NCLCT
      WRITE(6,4)(SUMA(I),I=1,L2)
    4 FORMAT(' ',5F15.7)
      I3=(I2-I1)
      I6=(I5-I4)
      WRITE(6,6)I3,I6
    6 FORMAT(' ',2I10)
      STOP
      END


      SUBROUTINE COLCT(X,N)
      COMMON SUMA(10)
C
C     COMPUTE POINT ESTIMATES USING ONE-DIMENSIONAL ARRAY
C
      INDX=5*(N-1)+1
      SUMA(INDX)=SUMA(INDX)+X
      SUMA(INDX+1)=SUMA(INDX+1)+X*X
      SUMA(INDX+2)=SUMA(INDX+2)+1.0
      SUMA(INDX+3)=AMIN1(SUMA(INDX+3),X)
      SUMA(INDX+4)=AMAX1(SUMA(INDX+4),X)
      RETURN
      END
```

```
      FUNCTION DRAND(IY)
C
C     GENERATE RANDOM VARIATES
C
      IY=IY*65539
      IF(IY.LT.0)IY=IY+2147483647+1
      DRAND=IY*0.4656613D-9
      RETURN
      END
```

```
C
C     TO ILLUSTRATE THE USE OF EXTERNAL STATEMENTS.
C     FIRST CALL STATEMENT WILL CAUSE '**DRAND**' TO BE
C     PRINTED. SECOND CALL WILL CAUSE '**ARAND**' TO BE
C     PRINTED.
C
      EXTERNAL DRAND,ARAND
      CALL NPOSN(DRAND)
      CALL NPOSN(ARAND)
      STOP
      END


      SUBROUTINE RNORM(BNAME)
      CALL BNAME
      RETURN
      END


      SUBROUTINE NPOSN(ANAME)
      EXTERNAL ANAME
      CALL RNORM(ANAME)
      RETURN
      END


      SUBROUTINE DRAND
      WRITE(6,1)
    1 FORMAT(' ','**DRAND**')
      RETURN
      END


      SUBROUTINE ARAND
      WRITE(6,1)
    1 FORMAT(' ','**ARAND**')
      RETURN
      END
```

Appendix A.2 External Statements.

```
C*************************************************************
C      TEST PROGRAMS TO COMPARE THE SPEED OF BAL AND FORTRAN
C      RANDOM NUMBER GENERATORS
C*************************************************************
       DIMENSION QSET(100)
       COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,
      1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,
      2NEVNT,NHIST,NOQ,NPRPT,NOT,NPRMS,NPUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,
      3NVNQ,NENQ,NFIL,NQ,NPARM,NOTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,
      4TNOW,TBEG,TFIN,NAME(6),NSET(1)
       EQUIVALENCE (NSET(1),QSET(1))
       IX=0
       NSET(IX+2)=33333
C
C      TOTAL TIME FOR GENERATING 100000 RANDOM NUMBERS USING
C      BAL GENERATOR
C
       CALL INTIME(I1)
       DO 1 I=1,100000
       R=DRAND(2)
     1 CONTINUE
C
C      TIME FOR EXECUTING JUST THE 'DO-LOOP'
C
       CALL INTIME(I2)
       DO 2 I=1,100000
     2 CONTINUE
       CALL INTIME(I3)
       J=I2-I1
       K=I3-I2
       WRITE(6,3)J,K
     3 FORMAT(' ',2I10)
       STOP
       END
```

```
DRAND    CSECT
         USING  *,15          INITIAL LINKAGE.
         STM    2,8,28(13)
         L      2,0(1)        LOAD ADDRESS OF THE VARIABLE PASSED.
         L      3,=A(IX)      COMPUTE ADDRESS OF THE SEED STORED IN NSET
         L      7,0(2)
         A      7,0(3)
         S      7,=F'1'
         M      6,=F'4'
         L      8,=A(NSET)
         AR     7,8
         L      5,A           COMPUTE NEXT INTEGER RANDOM NUMBER
         M      4,0(7)        AS NSET(IX+I)=A*NSET(IX+I)(MOD P).
         D      4,P
         ST     4,0(7)
         SRL    4,7           COMPUTE NEXT REAL RANDOM NUMBER AND STORE
         A      4,CHAR        IT IN THE FLOATING POINT REGISTER 0.
         ST     4,WORD
         LE     0,WORD
         LM     2,8,28(13)    TERMINAL LINKAGE.
         BR     14
CHAR     DC     F'1073741824' CONSTANTS. CHAR FIRST SO A IS ON DOUBLE
A        DC     F'16807'      WORD BOUNDARY. MAKES LM INSTRUCTION FASTER
P        DC     F'2147483647'
```

Appendix A.3 Comparison of Speed. BAL Vs. FORTRAN Generator.

```
WORD    DS    F
        COM
IX      DS    59F          IX IS CONSIDERED AS EQUIVALENT TO 59
NSET    DS    F            WORDS TO COMPUTE THE ADDRESS OF NSET.
        END
```

```
      COMMON ID,IM,INIT,JEVNT,JMNIT,MEA,MSTOP,MX,MXC,NCLCT,
     1NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT,TNOW,
     2TBEG,TFIN,MXX,NPRNT,NCRDR,NEP,VNQ(14),IMM,MAXQS,
     3MAXNS,ATRIB(10),ENQ(14),INN(14),JCELS(20,30),
     4KRANK(14),MAXNQ(14),MFE(14),MLC(14),NCELS(20),NQ(14),
     5PARAM(20,4),QTIME(14),SSUMA(25,5),SUMA(25,5),NAME(6),
     6NPROJ,MON,NDAY,NYR,JCLR,JTRIB(12),IX(8),MLE(14)
      IX(2)=33333
C
C     TIME FOR GENERATING 100000 RANDOM NUMBERS USING
C     FORTRAN GENERATOR
C
      CALL INTIMF(I1)
      DO 1 I=1,100000
      R=DRAND(2)
    1 CONTINUE
C
C     TIME FOR EXECUTING JUST THE 'DO-LOOP
C
      CALL INTIME(I2)
      DO 2 I=1,100000
    2 CONTINUE
      CALL INTIME(I3)
      J=I2-I1
      K=I3-I2
      WRITE(6,3)J,K
    3 FORMAT(' ',2I10)
      STOP
      END
```

```
      FUNCTION DRAND(I)                                          RKD01870
      COMMON ID,IM,INIT,JEVNT,JMNIT,MEA,MSTOP,MX,MXC,NCLCT,
     1NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT,TNOW,
     2TBEG,TFIN,MXX,NPRNT,NCRDR,NEP,VNQ(14),IMM,MAXQS,
     3MAXNS,ATRIB(10),ENQ(14),INN(14),JCELS(20,30),
     4KRANK(14),MAXNQ(14),MFE(14),MLC(14),NCELS(20),NQ(14),
     5PARAM(20,4),QTIME(14),SSUMA(25,5),SUMA(25,5),NAME(6),
     6NPROJ,MON,NDAY,NYR,JCLR,JTRIB(12),IX(8),MLE(14)
      IX(I)=IX(I)*65539                                          RKD01940
      IF(IX(I).LT.0)   IX(I) = IX(I) + 2147483647 + 1            RKD01950
      DRAND = IX(I)*0.4656613D-9                                 RKD01960
      IF(MOD(I,2).EQ.0) DRAND=1.0-DRAND                          RKD01961
      RETURN                                                     RKD01970
      END                                                        RKD01980
```

```
C*****************************************************************
C     PROGRAM FOR GENERATING A SET OF RANDOM NUMBERS AND MAKING
C     ONE-DIMENSIONAL AND TWO-DIMENSIONAL CHI-SQUARE TESTS TO TEST THE
C     RANDOMNESS OF THE NUMBERS GENERATED.
C*****************************************************************
C     DESCRIPTION OF THE VARIABLES USED:
C     NCOUNT,CCOUNT-COUNTERS FOR ONE-DIMENSIONAL CHI-SQUARE TESTS
C     MCOUNT,DCOUNT-COUNTERS FOR TWO-DIMENSIONAL CHI-SQUARE TESTS
C     KCOUNT,BCOUNT-COUNTERS FOR TWO-DIMENSIONAL CHI-SQUARE TESTS
C     PARAMETERS TO BE SUPPLIED
C     NMBRS          -TOTAL NUMBER OF RANDOM NUMBERS GENERATED
C     ICELS          -NUMBER OF CELLS FOR ONE-DIMENSIONAL CHI-SQUARE TEST
C     KCELS          -NUMBER OF CELLS FOR TWO-DIMENSIONAL CHI-SQUARE TEST
C     IY             -SEED NUMBER FOR RANDOM NUMBER GENERATOR
C*****************************************************************
      DIMENSION NCOUNT(200),CCOUNT(200),MCOUNT(200),DCOUNT(200),
     1KCOUNT(30,30),BCOUNT(30,30),A(10500)
C
C     TO READ PARAMETERS
C
      READ(5,1)NMBRS,ICELS,KCELS,IY
    1 FORMAT(8I10)
      WRITE(6,2)
    2 FORMAT('1',' INPUT PARAMETERS')
      WRITE(6,3)
    3 FORMAT('0','      NMBRS      ICELS      KCELS          IY')
      WRITE(6,4)NMBRS,ICELS,KCELS,IY
    4 FORMAT(' ',4I10)
C
C     TO INITIALIZE THE COUNTERS
C
      DO 5 I=1,ICELS
      NCOUNT(I)=0
      MCOUNT(I)=0
    5 CONTINUE
      DO 6 I=1,KCELS
      DO 6 J=1,KCELS
      KCOUNT(I,J)=0
      BCOUNT(I,J)=0.0
    6 CONTINUE
      AMBRS=FLOAT(NMBRS)
      ACELS=FLOAT(ICELS)
      BCELS=FLOAT(KCELS)
C
C     TO GENERATE RANDOM NUMBERS AND TO COUNT FREQUENCIES IN EACH CELL
C
      A(1)=DRAND(IY)
      I=A(1)*ACELS+1.0
      J=A(1)*BCELS+1.0
      NCOUNT(I)=NCOUNT(I)+1
      MCOUNT(J)=MCOUNT(J)+1
      DO 10 I=2,NMBRS
      A(I)=DRAND(IY)
      J=A(I-1)*BCELS+1.0
      K=A(I)*BCELS+1.0
      L=A(I)*ACELS+1.0
      KCOUNT(J,K)=KCOUNT(J,K)+1.0
      NCOUNT(L)=NCOUNT(L)+1
      MCOUNT(K)=MCOUNT(K)+1
   10 CONTINUE
```

Appendix A.4 Chi-Square Tests.

```
C
C      ONE-DIMENSIONAL CHI-SQUARE TEST
C
       SUM=0.0
       DO 11 I=1,ICELS
       CCOUNT(I)=NCOUNT(I)
       SUM=SUM+(CCOUNT(I)-AMBRS/ACELS)**2
   11  CONTINUE
       CHISQ1=ACELS*SUM/AMBRS
C
C      TWO-DIMENSIONAL CHI-SQUARE TEST
C
       SUM=0.0
       DO 12 I=1,KCELS
       DCOUNT(I)=MCOUNT(I)
       SUM=SUM+(DCOUNT(I)-AMBRS/BCELS)**2
   12  CONTINUE
       CHISQ3=BCELS*SUM/AMBRS
       SUM=0.0
       DO 13 I=1,KCELS
       DO 13 J=1,KCELS
       BCOUNT(I,J)=KCOUNT(I,J)
       SUM=SUM+(BCOUNT(I,J)-(AMBRS-1.)/BCELS**2)**2
   13  CONTINUE
       CHISQ2=(BCELS**2)*SUM/(AMBRS-1.)
       CHISQ=CHISQ2-CHISQ3
       NDF=KCELS**2-KCELS
C
C      TO PRINT OUT RESULTS
C
       WRITE(6,14)
   14  FORMAT('0',' FREQUENCIES FOR ONE-DIMENSIONAL CHI-SQUARE TEST')
       WRITE(6,15)(NCOUNT(I),I=1,ICELS)
   15  FORMAT(5X,10I10/)
       WRITE(6,16)CHISQ1
   16  FORMAT(//,5X,' VALUE OF CHI-SQUARE=',F10.4)
       WRITE(6,17)
   17  FORMAT('1',' FREQUENCIES FOR TWO-DIMENSIONAL CHI-SQUARE TEST')
       WRITE(6,18)((KCOUNT(I,J),I=1,KCELS),J=1,KCELS)
   18  FORMAT(5X,20I6/)
       WRITE(6,19)CHISQ2
   19  FORMAT(//,5X,' VALUE OF CHISQ2=',F10.4)
       WRITE(6,20)CHISQ
   20  FORMAT(//,5X,' VALUE OF CHI-SQUARE=',F10.4)
       WRITE(6,21)NDF
   21  FORMAT(//,5X,' DEGREES OF FREEDOM=',I4)
       STOP
       END
```

```
$APARM
C*************************************************************************
C     A JOB SHOP SIMULATION USING GASP IIA.
C*************************************************************************
C     LIST OF NON-GASP FORTRAN VARIABLES.
C     BUS       INDICATOR,(ZERO) IF A MACHINE IS IDLE,
C                         (ONE) IF A MACHINE IS BUSY.
C     NOP       NUMBER OF OPERATIONS PER JOB.
C     MACH      NUMBER OF MACHINES IN THE SYSTEM.
C     AINT      TIME INTERVAL BETWEEN TWO ARRIVALS.
C     AJTM      TIME ALLOWED FOR THE COMPLETION OF A JOB.
C     XISYS     NUMBER OF ORDERS IN THE SYSTEM.
C     RESET     'RESET' ASSUMES THE VALUE OF 'TNOW' AT THE TIME OF
C               RESETTING IN SUBROUTINE 'EVNTS'.
C*************************************************************************
      DIMENSION NSET(1000),QSET(600)
      COMMON IB,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,NCLCT,
     1NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT,TNOW,
     2TREG,TRIN,MXX,NPRNT,NCRDR,NCP,VNQ(14),IUM,MAXQS,
     3MAXNS,ATR(10),TNQ(14),TNB(14),JCELS(20,30),
     4KRANK(14),MAXNQ(14),MFE(14),MLC(14),NCELS(20),NQ(14),
     5PARAM(20,4),QTIME(14),SSUMA(25,5),SUMA(25,5),NAME(6),
     6NPROJ,MON,NDAY,NYR,JCLR,JTRIB(12),IX(8),MLE(14)
      COMMON/U/AINT,AJTM,MACH,NOP,XISYS,BUS(6)
      NCRDR=1
      NPRNT=3
      AINT=24.
      AJTM=72.
      MACH=3
      NOP=3
C
C     TO START WITH THERE ARE NO JOBS IN THE SYSTEM, ALL THE QUEUES ARE
C     EMPTY AND ALL THE MACHINES ARE IDLE.
C
      XISYS=0.
      DO 1 I=1,MACH
      BUS(I)=0.
    1 CONTINUE
      CALL GASP(NSET,QSET)
      STOP
      END
```

Appendix A.5 Job Shop Simulation with GASP IIA.

```
      SUBROUTINE EVNTS(I,NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON/C0/IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,NCLCT,
     1NHIST,NOQ,NPRPT,NOT,NPRMS,NFUM,NPUMS,NSTAT,OUT,TNOW,
     2TREG,TFIN,MXX,NPRNT,NCRDR,NFP,VUQ(14),IMM,MAXQS,
     3MAXMS,ATRIB(12),ENQ(14),INL(14),JCELS(20,30),
     4KRANK(14),MAXNQ(14),MFE(14),MLC(14),NCELS(20),NQ(14),
     5PARAM(20,4),QTIME(14),SSUMA(25,5),SUMA(25,5),NAME(6),
     6NPROJ,MON,NDAY,NYR,JCLR,JTBIS(12),IX(8),MLE(14)
      COMMON/U/AINT,AJTM,MACH,NOP,XISYS,BUS(6)
      GO TO (1,2,3),I
    1 CALL ARRVL(NSET,QSET)
      RETURN
    2 CALL ENDOP(NSET,QSET)
      RETURN
    3 CALL OTPUT(NSET,QSET)
      RETURN
      END
```

```
      SUBROUTINE ARRVL(NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON IB,IM,INIT,JEVNT,JWAIT,MEA,MSTOP,MX,MXC,NCLCT,
     1NHIST,NOO,NOPRT,NOT,NPRMS,NRUN,NSOLS,NSTAT,OUT,TNOW,
     2TREG,TRIN,MXX,NPRNT,NCRDR,IBP,VAO(14),IMM,MAXQS,
     3MAXQS,ATRIB(10),FFO(14),IMM(14),JCELS(20,30),
     4KRANK(14),MAXNQ(14),MFE(14),NLC(14),NOELS(20),NQ(14),
     5PARAM(20,4),QTIME(14),SSUM(25,5),SUMA(25,5),NAME(6),
     6NPROJ,MON,NDAY,NYR,JCLR,JTRIB(12),IX(8),MLE(14)
      COMMON/U/AINT,AJTM,MACH,NOP,XISYS,AUS(6)
C
C     FILE THE NEXT ARRIVAL EVENT.
C
      ATRIB(1) = TNOW + AINT
      JTRIB(1) = 1
      CALL FILEM(1,NSET,QSET)
C
C     FIND OUT NUMBER OF JOBS ARRIVING (NJOBS). COLLECT STATISTICS ON
C     NJOBS.
C
      NJOBS=NPOSN(1,5)
      XJOBS=NJOBS
      CALL COLCT(XJOBS,1,NSET,QSET)
      CALL HISTO(XJOBS,2.,2.,1)
C
C     COLLECT STATISTICS ON XISYS.
C
      CALL TMST(XISYS,TNOW,NSTAT,NSET,QSET)
      XISYS=XISYS+XJOBS
C
C     FILE ALL THE JOBS JUST ARRIVED IN THE PROPER QUEUE, WITH
C     FIXED POINT ATTRIBUTES     1. DUMMY
C                                2. OPERATION NUMBER
C                                3. MACHINE NUMBER AND
C     FLOATING POINT ATTRIBUTES  1. TIME OF ARRIVAL IN QUEUE
C                                2. DUE DATE
C                                3. OPERATION TIME.
C
      ATRIB(1) = TNOW
      ATRIB(2) = TNOW + AJTM
      JTRIB(1)=0
      JTRIB(2) = 0
C
C     GENERATE MACHINE NUMBER.
C
      DO 1 I=1,NJOBS
      RN= DRAND(3)
      DO 2 NEXTM =1, MACH
      IF(PARAM(I,NEXTM) - RN) 2,3,3
    2 CONTINUE
    3 JTRIB(3) = NEXTM
C
C     GENERATE OPERATION TIME.
C
      OPTM= RLOGN(2,6)
      ATRIB(3)=OPTM
C
C     TAKE STATISTICS ON THE NUMBER OF JOBS IN THE QUEUE AND FILE THE
C     NEW JOB IN THE PROPER QUEUE.
C
```

```
      JQ=NEXTM+1
      ANQ=NQ(JQ)
      CALL TMST(ANQ,TNOW,NEXTM,NSET,QSET)
      CALL FILEM(JQ,NSET,QSET)
C
C     COLLECT STATISTICS ON OPERATION TIME.
C
      CALL COLCT(OPTM,2,NSET,QSET)
      CALL HISTO(OPTM,,.25,.25,2)
    1 CONTINUE
C
C     TEST THE MACHINES TO SEE IF THEY ARE BUSY OR IDLE. IF A MACHINE
C     IS IDLE, TEST THE QUEUE. IF THERE IS A JOB IN THE QUEUE, COLLECT
C     STATISTICS ON THE MACHINE UTILIZATION, THEN MAKE THE MACHINE
C     BUSY AND FILE AN END OPERATION EVENT WITH
C     FIXED POINT ATTRIBUTES          1. EVENT CODE
C                                     2. OPERATION NUMBER
C                                     3. MACHINE NUMBER AND
C     FLOATING POINT ATTRIBUTES       1. EVENT TIME
C                                     2. DUE DATE
C
      DO 4 I=1,MACH
      IF(BUS(I))5,6,4
    5 CALL ERROR(1,NSET,QSET)
    6 JQ=I+1
      IF(NQ(JQ))8,4,9
    8 CALL ERROR(2,NSET,QSET)
    9 NCODE=I+MACH
      CALL TMST(0., TNOW,NCODE,NSET,QSET)
      BUS(I) = 1.
      ANQ=NQ(JQ)
      CALL TMST(ANQ,TNOW,I,NSET,QSET)
      NCOL=MFE(JQ)
      CALL RMOVE(NCOL,JQ,NSET,QSET)
      ATRIB(1) = TNOW + ATRIB(3)
      JTRIB(1) = 2
      CALL FILEM(1,NSET,QSET)
    4 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ENDOP(NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON IQ,IM,INIT,JEVNT,JMNIT,NEA,MSTOP,MX,MXC,NCLCT,
     1NHIST,NOO,NERRT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT,TNOW,
     2TREG,TFIN,MXX,NPRNT,NCRDR,NER,VNO(14),IMM,MAXQS,
     3MAXNS,ATRIB(12),ENO(14),INV(14),JCELS(20,30),
     4KRANK(14),MAXNQ(14),MFE(14),MLC(14),NCELS(20),NQ(14),
     5PARAM(20,4),QTIME(14),SSUMA(25,5),SUMA(25,5),NAME(6),
     6NPROJ,MON,NDAY,NYR,JCLR,JTRIB(12),IX(8),MLE(14)
      COMMON/U/AINT,AJTM,MACH,NOP,XISYS,BUS(6)
C
C     STORE THE MACHINE NUMBER.
C
      NSAVE=JTRIB(3)
      JTRIB(2)=JTRIB(2)+1
C
C     IF ALL OPERATIONS ARE OVER COLLECT STATISTICS ON SLACK TIME.
C
      IF(JTRIB(2)-NOP)1,2,2
    2 SLKTM=ATRIB(2)-TNOW
      CALL COLCT(SLKTM,3,NSET,QSET)
      CALL HISTO(SLKTM,-15.,3.,3)
      CALL TMST(XISYS,TNOW,NSTAT,NSET,QSET)
      XISYS=XISYS-1.
      GO TO 7
C
C     GENERATE MACHINE NUMBER AND FILE THE JOB IN ITS QUEUE WITH
C     FIXED POINT ATTRIBUTES          1. DUMMY
C                                     2. OPERATION NUMBER
C                                     3. MACHINE NUMBER AND
C     FLOATING POINT ATTRIBUTES       1. TIME OF ARRIVAL IN QUEUE
C                                     2. DUE DATE
C                                     3. OPERATION TIME.
C
    1 RN=DRAND(3)
      INDX=NSAVE+1
      DO  3 NEXTM = 1,MACH
      IF(PARAM(INDX,NEXTM)-RN) 3,4,4
    3 CONTINUE
    4 JTRIB(3) = NEXTM
      JTRIB(1)= 0
C
C      GENERATE OPERATION TIME AND COLLECT STATISTICS ON IT.
C
      OPTM=RLOGN(2,6)
      CALL COLCT(OPTM,2,NSET,QSET)
      CALL HISTO(OPTM,.25,.25,2)
      ATRIB(3) = OPTM
      JQ=NEXTM+1
      ANQ=NQ(JQ)
      CALL TMST(ANQ,TNOW,NEXTM,NSET,QSET)
      CALL FILEM(JQ,NSET,QSET)
C
C     CHECK IF THE MACHINE IS BUSY OR IDLE. IF THE MACHINE IS IDLE
C     COLLECT STATISTICS ON IDLE TIME AND SET THE MACHINE BUSY. COLLECT
C     STATISTICS ON THE NUMBER OF JOBS IN THE QUEUE. REMOVE THE JOB FROM
C     THE QUEUE AND FILE AN END-OF-OPERATION EVENT.
C
      IF(BUS(NEXTM)) 5,6,7
    5 CALL ERROR(1,NSET,QSET)
```

```
      6 NCODE=NEXTM+MACH
        CALL TMST(0.,TNOW,NCODE,NSET,QSET)
        BUS(NEXTM)= 1.
        ANQ=NQ(JQ)
        CALL TMST(ANQ,TNOW,NEXTM,NSET,QSET)
        NCOL=MFE(JQ)
        CALL RMOVE(NCOL,JQ,NSET,QSET)
        ATRIB(1) = TNOW + ATRIB(3)
        JTRIB(1) = 2
        CALL FILEM(1,NSET,QSET)
C
C       TEST AND SEE IF THERE ARE ANY JOBS WAITING FOR THE PREVIOUS
C       MACHINE . IF THERE IS ANY JOB COLLECT STATISTICS ON THE JOBS IN
C       QUEUE. REMOVE THE JOB FROM THE QUEUE AND FILE AN END-OF-OPERATION
C       EVENT. IF THERE IS NO JOB COLLECT STATISTICS ON THE BUSY TIME AND
C       MAKE THE MACHINE IDLE.
C
      7 JQ=NSAVE+1
        IF(NQ(JQ)) 8,9,10
      8 CALL ERROR(2,NEST,QSET)
      9 NCODE=NSAVE+MACH
        CALL TMST(1.,TNOW,NCODE,NSET,QSET)
        BUS(NSAVE)=0.
        RETURN
     10 ANQ=NQ(JQ)
        CALL TMST(ANQ,TNOW,NSAVE,NSET,QSET)
        NCOL=MFE(JQ)
        CALL RMOVE(NCOL,JQ,NSET,QSET)
        ATRIB(1)= TNOW + ATRIB(3)
        JTRIB(1) = 2
        CALL FILEM(1,NSET,QSET)
        RETURN
        END
```

```
      SUBROUTINE OTPUT(NSET,QSET)
      DIMENSION NSET(1),QSET(1)
      COMMON ID,IM,IUIT,JEVNT,JMNIT,MEA,MSTOP,MX,MXC,NCLCT,
     1NHIST,NOS,NPRT,NRT,NRMS,NSEN,NTIMS,NSTAT,OUT,TNOW,
     2TREG,TEIP,MXX,NPRNT,NCRDR,NEP,VND(14),IMM,MAXQS,
     3MAXNS,ATRIB(10),END(14),INE(14),JCELS(20,30),
     4KRANK(14),MAXND(14),MEF(14),MLE(14),NCELS(20),NQ(14),
     5PARAM(20,4),QTIME(14),SSUMA(25,5),SUMA(25,5),NAME(6),
     6NPROJ,MON,NDAY,NYR,JCLR,JTRIB(12),IX(8),MLE(14)
      COMMON/U/AINT,AJTM,MACH,NJP,XISYS,BUS(6)
      DIMENSION AVQ(10),AUT(10)
C
C     FILE NEXT REPORT EVENT
C
      ATRIB(1)=TNOW+ 48.
      JTRIB(1)=3
      CALL FILEM(1,NSET,QSET)
      DO 1 I=1,MACH
      AA=BUS(I)
      NCODE=MACH+I
      CALL TMST(AA,TNOW,NCODE,NSET,QSET)
    1 CONTINUE
C
C     CALCULATE STATISTICS:  AVOP       -AVERAGE OPERATION TIME
C                            AVORD      -AVERAGE NUMBER OF ORDERS RECD
C                            AVSLK      -AVERAGE SLACK TIME
C                            AVQ(I)     -AVERAGE NUMBER OF JOBS IN 'Q'
C                            AUT(I)     -AVERAGE UTILIZATION OF M/C 'I'
C                            AVSYS      -AVERAGE NUMBER OF JOBS IN SYSTEM
C
      AVORD=SUMA(1,1)/SUMA(1,3)
      AVOP=SUMA(2,1)/SUMA(2,3)
      AVSLK = SUMA(3,1)/SUMA(3,3)
      DO 3 I=1,MACH
      AVQ(I)=(SSUMA(I,2)/SSUMA(I,1))
    3 CONTINUE
      IF(TNOW.NE.48.)GO TO 6
      WRITE(NPRNT,4)
    4 FORMAT('1','       AVOP      AVORD      AVSLK      AVQ 1      AVQ 2
     1 AVQ 3      AUT 1      AUT 2      AUT 3      AVSYS')
      LY=MACH+1
      LZ=2*MACH
    6 DO 2 I=LY,LZ
      AUT(I)=SSUMA(I,2)/SSUMA(I,1)
    2 CONTINUE
      AVSYS=(SSUMA(NSTAT,2)/SSUMA(NSTAT,1))
      WRITE(NPRNT,5)AVOP,AVORD,AVSLK,(AVQ(I),I=1,MACH),
     2(AUT(I),I=LY,LZ),AVSYS
    5 FORMAT(' ',10F10.4)
      RETURN
      END
```

```
FORTRAN IV G LEVEL  21                MAIN              DATE = 73045        22/45/31

            C*************************************************************************
            C     A JOB SHOP SIMULATION USING GASP IIP.
            C*************************************************************************
            C     LIST OF NON-GASP FORTRAN VARIABLES.
            C     BUS       INDICATOR,(ZERO) IF A MACHINE IS IDLE,
            C                          (ONE) IF A MACHINE IS BUSY.
            C     NOP       NUMBER OF OPERATIONS PER JOB.
            C     MACH      NUMBER OF MACHINES IN THE SYSTEM.
            C     AINT      TIME INTERVAL BETWEEN TWO ARRIVALS.
            C     AJTM      TIME ALLOWED FOR THE COMPLETION OF A JOB.
            C     XISYS     NUMBER OF ORDERS IN THE SYSTEM.
            C*************************************************************************
0001              DIMENSION QSET(1810)
0002              COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,
                 1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,
                 2NEVNT,NHIST,NTQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,
                 3NVNQ,NENQ,NFIL,NQ,NPARM,NOTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,
                 4TNOW,TREG,TFIN,NAME(6),NSET(1)
0003              EQUIVALENCE (NSET(1),QSET(1))
0004              COMMON/U/AINT,AJTM,MACH,NOP,XISYS,BUS(6)
0005              NCRDR=1
0006              NPRNT=3
0007              AINT=24.
0008              AJTM=72.
0009              MACH=3
0010              NOP=3
            C
            C     TO START WITH THERE ARE NO JOBS IN THE SYSTEM, ALL THE QUEUES ARE
            C     EMPTY AND ALL THE MACHINES ARE IDLE.
            C
0011              XISYS=0.
0012              DO 1 I=1,MACH
0013              BUS(I)=0.
0014            1 CONTINUE
0015              CALL GASP
0016              STOP
0017              END
```

Appendix A.6 Job Shop Simulation with GASP IIP.

```
FORTRAN IV G LEVEL  21                  EVNTS                DATE = 73045          22/43/31

0001                  SUBROUTINE EVNTS
0002                  DIMENSION QSET(1)
0003                  COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,
                     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLF,MCLCT,MTMST,MHIST,NCLCT,
                     2NEVNT,NHIST,NUD,NQRPT,NDT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,
                     3NVNQ,NENQ,NFIL,NQ,NPARM,NOTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,
                     4TNOW,TBEG,TFIN,NAME(6),NSET(1)
0004                  EQUIVALENCE (NSET(1),QSET(1))
0005                  COMMON/U/AINT,AJTM,MACH,NOP,XISYS,BUS(6)
0006                  JSW=JEVNT-3
0007                  GO TO (1,2,3),JSW
0008                1 CALL ARRVL
0009                  RETURN
0010                2 CALL ENDOP
0011                  RETURN
0012                3 CALL OTPUT
0013                  RETURN
0014                  END
```

```
FORTRAN IV G LEVEL  21                ARRVL                DATE = 73045          22/43/31

0001                  SUBROUTINE ARRVL
0002                  EXTERNAL DRAND
0003                  DIMENSION QSET(1)
0004                  COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,
                     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTNST,MHIST,NCLCT,
                     2NEVNT,NHIST,NOQ,NOPPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,
                     3NVNQ,NENQ,NFIL,NQ,NPARM,NOTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,
                     4TNOW,TBEG,TFIN,NAME(6),NSET(1)
0005                  EQUIVALENCE (NSET(1),QSET(1))
0006                  COMMON/U/AINT,AJTM,MACH,NOP,XISYS,BUS(6)
                C
                C     FILE THE NEXT ARRIVAL EVENT.
                C
0007                  IN=LOCAT(1,MFA,1)
0008                  NSET(IN)=4
0009                  QSET(IN+1)=TNOW+AINT
0010                  CALL FILEM(1)
                C
                C     FIND OUT NUMBER OF JOBS ARRIVING (NJOBS). COLLECT STATISTICS ON
                C     NJOBS.
                C
0011                  NJOBS=NPOSN(1,5,DRAND)
0012                  XJOBS=NJOBS
0013                  CALL COLCT(XJOBS,1)
0014                  CALL HISTO(XJOBS,2.,2.,1)
                C
                C     COLLECT STATISTICS ON XISYS.
                C
0015                  CALL TMST(XISYS,NSTAT)
0016                  XISYS=XISYS+XJOBS
                C
                C     FILE ALL THE JOBS JUST ARRIVED IN THE PROPER QUEUE, WITH
                C     FIXED POINT ATTRIBUTES    1. DUMMY
                C                               2. OPERATION NUMBER
                C                               3. MACHINE NUMBER AND
                C     FLOATING POINT ATTRIBUTES 1. TIME OF ARRIVAL IN QUEUE
                C                               2. DUE DATE
                C                               3. OPERATION TIME.
                C
0017                  DO 1 I=1,NJOBS
0018                  IN=LOCAT(1,MFA,1)
0019                  NSET(IN)=0
                C
                C     GENERATE MACHINE NUMBER.
                C
0020                  RN=DRAND(3)
0021                  INDX=NPARM+1
0022                  DO 2 NEXTM=1,MACH
0023                  IF(QSET(INDX)-RN)2,3,3
0024                2 INDX=INDX+1
0025                3 NSET(IN+1)=0
0026                  NSET(IN+2)=NEXTM
0027                  QSET(IN+3)=TNOW
0028                  QSET(IN+4)=TNOW+AJTM
                C
                C     GENERATE OPERATION TIME.
                C
0029                  OPTM=RLOGN(2,6,DRAND)
```

```
FORTRAN IV G LEVEL  21                ARRVL                DATE = 73045          22/43/31

0030                  QSET(IN+5)=OPTM
                C
                C     TAKE STATISTICS ON THE NUMBER OF JOBS IN THE QUEUE AND FILE THE
                C     NEW JOB IN THE PROPER QUEUE.
                C
0031                  JQ=NEXTM+1
0032                  ANQ=NSET(NQ+JQ)
0033                  CALL TMST(ANQ,NEXTM)
0034                  CALL FILEM(JQ)
                C
                C     COLLECT STATISTICS ON OPERATION TIME.
                C
0035                  CALL COLCT(OPTM,2)
0036                  CALL HISTO(OPTM,.25,.25,2)
0037                1 CONTINUE
                C
                C     TEST THE MACHINES TO SEE IF THEY ARE BUSY OR IDLE. IF A MACHINE
                C     IS IDLE, TEST THE QUEUE. IF THERE IS A JOB IN THE QUEUE, COLLECT
                C     STATISTICS ON THE MACHINE UTILIZATION, THEN MAKE THE MACHINE
                C     BUSY AND FILE AN END OPERATION EVENT WITH
                C     FIXED POINT ATTRIBUTES          1. EVENT CODE
                C                                     2. OPERATION NUMBER
                C                                     3. MACHINE NUMBER AND
                C     FLOATING POINT ATTRIBUTES       1. EVENT TIME
                C                                     2. DUE DATE
                C
0038                  DO 4 I=1,MACH
0039                  IF(BUS(I))5,6,4
0040                5 CALL ERROR(1)
0041                6 JQ=I+1
0042                  IF(NSET(NQ+JQ))8,4,9
0043                8 CALL ERROR(2)
0044                9 NCODE=1+MACH
0045                  CALL TMST(0.,NCODE)
0046                  BUS(I)=1.
0047                  ANQ=NSET(NQ+JQ)
0048                  CALL TMST(ANQ,I)
0049                  NCOL=NSET(MFE+JQ)
0050                  CALL RMOVE(NCOL,JQ)
                C
                C     NOW THE COLUMN REMOVED 'NCOL' WILL BE 'MFA'.
                C
0051                  IN=LOCAT(1,MFA,1)
0052                  NSET(IN)=5
0053                  QSET(IN+3)=TNOW+QSET(IN+5)
0054                  CALL FILEM(1)
0055                4 CONTINUE
0056                  RETURN
0057                  END
```

```
FORTRAN IV G LEVEL  21                ENDOP                DATE = 73045          22/43/31

0001                  SUBROUTINE ENDOP
0002                  EXTERNAL DRAND
0003                  DIMENSION QSET(1)
0004                  COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,
                     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,
                     2NFVNT,NHIST,NQQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,
                     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,
                     4TNOW,TBEG,TFIN,NAME(6),NSET(1)
0005                  EQUIVALENCE (NSET(1),QSET(1))
0006                  COMMON/U/AINT,AJTM,MACH,NOP,XISYS,BUS(6)
                C
                C     STORE THE MACHINE NUMBER.
                C
0007                  NCOL=NSET(MFE+1)
0008                  IN=LOCAT(1,NCOL,1)
0009                  NSAVE=NSET(IN+2)
0010                  OPNO=NSET(IN+1)+1
                C
                C     IF ALL OPERATIONS ARE OVER COLLECT STATISTICS ON SLACK TIME.
                C
0011                  IF(OPNO-NOP)1,2,2
0012                2 SLKTM=QSET(IN+4)-TNOW
0013                  CALL COLCT(SLKTM,3)
0014                  CALL HISTO(SLKTM,-15.,3.,3)
0015                  CALL TMST(XISYS,NSTAT)
0016                  XISYS=XISYS-1.
0017                  GO TO 7
                C
                C     GENERATE MACHINE NUMBER AND FILE THE JOB IN ITS QUEUE WITH
                C     FIXED POINT ATTRIBUTES          1. DUMMY
                C                                     2. OPERATION NUMBER
                C                                     3. MACHINE NUMBER AND
                C     FLOATING POINT ATTRIBUTES       1. TIME OF ARRIVAL IN QUEUE
                C                                     2. DUE DATE
                C                                     3. OPERATION TIME.
                C
0018                1 RN=DRAND(3)
0019                  INDX=4*NSAVE+(NPARM+1)
0020                  DO 3 NEXTM=1,MACH
0021                  IF(QSET(INDX)-RN)3,4,4
0022                3 INDX=INDX+1
0023                4 IP=LOCAT(1,MFA,1)
0024                  NSET(IP)=0
0025                  NSET(IP+1)=OPNO
0026                  NSET(IP+2)=NEXTM
0027                  QSET(IP+3)=TNOW
                C
                C     GENERATE OPERATION TIME AND COLLECT STATISTICS ON IT.
                C
0028                  OPTM=RLOGN(2,6,DRAND)
0029                  CALL COLCT(OPTM,2)
0030                  CALL HISTO(OPTM,.25,.25,2)
0031                  QSET(IP+4)=QSET(IN+4)
0032                  QSET(IP+5)=OPTM
0033                  JQ=NEXTM+1
0034                  ANQ=NSET(NQ+JQ)
0035                  CALL TMST(ANQ,NEXTM)
0036                  CALL FILEM(JQ)
```

```
FORTRAN IV G LEVEL  21                ENDOP                DATE = 73045          22/43/31

                C
                C     CHECK IF THE MACHINE IS BUSY OR IDLE. IF THE MACHINE IS IDLE
                C     COLLECT STATISTICS ON IDLE TIME AND SET THE MACHINE BUSY. COLLECT
                C     STATISTICS ON THE NUMBER OF JOBS IN THE QUEUE. REMOVE THE JOB FROM
                C     THE QUEUE AND FILE AN END-OF-OPERATION EVENT.
                C
0037                  IF(BUS(NEXTM))5,6,7
0038                5 CALL ERROR(1)
0039                6 NCODE=NEXTM+MACH
0040                  CALL TMST(0.,NCODE)
0041                  BUS(NEXTM)=1.
0042                  ANQ=NSET(NQ+JQ)
0043                  CALL TMST(ANQ,NEXTM)
0044                  NCOL=NSET(MFE+JQ)
0045                  CALL RMOVE(NCOL,JQ)
                C
                C     NOW COLUMN REMOVED 'NCOL' WILL BE 'MFA'.
                C
0046                  IN=LOCAT(1,MFA,1)
0047                  NSET(IN)=5
0048                  QSET(IN+3)=TNOW+QSET(IN+5)
0049                  CALL FILEM(1)
                C
                C     TEST AND SEE IF THERE ARE ANY JOBS WAITING FOR THE PREVIOUS
                C     MACHINE . IF THERE IS ANY JOB COLLECT STATISTICS ON THE JOBS IN
                C     QUEUE. REMOVE THE JOB FROM THE QUEUE AND FILE AN END-OF-OPERATION
                C     EVENT. IF THERE IS NO JOB COLLECT STATISTICS ON THE BUSY TIME AND
                C     MAKE THE MACHINE IDLE.
                C
0050                7 JQ=NSAVE+1
0051                  IF(NSET(NQ+JQ))8,9,10
0052                8 CALL ERROR(2)
0053                9 NCODE=NSAVE+MACH
0054                  CALL TMST(1.,NCODE)
0055                  BUS(NSAVE)=0.
0056                  RETURN
0057               10 ANQ=NSET(NQ+JQ)
0058                  CALL TMST(ANQ,NSAVE)
0059                  NCOL=NSET(MFE+JQ)
0060                  CALL RMOVE(NCOL,JQ)
0061                  IN=LOCAT(1,MFA,1)
0062                  NSET(IN)=5.
0063                  QSET(IN+3)=TNOW+QSET(IN+5)
0064                  CALL FILEM(1)
0065                  RETURN
0066                  END
```

```
FORTRAN IV G LEVEL  21                OTPUT              DATE = 73045          22/43/31

0001                  SUBROUTINE OTPUT
0002                  DIMENSION QSET(1)
0003                  COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,
                     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MIMST,MHIST,NCLCT,
                     2NEVNT,NHIST,NOQ,NURPT,NQT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,
                     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYP,NRSET,NXX,OUT,
                     4TNOW,TBEG,TFIN,NAME(6),NSET(1)
0004                  EQUIVALENCE (NSET(1),QSET(1))
0005                  COMMON/U/AINT,AJTM,MACH,NOP,XISYS,BUS(6)
0006                  DIMENSION AVQ(10),AUT(10)
                C
                C     FILE NEXT REPORT EVENT
                C
0007                  IN=LOCAT(1,MFA,1)
0008                  NSET(IN)=6
0009                  QSET(IN+1)=TNOW+48.
0010                  CALL FILEM(1)
0011                  DO 1 I=1,MACH
0012                  AA=BUS(I)
0013                  NCODE=MACH+I
0014                  CALL TMST(AA,NCODE)
0015                1 CONTINUE
                C
                C     CALCULATE STATISTICS:  AVOP      -AVERAGE OPERATION TIME
                C                            AVORD     -AVERAGE NUMBER OF ORDERS RECD
                C                            AVSLK     -AVERAGE SLACK TIME
                C                            AVQ(I)    -AVERAGE NUMBER OF JOBS IN 'Q'
                C                            AUT(I)    -AVERAGE UTILIZATION OF M/C 'I'
                C                            AVSYS     -AVERAGE NUMBER OF JOBS IN SYSTEM
                C
0016                  AVORD=QSET(NSUMA+1)/QSET(NSUMA+3)
0017                  AVOP=QSET(NSUMA+6)/QSET(NSUMA+8)
0018                  AVSLK=QSET(NSUMA+11)/QSET(NSUMA+13)
0019                  INDX=5*NCLCT+(NSUMA+1)
0020                  DO 2 I=1,MACH
0021                  AVQ(I)=QSET(INDX+1)/QSET(INDX)
0022                2 INDX=INDX+5
0023                  DO 3 I=1,MACH
0024                  AUT(I)=QSET(INDX+1)/QSET(INDX)
0025                3 INDX=INDX+5
0026                  AVSYS=QSET(INDX+1)/QSET(INDX)
0027                  IF(TNOW.NE.43.)GO TO 6
0028                  WRITE(NPRNT,4)
0029                4 FORMAT('1','      AVOP      AVORD      AVSLK      AVQ 1      AVQ 2
                     1 AVQ 3      AUT 1      AUT 2      AUT 3      AVSYS')
0030                6 WRITE(NPRNT,5)AVOP,AVORD,AVSLK,(AVQ(I),I=1,MACH),
                     2(AUT(I),I=1,MACH),AVSYS
0031                5 FORMAT(' ',10F10.4)
0032                  RETURN
0033                  END
```

```
 ECHO CHECK ON INPUT DATA

 CARD TYPE  1
   NAME       NPROJ  MON NDAY  NYR  NRUNS
RAJAGOPALAN.    1    11   25  1972    1

 CARD TYPE  2
 NPRMS NHIST NCLCT NSTAT     ID    MXX    NOQ NEVNT
     6     3     3     7    200     8     4     6

 CARD TYPE  3
                                FOR EVENTS
FIXED POINT ATTRIBUTES              1    1    3    1    3    1
FLOATING POINT ATTRIBUTES           1    1    1    1    2    1
                                FOR FILES
FIXED POINT ATTRIBUTES              3    3    3
FLOATING POINT ATTRIBUTES           3    3    3

 CARD TYPE  4
NO. OF CELLS
   13   14   13

 CARD TYPE  5
KRANK
    1    1    1    1

 CARD TYPE  6
INN
    1    1    1    1

 CARD TYPE  7
PARAMETERS
                    PARAMETER NO.    1      0.4000      0.9000      1.0000      0.0
                    PARAMETER NO.    2      0.0         0.6000      1.0000      0.0
                    PARAMETER NO.    3      0.3000      0.3000      1.0000      0.0
                    PARAMETER NO.    4      0.5000      1.0000      1.0000      0.0
                    PARAMETER NO.    5     11.0000      0.0       100.0000      0.0
                    PARAMETER NO.    6      0.6628     -4.0000      2.5000      0.2450

 CARD TYPE  8
   MSTOP JCLR  NORPT  NEP     TBEG      TFIN    NSEED
     1     1     0     4      0.0    5400.000     3

 CARD TYPE  9
SEEDS FOR RANDOM NUMBER GENERATION
  11111  33333  77777

 CARD TYPE 10
RESET TIMES FOR COLCT
   480.000   480.000   480.000

RESET TIMES FOR TMST
   480.000   480.000   480.000   480.000   480.000   480.000   480.000

RESET TIMES FOR HISTO
   480.000   480.000   480.000
```

Appendix A.6(a) Simulation Results with GASP IIP.

POINTERS

INN	KRANK	MAXNQ	MFE	MLC	MLE	NQ	NVNQ	NENQ	NQTM	MONIT
0	4	8	12	16	20	24	28	32	36	40
MCLCT	MTMST	NRSET	MHIST	JHIST	IX	NFIL				
138	141	148	155	158	207	210				

CARD TYPE 11

FILE NO.,	ATTRIBUTES		
1	4	0.0	
1	6	48.000000	

SIMULATION PROJECT NO. 1 BY RAJAGOPALAN.

DATE 11/ 25/ 1972 RUN NUMBER 1

CONTENTS OF FILE NO. 1

NSET AND QSET

1	2	9999		4
			0.0	
2	7777	1		6
			0.480000E 02	

CONTENTS OF FILE NO. 2

NSET AND QSET

THE FILE IS EMPTY

CONTENTS OF FILE NO. 3

NSET AND QSET

THE FILE IS EMPTY

CONTENTS OF FILE NO. 4

NSET AND QSET

THE FILE IS EMPTY

AVOP	AVORD	AVSLK	AVQ 1	AVQ 2	AVQ 3	AUT 1	AUT 2	AUT 3	AVSYS
1.9287	9.0000	56.4099	0.5050	2.4956	2.6536	0.4352	0.7238	0.6561	7.3861
1.9577	11.0000	48.0162	0.9718	6.2164	2.0753	0.6454	0.8619	0.7388	11.4472
1.9842	10.8333	42.0037	1.2543	8.7486	1.4156	0.6358	0.9070	0.7594	13.6999
1.9785	9.7500	36.2547	1.4066	10.6623	1.0688	0.6222	0.9310	0.6968	15.3497
1.9639	9.1000	34.1198	1.4141	9.8142	0.8778	0.6042	0.9448	0.6757	14.3062
1.9689	9.9167	36.6604	1.5881	9.1638	0.9731	0.6579	0.9501	0.7212	14.0512
1.9719	9.9286	37.2069	1.6869	9.1480	0.9204	0.6785	0.9572	0.7197	14.2088
1.9745	10.5000	36.9461	1.8679	10.1233	0.8667	0.7140	0.9626	0.7322	15.2377
1.9753	10.4444	34.8702	1.9427	10.7487	0.8189	0.7418	0.9667	0.7430	15.9565
1.9835	10.2000	33.9108	1.9405	10.6755	0.8578	0.7454	0.9700	0.7502	15.9711
1.9873	10.1364	33.8000	1.9840	10.9043	0.7905	0.7215	0.9726	0.7446	16.0378
1.9916	10.2093	33.6513	1.9177	10.9874	0.7674	0.7329	0.9750	0.7329	15.9119
1.9948	10.6538	32.9139	2.0799	11.8842	0.8983	0.7524	0.9770	0.7449	17.3573
2.0002	10.7143	30.5496	2.0990	13.3410	0.9369	0.7596	0.9786	0.7484	19.8503
1.9942	10.9667	28.0989	2.1235	15.1469	0.9309	0.7614	0.9800	0.7544	20.6976
1.9848	11.0000	24.9089	2.2355	16.9096	0.8832	0.7713	0.9813	0.7439	22.5261
1.9917	11.0294	21.2331	2.1853	18.5783	0.9395	0.7710	0.9824	0.7553	24.1579
1.9846	10.9722	17.3650	2.2011	19.9726	0.9982	0.7775	0.9834	0.7653	25.6967
1.9859	10.6579	14.4840	2.1041	21.1173	0.9529	0.7672	0.9842	0.7484	26.6699
1.9880	10.5250	11.7470	2.0203	21.6550	0.9236	0.7605	0.9850	0.7462	27.0837
1.9915	10.4762	9.0140	2.1249	22.0068	0.8978	0.7714	0.9857	0.7516	27.5371
1.9897	10.7273	7.7669	2.1588	22.5239	1.0737	0.7781	0.9864	0.7629	28.2507
1.9895	10.5652	6.9862	2.0844	23.0702	1.0545	0.7676	0.9870	0.7585	28.6748
1.9933	10.7083	5.6654	2.3738	23.5837	1.0645	0.7773	0.9875	0.7590	29.5439
1.9943	10.7000	3.5974	2.6551	24.1651	1.0805	0.7862	0.9880	0.7656	30.4407
1.9935	10.6346	2.1858	2.6360	24.7399	1.0445	0.7876	0.9885	0.7635	30.9551
1.9925	10.6481	1.3234	2.6160	25.0481	1.0848	0.7907	0.9889	0.7699	31.2980
1.9943	10.5536	0.2084	2.5780	25.2399	1.0943	0.7899	0.9893	0.7733	31.4476
1.9951	10.6207	-0.6551	2.5709	25.4105	1.1090	0.7892	0.9897	0.7783	31.6422
1.9986	10.7667	-1.1665	2.8049	25.5579	1.1879	0.7960	0.9900	0.7857	32.1348
1.9982	10.7903	-1.3860	3.0602	25.8332	1.1957	0.8006	0.9903	0.7901	32.6720
1.9972	10.8125	-2.5294	3.0857	26.3459	1.1646	0.8064	0.9906	0.7839	33.1608
1.9976	10.9091	-3.1123	2.9992	27.0324	1.1720	0.8021	0.9909	0.7841	33.7325
1.9981	10.8824	-3.9389	2.9512	27.6941	1.2339	0.8042	0.9912	0.7904	34.4519
1.9969	10.9143	-5.1298	3.0174	28.3976	1.2161	0.8098	0.9914	0.7913	35.1843
1.9973	10.9306	-6.7353	3.0921	29.0895	1.1905	0.8151	0.9917	0.7901	35.9805
1.9971	10.9189	-9.1786	3.2053	29.7831	1.1866	0.8201	0.9919	0.7885	36.7295
1.9980	10.9605	-10.0774	3.3787	30.4129	1.1458	0.8248	0.9921	0.7896	37.5235
1.9983	10.9231	-11.8132	3.4076	30.9856	1.1247	0.8293	0.9923	0.7876	38.1994
1.9982	10.9500	-12.9210	3.5122	31.5961	1.1731	0.8325	0.9925	0.7929	38.8896
1.9974	10.9146	-14.8352	3.4756	32.1204	1.1572	0.8309	0.9927	0.7932	39.3564
1.9999	10.8810	-15.5077	3.3954	32.6316	1.1464	0.8266	0.9929	0.7923	39.7508
1.9973	10.8963	-16.4086	3.3583	33.0837	1.2084	0.8305	0.9930	0.7944	40.2729
1.9983	10.9091	-17.2796	3.4199	33.4828	1.1864	0.8343	0.9932	0.7931	40.6894
1.9999	10.9556	-18.7978	3.4947	33.8997	1.1700	0.8380	0.9933	0.7915	41.1763
2.0035	10.9457	-19.7605	3.4828	34.3511	1.1587	0.8410	0.9935	0.7934	41.6189
2.0030	10.9255	-20.5807	3.4278	34.8791	1.1423	0.8406	0.9936	0.7901	42.0565
2.0014	10.9583	-21.6340	3.4792	35.4935	1.1354	0.8419	0.9938	0.7893	42.6149
2.0019	11.0306	-22.9353	3.4664	36.2073	1.1389	0.8449	0.9939	0.7889	43.4529
2.0027	11.0200	-24.1001	3.4406	36.9946	1.1140	0.8459	0.9940	0.7872	44.1841
2.0032	10.9706	-25.3645	3.4332	37.7163	1.0979	0.8465	0.9941	0.7852	44.8662
2.0010	10.9808	-26.7563	3.4153	38.3548	1.0954	0.8473	0.9942	0.7868	45.4867
2.0016	10.9528	-28.4016	3.3956	39.0521	1.1140	0.8438	0.9943	0.7906	46.1817
2.0004	10.9167	-29.9140	3.3847	39.7414	1.0951	0.8401	0.9945	0.7868	46.8188
1.9990	10.9455	-31.2625	3.3733	40.3336	1.0774	0.8393	0.9946	0.7929	47.3985
2.0021	10.9643	-32.9110	3.4598	40.9502	1.0632	0.8421	0.9947	0.7862	48.0848
2.0023	10.9561	-34.8494	3.5690	41.5241	1.0538	0.8449	0.9947	0.7790	48.6880
2.0027	10.9655	-36.2517	3.6748	42.0705	1.0595	0.8476	0.9948	0.7814	49.4067
2.0018	10.9576	-37.4365	3.7168	42.5786	1.0844	0.8502	0.9949	0.7847	50.0160

2.0014	10.9167	-38.5100	3.6976	43.1040	1.0679	0.8483	0.9950	0.7831	50.4611
2.0012	10.9180	-39.8519	3.6597	43.5852	1.0571	0.8469	0.9951	0.7826	50.9221
2.0010	10.9194	-40.8300	3.6153	44.1195	1.0527	0.8472	0.9952	0.7819	51.4257
1.9989	10.9127	-41.6844	3.5803	44.6777	1.0459	0.8462	0.9952	0.7814	51.9077
1.9996	10.8826	-42.7815	3.5668	45.1387	1.0487	0.8470	0.9953	0.7821	52.3764
2.0006	10.8538	-43.8041	3.5308	45.5667	1.0408	0.8472	0.9954	0.7814	52.7563
1.9998	10.8636	-45.0977	3.4893	45.9851	1.0332	0.8441	0.9955	0.7826	53.1181
1.9994	10.8657	-46.3568	3.4603	46.4415	1.0285	0.8384	0.9955	0.7819	53.5429
2.0004	10.8897	-47.5611	3.4489	46.9306	1.0313	0.8408	0.9956	0.7815	54.0152
2.0001	10.9058	-48.5664	3.4322	47.4762	1.0420	0.8420	0.9957	0.7842	54.5748
2.0002	10.9000	-49.1280	3.4081	47.9757	1.0624	0.8437	0.9957	0.7871	55.0848
1.9996	10.8873	-49.8928	3.4287	48.4732	1.0519	0.8459	0.9958	0.7861	55.5826
1.9999	10.9167	-51.0335	3.4313	48.9756	1.0501	0.8474	0.9958	0.7867	56.0438
1.9987	10.8973	-52.5550	3.4094	49.4250	1.0535	0.8456	0.9959	0.7859	56.4816
1.9984	10.8851	-53.7425	3.3866	49.8514	1.0443	0.8448	0.9960	0.7839	56.8947
1.9985	10.8667	-54.2396	3.3506	50.2283	1.0387	0.8434	0.9960	0.7827	57.2216
1.9995	10.9276	-55.3878	3.3726	50.6837	1.0476	0.8437	0.9961	0.7845	57.7335
1.9991	10.9749	-55.9311	3.3851	51.1875	1.1831	0.8459	0.9961	0.7873	58.3726
1.9992	10.9808	-57.0127	3.4142	51.7357	1.2531	0.8477	0.9962	0.7900	58.9685
1.9987	11.0000	-58.4600	3.4480	52.3602	1.2444	0.8497	0.9962	0.7887	59.6469
1.9995	10.9937	-59.4262	3.4715	52.9863	1.2335	0.8515	0.9963	0.7886	60.3373
1.9989	11.0309	-60.2539	3.5473	53.6184	1.2412	0.8534	0.9963	0.7912	60.9560
1.9995	11.0061	-61.5325	3.5913	54.2344	1.2322	0.8552	0.9963	0.7904	61.7527
1.9993	10.9639	-62.8576	3.5792	54.9094	1.2294	0.8539	0.9964	0.7902	62.3450
1.9988	10.9762	-64.0025	3.5477	55.5046	1.2215	0.8516	0.9964	0.7903	62.9135
1.9992	10.9294	-65.1510	3.5183	56.0730	1.2093	0.8505	0.9965	0.7889	63.4377
1.9985	10.8895	-66.7894	3.4788	56.5463	1.1946	0.8474	0.9965	0.7850	63.8455
1.9978	10.9060	-68.3675	3.4653	57.0246	1.1913	0.8464	0.9966	0.7849	64.3075
1.9972	10.8864	-69.3854	3.4288	57.5146	1.1825	0.8461	0.9966	0.7826	64.7340
1.9972	10.9382	-71.1466	3.4272	57.9786	1.1747	0.8456	0.9966	0.7822	65.2140
1.9988	10.9556	-71.8288	3.4494	58.5066	1.1705	0.8473	0.9967	0.7836	65.7447
1.9982	10.9780	-73.0115	3.5251	59.0487	1.1691	0.8490	0.9967	0.7825	66.3533
1.9972	10.9728	-74.5649	3.5737	59.6203	1.1615	0.8506	0.9967	0.7841	66.9917
1.9990	10.9677	-75.0240	3.5615	60.1959	1.1539	0.8522	0.9968	0.7833	67.5661
1.9982	11.0000	-76.4777	3.5525	60.8401	1.1470	0.8519	0.9968	0.7816	68.1668
1.9973	10.9842	-77.5730	3.5203	61.4959	1.1367	0.8492	0.9968	0.7802	68.7457
1.9976	10.9896	-78.4940	3.4903	62.1479	1.1467	0.8496	0.9969	0.7808	69.4128
1.9969	11.0052	-79.4561	3.4800	62.7737	1.1349	0.8497	0.9969	0.7791	70.0136
1.9974	11.0000	-80.7762	3.4761	63.4584	1.1271	0.8497	0.9969	0.7776	70.6731
1.9993	10.9697	-81.3689	3.4501	64.1734	1.1281	0.8480	0.9970	0.7773	71.3726
1.9993	10.9850	-83.1998	3.4582	64.8683	1.1280	0.8495	0.9970	0.7779	72.0798
2.0009	11.0099	-84.6221	3.4515	65.6449	1.1263	0.8508	0.9970	0.7782	72.8357
2.0009	11.0000	-85.9436	3.4280	66.5166	1.1178	0.8497	0.9971	0.7790	73.6854
2.0004	10.9951	-87.6740	3.4081	67.3463	1.1078	0.8480	0.9971	0.7774	74.4233
2.0011	11.0192	-89.6024	3.4186	68.1538	1.1059	0.8495	0.9971	0.7785	75.2987
2.0010	11.0296	-90.7784	3.4219	68.9813	1.0997	0.8500	0.9971	0.7780	76.1298
2.0001	11.0330	-92.1011	3.4107	69.8606	1.1124	0.8500	0.9972	0.7796	77.0125
1.9999	11.0701	-93.4764	3.4360	70.7679	1.1092	0.8514	0.9972	0.7791	77.8610
1.9987	11.0463	-95.0838	3.4309	71.6398	1.1155	0.8512	0.9972	0.7796	78.8446
1.9999	11.0321	-97.6426	3.4301	72.5221	1.1113	0.8525	0.9973	0.7787	79.6923
1.9996	11.0409	-99.1814	3.4168	73.3982	1.1057	0.8529	0.9973	0.7785	80.5670
1.9999	10.9955	-100.4429	3.3939	74.2259	1.1008	0.8521	0.9973	0.7773	81.3544
1.9994	11.0089	-101.8657	3.3922	75.0136	1.1042	0.8534	0.9973	0.7780	82.1428

GASP SUMMARY REPORT

SIMULATION PROJECT NO. 1 BY RAJAGOPALAN.

DATE 11/ 25/ 1972 RUN NUMBER 1

PARAMETER NO.	1	0.4000	0.9000	1.0000	0.0
PARAMETER NO.	2	0.0	0.6000	1.0000	0.0
PARAMETER NO.	3	0.3000	0.3000	1.0000	0.0
PARAMETER NO.	4	0.5000	1.0000	1.0000	0.0
PARAMETER NO.	5	11.0000	0.0	100.0000	0.0
PARAMETER NO.	6	0.6628	-4.0000	2.5000	0.2450

GENERATED DATA

CODE	MEAN	STD.DEV.	MIN.	MAX.	OBS.
1	10.9911	3.3280	1.0000	20.0000	226
2	1.9995	0.5080	0.8373	4.4826	7255
3	-103.7438	122.4713	-547.3711	66.9372	2315

TIME GENERATED DATA

CODE	MEAN	STD.DEV.	MIN.	MAX.	TOTAL TIME
1	3.3834	3.3207	0.0	15.0000	5400.0000
2	75.4209	43.5387	0.0	173.0000	5400.0000
3	1.0998	1.7659	0.0	15.0000	5400.0000
4	0.8527	0.3544	0.0	1.0000	5400.0000
5	0.9973	0.0516	0.0	1.0000	5376.0000
6	0.7769	0.4163	0.0	1.0000	5398.4531
7	82.5378	43.7139	0.0	186.0000	5400.0000

GENERATED FREQUENCY DISTRIBUTIONS

CODE	HISTOGRAMS														
1	2	3	6	19	35	53	67	22	12	6	1	0	0	0	0
2	0	0	0	34	247	838134514651295				909	564	298	114	88	28
	30														
3	1733	22	16	14	15	24	16	7	13	10	14	9	13	13	396

FILE PRINTOUT, FILE NO. 1

AVERAGE NUMBER IN FILE WAS 4.6256
STD. DEV. 0.5861
MAXIMUM 6

CONTENTS OF FILE NO. 1

NSET AND QSET

77	5	9999	5	2	3	
			0.540012E 04	0.513600E 04		
5	114	77	5	1	2	

			0.540094E 04	0.513600E 04		
114	104	5	5	0		1
			0.540121E 04	0.547200E 04		
104	72	114	6			
			0.542400E 04			
72	7777	104	4			
			0.542400E 04			

FILE PRINTOUT, FILE NO. 2

AVERAGE NUMBER IN FILE WAS	3.3834
STD. DEV.	3.3207
MAXIMUM	15

CONTENTS OF FILE NO. 2

			NSET AND QSET		
110	12	9999	0	0	1
			0.540000E 04	0.547200E 04	0.226262E 01
12	7777	110	0	0	1
			0.540000E 04	0.547200E 04	0.286506E 01

FILE PRINTOUT, FILE NO. 3

AVERAGE NUMBER IN FILE WAS	75.4209
STD. DEV.	43.5387
MAXIMUM	173

CONTENTS OF FILE NO. 3

			NSET AND QSET		
178	135	9999	0	2	2
			0.507487E 04	0.513600E 04	0.210166E 01
135	57	178	0	2	2
			0.507890E 04	0.513600E 04	0.173826E 01
57	48	135	0	2	2
			0.508646E 04	0.487200E 04	0.164114E 01
48	113	57	0	0	2
			0.508800E 04	0.516000E 04	0.137725E 01
113	102	48	0	0	2
			0.508800E 04	0.516000E 04	0.170024E 01
102	115	113	0	0	2
			0.508800E 04	0.516000E 04	0.278543E 01
115	39	102	0	0	2
			0.508800E 04	0.516000E 04	0.159769E 01
39	85	115	0	0	2
			0.508800E 04	0.516000E 04	0.248159E 01
85	13	39	0	0	2
			0.508800E 04	0.516000E 04	0.226071E 01
13	37	85	0	0	2

			0.540000E 04	0.547200E 04	0.222028E 01
47	55	150	0	0	2
			0.540000E 04	0.547200E 04	0.110180E 01
55	7777	47	0	0	2
			0.540000E 04	0.547200E 04	0.185241E 01

FILE PRINTOUT, FILE NO. 4

AVERAGE NUMBER IN FILE WAS 1.0998
STD. DEV. 1.7659
MAXIMUM 15

CONTENTS OF FILE NO. 4

NSET AND QSET

86	149	9999	0	0	3					
			0.540000E 04	0.547200E 04	0.292620E 01					
149	7777	86	0	0	3					
			0.540000E 04	0.547200E 04	0.339513E 01					
1.9995	10.9911	-103.7438	3.3834	75.4209	1.0998	0.8527	0.9973	0.7770	82.5378	

APPENDIX B

GASP IIP SUBPROGRAMS

```
$$PARM
ARAND    CSECT                                                          ARA00010
         USING  *,15          INITIAL LINKAGE.                          ARA00020
         STM    2,8,28(13)                                              ARA00030
         L      2,0(1)        LOAD ADDRESS OF THE VARIABLE PASSED.      ARA00040
         L      3,=A(IX)      COMPUTE ADDRESS OF THE SEED STORED IN NSET ARA00050
         L      7,0(2)                                                  ARA00060
         A      7,0(3)                                                  ARA00070
         S      7,=F'1'                                                 ARA00080
         M      6,=F'4'                                                 ARA00090
         L      8,=A(NSET)                                              ARA00100
         AR     7,8                                                     ARA00110
         L      5,A           COMPUTE NEXT INTEGER RANDOM NUMBER        ARA00120
         M      4,0(7)        AS NSET(IX+I)=A*NSET(IX+I)(MOD P).        ARA00130
         D      4,P                                                     ARA00140
         ST     4,0(7)                                                  ARA00150
         SRL    4,7           COMPUTE NEXT REAL RANDOM NUMBER R AND     ARA00160
         A      4,CHAR        STORE 1-R IN THE FLOATING POINT REGISTER 0 ARA00170
         ST     4,WORD                                                  ARA00180
         LE     0,ONE                                                   ARA00190
         SE     0,WORD                                                  ARA00200
         LM     2,8,28(13)    TERMINAL LINKAGE.                         ARA00210
         BR     14                                                      ARA00220
CHAR     DC     F'1073741824' CONSTANTS. CHAR FIRST SO A IS ON DOUBLE   ARA00230
A        DC     F'16807'      WORD BOUNDARY. MAKES LM INSTRUCTION FASTER ARA00240
P        DC     F'2147483647'                                           ARA00250
ONE      DC     F'1.'                                                   ARA00260
WORD     DS     F                                                       ARA00270
         COM                                                            ARA00280
IX       DS     59F           IX IS CONSIDERED AS EQUIVALENT TO 59      ARA00290
NSET     DS     F             WORDS TO COMPUTE THE ADDRESS OF NSET.     ARA00300
         END                                                            ARA00310
```

```
      SUBROUTINE COLCT (X,N)                                            COL00010
      DIMENSION QSET(1)                                                 COL00020
      COMMON IX,IM,INN,IQ,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,COL00030
     1MUN,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,    COL00040
     2NEVNT,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, COL00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  COL00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    COL00070
      EQUIVALENCE (NSET(1),QSET(1))                                     COL00080
C                                                                       COL00090
C     TO COMPUTE POINT ESTIMATES FOR THE VARIABLE SPECIFIED.            COL00100
C                                                                       COL00110
      IF (N) 2,2,1                                                      COL00120
    2 CALL ERROR(60)                                                    COL00130
    1 IF (N- NCLCT) 3,3,2                                               COL00140
    3 INDX=5*(N-1)+(NSUMA+1)                                            COL00150
      QSET(INDX)=QSET(INDX)+X                                           COL00160
      QSET(INDX+1)=QSET(INDX+1)+X*X                                     COL00170
      QSET(INDX+2)=QSET(INDX+2)+1.0                                     COL00180
      QSET(INDX+3)=AMIN1(QSET(INDX+3),X)                                COL00190
      QSET(INDX+4)=AMAX1(QSET(INDX+4),X)                                COL00200
      RETURN                                                            COL00210
      END                                                               COL00220
```

```
      SUBROUTINE DATAX                                                  DAT00010
      DIMENSION QSET(1)                                                 DAT00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,IU,MFA,MSTOP,DAT00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,    DAT00040
     2NEVNT,NHIST,NQQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRUR,NEP, DAT00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NPSET,NXX,OUT,  DAT00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    DAT00070
      EQUIVALENCE (NSET(1),QSET(1))                                     DAT00080
  101 FORMAT(6A2,I4,I2,I2,I4,I4)                                        DAT00090
  102 FORMAT(1H1,29X,' SIMULATION PROJECT NO.',I4,2X,'BY',2X,6A2//,     DAT00100
     130X,4HDATE,I3,1H/,I3,1H/,I5,12X,10HRUN NUMBER,I5)                 DAT00110
  103 FORMAT(16I5)                                                      DAT00120
  104 FORMAT(4I5,2F10.3,I4)                                             DAT00130
  105 FORMAT(11I7)                                                      DAT00140
  106 FORMAT(4F10.4)                                                    DAT00150
  107 FORMAT(20X,14H PARAMETER NO.,I5,4F12.4)                           DAT00160
  108 FORMAT(' PARAMETERS')                                             DAT00170
  765 FORMAT(' ',6A2,I4,I6,I5,I6,I5)                                    DAT00180
  844 FORMAT(' ',8G16.8)                                                DAT00190
  855 FORMAT(' FILE NO., ATTRIBUTES')                                   DAT00200
  876 FORMAT('    NAME      NPROJ  MON NDAY  NYR  NRUNS')               DAT00210
  944 FORMAT(' ',15I7)                                                  DAT00220
  954 FORMAT('  NPRMS NHIST NCLCT NSTAT    ID    MXX    NQQ NEVNT')    DAT00230
  955 FORMAT(' ',4I6,2F10.3,I6)                                         DAT00240
  965 FORMAT(' NO. OF CELLS')                                           DAT00250
  966 FORMAT(' SEEDS FOR RANDOM NUMBER GENERATION')                     DAT00260
  967 FORMAT(' ',20I5)                                                  DAT00270
  968 FORMAT(' ',20I6)                                                  DAT00280
  976 FORMAT(' KRANK')                                                  DAT00290
  977 FORMAT('    MSTOP JCLR  NORPT  NEP     TBEG      TFIN   NSEED')  DAT00300
  978 FORMAT(' INN')                                                    DAT00310
  987 FORMAT('1',' ECHO CHECK ON INPUT DATA')                           DAT00320
 1110 FORMAT(8G10.4)                                                    DAT00330
 1170 FORMAT(' ',15I8)                                                  DAT00340
 1190 FORMAT('-POINTERS')                                               DAT00350
 1200 FORMAT(' ',T7,'INN   KRANK   MAXNQ     MFE     MLC     MLE     NQDAT00360
     1    NVNQ    NENQ   NQTM   MONIT      IM     IMM   NPARM   NSUMA')DAT00370
 1220 FORMAT(' ',T5,'MCLCT   MTMST   NRSET   MHIST   JHIST      IX    NFDAT00380
     1IL')                                                              DAT00390
 1240 FORMAT(8F10.3)                                                    DAT00400
 1250 FORMAT(' RESET TIMES FOR COLCT')                                  DAT00410
 1260 FORMAT(' ',8F10.3)                                                DAT00420
 1270 FORMAT('-RESET TIMES FOR TMST')                                   DAT00430
 1280 FORMAT('-RESET TIMES FOR HISTO')                                  DAT00440
 1330 FORMAT(' ',T35,'FOR EVENTS')                                      DAT00450
 1340 FORMAT(' ',T35,'FOR FILES')                                       DAT00460
 1350 FORMAT('0',' CARD TYPE',I3)                                       DAT00470
 1360 FORMAT(' ','FIXED POINT ATTRIBUTES',(T35,16I5))                   DAT00480
 1370 FORMAT(' ','FLOATING POINT ATTRIBUTES',(T35,16I5))                DAT00490
      IF(NOT)23,1,2                                                     DAT00500
C                                                                       DAT00510
C*****NEP IS A CONTROL VARIABLE FOR DETERMINING THE STARTING CARD       DAT00520
C*****TYPE FOR MULTIPLE RUN PROBLEMS. THE VALUE OF NEP SPECIFIES THE    DAT00530
C*****STARTING CARD TYPE.                                               DAT00540
C                                                                       DAT00550
    2 NT=NEP                                                            DAT00560
      GO TO (1,5,1320,6,1290,42,8,43,1310,1230,299,15,1300),NT          DAT00570
   23 CALL ERROR(100)                                                   DAT00580
```

```
    1 NOT=1                                                                 DAT00590
      NRUN=1                                                                DAT00600
C                                                                           DAT00610
C*****DATA CARD TYPE ONE.                                                   DAT00620
C                                                                           DAT00630
      READ(NCRDR,101) NAME,NPROJ,MON,NDAY,NYR,NRUNS                         DAT00640
      WRITE(NPRNT,987)                                                      DAT00650
      NTYPE=1                                                               DAT00660
      WRITE(NPRNT,1350)NTYPE                                                DAT00670
      WRITE(NPRNT,876)                                                      DAT00680
      WRITE(NPRNT,765)NAME,NPROJ,MON,NDAY,NYR,NRUNS                         DAT00690
      IF(NRUNS)30,30,5                                                      DAT00700
   30 STOP                                                                  DAT00710
C                                                                           DAT00720
C*****DATA CARD TYPE TWO.                                                   DAT00730
C                                                                           DAT00740
    5 READ(NCRDR,103)NPRMS,NHIST,NCLCT,NSTAT,ID,MXX,NOQ,NEVNT               DAT00750
      NXX=MXX-2                                                             DAT00760
      NTYPE=2                                                               DAT00770
      WRITE(NPRNT,1350)NTYPE                                                DAT00780
      WRITE(NPRNT,954)                                                      DAT00790
      WRITE(NPRNT,968)NPRMS,NHIST,NCLCT,NSTAT,ID,MXX,NOQ,NEVNT              DAT00800
C                                                                           DAT00810
C     CALCULATION OF POINTERS.                                              DAT00820
C                                                                           DAT00830
      INN=0                                                                 DAT00840
      KRANK=NOQ                                                             DAT00850
      MAXNQ=2*NOQ                                                           DAT00860
      MFE=3*NOQ                                                             DAT00870
      MLC=4*NOQ                                                             DAT00880
      MLE=5*NOQ                                                             DAT00890
      NQ=6*NOQ                                                              DAT00900
      NVNQ=7*NOQ                                                            DAT00910
      NENQ=8*NOQ                                                            DAT00920
      NQTM=9*NOQ                                                            DAT00930
      MONIT=10*NOQ                                                          DAT00940
      IM=MONIT+NEVNT                                                        DAT00950
      IMM=IM+NEVNT+NOQ-1                                                    DAT00960
      NPARM=IMM+NEVNT+NOQ-1                                                 DAT00970
      NSUMA=NPARM+(4*NPRMS)                                                 DAT00980
      MCLCT=5*(NSTAT+NCLCT)+NSUMA                                           DAT00990
      MTMST=MCLCT+NCLCT                                                     DAT01000
      NRSET=MTMST+NSTAT                                                     DAT01010
      MHIST=NRSET+NSTAT                                                     DAT01020
      JHIST=MHIST+NHIST                                                     DAT01030
C                                                                           DAT01040
C     DATA CARD TYPE THREE. READ IN THE NUMBER OF FIXED POINT ATTRIBUTES    DAT01050
C     AND FLOATING POINT ATTRIBUTES FOR EACH EVENT IN FILE 1 AND FOR        DAT01060
C     JQ=2,3....NOQ., WHERE JQ IS THE FILE NUMBER.                          DAT01070
C                                                                           DAT01080
 1320 L1=NEVNT+1                                                            DAT01090
      L2=NEVNT+NOQ-1                                                        DAT01100
      READ(NCRDR,103)(NSET(IM+I),I=1,L2)                                    DAT01110
      READ(NCRDR,103)(NSET(IMM+I),I=1,L2)                                   DAT01120
      NTYPE=3                                                               DAT01130
      WRITE(NPRNT,1350)NTYPE                                                DAT01140
      WRITE(NPRNT,1330)                                                     DAT01150
      WRITE(NPRNT,1360)(NSET(IM+I),I=1,NEVNT)                               DAT01160
      WRITE(NPRNT,1370)(NSET(IMM+I),I=1,NEVNT)                              DAT01170
      IF(NOQ.EQ.1)GO TO 1380                                                DAT01180
```

```
      WRITE(NPRNT,1340)                                                 DAT01190
      WRITE(NPRNT,1360)(NSET(IM+I),I=L1,L2)                             DAT01200
      WRITE(NPRNT,1370)(NSET(IMM+I),I=L1,L2)                            DAT01210
C                                                                       DAT01220
C*****DATA CARD TYPE FOUR  IS USED ONLY IF NHIST IS GREATER THAN ZERO   DAT01230
C*****SPECIFY THE NUMBER OF CELLS IN HISTOGRAMS NOT INCLUDING END CELLS.DAT01240
C                                                                       DAT01250
 1380 IF(NHIST) 41,41,6                                                 DAT01260
    6 READ(NCRDR,103)(NSET(JHIST+I),I=1,NHIST)                          DAT01270
      NTYPE=4                                                           DAT01280
      WRITE(NPRNT,1350)NTYPE                                            DAT01290
      WRITE(NPRNT,965)                                                  DAT01300
      WRITE(NPRNT,967)(NSET(JHIST+I),I=1,NHIST)                         DAT01310
C                                                                       DAT01320
C*****DATA CARD TYPE FIVE                                              DAT01330
C*****SPECIFY KRANK=RANKING ROW.                                       DAT01340
C                                                                       DAT01350
C                                                                       DAT01360
C*****KRANK = 1,2,....IM FOR ROW KRANK IN QSET.                        DAT01370
C*****KRANK = 101,102,....,100+IMM FOR ROW (KRANK-100) IN NSET.        DAT01380
C                                                                       DAT01390
      NSUM=0                                                            DAT01400
      DO 1120 I=1,NHIST                                                 DAT01410
 1120 NSUM=NSUM+NSET(JHIST+I)+2                                         DAT01420
      LU=NSUM+NHIST                                                     DAT01430
      GO TO 1290                                                        DAT01440
   41 LU=0                                                              DAT01450
 1290 READ(NCRDR,103)(NSET(KRANK+I),I=1,NOQ)                            DAT01460
      NTYPE=5                                                           DAT01470
      WRITE(NPRNT,1350)NTYPE                                            DAT01480
      WRITE(NPRNT,976)                                                  DAT01490
      WRITE(NPRNT,967)(NSET(KRANK+I),I=1,NOQ)                           DAT01500
C                                                                       DAT01510
C*****DATA CARD TYPE SIX                                               DAT01520
C*****SPECIFY INN=1 FOR FIFO, INN=2 FOR LIFO<                          DAT01530
C                                                                       DAT01540
   42 READ(NCRDR,103)(NSET(INN+I),I=1,NOQ)                              DAT01550
      NTYPE=6                                                           DAT01560
      WRITE(NPRNT,1350)NTYPE                                            DAT01570
      WRITE(NPRNT,978)                                                  DAT01580
      WRITE(NPRNT,967)(NSET(INN+I),I=1,NOQ)                             DAT01590
      IF(NPRMS) 23,43,8                                                 DAT01600
    8 INDX=NPARM                                                        DAT01610
C                                                                       DAT01620
C*****DATA CARD TYPE  7  IS USED ONLY IF NPRMS IS GREATER THAN ZERO<    DAT01630
C                                                                       DAT01640
      NTYPE=7                                                           DAT01650
      WRITE(NPRNT,1350)NTYPE                                            DAT01660
      WRITE(NPRNT,108)                                                  DAT01670
      DO 9 I=1,NPRMS                                                    DAT01680
      READ(NCRDR,106)(QSET(INDX+J),J=1,4)                               DAT01690
      WRITE(NPRNT,107) I,(QSET(INDX+J),J=1,4)                           DAT01700
      INDX=INDX+4                                                       DAT01710
    9 CONTINUE                                                          DAT01720
C                                                                       DAT01730
C*****DATA CARD TYPE EIGHT. THE NEP VALUE IS FOR THE NEXT RUN.         DAT01740
C                                                                       DAT01750
   43 READ (NCRDR,104) MSTOP,JCLR,NORPT,NEP,TBEG,TFIN,NSEED            DAT01760
      NTYPE=8                                                           DAT01770
      WRITE(NPRNT,1350)NTYPE                                            DAT01780
```

```
      WRITE(NPPNT,977)                                                  DAT01790
      WRITE(NPRNT,955)MSTOP,JCLR,NORPT,NEP,TBEG,TFIN,NSEED             DAT01800
      IX=JHIST+LU                                                       DAT01810
      NFIL=IX+NSEED                                                     DAT01820
C                                                                       DAT01830
C*****DATA CARD TYPE 9. READ IN SEED VALUES FOR RANDOM NUMBER GENERATOR DAT01840
C                                                                       DAT01850
 1310 IF(NSEED)23,26,27                                                 DAT01860
   27 READ(NCRDR,105)(NSET(IX+I),I=1,NSEED)                             DAT01870
      NTYPE=9                                                           DAT01880
      WRITE(NPRNT,1350)NTYPE                                            DAT01890
      WRITE(NPRNT,966)                                                  DAT01900
      WRITE(NPRNT,944)(NSET(IX+I),I=1,NSEED)                            DAT01910
      TNOW=TBEG                                                         DAT01920
      DO 142 J=1,NDQ                                                    DAT01930
  142 QSET(NQTM+J)=TNOW                                                 DAT01940
   26 JMNIT=0                                                           DAT01950
C                                                                       DAT01960
C*****DATA CARD TYPE 10                                                 DAT01970
C*****READ IN THE VALUES OF RESET TIMES FOR DIFFERENT VARIABLES         DAT01980
C                                                                       DAT01990
 1230 NTYPE=10                                                          DAT02000
      WRITE(NPRNT,1350)NTYPE                                            DAT02010
      IF(NCLCT.EQ.0)GO TO 1231                                          DAT02020
      READ(NCRDR,1240)(QSET(MCLCT+I),I=1,NCLCT)                         DAT02030
      WRITE(NPRNT,1250)                                                 DAT02040
      WRITE(NPRNT,1260)(QSET(MCLCT+I),I=1,NCLCT)                        DAT02050
 1231 IF(NSTAT.EQ.0)GO TO 1232                                          DAT02060
      READ(NCRDR,1240)(QSET(MTMST+I),I=1,NSTAT)                         DAT02070
      WRITE(NPRNT,1270)                                                 DAT02080
      WRITE(NPRNT,1260)(QSET(MTMST+I),I=1,NSTAT)                        DAT02090
 1232 IF(NHIST.EQ.0)GO TO 1233                                          DAT02100
      READ(NCRDR,1240)(QSET(MHIST+I),I=1,NHIST)                         DAT02110
      WRITE(NPRNT,1280)                                                 DAT02120
      WRITE(NPRNT,1260)(QSET(MHIST+I),I=1,NHIST)                        DAT02130
C                                                                       DAT02140
C*****PRINT POINTER VALUES.                                             DAT02150
C                                                                       DAT02160
 1233 WRITE(NPRNT,1190)                                                 DAT02170
      WRITE(NPRNT,1200)                                                 DAT02180
      WRITE(NPRNT,1170)INN,KRANK,MAXNQ,MFE,MLC,MLE,NQ,NVNQ,NENQ,NQTM,  DAT02190
     1MONIT,IM,IMM,NPARM,NSUMA                                          DAT02200
      WRITE(NPRNT,1220)                                                 DAT02210
      WRITE(NPRNT,1170)MCLCT,MTMST,NRSET,MHIST,JHIST,IX,NFIL            DAT02220
      NTYPE=11                                                          DAT02230
      WRITE(NPRNT,1350)NTYPE                                            DAT02240
      WRITE(NPRNT,855)                                                  DAT02250
      IF(TNOW.EQ.TBEG)MFA=1                                             DAT02260
C                                                                       DAT02270
C*****DATA CARD TYPE 11                                                 DAT02280
C*****INITIALIZE NSET AND QSET BY JQ EQUAL TO A NEGATIVE VALUE ON FIRST DAT02290
C     EVENT CARD.                                                       DAT02300
C*****READ IN INITIAL EVENTS. END INITIAL EVENTS AND ENTITIES WITH JQ   DAT02310
C*****EQUAL TO ZERO.                                                    DAT02320
C                                                                       DAT02330
  299 DO 300 JS=1,ID                                                    DAT02340
      L1=(MFA-1)*MXX+(NFIL+1)                                           DAT02350
      READ(NCRDR,1110)JQ,NSET(L1)                                       DAT02360
      IF (JQ)44,15,320                                                  DAT02370
   44 CALL SET                                                          DAT02380
```

```
      GO TO 300                                                          DAT02390
  320 I1=NEVNT+JQ-1                                                      DAT02400
      IF(JQ.EQ.1)I1=NSET(L1)                                             DAT02410
      I2=NSET(IM+I1)                                                     DAT02420
      I3=NSET(IMM+I1)                                                    DAT02430
      NSUM=I2+I3                                                         DAT02440
      IF(NXX-NSUM)23,310,310                                             DAT02450
  310 L2=L1+I2-1                                                         DAT02460
      L3=L2+1                                                            DAT02470
      L4=L2+I3                                                           DAT02480
      IF(I2.EQ.0)GO TO 311                                               DAT02490
      IF(I3.EQ.0)GO TO 312                                               DAT02500
      READ(NCRDR,1110)(NSET(I),I=L1,L2),(QSET(J),J=L3,L4)                DAT02510
      WRITE(NPRNT,844)JQ,(NSET(I),I=L1,L2),(QSET(J),J=L3,L4)             DAT02520
      GO TO 313                                                          DAT02530
  311 READ(NCRDR,1110)(QSET(J),J=L3,L4)                                  DAT02540
      WRITE(NPRNT,844)JQ,(QSET(J),J=L3,L4)                               DAT02550
      GO TO 313                                                          DAT02560
  312 READ(NCRDR,1110)(NSET(I),I=L1,L2)                                  DAT02570
      WRITE(NPRNT,844)JQ,(NSET(I),I=L1,L2)                               DAT02580
  313 CALL FILEM(JQ)                                                     DAT02590
  300 CONTINUE                                                           DAT02600
C                                                                        DAT02610
C*****JCLR BE POSITIVE FOR INITIALIZATION OF STORAGE ARRAYS.             DAT02620
C                                                                        DAT02630
   15 IF( JCLR )1300,1300,10                                             DAT02640
   10 IF(NCLCT)23,110,116                                                DAT02650
  116 INDX=NSUMA+1                                                       DAT02660
      DO 18 I=1,NCLCT                                                    DAT02670
      QSET(INDX)=0.0                                                     DAT02680
      QSET(INDX+1)=0.0                                                   DAT02690
      QSET(INDX+2)=0.0                                                   DAT02700
      QSET(INDX+3)=1.0E20                                                DAT02710
      QSET(INDX+4)=-1.0E20                                               DAT02720
      INDX=INDX+5                                                        DAT02730
   18 CONTINUE                                                           DAT02740
  110 IF(NSTAT)23,111,117                                                DAT02750
  117 INDX=5*NCLCT+(NSUMA+1)                                             DAT02760
      DO 360 I=1,NSTAT                                                   DAT02770
      QSET(INDX)=TNOW                                                    DAT02780
      QSET(INDX+1)=0.0                                                   DAT02790
      QSET(INDX+2)=0.0                                                   DAT02800
      QSET(INDX+3)=1.0E20                                                DAT02810
      QSET(INDX+4)=-1.0E20                                               DAT02820
      INDX=INDX+5                                                        DAT02830
  360 CONTINUE                                                           DAT02840
  111 IF(NHIST)23,1300,118                                               DAT02850
  118 LL=NHIST+1                                                         DAT02860
      DO 390 I=LL,LU                                                     DAT02870
  390 NSET(JHIST+I)=0                                                    DAT02880
C                                                                        DAT02890
C*****PRINT OUT PROGRAM IDENTIFICATION INFORMATION.                      DAT02900
C                                                                        DAT02910
 1300 WRITE(NPRNT,102) NPROJ,NAME,MON,NDAY,NYR,NRUN                      DAT02920
      RETURN                                                             DAT02930
      END                                                                DAT02940
```

```
DRAND    CSECT                                                          DRA00010
         USING  *,15          INITIAL LINKAGE.                          DRA00020
         STM    2,8,28(13)                                              DRA00030
         L      2,0(1)        LOAD ADDRESS OF THE VARIABLE PASSED.      DRA00040
         L      3,=A(IX)      COMPUTE ADDRESS OF THE SEED STORED IN NSET DRA00050
         L      7,0(2)                                                  DRA00060
         A      7,0(3)                                                  DRA00070
         S      7,=F'1'                                                 DRA00080
         M      6,=F'4'                                                 DRA00090
         L      8,=A(NSET)                                              DRA00100
         AR     7,8                                                     DRA00110
         L      5,A           COMPUTE NEXT INTEGER RANDOM NUMBER        DRA00120
         M      4,0(7)        AS NSET(IX+I)=A*NSET(IX+I)(MOD P).        DRA00130
         D      4,P                                                     DRA00140
         ST     4,0(7)                                                  DRA00150
         SRL    4,7           COMPUTE NEXT REAL RANDOM NUMBER AND STORE DRA00160
         A      4,CHAR        IT IN THE FLOATING POINT REGISTER 0.      DRA00170
         ST     4,WORD                                                  DRA00180
         LE     0,WORD                                                  DRA00190
         LM     2,8,28(13)    TERMINAL LINKAGE.                         DRA00200
         BR     14                                                      DRA00210
CHAR     DC     F'1073741824' CONSTANTS. CHAR FIRST SO A IS ON DOUBLE   DRA00220
A        DC     F'16807'      WORD BOUNDARY. MAKES LM INSTRUCTION FASTER DRA00230
P        DC     F'2147483647'                                           DRA00240
WORD     DS     F                                                       DRA00250
         COM                                                            DRA00260
IX       DS     59F           IX IS CONSIDERED  AS EQUIVALENT TO 59     DRA00270
NSET     DS     F             WORDS TO COMPUTE THE ADDRESS OF NSET.     DRA00280
         END                                                            DRA00290
```

```
      FUNCTION ERLNG(I,J,ANAME)                                         ERL00010
      DIMENSION QSET(1)                                                 ERL00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,ERL00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,    ERL00040
     2NEVNT,NHIST,NQQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCROR,NEP, ERL00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  ERL00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    ERL00070
      EQUIVALENCE (NSET(1),QSET(1))                                     ERL00080
C                                                                       ERL00090
C     TO GENERATE ERLANG VARIATES. WHEN K=1 EXPONENTIAL VARIATES ARE    ERL00100
C     GENERATED.                                                        ERL00110
C                                                                       ERL00120
      INDX=4*(J-1)+NPARM                                                ERL00130
      K=QSET(INDX+4)                                                    ERL00140
      IF(K-1)8,10,10                                                    ERL00150
    8 WRITE(3,20)J                                                      ERL00160
   20 FORMAT(' K = 0 FOR ERLANG',I7)                                    ERL00170
      STOP                                                              ERL00180
   10 R=1                                                               ERL00190
      DO 2 L=1,K                                                        ERL00200
      R=R*ANAME(I)                                                      ERL00210
    2 CONTINUE                                                          ERL00220
      ERLNG= -QSET(INDX+1)*ALOG(R)                                      ERL00230
      IF(ERLNG-QSET(INDX+2))7,5,6                                       ERL00240
    7 ERLNG=QSET(INDX+2)                                                ERL00250
    5 RETURN                                                            ERL00260
    6 IF(ERLNG-QSET(INDX+3))5,5,4                                       ERL00270
```

```
    4 ERLNG=QSET(INDX+3)                                                  ERL00280
      RETURN                                                              ERL00290
      END                                                                 ERL00300



      SUBROUTINE ERROR(NCODE)                                             ERR00010
      DIMENSION QSET(1)                                                   ERR00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MEA,MSTOP,ERR00030
     1MON,MONIT,MXX,MAXNS,MAXND,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,     ERR00040
     2NEVNT,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,  ERR00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,   ERR00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                      ERR00070
      EQUIVALENCE (NSET(1),QSET(1))                                       ERR00080
   10 FORMAT(//36X16HERROR EXIT, TYPE,I3,7H ERROR.//21H FILE STATUS AT    ERR00090
     1TIME,F10.4/)                                                        ERR00100
   11 FORMAT(1H1)                                                         ERR00110
   14 FORMAT('0',' COLCT VALUES'/)                                        ERR00120
   16 FORMAT(I10,5F10.4)                                                  ERR00130
   19 FORMAT('0',' TMST VALUES')                                          ERR00140
   23 FORMAT('0',' HISTOGRAMS')                                           ERR00150
   26 FORMAT(17X,I3,5X,23I4)                                              ERR00160
C                                                                         ERR00164
C     WRITE CURRENT TIME AND EVENT CODE. PRINT OUT ALL FILES.             ERR00165
C                                                                         ERR00166
      WRITE (NPRNT,10) NCODE,TNOW                                         ERR00170
      DO 1 I=1,NOQ                                                        ERR00180
      CALL PRNFL(I)                                                       ERR00190
    1 CONTINUE                                                            ERR00200
      WRITE (NPRNT,11)                                                    ERR00210
      IF(NCLCT)12,12,13                                                   ERR00220
C                                                                         ERR00224
C     PRINT OUT POINT ESTIMATES.                                          ERR00225
C                                                                         ERR00226
   13 WRITE(NPRNT,14)                                                     ERR00230
      INDX=NSUMA                                                          ERR00240
      DO 15 I=1,NCLCT                                                     ERR00250
      WRITE(NPRNT,16)I,(QSET(INDX+J),J=1,5)                               ERR00260
   15 INDX=INDX+5                                                         ERR00270
      WRITE (NPRNT,11)                                                    ERR00280
   12 IF(NSTAT)17,17,18                                                   ERR00290
C                                                                         ERR00294
C     PRINT OUT TIME WEIGHTED STATISTICS.                                 ERR00295
C                                                                         ERR00296
   18 WRITE (NPRNT,19)                                                    ERR00300
      INDX=5*NCLCT+NSUMA                                                  ERR00310
      DO 20 I=1,NSTAT                                                     ERR00320
      WRITE(NPRNT,16)I,(QSET(INDX+J),J=1,5)                               ERR00330
   20 INDX=INDX+5                                                         ERR00340
      WRITE (NPRNT,11)                                                    ERR00350
   17 IF(NHIST)21,21,22                                                   ERR00360
C                                                                         ERR00364
C     PRINT OUT FREQUENCY COUNTS FOR HISTOGRAMS.                          ERR00365
C                                                                         ERR00366
   22 WRITE (NPRNT,23)                                                    ERR00370
      DO 24 I=1,NHIST                                                     ERR00380
      L2=NHIST+JHIST                                                      ERR00390
      DO 25 J=1,I                                                         ERR00400
   25 L2=L2+NSET(JHIST+J)+2                                               ERR00410
```

```
      L1=L2-NSET(JHIST+I)-1                                              ERR00420
      WRITE(NPRNT,26)I,(NSET(J),J=L1,L2)                                 ERR00430
   24 CONTINUE                                                           ERR00440
   21 NFOOL=0                                                            ERR00450
      IF(NFOOL)27,28,27                                                  ERR00460
   27 RETURN                                                             ERR00470
   28 CONTINUE                                                           ERR00480
      STOP                                                               ERR00490
      END                                                                ERR00500



      SUBROUTINE FINDN(NVAL,MCODE,JQ,JATT,KCOL)                          FDN00010
      DIMENSION QSET(1)                                                  FDN00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,FDN00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,     FDN00040
     2NEVNT,NHIST,NQQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCROR,NEP,  FDN00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,   FDN00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                     FDN00070
      EQUIVALENCE (NSET(1),QSET(1))                                      FDN00080
C                                                                        FDN00090
C*****THE COLUMN WHICH IS THE BEST CANDIDATE IS KBEST                    FDN00100
C                                                                        FDN00110
      KBEST=0                                                            FDN00120
C                                                                        FDN00130
C*****THE NEXT COLUMN TO BE CONSIDERED AS A CANDIDATE IS NEXTK           FDN00140
C                                                                        FDN00150
      NEXTK=NSET(MFE+JQ)                                                 FDN00160
      IF(NEXTK) 16,1,2                                                   FDN00170
   16 CALL ERROR(50)                                                     FDN00180
    1 KCOL=KBEST                                                         FDN00190
      RETURN                                                             FDN00200
C                                                                        FDN00210
C*****MGRNV IS +1 FOR GREATER THAN SEARCH AND -1 FOR LESS THAN SEARCH    FDN00220
C*****NMAMN IS +1 FOR MAXIMUM AND -1 FOR MINIMUM                         FDN00230
C*****FOR SEARCH FOR EQUALITY THE SIGN OF MGRNV AND NMAMN ARE NOT USED   FDN00240
C                                                                        FDN00250
    2 GO TO (11,12,13,14,11),MCODE                                       FDN00260
   11 MGRNV=1                                                            FDN00270
      NMAMN=1                                                            FDN00280
      GO TO 20                                                           FDN00290
   12 MGRNV=1                                                            FDN00300
      NMAMN=-1                                                           FDN00310
      GO TO 20                                                           FDN00320
   13 MGRNV=-1                                                           FDN00330
      NMAMN=1                                                            FDN00340
      GO TO 20                                                           FDN00350
   14 MGRNV=-1                                                           FDN00360
      NMAMN=-1                                                           FDN00370
   20 INDX = (NEXTK-1)*MXX+JATT+NFIL                                     FDN00380
      IF (MGRNV*(NSET(INDX)-NVAL)) 4,21,66                               FDN00390
C                                                                        FDN00400
C*****WHEN EQUALITY IS OBTAINED TEST FOR MCODE=5, THE SEARCH FOR A       FDN00410
C*****SPECIFIED VALUE                                                    FDN00420
C                                                                        FDN00430
   21 IF(MCODE-5) 4,15,4                                                 FDN00440
   66 IF (MCODE-5) 6,4,6                                                 FDN00450
    6 IF(KBEST) 16,8,7                                                   FDN00460
    7 IF(NMAMN*(NSET(INDX)-NSET(KINDX))) 4,4,8                           FDN00470
```

```
    8 KBEST = NEXTK                                                    FDN00480
      KINDX = INDX                                                     FDN00490
    4 INDS = (NEXTK)*MXX - 1                                           FDN00500
      NEXTK = NSET(INDS+NFIL)                                          FDN00510
      IF(NEXTK-7777)20,1,1                                             FDN00520
   15 KCOL=NEXTK                                                       FDN00530
      RETURN                                                           FDN00540
      END                                                              FDN00550



      SUBROUTINE FINDQ(QVAL,MCODE,JQ,JATT,TOL,KCOL)                    FDQ00010
      DIMENSION QSET(1)                                                FDQ00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,FDQ00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,   FDQ00040
     2NEVNT,NHIST,NQQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCROR,NEP, FDQ00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  FDQ00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                   FDQ00070
      EQUIVALENCE (NSET(1),QSET(1))                                    FDQ00080
C                                                                      FDQ00090
C*****THE COLUMN WHICH IS THE BEST CANDIDATE IS KBEST                  FDQ00100
C                                                                      FDQ00110
      KBEST=0                                                          FDQ00120
C                                                                      FDQ00130
C*****THE NEXT COLUMN TO BE CONSIDERED AS A CANDIDATE IS NEXTK         FDQ00140
C                                                                      FDQ00150
      NEXTK=NSET(MFE+JQ)                                               FDQ00160
      IF(NEXTK) 16,1,2                                                 FDQ00170
   16 CALL ERROR(55)                                                   FDQ00180
    1 KCOL=KBEST                                                       FDQ00190
      RETURN                                                           FDQ00200
C                                                                      FDQ00210
C*****XGRNV IS +1 FOR GREATER THAN SEARCH AND -1 FOR LESS THAN SEARCH  FDQ00220
C*****XMAMN IS +1 FOR MAXIMUM AND -1 FOR MINIMUM                       FDQ00230
C*****FOR SEARCH FOR EQUALITY THE SIGN OF XGRNV AND XMAMN ARE NOT USED FDQ00240
C                                                                      FDQ00250
    2 GO TO (11,12,13,14,11),MCODE                                     FDQ00260
   11 XGRNV=1.                                                         FDQ00270
      XMAMN=1.                                                         FDQ00280
      GO TO 20                                                         FDQ00290
   12 XGRNV=1.                                                         FDQ00300
      XMAMN=-1.                                                        FDQ00310
      GO TO 20                                                         FDQ00320
   13 XGRNV=-1.                                                        FDQ00330
      XMAMN=1.                                                         FDQ00340
      GO TO 20                                                         FDQ00350
   14 XGRNV=-1.                                                        FDQ00360
      XMAMN=-1.                                                        FDQ00370
   20 INDX=(NEXTK-1)*MXX+NFIL                                          FDQ00380
      I1=NEVNT+JQ-1                                                    FDQ00390
      IF(JQ.EQ.1)I1=NSET(INDX+1)                                       FDQ00400
      NFIX=NSET(IM+I1)                                                 FDQ00410
      INDX=INDX+NFIX+JATT                                              FDQ00420
      TEMP = XGRNV*(QSET(INDX)-QVAL)                                   FDQ00430
      TEM  = TEMP                                                      FDQ00440
      IF (TEMP) 30,31,31                                               FDQ00450
   30 TEM = -TEMP                                                      FDQ00460
   31 IF(TEM-TOL) 21,21,33                                             FDQ00470
   33 IF(TEMP) 4,21,66                                                 FDQ00480
```

```
C                                                                       FDQ00490
C*****WHEN EQUALITY IS OBTAINED TEST FOR MCODE=5, THE SEARCH FOR A      FDQ00500
C*****SPECIFIED VALUE                                                   FDQ00510
C                                                                       FDQ00520
   21 IF(MCODE-5) 4,15,4                                                FDQ00530
   66 IF (MCODE-5) 6,4,6                                                FDQ00540
    6 IF(KBEST) 16,8,7                                                  FDQ00550
    7 IF(XMAMN*(QSET(INDX)-QSET(KINDX))) 4,4,8                          FDQ00560
    8 KBEST = NEXTK                                                     FDQ00570
      KINDX = INDX                                                      FDQ00580
    4 INDS = (NEXTK)*MXX - 1                                            FDQ00590
      NEXTK = NSET(INDS+NFIL)                                           FDQ00600
      IF(NEXTK-7777)20,1,1                                              FDQ00610
   15 KCOL=NEXTK                                                        FDQ00620
      RETURN                                                            FDQ00630
      END                                                               FDQ00640
```

```
      SUBROUTINE GASP                                                   GAS00010
      DIMENSION QSET(1)                                                 GAS00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,GAS00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLF,MCLCT,MTMST,MHIST,NCLCT,    GAS00040
     2NEVNT,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, GAS00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NOTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  GAS00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    GAS00070
      EQUIVALENCE (NSET(1),QSET(1))                                     GAS00080
      NOT=0                                                             GAS00090
    1 CALL DATAX                                                        GAS00100
C                                                                       GAS00110
C     PRINT OUT THE FILES.                                              GAS00120
C                                                                       GAS00125
      DO 2 I=1,NOQ                                                      GAS00130
      CALL PRNFL(I)                                                     GAS00140
    2 CONTINUE                                                          GAS00150
      WRITE(NPRNT,3)                                                    GAS00160
    3 FORMAT(1H1,58X,'** INTERMEDIATE RESULTS. **'//)                   GAS00170
C                                                                       GAS00180
C*****OBTAIN NEXT EVENT WHICH IS FIRST ENTRY IN FILE 1. OBTAIN EVENT    GAS00190
C*****TIME AND EVENT CODE WHICH ARE STORED IN QSET AND NSET RESPECTIVELYGAS00200
C                                                                       GAS00210
    4 MFE1=NSET(MFE+1)                                                  GAS00220
      IF(MFE1)5,5,6                                                     GAS00230
    5 CALL ERROR(80)                                                    GAS00240
    6 INDX=(MFE1-1)*MXX+NFIL+1                                          GAS00250
      JEVNT=NSET(INDX)                                                  GAS00260
      INDX=INDX+NSET(IM+JEVNT)                                          GAS00270
      TNOW=QSET(INDX)                                                   GAS00280
C                                                                       GAS00290
C      TEST TO SEE IF EVENT INFORMATION IS TO BE PRINRED.               GAS00300
C                                                                       GAS00310
      IF(JMNIT)14,27,25                                                 GAS00320
   25 IF(NSET(MONIT+JEVNT))14,27,26                                     GAS00330
   26 CALL MONTR                                                        GAS00340
C                                                                       GAS00350
C     TEST TO SEE IF THIS IS AN EVENT WITH ONE OF THE STANDARD CODES.   GAS00360
C     EVENT CODE  1  IS USED TO GET THE DUMP OUT OF FILES. EVENT CODE   GAS00370
C     2 IS USED TO TRIGGER THE MONITORING EVENT. EVENT CODE 3 TRIGGERS  GAS00380
C     THE RESETTING OF VARIABLES.                                       GAS00390
```

```
C                                                                       GAS00400
   27 IF(JEVNT-3)7,7,8                                                  GAS00410
    7 IF(JEVNT-2)9,10,11                                                GAS00420
C                                                                       GAS00430
C                                                                       GAS00430
C     PRINT OUT THE FILES.                                              GAS00440
C                                                                       GAS00450
    9 DO 12 I=1,NOQ                                                     GAS00460
      CALL PRNFL(I)                                                     GAS00470
   12 CONTINUE                                                          GAS00480
      CALL RMOVE(MFE1,1)                                                GAS00490
      GO TO 4                                                           GAS00500
C                                                                       GAS00510
C     CHANGE THE VALUE OF JMNIT (0 OR 1).                               GAS00520
C                                                                       GAS00530
   10 CALL RMOVE(MFE1,1)                                                GAS00540
      IF(JMNIT)14,15,16                                                 GAS00550
   14 CALL ERROR(85)                                                    GAS00560
   15 JMNIT=1                                                           GAS00570
      READ(NCRDR,17)(NSET(MONIT+IJ),IJ=1,NEVNT)                         GAS00580
   17 FORMAT(80I1)                                                      GAS00590
      GO TO 4                                                           GAS00600
   16 JMNIT=0                                                           GAS00610
      GO TO 4                                                           GAS00620
C                                                                       GAS00630
C     TO RESET THE VARIABLES.                                           GAS00640
C                                                                       GAS00650
   11 CALL RESET                                                        GAS00660
      CALL RMOVE(MFE1,1)                                                GAS00670
      GO TO 4                                                           GAS00680
    8 CALL EVNTS                                                        GAS00690
      CALL RMOVE(MFE1,1)                                                GAS00700
      IF(MSTOP)18,4,20                                                  GAS00710
   18 MSTOP=0                                                           GAS00720
      IF(NORPT)14,21,22                                                 GAS00730
   20 IF(TNOW-TFIN)4,21,21                                              GAS00740
   21 CALL SUMRY                                                        GAS00750
      CALL OTPUT                                                        GAS00760
C                                                                       GAS00770
C     TEST NUMBER OF RUNS REMAINING.                                    GAS00780
C                                                                       GAS00790
   22 IF(NRUNS-1)14,23,24                                               GAS00800
   24 NRUNS=NRUNS-1                                                     GAS00810
      NRUN=NRUN+1                                                       GAS00820
      GO TO 1                                                           GAS00830
   23 RETURN                                                            GAS00840
      END                                                               GAS00850
```

```
      SUBROUTINE HISTO (X1,A,W,N)                                       HIS00010
      DIMENSION QSET(1)                                                 HIS00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,HIS00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,    HIS00040
     2NEVNT,NHIST,NOQ,NORPT,NOT,NPRMS,NPUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, HIS00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  HIS00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    HIS00070
      EQUIVALENCE (NSET(1),QSET(1))                                     HIS00080
      IF (N-NHIST) 11,11,2                                              HIS00090
```

```
    2 WRITE (NPRNT,250) N                                                  HIS00100
  250 FORMAT(19H ERROR IN HISTOGRAM,I4//)                                  HIS00110
      STOP                                                                 HIS00120
   11 IF(N)2,2,3                                                           HIS00130
C                                                                          HIS00140
C*****TRANSLATE X1 BY SUBTRACTING A IF X.LE.A THEN ADD 1 TO FIRST CELL     HIS00150
C                                                                          HIS00160
    3 X = X1 - A                                                           HIS00170
      IF (X)6,7,7                                                          HIS00180
    6 IC = 1                                                               HIS00190
      GO TO 8                                                              HIS00200
C                                                                          HIS00210
C*****DETERMINE CELL NUMBER IC. ADD 1 FOR LOWER LIMIT CELL AND 1 FOR       HIS00220
C*****TRUNCATION                                                           HIS00230
C                                                                          HIS00240
    7 IC = X/W + 2.                                                        HIS00250
      IF(IC-NSET(JHIST+N)-1)8,8,9                                          HIS00260
    9 IC=NSET(JHIST+N)+2                                                   HIS00270
    8 L2=NHIST+JHIST                                                       HIS00280
      DO 12 I=1,N                                                          HIS00290
   12 L2=L2+NSET(JHIST+I)+2                                                HIS00300
      L1=L2-NSET(JHIST+N)+(IC-2)                                           HIS00310
      NSET(L1)=NSET(L1)+1                                                  HIS00320
      RETURN                                                               HIS00330
      END                                                                  HIS00340
```

```
      FUNCTION LOCAT(NCODE,K,JATT)                                         LOC00010
      DIMENSION QSET(1)                                                    LOC00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLP,JHIST,KRANK,LU,MFA,MSTOP,LOC00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,     LOC00040
     2NEVNT,NHIST,NQQ,NPRPT,NDT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,  LOC00050
     3NVNQ,NENQ,NFIL,ND,NPARM,NQTM,NSUMA,NPRUJ,NDAY,NYR,NPSET,NXX,OUT,   LOC00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                      LOC00070
      EQUIVALENCE (NSET(1),QSET(1))                                        LOC00080
      IF(MXX-JATT)1000,100,100                                             LOC00090
 1000 CALL ERROR(140)                                                      LOC00100
  100 GO TO (200,300),NCODE                                                LOC00110
  200 IF(K-ID)400,400,2000                                                 LOC00120
 2000 CALL ERROR(145)                                                      LOC00130
C                                                                          LOC00140
C     TO COMPUTE THE CELL NUMBER FOR THE GIVEN COLUMN AND ATTRIBUTE        LOC00150
C     NUMBER.                                                              LOC00160
C                                                                          LOC00170
  400 LOCAT=(K-1)*MXX+JATT+NFIL                                            LOC00180
      RETURN                                                               LOC00190
  300 IF(K-MAXNS)500,500,3000                                              LOC00200
 3000 CALL ERROR(87)                                                       LOC00210
C                                                                          LOC00220
C     TO COMPUTE COLUMN NUMBER FOR THE GIVEN CELL AND ATTRIBUTE NUMBER.    LOC00230
C                                                                          LOC00240
  500 LOCAT=1+(K-JATT)/MXX                                                 LOC00250
      RETURN                                                               LOC00260
      END                                                                  LOC00270
```

```
      SUBROUTINE MONTR                                                  MON00010
      DIMENSION QSET(1)                                                 MON00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,MON00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,    MON00040
     2NEVNT,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, MON00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NOTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  MON00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    MON00070
      EQUIVALENCE (NSET(1),QSET(1))                                     MON00080
  103 FORMAT(' ',T30,7E14.6)                                            MON00090
  105 FORMAT  (/10X23HCURRENT EVENT....TIME =,F8.2,5X7HEVENT =,I7)      MON00100
  106 FORMAT(11X,2I5,(T30,7I14))                                        MON00110
  250 FORMAT(///10X,'CURRENT EVENT(NSET).')                             MON00120
  255 FORMAT(///10X,'CURRENT EVENT (QSET).')                            MON00130
  260 FORMAT(///36X,' ERROR EXIT, TYPE 130 ERROR.')                     MON00140
C                                                                       MON00150
C     PRINT ALL THE ATTRIBUTES OF THE CURRENT EVENT.                    MON00160
C                                                                       MON00170
      L1=(NSET(MFE+1)-1)*MXX+NFIL+1                                     MON00180
      NFIX=NSET(IM+JEVNT)                                               MON00190
      NFLT=NSET(IMM+JEVNT)                                              MON00200
      NSUM=NFIX+NFLT                                                    MON00210
      IF(NXX-NSUM)2,7,7                                                 MON00220
    2 WRITE(NPRNT,260)                                                  MON00230
    7 L2=L1+NFIX-1                                                      MON00240
      L3=L2+1                                                           MON00250
      L4=L2+NFLT                                                        MON00260
      L5=L1+MXX-2                                                       MON00270
      L6=L5+1                                                           MON00280
      WRITE(NPRNT,105)TNOW,JEVNT                                        MON00290
      WRITE(NPRNT,250)                                                  MON00300
      WRITE(NPRNT,106)(NSET(I),I=L5,L6),(NSET(I),I=L1,L2)               MON00310
      WRITE(NPRNT,255)                                                  MON00320
      WRITE(NPRNT,103)(QSET(I),I=L3,L4)                                 MON00330
      RETURN                                                            MON00340
      END                                                               MON00350
```

```
      FUNCTION NPOSN(I,J,CNAME)                                         NPO00010
      EXTERNAL CNAME                                                    NPO00020
      DIMENSION QSET(1)                                                 NPO00030
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,NPO00040
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,    NPO00050
     2NEVNT,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, NPO00060
     3NVNQ,NENQ,NFIL,NQ,NPARM,NOTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  NPO00070
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    NPO00080
      EQUIVALENCE (NSET(1),QSET(1))                                     NPO00090
C                                                                       NPO00100
C     TO GENERATE POISSON VARIATES. PARAMETERS TO BE SUPPLIED ARE THE   NPO00110
C     EXPECTED VALUE,MINIMUM AND THE MAXIMUM VALUE OF THE VARIABLE.     NPO00120
C                                                                       NPO00130
      NPOSN=0                                                           NPO00140
      INDX=4*(J-1)+NPARM                                                NPO00150
      P=QSET(INDX+1)                                                    NPO00160
    1 IF(P-6.0) 2,2,4                                                   NPO00170
    2 Y=EXP(-P)                                                         NPO00180
      X=1.0                                                             NPO00190
    3 X=X*CNAME(I)                                                      NPO00200
      IF(X-Y)6,8,8                                                      NPO00210
```

```
    4 TEMP=QSET(INDX+4)                                                   NPQ00220
      QSET(INDX+4)=SQRT(QSET(INDX+1))                                     NPQ00230
      NPOSN=RNORM(I,J,CNAME)+0.5                                          NPQ00240
      QSET(INDX+4)=TEMP                                                   NPQ00250
      IF(NPOSN) 4,6,6                                                     NPQ00260
    6 KK=QSET(INDX+2)                                                     NPQ00270
      KKK=QSET(INDX+3)                                                    NPQ00280
      NPOSN=KK+NPOSN                                                      NPQ00290
      IF(NPOSN-KKK) 7,7,9                                                 NPQ00300
    7 RETURN                                                              NPQ00310
    8 NPOSN=NPOSN+1                                                       NPQ00320
      GO TO 3                                                             NPQ00330
    9 NPOSN=QSET(INDX+3)                                                  NPQ00340
      RETURN                                                              NPQ00350
      END                                                                 NPQ00360


      FUNCTION PRODQ(JT,JATT,JQ)                                          PDQ00010
      DIMENSION QSET(1)                                                   PDQ00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,PDQ00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,     PDQ00040
     2NEVNT,NHIST,NQQ,NOPPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,  PDQ00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,   PDQ00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                      PDQ00070
      EQUIVALENCE (NSET(1),QSET(1))                                       PDQ00080
      PRODQ=1.                                                            PDQ00090
   15 IF(JQ-NQQ)1,1,2                                                     PDQ00100
    2 CALL ERROR(20)                                                      PDQ00110
    1 IF(NSET(NQ+JQ))3,3,4                                                PDQ00120
    3 PRODQ=0.                                                            PDQ00130
      RETURN                                                              PDQ00140
    4 MTEM=NSET(MFE+JQ)                                                   PDQ00150
   12 INDX=(MTEM-1)*MXX+NFIL                                              PDQ00160
      I1=NEVNT+JQ-1                                                       PDQ00170
      IF(JQ.EQ.1)I1=NSET(INDX+1)                                          PDQ00180
      NFIX=NSET(IM+I1)                                                    PDQ00190
      NFLT=NSET(IMM+I1)                                                   PDQ00200
C                                                                         PDQ00204
C     JATT IS AN INTEGER IF JT=1; A REAL NUMBER IF JT=2.                  PDQ00205
C                                                                         PDQ00206
      GO TO (5,6),JT                                                      PDQ00210
    5 IF(JATT-NFIX)7,7,8                                                  PDQ00220
    8 IF(JATT-NXX)9,9,10                                                  PDQ00230
    9 CALL ERROR(25)                                                      PDQ00240
   10 IF(JATT-MXX)7,7,9                                                   PDQ00250
    7 PRODQ=PRODQ*NSET(INDX+JATT)                                         PDQ00260
      GO TO 14                                                            PDQ00270
    6 IF(JATT-NFLT)11,11,9                                                PDQ00280
   11 PRODQ=PRODQ*QSET(INDX+NFIX+JATT)                                    PDQ00290
   14 INDX=MTEM*MXX-1                                                     PDQ00300
      MTEM=NSET(INDX+NFIL)                                                PDQ00310
      IF(MTEM-7777)13,12,13                                               PDQ00320
   13 RETURN                                                              PDQ00330
      END                                                                 PDQ00340
```

```
      SUBROUTINE PRNTQ(JQ)                                                  PNQ00010
      DIMENSION QSET(1)                                                     PNQ00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MEA,MSTOP,PNQ00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,     PNQ00040
     2NEVNT,NHIST,NNQ,NORPT,NOT,NPRMS,NRUN,NSUNS,NSTAT,NPRNT,NCRDR,NEP,  PNQ00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,   PNQ00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                      PNQ00070
      EQUIVALENCE (NSET(1),QSET(1))                                         PNQ00080
      WRITE (NPRNT,100) JQ                                                  PNQ00090
      IF (TNOW - TBEG) 12,12,13                                             PNQ00100
   12 WRITE (NPRNT,105)                                                     PNQ00110
      RETURN                                                                PNQ00120
C                                                                           PNQ00130
C*****COMPUTE EXPECT NO. IN FILE JQ UP TO PRESENT THIS MAY BE USEFUL       PNQ00140
C*****IN SETTING THE VALUE OF ID                                           PNQ00150
C                                                                           PNQ00160
   13 XNQ=NSET(NQ+JQ)                                                       PNQ00170
      X=(QSET(NENQ+JQ)+XNQ*(TNOW-QSET(NQTM+JQ)))/(TNOW-TBEG)                PNQ00180
      STD=((QSET(NVNQ+JQ)+XNQ*XNQ*(TNOW-QSET(NQTM+JQ)))/(TNOW-TBEG)         PNQ00190
     1-X*X)**0.5                                                            PNQ00200
      WRITE (NPRNT,104) X,STD,NSET(MAXNQ+JQ)                                PNQ00210
C                                                                           PNQ00220
C*****PRINT FILE IN PROPER ORDER REQUIRES TRACING THROUGH THE POINTERS     PNQ00230
C*****OF THE FILE                                                          PNQ00240
C                                                                           PNQ00250
      ENTRY PRNFL(JQ)                                                       PNQ00260
      WRITE (NPRNT,101)JQ                                                   PNQ00270
      WRITE(NPRNT,200)                                                      PNQ00280
      LINE = NSET(MFE+JQ)                                                   PNQ00290
      IF(LINE-1)4,1,1                                                       PNQ00300
    4 WRITE (NPRNT,102)                                                     PNQ00310
    2 RETURN                                                                PNQ00320
    1 L1=(LINE-1)*MXX+NFIL+1                                                PNQ00330
      I1=NEVNT+JQ-1                                                         PNQ00340
      IF(JQ.EQ.1)I1=NSET(L1)                                                PNQ00350
      I2=NSET(IM+I1)                                                        PNQ00360
      I3=NSET(IMM+I1)                                                       PNQ00370
      NSUM=I2+I3                                                            PNQ00380
      IF(NXX-NSUM)3,6,6                                                     PNQ00390
    6 L2=L1+I2-1                                                            PNQ00400
      L3=L2+1                                                               PNQ00410
      L4=L2+I3                                                              PNQ00420
      L5=L1+MXX-2                                                           PNQ00430
      L6=L5+1                                                               PNQ00440
      IF(I2.GT.0)GO TO 8                                                    PNQ00450
      WRITE(NPRNT,106)LINE,(NSET(I),I=L5,L6)                                PNQ00460
      GO TO 9                                                               PNQ00470
    8 WRITE(NPRNT,106)LINE,(NSET(I),I=L5,L6),(NSET(I),I=L1,L2)              PNQ00480
      IF(I3.EQ.0)GO TO 7                                                    PNQ00490
    9 WRITE(NPRNT,103)(QSET(I),I=L3,L4)                                     PNQ00500
    7 INDX=LINE*MXX-1+NFIL                                                  PNQ00510
      LINE=NSET(INDX)                                                       PNQ00520
      IF(LINE-7777)1,2,5                                                    PNQ00530
    3 WRITE(NPRNT,201)                                                      PNQ00540
    5 WRITE (NPRNT,199)                                                     PNQ00550
  100 FORMAT(//39X25H FILE PRINTOUT, FILE  NO.,I3)                          PNQ00560
  101 FORMAT(//45X,'CONTENTS OF FILE NO.',I3//)                             PNQ00570
  102 FORMAT(/43X18HTHE FILE  IS EMPTY//)                                   PNQ00580
  103 FORMAT(' ',T30,7E14.6)                                                PNQ00590
  104 FORMAT(/35X,27HAVERAGE NUMBER IN FILE  WAS,F10.4,/35X,9HSTD. DEV.,PNQ00600
```

```
   1 18X,F10.4,/35X,7HMAXIMUM,24X,I4)                                    PNQ00610
 105 FORMAT(/35X,25H NO PRINTOUT TNOW = TBEG //)                         PNQ00620
 106 FORMAT(1X,I5,5X,2I5,(T30,7I14))                                     PNQ00630
 199 FORMAT(///36X26HERROR EXIT, TYPE 90 ERROR.)                         PNQ00640
 200 FORMAT(50X,'NSET AND QSET'//)                                       PNQ00650
 201 FORMAT(///36X,' ERROR EXIT, TYPE 95 ERROR.')                        PNQ00660
     STOP                                                                PNQ00670
     END                                                                 PNQ00680


     SUBROUTINE RESET                                                    RES00010
     DIMENSION QSET(1)                                                   RES00020
     COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,RES00030
    1MDN,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,     RES00040
    2NEVNT,NHIST,NQQ,NQRPT,NQT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,  RES00050
    3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,   RES00060
    4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                      RES00070
     EQUIVALENCE (NSET(1),QSET(1))                                       RES00080
C                                                                        RES00090
C    TO RESET A PARTICULAR SET OF VARIABLES IN 'COLCT' TO ZERO.          RES00100
C                                                                        RES00110
     IF(NCLCT)10,3,1                                                     RES00120
  10 CALL ERROR(150)                                                     RES00130
   1 INDX=NSUMA+1                                                        RES00140
     DO 2 I=1,NCLCT                                                      RES00150
     IF (QSET(MCLCT+I).NE.TNOW) GO TO 2                                  RES00160
     QSET(INDX)=0.0                                                      RES00170
     QSET(INDX+1)=0.0                                                    RES00180
     QSET(INDX+2)=0.0                                                    RES00190
     QSET(INDX+3)=1.0E20                                                 RES00200
     QSET(INDX+4)=-1.0E20                                                RES00210
     INDX=INDX+5                                                         RES00220
   2 CONTINUE                                                            RES00230
C                                                                        RES00240
C    RESET THE SET OF VARIABLES IN 'TMST' TO ZERO.                       RES00250
C                                                                        RES00260
   3 IF(NSTAT)10,4,5                                                     RES00270
   5 INDX=NCLCT*5+NSUMA+1                                                RES00280
     DO 6 I=1,NSTAT                                                      RES00290
     IF(QSET(MTMST+I).NE.TNOW) GO TO 6                                   RES00300
     QSET(NRSET+I)=TNOW                                                  RES00310
     QSET(INDX)=TNOW                                                     RES00320
     QSET(INDX+1)=0.0                                                    RES00330
     QSET(INDX+2)=0.0                                                    RES00340
     QSET(INDX+3)=1.0E20                                                 RES00350
     QSET(INDX+4)=-1.0E20                                                RES00360
     INDX=INDX+5                                                         RES00370
   6 CONTINUE                                                            RES00380
C                                                                        RES00390
C    RESET THE DESIRED SET OF VARIABLES IN 'HISTO' TO ZERO.              RES00400
C                                                                        RES00410
   4 IF(NHIST)10,7,8                                                     RES00420
   8 DO 9 I=1,NHIST                                                      RES00430
     IF(QSET(MHIST+I).NE.TNOW) GO TO 9                                   RES00440
     L2=JHIST+NHIST                                                      RES00450
     DO 11 J=1,I                                                         RES00460
  11 L2=L2+NSET(JHIST+J)+2                                               RES00470
     L1=L2-NSET(JHIST+I)-1                                               RES00480
```

```
      DO 12 J=L1,L2                                                    RES00490
   12 NSET(J)=0                                                        RES00500
    9 CONTINUE                                                         RES00510
C                                                                      RES00520
C     TO CALL SUBROUTINE DATAN IF ANY ADDITIONAL DATA CARDS ARE TO BE  RES00530
C     READ IN AT THIS TIME. 'JCLR' SHOULD ALWAYS BE LESS THAN OR EQUAL RES00540
C     TO ZERO WHEN DATAN SUBROUTINE IS USED AT THE TIME OF RESETTING.  RES00550
C                                                                      RES00560
    7 IN=(NSET(MFE+1)-1)*MXX+2+NFIL                                    RES00570
      IF(NSET(IN))13,13,14                                             RES00580
   14 NSTORE=NEP                                                       RES00590
      JOLD=JCLR                                                        RES00600
      NEP=NSET(IN+1)                                                   RES00610
      JCLR=0                                                           RES00620
      CALL DATAX                                                       RES00630
      NEP=NSTORE                                                       RES00640
      JCLR=JOLD                                                        RES00650
   13 RETURN                                                           RES00660
      END                                                              RES00670
```

```
      FUNCTION RLOGN(I,J,BNAME)                                        RLN00010
      EXTERNAL BNAME                                                   RLN00020
      DIMENSION QSET(1)                                                RLN00030
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,RLN00040
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,   RLN00050
     2NEVNT,NHIST,NQQ,NQRPT,NQT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, RLN00060
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  RLN00070
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                   RLN00080
      EQUIVALENCE (NSET(1),QSET(1))                                    RLN00090
C                                                                      RLN00095
C*****THE PARAMETERS USED WITH RLOGN ARE THE MEAN AND THE STANDARD     RLN00100
C*****DEVIATION OF A NORMAL DISTRIBUTION.                              RLN00110
C                                                                      RLN00120
      VA= RNORM(I,J,BNAME)                                             RLN00130
      RLOGN=EXP(VA)                                                    RLN00140
      RETURN                                                           RLN00150
      END                                                              RLN00160
```

```
      FUNCTION RNORM(I,J,ANAME)                                        RNM00010
      DIMENSION QSET(1)                                                RNM00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,RNM00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,   RNM00040
     2NEVNT,NHIST,NQQ,NQRPT,NQT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, RNM00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  RNM00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                   RNM00070
      EQUIVALENCE (NSET(1),QSET(1))                                    RNM00080
C                                                                      RNM00090
C     TO GENERATE NORMAL DEVIATES. PARAMETERS TO BE SPECIFIED ARE THE  RNM00100
C     MEAN,MINIMUM,MAXIMUM AND THE STANDARD DEVIATION OF THE           RNM00110
C     DISTRIBUTION.                                                    RNM00120
C                                                                      RNM00130
      RA=ANAME(I)                                                      RNM00140
      RB=ANAME(I)                                                      RNM00150
      V=(-2.0*ALOG(RA))**0.5*COS(6.283*RB)                             RNM00160
```

```
      INDX=4*(J-1)+NPARM                                                RNM00170
      RNORM=V*QSET(INDX+4)+QSET(INDX+1)                                 RNM00180
      IF(RNORM-QSET(INDX+2))6,7,8                                       RNM00190
    6 RNORM=QSET(INDX+2)                                                RNM00200
    7 RETURN                                                            RNM00210
    8 IF(RNORM-QSET(INDX+3))7,7,9                                       RNM00220
    9 RNORM=QSET(INDX+3)                                                RNM00230
      RETURN                                                            RNM00240
      END                                                               RNM00250



      SUBROUTINE SET                                                    SET00010
      DIMENSION QSET(1)                                                 SET00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,SET00030
     1MON,MONTT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,    SET00040
     2NEVNT,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, SET00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  SET00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    SET00070
      EQUIVALENCE (NSET(1),QSET(1))                                     SET00080
C                                                                       SET00090
C     THE TWO SEPARATE ARRAYS NSET AND QSET HAVE BEEN MADE EQUIVALENT   SET00100
C     AND   A  PARTICULAR LOCATION OF THE ARRAY CAN BE USED EITHER BY   SET00110
C     NSET OR BY QSET. POINTERS FOR THE ENTRIES ARE STORED AS BEFORE IN SET00120
C     NSET. QSET AND NSET ARE ONE DIMENSIONAL ARRAYS BUT MAY BE THOUGHT SET00130
C     OF AS TWO DIMENSIONAL ARRAYS WITH BOTH NUMBER OF ROWS AND COLUMNS SET00140
C     VARIABLE AT EXECUTION TIME. FUNCTION LOCAT WILL LINK              SET00150
C     (ROW,COL) TO INDX. FOR FILE 1,FIRST CELL IN QSET CONTAINS EVENT   SET00160
C     TIME AND FIRST CELL IN NSET CONTAINS EVENT CODE.                  SET00170
C                                                                       SET00180
C*****SET UP POINTERS FOR A FILE JQ, NSET(INN+JQ)=1 IS FIFO AND         SET00190
C*****AND NSET(INN+JQ)=2 IS LIFO.                                       SET00200
C                                                                       SET00210
      KOL=7777                                                          SET00220
      KOF = 8888                                                        SET00230
      KLE = 9999                                                        SET00240
      MAXNS = ID * MXX                                                  SET00250
C                                                                       SET00260
C*****INITIALIZE POINTERS IN NSET AND QSET.                             SET00270
C                                                                       SET00280
      DO 1 I = 1,ID                                                     SET00290
      INDX = I * MXX+NFIL                                               SET00300
      NSET(INDX - 1) = I + 1                                            SET00310
    1 NSET(INDX) = I - 1                                                SET00320
      NSET(MAXNS - 1+NFIL)=KOF                                          SET00330
      DO 3 K = 1,NOQ                                                    SET00340
      NSET(NQ+K)=0                                                      SET00350
      NSET(MLC+K)=0                                                     SET00360
      NSET(MFE+K)=0                                                     SET00370
      NSET(MAXNQ+K)=0                                                   SET00380
      NSET(MLE+K)=0                                                     SET00390
      QSET(NENQ+K)=0.0                                                  SET00400
      QSET(NVNQ+K)=0.0                                                  SET00410
    3 QSET(NQTM+K)=TNOW                                                 SET00420
C                                                                       SET00430
C*****FIRST AVAILABLE COLUMN = 1                                        SET00440
C                                                                       SET00450
      MFA = 1                                                           SET00460
      OUT = 0.0                                                         SET00470
```

```
      DO 9 I=1,NSTAT                                                    SET00480
    9 QSET(NRSET+I)=TBEG                                                SET00490
      RETURN                                                            SET00500
C                                                                       SET00510
C*****MFEX IS FIRST ENTRY IN FILE WHICH HAS NOT BEEN COMPARED WITH ITEM SET00520
C*****TO BE INSERTED                                                    SET00530
C                                                                       SET00540
      ENTRY RMOVE(KCOLL,JQ)                                             SET00550
      IF(KCOLL)27,27,28                                                 SET00560
   27 CALL ERROR (110)                                                  SET00570
   28 NSET(MLC+JQ)=KCOLL                                                SET00580
      OUT=1.0                                                           SET00590
      ENTRY FILEM(JQ)                                                   SET00600
      MFEX=NSET(MFE+JQ)                                                 SET00610
C                                                                       SET00620
C*****KNT IS A CHECK CODE TO INDICATE THAT NO COMPARISONS HAVE BEEN MADESET00630
C                                                                       SET00640
      KNT = 2                                                           SET00650
C                                                                       SET00660
C*****KS IS THE ROW ON WHICH ITEMS OF FILE JQ ARE RANKED                SET00670
C                                                                       SET00680
      KS=NSET(KRANK+JQ)                                                 SET00690
      KSJ = 1                                                           SET00700
      IF (KS - 100) 1020,100,1000                                       SET00710
 1000 KSJ=2                                                             SET00720
      KS = KS - 100                                                     SET00730
C                                                                       SET00740
C*****TEST FOR PUTTING VALUE IN OR OUT                                  SET00750
C*****IF OUT EQUALS ONE AN ITEM IS TO BE REMOVED FROM FILE JQ.  IF OUT  SET00760
C*****IS LESS THAN ONE AN ITEM IS TO BE INSERTED IN FILE JQ             SET00770
C                                                                       SET00780
 1020 IF(OUT)100,8,5                                                    SET00790
C                                                                       SET00800
C*****PUTTING AN ENTRY IN FILE JQ                                       SET00810
C*****THE ITEM TO BE INSERTED WILL BE PUT IN COLUMN MFA                 SET00820
C                                                                       SET00830
    8 INDX = MFA * MXX - 1+NFIL                                         SET00840
      NXFA = NSET(INDX)                                                 SET00850
C                                                                       SET00860
C*****IF NSET(INN+JQ) EQUALS TO TWO THE FILE IS A LIFO FILE. IF         SET00870
C*****NSET(INN+JQ) IS ONE THE FILE IS A FIFO FILE FOR FIFO FILES TRY TO SET00880
C*****INSERT STARTING AT THE END OF FILE. MLEX IS LAST ENTRY IN FILE    SET00890
C*****WHICH HAS NOT BEEN COMPARED WITH ITEMS TO BE INSERTED.            SET00900
C                                                                       SET00910
      IF (NSET(INN+JQ)-1)100,7,6                                        SET00920
    7 MLEX=NSET(MLE+JQ)                                                 SET00930
C                                                                       SET00940
C*****IF MLEX IS ZERO FILE IS EMPTY.  ITEM TO BE INSERTED WILL BE ONLY  SET00950
C*****ITEM IN FILE.                                                     SET00960
C                                                                       SET00970
      IF (MLEX) 100,10,11                                               SET00980
   10 INDX = MFA * MXX+NFIL                                             SET00990
      NSET(INDX) = KLE                                                  SET01000
      NSET(MFE+JQ)=MFA                                                  SET01010
C                                                                       SET01020
C*****THERE IS NO SUCCESSOR OF ITEM INSERTED.  SINCE ITEM WAS INSERTED  SET01030
C*****IN COLUMN MFA THE LAST ENTRY OF FILE JQ IS IN COLUMN MFA.         SET01040
C                                                                       SET01050
   17 INDX = MFA * MXX - 1+NFIL                                         SET01060
      NSET(INDX) = KOL                                                  SET01070
```

```
      NSET(MLE+JQ)=MFA                                                  SET01080
C                                                                       SET01090
C*****SET NEW MFA EQUAL TO SUCCESSOR OF OLD MFA.  THAT IS NXFA.  THE    SET01100
C*****NEW MFA HAS NO PREDECESSOR SINCE IT IS THE FIRST AVAILABLE COLUMN SET01110
C*****FOR STORAGE.                                                      SET01120
C                                                                       SET01130
   14 MFA =NXFA                                                         SET01140
      IF(MFA.GE.KOF)GO TO 238                                           SET01150
      INDX = NXFA*MXX+NFIL                                              SET01160
      NSET(INDX) = KLE                                                  SET01170
C                                                                       SET01180
C*****UPDATE STATISTICS OF FILE JQ                                      SET01190
C                                                                       SET01200
  239 XNQ = NSET(NQ+JQ)                                                 SET01210
      QSET(NENQ+JQ)=QSET(NENQ+JQ)+XNQ*(TNOW-QSET(NQTM+JQ))              SET01220
      QSET(NVNQ+JQ)=QSET(NVNQ+JQ)+XNQ*XNQ*(TNOW-QSET(NQTM+JQ))          SET01230
      QSET(NQTM+JQ)=TNOW                                                SET01240
      NSET(NQ+JQ)=NSET(NQ+JQ)+1                                         SET01250
      NSET(MAXNQ+JQ)=MAXO(NSET(MAXNQ+JQ),NSET(NQ+JQ))                   SET01260
      NSET(MLC+JQ)=NSET(MFE+JQ)                                         SET01270
      RETURN                                                            SET01280
C                                                                       SET01290
C*****TEST TO DETERMINE IF RANKING ATTRIBUTE IS IN QSET (KRANK.LT.100)  SET01300
C*****OR NSET (KRANK .GT.100).                                          SET01310
C                                                                       SET01320
   11 GO TO (1100,1120),KSJ                                             SET01330
 1100 INDX1=(MFA-1)*MXX+NFIL                                            SET01340
      I1=NEVNT+JQ-1                                                     SET01350
      IF(JQ.EQ.1)I1=NSET(INDX1+1)                                       SET01360
      INDX1=INDX1+NSET(IM+I1)+KS                                        SET01370
      INDX2=(MLEX-1)*MXX+NFIL                                           SET01380
      I2=I1                                                             SET01390
      IF(JQ.EQ.1)I2=NSET(INDX2+1)                                       SET01400
      INDX2=INDX2+NSET(IM+I2)+KS                                        SET01410
      IF (QSET(INDX1).LT.QSET(INDX2))GO TO 12                           SET01420
      GO TO 13                                                          SET01430
 1120 INDX1 = (MFA - 1) * MXX + KS+NFIL                                 SET01440
      INDX2 = (MLEX - 1) * MXX + KS+NFIL                                SET01450
C                                                                       SET01460
C*****TEST RANKING VALUE OF NEW ITEM AGAINST VALUE OF ITEM IN COLUMN     SET01470
C*****MLEX                                                              SET01480
C                                                                       SET01490
      IF (NSET(INDX1).LT.NSET(INDX2))GO TO 12                           SET01500
C                                                                       SET01510
C*****INSERT ITEM AFTER COLUMN MLEX.  LET SUCCESSOR OF MLEX BE MSU.     SET01520
C                                                                       SET01530
   13 INDX = MLEX * MXX - 1+NFIL                                        SET01540
      MSU = NSET(INDX)                                                  SET01550
      NSET(INDX) = MFA                                                  SET01560
      INDX = MFA * MXX+NFIL                                             SET01570
      NSET(INDX) = MLEX                                                 SET01580
      GO TO (18,17),KNT                                                 SET01590
C                                                                       SET01600
C*****SINCE KNT EQUALS ONE A COMPARISON WAS MADE AND THERE IS A          SET01610
C*****SUCCESSOR TO MLEX, I.E., MSU IS NOT EQUAL TO KOL.  POINT COLUMN   SET01620
C*****MFA TO MSU AND VICE VERSA.                                        SET01630
C                                                                       SET01640
   18 INDX = MFA * MXX - 1+NFIL                                         SET01650
      NSET(INDX) = MSU                                                  SET01660
      INDX = MSU * MXX+NFIL                                             SET01670
```

```
      NSET(INDX) = MFA                                                  SET01680
      GO TO 14                                                          SET01690
C                                                                       SET01700
C*****SET KNT TO ONE SINCE A COMPARISON WAS MADE.                       SET01710
C                                                                       SET01720
   12 KNT = 1                                                           SET01730
C                                                                       SET01740
C*****TEST MFA AGAINST PREDECESSOR OF MLEX BY LETTING MLEX EQUAL        SET01750
C*****PREDECESSOR OF MLEX.                                             SET01760
C                                                                       SET01770
      INDX = MLEX * MXX+NFIL                                            SET01780
      MLEX = NSET(INDX)                                                 SET01790
      IF(MLEX-KLE)  11,16,11                                            SET01800
C                                                                       SET01810
C*****IF MLEX HAD NO PREDECESSOR MFA IS FIRST IN FILE.                  SET01820
C                                                                       SET01830
   16 INDX = MFA * MXX+NFIL                                             SET01840
      NSET(INDX) = KLE                                                  SET01850
      NSET(MFE+JQ)=MFA                                                  SET01860
C                                                                       SET01870
C*****SUCCESSOR OF MFA IS MFEX AND PREDECESSOR OF MFEX IS MFA. (NOTE AT SET01880
C*****THIS POINT MLEX = MFEX IF FIFO WAS USED).                         SET01890
C                                                                       SET01900
   26 INDX = MFA * MXX - 1+NFIL                                         SET01910
      NSET(INDX) = MFEX                                                 SET01920
      INDX = MFEX * MXX+NFIL                                            SET01930
      NSET(INDX) = MFA                                                  SET01940
      GO TO 14                                                          SET01950
C                                                                       SET01960
C*****FOR LIFO OPERATION TRY TO INSERT ITEM STARTING AT BEGINNING OF    SET01970
C*****FILE JQ.                                                         SET01980
C*****IF MFEX IS 0, NO ENTRIES ARE IN FILE JQ.  THIS CASE WAS CONSIDEREDSET01990
C*****PREVIOUSLY AT STATEMENT 10.                                      SET02000
C                                                                       SET02010
    6 IF (MFEX) 100,10,19                                               SET02020
C                                                                       SET02030
C*****TEST RANKING VALUE OF NEW ITEM AGAINST VALUE OF ITEM IN COLUMN    SET02040
C*****MFEX.                                                            SET02050
C                                                                       SET02060
   19 GO TO (1200,1220),KSJ                                             SET02070
 1200 INDX1=(MFA-1)*MXX+NFIL                                            SET02080
      I1=NEVNT+JQ-1                                                     SET02090
      IF(JQ.EQ.1)I1=NSET(INDX1+1)                                       SET02100
      INDX1=INDX1+NSET(IM+I1)+KS                                        SET02110
      INDX2=(MLEX-1)*MXX+NFIL                                           SET02120
      I2=I1                                                             SET02130
      IF(JQ.EQ.1)I2=NSET(INDX2+1)                                       SET02140
      INDX2=INDX2+NSET(IM+I2)+KS                                        SET02150
      IF (QSET(INDX1).LT.QSET(INDX2))GO TO 20                           SET02160
      GO TO 21                                                          SET02170
 1220 INDX1 = (MFA - 1) * MXX + KS+NFIL                                 SET02180
      INDX2 = (MFEX - 1) * MXX + KS+NFIL                                SET02190
      IF (NSET(INDX1).GE.NSET(INDX2))GO TO 21                           SET02200
C                                                                       SET02210
C*****IF NEW VALUE IF LOWER, MFA MUST BE COMPARED AGAINST SUCCESSOR OF  SET02220
C*****MFEX.                                                            SET02230
C                                                                       SET02240
   20 KNT = 1                                                           SET02250
C                                                                       SET02260
C*****LET MPRE = MFEX AND LET MFEX BE THE SUCCESSOR OF MFEX.            SET02270
```

```
C                                                                       SET02280
      MPRE = MFEX                                                       SET02290
      INDX = MFEX * MXX - 1+NFIL                                        SET02300
      MFEX = NSET(INDX)                                                 SET02310
      IF (MFEX-KOL)  19,24,19                                           SET02320
C                                                                       SET02330
C*****IF NEW VALUE IS HIGHER, IT SHOULD BE INSERTED BETWEEN MFEX AND ITSSET02340
C*****PREDECESSOR.                                                     SET02350
C*****IF KNT = 2, MFEX HAS NO PREDECESSOR,  GO TO STATEMENT 16.  IF KNT SET02360
C*****= 1, A COMPARISON WAS MADE AND A VALUE OF MPRE HAS ALREADY BEEN   SET02370
C*****OBTAINED ON THE PREVIOUS ITERATION.  SET KNT = 2 TO INDICATE THIS.SET02380
C                                                                       SET02390
   21 GO TO (22,16),KNT                                                 SET02400
   22 KNT = 2                                                           SET02410
C                                                                       SET02420
C*****MFA IS TO BE INSERTED AFTER MPRE.  MAKE MPRE THE PREDECESSOR OF   SET02430
C*****MFA AND MFA THE SUCCESSOR OF MPRE.                                SET02440
C                                                                       SET02450
   24 INDX = MFA * MXX+NFIL                                             SET02460
      NSET(INDX) = MPRE                                                 SET02470
      INDX = MPRE * MXX - 1+NFIL                                        SET02480
      NSET(INDX) = MFA                                                  SET02490
C                                                                       SET02500
C*****IF KNT WAS NOT RESET TO 2, THERE IS NO SUCCESSOR OF MFA.  POINTERSSET02510
C*****ARE UPDATED AT STATEMENT 17.  IF KNT = 2, IT WAS RESET AND THE    SET02520
C*****SUCCESSOR OF MFA IS MFEX.                                         SET02530
C                                                                       SET02540
      GO TO (17,26), KNT                                                SET02550
C                                                                       SET02560
C*****REMOVAL OF AN ITEM FROM FILE JQ.                                  SET02570
C*****RESET OUT TO 0 AND CLEAR COLUMN REMOVED.  LET JL EQUAL SUCCESSOR  SET02580
C*****OF COLUMN REMOVED AND JK EQUAL PREDECESSOR OF COLUMN REMOVED.     SET02590
C*****IF JL = KOL, MLC WAS LAST ENTRY.  IF JK = KLE, MLC WAS FIRST ENTRYSET02600
C*****MLC WAS NOT FIRST OR LAST ENTRY.  UPDATE POINTERS SO THAT JL IS   SET02610
C*****SUCCESSOR OF JK AND JK IS PREDECESSOR OF JL.                      SET02620
C                                                                       SET02630
    5 OUT = 0.0                                                         SET02640
C                                                                       SET02650
C*****UPDATE POINTING SYSTEM TO ACCOUNT FOR REMOVAL OF MLC (JQ).  COLUMNSET02660
C*****REMOVED IS ALWAYS SET TO MLC(JQ) BY SUBROUTINE RMOVE.             SET02670
C                                                                       SET02680
      INDX=NSET(MLC+JQ)*MXX+NFIL                                        SET02690
      JL = NSET(INDX - 1)                                               SET02700
      JK = NSET(INDX)                                                   SET02710
      IF(JL.EQ.KOL)GO TO 34                                             SET02720
      IF(JK.EQ.KLE)GO TO 36                                             SET02730
      INDX = JK * MXX - 1+NFIL                                          SET02740
      NSET(INDX) = JL                                                   SET02750
      INDX = JL * MXX+NFIL                                              SET02760
      NSET(INDX) = JK                                                   SET02770
C                                                                       SET02780
C*****UPDATE POINTERS.                                                  SET02790
C                                                                       SET02800
   37 INDX = NSET(MLC+JQ)*MXX-1+NFIL                                    SET02810
      NSET(INDX) = MFA                                                  SET02820
      NSET(INDX+1) = KLE                                                SET02830
      IF(MFA.GE.KOF)GO TO 235                                           SET02840
      INDX=MFA*MXX+NFIL                                                 SET02850
      NSET(INDX)=NSET(MLC+JQ)                                           SET02860
  235 MFA = NSET(MLC+JQ)                                                SET02870
```

```
      NSET(MLC+JQ)=NSET(MFE+JQ)                                          SET02880
C                                                                        SET02890
C*****UPDATING FILE STATISTICS                                           SET02900
C                                                                        SET02910
      XNQ = NSET(NQ+JQ)                                                  SET02920
      QSET(NENQ+JQ)=QSET(NENQ+JQ)+XNQ*(TNOW-QSET(NQTM+JQ))               SET02930
      QSET(NVNQ+JQ)=QSET(NVNQ+JQ)+XNQ*XNQ*(TNOW-QSET(NQTM+JQ))           SET02940
      QSET(NQTM+JQ)=TNOW                                                 SET02950
      NSET(NQ+JQ)=NSET(NQ+JQ)-1                                          SET02960
      RETURN                                                             SET02970
C                                                                        SET02980
C*****MLC WAS FIRST ENTRY BUT NOT LAST ENTRY.  UPDATE POINTERS.          SET02990
C                                                                        SET03000
   36 INDX = JL * MXX+NFIL                                               SET03010
      NSET(INDX) = KLE                                                   SET03020
      NSET(MFE+JQ)=JL                                                    SET03030
      GO TO 37                                                           SET03040
   34 IF(JK.EQ.KLE)GO TO 39                                              SET03050
C                                                                        SET03060
C*****MLC WAS LAST ENTRY BUT NOT FIRST ENTRY.  UPDATE POINTERS.          SET03070
C                                                                        SET03080
      INDX = JK *MXX - 1+NFIL                                            SET03090
      NSET(INDX) = KOL                                                   SET03100
      NSET(MLE+JQ)=JK                                                    SET03110
      GO TO 37                                                           SET03120
C                                                                        SET03130
C*****MLC WAS BOTH THE LAST AND FIRST ENTRY, THEREFORE, IT IS THE ONLY    SET03140
C*****ENTRY.                                                             SET03150
C                                                                        SET03160
   39 NSET(MFE+JQ)=0                                                     SET03170
      NSET(MLE+JQ)=0                                                     SET03180
      GO TO 37                                                           SET03190
  100 CALL ERROR (40)                                                    SET03200
      RETURN                                                             SET03210
      END                                                                SET03220
```

```
      FUNCTION SUMQ(JT,JATT,JQ)                                          SMQ00010
      DIMENSION QSET(1)                                                  SMQ00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLP,JHIST,KRANK,LU,MFA,MSTOP,SMQ00030
     1MON,MONIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,     SMQ00040
     2NEVNT,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP,  SMQ00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,   SMQ00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                     SMQ00070
      EQUIVALENCE (NSET(1),QSET(1))                                      SMQ00080
      SUMQ=0.                                                            SMQ00090
   15 IF(JQ-NOQ)1,1,2                                                    SMQ00100
    2 CALL ERROR(20)                                                     SMQ00110
    1 IF(NSET(NQ+JQ))3,3,4                                               SMQ00120
    3 RETURN                                                             SMQ00130
    4 MTEM=NSET(MFE+JQ)                                                  SMQ00140
   12 INDX=(MTEM-1)*MXX+NFIL                                             SMQ00150
      I1=NEVNT+JQ-1                                                      SMQ00160
      IF(JQ.EQ.1)I1=NSET(INDX+1)                                         SMQ00170
      NFIX=NSET(IM+I1)                                                   SMQ00180
      NFLT=NSET(IMM+I1)                                                  SMQ00190
C                                                                        SMQ00194
C     JATT IS AN INTEGER IF JT=1; A REAL NUMBER IF JT=2.                 SMQ00195
```

```
C                                                                       SMQ00196
      GO TO (5,6),JT                                                    SMQ00200
    5 IF(JATT-NFIX)7,7,8                                                SMQ00210
    8 IF(JATT-NXX)9,9,10                                                SMQ00220
    9 CALL ERROR(25)                                                    SMQ00230
   10 IF(JATT-MXX)7,7,9                                                 SMQ00240
    7 SUMQ=SUMQ+NSET(INDX+JATT)                                         SMQ00250
      GO TO 14                                                          SMQ00260
    6 IF(JATT-NFLT)11,11,9                                              SMQ00270
   11 SUMQ=SUMQ+QSET(INDX+NFIX+JATT)                                    SMQ00280
   14 INDX=MTEM*MXX-1                                                   SMQ00290
      MTEM=NSET(INDX+NFIL)                                              SMQ00300
      IF(MTEM-7777)13,12,13                                             SMQ00310
   13 RETURN                                                            SMQ00320
      END                                                               SMQ00330



      SUBROUTINE SUMRY                                                  SMY00010
      DIMENSION QSET(1)                                                 SMY00020
      COMMON IX,IN,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,SMY00030
     1MON,MUNIT,MXX,MAXNS,MAXNQ,MFE,MLC,MLE,MCLCT,MTMST,MHIST,NCLCT,    SMY00040
     2NEVNT,NHIST,NQQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,NPRNT,NCRDR,NEP, SMY00050
     3NVNQ,NENQ,NFIL,NQ,NPARM,NQTM,NSUMA,NPROJ,NDAY,NYR,NRSET,NXX,OUT,  SMY00060
     4TNOW,TBEG,TFIN,NAME(6),NSET(1)                                    SMY00070
      EQUIVALENCE (NSET(1),QSET(1))                                     SMY00080
   21 FORMAT (1H1,39X,23H**GASP SUMMARY REPORT**/)                      SMY00090
   23 FORMAT (//44X,18H**GENERATED DATA**/ 27X,4HCODE,4X,4HMEAN,6X,8HSTDSMY00100
     1,DEV.,5X,4HMIN.,7X,4HMAX.,5X,4HOBS./)                            SMY00110
   24 FORMAT (27X,I3,4F11.4,I7)                                         SMY00120
   25 FORMAT    (/37X,37H**GENERATED FREQUENCY DISTRIBUTIONS**/ 27X,4HCODSMY00130
     1E,20X,10HHISTOGRAMS/)                                             SMY00140
   26 FORMAT(27X,I3,5X,15I4/(35X,15I4))                                 SMY00150
   29 FORMAT  (/44X,23H**TIME GENERATED DATA**/ 27X,4HCODE,4X,4HMEAN,6X,SMY00160
     18HSTD.DEV.,5X,4HMIN.,7X,4HMAX.,3X,10HTOTAL TIME/)                 SMY00170
   30 FORMAT (27X,I3,5F11.4)                                            SMY00180
   63 FORMAT(27X,I3,10X18HNO VALUES RECORDED)                           SMY00190
  102 FORMAT      (30X,22HSIMULATION PROJECT NO.,I4,2X,2HBY,2X,         SMY00200
     1 6A2//,30X,4HDATE,I3,1H/,I3,1H/,I5,12X,10HRUN NUMBER,I5//)       SMY00210
  107 FORMAT(20X,14H PARAMETER NO.,I5,4F12.4)                           SMY00220
  199 FORMAT(///36X,' ERROR EXIT, TYPE 120 ERROR.')                     SMY00230
      WRITE (NPRNT,21)                                                  SMY00240
      WRITE (NPRNT,102) NPROJ,NAME,MON,NDAY,NYR,NRUN                    SMY00250
      IF (NPRMS) 147,147,146                                            SMY00260
  146 INDX=NPARM                                                        SMY00270
      DO 64 I=1,NPRMS                                                   SMY00280
      WRITE (NPRNT,107) I,(QSET(INDX+J),J=1,4)                          SMY00290
      INDX=INDX+4                                                       SMY00300
   64 CONTINUE                                                          SMY00310
  147 IF(NCLCT)5,60,66                                                  SMY00320
    5 WRITE (NPRNT,199)                                                 SMY00330
      STOP                                                              SMY00340
   66 WRITE (NPRNT,23)                                                  SMY00350
C                                                                       SMY00360
C*****COMPUTE AND PRINT STATISTICS GATHERED BY CLCT                     SMY00370
C                                                                       SMY00380
      INDX=(NSUMA+1)                                                    SMY00390
      DO 2 I=1,NCLCT                                                    SMY00400
      IF(QSET(INDX+2)) 5,62,61                                          SMY00410
```

```
   62 WRITE (NPRNT,63) I                                                SMY00420
      GO TO 3                                                           SMY00430
   61 XS=QSET(INDX)                                                     SMY00440
      XSS=QSET(INDX+1)                                                  SMY00450
      XN=QSET(INDX+2)                                                   SMY00460
      IF(XN-1) 201,201,200                                              SMY00470
  201 AVG=XS/XN                                                         SMY00480
      STD=0.0                                                           SMY00490
      GO TO 202                                                         SMY00500
  200 AVG=XS/XN                                                         SMY00510
      STD=(((XN*XSS)-(XS*XS))/(XN*(XN-1.0)))**.5                        SMY00520
  202 N=XN                                                              SMY00530
      WRITE(NPRNT,24)I,AVG,STD,QSET(INDX+3),QSET(INDX+4),N              SMY00540
    3 INDX=INDX+5                                                       SMY00550
    2 CONTINUE                                                          SMY00560
   60 IF(NSTAT)5,67,4                                                   SMY00570
    4 WRITE (NPRNT,29)                                                  SMY00580
C                                                                       SMY00590
C*****COMPUTE AND PRINT STATISTICS GATHERED BY TMST                     SMY00600
C                                                                       SMY00610
      INDX=5*NCLCT+(NSUMA+1)                                            SMY00620
      DO 6 I = 1,NSTAT                                                  SMY00630
      IF(QSET(INDX))5,71,72                                             SMY00640
   71 WRITE (NPRNT,63) I                                                SMY00650
      GO TO 10                                                          SMY00660
   72 XT=QSET(INDX)-QSET(NRSET+I)                                       SMY00670
      XS=QSET(INDX+1)                                                   SMY00680
      XSS=QSET(INDX+2)                                                  SMY00690
      AVG = XS/XT                                                       SMY00700
      STD = (XSS/XT-AVG*AVG)**.5                                        SMY00710
      WRITE(NPRNT,30)I,AVG,STD,QSET(INDX+3),QSET(INDX+4),XT             SMY00720
   10 INDX=INDX+5                                                       SMY00730
    6 CONTINUE                                                          SMY00740
   67 IF(NHIST)5,75,9                                                   SMY00750
    9 WRITE (NPRNT,25)                                                  SMY00760
C                                                                       SMY00770
C*****PRINT HISTOGRAMS                                                  SMY00780
C                                                                       SMY00790
      DO 12 I=1,NHIST                                                   SMY00800
      L2=NHIST+JHIST                                                    SMY00810
      DO 13 J=1,I                                                       SMY00820
   13 L2=L2+NSET(JHIST+J)+2                                             SMY00830
      L1=L2-NSET(JHIST+I)-1                                             SMY00840
      WRITE(NPRNT,26)I,(NSET(J),J=L1,L2)                                SMY00850
   12 CONTINUE                                                          SMY00860
C                                                                       SMY00870
C*****PRINT FILES AND FILE STATISTICS                                   SMY00880
C                                                                       SMY00890
   75 DO 15 I=1,NDQ                                                     SMY00900
      CALL PRNTQ(I)                                                     SMY00910
   15 CONTINUE                                                          SMY00920
      RETURN                                                            SMY00930
      END                                                               SMY00940



      SUBROUTINE TMST  (X,N)                                            TMS00010
      DIMENSION QSET(1)                                                 TMS00020
      COMMON IX,IM,INN,ID,IMM,JEVNT,JMNIT,JCLR,JHIST,KRANK,LU,MFA,MSTOP,TMS00030
```

GASP IIP

A MODIFIED VERSION OF GASP IIA

by

RAJAGOPALAN KRISHNASWAMY

B.E. (Mechanical Engineering)

University of Madras, India, 1970

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1973

ABSTRACT

In this work GASP IIA, a FORTRAN based simulation language has been modified and extended so as to improve its potentials and performance characteristics. GASP IIP is the result of this effort. Many of the constraints present in GASP IIA have been eliminated; new facilities have been added and existing facilities have been made more effective. GASP IIP utilizes the allotted core space better than GASP IIA. Even with all the added facilities, the total core space requirements of GASP IIP is comparable to that of GASP IIA. Trial simulation runs have shown that GASP IIP is also faster in execution than GASP IIA.