INTERFACING GPSS/H WITH GKS
by
Raphael S. Parambi
Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Master of Science in Industrial Engineering & Operations Research

APPROVED:

Aseem Chandawarkar, Chairman

Marilyn S. Jones Arvid Myklebust

May 28, 1987
Blacksburg, Virginia
INTERFACING GPSS/H WITH GKS

by

Raphael S. Parambi

Aseem Chandawarkar, Chairman

Industrial Engineering & Operations Research

(ABSTRACT)

This thesis report discusses the research that led to the interfacing of the latest version of GPSS, namely GPSS/H, and GKS, a graphics system which allows programs to support a wide variety of graphics devices.

The objective of the research was to represent the results of a syntactically and logically correct GPSS simulation model graphically.

A computer program to modify a running GPSS program to a form compatible for graphical emulation was developed. The modifier program identifies the event causing statements in the GPSS program and inserts CALL statements to invoke external subroutines to record these events. Further, it also adds CALL statements to record the statistics for the queues, facilities and storages in the model. The data recorded is in the form of data files. An animation program which takes as input, the data generated by the modified GPSS program was also developed.

The GPSS programmer needs only to take care of the correct modeling of the system and know nothing at all about graphics and animation.
Acknowledgements

At the outset, I would like to thank Dr. Aseem Chandawarkar for agreeing to work with me on a topic which has fascinated me a lot - COMPUTER GRAPHICS. This created a lot of motivation which helped me find a way out of the impasses I encountered within the course of the research. I would also like to thank him for his guidance in the presentation of my thesis work at The Winter Simulation Conference 1986, held at Washington D.C. And finally I would like to thank him for the patience he exhibited in going through the numerous drafts of my thesis report.

Computer Graphics is the subject that helped me cross the last hurdle of my M.S program. For this, I would like to thank Dr. Arvid Myklebust of The Department of Mechanical Engineering. It was from the series of CAD/CAM courses he introduced at Virginia Tech that I came to know of Computer Graphics. Throughout the course he was of great help in answering my questions about the subject matter. I would also like to thank him for giving me access to the lab and a computer account for research. This made it possible for me to work at nights and expedite my work.

I would also like to thank Dr. Marilyn S. Jones for the help she rendered during the early stages of my research. Her suggestions were of great value during my literature review and organizing the research.
I cannot forget my parents, sister and brother for their love and support.

Finally I would like to thank my uncle, , who made it possible for me to pursue graduate studies in The United States.
Table of Contents

CHAPTER 1. INTRODUCTION. ........................................... 1
1.1 BACKGROUND. .................................................... 1
1.2 SIMULATION LANGUAGES. .......................................... 3
1.3 COMPUTER GRAPHICS IN SIMULATION. .............................. 5
1.4 PROBLEM STATEMENT. ............................................. 6
1.5 OBJECTIVE OF THE RESEARCH. .................................... 6

CHAPTER 2. LITERATURE REVIEW. ...................................... 8

CHAPTER 3. DESCRIPTION OF METHODOLOGY. .......................... 12
3.1 DEFINITIONS. ..................................................... 12
3.2 GRAPHICS GENERATORS. .......................................... 13
3.3 INTERFACING METHODOLOGY. .................................... 14

CHAPTER 4. SYSTEM DESIGN. ......................................... 20
4.1 IMPLEMENTATION LANGUAGE. ..................................... 20
4.2 SYSTEM STRUCTURE. .............................................. 21
## List of Illustrations

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System Methodology</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Flow Chart for Program CONVT</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>Flow Chart for Subroutine HIST</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>Flow Chart for Subroutine GRAF</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>Flow Chart for Subroutine ANIM</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>Flow Chart for Subroutine STAT</td>
<td>44</td>
</tr>
<tr>
<td>7</td>
<td>The Main-Menu</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>The Queue/Storage Traffic Menu (animation)</td>
<td>52</td>
</tr>
<tr>
<td>9</td>
<td>Queue names input query (animation)</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>The Facility Status menu</td>
<td>54</td>
</tr>
<tr>
<td>11</td>
<td>The Queue/Storage Traffic Plot Menu</td>
<td>56</td>
</tr>
<tr>
<td>12</td>
<td>Queue Names Input Query (plot)</td>
<td>57</td>
</tr>
<tr>
<td>13</td>
<td>The Facility/Storage Histogram Menu</td>
<td>59</td>
</tr>
<tr>
<td>14</td>
<td>Storages Names Input Query</td>
<td>60</td>
</tr>
<tr>
<td>15</td>
<td>Utilization Histogram (hardcopy)</td>
<td>72</td>
</tr>
<tr>
<td>16</td>
<td>Plot for Queue Traffic (hardcopy)</td>
<td>73</td>
</tr>
<tr>
<td>17</td>
<td>Queue Traffic Animation (hardcopy)</td>
<td>74</td>
</tr>
<tr>
<td>18</td>
<td>Facility Status Animation (hardcopy)</td>
<td>75</td>
</tr>
</tbody>
</table>
List of Tables

Table 1. Example Color Codes. ........................................... 32
Table 2. Example Fill Area Codes. ................................. 33
Table 3. Example Text Codes. ............................................ 36
CHAPTER 1. INTRODUCTION.

1.1 BACKGROUND.

In today's world, a managerial task can be a problem of great magnitude due to the interaction of a large number of factors one encounters within the business world. Consequently, the manager is forced to recognize and understand the interaction of an increasing number of components within the organization and its environment. The process of making a decision involves the identification, evaluation and comparison of alternative courses of action.

Each action is evaluated on the basis of the objectives of the decision maker in the light of a variety of conditions, which may prevail during the period in which the decision is in effect. Because of the number of factors which must be considered and the complexity of their interaction, the manager turns to a system model for a quantitative analysis of the impact of each alternative decision under each set of conditions anticipated. Such an analysis can offer significant insight into the propriety of each decision alternative.
The discussion so far suggests that the analyst needs a vehicle to study the system under consideration. The vehicle to be used for such a purpose has to represent the system under study and must provide a means of predicting the response of the system to a set of input conditions. Since experimentation with the existing or physical system is impractical and sometimes infeasible, the analyst requires a surrogate which provides an efficient and economical means for testing alternative solution proposals. The surrogate used as a testing device is called a model and may be defined as a representation of some aspect of reality without the presence of that reality. This model is only an approximation of reality and is in no way a one-to-one mapping of the actual system. The accuracy of the approximation will usually diminish as the complexity of the real system increases [18]. A simulation model is one of the many types of models which may be used to this end, the others being: iconic, analytical, mathematical, etc.

A simulation model, like a mathematical model, is used to describe the interactive behavior of a system and its environment under prescribed conditions of operation. The inputs to the model are the operating conditions and decision alternatives to be used; while the output is the response of the system, which may be represented by different measures of performance. Whatever the measure of performance, an important feature of a simulation model is its ability to provide quantitative information, which offers a basis for the assessment and comparison of alternative decision strategies. A simulation model traces the behavior of the system by an event basis, appropriately modifying the status of the system as each event occurs. The principal advantages of simulation modeling are:

1. A variety of systems may be successfully modeled.

2. The availability of the technique to professionals who lack proficiency in mathematics and probability.

3. The relative ease in demonstrating what the model does [18].

CHAPTER 1. INTRODUCTION.
Some of the applications of simulation models are:

1. Simulation of the operations at a large airport by an airline company to test the effects of changes of company policies.

2. Simulation of the flux of uncharged particles through a radiation shield to determine the intensity of the radiation that penetrates the shield.

3. Simulation of steel making operations to evaluate the changes in operating practices.


5. Simulation of the operation of a production line to determine the amount of in-process storage space that should be provided.

With the advent of computers, solving complex problems using simulation has become feasible. Simulation has emerged as a powerful technique available to those responsible for the analysis and design of complex systems in many disciplines.

1.2 SIMULATION LANGUAGES.

The translation of a model into a medium which can be interpreted by a computer is the purpose of a programming language. Programming languages can be divided into two categories, when viewed from a simulation standpoint: (i) General purpose languages like FORTRAN, BASIC etc. (ii) Special purpose languages like GPSS, SLAM etc. A special purpose language is also designed to provide a translation mechanism for a broad range of problems; but, in this case, the range is...
limited to a specific generic class of problems. Of particular interest here are simulation languages like GPSS, SLAM, SIMAN, etc. Simulation languages lead to reduced programming and debugging time because their structure assists the analyst in designing the model. Special purpose languages also have the added capability of presenting the results statistically and in neat tabular forms, thereby relieving the model builder of unnecessary programming.

Among the many simulation languages available, GPSS is the most popular and one of the oldest. GPSS is a very popular simulation language that fulfills all the criteria for simulation language selection. GPSS greatly reduces the task of building models for certain types of discrete-event simulations. Thus, GPSS can be applied in most industrial, financial and engineering environments to solve a variety of problem situations. GPSS offers a rich set of semantics, and yet is sparse in its syntax and no other language can be learned as quickly and no other is as compact as GPSS. The latest version of GPSS, namely GPSS/H, is interactive and has the capability of interfacing with external routines written in FORTRAN or PL/1. This is very useful for reading data from or writing data to external files, thus making it possible for postprocessors to analyze the output of a simulation run [19,3]. Further, GPSS/H is an industry standard general purpose simulation language [19].

The GPSS world view involves visualizing units of traffic (transactions) which move along from block to block in the model as the simulation proceeds. This world view is so natural to the modeling of queueing systems that several other notable simulation languages now also offer a similar world view. Thus the effect of this cross fertilization can be found in SLAM, SIMAN, SIMSCRIPT and SIMULA [19].

Disadvantages of GPSS are that it has weak input/output capabilities and weak computational capabilities. These disadvantages can be offset to a great extent by interfacing a GPSS model with one or more FORTRAN subroutines [19].

CHAPTER 1. INTRODUCTION.
### 1.3 COMPUTER GRAPHICS IN SIMULATION.

Recent years have seen increasing use of simulation models for the analysis of manufacturing as well as other systems. Typical methods of displaying simulation output - a printed trace of events occurring over time or a statistical summary - are often inadequate [14]. For simulation to be effective as a design tool, it needs to go beyond providing statistical summary reports.

"All viable simulation languages contain the same type of a trace feature that allows the user to march along time and examine the performance of the system 'line by line'. In addition to it being a tiring and time-consuming task, much of the inter-relationships and the proper view of the overall system is lost" [2, p.659].

Graphics provides an effective means of displaying the results of simulation runs. Some of the benefits of graphics are:

1. **Models with graphics have user confidence to a high degree, and hence the information generated is more likely to be acted upon.**

2. **Analysts understand the 'messages' of a simulation run much more clearly through animated graphics.**

3. **Graphics results in increased user involvement.**

4. **Graphics provides an additional dimension to the verification of a simulation model [7].**

Simulation languages like SIMAN and SLAM have built-in capabilities which present results of a simulation run in business graphics. But no attempt has been made so far to incorporate graphics into a popular simulation language like GPSS.
1.4 PROBLEM STATEMENT.

Since the system of simulation models could be enhanced by the provision of interactive graphics capability, the problem was thus, identified as being, "Presentation of GPSS Results in a Graphical Form ". In the development of an interactive graphics capability, the main consideration was the display of a large amount of output data in a graphical form, so that it could easily and quickly be understood by users. A second consideration was that the system should be easy to use with absolutely no prior training.

1.5 OBJECTIVE OF THE RESEARCH.

With the above considerations in mind, the objective of the research was to interface GPSS/H and a suitable graphics software in order to provide the user with a quick and easy way to view the results of a GPSS simulation model.

With this objective as a basis, the following features were developed for the system:

1. Absolutely no training required by the user to view the results graphically.
2. Animate the arrivals and departures at queues.
3. Plot the number of transactions in a queue with respect to time.
4. Draw histograms to display the utilization of facilities and storages.
5. Animate the status of facilities with respect to time (busy/idle).
6. Animate the arrivals and departures at storages.

7. Plot the storage contents with respect to time.

8. Display eight queues/facilities/storages at a time.

The graphics routines are totally menu-driven and any number of repetitions are possible.
CHAPTER 2. LITERATURE REVIEW.

The introduction of computer graphics in simulation dates back to 1983. Since then, graphics has been provided to enhance the system of simulation models by SIMAN and SLAM.

The use of graphics may be application-oriented or general purpose. The application-oriented approach typically involves the building of a pictorial model and the mapping of the results of the model into a pictorial layout. The problem with the application-oriented approach is that the user is restricted to one system or a class of systems. Not all languages have graphics capabilities. From a review of available literature, it was found that SIMAN and SLAM II are the only languages that have incorporated graphics in a general-purpose form.

The first paper on SIMAN dates back to 1983. Medeiros and Larkin used graphics to present the results of a SIMAN job-shop model [13]. It was done to cater to the needs of different models and was not restricted to one particular problem. The most recent paper dates back to 1985. Lisa A. Pegden, et al, used a package called CINEMA to present the results of a manufacturing system simulation as well as for animation [15].

CINEMA is a general purpose, micro-computer based animation system designed to work with the SIMAN simulation language. Animations are generated by first constructing a SIMAN simulation
model of the system. With minor exceptions, the SIMAN model is constructed without special consideration to whether it will be run with an animation. The CINEMA program is then used to construct a corresponding animation layout which is a graphical depiction of the physical components of the system being modeled. The user, then, executes the SIMAN simulation model in conjunction with the CINEMA layout to generate a graphical animation.

The CINEMA system is designed to run on an IBM PC-AT with 640K bytes of memory, and an 80287 math co-processor. Graphics specific hardware includes a proprietary, high resolution (832 x 624), noninterlaced color graphics board (capable of displaying 16 simultaneous colors from a palette of 4096) and a high resolution, fast scanning, 19-inch diagonal color monitor [15].

In the case of SLAM II, the first work was done in 1983 by Bahram, et al [12]. They incorporated graphics for animation as well as to present the results of an automated assembly loop. Another case of SLAM II dates back to 1984, work done by Imhoff, et al [10]. They used INSIGHT with SLAM II, which is a graphics system that assists a model builder in the development and interpretation of a simulation model. The interactive capabilities of INSIGHT were applied to the Monitored Retrievable Storage (MRS) program. The purpose of the MRS program is to evaluate designs for a receiving and handling MRS facility to package spent fuel and high-level radioactive wastes for storage. A simulation model of the proposed facility was built using SLAM II and the model was analyzed using the capabilities of the graphics editors of INSIGHT.

The shortcomings of the current version of the INSIGHT system are its requirements of a mainframe computer environment and an expensive, high-resolution graphics scope and digitizer pad. While this equipment is available at large companies, access is often difficult. Also, most smaller firms generally do not have this equipment. Therefore, interactive graphics systems will have better accessibility if they can run on smaller computers (i.e. 16 bit processors) and use less expensive graphics software.
A second problem with the current INSIGHT system is that the user must be reasonably proficient with computer systems. The interface to the graphics editors must be extremely user friendly. Another requirement is a basic familiarity with simulation techniques and languages.

Robert Heinzel developed a package to be used in the planning of production and material flow systems [6]. The shortcomings of this development are: It requires a DEC VAX 11/780 with the VMS operating system. An intelligent graphics-terminal IMLAC 3205 with a lightpen is required as peripheral hardware.

Another popular graphics software is TESS [20]. TESS provides a framework for the collection of data during simulation runs, for the selection of data to appear on graphs, and for the generation of presentation graphs and animations. In order to view the results graphically, the user has to provide language statements. For example, suppose that individual observations of the lengths of three queues, QUEUE1, QUEUE2 and QUEUE3 have been collected and jointly labeled QUEUES and the simulated SCENARIO being called CURRENT. The TESS language statement

\[
\text{GRAPH DATA NAMED(QUEUES)}
\]

\[
\text{SCENARIO(CURRENT)};
\]

will produce a graph showing the lengths of three queues over time on the current scenario.

Two papers dating back to 1983 described graphics linked to GPSS. Eaton-Kenway used graphics and GPSS in the simulation of a wire-guided vehicle system [3]. In order to use the package, an engineering drawing had to be reduced by 50%. Further, it required a data tablet and needed additional work from an analyst to pass on the required information for graphical display. The other uses Autogram, a software developed by Autosimulations, Inc. It allows the designer of an automated manufacturing facility to easily construct a simulation model [14]. The output from the
model is displayed graphically as an animated representation of the facility. The UNIX operating system is a prerequisite for this software.

The problems with the existing systems were identified as:

(1) Application oriented

(2) Hardware and operating system dependent

(3) Training required to operate the graphics routines.

The intention of this research was to overcome these problems as far as possible. GPSS is well suited to the simulation of a wide range of problems like transportation, computers, health services, manpower planning, etc. Thus, the objective in constructing the animation program was to provide a general purpose representation of the state of a simulated system over time, which would allow rapid determination of system capacity and utilization.

Graphics software that was decided to be used is 'The Graphical Kernel System (GKS)' which is also the ISO and ANSI standard.

GKS is a graphics system for 2-D graphics which allows programs to support a wide variety of graphics devices. It is defined independently of programming languages. GKS has been defined so that it is equally applicable as a graphics system in a wide range of different environments. In present-day graphics, there is a wide range of devices on the market for input and output. Depending on whether the application environment is simple or complex, an operator may be working on a single device or a number of devices. The aim in GKS is that all these environments should be catered to in a similar way while still allowing the application program to make best use of both the total environment available and the specific characteristics of each device [4].
CHAPTER 3. DESCRIPTION OF METHODOLOGY.

This chapter presents the conceptual development of the system. The chapter first covers some terms necessary in understanding the concepts, then proceeds to discuss the methodology.

3.1 DEFINITIONS.

The terminology used in simulation varies widely with the language. Therefore, to make the material contained in the rest of the document easier to comprehend, a list of terms and concepts used in GPSS and relevant to this research are explained in the following text.
**QUEUE**  An entity that can accept only one transaction at a time; it can be used to model single server components of a system (e.g., bank cashier, machine tool producing single items.) [2].

**FACILITY**  An entity that will accommodate several transactions (e.g., buffer storage in a computer; capacity of a hotel.) [2].

**STORAGE**  A statistics gathering entity that can accept any number of transactions (e.g., customers waiting for service at a bank; jobs waiting for computer processing.) [2].

**EVENT**  Events are things that occur at some point in time and change the state of some parameter in the system.

**ATTRIBUTE**  Attributes are characteristics of events and entities. (e.g., an event is: "a machine just finished processing a part" and the attribute is "which machine finished, when it finished, etc., ".)

Queues, facilities and storages are generally referred to as "entities" in GPSS.

### 3.2 GRAPHICS GENERATORS.

Graphics generators fall under two categories: real-time and postprocessing. With real-time, the graphics is carried out during the simulation. The problem with this mode is that it is very time consuming and not very flexible - if the user wishes to rerun the graphics program, the simulation model also has to be rerun. Thus, real-time processing is a time-consuming and costly affair. The solution is postprocessing or running the graphics program independently of the simulation pro-
gram. The postprocessor reads data from the data file created during the simulation run and uses this for repeated animation runs. The graphics routines developed for this research, work in a postprocessing mode for their obvious advantage.

A description of the methodology on a macro level is shown in Figure 1. As shown in Figure 1, the input to the entire system is a syntactically and logically correct GPSS program.

This program is read line by line by CONVT, a FORTRAN program which is the "modifier". CONVT converts the original GPSS program into a modified version, GMOD, which is graphics compatible. In other words, at the appropriate points in the input program, executable statements, which call the external FORTRAN subroutines, GRAPH and CHART are inserted. These subroutines create the data files which form a database. The database contains information pertaining to the different queues, facilities and storages in the model for the course of the simulation run. The database becomes the input to the graphics program DISP, which displays the results graphically.

3.3 INTERFACING METHODOLOGY.

An important point to note about GPSS is that everything is based on queues, facilities and storages [19]. Transactions like parts to be machined or customers at a bank arrive at a facility to be served, the machine or the bank being the facility. When a facility is busy, the arriving transaction waits at a particular queue until the required facility is free.

GPSS blocks that could cause events or a change in the simulated system were thus identified as: QUEUE and DEPART (associated with a queue), SEIZE and RELEASE (associated with a facility), ENTER and LEAVE (associated with a storage). Since such blocks are event causing, all
Figure 1. System Methodology.

CHAPTER 3. DESCRIPTION OF METHODOLOGY.
information associated with them is recorded for later use by postprocessors. QUEUE, SEIZE and ENTER are associated with incoming transactions and are assigned an attribute of +1, while the other three have an attribute of -1 since they are associated with outgoing transactions.

The ability of GPSS to interface with external PL/I or FORTRAN subroutines, established the link between GPSS and GKS, the graphics software. These subroutines are used to record data in a file or read data from a file. They were used in this research to create data files containing important information on all the queues, facilities and storages in the simulation model. The events and statistics for the simulation run form the information to be recorded. The subroutines are invoked by imbedding CALL statements in the GPSS program.

The format of the CALL statement is shown below.

```
BCALL &name (a, b, ..................,n)
```

where 'name' is the name of the external subroutine and 'a', 'b', etc. are arguments passed on by the GPSS program to the FORTRAN subroutine.

To create GMOD, CONVT has to identify the mentioned GPSS blocks and insert the appropriate BCALL statement after every occurrence of such a block, thereby notifying the occurrence of an observable change. To achieve this, CONVT reads the input GPSS program line by line from the file, INPUT DATA A1. Every time one of the blocks is encountered, an associated flag is set ON. The SET ON flag causes the BCALL statement to be printed following the block. Conversely, the flag in the SET OFF state does not print a BCALL statement. As an example, let us take the following program segment.

```
SEIZE MACH
ADVANCE 9
```
When SEIZE is encountered, the flag to print the BCALL statement related to SEIZE is set on. It takes the form

```
BCALL &GRAPH (Cl,'MACH',' ',' ',' ',200,1)
```

CONVT after analysis deciphers MACH to be the name of a facility and gathers all the other important attributes associated with this facility. With respect to the above BCALL statement:

- GRAPH is the name of the externally interfacing FORTRAN subroutine.
- Cl is the simulation clock time.
- MACH is the name of an entity (facility here).
- 200 is the identification tag for a facility. (100 being for a queue and 300 for a storage.)
- 1 is the tag for an incoming transaction. (-1 being for an outgoing.)

The modified program segment is shown below.

```
SEIZE MACH
BCALL &GRAPH (Cl,'MACH',' ',200,1)
ADVANCE 9
```

When the modified program is rerun, immediately after the event block (SEIZE), the external subroutine GRAPH is invoked. This causes all the information on the facility to be recorded in the data file being created by GRAPH.
The entire program is scanned and the associated BCALL statements are inserted immediately after the occurrence of statements causing an observable change. This takes care of recording the events and attributes for each entity in the model.

As with any other simulation language, GPSS also provides a statistical report. For example, it gives information on the number of transactions that passed through a queue, the utilization of facilities and storages, etc. The statistics are collected by imbedding formatted statements in a GPSS program. In order to display the statistical information graphically in a postprocessing mode, they have to be contained in a file. Thus, another function of CONVT is to create this data file. Statements to collect these statistics are inserted in the GPSS program. The statistics gathered are then passed on as arguments to the CALL statement to invoke the external FORTRAN subroutine CHART. Subroutine CHART, thus, creates the statistics data file.

The following is an example to record the utilization of facility MACH, in the file created by subroutine CHART.

\[
\text{LET } \&RQ = \text{FR$\text{MACH}}
\]

In the above statement, variable RQ takes the utilization value of facility MACH. This value can now be passed on as an argument in the CALL statement to subroutine CHART along with the other parameters. It takes the following form:

\[
\text{CALL } \&\text{CHART('202','MACH', } \&\text{RQ,200)}
\]

With respect to the above statement:

- 202 is the identification tag for the utilization value.
- MACH is the name of the facility.
• RQ is the variable that takes the utilization value.

• 200 is the identification tag for a facility.

The subroutines GRAPH and CHART when invoked by GMOD create the database, which becomes the input to the graphics program.

The statements inserted by CONVT in GMOD to create the data files are compatible with the format for the READ statements in DISP, the graphics program. This gets rid of the burden of learning any operational commands to view the results graphically.

The different forms of displays like the histogram and plots are handled by different subroutines. For each user selected name, the relevant data file is scanned for a possible match. When a match occurs, the matched data becomes the input to the current graphics subroutine.

Since GPSS/H is interactive, the user can change the input parameters to the model at ease. As mentioned before, the graphics program DISP is menu-driven and no time is lost in viewing the effects of the changes in the input parameters.
CHAPTER 4. SYSTEM DESIGN.

This chapter presents the technical constructs used in the design of the system. This includes the overall program structure, description of the system files, data structure and a discussion of the underlying algorithms that are critical to the operation of the system.

4.1 IMPLEMENTATION LANGUAGE.

Once the general model concepts were decided upon, the computer language with which to implement the system was chosen. Since the subroutines interfacing externally with GPSS, as well as GKS, are FORTRAN based, it was decided to use FORTRAN.

The system was implemented on the IBM 4341 mainframe computer under the VM operating system. The graphics is displayed either on the Tektronix 4107 or 4115. These are high resolution color graphics terminals situated in the CAD/CAM Laboratory.
4.2 SYSTEM STRUCTURE.

There are two main programs, CONVT and DISP, two interfacing subroutines, GRAPH and CHART and several Input/Output files that comprise the system. Figure 1 shows the relationship among these system elements. In the following pages each element of the system is explained in greater detail.

4.2.1 INPUT GPSS PROGRAM.

This is a syntactically and logically correct GPSS program residing in file, INPUT DATA A1. The program becomes the input to the FORTRAN program, CONVT and is converted to GMOD.

The GPSS programmer has to bear in mind that the OPERCOL statement is not to be used in the original program. This is because CONVT inserts the OPERCOL statement.

The function of the OPERCOL statement is to declare the margin limit for the statements in the GPSS program. Since the BCALL statements inserted by CONVT are quite lengthy, declaration by CONVT was thought to be more appropriate.

4.2.2 PROGRAM CONVT.

This program might be referred to as the modifier program. A flow chart showing the underlying principle is shown in Figure 2.
Figure 2. Flow Chart for Program CONVT.
The function of this FORTRAN program is to read an executable GPSS program from the file, INPUT DATA A1, analyze it and make it graphics compatible. The changes made are the CALL statements inserted to invoke the external FORTRAN subroutines. As mentioned in Section 3.3, this is done by identifying the keywords: QUEUE, DEPART, SEIZE, RELEASE, ENTER and LEAVE. Besides, the program should also keep track of the names of the queues, facilities and storages. The names are used later to insert statements to collect statistics on entities.

After every occurrence of a keyword, a CALL statement that records the simulation clock time, the associated name and the identification tags has to be inserted. 100 is set as the identification tag for a queue, 200 for a facility and 300 for a storage. The identification tag for an incoming transaction is 1 and -1 for an outgoing transaction. This set of CALL statements to subroutine GRAPH creates the event data file.

CONVT makes use of flags to trigger the appropriate actions. The main flags and their purpose are the following:

1. Q: The letter Q has been read
2. D: The letter D has been read
3. S: The letter S has been read
4. O: The letter R has been read
5. E: The letter E has been read
6. L: The letter L has been read
7. FLAGQ: the word QUEUE has been identified.
8. FLAGD: the word DEPART has been identified.

CHAPTER 4. SYSTEM DESIGN.
9. FLAGS : the word SEIZE has been identified.

10. FLAGR : the word RELEASE has been identified.

11. FLAGE : the word ENTER has been identified.

12. FLAGL : the word LEAVE has been identified.

13. COUNT : indication of the first letter following a keyword.

The input program is read off from file, INPUT DATA A1 and GMOD is created in file, GMOD GPSS A1. Each line of file, INPUT DATA A1 is read character by character into an array R(I) of size 80. If the first character encountered in a line happens to be a "*", no action is taken since it is a comment. In this case, the entire line is duplicated into file, GMOD GPSS A1. After the possible comment statements, the first set of statements inserted are the following:

```
OPERCOL 60
```

meaning that GMOD has a limitation on character occupancy till the 60th column.

```
EXTERNAL &GRAPH

LOAD &GRAPH

EXTERNAL &CHART

LOAD &GRAPH
```
The above statements declare that GRAPH and CHART are external subroutines interfacing with GMOD.

\begin{verbatim}
INTEGER &IQ
REAL &RQ
\end{verbatim}

In the above statements, IQ is declared as the variable to collect the integer-valued statistic and RQ, the variable to collect the real-valued statistic.

CONVT then reads the remaining lines of file, INPUT DATA A1. If R(I) is a blank (' ') variable COUNT is set to 1. If R(I) is a Q/D/S/R/E/L then flag Q/D/S/R/E/L is set to 1. When flag Q/S/R/E/L is set to 1, the program transfers control to the particular routine to analyze the remaining elements in the array R(80). As an example, let us take the word QUEUE and analyze the routine to identify it. (The routines to identify the other keywords are based on the same lines.)

Let us say the letter Q was read in. Since R(I) was Q, flag Q is 1. When the next element of array R(80) is made current, control passes to the section to check for the occurrence of the letter U. If yes, the next element in R(I) is checked for the letter E. R(I) is checked in succession for the occurrence of the letters U, E, U and E, in the same sequence as they appear in the word QUEUE. To accomplish this, a counter QCNT is made use of. Everytime control comes to the section to check for the word QUEUE, the value of the counter QCNT is incremented by 1. When 1, an occurrence of the letter U sets flag U1 to 1. Similarly when QCNT equals 2, an occurrence of E sets E1 to 1, 3 and U sets U1 to 1, 4 and E sets E1 to 1. An occurrence of all the letters in QUEUE sets FLAGQ to 1, indicating QUEUE has been identified. It is also possible that the word could be something other than QUEUE by having a letter trailing the last E. In this case, QCNT will be 5 and this causes all flags associated with QUEUE to be reset to 0, thereby preventing a false identification.
Once a keyword has been identified, COUNT is set to 2. This causes branching off to the subroutine NAME that will determine the name of the queue, facility or storage that was last identified. If not, COUNT is set to 1 and the search for a keyword starts from the beginning.

Over and above determining the name, the purpose of the subroutine NAME is to overcome one of the limitations posed by the FORTHX compiler [8]. Subroutines GRAPH and CHART use the FORTHX compiler. One of the arguments passed to the subroutine is the name of the queue/facility/storage. It was decided at the outset to permit a maximum length of fifteen characters for names. FORTHX has a limitation of four characters per variable, which means the maximum word length is four. In order to overcome this problem, the following approach was adopted.

The name is read by CONVT into an array XQ of size 27. CONVT then splits the name into groups of four characters. Each group is then led by a quote and followed by a quote and comma. For example, let us take the word MACHINE. This would be split in the following manner.

```
'MACH','INE' ',' ','
```

This split name is stored in a temporary file. Each of the four groups is then passed on to four character variables in the subroutines GRAPH and CHART. It would appear thus as,

```
BCALL &GRAPH(C1, 'MACH','INE' ',' ',' ','200',1)
```

GRAPH treats each of the four groups as an argument while creating the data file. The groups are written adjacently and the original name is formed. Every time the subroutine NAME is called, the subroutine WNAME is called immediately after it. The purpose of this subroutine is to record the names of the queues, facilities and storages in separate files without duplicates. To handle the names of queues, subroutine WNAME calls subroutine QUE, and this creates unit 65 containing the names of all queues. Subroutine FACTY is called to create unit 66 for facility names and subroutine STOR to create unit 67 for storage names.
Once a keyword has been identified, the contents of array $R(80)$ and the BCALL statement to record the event are written into file, GMOD GPSS A1.

After these changes have been made, units 65, 66 and 67 are accessed and statements to collect the statistics for each queue, facility and storage are inserted. These statements are inserted just before the END statement. The statistics supplied by GPSS are in mixed form (integer and real). Before the integer values are passed on to CHART, they are converted to real by making the real variable $RQ$ equal to the integer, $IQ$. This is done to avoid any problems that may arise when the graphics program reads in these values. (All values are read in as real by DISP).

When GMOD is executed, it calls the external subroutines GRAPH and CHART to create the data files.

4.2.3 SUBROUTINES GRAPH and CHART.

These subroutines called by GMOD create the database consisting of the data files, unit 96 and unit 97. The format in which the data is recorded is mentioned in Section 4.2.4.

4.2.4 DATA FILES.

The unit 96 and unit 97 data files comprise the database. The data in unit 96 created by subroutine GRAPH has the following format:

- the simulation clock time
• name of the queue, facility or storage

• identification tag for a queue, facility or storage

• transaction tag.

CHART creates unit 97 with data in the following format:

• statistic identification tag

• name of the queue, facility or storage

• the relevant statistic

• identification tag for a queue, facility or storage.

4.2.5 PROGRAM DISP.

This section first describes some of the concepts in GKS, and later explains how these were used to create the program DISP.

In GKS, pictures are considered to be constructed from a number of basic building blocks. These basic building blocks, or primitives, as they are called, are of a number of types each of which can be used to describe a different component of a picture. The four main primitives in GKS are:

1. polyline: which draws a sequence of connected line segments.

2. polymarker: which marks a sequence of points with the same symbol.

3. fill area: which displays a specified area.
4. text: which draws a string of characters.

Associated with each primitive is a set of parameters which is used to define particular instances of that primitive. The following literature explains each GKS building block in greater detail and their primitives used in this research.

4.2.5.1 POLYLINE.

The main line drawing primitive is the polyline which is generated by invoking the function:

\[
\text{CALL GPL (N, XPTS,YPTS)}
\]

where XPTS and YPTS are arrays giving N points (XPTS(1),YPTS(1)) to (XPTS(N),YPTS(N)). The polyline generated consists of N-1 line segments joining adjacent points starting with the first point and ending with the last.

The color (attribute) of the line drawn can be set by calling the function:

\[
\text{CALL GPLCI (N)}
\]

where N is an integer and stands for a particular color.
4.2.5.2 POLYMARKER.

In order to mark a point of a particular X-Y co-ordinate, GKS provides the primitive polymarker. A polymarker is generated by invoking the function:

```
CALL GPM (N, XPTS, YPTS)
```

where the arguments are the same as for the polyline function, namely XPTS and YPTS are arrays giving N points (XPTS(1), YPTS(1)) to (XPTS(N), YPTS(N)). Polymarker places a centered marker at each point.

Polymarker may be specified with different representations and colors. This is done by assigning different values to a polymarker attribute called the polymarker and polymarker color index. The polymarker index is set by invoking the GKS function:

```
CALL GSPMI (M)
```

and the polymarker color index is set by the GKS function:

```
CALL GSPMCI (N)
```

where M and N are the desired values of the polymarker and polymarker color index.
4.2.5.3 FILL AREA.

GKS provides a fill area function to fill a particular area. Defining an area is a fairly simple extension of defining a polyline. An array of points which defines the boundary of the area is specified. A fill area may be generated by invoking the function:

\[ \text{CALL GFA (N, XPTS, YPTS)} \]

where \( \text{XPTS} \) and \( \text{YPTS} \) are arrays giving \( N \) points \((\text{XPTS}(1), \text{YPTS}(1)) \) to \((\text{XPTS}(N), \text{YPTS}(N))\). A fill area style may be generated by invoking the function:

\[ \text{CALL GFAIS (N)} \]

where \( N \) is the desired value of the fill area style index. Filled areas may be distinguished by their filling style and color. The fill area style is the way in which the area is filled. Some of the available styles are shown in Table 2.

The fill area color code is set by invoking the function:

\[ \text{CALL GSFACI (N)} \]

where \( N \) is the code for a particular color. Some of the representations for the color codes are shown in Table 1.
<table>
<thead>
<tr>
<th>CODE</th>
<th>COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WHITE</td>
</tr>
<tr>
<td>2</td>
<td>RED</td>
</tr>
<tr>
<td>3</td>
<td>GREEN</td>
</tr>
</tbody>
</table>
Table 2. Example Fill Area Codes.

<table>
<thead>
<tr>
<th>CODE</th>
<th>STYLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOLID</td>
</tr>
<tr>
<td>2</td>
<td>PATTERN</td>
</tr>
<tr>
<td>3</td>
<td>HATCH</td>
</tr>
</tbody>
</table>
4.2.5.4 TEXT.

GKS has a text primitive which is used to title pictures or place labels on them as appropriate. A text string may be generated by invoking the function:

CALL GTX (X, Y STRING)

where X, Y is the text position and string is a string of characters. Associated with the text primitive are a number of attributes.

The character height attribute determines the height and width of characters in the string. The character height is set by the function

CALL GSCHH (H)

where H is the character height. The representation of the text is accessed via the text index. The text index is set by the function

CALL GSTXI (N)

where N is the desired value of the text index. Some of the available text indices are shown in Table 3. As with the other primitives, the color for the text can be set by the function:

CALL GSTCXI (N)
4.2.5.5 SEGMENTS.

GKS provides a facility called a segment whereby a particular picture portion (segment) may be marked off with an identification number. Later on, when changes to the picture are to be made, only the relevant picture segments need to be manipulated to produce the desired effects instead of the entire picture.

An important segment attribute is its visibility. This determines whether a segment will be displayed or not. By default, when a segment is created it is visible. The visibility attribute may be changed by the function:

```
CALL GSVIS (ID, VIS)
```

where \( N = 0 \) specifies invisibility for segment number \( ID \) and \( N = 1 \) specifies visibility.

By suitable combinations of primitives and values of primitive attributes, complex pictures may be described to GKS with suitable annotations and titles.

Since the CALL statements are FORTRAN based, they could be very easily imbedded in a program designed to meet a specific set of needs.

The function of the graphics program, DISP, was mentioned in Chapter two as the objective of the research. Each of these functions is executed by a subroutine devoted for the specific purpose. DISP is menu-driven and based on the viewer's choice, control passes on to the respective subroutine. The principle on which each subroutine is built is explained in the sections that follow.
Table 3. Example Text Codes.

<table>
<thead>
<tr>
<th>CODE</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SIMPLEX ROMAN</td>
</tr>
<tr>
<td>2</td>
<td>SIMPLEX SCRIPT</td>
</tr>
<tr>
<td>8</td>
<td>TRIPLEX ITALIAN</td>
</tr>
</tbody>
</table>
4.2.5.6 SUBROUTINE HIST.

This subroutine creates histograms representing the utilization of facilities and storages. The flow chart for this subroutine is shown in Figure 3 and a hardcopy of the display in Figure 15. It is capable of displaying a maximum of either eight facilities or storages.

The X and Y axes are drawn using the polyline primitive and labelled using the text primitive. Since a histogram is nothing but a display of bars (rectangles) adjacent to each other, the fill area primitive is used to create the bars and its color attribute is used to differentiate one from the other. The width of the rectangle was decided to be ten units and a DO loop was written to generate the four X coordinates of each of the eight rectangles. This was easy to code, since X(1) and X(4) have the same value. Similarly, X(2) and X(3) have the same value and are offset from X(1) and X(4) by ten units. Coming to the next rectangle, its X(1) and X(4) are the same as the previous rectangle's X(2) and X(3). Likewise for the other rectangles. As far as the Y coordinates are concerned, the Y(1) and Y(2) are the same for each rectangle and thus, this is predetermined for each of the eight rectangles. The Y(3) and Y(4) for each rectangle is determined from the percent utilization value read from the statistics data file. The user selected names are stored in a file. From this file, each name is retrieved and the data file is scanned for a possible match. When a match occurs, the percent utilization for this name is used as the input value to subroutine HIST. This value is then customized to fit the co-ordinate system on the display screen and from it the Y(3) and Y(4) co-ordinates are calculated. Now we have the four X and Y co-ordinates sufficient for a fill area. This rectangle is assigned a color code and drawn using the fill area primitive. After this, it is labelled with the name right beneath the rectangle. The program then gets the next name and does the same. The color code integer is incremented by one every time a new name is fetched, to ensure that no two rectangles have the same color. After the entire display is made, the user has the option of making another selection or going to the main-menu.

CHAPTER 4. SYSTEM DESIGN.
Figure 3. Flow Chart for Subroutine HIST.
4.2.5.7 SUBROUTINE GRAF.

This subroutine plots the number of transactions at a particular queue or storage with respect to time. For this routine it was decided to display a maximum of either eight queues or storages.

The flow chart for this subroutine is shown in Figure 4 and a hardcopy of the display in Figure 16. The X and Y axes are drawn using the polyline primitive and for the time being, only the Y axis is labelled. In order to label the X axis, the event data file is scanned once, to determine the length of the entire simulation run. The simulated period is then divided into eight equal time intervals and the X axis labelled accordingly.

The names selected by the viewer are stored in a file. Each name is retrieved one at a time and the data file is scanned for a possible match. A counter which is initialized to zero before every name is fetched, is incremented by the relevant transaction value \( \frac{1}{-1} \) when a match occurs. The counter value forms the Y co-ordinate and the simulation clock time last read forms the X co-ordinate. After customizing the X-Y co-ordinate, the polymarker primitive is invoked to draw a mark with a color code that was set for the current name. The data file is further scanned and new X-Y co-ordinates are calculated till none remain for this name. After the last mark is made, the X-Y co-ordinates calculated for the polymarkers are made use of by the polyline primitive to connect the points with straight lines, thus giving a plot. Using the color code in use, the name is written using the text primitive. The same process is repeated for all the names selected and each is represented with a different color.

After exhausting the viewer’s selection list, the user has the option to make another selection or return to the main menu.
Figure 4. Flow Chart for Subroutine GRAF.
4.2.5.8 **SUBROUTINE ANIM.**

The function of this subroutine is to display (animate) the arrivals and departures at queues and storages. This is depicted by varying the height of a bar with respect to time. The program is designed to handle a maximum of either eight queues or storages. A differently colored strip is allotted to each entity. The flow chart for ANIM is shown in Figure 5 and a hardcopy of the display in Figure 17.

The subroutine first reads NQ, the number of queues or storages the user wishes to view. This causes the polyline routine to generate NQ number of scale markings and they are labelled with the respective names. Each of these is made into a separate segment. Next, for each of the NQ selections, fifteen rectangular boxes of one square unit are created using the fill area primitive with a specific color allotted to it, each box representing a transaction. Each box is created as a separate segment and their visibility is initially set to zero (invisible). In order to keep track of the number of transactions currently in each queue or storage, a counter is allotted for each. For each name keyed in by the user, the event data file is scanned for a possible match. If a match occurs, the transaction tag (1 for incoming, -1 for outgoing) is used. The value of the tag is used to increment the counter allotted for this name. The counter value N, is used to set the visibility of the Nth segment and all segments less than N to 1 (visible). For example, if the counter has a value two then segments one and two would be made visible, indicating that currently there are two transactions for that name. The same principle is applied to each of the eight name segments. By setting the visibility to 1/0 causes the height of the strip to move with time. The strip can indicate a maximum of fifteen units. Any number greater than this is depicted by words using the text primitive.

After the display is complete the viewer has the option of making another selection or returning to the main menu.
Figure 5. Flow Chart for Subroutine ANIM.
4.2.5.9 SUBROUTINE STAT.

The function of this subroutine is to animate the status of facilities with respect to time. It is capable of handling a maximum of eight facilities. The flow chart for ANIM is shown in Figure 6 and a hardcopy of the display in Figure 18.

The fill area primitive is made use of to do the animation. A box (rectangle) is devoted for each of the eight facilities. The transaction value read in (1/-1) is used to set the color attribute of the box. If the transaction tag is a +1, the color is set to green and to red if the transaction tag is a -1. The fill area primitive is then called to display the rectangle, which depicts the status.
Figure 6. Flow Chart for Subroutine STAT.
CHAPTER 5. PROBLEMS ENCOUNTERED.

One of the main problems, as mentioned before, is the limitation posed by the FORTHX compiler.

The other major problem is memory space in the computer. The original GPSS program is expanded to a great extent by the inclusion of the CALL statements. This problem is further augmented when the event data file is created.

Another problem encountered was the speed of the animation. The animation proceeds very slowly during peak hours when there are many users in the system.

Future systems could be implemented on the IBM PC/AT or other fast micro-computers which give sole devotion, since PC versions of GPSS are now available. The graphics software PHIGS, which has three dimensional capabilities could also be used [9]. These alternatives could alleviate the problems to some extent.
CHAPTER 6. CONCLUSION.

The need for a system like this has evolved from the increasing number of beneficiaries from a simulation model. This system is a tool that can cater to the needs of almost any level in the managerial hierarchy in an organization, since it was designed for those who lack computer proficiency. (The interactive features of GPSS/H and the menu facility of the developed graphics program made this possible.) The benefit derived from the system should assist commercial, government and educational institutions in evaluating proposed alternatives.

Future work could focus on displaying different types of entities at the same time. The system could be made more powerful by displaying the pictorial layout of the entire system. Features like freezing the animation could also be incorporated. The time spacing as recorded in the event data file could be used to proportionally space successive frames during the animation run. This feature could give a more realistic picture of the simulation run.

Since the graphics program is driven by data files and menus, a more powerful version of CONVT could be written. The purpose of this updated version would be to convert simulation programs written in languages that have the ability to interface with external subroutines.
BIBLIOGRAPHY.


Appendix A. USER'S GUIDE.

A.1. INTRODUCTION.

This document is the User's Guide to the system which was developed to present the results of a GPSS Simulation model graphically. It is assumed that the input GPSS model is a syntactically and logically correct GPSS program residing in file, INPUT DATA A1. If viewing the results is what the user is currently interested in, it is suggested that either The Tektronix 4115 or 4107, the high resolution graphics terminals be used.

A.2. RUNNING THE SYSTEM.

As mentioned in the preceding pages, the different elements of the system use different compilers like FORTVS, FORTHX, GPSS/H and so on. Thus it makes it mandatory that, before the execution of a particular element in the system, the necessary libraries are linked. For example, before
running the modifier program CONVT, the FORTVS library has to be linked. After this, comes the execution of GMOD and it interfaces with subroutines GRAPH and CHART using the FORTHX libraries. In order to leave the burden of switching over to the different libraries to the system, an EXEC file, SYSTEM was created. Thus to execute the system, the user types SYSTEM. This executes the different operations shown in Figure 1 and takes the user directly to the main-menu to display the results graphically.

The main-menu supplied by the graphics program DISP is shown in Figure 7.

In the following pages, the action taken by selecting each of the above choices will be dealt with in the order they are listed.

1. **ANIMATE QUEUE/STORAGE TRAFFIC**

The selection of this menu supplies the user with the queries shown in Figure 8.

Assuming that the user has selected 1, the system supplies the user with the query in Figure 9. The system displays the names of all the queues in the model to ease the selection process.

After all the names have been entered, DISP animates the arrivals and departures of transactions at the queues selected. The same holds true, had the selection been storages. If desired, the user can return to the menu shown in Figure 8 or return to the main-menu.

2. **ANIMATE THE STATUS OF FACILITIES (BUSY/IDLE)**

The selection of this menu supplies the user with the query shown in Figure 10.

In order to help the user with the selection of facility names, the system prints on the screen the names of all the facilities in the simulation model. Based on number requested, the system asks the user to input the required names. This causes DISP to animate the status of the facilities chosen.
THE MAIN-MENU

1. ANIMATE QUEUE/STORAGE TRAFFIC.
2. ANIMATE THE STATUS OF FACILITIES (BUSY/IDLE).
3. PLOT OF QUEUE/STORAGE TRAFFIC.
4. UTILIZATION HISTOGRAM FOR FACILITIES/STORAGES.

ENTER YOUR CHOICE (1/2/3/4):

Figure 7. The Main-Menu.
ENTER YOUR CHOICE (1 OR 2)

1. ANIMATION FOR QUEUES
2. ANIMATION FOR STORAGES

Figure 8. The Queue/Storage Traffic Menu (animation).
ENTER THE NUMBER TO BE ANIMATED (QUEUES).
(*** THE NUMBER SHOULD NOT BE GREATER THAN 8 ***)

Figure 9. Queue names input query (animation).
ENTER THE NUMBER TO BE ANIMATED (FACILITIES).
(*** THE NUMBER SHOULD NOT BE GREATER THAN 8 ***)

Figure 10. The Facility Status menu.
3. PLOT OF QUEUE/STORAGE TRAFFIC

The selection of this menu supplies the user with the query shown in Figure 11.

Assuming that the user has selected 1, the system supplies the user with the query in Figure 12. The system displays the names of the queues in the model to ease the selection process.
ENTER YOUR CHOICE (1 OR 2)

1. PLOT FOR QUEUES
2. PLOT FOR STORAGES

Figure 11. The Queue/Storage Traffic Plot Menu.
ENTER THE NUMBER TO BE PLOTTED (QUEUES).
(*** THE NUMBER SHOULD NOT BE GREATER THAN 8 ***)

Figure 12. Queue Names Input Query (plot).
After all the names have been entered, DISP makes a plot of the number of transactions vs the simulation time for the queues selected. The same holds true, had the selection been storages. If desired, the user can return to the menu shown in Figure 11 or return to the main-menu.

4. UTILIZATION HISTOGRAM FOR FACILITIES/STORAGES.

The selection of this menu supplies the user with the queries shown in Figure 13.

Assuming that the user has selected 2, the system supplies the user with the query in Figure 14.

Once the picture is drawn, the user has the option of returning to the menu shown in Figure 13 or returning to the main-menu.
ENTER YOUR CHOICE (1 OR 2)
1. FACILITY UTILIZATION
2. STORAGE UTILIZATION

Figure 13. The Facility/Storage Histogram Menu.
ENTER THE NUMBER NEEDED (STORAGES).
(*** THE NUMBER SHOULD NOT BE GREATER THAN 8 ***)

Figure 14. Storages Names Input Query.
Appendix B. AN EXAMPLE PROBLEM.

B.1. INTRODUCTION.

This Section takes an example GPSS program and shows the different elements created by the system.

B.2. PROBLEM DESCRIPTION.

The operating system of a computer installation controls the processing of jobs sent by three independent terminals. When a job arrives from a terminal, it enters a queue in front of the Input-Output unit 1 (IO1) and waits for processing. All of the queues in the system are handled in arrival order (First-Come First-Served). After the job is processed at IO1, it enters the queue in front of the CPU, it joins the queue in front of the Input-Output unit 2 (IO2). There is no limit on the number of jobs waiting in the queue in front of IO1 and CPU. But there is a limit of 10 jobs al-
lowed at IO2 (including the processing). If 10 jobs are at IO2 (9 in queue and 1 in processing), then any subsequent arrivals to IO2 are diverted to another I/O unit that is not included in our study.

Statistics of the distributions of interarrival times between jobs and the processing times at each service center are tabulated below. All values are stated in milliseconds (ms).

**INTERARRIVAL TIMES**

<table>
<thead>
<tr>
<th>Terminal Number</th>
<th>Distribution of the Interarrival Times</th>
<th>Mean Value of the Interarrival Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exponential</td>
<td>2000</td>
</tr>
<tr>
<td>2</td>
<td>Exponential</td>
<td>4000</td>
</tr>
<tr>
<td>3</td>
<td>Exponential</td>
<td>1250</td>
</tr>
</tbody>
</table>

**PROCESSING TIMES**

<table>
<thead>
<tr>
<th>Service Center</th>
<th>Distribution of the Processing Times</th>
<th>Mean Value of the Processing Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exponential</td>
<td>450</td>
</tr>
<tr>
<td>2</td>
<td>Exponential</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>Exponential</td>
<td>580</td>
</tr>
</tbody>
</table>

The GPSS source code listing for this problem is shown on the following pages.
THE INPUT GPSS PROGRAM

* TIME UNIT = MILLISECONDS

SIMULATE

XPDIS FUNCTION RN1,C24
0.0,0.0/0.1,0.104/0.2,0.222/0.3,0.355/0.4,0.509/0.5,0.69/
0.6,0.915/0.7,1.2/0.75,1.38/
0.8,1.6/0.84,1.83/0.88,2.12/0.9,2.3/
0.92,2.52/0.94,2.81/0.95,2.99/
0.96,3.2/0.97,3.5/0.98,3.9/0.99,4.6/
0.995,5.3/0.998,6.2/0.999,7/0.9997,8

IO2S STORAGE 10

RTIME TABLE M1,800,800,10

GENERATE 2000,FN$XPDIS
QUEUE SYSTEM
SEIZE IO1
ADVANCE 450,FN$XPDIS
RELEASE IO1

SEIZE CPU
ADVANCE 500,FN$XPDIS
RELEASE CPU

TRANSFER BOTH,,IOX1
ENTER IO2S
PRIORITY 1
SEIZE IO2
ADVANCE 580,FN$XPDIS
RELEASE IO2
LEAVE IO2S
TABULATE RTIME
DEPART SYSTEM
TERMINATE 1

IOX1 DEPART SYSTEM
TERMINATE

GENERATE 4000,FN$XPDIS
QUEUE SYSTEM
SEIZE IO1
ADVANCE 450,FN$XPDIS
RELEASE IO1

SEIZE CPU
ADVANCE 500,FN$XPDIS
RELEASE CPU

TRANSFER BOTH,,IOX2
ENTER IO2S
PRIORITY 1
SEIZE IO2
ADVANCE 580,FN$XPDIS
RELEASE IO2

Appendix B. AN EXAMPLE PROBLEM.
LEAVE IO2S
TABULATE RTIME
DEPART SYSTEM
TERMINATE 1

IOX2 DEPART SYSTEM
TERMINATE

GENERATE 1250,FN$XPDIS
QUEUE SYSTEM
SEIZE IO1
ADVANCE 450,FN$XPDIS
RELEASE IO1

SEIZE CPU
ADVANCE 500,FN$XPDIS
RELEASE CPU

TRANSFER BOTH,,IOX3
ENTER IO2S
PRIORITY 1
SEIZE IO2
ADVANCE 580,FN$XPDIS
RELEASE IO2
LEAVE IO2S
TABULATE RTIME
DEPART SYSTEM
TERMINATE 1

IOX3 DEPART SYSTEM
TERMINATE
START 75
END

This input GPSS program is made graphics compatible by imbedding the CALL statements in it by GMOD. A listing of GMOD is shown on the following pages.
GMOD LISTING

* OPERCOL 60
  EXTERNAL &GRAPH
  LOAD &GRAPH
  EXTERNAL &CHART
  LOAD &CHART
  INTEGER &IQ
  REAL &RQ

* TIME UNIT = MILLISECONDS

SIMULATE
XPDIS FUNCTION RN1,C24
0.0,0.0/0.1,0.104/0.2,0.222/0.3,0.355/0.4,0.509/0.5,0.69/
0.6,.915/0.7,1.2/.75,1.38/
0.8,1.6/0.84,1.83/0.88,2.12/0.9,2.3/
0.92,2.52/0.94,2.81/0.95,2.99/
0.96,3.2/0.97,3.5/0.98,3.9/0.99,4.6/
0.995,5.3/0.998,6.2/0.999,7/0.9997,8
IO2S STORAGE 10
RTIME TABLE M1,800,800,10

GENERATE 2000,FNSXPDIS
QUEUE SYSTEM
  BCALL &GRAPH(C1,'SYST','EM' ' ' ' ',100,1)
SEIZE IO1
  BCALL &GRAPH(C1,'IO1' ' ' ' ',200,1)
ADVANCE 450,FNSXPDIS
RELEASE IO1
  BCALL &GRAPH(C1,'IO1' ' ' ' ',200,-1)

SEIZE CPU
  BCALL &GRAPH(C1,'CPU' ' ' ' ',200,1)
ADVANCE 500,FNSXPDIS
RELEASE CPU
  BCALL &GRAPH(C1,'CPU' ' ' ' ',200,-1)

TRANSFER BOTH,,IOX1
ENTER IO2S
  BCALL &GRAPH(C1,'IO2S' ' ' ' ',300,1)
PRIORITY 1
SEIZE IO2
  BCALL &GRAPH(C1,'IO2' ' ' ' ',200,1)
ADVANCE 580,FNSXPDIS
RELEASE IO2
  BCALL &GRAPH(C1,'IO2' ' ' ' ',200,-1)
LEAVE IO2S
  BCALL &GRAPH(C1,'IO2S' ' ' ' ',300,-1)
TABULATE RTIME
DEPART SYSTEM
  BCALL &GRAPH(C1,'SYST','EM' ' ' ' ',100,-1)
TERMINATE 1

IOX1 DEPART SYSTEM
  BCALL &GRAPH(C1,'SYST','EM' ' ' ' ',100,-1)
TERMINATE

GENERATE 4000,FN$XPDIS
QUEUE SYSTEM
  BCALL &GRAPH(C1,'SYST','EM ',' ',' ',100,1)
SEIZE IO1
  BCALL &GRAPH(C1,'IO1 ',' ',' ',' ',200,1)
ADVANCE 450,FN$XPDIS
RELEASE IO1
  BCALL &GRAPH(C1,'IO1 ',' ',' ',' ',200,-1)

SEIZE CPU
  BCALL &GRAPH(C1,'CPU ',' ',' ',' ',200,1)
ADVANCE 500,FN$XPDIS
RELEASE CPU
  BCALL &GRAPH(C1,'CPU ',' ',' ',' ',200,-1)

TRANSFER BOTH,,IOX2
ENTER IO2S
  BCALL &GRAPH(C1,'IO2S',' ',' ',' ',300,1)
PRIORITY 1
SEIZE IO2
  BCALL &GRAPH(C1,'IO2 ',' ',' ',' ',200,1)
ADVANCE 580,FN$XPDIS
RELEASE IO2
  BCALL &GRAPH(C1,'IO2 ',' ',' ',' ',200,-1)
LEAVE IO2S
  BCALL &GRAPH(C1,'IO2S',' ',' ',' ',300,-1)
TABULATE RTIME
DEPART SYSTEM
  BCALL &GRAPH(C1,'SYST','EM ',' ',' ',100,-1)
TERMINATE 1

IOX2 DEPART SYSTEM
  BCALL &GRAPH(C1,'SYST','EM ',' ',' ',100,-1)
TERMINATE

GENERATE 1250,FN$XPDIS
QUEUE SYSTEM
  BCALL &GRAPH(C1,'SYST','EM ',' ',' ',100,1)
SEIZE IO1
  BCALL &GRAPH(C1,'IO1 ',' ',' ',' ',200,1)
ADVANCE 450,FN$XPDIS
RELEASE IO1
  BCALL &GRAPH(C1,'IO1 ',' ',' ',' ',200,-1)

SEIZE CPU
  BCALL &GRAPH(C1,'CPU ',' ',' ',' ',200,1)
ADVANCE 500,FN$XPDIS
RELEASE CPU
  BCALL &GRAPH(C1,'CPU ',' ',' ',' ',200,-1)

TRANSFER BOTH,,IOX3
ENTER IO2S
  BCALL &GRAPH(C1,'IO2S',' ',' ',' ',300,1)
PRIORITY 1

Appendix B. AN EXAMPLE PROBLEM.  66
SEIZE IO2
  BCALL &GRAPH(C1,'IO2',' ', ' ', ',200,1)
ADVANCE 580,FNSXPDIS
RELEASE IO2
  BCALL &GRAPH(C1,'IO2',' ', ' ', ',200,-1)
LEAVE IO2S
  BCALL &GRAPH(C1,'IO2S',' ', ' ', ',300,-1)
TABULATE RTIME
DEPART SYSTEM
  BCALL &GRAPH(C1,'SYST','EM ', ' ', ',100,-1)
TERMINATE 1

IOX3 DEPART SYSTEM
  BCALL &GRAPH(C1,'SYST','EM ', ' ', ',100,-1)
TERMINATE
START 75
  LET &RQ = QA$SYSTEM
CALL &CHART(101,'SYST','EM ', ' ', ',&RQ,100)
  LET &RQ = QT$SYSTEM
CALL &CHART(102,'SYST','EM ', ' ', ',&RQ,100)
  LET &RQ = QX$SYSTEM
CALL &CHART(103,'SYST','EM ', ' ', ',&RQ,100)
  LET &IQ = QC$SYSTEM
  LET &RQ = &IQ
CALL &CHART(104,'SYST','EM ', ' ', ',&RQ,100)
  LET &IQ = QM$SYSTEM
  LET &RQ = &IQ
CALL &CHART(105,'SYST','EM ', ' ', ',&RQ,100)
  LET &RQ = FR$IO1
CALL &CHART(201,'IO1',' ', ' ', ',&RQ,200)
  LET &RQ = FT$IO1
CALL &CHART(202,'IO1',' ', ' ', ',&RQ,200)
  LET &RQ = FR$CPU
CALL &CHART(201,'CPU',' ', ' ', ',&RQ,200)
  LET &RQ = FT$CPU
CALL &CHART(202,'CPU',' ', ' ', ',&RQ,200)
  LET &RQ = FR$IO2
CALL &CHART(201,'IO2',' ', ' ', ',&RQ,200)
  LET &RQ = FT$IO2
CALL &CHART(202,'IO2',' ', ' ', ',&RQ,200)
  LET &RQ = SA$IO2S
CALL &CHART(301,'IO2S',' ', ' ', ',&RQ,300)
  LET &RQ = SR$IO2S
CALL &CHART(302,'IO2S',' ', ' ', ',&RQ,300)
  LET &RQ = ST$IO2S
CALL &CHART(303,'IO2S',' ', ' ', ',&RQ,300)
  LET &IQ = SC$IO2S
  LET &RQ = &IQ
CALL &CHART(304,'IO2S',' ', ' ', ',&RQ,300)
  LET &IQ = SM$IO2S
  LET &RQ = &IQ
CALL &CHART(305,'IO2S',' ', ' ', ',&RQ,300)
END
The execution of GMOD creates the data files shown in the following pages.
THE EVENT DATA FILE

989 SYSTEM 100 1
989 IO1 200 1
1043 SYSTEM 100 1
1167 IO1 200 -1
1167 CPU 200 1
1167 IO1 200 1
1357 CPU 200 -1
1357 IO2S 300 1
1357 IO2 200 1
1410 IO1 200 -1
1410 CPU 200 1
1536 CPU 200 -1
1536 IO2S 300 1
1721 SYSTEM 100 1
1721 IO1 200 1
1753 IO1 200 -1
1753 CPU 200 1
1865 CPU 200 -1
1865 IO2S 300 1
2455 SYSTEM 100 1
2455 IO1 200 1
2521 IO1 200 -1
2521 CPU 200 1
2948 SYSTEM 100 1
2948 IO1 200 1
3078 SYSTEM 100 1
3149 CPU 200 -1
3149 IO2S 300 1
3305 IO1 200 -1
3305 CPU 200 1
3305 IO1 200 1
3347 IO2 200 -1
3347 IO2S 300 -1
3347 SYSTEM 100 -1
3347 IO2 200 1
3444 IO1 200 -1
3539 CPU 200 -1
3539 IO2S 300 1
3539 CPU 200 1
3680 CPU 200 -1
3680 IO2S 300 1
3897 IO2 200 -1
3897 IO2S 300 -1
3897 SYSTEM 100 -1
3897 IO2 200 1
4986 IO2 200 -1
4986 IO2S 300 -1
4986 SYSTEM 100 -1
4986 IO2 200 1
5090 SYSTEM 100 1
5090 IO1 200 1
5256 IO1 200 -1
5256 CPU 200 1
5286 SYSTEM 100 1

Appendix B. AN EXAMPLE PROBLEM. 69
Appcndix B. AN EXAMPLE PROBLEM.
THE STATISTICS DATA FILE

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>SYSTEM</td>
<td>5. 100</td>
</tr>
<tr>
<td>102</td>
<td>SYSTEM</td>
<td>3769. 100</td>
</tr>
<tr>
<td>103</td>
<td>SYSTEM</td>
<td>3769. 100</td>
</tr>
<tr>
<td>104</td>
<td>SYSTEM</td>
<td>77. 100</td>
</tr>
<tr>
<td>105</td>
<td>SYSTEM</td>
<td>13. 100</td>
</tr>
<tr>
<td>201</td>
<td>IO1</td>
<td>592. 200</td>
</tr>
<tr>
<td>202</td>
<td>IO1</td>
<td>410. 200</td>
</tr>
<tr>
<td>201</td>
<td>CPU</td>
<td>726. 200</td>
</tr>
<tr>
<td>202</td>
<td>CPU</td>
<td>503. 200</td>
</tr>
<tr>
<td>201</td>
<td>IO2</td>
<td>722. 200</td>
</tr>
<tr>
<td>202</td>
<td>IO2</td>
<td>514. 200</td>
</tr>
<tr>
<td>301</td>
<td>IO2S</td>
<td>2. 300</td>
</tr>
<tr>
<td>302</td>
<td>IO2S</td>
<td>176. 300</td>
</tr>
<tr>
<td>303</td>
<td>IO2S</td>
<td>1235. 300</td>
</tr>
<tr>
<td>304</td>
<td>IO2S</td>
<td>76. 300</td>
</tr>
<tr>
<td>305</td>
<td>IO2S</td>
<td>6. 300</td>
</tr>
</tbody>
</table>

These data files form the input to the graphics program, DISP. The following pages show the results of the simulation run in graphics form.
Figure 15. Utilization Histogram (hardcopy).

Appendix B. AN EXAMPLE PROBLEM.
Figure 16. Plot for Queue Traffic (hardcopy).

Appendix B. AN EXAMPLE PROBLEM.
Figure 17. Queue Traffic Animation (hardcopy).

Appendix B. AN EXAMPLE PROBLEM.
Figure 18. Facility Status Animation (hardcopy).
Appendix C. PROGRAM LISTINGS.

The following pages list the programs used in the execution of the system.
THE EXEC FILE

&TRACE OFF
CLEAR
GLOBAL TXTLIB VFORTLIB CMSLIB VPIUTIL
GLOBAL LOADLIB VFLODLIB
FILEDEF 45 DISK INPUT DATA A1
FILEDEF 57 DISK GMOD GPSS A
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
SYSTEM EXECUTION COMMENCING
LOAD CONVT
START
GLOBAL TXTLIB FORTXLIB CMSLIB
FORTHX GRAPH
CLEAR
GLOBAL TXTLIB FORTXLIB CMSLIB
FORTHX CHART
CLEAR
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
&TYPE
GPSSH GMOD
GLOBAL TXTLIB VPIUTIL VFORTLIB CMSLIB GKSLIB
GLOBAL LOADLIB VFLODLIB
LOAD UNIT
START
ERASE LOAD MAP A5

Appendix C. PROGRAM LISTINGS.
PROGRAM CONVT

PROGRAM CONVT

C**************************************************************************
C CONVT IS USED TO READ AN EXECUTABLE GPSS PROGRAM AND MAKE IT GRAPHICS
C COMPATIBLE BY INSERTING THE CALL STATEMENTS.
C**************************************************************************

C**************************************************************************
C THE GPSS OBJECT PROGRAM IS STORED IN FILE 45. READ EACH CHARACTER
C BY CHARACTER AND STORE INTO ARRAY R(80).
C**************************************************************************

C**************************************************************************
C DECLARATIONS
C**************************************************************************

CHARACTER R(80)*1, U(80)*1
CHARACTER XQ(80)*1
CHARACTER QID*30, QXD*16
CHARACTER X1(80)*1
CHARACTER QX(10)*3, FX(3)*3, S(6)*3
CHARACTER *12 T
CHARACTER QUN(100)*16
CHARACTER FCN(100)*16
CHARACTER STN(100)*16

INTEGER QT2

COMMON QCN, QUN, QDN, FCN, QT2, FT2, QT, XQ, QID, QXD, X1
COMMON ST2, U, STN

C**************************************************************************
C INITIALIZATIONS
C**************************************************************************

FT2 = 0
QT2 = 0
ST2 = 0
ET = 0
ST = 0
QPL = 0
FFL = 0
SFL = 0
LOFLAG = 1
DO 27 IH = 1, 80
   QUN(IH) = ''
   FCN(IH) = ''
   STN(IH) = ''
27 CONTINUE

C**************************************************************************
C START READING A GPSS PROGRAM FROM A FILE
C**************************************************************************

2 READ (45, 4, END = 3) (R(I), I = 1, 80)
4 FORMAT (80(A1))

C**************************************************************************
C INITIALIZE ALL COUNTERS AND FLAGS
C**************************************************************************

C FLAGQ: FLAG INDICATING A LETTER 'Q' HAS BEEN READ
C FLAGD: FLAG INDICATING A LETTER 'D' HAS BEEN READ
C FLAS: FLAG INDICATING A LETTER 'S' HAS BEEN READ
C FLAGR: FLAG INDICATING A LETTER 'R' HAS BEEN READ
C FLAGE: FLAG INDICATING A LETTER 'E' HAS BEEN READ
C FLAGL: FLAG INDICATING A LETTER 'L' HAS BEEN READ

Appendix C. PROGRAM LISTINGS.
FLAG = 0
COUNT = 0
FLAGQ = 0
FLAGD = 0
FLAGS = 0
FLAGER = 0
FLAGR = 0
FLAGE = 0
FLAGD = 0
FIRST = 0
QDN = 0
QCN = 1

DO 28 IQ = 1,80
   IF (R(I).EQ.""),OR.(IQ.EQ.6),OR.(IQ.EQ.7),OR.
   *IQ.EQ.8),OR.(IQ.EQ.13),OR.(IQ.EQ.14),OR.(IQ.EQ.15),OR.(IQ.EQ.20).
   *IQ.EQ.21),OR.
   *IQ.EQ.22),OR.(IQ.EQ.27) GOTO 28
28 CONTINUE

DO 29 IQ = 1,80
   X(I) = ""  
29 CONTINUE

THE FIRST CHARACTER IS A "", PRINT ALL WHAT IS READ IN.

IF (R(I).EQ.""),GOTO 23

IF THE FIRST ONE IS A BLANK, SET 'FLAG' ON AND INITIALIZE.

IF (R(I).EQ."") THEN
   FLAG = 0
   QCN = 0
   DONT = 0
   SCNT = 0
   RQNT = 0
   ECNT = 0
   LQNT = 0
   Q = 0
   D = 0
   S = 0
   O = 0
   E = 0
   L = 0
   U1 = 0
   E1 = 0
   U2 = 0
   E2 = 0
   D = 0
   DP = 0
   DA = 0
   DR = 0
   DT = 0

Appendix C. PROGRAM LISTINGS.
THE FOLLOWING ARE THE BRANCHES TO DETERMINE THE KEYWORDS.

THE FOLLOWING SECTION IS TO DETERMINE THE OCCURRENCE OF THE KEYWORD 'QUEUE'

15 QCNT = QCNT + 1
IF(QCNT.EQ.1) THEN
IF(R(I).EQ.'U') U1 = 1
ENDIF
IF(QCNT.EQ.2) THEN
IF(R(I).EQ.'E') E1 = 1
ENDIF
IF(QCNT.EQ.3) THEN
IF(R(I).EQ.'U') U2 = 1
ENDIF
IF(QCNT.EQ.4) THEN
IF(R(I).EQ.'E') E2 = 1
ENDIF
IF(QCNT.EQ.5) THEN
Q = 0
ENDIF
U1 = 0
E1 = 0
U2 = 0
E2 = 0
ENDIF
GOTO 21

THE FOLLOWING SECTION IS TO DETERMINE THE OCCURRENCE OF
THE KEYWORD "DEPART"

16 DCNT = DCNT + 1
IF(DCNT.EQ.1) THEN
IF(R(I).EQ.'E') DE = 1
ENDIF
IF(DCNT.EQ.2) THEN
IF(R(I).EQ.'P') DP = 1
ENDIF
IF(DCNT.EQ.3) THEN
IF(R(I).EQ.'A') DA = 1
ENDIF
IF(DCNT.EQ.4) THEN
IF(R(I).EQ.'R') DR = 1
ENDIF
IF(DCNT.EQ.5) THEN
IF(R(I).EQ.'T') DT = 1
ENDIF
IF(DCNT.EQ.6) THEN
D = 0
DE = 0
DP = 0
DA = 0
DR = 0
DT = 0
ENDIF
GOTO 21

THE FOLLOWING SECTION IS TO DETERMINE THE OCCURRENCE OF
THE KEYWORD "SEIZE"

17 SCNT = SCNT + 1
IF(SCNT.EQ.1) THEN
IF(R(I).EQ.'E') SE1 = 1
ENDIF
IF(SCNT.EQ.2) THEN
IF(R(I).EQ.'T') SI = 1
ENDIF
IF(SCNT.EQ.3) THEN
IF(R(I).EQ.'Z') SZ = 1
ENDIF
IF(SCNT.EQ.4) THEN
IF(R(I).EQ.'E') SE2 = 1
ENDIF
IF(SCNT.EQ.5) THEN
S = 0
SE1 = 0
SI = 0
SZ = 0
SE2 = 0
ENDIF
GOTO 21

THE FOLLOWING SECTION IS TO DETERMINE THE OCCURRENCE OF
THE KEYWORD "RELEASE"

18 RCNT = RCNT + 1
IF(RCNT.EQ.1) THEN
IF(R(I).EQ.'E') REI = 1
ENDIF

Appendix C. PROGRAM LISTINGS.
IF (RCNT.EQ.2) THEN
IF (R(I).EQ.'L') RL = 1
ENDIF
IF (RCNT.EQ.3) THEN
IF (R(I).EQ.'E') RE2 = 1
ENDIF
IF (RCNT.EQ.4) THEN
IF (R(I).EQ.'A') RA = 1
ENDIF
IF (RCNT.EQ.5) THEN
IF (R(I).EQ.'S') RS = 1
ENDIF
IF (RCNT.EQ.6) THEN
IF (R(I).EQ.'E') RE3 = 1
ENDIF
IF (RCNT.EQ.7) THEN
O = 0
RE1 = 0
RL = 0
RE2 = 0
RA = 0
RS = 0
RE3 = 0
ENDIF
GOTO 21

C THE FOLLOWING SECTION IS TO DETERMINE THE OCCURRENCE OF
C THE KEYWORD 'ENTER'
C

19 ECNT = ECNT + 1
IF (ECNT.EQ.1) THEN
IF (R(I).EQ.'N') EN = 1
ENDIF
IF (ECNT.EQ.2) THEN
IF (R(I).EQ.'T') ET = 1
ENDIF
IF (ECNT.EQ.3) THEN
IF (R(I).EQ.'E') EE = 1
ENDIF
IF (ECNT.EQ.4) THEN
IF (R(I).EQ.'R') ER = 1
ENDIF
IF (ECNT.EQ.9) THEN
E = 0
EN = 0
ET = 0
EE = 0
ER = 0
ENDIF
GOTO 21

C THE FOLLOWING SECTION IS TO DETERMINE THE OCCURRENCE OF
C THE KEYWORD 'LEAVE'
C

20 LCNT = LCNT + 1
IF (LCNT.EQ.1) THEN
IF (R(I).EQ.'E') LE1 = 1
ENDIF
IF (LCNT.EQ.2) THEN
IF (R(I).EQ.'A') LA = 1
ENDIF
IF (LCNT.EQ.3) THEN
IF (R(I).EQ.'V') LV = 1
ENDIF
IF (LCNT.EQ.4) THEN
IF (R(I).EQ.'E') LE2 = 1
ENDIF
IF (LCNT.EQ.5) THEN
L = 0
LE1 = 0
LA = 0
LV = 0

Appendix C. PROGRAM LISTINGS.
IF ((E.EQ.1).AND.(EN.EQ.1).AND.(ET.EQ.1).AND.(EE.EQ.1)
AND.(ER.EQ.1)) FLAGE = 1
C
BCALL NAME(...,...) FOR KEYWORD 'LEAVE'
C

IF ((L.EQ.1).AND.(LE1.EQ.1).AND.(LA.EQ.1).AND.(LV.EQ.1)
AND.(LE2.EQ.1)) FLAGL = 1
C
AFTER READING THE KEYWORD CALL THE ROUTINE TO DETERMINE THE
ASSOCIATED NAME OF THE QUEUE/FACILITY/STORAGE.
C
C

22 IF ((FLAGQ.EQ.1).AND.(COUNT.EQ.2)) CALL NAME(R(I))
IF ((FLAGD.EQ.1).AND.(COUNT.EQ.2)) CALL NAME(R(I))

Appendix C. PROGRAM LISTINGS.
THE FOLLOWING SECTION WRITES GMOD FILE 55

IF (FLAGS.EQ.1).AND.(COUNT.EQ.2) CALL NAME(R(I))
IF (FLAGR.EQ.1).AND.(COUNT.EQ.2) CALL NAME(R(I))
IF (FLAGE.EQ.1).AND.(COUNT.EQ.2) CALL NAME(R(I))
IF (FLAGL.EQ.1).AND.(COUNT.EQ.2) CALL NAME(R(I))

CONTINUE

THE FOLLOWING SECTION WRITES GMOD FILE 55

C...........................................................
23 WRITE (57,6) (R(I),I = 1,80)
6 FORMAT (8X(A(I)))

C........................................................................
C
C INSERT THE DECLARATION STATEMENTS IN THE GPSS PROGRAM.
C........................................................................

C

IF (LOFLG.EQ.1) THEN
WRITE(57,24)
24 FORMAT (7X,'OPERCOL 60')
WRITE(57,13)
WRITE(57,14)
WRITE(57,15)
WRITE(57,25)
25 FORMAT (7X,'INTEGER &RQ')
26 FORMAT (7X,'REAL &RQ')
LOFLG = 0
ENDIF

IF (FLAGQ.EQ.1) THEN
QFL = 1
CALL WNAME
CALL QUE
WRITE (57,7)QID
7 FORMAT (7X,'BCALL&GRAPH(C1,'A27',1<X),1)')
ENDIF

IF (FLAGD.EQ.1) THEN
CALL WNAME
WRITE (57,8)QID
8 FORMAT (7X,'BCALL&GRAPH(C1,'A27',2<X),1)')
ENDIF

IF (FLAGR.EQ.1) THEN
CALL WNAME
WRITE (57,10)QID
10 FORMAT (7X,'BCALL&GRAPH(C1,'A27',2<X),1)')
ENDIF

IF (FLAGE.EQ.1) THEN
SFL = 1
CALL WNAME
CALL STOR
WRITE (57,11)QID
11 FORMAT (7X,'BCALL&GRAPH(C1,'A27',3<X),1)')
ENDIF

Appendix C. PROGRAM LISTINGS.
GOTO 2

C...........................................................
C COLLECT ALL THE STATISTICS SUPPLIED BY GPSS FOR ALL THE
C QUEUES IN THE GPSS PROGRAM.
C...........................................................

3 BACKSPACE (UNIT = 57)
REWRITE (UNIT = 65)
33 READ (65,34,END = 36) (U(I),J = 1,79)
34 FORMAT (IX,79(A1))
QCN = 1
DO 37 I = 1,79
CALL NAME (U(I))
37 CONTINUE
CALL QNAME
GOTO 33
36 REWRITE (UNIT = 65)
REWRITE (UNIT = 75)
Q(1) = 'QAS'
Q(2) = 'QTS'
Q(3) = 'QXS'
Q(4) = 'QCS'
Q(5) = 'QMS'
U(1) = 101
U(2) = 102
U(3) = 103
U(4) = 104
U(5) = 105
IF(QFL.EQ.1) THEN
30 READ (65,31,END = 45) QXD
31 FORMAT(1X,A16)
READ (75,38,END = 45) QID
38 FORMAT(1X,A27)
DO 32 I = 1,5
CALL OTRAN(QS(I),QXD)
WRITE(76,'(I3,A27)') QID
32 CONTINUE
GOTO 30
ENDIF
C...........................................................
C COLLECT ALL THE STATISTICS SUPPLIED BY GPSS FOR ALL THE
C FACILITIES IN THE GPSS PROGRAM.
C...........................................................

45 REWRITE (UNIT = 66)
51 READ (66,34,END = 50) (U(I),I = 1,79)
52 FORMAT (IX,79(A1))
QCN = 1
DO 56 I = 1,79
CALL NAME (U(I))
56 CONTINUE
WRITE(76,'(A1)') QID
GOTO 51
50 REWRITE (UNIT = 66)
REWRITE (UNIT = 76)
F(1) = 'FRS'
F(2) = 'FTS'
U(1) = 201
U(2) = 202
IF(FFL.EQ.1) THEN
54 READ (66,31,END = 58) QXD
READ (76,38,END = 58) QID
DO 55 I = 1,2
CALL OTRAN(FS(I),QXD)
WRITE(76,'(I3,A27)') QID
55 CONTINUE
GOTO 54
ENDIF
C...........................................................
C

Appendix C. PROGRAM LISTINGS.
SUBROUTINE QTRAN(X,QXD)
C
THE FUNCTION OF THIS SUBROUTINE IS TO INSERT STATEMENTS
to collect integer values.
C
CHARACTER QXD*16
CHARACTER X*3
WRITE(57,1) X ,QXD
1 FORMAT(7X,'LET JLR= ',A3,A16) RETURN
END

SUBROUTINE IR(X,QXD)
C
THE FUNCTION OF THIS SUBROUTINE IS TO INSERT STATEMENTS
to collect real values.
C
CHARACTER QXD*16
CHARACTER X*3
WRITE(57,1) X ,QXD
1 FORMAT(7X,'LET &IQ= ',A3,A16) WRITE(57,2)
END
SUBROUTINE NAME(ZY)

THE FUNCTION OF THIS SUBROUTINE IS TO CONVERT THE FOUR PART NAME INTO THE ORIGINAL WHOLE NAME.

2 FORMAT(7X,'LET&RQ = &IQ')
RETURN
END

SUBROUTINE WNAME

THIS SUBROUTINE IS USED TO WRITE THE FOUR PART NAME INTO FILE 38 AND THIS IS READ AS THE VARIABLE QID.

SUBROUTINE QNAME

Appendix C. PROGRAM LISTINGS.
SUBROUTINE QUE

ELIMINATE THE OCCURRENCE OF DUPLICATE QUEUE NAMES AND WRITE THE MODIFIED LIST INTO FILE 65.

INTEGER QT2
COMMON QCN, QUN, QDN, FCN, QT2, QT, XQ, QID, QXD, X1, STN
COMMON ST2
WRITE (65, *) (XQ(J), J = 1, 27)
RETURN
END

SUBROUTINE FACTY

ELIMINATE THE OCCURRENCE OF DUPLICATE FACILITY NAMES AND WRITE THE MODIFIED LIST INTO FILE 66.

INTEGER QT2
COMMON QCN, QUN, QDN, FCN, QT2, QT, XQ, QID, QXD, X1, STN
COMMON ST2
WRITE (66, *) (XQ(J), J = 1, 16)
READ (66, 15, END = 2) QXD
IF (QT2.EQ.0) THEN
QT2 = QT2 + 1
QUN(QT2) = QXD
WRITE (65, *) QUN(QT2)
GOTO 2
ENDIF
IF (QT2.EQ.1) THEN
IF (QXD.EQ. QUN(1)) GOTO 2
QT2 = QT2 + 1
QUN(QT2) = QXD
WRITE (65, *) QUN(QT2)
GOTO 2
ENDIF
IF (QT2.EQ.2) THEN
DO 1 I = 1, QT2
IF (QXD.EQ. QUN(I)) GOTO 2
1 CONTINUE
QT2 = QT2 + 1
QUN(QT2) = QXD
WRITE (65, *) QUN(QT2)
ENDIF
C WRITE (5, *) QT1 = QUN(1)
C WRITE (5, *) QT2 = QUN(2)
C WRITE (5, *) QT3 = QUN(3)
C WRITE (5, *) QT4 = QUN(4)
2 RETURN
END
SUBROUTINE STOR

C

C ELIMINATE THE OCCURRENCE OF DUPLICATE STORAGE NAMES AND WRITE THE MODIFIED LIST INTO FILE 67.

C

C

CHARACTER R(80)*1
CHARACTER XQ(80)*1
CHARACTER X1(80)*1
CHARACTER QID*30, QXD*16
CHARACTER QUN(100)*16
CHARACTER FCN(100)*16
CHARACTER STN(100)*16
INTEGER QT2
COMMON QCN, QUN, QDN, FCN, QT2, FT2, QT, XQ, QID, QXD, X1, STN
COMMON ST2
WRITE (60,*) (X1(J), J = 1, 16)
BACKSPACE (UNIT = 60)
READ (60,15,END = 2) QXD
15 FORMAT(1X, A16)

IF(FT2.EQ.0) THEN
FT2 = FT2 + 1
FCN(FT2) = QXD
WRITE(66, FCN(1))
GOTO 2
ENDIF

IF(FT2.EQ.1) THEN
IF(QXD.EQ.FCN(1)) GOTO 2
IF(QXD.EQ.FCN(FT2)) GOTO 2
ENDIF

IF(ST2.EQ.0) THEN
ST2 = ST2 + ISTN(ST2) = QXD
WRITE(67, STN(S))
ENDIF

IF(ST2.EQ.1) THEN
IF(QXD.EQ.STN(1)) GOTO 2
ENDIF

IF(ST2.EQ.2) THEN
DO 1 I = 1, FT2
IF(QXD.EQ.STN(I)) GOTO 2
CONTINUE
FI
FCN(FT2) = QXD
WRITE(66, FCN(FT2))
ENDIF

2 RETURN
END
IF(ST2.GE.2) THEN
  DO 1 I = 1,ST2
  IF(QXD.EQ.STN(I)) GOTO 2
  1 CONTINUE
  ST2 = ST2 + 1
  STN(ST2) = QXD
  WRITE(67,*) STN(ST2)
ENDIF
2 RETURN
END
SUBROUTINE GRAPH

SUBROUTINE GRAPH(CLOCK, I, J, K, L, M, ENT)

C******************************************************************************
C* THIS SUBROUTINE INTERFACES WITH GMOD TO RECORD THE OCCURRENCE OF AN EVENT. WHEN AN EVENT OCCURS, THE ATTRIBUTES ASSOCIATED WITH THE PARTICULAR ENTITY ARE RECORDED.
C******************************************************************************
C
C VARIABLES:

C CLOCK : SIMULATION CLOCK TIME
C I,J,K,L,M : FOUR PART NAME
C I : IDENTIFICATION TAG
C ENT : +1/-1

INTEGER*4 CLOCK
INTEGER*4 ENT

WRITE (96, 10) CLOCK, I, J, K, L, M, ENT
10 FORMAT (I4,2X,A4,2X,I4,2X,I2)

RETURN
END

Appendix C. PROGRAM LISTINGS. 91
SUBROUTINE CHART

SUBROUTINE CHART(I,A,B,C,D,E,J)

*****************************************************************************
  C THIS SUBROUTINE INTERFACES WITH GMOD TO COLLECT THE STATISTICS ASSOCIATED WITH EACH
  C QUEUE/FACILITY/STORAGE IN THE PROGRAM.
  
C VARIABLES:
C
C CLOCK  : SIMULATION CLOCK TIME
C J,K,L,M : FOUR PART NAME
C E : IDENTIFICATION TAG
C ENT    : +1/-1

*****************************************************************************

WRITE (97,1)I,A,B,C,D,E,J
1 FORMAT (I4,2X,4(A4),2X,F5.0,2X,13)
RETURN
END
SUBROUTINE MENU

   CHARACTER*3 MR
   XT11 = 2.0
   XT12 = 48.0
   XT13 = 17.0
   YT11 = 90.0
   YT12 = 60.0
   YT13 = 48.0
   YT14 = 45.0
   YT15 = 42.0
   YT16 = 39.0
   YT17 = 36.0

   CALL GCLRWK(1,1)
   CALL GCLRWK(2,1)
   CALL GCRSG(945)
   CALL GSCHXP(1.50)
   CALL GSCHH(1.8)
   CALL GSTXFP(4.2)
   CALL GSTXCI(13)
   CALL GTX(XT11,YT11,'PRESENTING GPSS/H RESULTS WITH GKS')
   CALL GSCHXP(1.3)
   CALL GSCHH(1.6)
   CALL GSTXFP(3.2)
   CALL GSTXCI(3)
   CALL GTX(XT12,YT12,'MAIN MENU')
   CALL GSCHXP(1.20)
   CALL GSCHH(1.8)
   CALL GSTXFP(1.2)
   CALL GSTXCI(7)
   CALL GTX(XT13,YT13,'ANIMATE QUEUE/STORAGE TRAFFIC')
   CALL GTX(XT13,YT14,'ANIMATE THE STATUS OF FACILITIY')
   CALL GTX(XT13,YT15,'PLOT OF QUEUE/STORAGE TRAFFIC')
   CALL GTX(XT13,YT16,'UTILIZATION HISTOGRAM (FACLTY/STRGB)')

1   WRITE(*,*),$ENTER YOUR CHOICE (1/2/3/4)$
   READ(5,*) IH
   CALL GCLRWK(1,1)
   IF(IH.GT.4) GOTO 1
   IF(IH.EQ.1) CALL ANIM
   IF(IH.EQ.2) CALL STAT
   IF(IH.EQ.3) CALL GRAFT
   IF(IH.EQ.4) CALL HIST
   WRITE(*,*),$WOULD YOU LIKE TO GO TO THE MAIN MENU? (Y/N)$
   READ(5,9) MR
   IF(MR.EQ.'Y') GOTO 3
   CALL QUIT

3 CALL GCLRWK(1,1)
C CALL GCLRWK(2,1)
C CALL GCRSG(945)
   CALL GSCHXP(1.50)
   CALL GSCHH(1.8)
   CALL GSTXFP(4.2)
   CALL GSTXCI(13)
   CALL GTX(XT11,YT11,'PRESENTING GPSS/H RESULTS WITH GKS')
   CALL GSCHXP(1.3)
   CALL GSCHH(1.6)
   CALL GSTXFP(3.2)
   CALL GSTXCI(3)
   CALL GTX(XT12,YT12,'MAIN MENU')
   CALL GSCHXP(1.20)
   CALL GSCHH(1.8)
   CALL GSTXFP(1.2)
   CALL GSTXCI(7)
   CALL GTX(XT13,YT13,'ANIMATE QUEUE/STORAGE TRAFFIC')
   CALL GTX(XT13,YT14,'ANIMATE THE STATUS OF FACILITIY')
   CALL GTX(XT13,YT15,'PLOT OF QUEUE/STORAGE TRAFFIC')
   CALL GTX(XT13,YT16,'UTILIZATION HISTOGRAM (FACLTY/STRGB)')

Appendix C. PROGRAM LISTINGS.
SUBROUTINE INIT

* THE PURPOSE OF THIS ROUTINE IS TO OPEN AND INITIALIZE GKS AND THE APPROPRIATE WORKSTATIONS.

INTEGER KTYPE, KERROR, KJUNK
INTEGER I, ID, J, KNUM
INTEGER KFLAG1 (13), KFLAG2 (13)

CHARACTER TYPE*7
CHARACTER STRING*10
COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNDM, QNI, SOFC
COMMON KTIME

OPENING THE FILE FOR READING AND WRITING.

CALL UOPENF(I, 'ERRORS DATAFILE', JERROR)
IF (JERROR .NE. 0) THEN
WRITE (6, *) 'CANNOT OPEN ERROR LOGGING FILE JERROR'
ENDIF

OPEN GKS

THIS SECTION ALLOWS THE USER TO INPUT THE WORKSTATION TYPE.

75 WRITE (6, *) 'PLEASE ENTER TERMINAL TYPE'
READ (5,100) KWKTYP
100 FORMAT (I6)

VALIDATE WORKSTATION TYPE. LOOP UNTIL VALID TYPE IS INPUT.

CALL GQEWK (0, KERROR, KNUM, KJUNK)
TYPE = 'INVALID'
DO 200 I = 1, KNUM
J = I
CALL GQEWK (J, KERROR, KJUNK, KTYPE)
IF (KWKTYP .EQ. KTYPE) TYPE = 'VALID'
200 CONTINUE
IF (TYPE .EQ. 'INVALID') GOTO 75

OPEN AND ACTIVATE WORKSTATION I, USE LUN 5, AND KWKTYP FOR
TYPE OPEN AND ACTIVATE WISS, USE WKID = 2 AND LUN 21

CALL GOPWK (1, 5, KWKTYP)
CALL GOPWK (2, 21, 100)
CALL GACWK (1)
CALL GACWK (2)

CALL THIS ROUTINE TO SET VIEWPORTS

CALL XFORM
RETURN

Appendix C. PROGRAM LISTINGS.
SUBROUTINE XFORM

C
C THIS SUBROUTINE IS USED TO SET THE VIEWPORTS
C
COMMON KERFIL, KWIFIL, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM,QNISOF
COMMON F,TIME
REAL XWMAX, XWMIN, YWMAX, YWMIN
REAL XVMAX, XVMIN, YVMAX, YVMIN
DATA XWMIN, XWMAX/0.0,1.0/0.0,1.0/
DATA YWMIN, YWMAX/0.0,1.0/0.0,1.0/
DATA XVMIN, XVMAX/0.0,1.0/0.0,1.0/
DATA YVMIN, YVMAX/0.0,1.0/0.0,1.0/
CALL GSWN (1, XWMIN, XWMAX, YWMIN, YWMAX)
CALL GSVP (1, XVMIN, XVMAX, YVMIN, YVMAX)
CALL GSELNT (1)
CALL GQOMDS(KWKIDP,KERROR,XUNIT, XMETE, YMETE, XRAS, YRAS)
IF (KERROR.NE.0) WRITE (5,*) 'INQUIRE SIZE ERROR'
IF (XUNIT.EQ.0) THEN
   XMAX = XMETER
   YMAX = YMETER
ELSE
   XMAX = XRAS
   YMAX = YRAS
ENDIF
CALL GSWKWN (KWKID,0,1,0,1,0)
RETURN
END

SUBROUTINE STAT

C
C THE FUNCTION OF THIS SUBROUTINE IS ANIMATE THE STATUS OF FACILITIES WITH RESPECT TO TIME.
C
INTEGER CLOCK, XAC,FLAG0,FLAG1,FLAG2
REAL XIM(4),YIM(4),TSX,TSY
CHARACTER*16 REPLY,NAM£
CHARACTER*5 TIME
CHARACTER*6 QSEL
CHARACTER DQ(10)*16
CHARACTER*16 ENT,ASEG,QN
COMMON KERFIL, KWIFIL, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QN, ISO
COMMON F,TIME
DATA XIM/0.0,1.0,1.0,0.0/
DATA YIM/0.0,0.0,1.0,1.0/
2 FORMAT (14,2X,16,2X,14,2X,12)
6 FORMAT(A16)
DO 22 I = 1,8
   DQ(0) = ".
22 CONTINUE
   SCON = 0
   SOF = 0
   KDTEC = 1
   F = 0
   KNDTEC = 0

Appendix C. PROGRAM LISTINGS. 95
KDEVNR = 1
KSEGTM = 100
KWID = 1
KWIFIL = 21
KCONID = 5
T = 0
QIMAX = 16
S = 0
QNI = 0

36 REWIND (UNIT = 66)
36 REWIND (UNIT = 96)
WRITE (5, *) 'THESE ARE THE FACILITIES IN THE MODEL'
WRITE (5, *)
98 READ (66, END = 10) QON
WRITE (5, *) QON
GOTO 98

10 WRITE (5, *)
WRITE (5, *) 'ENTER THE NUMBER OF FACILITIES TO BE ANIMATED'
WRITE (5, *) 'THE NUMBER SHOULD NOT BE GREATER THAN 8'
READ (5, *) NQ
REWIND (UNIT = 55)
DO 25 I = 1, NQ
  WRITE (5, *) 'ENTER FACILITY #', I
  READ (5, 6) DO (NQ)
  WRITE (5, 6) DO (NQ)
25 CONTINUE
1 REWIND (UNIT = 96)

FLAG1 = 0
FLAG2 = 0
FLAG3 = 0
FLAG4 = 0
FLAG5 = 0
FLAG6 = 0
FLAG7 = 0
FLAG8 = 0
CLX = 37.0
CLY = 80.0
TSX = 46.0
TSY = 80.0
LX = 1.0
LY = 1.0
TLX = 32.0
TLY = 85.0

CALL GSCHXP (1, 30)
CALL GSCHH (1, 7)
CALL GSTXFP (3, 2)
CALL GSTXCI (5)
CALL GTX (TLX, TLY, 'FACILITY STATUS ANIMATION')
CALL GSTXCI (2)
CALL GTX (CLX, CLY, 'CLOCK: ')
CALL GSCHXP (1, 30)
CALL GSCHH (1, 5)
CALL GSTXFP (1, 2)
CALL GSTXCI (13)
REWIND (UNIT = 55)
XN = -10.0
YN = 55.0
DO 111 I = 1, NQ
  IF (I.LEQ.5) THEN
    XN = -10.0
    YN = 5.0
  ENDIF
  XN = XN + 20.0
READ (55, 6) QON
CALL GTX (XN, YN, QON)
CONTINUE

REWIND (UNIT = 55)
DO 21 J = 1,NQ
READ(55,6) DOG)
21 CONTINUE
CALL GSTXCI (1)

7 READ (6,2,END = 3) M,NAME,IAL,XAC
IF(M.GE.9000) GOTO 3
WRITE ('TIME,'(BN,IS)') M
CALL GCRSG (500)
CALL GTX (TSX,TSY, TIME)
CALL GCLSG

IF ((IAL.EQ.200).AND.(NAME.EQ.DQ(1))) CALL AST1 (XAC)
IF ((IAL.EQ.200).AND.(NAME.EQ.DQ(2))) CALL AST2 (XAC)
IF ((IAL.EQ.200).AND.(NAME.EQ.DQ(3))) CALL AST3 (XAC)
IF ((IAL.EQ.200).AND.(NAME.EQ.DQ(4))) CALL AST4 (XAC)
IF ((IAL.EQ.200).AND.(NAME.EQ.DQ(5))) CALL AST5 (XAC)
IF ((IAL.EQ.200).AND.(NAME.EQ.DQ(6))) CALL AST6 (XAC)
IF ((IAL.EQ.200).AND.(NAME.EQ.DQ(7))) CALL AST7 (XAC)
IF ((IAL.EQ.200).AND.(NAME.EQ.DQ(8))) CALL AST8 (XAC)
C DO 56 I = 1,2
C CALL GSFAIS(I)
C CALL GSFASIC(0)
C CALL GFA(4,XIM,YIM)
C 56 CONTINUE
CALL GDSG(500)
GOTO 7

3 WRITE (5,*) 'DO YOU WANT A REPLAY ? (Y/N)' READ (5,6) REPLY
IF (REPLY.EQ.'Y') THEN
CALL GCLRWK (1,1)
ENDIF IF (XAC.EQ.1) THEN
CALL GSFAIS(3)
CALL GSFASIC(1)
ENDIF IF (XAC.EQ.-1) THEN
CALL GSFASIC(2)
ENDIF
RETURN
END

SUBROUTINE AST1(XAC)

INTEGER XAC
REAL AX(4), AY(4)
COMMON KERFIL, KWIFIL, KWKYP, KWKID, KISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNISO
COMMON F,TIME

AX(1) = 10.0
AX(2) = 20.0
AX(3) = 30.0
AX(4) = 40.0
AY(1) = 60.0
AY(2) = 60.0
AY(3) = 70.0
AY(4) = 70.0
CALL GSFAIS(3)
CALL GSFASIC(1)
IF (XAC.EQ.1) THEN
CALL GSFAIS(1)
CALL GSFASIC(3)
CALL GFA(4,AX(1),AY(1))
ENDIF IF (XAC.EQ.-1) THEN
CALL GSFASIC(2)
CALL GFA(4,AX(1),AY(1))
ENDIF

Appendix C. PROGRAM LISTINGS.
C SUBROUTINE AST2(XAC)

INTEGER XAC
REAL BX(4), BY(4)
COMMON KERFIL, KWIFIL, KWKYP, KWKID, KWISS
COMMON KDTEC, KDTEC, KDEVR, KSEGNM, QNI, SOF
COMMON F, TIME

BX(1) = 30.0
BX(2) = 40.0
BX(3) = 40.0
BX(4) = 30.0
BY(1) = 60.0
BY(2) = 60.0
BY(3) = 70.0
BY(4) = 70.0

CALL GSFAIS(3)
CALL GSFSII(2)

IF (XAC.EQ.1) THEN
  CALL GSFAIS(1)
  CALL GSFSII(3)
  CALL GFA(4,BX(1),BY(1))
ENDIF

IF (XAC.EQ.-1) THEN
  CALL GSFSII(2)
  CALL GFA(4,BX(1),BY(1))
ENDIF

RETURN
END

C SUBROUTINE AST3(XAC)

INTEGER XAC
REAL CX(4), CY(4)
COMMON KERFIL, KWIFIL, KWKYP, KWKID, KWISS
COMMON KDTEC, KDTEC, KDEVR, KSEGNM, QNI, SOF
COMMON F, TIME

CX(1) = 50.0
CX(2) = 60.0
CX(3) = 60.0
CX(4) = 50.0
CY(1) = 60.0
CY(2) = 60.0
CY(3) = 70.0
CY(4) = 70.0

CALL GSFAIS(3)
CALL GSFSII(3)

IF (XAC.EQ.1) THEN
  CALL GSFAIS(1)
  CALL GSFSII(3)
  CALL GFA(4,CX(1),CY(1))
ENDIF

IF (XAC.EQ.-1) THEN
  CALL GSFSII(2)
  CALL GFA(4,CX(1),CY(1))
ENDIF

RETURN
SUBROUTINE AS'T4(XAC)

INTEGER XAC
REAL DX(4), DY(4)
COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KNDTEC, KDEVNR, KSEGNM, QNL, SOF
COMMON F, TIME

DX(1) = 70.0
DX(2) = 80.0
DX(3) = 80.0
DX(4) = 70.0
DY(1) = 60.0
DY(2) = 60.0
DY(3) = 70.0
DY(4) = 70.0

CALL GSFAIS(3)
CALL GSFAIS(4)

IF (XAC.EQ.1) THEN
  CALL GSFAIS(1)
  CALL GSFAIC(3)
  CALL GFA(4, DX(1), DY(1))
ENDIF

IF (XAC.EQ.-1) THEN
  CALL GSFAIC(2)
  CALL GFA(4, DX(1), DY(1))
ENDIF

RETURN
END

SUBROUTINE AS'T5(XAC)

INTEGER XAC
REAL EX(4), EY(4)
COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KNDTEC, KDEVNR, KSEGNM, QNL, SOF
COMMON F, TIME

EX(1) = 10.0
EX(2) = 20.0
EX(3) = 20.0
EX(4) = 10.0
EY(1) = 20.0
EY(2) = 20.0
EY(3) = 30.0
EY(4) = 30.0

CALL GSFAIS(3)
CALL GSFAIS(5)

IF (XAC.EQ.1) THEN
  CALL GSFAIS(1)
  CALL GSFAIC(3)
  CALL GFA(4, EX(1), EY(1))
ENDIF

IF (XAC.EQ.-1) THEN
  CALL GSFAIC(2)
  CALL GFA(4, EX(1), EY(1))
ENDIF

RETURN
END
SUBROUTINE AST6(XAC)

INTEGER XAC
REAL FX(4), FY(4)
COMMON KERFIL, KWIFIL, KWKTyp, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNL, SOF
COMMON F,T,TIME

FX(1) = 30.0
FX(2) = 40.0
FX(3) = 40.0
FX(4) = 30.0
FY(1) = 20.0
FY(2) = 20.0
FY(3) = 30.0
FY(4) = 30.0

CALL GSFAIS(3)
CALL GSFAIS(6)

IF (XAC .EQ. 1) THEN
   CALL GSFAIS(1)
   CALL GSFAIC(3)
   CALL GFA(4, FX(1), FY(1))
ENDIF

RETURN
END

SUBROUTINE AST7(XAC)

INTEGER XAC
REAL GX(4), GY(4)
COMMON KERFIL, KWIFIL, KWKTyp, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNL, SOF
COMMON F,T,TIME

GX(1) = 50.0
GX(2) = 60.0
GX(3) = 60.0
GX(4) = 50.0
GY(1) = 20.0
GY(2) = 20.0
GY(3) = 30.0
GY(4) = 30.0

CALL GSFAIS(3)
CALL GSFAIS(7)

IF (XAC .EQ. 1) THEN
   CALL GSFAIS(1)
   CALL GSFAIC(3)
   CALL GFA(4, GX(1), GY(1))
ENDIF

RETURN
END

SUBROUTINE AST8(XAC)

INTEGER XAC
REAL HX(4), HY(4)
COMMON KERFIL, KWIFIL, KWKTyp, KWKID, KWISS

Appendix C. PROGRAM LISTINGS.
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNI, SOF
COMMON F, TIME

HX(1) = 70.0
HX(2) = 80.0
HX(3) = 80.0
HX(4) = 70.0
HY(1) = 20.0
HY(2) = 20.0
HY(3) = 30.0
HY(4) = 30.0

CALL GSFAIS(3)
CALL GSFAIK(3)

IF (XAC.EQ.1) THEN
   CALL GSFAIS(1)
   CALL GSFACI(3)
ENDIF

IF (XAC.EQ.-1) THEN
   CALL GSFACI(2)
   CALL GFA(4, HX(1), HY(1))
ENDIF

RETURN
END

SUBROUTINE GRAF

COMMON KERFIL, KWIFIL, KWKTYPE, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNI, SOF
COMMON F, TIME

INTEGER CLOCK, XAC
INTEGER TRT, TRN

REAL XMAR(20), YMAR(20), XI(4), YI(4), SXAR(16), SYAR(16)
REAL XM(30), YM(30), FX(40), FY(40)
REAL XM(4), YIM(4), ZX(130), ZY(130)

CHARACTER*16 REPLY
CHARACTER*5 TIME
CHARACTER*10 DQ(10)*16
CHARACTER*16 ENTASEG
CHARACTER*1 STRING
CHARACTER*16 NAME*16, GQN*16
CHARACTER WRT*8, TRX*8, GSEL*7

C::: This subroutine is used to plot the arrival & departure queues and storages with respect to time.
C* A counter 'TRN' is used for all manipulations with every transaction TRN.
C* Incremented by +1 or -1, TRN is used to calculate the Y co-ordinate while the time is used to calculate the X co-ordinate. The points are marked and finally joined by lines.

C**********************************************************************

DATA XI/10.0, 9.0, 9.0, 9.0, 9.0/ DATA YI/10.0, 9.0, 9.0, 9.0/ DATA XMAR/7.0, 10.0, 7.0, 10.0, 7.0, 10.0, 7.0, 10.0/ DATA XM/15.0, 20.0, 25.0, 26.0, 27.0, 28.0, 29.0/ APPENDIX C. PROGRAM LISTINGS.
*58.0,58.0,66.0,66.0,74.0,74.0,82.0,82.0,90.0,90.0/  
DATA SXAR/20.0,20.0,30.0,30.0,40.0,40.0,50.0,50.0,60.0,60.0,  
*70.0,70.0,80.0,80.0,90.0,90.0/  
DATA SYAR/10.0,8.0,10.0,8.0,10.0,8.0,10.0,8.0,10.0,8.0,  
*10.0,8.0,10.0,8.0,10.0,8.0/  
DATA XIM/0.0,1.0,1.0,1.0,0.0/  
DATA YIM/0.0,0.0,0.0,0.0,1.0/  
DO 15 I = 1,150  
ZX(I) = 0  
ZY(I) = 0  
15 CONTINUE

C VALUES FOR THE CO-ORDINATES AND LABELS.
C
TITX = 20.0  
TITY = 93.0  
PX1 = 33.0  
PY1 = 1.0  
PX2 = 90.0  
PY2 = 50.0  
PY3 = 53.0

10 REWIND(UNIT = 96)  
REWIND(UNIT = 55)

C GET THE INPUT (DISPLAY).  
C VAR: DQ(I) TAKES THE INPUT.
C
36 WRITE($,*) 'ENTER YOUR CHOICE (1 OR 2)'  
WRITE($,*) '1. PLOT QUEUES'  
WRITE($,*) '2. PLOT STORAGES'  
READ (5,*) IKN  
IF ((IKN.EQ.1).OR.(IKN.EQ.2)) THEN  
IF (IKN.EQ.1) THEN  
GSEL = 'QUEUE'  
GIDT = 100  
ENDIF  
IF (IKN.EQ.2) THEN  
GSEL = 'STORAGE'  
GIDT = 300  
ENDIF  
ELSE  
GOTO 36  
ENDIF  
IF (GSEL.EQ.'QUEUE') REWIND (UNIT = 65)  
IF (GSEL.EQ.'STORAGE') REWIND (UNIT = 67)  
IF (GSEL.EQ.'QUEUE') THEN  
WRITE (5,*) 'THESE ARE THE QUEUES IN THE MODEL'  
WRITE (5,*) '—-——-—-·-—-·——·'  
READ (65,6,END = 43) GQN  
WRITE (5,*) GQN  
GOTO 98  
ENDIF  
IF (GSEL.EQ.'STORAGE') THEN  
WRITE (5,*) 'THESE ARE THE STORAGES IN THE MODEL'  
WRITE (5,*) '—-····-··•-—————-———-'  
READ (67,6,END = 43) GQN  
WRITE (5,*) GQN  
GOTO 99  
ENDIF  
43 WRITE (5,*) 'ENTER THE NUMBER TO BE PLOTTED',GSEL,'  
WRITE(5,*)'THE NUMBER SHOULD NOT BE GREATER THAN 8'  
READ (5,*) NQ  
IF (NQ.GT.8) GOTO 10

Appendix C. PROGRAM LISTINGS.  
102
IF (NQ.GT.1) THEN
   DO 1 I = 1,NQ
      WRITE (5,'(5,*)') 'ENTER ',GSEL,' #',I
      READ (5) DQ(I)
      WRITE (5,*) DQ(I)
   CONTINUE
ELSE
   WRITE (5,'(5,*)') 'ENTER ',GSEL,' # 1'
   READ (5) DQ(1)
   WRITE (5,*) DQ(1)
ENDIF
CALL GSLWSC(4.1)
CALL GSPLCI(1)
17 FORMAT (1X,A16)
6 FORMAT (A16)
C
C DRAW THE X AND Y AXES AND LABEL THEM.
C
CALL GPL(2,X1(1),Y1(1))
CALL GPL(2,X1(3),Y1(3))
CALL GSPLCI(2)
DO 1 I = 3,21,2
   CALL GPL(2,XMAR(I-2),YMAR(I-2))
1 CONTINUE
DO 13 I = 3,17,2
   CALL GPL(2,SXAR(I-2),SYAR(I-2))
13 CONTINUE
CALL GSCHXP (1.50)
CALL GSCHH (2.0)
CALL GSTXFP (7.2)
CALL GSTXCI (7)
WX = 1.0
WY = 9.7
NO = 0
CALL GSCHXP (1.20)
CALL GSCHH (1.5)
CALL GSTXFP (1.2)
CALL GSTXCI (1)
DO 12 I = 1,10
   WY = WY + 8
   NO = NO + 10
   WRITE (WRT,'(BN,I3)')NO
   CALL GTX(WX,WY,WRT)
12 CONTINUE
CALL GSCHXP (1.50)
CALL GSCHH (1.5)
CALL GSTXFP (2.2)
CALL GSTXCI (13)
CALL GTX(TITX,TITY,'QUEUE/STORAGE TRAFFIC PLOT')
CALL GSCHXP (1.30)
CALL GSCHH (1.3)
CALL GSTXFP (3.2)
CALL GSTXCI (5)
CALL GTX(PX1,PY1,'TIME (SIMULATION CLOCK)')
CALL GTX(PX2,PY2,'NO. IN')
CALL GTX(PX2,PY2,'QUEUE')
C
C GET THE SIMULATION RUN TIME AND BASED ON THIS VALUE DIVIDE
C THE X-AXIS INTO EIGHT EQUAL PARTS.
C
REWIND (UNIT = 96)
2 READ (96,7,END = 3) M, NAME, IAL, XAC
   IF (M.GE.9000) GOTO 3
7 FORMAT (I4,2X,A16.2X,I4,2X,I2)
   MAX = M
   GOTO 2
122 CALL GSCHXP (1.20)
   CALL GSCHH (1.3)
   CALL GSTXFP (1.2)
   CALL GSTXCI (1)
3 MAX = M
   REWIND (UNIT = 96)
   XX = 90
   YY = 4
   IR = M
   WRITE (TRX.'(BN,I5)') IR
   CALL GSTXCI (1)
   CALL GTX(XX,YY,TRX)
   DO 8 I = 1, 8
      A = I
      XR = ((MAX) * ((8-A)/8))
      IR = XR
      XX = XX + 10
      WRITE (TRX.'(BN,I5)') IR
      CALL GTX(XX,YY,TRX)
8 CONTINUE
   REWIND (UNIT = 96)
   PX = 91.0
   PY = 45.0
   DO 14 N = 1, NQ
      REWIND (UNIT = 96)
      TRT = 0
      TRN = 0
C......................................................................
C MATCH THE NAMES KEYED WITH THEIR OCCURRENCE IN THE DATABASE
C AND GET THE ASSOCIATED VALUES.
C......................................................................
   CALL GSCHXP (1.30)
   CALL GSCHH (1.6)
   CALL GSMK(I)
   READ (55,17,END = 9) REPLY
4 READ (96,7,END = 9) M, NAME, IAL, XAC
   IF (M.GE.9250) GOTO 9
   IF ((NAME.EQ.REPLY).AND.(G1DT.EQ.IAL)) THEN
      TRT = TRT + 1
   END IF
C......................................................................
C INCREMENT THE COUNTER WITH THE OCCURRENCE OF AN EVENT.
C......................................................................
      TRN = TRN + XAC
      C = M
      D = MAX
C......................................................................
C COMPUTE THE X AND Y CO-ORDINATES.
C......................................................................
      ZX(TRT) = 10 + ((C*80)/D)
      ZY(TRT) = 10 + (TRN*(4.0/5.0))
      CALL GSPMCI(N + 1)
C......................................................................
C MARK THE POINT.
C......................................................................
      CALL GPM(I,ZX(TRT),ZY(TRT))
      IF (TRT.GT.1) THEN
      END IF
C......................................................................
Appendix C. PROGRAM LISTINGS. 104
C JOIN THE POINTS WITH A STRAIGHT LINE.
C
CALL GSPLC(N + 1)
CALL GPL(TRT,ZX(1),ZY(1))
ENDIF
ENDIF
GOTO 4
PY = PY-3
CALL GSTXCI(N + 1)

C WRITE THE NAME OF THE FACILITY/STORAGE.
C
CALL GTX(PX,PY,REPLY)
14 CONTINUE
C CALL GCLS

WRITE (S,') 'DO YOU WANT A REPLAY ? (Y/N)'
READ (5,6) REPLY
IF (REPLY.EQ.'Y') THEN
CALL GCLRWK(1,1)
C CALL GCLRWK(2,1)
GOTO 10
ENDIF
RETURN
END

C*********************************************************/
C
C SUBROUTINE HIST
C*********************************************************/

C THIS SUBROUTINE IS USED TO CREATE A HISTOGRAM TO DISPLAY FACILITY & STORAGE UTILIZATION. THE USER INPUTS THE FACILITY/STORAGE NEEDED. THESE ARE STORED IN A FILE. EACH NAME IS READ AND ITS ASSOCIATED DATA OBTAINED. THE NUMERICAL VALUES READ IN ARE USED TO DETERMINE THE Y-COORDINATE OF THE HISTOGRAM.
C
C*********************************************************/

INTEGER CLOCK, XAC
REAL XIM(4),YIM(4)
REAL XMAR(8), YMAR(8)
REAL XI(4),Y1(4),XM(30),YM(30),FX(40),FY(40)
CHARACTER*16 REPLY
CHARACTER*5 TIME
CHARACTER DQ(10)*16
CHARACTER*8 STRING,HSEL*8
CHARACTER NAME*16,HQN*16
COMMON KERFIL, KWIFIL, KWKYP, KWKID, KWISS
COMMON KDTAC, KNDTAC, KDEVNR, KSEGNM, QNILSO
COMMON F,TIME

DATA XIM/0.0,1.0,1.0,0.0/
DATA YIM/0.0,0.0,1.0,1.0/

C
C ASSIGNMENT STATEMENTS
C
DATA XI/10.0,90.0,9.90,0.90/
DATA Y1/9.90,9.90,10.0,90.0/
DATA XMAR/7.0,10.0,7.0,0.7,0,10.0,7.0,10.0/ DATA YMAR/30.0,30.0,50.0,50.0,70.0,70.0,90.0,90.0/
DATA XI0/0.0,1.0,1.0,0.0/ DATA YI0/0.0,0.0,1.0,1.0/

Appendix C. PROGRAM LISTINGS.
TR = 1.0
TY1 = 30.0
TY2 = 70.0
TY3 = 90.0
TY4 = 90.0
TITX = 30.0
TITY = 93.0

C FILE 97 IS THE FILE CREATED BY GMOD AND CONTAINS THE
C STORAGES GATHERED DURING THE SIMULATION RUN.
C FILE 55 IS THE FILE WHERE THE USER INPUTS FOR FACILITIES
C AND STORAGES ARE STORED. A MAXIMUM OF EIGHT FACILITIES AND
C STORAGES IS ALLOWED, IF MORE ANOTHER REQUEST IS MADE.

10 REWIND (UNIT = 97)
REWIND (UNIT = 55)
REWIND (UNIT = 66)

WRITE(5, *) 'ENTER YOUR CHOICE (1 OR 2)'
WRITE(5, *) 1. HISTOGRAM FOR FACILITIES
WRITE(5, *) 2. HISTOGRAM FOR STORAGES
READ (5, *) IHN

IF (IHN.EQ.1).OR.(IHN.EQ.2) THEN
IF (IHN.EQ.1) THEN
HSEL = 'FACIL'
HIDT = 201
ENDIF
IF (IHN.EQ.2) THEN
HSEL = 'STORAGE'
HIDT = 302
ENDIF
ELSE
GOTO 36
ENDIF

IF (NO.GT.8) GOTO 10
IF (HSEL.EQ.'FACILITY') REWIND (UNIT = 66)
IF (HSEL.EQ.'STORAGE') REWIND (UNIT = 67)
IF (HSEL.EQ.'FACILITY') THEN
WRITE (5, *) 'THESE ARE THE FACILITIES IN THE MODEL'
WRITE (5, '  ')
98 READ (66,6,END = 43) HQN
WRITE (5, *) HQN
GOTO 98
ENDIF
IF (HSEL.EQ.'STORAGE') THEN
WRITE (5, *) 'THESE ARE THE STORAGES IN THE MODEL'
WRITE (5, '  ')
99 READ (67,6,END = 43) HQN
WRITE (5, *) HQN
GOTO 99
ENDIF

43 WRITE(5,*)
WRITE(5,*)'ENTER THE NUMBER TO BE DISPLAYED','(HSEL,)'
WRITE(5,*)'(THE NUMBER SHOULD NOT BE GREATER THAN 8")'
READ (5,*) NQ

IF (NQ.GT.1) THEN
DO 5 I = 1,NQ
WRITE (5,*)
WRITE (5,*) 'ENTER ',HSEL,' #',I
READ (5,0) DQ(I)
WRITE (5,0) DQ(I)
5 CONTINUE
ELSE
WRITE (5,*)
WRITE (5,*) 'ENTER ',HSEL,' # 1'
READ (5,0) DQ(I)
WRITE (5,0) DQ(I)
ENDIF

Appendix C. PROGRAM LISTINGS.
C DRAW THE AXES AND WRITE THE COORDINATE VALUES.

C..............................................................................

C CALL GCRSG (300)
CALL GSLWSC(4,1)
CALL GSLPLCI(1)
CALL GPL(2,X1(1),Y1(1))
CALL GPL(2,X1(3),Y1(3))
CALL GSLPLCI(2)

DO 1 I = 3,9,2
   CALL GPL(2,XMAR(I)*2,YMAR(I)*2))
1 CONTINUE

CALL GSCHXP (1,30)
CALL GSCHH (1,4)
CALL GSTXFP (3,2)
CALL GSTXCI (5)
CALL G'TX (TR,TY1,'25
CALL G'TX (TR,TY2,'50
CALL GTX (TR,TY3,'75
CALL G'TX (TR,TY4,'100
CALL GTX (TR,TY5,'UTILIZATION,HISTOGRAM')

K = 0
TX = 0

DO 2 I = 1,NQ
   TX = TX + 10
   IF (J.EQ.2).OR.(J.EQ.3) INC = 10
   IF (J.EQ.1).OR.(J.EQ.4) INC = 0
   FX(K+J) = TX + INC
2 CONTINUE

K = K+4

DO 3 I = 1,12
   FY(XY+1) = TY
   FY(XY+2) = TY
   XY = XY+4
3 CONTINUE

IFA = 0
XY = 0
NO = 161

C..............................................................................

C READ THE NAME FROM FILE 55.
C..............................................................................

READ UNIT = 55
XN = 0
YN = 5
ARX = 0
ARY = 0

CALL GSFAS(1)
CALL GSCHXP (1,20)
CALL GSCHH (1,5)
CALL GSTXFP (1,2)
CALL GSTXCI (1,7)

DO 6 R = 1,NQ
   XX = K
   IF (MOD(XX,20).EQ.0) ARY = -3
   INC = XN+10
   XN = XN+10
   YN = YN+ARY
   READ (55,17,END = 8) FLY
6 CONTINUE

C..............................................................................

C READ THE ASSOCIATED VALUE FOR THE CURRENT FACILITY/STORAGE

Appendix C. PROGRAM LISTINGS.
C FROM FILE 97.

REWIND (UNIT = 97)

READ (97, END = 9) M, NAME, VAL, IDT
IF (VAL.GT.1000.0) VAL = 1000

DO 14 J = 1, 2
   CALL GSFAC(0)
   CALL GFA(4, XIM, YIM)
14 CONTINUE

IF ((HIDT.EQ.M).AND. (NAME.EQ.REPLY)) THEN
   C
   CALCULATE THE Y COORDINATE FROM THE VALUE READ IN.
   Y = ((VAL*80)/1000) + 10.0
   FY(3+XY) = Y
   FY(4+XY) = Y
   XY = XY + 4
   CALL GSFAC(K+1)

   C DRAW THE BAR.
   CALL GFA(4, FX(K+1FA), FY(K+IFA))
   IFA = IFA + 3
   CALL GTX(XN,YN,REPLY)
   C CALL GCLS

ENDIF

GOTO 12

ARY = 3
7 FORMAT(1X, I3, 2X, A16, 27(, F5.0, 27(, I3))
8 CONTINUE

C REQUEST FOR A REPLAY - IF YES GO THROUGH THE WHOLE PROCESS. IF NOT RETURN TO THE MAIN MENU.

11 WRITE (5,*) 'DO YOU WANT A REPLAY? (Y/N)'
   READ (5, 6) REPLY
   IF (REPLY.EQ.'Y') THEN
      CALL GCLR, W(K+1)
      CALL GCLR, W(K+2)
      GOTO 10
   ENDIF
   RETURN
END

C

SUBROUTINE ANIM

INTEGER CLOCK, XAC, FLAG0, FLAG1, FLAG2
REAL XIM(4), YIM(4)
CHARACTER*16 REPLY
CHARACTER*4 TIME
CHARACTER*8 QSEL
CHARACTER DQ(10)*16
CHARACTER*16 ENT, ASEG, QQN, NA

Appendix C. PROGRAM LISTINGS. 108
COMMON KERFIL, KWIFIL, KWKTYP, KWID, KWSS
COMMON KDTBC, KNDTEC, KDEVNR, KSEG, QNI, SOF
COMMON F, TIME

DATA XIM/0.0, 1.0, 1.0, 0.0/
DATA YIM/0.0, 0.0, 1.0, 1.0/
2 FORMAT (4, 2X, A16, 2X, I4, 2X, I2)
6 FORMAT (A16)

SCON = 0
SOF = 0
KDTBC = 1
F = 0
KNDTEC = 0
KDEVNR = 1
KSEG = 100
KWID = 1
KWSS = 2
KERFIL = 1
KWIFIL = 21
KCONID = 5
T = 0
QIMAX = 16
S = 0
QNI = 0

36 WRITE(5,*) 'ENTER YOUR CHOICE (1 OR 2)'
WRITE(5,*) '1. ANIMATE QUEUE TRAFFIC'
WRITE(5,*) '2. ANIMATE STORAGE TRAFFIC'
READ (5,*) IKQ
IF ((IKQ.EQ.1).OR.(IKQ.EQ.2)) THEN
IF (IKQ.EQ.1) THEN
QSEL = 'QUEUE'
QDT = 100
ENDIF
IF (IKQ.EQ.2) THEN
QSEL = 'STORAGE'
QDT = 300
ENDIF
ELSE
GOTO 36
ENDIF

IF (QSEL.EQ.'QUEUE') REWIND (UNIT = 65)
IF (QSEL.EQ.'QUEUE') THEN
WRITE (5,*) 'THESE ARE THE QUEUES IN THE MODEL'
READ (65, 6, END = 10) QQN
WRITE (5,*) QQN
GOTO 98
ENDIF

IF (QSEL.EQ.'STORAGE') REWIND (UNIT = 67)
IF (QSEL.EQ.'STORAGE') THEN
WRITE (5,*) 'THESE ARE THE STORAGES IN THE MODEL'
READ (67, 6, END = 10) QQN
WRITE (5,*) QQN
GOTO 99
ENDIF

10 WRITE (5,*) 'ENTER THE NUMBER TO BE ANIMATED ','(',QSEL,')'
WRITE(5,*) '*THE NUMBER SHOULD NOT BE GREATER THAN 8**'
READ (5,*) NQ
IF (NQ.GT.8) GOTO 10
IF (NQ.GT.1) THEN
DO 25 I = 1, NQ
WRITE (5,*) 'ENTER ',QSEL,' #', I
READ (5,6) DQ(I)
25 CONTINUE

Appendix C. PROGRAM LISTINGS.
WRITE (55,6) DQ(I)
CONTINUE
ELSE
WRITE (55,6) 'ENTER 'QSEL' # 1'
READ (55,6) DQ(I)
WRITE (55,6) DQ(I)
ENDIF.

REWIND (UNIT = 96)
REWIND (UNIT = 55)

FLAG1 = 0
FLAG2 = 0
FLAG3 = 0
FLAG4 = 0
FLAG5 = 0
FLAG6 = 0
FLAG7 = 0
FLAG8 = 0
CLX = 48.0
CLY = 55.0
TTX = 50.0
TTY = 3.0
TSX = 58.0
TSY = 55.0
LBX = 1.0
LBY = 1.0

CALL GSCHXP (1,30)
CALL GSCHH (1,5)
CALL GSTXP (3,2)
CALL GSTXI (5)
CALL GTX (CLX,CLY,'CLOCK: ')
CALL GSCHXP (1,50)
CALL GSCHH (1,5)
CALL GSTXP (1,2)
CALL GSTXI (5)
CALL GTX ('TTX,TTY,'QUEUE/STORAGE TRAFFIC ANIMATION')

CALL MARK (NQ)
CALL GSCHXP (1,20)
CALL GSCHH (1,5)
CALL GSTXP (1,2)
CALL GSTXI (1)

REWIND (UNIT = 55)

DO 9 K = 1,NQ
READ (55,6) DQ(K)
9 CONTINUE

READ (96,2,END = 3) CLOCK,ENT,IDT,XAC
IF (CLOCK.GE.9000) GOTO 3
IF (DQ(1).EQ.ENT) THEN
CALL ASQ1(XAC,FLAG1)
FLAG1 = 7
ENDIF
IF (ENT.EQ.DQ(2)) THEN
CALL ASQ2(XAC,FLAG2)
FLAG2 = 7
ENDIF
IF (ENT.EQ.DQ(3)) THEN
CALL ASQ3(XAC,FLAG3)
FLAG3 = 7
ENDIF
IF (ENT.EQ.DQ(4)) THEN
CALL ASQ4(XAC,FLAG4)
FLAG4 = 7
ENDIF
IF (ENT.EQ.DQ(5)) THEN
CALL ASQ5(XAC,FLAG5)
FLAG5 = 7
ENDIF
IF (ENT.EQ.DQ(6)) THEN
CALL ASQ6(XAC,FLAG6)
FLAG6 = 7
ENDIF
IF (ENT.EQ.DQ(7)) THEN
   CALL ASQ7(XAC,FLAG7)
   FLAG7 = 7
ENDIF

IF (ENT.EQ.DQ(8)) THEN
   CALL ASQ8(XAC,FLAG8)
   FLAG8 = 7
ENDIF

WRITE (TIME, 'BN,I4') CLOCK
CALL GCRSG (500)
CALL GTX (TSX,TSY, TIME)
CALL GCLSG
DO 32 IN = 1,2
   CALL GSFAS(0)
   CALL GSFACI(0)
   CALL GFA (4,XIM,YIM)
32 CONTINUE
CALL GDSG (500)
GOTO 7
3 WRITE (5,*), 'DO YOU WANT A REPLAY ? (Y/N)'
   READ (5,6) REPLY
IF (REPLY.EQ.'Y') THEN
   CALL GCLRWK (1,1)
   CALL GCLRWK (2,1)
   GOTO 36
ENDIF
RETURN
END

C SUBROUTINE LEGEND
COMMON KERFIL, KWIFIL, KWKID, KWISI
COMMON KDTEC, KNDTEC, KDEVNR, KSEGDNM, QNI, LOF
COMMON F, TIME
REAL BOX(5), YBUSY(5)
REAL XIDLE(5), YIDLE(5)
REAL XBREAK(5), YBREAK(5)
REAL BSFY(5), BSNFY(5)
CHARACTER*5 REPLY

TPOSX = 92.5
TPOSY = 20.0
ACX = 92.0
ACY = 16.7
BUSX = 92.0
BUSY = 11.75
DLX = 92.0
DLY = 14.5
BRKX = 92.0
BRKY = 9.5
STOX = 92.0
STOY = 5.75
FX = 92.0
FY = 3.15
AFX = 92.0
AFY = 0.5

DATA BOX/90.0,91.0,91.0,90.0,91.0/
DATA YIDLE/14.5,14.5,15.8,15.8,14.5/
DATA YBUSY/11.75,11.75,13.05,13.05,11.75/
DATA YBREAK/9.5,9.5,10.8,10.8,9.5/
DATA BSFY/3.15,3.15,4.45,4.45,3.15/
DATA BSNFY/0.5,0.5,1.8,1.8,0.5/
CALL GCRSG(400)
N = 0
C WRITE (A) 'DO YOU WANT TO SEE THE LEGEND ?'
C READ (5,20) REPLY
C20 FORMAT (A5)
C IF (REPLY.EQ.'Y') N = 1
   CALL GSVIS (100,1)
   CALL GTX (TPOSX,TPOSY, 'LEGEND')
CALL GTX (ACX, ACY; FACILITY :‘)
CALL GTX (BUSX, BUSY, BUSY‘)
CALL GTX (DLX, DLY; ‘IDLE‘)
CALL GTX (BRKX, BRKY; ‘BREAK‘)
CALL GTX (STOX, STOY; ‘STOR,QUE‘)
CALL GTX (FX, FY; ‘FULL‘)
CALL GTX (AFX, AFY; ‘NFULL‘)

CALL GSFAIS (1)
CALL GSFACI (3)
CALL GFA (5, BOX, YBUSY)
CALL GSFACI (7)
CALL GFA (5, BOX, YIDLE)
CALL GSFACI (3)
CALL GFA (5, BOX, YBREAK)
CALL GSFAIS (2)
CALL GSFACI (2)
CALL GFA (5, BOX, BSNFY)
CALL GSFAIS (I)
CALL GSFACI (3)
CALL GFA (5, BOX, B$FY)
CALL GSFACI (7)
CALL GFA (5, BOX, YBREAK)
CALL GSFACI (2)
CALL GSFAIS (2)
CALL GSFACI (2)
CALL GFA (5, BOX, B$FY)
CALL GSFACI (7)
CALL GFA (5, BOX, BSNFY)
CALL GCLSG
RETURN
END

SUBROUTINE FACT (XAC)

COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNI, SOF
COMMON F, TIME

INTEGER XAC
REAL XMACHI (12), YMACHI (12)

DATA XMACHI/S0.0,60.0,60.0,58.0,58.0,
+ 54.0,54.0,55.0,55.0,54.0,54.0
+ ,50.0/
DATA YMACHI/40.0,40.0,56.0,56.0,45.0,45.0,
+ 47.0,47.0,49.0,49.0,58.0
+ ,58.0/

IF (XAC.EQ.I) THEN
IF (F.EQ..) CALL GDSG (I2)
CALL GCRSG (I2)
CALL GSFAIS (I)
CALL GSFACI (7)
CALL GFA (5, XMACHI, YMACHI)
CALL GCLSG
ELSE
IF (XAC.EQ.-.1) THEN
CALL GDSG (I2)
CALL GCRSG (I2)
F = 1
CALL GSFAIS (I)
CALL GSFACI (7)
CALL GFA (5, XMACHI, YMACHI)
CALL GCLSG
ENDIF
ENDIF
RETURN
END

SUBROUTINE MARK (NQ)

CHARACTER*8 STRING
CHARACTER NAME*16

REAL SX(60), SY(60), TY(60)

COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNI, SOF
COMMON F, TIME

ARK = 0
FNX = 1.0
FNY = 58.0
DO 1 K = 1,NQ
CALL GCRSG(K)
IF (K.EQ.5) THEN
   SX(I) = 3.0 · SY(I) - 10.0
   SY(2) = 10.0
   ARK = 0
ENDIF
SX(I) = SX(I) + ARK
DO 111 I = 1,34
   X = I
IF (MOD (K,2.0) .EQ. 0) THEN
   SX(I) = SX(I-1) + 1.5
   IF (I.EQ.2) GOTO 111
   SY(I) = SY(I-1)
ELSE
   IF (I.EQ.1) GOTO 111
   SY(I) = SY(I-1) + 2.0
   SX(I) = SX(I)
ENDIF
111 CONTINUE
SY(1) = 60.0
SY(2) = 60.0
   SY(1) = 10.0
   SY(2) = 10.0
ENDIF
J = 1
CALL GSPLCI(K)
DO 2 J = 1,16
   IF ((J.EQ.2).OR.(J.EQ.6).OR.(J.EQ.11).OR.(J.EQ.16)) THEN
      CALL GSPLCI(2)
   ELSE
      CALL GSPLCI(I)
   ENDIF
   CALL GPL (2, SX(I), SY(I))
   J = J + 2
2 CONTINUE
QX = SX(I) + 1.0
QY = SY(I) + 1.0
X1 = SX(I) + 3.0
Y1 = SY(I) + 1.5
Y2 = SY(I) + 9.0
Y3 = SY(I) + 18.75
Y4 = SY(I) + 29.3
NX = 2.0
NY = 57
IF (K.EQ.5) THEN
   FNX = 1.0
   FNY = 8.0
ENDIF
FNX = FNX + ARK
READ (55,3,END=2) NAME
3 FORMAT (16)
CALL GSCHXP (1,20)
CALL GSCHH (1,3)
CALL GSTXFP (1,2)
CALL GSTXCI (1)
CALL GTX(X1,Y1,'1')
CALL GTX(X1,Y2,'5')
CALL GTX(X1,Y3,'10')
CALL GTX(X1,Y4,'15')
CALL GTX(FNX,FNY,NAME)
ARK = 20.0
CALL GCLSG
IF (K.EQ.1) CALL CQS1
IF (K.EQ.2) CALL CQS2
IF (K.EQ.3) CALL CQS3
IF (K.EQ.4) CALL CQS4
IF (K.EQ.5) CALL CQS5
IF (K.EQ.6) CALL CQS6
IF (K.EQ.7) CALL CQS7

Appendix C. PROGRAM LISTINGS.
IF (K.EQ.8) CALL CQS8
1 CONTINUE
RETURN
END

SUBROUTINE CQS1
C
COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KDTEC, KDEVNR, KSEGNNM, QNI, SOF
COMMON F, TIME
REAL SX(60), SY(60), RX(60), RY(60)
SX(1) = 1.0
SY(1) = 60.0
SY(2) = 60.0
DO 1 I = 1, 34
X = I
IF (MOD(X, 2.0) .EQ. 0) THEN
SX(I) = SX(I-1) + 2.0
IF (I.EQ.1) GOTO 1
SY(I) = SY(I-1)
ELSEIF (I.EQ.1) GOTO 1
SY(I) = SY(I-1) + 2.0
SX(I) = SX(I)
ENDIF.
1 CONTINUE
SY(1) = 60.0
SY(2) = 60.0
C = 1
DO 2 J = 1, 34
RX(J) = SX(C)
RX(J+1) = SX(C+1)
RX(J+2) = SX(C+3)
RX(J+3) = SX(C+2)
C = C+2
2 CONTINUE
RX(34) = RX(2)
I = 1
DO 3 K = 1, 16
CALL GCRSG (10 + K)
CALL GSVIS (10 + K, 0)
IF (K.EQ.16) THEN
CALL GSFAIS(2)
CALL GSFAIC(2)
ELSE
CALL GSFAIS (1)
CALL GSFAIC (3)
ENDIF.
CALL GFA (4, RX(I), SY(J))
CALL GCLS
I = I + 2
3 CONTINUE
RETURN
END

SUBROUTINE CQS2
C
COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KDTEC, KDEVNR, KSEGNNM, QNI, SOF
COMMON F, TIME
REAL SX(60), SY(60), RX(60), RY(60)
SX(1) = 21.0
SY(1) = 60.0
SY(2) = 60.0
Appendix C. PROGRAM LISTINGS.
DO 1 I = 1,34
   X = 1
   IF (MOD (X,2.0) .EQ. 0) THEN
      SX(I) = SX(I-1) + 2.0
      IF (I.EQ.2) GOTO 1
      SY(I) = SY(I-1)
   ELSE
      IF (I.EQ.1) GOTO 1
      SY(I) = SY(I-1) + 2.0
      SX(I) = SX(I)
   ENDIF
1 CONTINUE
   SY(1) = 60.0
   SY(2) = 60.0
   C = 1
2 DO 2 J = 1,33,4
   RX(J) = SX(C)
   RX(J+1) = SX(C+1)
   RX(J+2) = SX(C+3)
   RX(J+3) = SX(C+2)
   C = C + 2
2 CONTINUE
   RX(34) = RX(2)
   U = 1
3 DO 3 K = 1,16
   CALL GCRSG (28 + K)
   CALL GSVIS (28 + K,0)
   IF (K.EQ.16) THEN CALL GSFAIS(2)
      CALL GSFAIC(2)
   ELSE
      CALL GSFAIS (1)
      CALL GSFAIC (4)
   ENDIF
   CALL GFA (4,RX(U),SY(U))
   CALL GCLSG
   U = U + 2
3 CONTINUE
RETURN
END

SUBROUTINE CQS3

COMMON KERFIL, KWIFIL, KWKYP, KWKID, KWES
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNNM, QNILSOF
COMMON F,TIME
REAL SX(60), SY(60), RX(60), RY(60)

DO 1 I = 1,34
   X = 1
   IF (MOD (X,2.0) .EQ. 0) THEN
      SX(I) = SX(I-1) + 2.0
      IF (I.EQ.2) GOTO 1
      SY(I) = SY(I-1)
   ELSE
      IF (I.EQ.1) GOTO 1
      SY(I) = SY(I-1) + 2.0
      SX(I) = SX(I)
   ENDIF
1 CONTINUE
   SY(1) = 60.0
   SY(2) = 60.0
   C = 1
2 DO 2 J = 1,33,4
   RX(J) = SX(C)
2 CONTINUE
RX(J+1) = SX (C+1)
RX(J+2) = SX (C+3)
RX(J+3) = SX (C+2)
C = C+2
2 CONTINUE
RX(34) = RX(2)
J = 1

DO 3 K = 1,16
   CALL GCRSG (46 + K)
   CALL GSVIS (46 + K,0)
   IF (K.EQ.16) THEN
      CALL GSFAIS(2)
      CALL GSFAI(2)
   ELSE
      CALL GSFAIS (1)
      CALL GSFAI (5)
   ENDIF
   CALL GFA (4,RX(1),SY(1))
   CALL GCLSG
   J = J + 2
3 CONTINUE
RETURN
END

C:---------------------------------------------------------------

SUBROUTINE CQS4

COMMON KERFIL, KWIFIL, KWKYP, KWKID, KWISS
COMMON KDTEC, KDEVNR, KSEGNM, QN1, SOFC
COMMON F, TIME

REAL SX(60), SY(60), RX(60), RY(60)

SX(1) = 61.0
SY(1) = 60.0
SY(2) = 60.0

DO 1 I = 1,34
   X = 1
   IF (MOD (X, 2.0) .EQ. 0) THEN
      SX(I) = SX(I-1) + 2.0
   ELSE
      IF (I.EQ.1) GOTO 1
      SY(I) = SY(I-1) + 2.0
      SX(I) = SX(I)
   ENDIF
1 CONTINUE

SY(1) = 60.0
SY(2) = 60.0

C = 1

DO 2 J = 1,33,4
   RX(J) = SX (C)
   RX(J+1) = SX (C+1)
   RX(J+2) = SX (C+3)
   RX(J+3) = SX (C+2)
   C = C+2
2 CONTINUE

RX(34) = RX(2)
J = 1

DO 3 K = 1,16
   CALL GCRSG (64 + K)
   CALL GSVIS (64 + K,0)
   IF (K.EQ.16) THEN
      CALL GSFAIS(2)
      CALL GSFAI(2)
   ELSE
      CALL GSFAIS (1)
      CALL GSFAI (6)
   ENDIF
3 CONTINUE

Appendix C. PROGRAM LISTINGS.
CALL GFA (4,RX(I),SY(I))
CALL GCLSG
I = I + 2
3 CONTINUE
RETURN
END

SUBROUTINE CQS5

COMMON KERFII, KWIFIL, KWKTyp, KWKGID, KWSS
COMMON KDTEC, KDTEC, KDEVRK, KSEGNN, QNLSSF
COMMON F,TIME

REAL SX(60), SY(60), RX(60), RY(60)

SX(1) = 1.0
SY(1) = 10.0
SY(2) = 10.0

DO 1 I = 1,34
X = I
IF (MOD (X,2) .EQ. 0) THEN
SX(I) = SX(I-1) + 2.0
SY(I) = SY(I-1)
ELSE
IF (I .EQ. 2) GOTO 1
SY(I) = SY(I-1) + 2.0
SX(I) = SX(I-1)
ENDIF
1 CONTINUE
SY(1) = 10.0
SY(2) = 10.0
C = 1

DO 2 J = 1,33,4
RX(J) = SX (C)
RX(J+1) = SX (C+1)
RX(J+2) = SX (C+3)
RX(J+3) = SX (C+2)
C = C + 2
2 CONTINUE

RX(34) = RX(2)
I = 1

DO 3 K = 1,16
CALL GCRSG (82 + K)
CALL GSVIS (82 + K)
IF (K .EQ. 16) THEN
CALL GSFAIS(2)
CALL GSFACI(2)
ELSE
CALL GSFAIS (1)
CALL GSFACI (7)
ENDIF
CALL GFA (4,RX(I),SY(I))
CALL GCLSG
I = I + 2
3 CONTINUE
RETURN
END

SUBROUTINE CQS6

COMMON KERFII, KWIFIL, KWKTyp, KWKGID, KWSS
COMMON KDTEC, KDTEC, KDEVRK, KSEGNN, QNLSSF
COMMON F,TIME

REAL SX(60), SY(60), RX(60), RY(60)

Appendix C. PROGRAM LISTINGS.
SX(I) = 21.0
SY(I) = 10.0
SY(2) = 10.0

DO 1 I = 1, 34
   X = I
   IF (MOD (X,2.0) .EQ. 0) THEN
      SX(I) = SX(I-1) + 2.0
   IF (LEQ.X) GOTO 1
   SY(I) = SY(I-1)
   ELSE
      IF (LEQ.X) GOTO 1
      SY(I) = SY(I-1) + 2.0
      SX(I) = SX(I)
   ENDIF
1 CONTINUE

SY(I) = 10.0
SY(2) = 10.0
C = 1

DO 2 J = 1, 34
   RX(J) = SX(C)
   RX(J+1) = SX(C+1)
   RX(J+2) = SX(C+3)
   RX(J+3) = SX(C+2)
2 C = C + 2

RX(34) = RX(2)

U = 1

DO 3 K = 1, 16
   CALL GCRSG (100 + K)
   CALL GSVIS (100 + K, 0)
   IF (K.LEQ.16) THEN
      CALL GSFAIS(2)
      CALL GSFAIC(2)
   ELSE
      CALL GSFAIS(1)
      CALL GSFAIC(8)
   ENDIF
   CALL GFA (4, RX(U), SY(U))
   CALL GCLSG
   U = U + 2
3 CONTINUE

RETURN
END

SUBROUTINE CQS7

C
COMMON KERFIL, KWIFIL, KWKYP, KWKID, KWISS
COMMON KNDTEC, KDEVNR, KSEGTM, QNILSO
COMMON F, TIME
REAL SX(60), SY(60), RX(60), RY(60)

SX(1) = 41.0
SY(1) = 10.0
SY(2) = 10.0

DO 1 I = 1, 34
   X = I
   IF (MOD (X,2.0) .EQ. 0) THEN
      SX(I) = SX(I-1) + 2.0
   IF (LEQ.X) GOTO 1
   SY(I) = SY(I-1)
   ELSE
      IF (LEQ.X) GOTO 1
      SY(I) = SY(I-1) + 2.0
      SX(I) = SX(I)
   ENDIF
1 CONTINUE

SY(1) = 10.0
SY(2) = 10.0
C = 1

Appendix C. PROGRAM LISTINGS.
DO 2 J = 1,33,4
RX(J) = SX(C)
RX(J+1) = SX(C+1)
RX(J+2) = SX(C+3)
RX(J+3) = SX(C+2)
C = C+2
2 CONTINUE

RX(34) = RX(2)
IJ = 1

DO 3 K = 1,16
CALL GCRSG (117 + K)
CALL GSVIS (117 + K,0)
IF (K.EQ.16) THEN
CALL GSFAS(2)
CALL GSFACI(2)
ELSE
CALL GSFAS (1)
CALL GSFACI (9)
ENDIF
CALL GFA (4,RX(U),SY(U))
CALL GCLSG
IJ = IJ + 2
3 CONTINUE
RETURN
END

SUBROUTINE CQS8

COMMON KERFIL, KWIFIL, KWKTyp, KWKID, KWISS
COMMON KDTETC, KNDTETC, KDEVNR, KSEGNM,QNILSOF
COMMON F,TIME

REAL SX(60), SY(60), RX(60), RY(60)

SX(1) = 61.0
SY(1) = 10.0
SY(2) = 10.0

DO 1 I = 1,34
X = I
IF (MOD (X,2.0) .EQ. 0) THEN
SX(I) = SX(I-1) + 2.0
SY(I) = SY(I-1) + 2.0
ELSE
IF (I.EQ.1) GOTO 1
SY(I) = SY(I-1) + 2.0
SX(I) = SX(I)
ENDIF
1 CONTINUE

SY(1) = 10.0
SY(2) = 10.0
C = 1

DO 2 J = 1,33,4
RX(J) = SX(C)
RX(J+1) = SX(C+1)
RX(J+2) = SX(C+3)
RX(J+3) = SX(C+2)
C = C+2
2 CONTINUE

RX(34) = RX(2)
IJ = 1

DO 3 K = 1,16
CALL GCRSG (135 + K)
CALL GSVIS (135 + K,0)
IF (K.EQ.16) THEN
CALL GSFAS(2)
CALL GSFACI(2)
ELSE
CALL GSFAIS (1)
CALL GSFACL (10)
ENDIF
CALL GFA (4, RX(IJ), SY(IJ))
CALL GCLSG
IJ = IJ + 2
3 CONTINUE
RETURN
END

C:::::::::;::::::::;:::2:::::::::::::::::::::::::::;::::::2:::
C
SUBROUTINE ASQ1(INQ, FLAG1)
COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNIL, SOF
COMMON F, TIME
IF (FLAG1.EQ.0) THEN
   NIQ = 0
   NSEG = 0
ENDIF
IF (INQ.EQ.1) THEN
   NIQ = NIQ + 1
   NSEG = NSEG + 10
IF (NSEG.GT.26) THEN
   CALL EXQ1(NIQ, S)
   GOTO 1
ENDIF
ENDIF
IF (S.EQ.1) CALL GDSG(800)
CALL GSVIS (NSEG, 1)
ELSE
IF (INQ.EQ.1) THEN
   NIQ = NIQ + INQ
   NSEG = NSEG + INQ
IF (S.EQ.1) CALL GDSG(800)
IF (NSEG.EQ.26) THEN
   CALL EXQ1(NIQ, S)
   GOTO 1
ENDIF
ENDIF
ENDIF
S = 01
RETURN
END

C:::::::::;::::::::;:::2:::::::::::::::::::::::::::;::::::2:::
C
SUBROUTINE EXQ1(NSEG, S)
CHARACTER*5 QIL
DIX = 1.0
DIY = 55.0
EIX = 9.0
CALL GSHXP (1.20)
CALL GSHH (1.5)
CALL GTSFDP (1.2)
CALL GSTXCI (15)
CALL GTX(DIX, DIY, TOTAL)
IF (S.EQ.1) CALL GDSG(800)
CALL GCRSG(800)
WRITE (QIL, ' (BN, 15)') NSEG
CALL GTVX(EIX, DIY, QIL)
S = 1
CALL GCLSG
RETURN
END

C:::::::::;::::::::;:::2:::::::::::::::::::::::::::;::::::2:::
C
SUBROUTINE ASQ2(INQ, FLAG2)
COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS

Appendix C. PROGRAM LISTINGS.
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNNM,QNI,SOF
COMMON F,TIME

IF (FLAG2.EQ.0) THEN
  N30 = 0
  NSEG = 0
ENDIF
IF (INQ.EQ.1) THEN
  N30 = N3Q + 1
  NSEG = N3Q + 28
  IF (NSEG.GT.44) THEN
    CALL EXQ2(N3Q,S)
    GOTO 1
  ENDIF
ENDIF

IF (S.EQ.1) CALL GDSG(801)
CALL GSVIS (NSEG,1)
ELSE
  IF (INQ.EQ.-1) THEN
    N30 = N3Q + INQ
    NSEG = NSEGN + INQ
    IF (S.EQ.1) CALL GDSG(801)
    IF (NSEG.EQ.44) GOTO 1
    IF (NSEG.LT.44) CALL GSVIS(NSEG + 1,0)
    IF (NSEG.GT.44) THEN
      CALL EXQ2(N3Q,S)
      GOTO 1
    ENDIF
  ENDIF
ENDIF
END

SUBROUTINE EXQ2(NSEG,S)

CHARACTER*5 Q1L
DIX = 21.0
DIY = 55.0
EIX = 29.0
CALL GSCHXP(1,20)
CALL GSCHHI(1.5)
CALL GSTXFP(1.2)
CALL GSTXCI(1.5)
CALL GTX(DIX,DIY,TOTAL)
IF (S.EQ.1) CALL GDSG(801)
CALL GCRSG(801)
WRITE (QIL,'(BN,I5)') NSEG 
  CALL GTX(EIX,DIY,Q1L)
S = 1
CALL GCLSG
RETURN
END

SUBROUTINE ASQ3(INQ,FLAG3)

COMMON KERFIL, KWIFIL, KWKID, KWSS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNNM,QNI,SOF
COMMON F,TIME

IF (FLAG3.EQ.0) THEN
  N30 = 0
  NSEG = 0
ENDIF
IF (INQ.EQ.1) THEN
  N30 = N3Q + 1
  NSEG = N3Q + 46
  IF (NSEG.GT.62) THEN
    CALL EXQ3(N3Q,S)
    GOTO 1
  ENDIF
ENDIF

IF (S.EQ.1) CALL GDSG(802)
CALL GSVIS (NSEG,1)
ELSE
IF (INQ.EQ.-1) THEN
N4Q = N4Q + INQ
NSEG = NSEG + INQ
IF (S.EQ.1) CALL GDSG(803)
IF (NSEG.GT.80) GOTO 1
ELSEIF (S.EQ.1) THEN
N4Q = N4Q + INQ
NSEG = NSEG + INQ
IF (S.EQ.1) CALL GDSG(803)
IF (NSEG.GT.80) GOTO 1
ENDIF
ENDIF
S = 0
1 RETURN
END

SUBROUTINE EXQ3(NSEG,S)

CHARACTER*5 QIL
D1X = 41.0
D1Y = 55.0
E1X = 49.0
CALL GSCI-IXP (1.20)
CALL GSTXFP (1.2)
CALL GSTXCI (15)
CALL GTX(D1X,D1Y,1.0)
IF (S.EQ.1) CALL GDSG(802) 'CALL GCRSG(802)
WRITE (QIL,'(BN,I5)')NSEG
CALL GTX(E1X,D1Y,Q1L)
CALL GCLSG
RETURN

SUBROUTINE ASQ4(INQ,FLAG4)

COMMON KERFIL, KWIFIL, KWKTP, KWKID, KWISS
COMMON KNDTEC, KDEVNR, KSEGNM, QNIL, SOF
COMMON F, TIME
IF (FLAG4.EQ.0) THEN
N4Q = 0
NSEG = 0
ENDIF
IF (INQ.EQ.1) THEN
N4Q = N4Q + 1
NSEG = N4Q + 64
IF (NSEG.GT.80) THEN
CALL EXQ4(N4Q,S)
GOTO 1
ENDIF
IF (S.EQ.1) CALL GDSG(803)
CALL GSVIS (NSEG,1)
ELSEIF (INQ.EQ.-1) THEN
N4Q = N4Q + INQ
NSEG = NSEG + INQ
IF (S.EQ.1) CALL GDSG(803)
IF (NSEG.GT.80) GOTO 1
ENDIF
ENDIF
S = 0
1 RETURN
END

SUBROUTINE EXQ3(NSEG,S)
SUBROUTINE EXQ(NSEG,S)

C CHARACTER*5 QIL

DIX = 61.0
DIY = 55.0
EIX = 69.0
CALL GSCHXP(1,20)
CALL GSCHH(1.5)
CALL GSTXFP(1.2)
CALL GSTXCI(15)
CALL GTX(DIX,DIY,TOTAL)
IF (S.EQ.1) CALL GDSG(803)
CALL GCRSG(803)
WRITE (01L,'(BN,I5)') NSEG
CALL GCLSG
RETURN
END

SUBROUTINE ASQ(SIN,Q,FLAG5)

C

COMMON KERFIL, KWIFIL, KWKTyp, KWKID, KWSS,
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM,QNL,SOF
COMMON F,TIME

IF (FLAG5.EQ.0) THEN
    NSQ = 0
    NSEG = 0
ENDIF

IF (INQ.EQ.1) THEN
    NSQ = NSQ + 1
    NSEG = NSEG + 82
    IF (NSEG.GT.98) THEN
        CALL EXQ(NSEG,S)
    ENDIF
ENDIF

IF (S.EQ.1) CALL GDSG(804)
CALL GSVIS(NSEG,1)
ELSE
    IF (INQ.EQ.-1) THEN
        NSQ = NSQ + INQ
        NSEG = NSEG + INQ
        IF (S.EQ.1) CALL GDSG(804)
        IF (NSQ.GT.98) THEN
            CALL EXQ(NSEG,S)
        ENDIF
    ENDIF
ENDIF
RETURN
END

SUBROUTINE EXQ(NSEG,S)

C

CHARACTER*5 QIL

DIX = 61.0
DIY = 55.0
EIX = 69.0
CALL GSCHXP(1,20)
CALL GSCHH(1.5)
CALL GSTXFP(1.2)
CALL GSTXCI(15)
CALL GTX(DIX,DIY,TOTAL)
IF (S.EQ.1) CALL GDSG(803)
CALL GCRSG(803)
WRITE (QIL,'(BN,I5)') NSEG
CALL GTX(EIX,D1Y,QIL)
S = 1
CALL GCLSG
RETURN
END

SUBROUTINE ASQ6(INQ,FLAG6)

COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEG_NM, QNI, SOF
COMMON F, TIME

IF (FLAG6.EQ.0) THEN
   N6Q = 0
   NSEG = 0
ENDIF
IF (INQ.EQ.1) THEN
   N6Q = N6Q + 1
   NSEG = NSEG + 1
   IF (NSEG.GT.116) THEN
      CALL EXQ6(N6Q,S)
      GOTO 1
   ENDIF
IF (S.EQ.1) CALL GDSG(805)
ELSE
   IF (INQ.EQ.-1) THEN
      N6Q = N6Q + INQ
      NSEG = NSEG + INQ
      IF (S.EQ.1) CALL GDSG(805)
      IF (NSEG.GT.116) GOTO 1
      IF (NSEG.LT.116) CALL GSVIS(NSEG + 1,0)
      IF (NSEG.GT.116) THEN
         CALL EXQ6(N6Q,S)
         GOTO 1
      ENDIF
   ENDIF
ENDIF
ENDIF
S = 0
1 RETURN
END

SUBROUTINE EXQ6(NSEG,S)

CHARACTER*5 QIL

D1X = 21.0
D1Y = 5.0
EIX = 29.0
CALL GSCX(1,20)
CALL GSCXH(1,9)
CALL GSTFXP(1,2)
CALL GSTFCI(15)
CALL GTX(D1X,D1Y,TOTAL)
IF (S.EQ.1) CALL GDSG(805)
CALL GCRSG(805)
WRITE(QIL,(BN,I5)) NSEG
CALL GTX(EIX,D1Y,QIL)
S = 1
CALL GCLSG
RETURN
END

SUBROUTINE ASQ7(INQ,FLAG7)

COMMON KERFIL, KWIFIL, KWKTYP, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEG_NM, QNI, SOF
COMMON F, TIME

IF (FLAG7.EQ.0) THEN
   N7Q = 0
   NSEG = 0
ENDIF
IF (INQ.EQ.1) THEN
   N7Q = N7Q + 1
   NSEG = NSEG + 1
   IF (NSEG.GT.116) THEN
      CALL EXQ7(N7Q,S)
      GOTO 1
   ENDIF
IF (S.EQ.1) CALL GDSG(805)
ELSE
   IF (INQ.EQ.-1) THEN
      N7Q = N7Q + INQ
      NSEG = NSEG + INQ
      IF (S.EQ.1) CALL GDSG(805)
      IF (NSEG.GT.116) GOTO 1
      IF (NSEG.LT.116) CALL GSVIS(NSEG + 1,0)
      IF (NSEG.GT.116) THEN
         CALL EXQ7(N7Q,S)
         GOTO 1
      ENDIF
   ENDIF
ENDIF
ENDIF
S = 0
1 RETURN
END

Appendix C. PROGRAM LISTINGS. 124
SUBROUTINE EXQ7(NSEG,S)

C

CHARACTER*5 QIL

DIX = 41.0
D1Y = 5.0
EIX = 49.0
CALL GSCHXP (1.20)
CALL GSCHH (1.5)
CALL GSTDFF (1.2)
CALL GSTXCI (15)
CALL G1TX(D1X,D1Y,"TOTAL")
IF (S.EQ.1) CALL GDSG(806)
CALL GSCSG(806)
WRITE (QIL,'(1I8)') NSEG
CALL GCLS
RETURN

SUBROUTINE ASQ8(INQ,FLAG8)

C

COMMON KERFIL, KWIFIL, KWKTP, KWKID, KWSI
COMMON KDF, KENV, KDEVNL, KSEG, QN1, SOF
COMMON FT,TIME

IF (FLAG8.EQ.0) THEN
  N8Q = 0
  NSEG = 0
ENDIF
IF (INQ.EQ.1) THEN
  N8Q = N8Q + 1
  NSEG = N8Q + 117
IF (NSEG.GT.133) THEN
  CALL EXQ7(N8Q,S)
  GOTO 1
ENDIF
IF (S.EQ.1) CALL GDSG(806)
CALL GSVIS(NSEG,1)
ELSE
IF (INQ.EQ.-1) THEN
  N8Q = N8Q + INQ
  NSEG = NSEG + INQ
IF (S.EQ.1) CALL GDSG(807)
ELSE
  IF (INQ.EQ.-1) THEN
    N8Q = N8Q + INQ
    NSEG = NSEG + INQ
    IF (S.EQ.1) CALL GDSG(807)
    IF (NSEG.EQ.133) GOTO 1
ENDIF
ENDIF
ENDIF
S = 0
1 RETURN
END
IF (NSEG.LT.151) CALL GSVIS(NSEG + 1,0)
IF (NSEG.GT.151) THEN
   CALL EXQ8(NSEG,S)
   GOTO 1
ENDIF
ENDIF
ENDIF
S = 0
RETURN
END

SUBROUTINE EXQ8(NSEG,S)

C

CHARACTER*5 QIL

DIX = 61.0
DIY = 5.0
EIX = 69.0
CALL GSCHXP (1.20)
CALL GSCHH (1.5)
CALL GSTXFP (1.2)
CALL GSTXCI (15)
CALL GTX(DIX,DIY,TOTAL)
IF (S.EQ.1) CALL GDSG(807)
CALL GCRSG(807)
WRITE (QIL,'(B,N,I5)')NSEG
SAL; GTX(EIX,D1Y,Q1L)
CALL GCLSG
RETURN
END

SUBROUTINE ASTO(SCON)

C

INTEGER SCON
REAL SBOX(8),SBOY(8),VX(15)

COMMON KERFIL, KWIFIL, KWKTPK, KWKID, KWSIS
COMMON KDIAT, Department, KDIAT, KDEVNR, KSEGNM, QNI, SOF
COMMON F, TIME

DATA SBOX/70.0,85.0,85.0,83.0,83.0,72.0,72.0,70.0/
DATA SBOY/40.0,40.0,41.0,41.0,50.0,50.0,41.0,41.0/

IF(SCON.EQ.-1) GOTO 2
QNI = SCON + QNI
IF (QNIL.EQ.STOMAX) GOTO 4
GO TO 5
4 CALL GDSG(431)
   F = 0
   CALL GCRSG (432)
   CALL GSFAIS (1)
   CALL GSFAIS(3)
   CALL GFA (8,SBOX,SBOY)
   SOF = 1
   CALL GCLSG
   GOTO 3
5 IF (F.EQ.1) CALL GDSG(431)
   F = 0
   IF (SOF.EQ.1) THEN
      CALL GDSG (432)
      SOF = 0
   ENDIF
   CALL GCRSG(431)
   F = 1
   CALL GSFAIS (1)
   CALL GSFAIS(6)
   CALL GFA (8,SBOX,SBOY)
   CALL GCLSG
   GOTO 3
2 VY = 44.5
QNI = QNI + SCON
3 RETURN
END

Appendix C. PROGRAM LISTINGS. 126
SUBROUTINE QUIT

COMMON KERFIL, KWIFIL, KWKTyp, KWKID, KWISS
COMMON KDTEC, KNDTEC, KDEVNR, KSEGNM, QNI, SOF
COMMON F, TIME

CALL GDAWK (1)
CALL GDAWK (2)
CALL GCLWK (1)
CALL GCLWK (2)
CALL GCLKS
STOP
END
The vita has been removed from the scanned document