THE UNIVERSITY OF MICHIGAN

INDUSTRY PROGRAM OF THE COLLEGE OF ENGINEERING

GENERAL PURPOSE SIMULATION SYSTEM/360: INTRODUCTORY CONCEPTS AND CASE STUDIES

Thomas J. Schriber

October, 1968

IP-824

Preface

This material is the synopsis of a series of lectures covering General Purpose Simulation System/360 in a course on Simulation and Systems Analysis. The presentation pre-supposes that the reader is familiar with the role of simulation as a tool in systems analysis and is aware of its advantages and disadvantages relative to the analytical approaches which may sometimes be employed in its place. Some experience in computer programming and a grasp of elementary concepts in probability and statistics is also pre-supposed. Nevertheless, fundamental ideas from these latter two areas have been included when germane to the topic being considered directly: GPSS/360. Hence, it should be possible to move rapidly through many sections of the material.

The purpose of this monograph is to make it possible for the beginner to learn GPSS/360 in systematic fashion and with relatively efficient use of time. An attempt has been made to build up the capabilities of the language in a structured fashion, using case studies whenever possible to reinforce ideas and illustrate the use of language features in context. The internal logic of the simulator is also considered, and the question of model validity in light of this internal logic is discussed in the framework of several case studies. The relationship between the internal simulator logic and execution time efficiency is also considered.

A total of 17 fully-documented case studies is presented. Documentation includes statement of the problem, table of definitions, block diagrams, program listing, computer output, and time and cost information. The cases themselves serve as a convenient learning vehicle, and should provide ideas which the reader can use in constructing models appropriate to his own area of interest. The presentation in the monograph is believed to be adequately detailed and sufficiently extensive to make the reader capable of independent use of GPSS/360.

Table of Contents

List of Case Studies

List of Figures

---

*"Documentation" includes Block Diagrams, Program Listings, Output, and Time and Cost Information. Tables of Definitions are also included for all models but are not given Figure numbers.

Preliminary Edition

of

GENERAL PURPOSE SIMULATION SYSTEM/360:

INTRODUCTORY CONCEPTS AND CASE STUDIES

1. Some Preliminary Considerations

***GPSS is a software package supplying a well-defined framework which facilitates computer simulation of discrete systems. The language was developed and is maintained by IBM. Pertinent IBM literature covering the language is:

1) GPSS/360: Introductory User's Manual (H20-0204-1)

2) GPSS/360: User's Manual (H20-0326-2)

3) GPSS/360 OS Operator's Manual (H20-0311-3)

This literature will be referred to as IUM, User's Manual, and Operator's Manual, respectively.

***A GPSS model of a system may be expressed either as a block diagram or as the punchcard equivalent of a block diagram. The model builder usually begins by constructing a block diagram model of the system he intends to simulate. The punchcard version of the block diagram is then prepared and presented to the computer for implementation.

***A block diagram is a collection of characteristically-shaped figures (blocks) which are connected by directed line segments. There is a set of some 45 blocks which GPSS makes available to the user. The user selects certain of the blocks from this set according to the logical requirements of the model, arranging them so that they interact meaningfully with one another. It is this interaction which is analogous to (simulates) the interaction of elements in the real system being modeled.

***The Shell of a typical GPSS block diagram is shown in Figure 1.

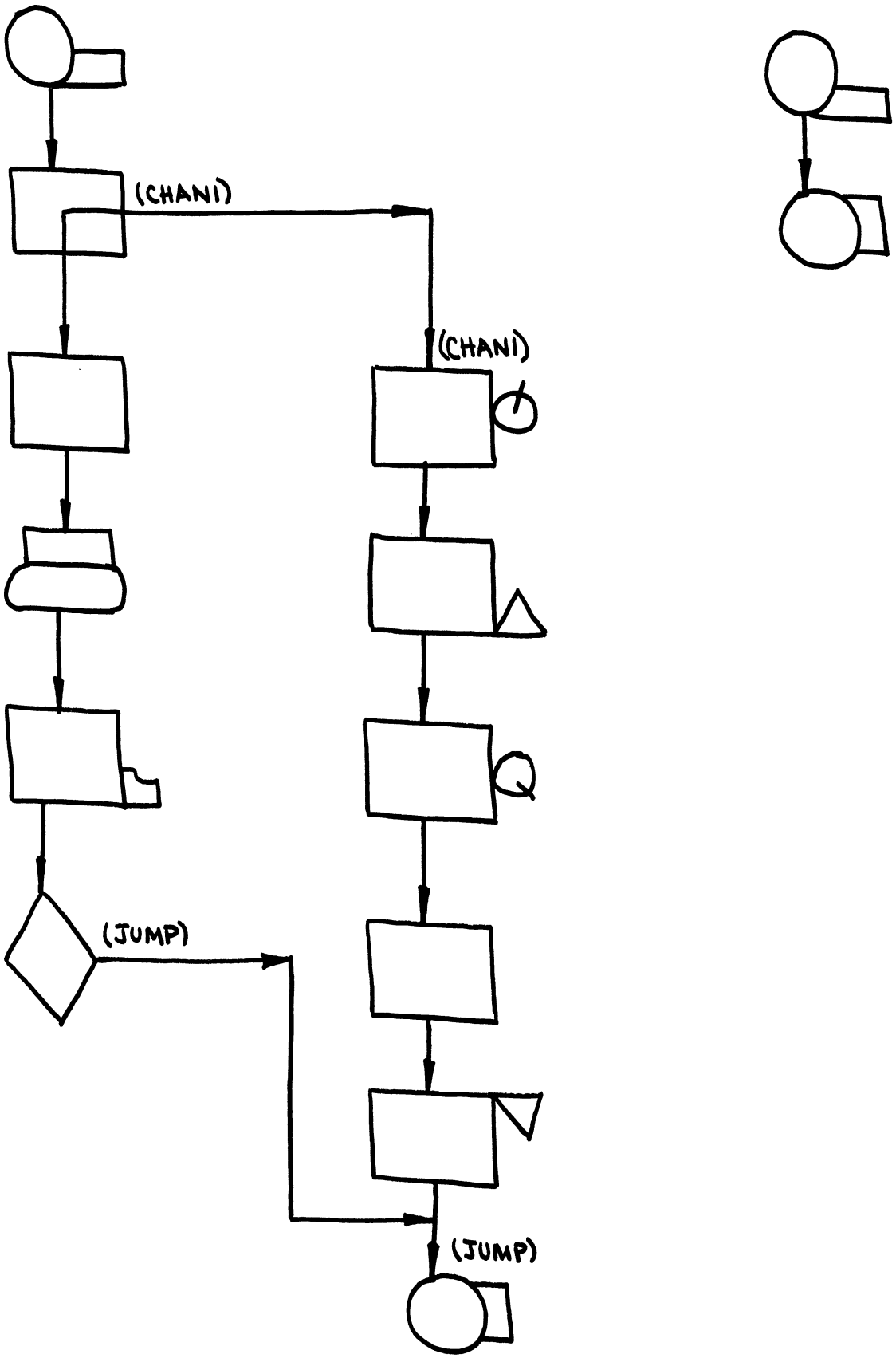


FIGURE 1: THE SHELL OF A TYPICAL GPSS BLOCK DIAGRAM

2. Transactions and Their Qualitative Features

***The directed line segments in Figure 1. suggest movement. In GPSS there are dynamic elements which flow from block to block in a block diagram. These dynamic elements are termed Transactions. The "meaning" of Transactions is determined by the model builder. This is done by establishing an analogy or correspondence between the Transactions and elements of the actual system being modeled.

***Examples of possible analogies between Transactions and elements of real systems:

System	Transaction
Supermarket	Shopper
Highway	Car
Inventory	Demand
Maintenance	Part

***Transactions can be both "created" and "destroyed" during the course of a simulation.

***As suggested by the directed line segments in Figure 1., Transactions move in general to the next sequential block in a block diagram.

***Transactions may, however, be directed to some non-sequential block.

***When a Transaction has been "set into motion", the simulator moves the Transaction forward, block-by-block, until either

a block which "holds" the Transaction is encountered, or

a block which refuses entry to the Transaction is encountered.

***Many Transactions can be "in process" at the same time (though they must, due to the inherent nature of a digital computer, be "updated" one at a time).

***As shown in Figure 1., blocks can be given symbolic names so that they can be referred to from various points in the block diagram.

***Various segments in a block diagram do not have to be interconnected.

3. The Simulation Clock

***One of the more interesting aspects of systems is their behavior as time elapses.

***GPSS maintains a simulated clock to record "what time it is", i.e. how far forward in time a system has been "moved".

***The simulated clock is automatically set to a value of zero when a simulation begins.

***The clock only registers integer values.

***The time unit is defined "implicitly" by the programmer. This simply means that all time data built into the model by the programmer must be in consistent units, such as minutes, milliseconds, days, or whatever. The time unit chosen is never "declared" to the simulator.

***Distinguish carefully between "simulated time" and "real time". Bear in mind that:

--The simulated clock "stands still" while real time elapses as the various steps required to update a model at a particular point in simulated time are carried out by the simulator.

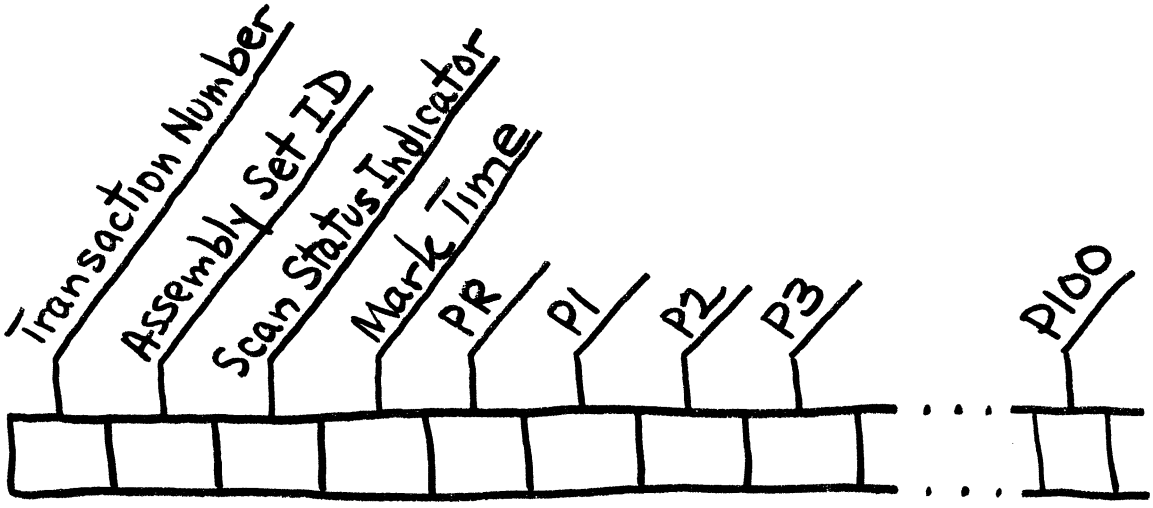
--Some systems are such that hours of real time may be required to move the system forward only minutes in simulated time (e.g. Computer Systems).

--On the other hand, observations equivalent to weeks, months, or even years of simulated time can often be made in only seconds of real time on the computer.

***GPSS is a "next event" simulator. That is, after a model has been fully updated at a given point in simulated time, the simulated clock is advanced to the nearest time at which another event is scheduled to occur.

4. Numerical Properties of Transactions

***Each Transaction in a model possesses its own set of properties, some of which are indicated schematically below:



***The schematic is a simple representation of a series of computer storage locations. Each storage location is shown as a rectangular compartment and has been tagged with a unique name, or address. This homely device will prove useful later when indirect addressing is considered.

***The properties "Transaction Number", "Assembly Set ID", "Scan Status Indicator", and "Mark Time" cannot be directly addressed by the programmer.

***The values of PR and P1, P2, P3, ... , P100 can be directly addressed by the programmer.

***The number of Transactions in a model is not a fixed quantity, but will generally vary as the simulation proceeds. The upper limit on the number of Transactions which may be active in a model depends on the amount of computer storage available (See Table 1, page 47, IUM, or Table 4, page 19, User's Manual).

Specific Comments on Numerical Properties:

***Transaction Number: GPSS labels each Transaction in a model with a number to facilitate internal bookkeeping tasks, and to enable the user to follow the movement of Transactions by use of a "trace" option.

***Assembly Set ID: GPSS views each Transaction as being a member of a particular "assembly set". The Assembly Set ID is simply a record of the assembly set to which a Transaction belongs.

***Scan Status Indicator: A Transaction is either "scan-active" or "scan-inactive" depending whether its Scan Status Indicator is Off or On respectively.

***Mark Time: Mark Time is a record of the time when

a) the Transaction was generated ("created"), or

b) the Transaction was last "marked".

The reading of the simulated clock is automatically copied into "Mark Time" when a Transaction is generated. If the value of Mark Time is to be subsequently altered as part of a programmer's logic, this alteration is carried out by causing the Transaction to pass through a particular GPSS block designed for that purpose.

***PR: PR is the mnemonic address for the priority level of a Transaction. PR may have any value from 0 to 127 (unsigned integer). Higher PR values correspond to higher priorities.

***Pj, j = 1,2,3, ... , 100: Pj is the mnemonic address for the j-th Parameter of a Transaction. Hence P3 is a reference to Parameter three, P21 is a reference to Parameter twenty one, etc. Parameter values are signed integers. A distinction is made between fullword and halfword Parameters. All Parameters associated with a particular Transaction are either of the fullword or the halfword variety. Fullword Parameters are allocated 4 bytes (32 bits) of storage. Halfword Parameters are allocated 2 bytes (16 bits) of storage. The largest and smallest numbers that can be stored in fullword and halfword Parameters are $\pm$2,147,483,647 and $\pm$32,767, respectively.

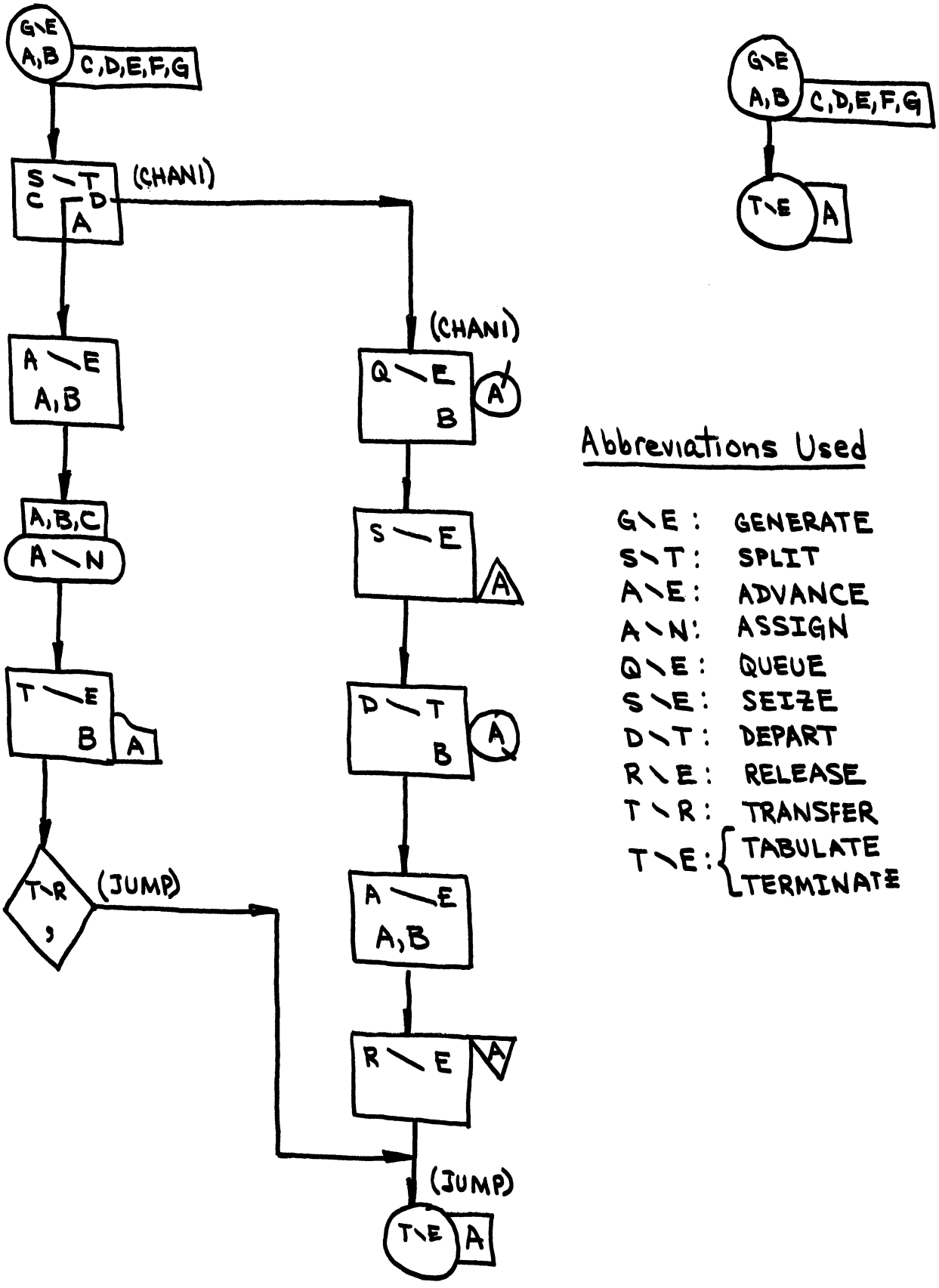


FIGURE 2: THE PARTICULAR GPSS BLOCK DIAGRAM WITH GENERAL DETAILS SHOWN

5. The Particular GPSS Block Diagram Shown with General Block Details

***Figure 2. is a repetition of Figure 1. with the general block details filled in.

***Each block carries with it three types of information:

1) Location: (optional) The "Location" of a block is the symbolic name given to the block. Blocks are usually not given symbolic names unless they must be referred to from one or more other blocks in the block diagram. Symbolic block names consist of from 3 to 5 alphanumeric characters, with the restriction that the first 3 characters must be alphabetic.

2) Operation: (not optional) The "Operation" of a block is the "verb" which is suggestive of the action the block is designed to have carried out. There is a one-to-one correspondence between the shape of a block and the action it represents.

3) Operands: (some required, some optional) A block's "Operands" provide the specific information on the basis of which the block's action occurs. The operands may be conveniently thought of as "arguments to a subroutine". The number of operands used by a block is a function of the particular block in question. No block uses more than 7 operands. Most blocks use only 1 or 2 operands. The operands are denoted in general as A,B,C,D,E,F, and G. For a particular block, some operands must always be supplied, whereas others are optional.

***As indicated in Figure 2, the convention followed in these notes will be to abbreviate certain Block information when Block Diagrams are presented. In particular, the "Operation" associated with a Block will always be represented with a diagonal line running between the first and last letters of the corresponding verb. This is done to save time and space. Redundancies are avoided because of the differences in Block shapes. Note that abbreviations cannot be used in preparing the punchcard equivalent of a Block Diagram.

***With this "preview" of the type of block detail to expect, we now take up particular blocks, one by one, and discuss the action they represent and the significance of their operands. After discussing 9 or 10 of the most basic blocks, we can begin to construct complete models of simple systems. As more and more block types are subsequently studied and mastered, the potential for modeling increasingly more sophisticated and complex systems in GPSS will become evident.

6. Creation of Transactions

***Two methods are available for creating Transactions:

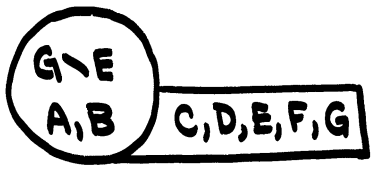
1) Creation by generation (using the GENERATE block), and

2) Creation by splitting (using the SPLIT block).

The GENERATE Block:

Purpose: Bring Transactions into existence

Shape and Operands:



Operand	Significance
A	Specifies the mean time interval between creation of successive Transactions (mean interarrival time).
B	Specifies the mean interarrival time modifier. Two options are available: Spread Modifier: used when the interarrival time is uniformly distributed over the range of times "A±B". Function Modifier: used when the distribution of interarrival times is not uniform but follows some other distribution, either theoretical or empirical
C	Optional operand; specifies an offset interval; if used, the C operand is interpreted as the time at which the first Transaction is to be created at the GENERATE block, i.e. first arrival time is deterministic; all subsequent times of creation are then stochastic (random) per the A and B operands; if not used (the "default case"), all times of Transaction creation are stochastic.

D — Optional operand; specifies the maximum number of Transactions which are to be created at the GENERATE block; known as the limit count; after the limit count is reached, that GENERATE block ceases to operate; in the default case, the GENERATE block continues to create Transactions throughout the duration of the simulation.

E — Optional operand; specifies the PR (priority level) value of all Transactions created at that GENERATE block; in the default case, a value of 0 is assigned to PR.

F — Optional operand; specifies the number of Parameters that each Transaction created at that GENERATE block is to possess. The F operand may be any number from 0 to 100. In the default case, 12 Parameters (P1 through P12) are given to each Transaction created. All Parameters are given an initial value of 0.

G — Optional operand; specifies whether the Parameters are to be fullword or halfword; if the G operand is "F", fullword Parameters are indicated. In the default case, or if the G operand is "H", Parameters are of the halfword variety.

Nature of Operation

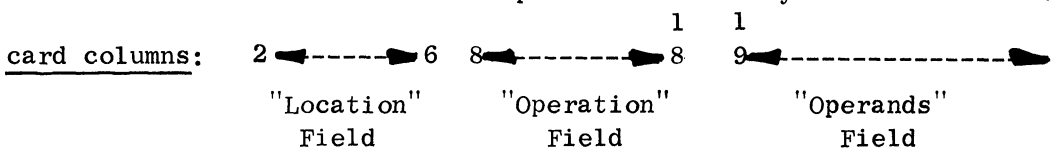
***C operand specified: the first Transaction arrives at the GENERATE block when simulated time equals the value of the C operand. The Transaction then immediately attempts to exit the GENERATE block and enter the next sequential block. The next sequential block might be of a type, however, that refuses entry under certain conditions (to be discussed later). If entry is refused, the Transaction remains in the GENERATE block until conditions change so that it can gain entry to the next sequential block. Note that the time of arrival of the next Transaction to the GENERATE block is not scheduled until its predecessor Transaction succeeds in exiting the GENERATE block. As soon as a Transaction succeeds in exiting the GENERATE block, the A and B operand values are used to schedule the time of arrival of its successor Transaction to the GENERATE block.

***C operand not specified: the operation is as described above, except that the A and B operand values are used to schedule the time of arrival of all Transactions to the GENERATE block, including the first.

***Note that there may be more than one GENERATE block in a model.

Punchcards Corresponding to Blocks in GPSS Block Diagrams:

***The three types of information carried by each block have already been described on page 7. Corresponding to these three types of information, there are three fields laid out on a punchcard to carry the information:



In review, the Location Field contains the symbolic name (if any) of the block for which this is the punchcard equivalent; the Operation Field contains the "verb" describing the action which characterizes the block; and the Operands Field contains the block operands, which must be entered in consecutive card columns and separated by commas, without any intervening blanks.

Examples for the GENERATE Block:

Example 1:

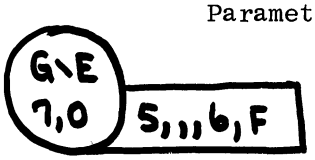
Punchcard: GENERATE 5,2

Explanation: Interarrival times are equiprobable over the range 5± 2, i.e. from 3 to 7 inclusive. Note that the simulated clock can register only integer values. Hence, in this case, a particular interarrival time could be any one of 5 values (3, 4, 5, 6, or 7).

The default case is taken for the other operands, which means:

no offset interval
no limit count
priority level of 0
twelve Parameters per Transaction
Parameters are of halfword variety

Example 2:



Punchcard: GENERATE 7,0,5,,,6,F

Explanation: The arrival of the first Transaction to the GENERATE block is to occur deterministically at time 5 (C operand). Subsequent interarrival times are equiprobable over the range 7±0, i.e. are always 7; hence the 2nd arrival occurs at time 12, the 3rd arrival at time 19, etc.

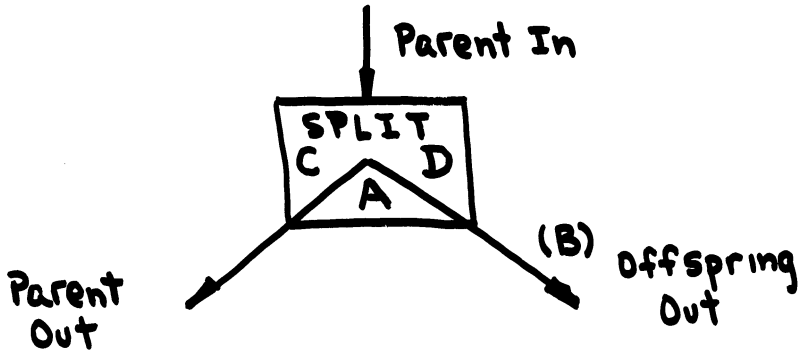
Other aspects of the operation:

no limit count
zero priority level
six Parameters per Transaction
fullword Parameters

The SPLIT Block:

Purpose: Cause "offspring" Transactions to be created by a "parent" Transaction

Shape and Operands:

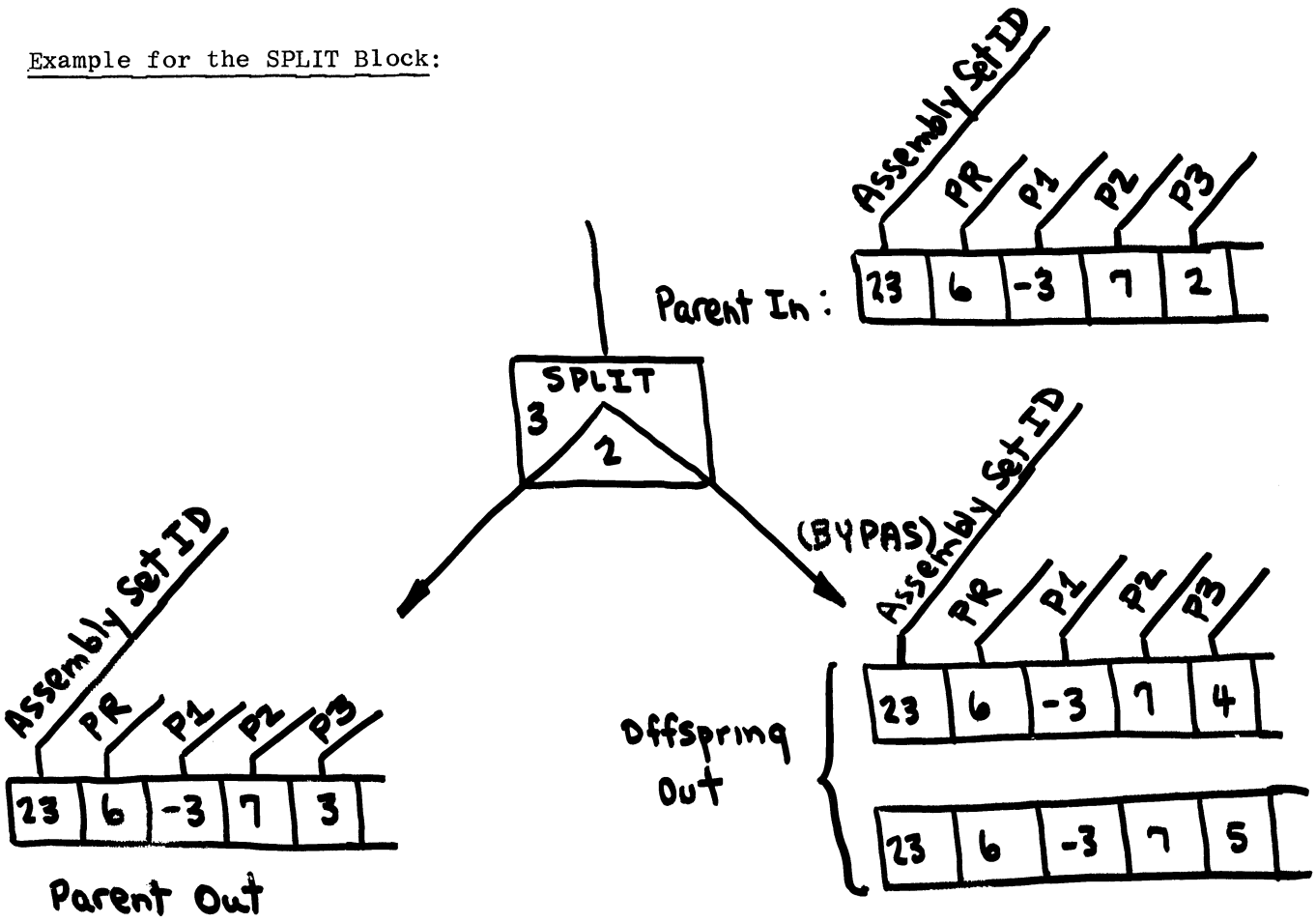


Operand	Significance
A	Specifies the number of offspring Transactions to be created.
B	Specifies the symbolic name of the non-sequential block to which the offspring are to be routed.
C	Optional operand; specifies a Parameter in which parent and offspring are to be serialized; in the default case, there is no serialization.
D	Optional operand; specifies the number of Parameters each offspring is to have; in the default case, the offspring have the same number of Parameters as the parent.

Nature of Operation

***Whenever a Transaction (known here as the "parent") moves into the SPLIT block, copies (known as the "offspring") of the Transaction are made; the parent exits the SPLIT block and moves on to the next sequential block in the model; the offspring exit the SPLIT block and are all routed to some non-sequential block in the model.

Example for the SPLIT Block:



Comments

***Offspring are identical to the parent in terms of Mark Time, priority level (PR), and Parameter values (excepting the Parameter used for serialization, if this option is exercised).

***The parent and its offspring are members of the same Assembly Set.

***Assembly Sets always contain only one member unless that member (Transaction) moves through a SPLIT block, thereby further populating the Assembly Set.

7. Destruction of Transactions

***Two methods are available for removing Transactions from a model:

1) Removal by termination (using the TERMINATE block), and

2) Removal by re-combination of assembly set members (using the ASSEMBLE block).

The TERMINATE Block:

Purpose: Destroy Transactions

Nature of Operation: All Transactions entering the TERMINATE block are destroyed without exception.

Shape and Operands:

Operand	Significance
A	Specifies the amount by which a special counter known as the termination counter is to be decremented each time a Transaction enters that TERMINATE block; the A operand may be zero.

Note: There may be more than one TERMINATE block in a model.

Termination Counter

***The termination counter is a storage location which is initialized with a value provided on a START card. As soon as the termination counter has been decremented to zero (or less), the simulation stops.

The START Card

***As suggested above, the START card provides the means used by the programmer to control the duration of a simulation run.

***The START card has no block equivalent in a GPSS block diagram.

***A START card is punched as follows, where, as before, A, B, C, and D represent operands:

	1	1
Card Columns:	8 ◄------► 8	9 ◄----------►
Contents:	START	A,B,C,D

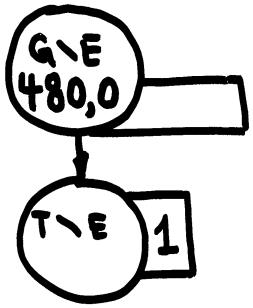
Operand	Significance
A	Specifies the value with which the termination counter is initialized at the beginning of a simulation run
B,C,D	To be explained later.

Problem:

Show how a simulation run can be terminated after the simulation has proceeded for 480 time units.

Solution:

Assume that all other TERMINATE blocks in the model have an A operand of zero. Include in the model this two-block segment:



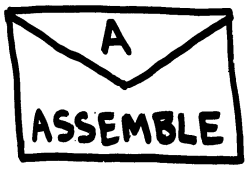
and use 1 as the A operand in the START card. Then, at time 480 a Transaction arrives at the above GENERATE block and moves to the TERMINATE block, causing the termination counter to be decremented from 1 to 0 and thereby stopping the simulation.

The ASSEMBLE Block:

Purpose: Re-combine members of the same Assembly Set

Nature of Operation: The first member of an Assembly Set to enter the ASSEMBLE block is held at the block. Other members of the Assembly Set which later arrive at the ASSEMBLE block are destroyed upon arrival. When a specified number have been destroyed, the member which was being held is permitted to exit the block and moves on to the next sequential block in the model.

Shape and Operands:



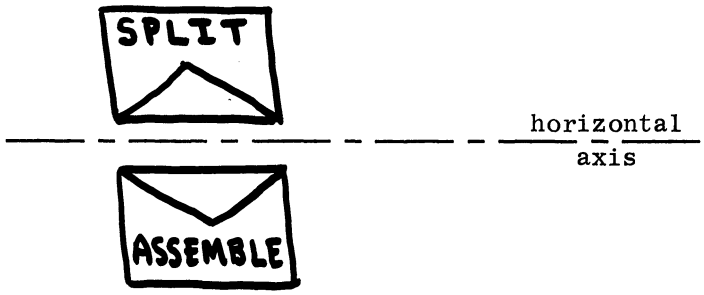
Operand	Significance
A	Specifies the assembly count, which is one greater than the number of Assembly Set Transactions which are to be destroyed before the first arrival is permitted to exit the block.

Note: An ASSEMBLE block may simultaneously assemble members of more than one Assembly Set. Also, there is no limit to the number of assemble activities that can be completed relative to a given Assembly Set.

8. Complementary Block Pairs in GPSS

***Many of the GPSS blocks exist in "complementary pairs". One member of the pair then has the effect of "undoing" or "reversing" the effect of the other pair member. For example, the ASSEMBLE block causes an activity which is the reverse of the SPLIT block's activity. In many cases, the "characteristic shape" of complementary pairs are horizontal-axis mirror images of each other. This makes the shapes easier to remember. Hence compare:

Complementary Block Pair:



9. Facilities in GPSS

***A Facility is the GPSS term for the capability of "providing service" to a Transaction. A Facility can accommodate only one Transaction at a time.

***The significance of a "Facility providing service to a Transaction" depends entirely on the analogy the programmer draws between the abstract concepts "Facility" and "Transaction" and the real world system being modeled.

***For example, a Transaction might be a shopper in a supermarket, and a Facility might be an express checkout counter. Then the event "a Transaction seizes a Facility" is equivalent to the event "the express checkout girl begins to check out a shopper". To say that the facility has been seized by a Transaction is to say that the checkout girl is in the process of checking out a shopper. And the event "a Transaction releases a Facility" is equivalent to the event "the girl finishes checking the shopper through".

***Many different Facilities may exist in a GPSS model. Facilities are differentiated from each other by number, i.e. we may speak of Facility 1, Facility 5, Facility 27, etc. The total number of different Facilities that may be used in one model depends on the amount of computer storage available (See Table 1, page 47, IUM, or Table 4, page 19, User's Manual). Even with a given amount of computer storage, the number of Facilities allowed may be increased (or decreased) at the expense of (or to the benefit of) one or more other GPSS entities. Such a redistribution of entities over available computer storage is effected with a REALLOCATE card (See page 21 et. seq., Operator's Manual).

***The status of a Facility is of interest, i.e. is a Facility available or unavailable at a given time? Facility status is automatically maintained by GPSS upon the modeler's request. The request is made by appropriate use of a complementary block pair relating to Facilities.

10. Complementary Block Pair for Facilities

Purpose: Change the recorded status of Facilities, i.e. change status from "available" to "not available" or vice versa.

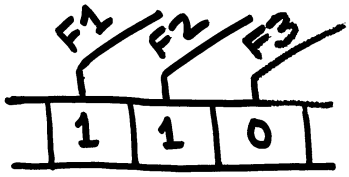
Cause certain statistics relative to Facilities to be automatically maintained.

Prevent Transactions from engaging a Facility which is not currently available (only one block of the complementary pair has this effect).

Shape and Operands:

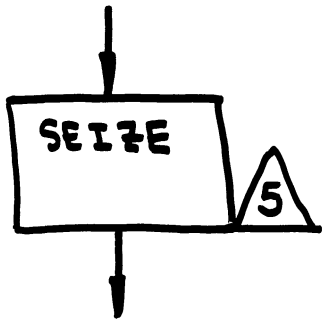
Operand	Significance
A	Specifies the number of the Facility which is to be SEIZEd or RELEASEd.

Nature of Operation: For each Facility used in a model, there is a storage location known as Fj, where j is the number of the particular Facility in question. The value of Fj is 0 if Facility number j is "available" (not currently engaged by a Transaction) and is 1 if the Facility is "not currently available". Hence, if this is the picture of storage at a particular time:



then Facility 1 and Facility 2 are not available, whereas Facility 3 is available.

When a Transaction moves down this path:



the following occurs:

if F5 = 0:

The Transaction enters the SEIZE block;
F5 is given a value of 1;
The Transaction exits the SEIZE block and moves to the next sequential block in the model.

if F5 = 1:

The Transaction is not permitted to enter the SEIZE block:
The Transaction waits at the Block "ahead of" the SEIZE Block until Facility 5 becomes available, at which time it competes with other waiting Transactions (if any) for the privilege of being the next Transaction to engage the Facility.

Note

***Relative to the competition suggested above, the question naturally arises: which Transaction will win out? If two or more Transactions are waiting at the SEIZE block, they gain entry to the block on a first-come, first-served basis if they have the same priority level (PR) value. In case of competition among Transactions with different priority levels, higher-prioritied Transactions are given preference. We are touching here upon the idea of "queue discipline": the rule by which a "server" determines whom to serve next, given that two or more people are waiting for his services. Unless the programmer provides alternative logic, queue discipline in GPSS is first-come, first-served (FIFO: first in, first out) within priority class. A Transaction with, say, PR = 7 will SEIZE a Facility before a Transaction with PR = 6, even though the latter Transaction has been waiting at the SEIZE block a longer time. (The programmer can model arbitrarily-defined queue disciplines in GPSS by using the LINK/UNLINK complementary block pair, to be discussed later.)

***Another question that occurs is: suppose the programmer wants to route the Transaction elsewhere in the model if it finds that a Facility it intended to engage is unavailable . . . can this be done? The answer is yes. The capability for carrying this out will be discussed later.

***In discussing the nature of operation of the GENERATE block on page 9, it was mentioned that "the next sequential block might be of a type, however, that refuses entry under certain conditions". The SEIZE block is the first such block that we have seen.

Similarly, when a Transaction comes down this path:

this occurs:

The Transaction enters the RELEASE block;
The value of F8 is changed from 1 to 0;
The Transaction exits the RELEASE block and moves on to the next sequential block in the model.

***Only a Transaction which has SEIZEd a facility can RELEASE that Facility. Running Error Stop No. 415 (see Appendix 4, page 69, IUM, or Appendix A, page 211, User's Manual) occurs if this rule is violated.

***The RELEASE block cannot refuse entry to a Transaction.

11. Facility Statistics Automatically Maintained by GPSS

***In addition to the storage location Fj, for each Facility in a model there are another three storage locations, known as FRj, FCj, and FTj, respectively, which contain these values:

FRj: the fractional time utilization of Facility j, in parts per thousand

FCj: the total number of Transactions which have SEIZEd Facility j during the course of the simulation

FTj: the average time each seizing Transaction held Facility j (integerized)

***These statistics are of course subject to change as the simulation proceeds.

***FRj, FCj, and FTj, together with Fj, constitute what is known as the "Standard Numerical Attributes" of Facilities (see page 29, IUM, or Table 2, page 10, User's Manual).

***These Standard Numerical Attributes can be referenced dynamically as a simulation proceeds. Hence their values can be used to control the course of the simulation.

12. Service Time in GPSS; the ADVANCE Block

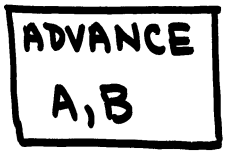
***Consider this block diagram segment:

Does it make sense? Probably not, because it is difficult to imagine the significance of having a Transaction SEIZE a Facility, and then immediately RELEASE it. Presumeably the Transaction should "hold" the Facility for a certain number of time units, while the Facility "provides service" to the Transaction. Holding time is specified by using the ADVANCE block.

The ADVANCE Block:

Purpose: Hold a Transaction for one or more time units at a particular point in a model before permitting it to move to the next sequential block.

Shape and Operands:



Operand	Significance
A	Specifies the mean holding time.
B	Specifies the mean holding time modifies; two options are available: Spread Modifier: used when the holding time is uniformly distributed over the range of times "A±B". Function Modifier: used when the distribution of holding times is not uniform but follows some other distribution, either theoretical or empirical.

Note

***The A and B operands in an ADVANCE block have the same significance as in a GENERATE block (page 5 of these notes).

***When a Transaction enters the ADVANCE block, the A and B operands are used to compute a holding time for that Transaction. After the holding time has elapsed, the Transaction exits the ADVANCE block and moves to the next sequential block.

***Any number of Transactions can be held simultaneously at an ADVANCE block. Each Transaction being held has its own particular holding time associated with it, and is not influenced by the presence of other Transactions in the ADVANCE block.

***An ADVANCE block can be used anyplace in a model, not just after a SEIZE block.

13. Queues (Waiting Lines) in GPSS

***A circumstance has already been seen in which it might be necessary for one or more Transactions to wait at some point in a model before they can gain entry to a block. If a Facility is not available, a Transaction attempting to SEIZE it must wait until it becomes available (unless programmer-supplied logic provides another alternative).

***It might be of interest to the modeler to gather statistics on Transactions waiting to gain entry to a block, such as:

Current length of the waiting line;
Average length to date of the waiting line;
Maximum length to date of the waiting line, and so on.

***A request for automatic maintenance of such statistics is made by using a complementary pair of waiting line blocks. One member of the pair records entry to the Queue; the other member records departure from the Queue.

***There can be many different Queues in a model. Like Facilities, Queues are numbered so that they can be distinguished from one another. The total number of different Queues that may be used in one model depends on the amount of computer storage availabe

14. Queue Statistics Automatically Maintained by GPSS

***The full set of statistics automatically maintained for each Queue in a model is:

Qj: the contents of Queue number j, i.e. the total number of units currently recorded as being in Queue number j.

QAj: The average contents to date of Queue number j (integerized).

QMj: The maximum contents to date of Queue number j.

QCj: The total number of entries to date in Queue j.

QZj: The total number of "zero entries" to date in Queue j. A "zero entry" is an entry which does not have to wait at all (zero residence time in the waiting line).

QTj: The average time spent in the waiting line (with zero residence time cases included in the average; integerized).

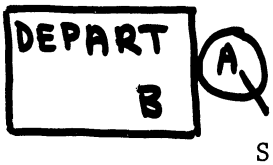
QXj: The average time spent in the waiting line (with zero residence time cases excluded from the average; integerized).

***The properties listed above constitute the Standard Numerical Attributes of Queues (see page 29, IUM, or Table 2, page 10, User's Manual). Analogous to the Standard Numerical Attributes for Facilities, the Queue SNA's can be referenced during the simulation and can be used to dynamically control the course of the simulation.

15. Complementary Block Pair for Waiting Lines

Purpose: Cause updating of the statistics being maintained for waiting lines.

Shape and Operands:



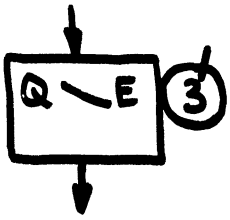
Operand	Significance
A	Specifies the number of the waiting line which the Transaction is to QUEUE up in or is to DEPART
B	Specifies the number of units by which the length of the waiting line is to be incremented (when QUEUEing up) or decremented (when DEPARTing). If the B operand is not specified, a value of 1 is automatically assumed.

***It is not necessary to use the QUEUE and DEPART blocks at every point in a model where waiting might occur. Waiting will occur, if necessary, whether the QUEUE/DEPART blocks are used or not. The blocks are used simply to indicate to GPSS that statistics are desired relative to the waiting process. If the modeler has no intention of using these statistics, it is wasteful of computer time to have them gathered and maintained. Hence, in many instances, it is preferable not to use the QUEUE/DEPART blocks.

***A waiting line for which statistics are requested is termed an explicit waiting line.

***The contents of a waiting line are not necessarily the same as the number of Transactions in the waiting line. For example, if a Transaction enters a QUEUE block with A operand of 7 and B operand of 3, then Q7 is increased by 3.

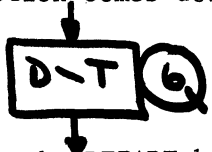
***When a Transaction comes down this path:



this occurs:

The Transaction enters the QUEUE block;
The full set of QUEUE Standard Numerical Attributes is updated for Queue 3;
The Transaction exits the QUEUE block (if possible) and moves into the next sequential block.

***Similarly, when a Transaction comes down this path:



this occurs:

The Transaction enters the DEPART block;
The full set of Queue Standard Numerical Attributes is updated for Queue 6;
The Transaction exits the DEPART block (if possible) and moves into the next sequential block.

16. Barber Shop: First Model

Statement of the Problem

Customers arrive at a one-chair barber shop with an average time between arrivals of 18 minutes. The interarrival time actually varies between 12 and 24 minutes, with equal likelihood. The barber gives one haircut every 15 minutes, on the average. The time per haircut ranges between 12 and 18 minutes, however, with equal likelihood. Model the barber shop in GPSS, then simulate the operation of the shop through eight hours of operation. Determine the percent idle time of the barber, the average waiting time of the customers, and the maximum number of customers who at one and the same time were waiting for the barber.

Approach Taken in Building the Model

This model is easily constructed as a single sequence of blocks. The order in which the blocks appear corresponds to the chronology which would be expected in the real system: customers arrive; if necessary, they wait their turn for the barber; then they "engage" the barber, get their hair cut, "release" the barber, and leave the shop. A separate two block segment is used as a timer to turn off the simulation.

Table of Definitions

Time Unit: 1 Minute

GPSS Entity	Interpretation
Transaction	A Customer in the Main Segment A "Timer" in the Timer Segment
Facility 3	The Barber
Queue 2	The Waiting Line

Block Diagram

Figure 3-A shows the GPSS block diagram model of the barber shop.

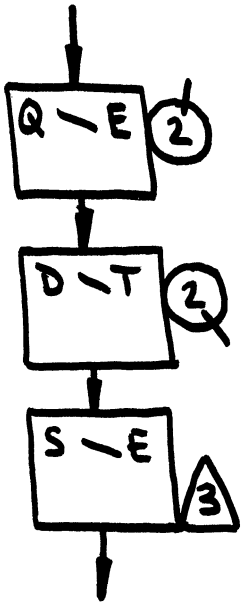
Discussion of Block Diagram

Figure 3-A shows the SEIZE block sandwiched between the QUEUE/DEPART block pair. This configuration may lead to two questions:

1) Suppose that a customer enters the shop and finds that the barber can immediately cut his hair. Since it is not necessary to join a waiting line, why does such a customer move through the QUEUE block before SEIZE-ing the barber?

2) Why does the customer DEPART the waiting line <u>after</u> SEIZE-ing the barber, rather than just before SEIZE-ing the barber?

The second question is answered by pointing out that the SEIZE block, occupying the position it does, <u>prevents</u> the customer from DEPARTing the Queue until it is logically possible for him to do so. Remember that the DEPART block cannot refuse entry to a Transaction. Hence the block sequence:



would not operate properly. Transactions would DEPART the Queue immediately after queueing up in it, then they would try to SIEZE the Facility. If the Facility were not available, waiting would be necessary, but the Queue statistics would not take this waiting into account.

In answer to the first question, realize that if the Facility is not engaged when a Transaction comes from the GENERATE block, then the Transaction will QUEUE up, SEIZE the Facility, and DEPART the QUEUE all during one and the same instant of simulated clock time. Residence time of such a Transaction in the Queue is then zero. A Transaction which QUEUEs and DEPARTs in this fashion is known in GPSS as a "Zero Entry".

Program Listing

The punchcard equivalent of the Figure 3-A model is shown in Figure 3-B, complete with the control cards required to run the model on The University of Michigan's 360/67 under MTS (Michigan Terminal System).

Program Output; Time and Cost Data

Figure 3-C shows a selected portion of the program output and includes information as to the duration and cost of the run.

Discussion of Output

At first glance, it is not clear how the requested statistics are made available to the modeler. It turns out that GPSS will automatically print out, at the conclusion of a simulation run, a full set of statistics for each Facility and Queue used in a model (full sets of statistics for other GPSS entities not yet discussed will be printed out as well). A typical set of such statistics can be seen by turning to page 83, IUM.

Note: Under appropriate circumstances, the modeler may want to suppress this "automatic" printout of the full set of model statistics at the conclusion of a simulation. (The programmer may, for example, use the a special GPSS block, the PRINT block, to have selected statistics of interest printed out during the course of a simulation, or just prior to termination of the simulation.) The automatic printout can be suppressed by specifying NP as the B operand in the START card.

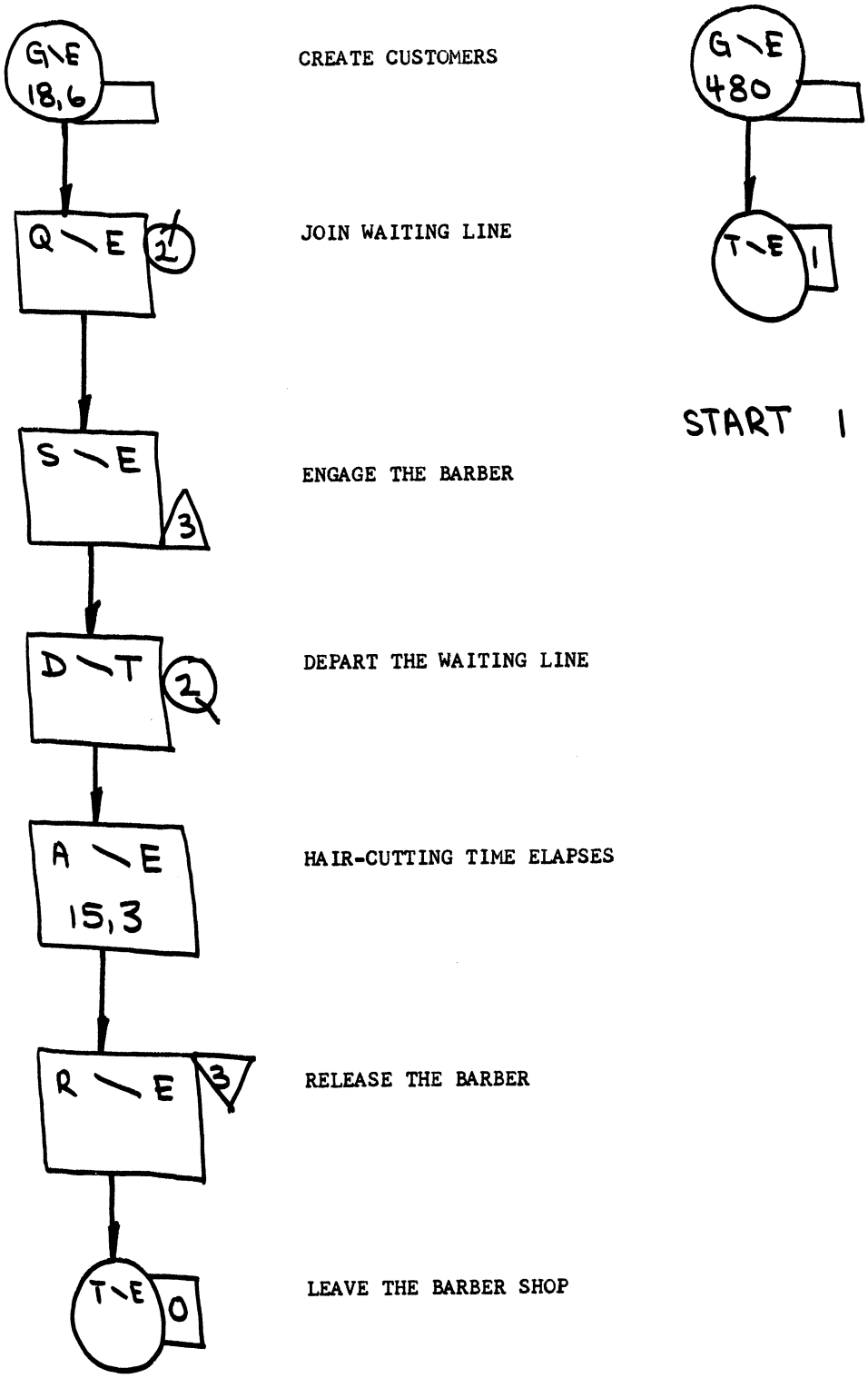


FIGURE 3-A: BLOCK DIAGRAM
(BARBER SHOP: FIRST MODEL)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
      SIMULATE
*
* THE BARBER SHOP PROBLEM - ONE BARBER, ONE TYPE OF CUSTOMER
* THIS MODEL MAINTAINS WAITING LINE STATISTICS
*
      GENERATE   18,6        CREATE CUSTOMERS
      QUEUE      2           CUSTOMERS QUEUE UP IF NECESSARY
      SEIZE      3           ENGAGE THE BARBER WHEN HE BECOMES AVAILABLE
      DEPART     2           CUSTOMER LEAVES THE QUEUE
      ADVANCE    15,3        CUSTOMER GETS HIS HAIR CUT
      RELEASE    3           RELEASE THE BARBER
      TERMINATE  0           LEAVE THE BARBER SHOP
*
      GENERATE   480         GENERATE A TIMER AFTER 8 HOURS OF SIMULATED TIME
      TERMINATE  1           SHUT OFF THE RUN
      START      1           CARRY OUT THE SIMULATION
      END                    RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                        SIGN-OFF
```

FIGURE 3-B: PROGRAM LISTING
(BARBER SHOP: FIRST MODEL)

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
3	.816	27	14.518	1	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
2	1	.085	27	16	59.2	1.518	3.727

$AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

```
    CPU TIME USED:

ASSEMBLY:       .273 SECONDS

EXECUTION:      .303 SECONDS

                **** ON AT  04:30.06
                **** OFF AT 04:30.22
                **** ELAPSED TIME        16.543  SEC.
                **** CPU TIME USED        4.552  SEC.
                **** STORAGE USED       152.013  PAGE-SEC.
                **** CARDS READ          22
                **** LINES PRINTED
                **** PAGES PRINTED
                **** CARDS PUNCHED        1
                **** DRUM READS
                **** APPROX. COST OF THIS RUN     $.41
```

FIGURE 3-C: SELECTED OUTPUT; TIME AND COST DATA
(BARBER SHOP: FIRST MODEL)

The model shown in Figure 3-A makes no provision for removing any Transactions that may be in the system at time 480 when the simulation is shut off. Such a provision can be included but requires use of several GPSS blocks not yet discussed. The "Barber Shop: Sixth Model" (Section 44) includes this provision.

In the output statistics for Facility 3, this information is noted:

1) The number of customers who "got into the chair" during the course of the day was 27 ("NUMBER ENTRIES").

2) The barber was busy 81.6% of the time ("AVERAGE UTILIZATION").

3) At the time the model shut off, there was a customer in the chair ("SEIZING TRANS. NO." is not blank, but shows "1" as the number of the Transaction currently engaging the Facility).

4) The average time for a haircut was 14.518 minutes ("AVERAGE TIME TRAN"). Compare this average with the ADVANCE Block A, B Operands of 15 and 3, respectively.

5) "PREEMPTING TRANS. NO." is blank, indicating that no Transaction was "preempting" the Facility when the simulation shut off. The pre-empty capability in GPSS will not be discussed until Section 61 in this monograph.

This information is available from the Queue 2 statistics:

1) There was never more than one customer in the waiting line (MAXIMUM CONTENTS").

2) A total of 27 customers entered the waiting line ("TOTAL ENTRIES") but of these, 16 were "ZERO ENTRIES", that is, entered and then immediately left the Queue without having had to wait at all. In total, 59.2% of the Queue entries were of the Zero Entry type ("PERCENT ZEROS").

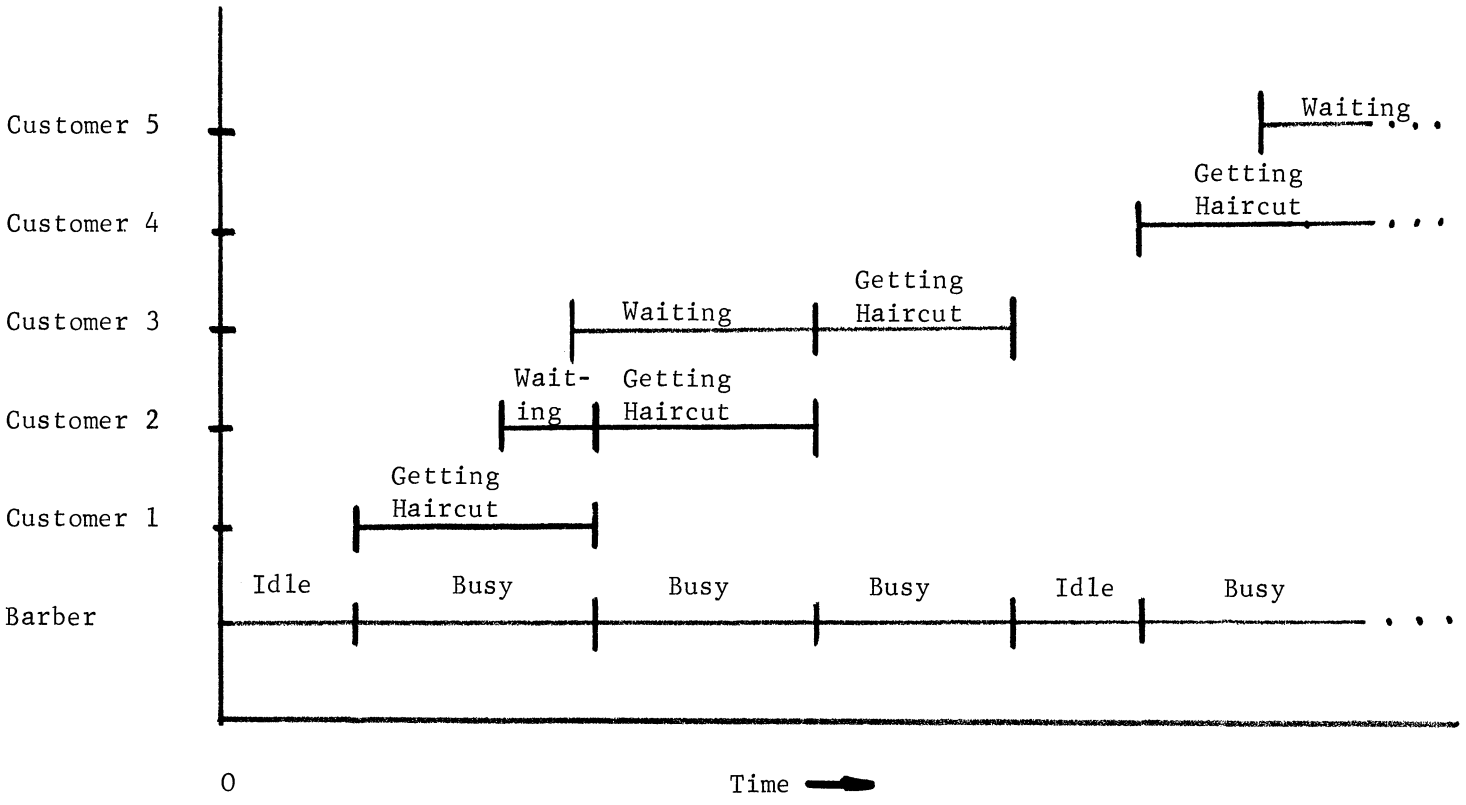
3) Those Queue entries which had to wait at all spent an average of 3.727 minutes in the Queue ("$AVERAGE TIME/TRANS").

4) On the average, there were 0.085 customers in the Queue ("AVERAGE CONTENTS").

Time and cost information indicates that 0.273 seconds of CPU (Central Processing Unit) time were used to assemble the program; and 0.303 seconds of CPU time were used in execution. This information, incidentally, is available only in GPSS/360 as embedded in the Michigan Terminal System, and should not be anticipated by users running under other Operating Systems. Other MTS statistics show Elapsed Time, Total CPU Time, Storage Used, etc. As indicated, the run cost about $0.41. All models appearing in this monograph were run in the Spring and Summer of 1968. If and when the accounting system changes, the approximate cost figures shown for these monograph models may also change somewhat.

17. Comments on the GPSS Approach to Internal Bookkeeping

***The time sequence of events in the barber shop model just studied may not be difficult to visualize. But the mind boggles in trying to clearly conceptualize the event sequence in a complicated system featuring many things which can happen at random. One possibility for keeping a record of system status is to use a graphical approach which might look like this for the barber shop:



This approach, although theoretically possible, is probably not very practical. The graphic approach is presented here briefly only to suggest the scope of the bookkeeping problem which is a part of simulation studies.

***A question which naturally arises is: What mechanism does GPSS use to keep track of the various Transactions moving through a model? The detailed answer to this question will be taken up in Section 45. It is useful to indicate now, however, that GPSS views each Transaction in a model as being on one(or, in some cases, two) of several "chains". Each Transaction on a chain may be thought of as a link in the chain. The concepts "front end of the chain" and "back end of the chain" are valid. There are five categories of chains:

1) Current Events Chain
2) Fugure Events Chain
3) Interrupt Chains
4) Matching Status Chains
5) User Chains

As the prose suggests, there is only one chain in each of the first two categories. User Chains are defined and used by the model builder for one or two specific reasons (to be discussed in Section 63), and there may be more than one such.

Briefly, the Current Events Chain is composed of all Transactions which are scheduled to experience movement through one or more Blocks at the current instant in simulated time, or as soon as possible. As the underlined phrase infers, included on the Current Events Chain are those Transactions which are experiencing a blocking condition (for example, which are scheduled to move into a SEIZE Block but cannot do so because the Facility involved is already engaged).

The Future Events Chain is composed of those Transactions which are not scheduled to move through one or more Blocks until some point in future time is reached. This condition can only result in either one of two ways:

1) A Transaction is at an ADVANCE Block, and is not scheduled to attempt to move on to the next sequential Block until a later time, or

2) A Transaction is scheduled to enter the model via a GENERATE Block at some future time.

Actually, a third situation can lead to inclusion of Transactions on the Future Events Chain, but only in a highly specialized context that need not be considered here.

Transactions are ordered on the Future Events Chain according to the time at which they are scheduled to again move. Suppose the simulated clock currently reads 48. Then a Transaction scheduled to move on from an ADVANCE Block at time 54 is nearer the front of the Future Events Chain than some Transaction scheduled to exit the same (or another) ADVANCE Block at, say, time 59.

GPSS updates the system by "scanning" the Current Events Chain from front to back, Transaction by Transaction. As each next Transaction is encountered, its Scan Status Indicator (see pages 4 and 5 in these notes) is first examined. A Transaction is "scan-inactive" if its Scan Status Indicator is On. The simulator does not attempt to move a scan-inactive Transaction forward in the model, but proceeds immediately to the next Transaction (if any) on the Current Events Chain. If, however, a Transaction is "scan-active" (i.e. its Scan Status Indicator is Off), the simulator attempts to move it forward along its scheduled path in the model until one of two conditions is encountered:

1) The Transaction moves into an ADVANCE Block where a non-zero holding time applies. In this case, the Transaction is then merged into the Future Events Chain according to the time when it is scheduled to attempt exiting the ADVANCE Block.

2) One of several blocking conditions is encountered, meaning that the Transaction cannot enter some scheduled next Block (of the five different Block types which can refuse entry to a Transaction, we have so far encountered only one: the SEIZE Block). In this case, the Transaction remains on the Current Events Chain and its Scan Status Indicator may or may not be turned on, depending on the type of blocking condition. (The Transaction may have successfully moved through several Blocks before the blocking condition was encountered. GPSS will of course update its record of the location of the Transaction in the model, even though the Transaction remains on the Current Events Chain.)

For various reasons to be discussed in Sections 29 and 45, the simulator may re-start its scan of the Current Events Chain at a given instant in simulated clock time. This possibility is mentioned now for the first time, even though the potential importance of re-starting the scan is probably not transparent.

Suppose now that the GPSS scan has resulted in processing the Transaction at the end of the Current Events Chain. (Remember that this does not necessarily mean the Current Events Chain is empty.) As its next step, the simulator then examines the Transaction at the front of the Future Events Chain and the simulated clock is advanced to whatever time that Transaction is scheduled to move again. That Transaction (and any behind it on the Future Events Chain also scheduled for movement at the new reading of the simulated clock) is then transferred from the Future Events Chain to the Current Events Chain. Incoming Transactions from the Future Events Chain are linked onto the Current Events Chain according to their PR (priority level) value. They are put behind any blocked Transactions with the same PR, in case any such were already on the Current Events Chain. Then the scan of the Current Events Chain is started and the basic cycle is repeated.

The prose description of GPSS "bookkeeping" admittedly may seem complicated on the first reading. These same ideas will be considered in greater detail in Section 45. Flowcharts describing the bookkeeping process will also be studied from the User's Manual. At this point, it is enough to be aware of the chain concept and to be cautioned that complicated systems cannot be modeled with confidence until these bookkeeping considerations have been mastered.

As a final point, realize that when Execution Errors are encountered in GPSS, part of the resulting output may include a printout of the Current Events Chain and the Future Events Chain. Such a printout can often be put to good use for debugging purposes. Pages 55-57 in the User's Manual display a typical chain printout and show how to interpret the information it provides.

18. GPSS Control Cards

***After a particular simulation run has been carried out with a model, the analyst may want to make one or more additional runs with the same model or with a somewhat modified version of it. Or, the user may want to study two or more entirely different models in a single batch run (or in a single terminal session). GPSS makes a set of Control Cards available to the user for this purpose.

***Consider the three conditions the model builder might like to bring about after a given simulation has been carried out (but before control is returned to the operating system):

1) Completely restore the model to its initial state (i.e. zero out the statistics gathered in the preceding run and remove any Transactions which may still be in the model). This may be desirable for several reasons:

   A) Another run is to be made with the identical model, or

   B) Slight modifications are to be made in the preceding model (such as changing the Limit Count in a GENERATE block; examples will be given later), then simulation with the modified model is to be carried out.

2) Partially restore the model to its initial state (i.e. zero out the statistics gathered in the preceding run but do not remove any Transactions which are in various intermediate locations in the model). If a system is at all complicated, a substantial amount of simulated time may have to elapse before the model reaches steady state conditions. The modeler may be interested in the behavior of the model as it moves through transient conditions, or after it reaches the steady state, or both. Partial restoration of the model can be specified by the user one or more times in GPSS, making it possible to gather statistics pertaining to any arbitrary time slice, either:

   A) as the model moves through transient conditions, or

   B) after the model has reached steady state.

3) Completely destroy the preceding model. This would be done before presenting the punchcard equivalent of an unrelated model which was to be studied.

***The Control Cards which bring about the conditions described above are:

Control Card	Effect
CLEAR	Completely restores the model to its initial state
RESET	Partially restores the model to its initial state
JOB	Completely destroys the preceding model

Similar to the START Card (which is also actually a Control Card), the "verbs" CLEAR, RESET, and JOB may be thought of as the card "Operation" and are punched in the Operation Field, starting in card column 8. The CLEAR and RESET Control Cards will now be illustrated in use with the preceding barber shop model.

19. Barber Shop: Second Model

Statement of the Problem

Show how to prepare a program deck for the preceding barber shop model which makes these runs possible in a single batch session:

1) First of all, carry out a simulation corresponding to an eight hour day, as before.

2) Then completely restore the model to its initial state and carry out a simulation for another eight hour day.

3) Completely restore the model, re-define the GENERATE block so that customers arrive every 12 $\pm$ 3 minutes, and simulate for a two hour period.

4) Then partially restore the model and simulate for the next two hour period (hours 3 and 4 of operation).

5) Again, partially restore the model and simulate for the next four hour period (hours 5 through 8 of operation).

Approach Taken in Building the Model

With the exception of the Control Cards, the model is identical to the one shown in Section 16. Note, however, that the GENERATE Block Operands must be modified at intermediate points in the simulation per the statement of the Problem. The two GENERATE Blocks have consequently been given the symbolic names KTRL1 and KTRL2 to facilitate these changes.

Table of Definitions

Identical to Table of Definitions in Section 16.

Block Diagram

Figure 4-A shows the GPSS block diagram model of the barber shop. The Control Card sequence to be used in the simulation is also indicated.

Program Listing

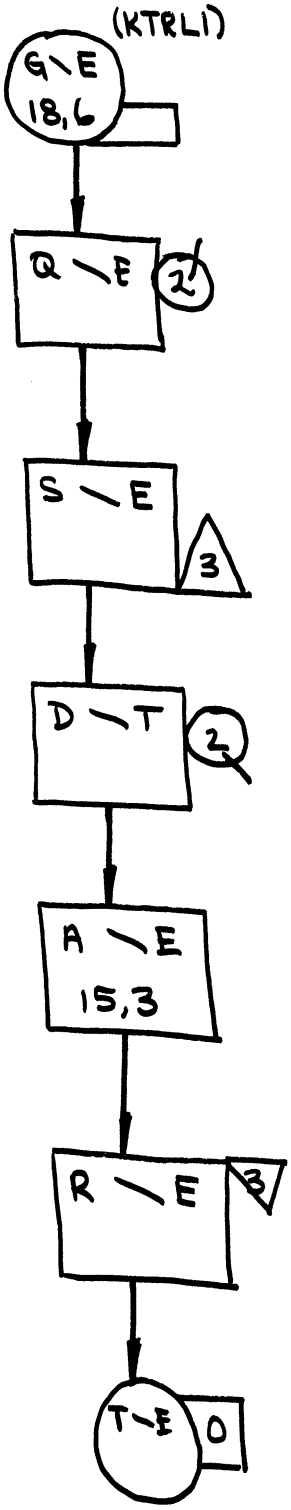
The punchcard equivalent of the Figure 4-A model is shown in Figure 4-B. The model itself is identical to that shown in Figure 3-A, except that the two GENERATE blocks have been given the symbolic names KTRL1 and KTRL2, respectively. This has been done so that the GENERATE blocks can be redefined with different A and B operands to meet the requirements of runs 3 through 5 above. The pattern of Control Cards (and re-definition of KTRL1 and KTRL2) should be examined closely. The reader should satisfy himself that this pattern corresponds to the sequence of runs requested above.

Program Output; Time and Cost Data

Figure 4-C shows selected program output and indicates time and cost data for the run.

Discussion of Output

A considerably larger part of the model output has been included in Figure 4-C than was the case for Figure 3-C (Output for the Section 16 model). This output has been labeled in segments as Figures 4-C-1 to 4-C-8



CREATE CUSTOMERS

JOIN WAITING LINE

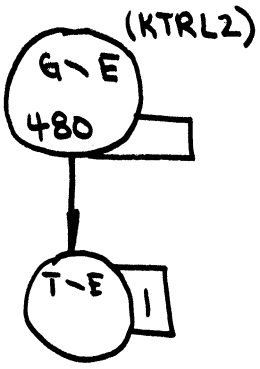
ENGAGE THE BARBER

DEPART THE WAITING LINE

HAIR-CUTTING TIME ELAPSES

RELEASE THE BARBER

LEAVE THE BARBER SHOP



CREATE A TIMER VIA CURRENT A,B OPERANDS IN GENERATE BLOCK NAMED KTRL2

SHUT OFF THE SIMULATION

```
          START     1
          CLEAR
          START     1
          CLEAR
KTRL1     GENERATE  12,3
KTRL2     GENERATE  120
          START     1
          RESET
          START     1
          RESET
KTRL2     GENERATE  240
          START     1
```

FIGURE 4-A: BLOCK DIAGRAM

(BARBER SHOP: SECOND MODEL)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
       SIMULATE
*
* THE BARBER SHOP PROBLEM - ONE BARBER, ONE TYPE OF CUSTOMER
* THIS MODEL ILLUSTRATES USE OF THE CONTROL CARDS 'CLEAR' AND 'RESET'
*
 KTRL1 GENERATE   18,6        CREATE CUSTOMER ARRIVALS
       QUEUE      2           CUSTOMERS QUEUE UP IF NECESSARY
       SEIZE      3           ENGAGE THE BARBER WHEN HE BECOMES AVAILABLE
       DEPART     2           CUSTOMER LEAVES THE QUEUE
       ADVANCE    15,3        CUSTOMER GETS HIS HAIR CUT
       RELEASE    3           RELEASE THE BARBER
       TERMINATE  0           LEAVE THE BARBER SHOP
*
 KTRL2 GENERATE   480         GENERATE A TIMER AFTER 8 HOURS OF SIMULATED TIME
       TERMINATE  1           SHUT OFF THE RUN
       START      1           CARRY OUT THE SIMULATION
*
       CLEAR                  COMPLETELY RESTORE THE PRECEDING MODEL
       START      1           CARRY OUT THE SIMULATION AGAIN
*
       CLEAR                  COMPLETELY RESTORE THE PRECEDING MODEL
 KTRL1 GENERATE   12,3        RE-DEFINE THE 'CUSTOMER' GEN BLOCK AS REQUESTED
 KTRL2 GENERATE   120         RE-DEFINE THE 'TIMER' GEN BLOCK AS REQUIRED
       START      1           CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
*
       RESET                  PARTIALLY RESTORE THE PRECEDING MODEL
       START      1           CONTINUE THE SIM STARTED WITH THE PRECEDING MODEL
*
       RESET                  PARTIALLY RESTORE THE PRECEDING MODEL
 KTRL2 GENERATE   240         RE-DEFINE THE 'TIMER' GEN BLOCK AS REQUIRED
       START      1           FINISH THE SIMULATION AS REQUESTED
       END                    RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                         SIGN-OFF
```

FIGURE 4-B: PROGRAM LISTING
(BARBER SHOP: SECOND MODEL)

BLOCK NUMBER	SYMBOL
1	KTRL1
8	KTRL2

FIGURE 4-C-1: BLOCK SYMBOL TABLE

```
  1      GENERATE    18     6
  2      QUEUE       2
  3      SEIZE       3
  4      DEPART      2
  5      ADVANCE     15     3
  6      RELEASE     3
  7      TERMINATE   0
 *
  8      GENERATE    480
  9      TERMINATE   1
         START       1
```

FIGURE 4-C-2: ASSEMBLED PROGRAM

RELATIVE CLOCK 480 ABSOLUTE CLOCK 480

BLOCK COUNTS

BLOCK	CURRENT	TOTAL
1	0	27
2	0	27
3	0	27
4	0	27
5	1	27
6	0	26
7	0	26
8	0	1
9	0	1

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
3	.816	27	14.518	1	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
2	1	.085	27	16	59.2	1.518	3.727

$AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

FIGURE 4-C-3: OUTPUT (FIRST EIGHT HOUR DAY SIMULATION)

RELATIVE CLOCK 480 ABSOLUTE CLOCK 480

BLOCK COUNTS

BLOCK	CURRENT	TOTAL
1	0	28
2	1	28
3	0	27
4	0	27
5	1	27
6	0	26
7	0	26
8	0	1
9	0	1

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
3	.899	27	16.000	3	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
2	1	.177	28	11	39.2	3.035	5.000

FIGURE 4-C-4: OUTPUT (SECOND EIGHT HOUR DAY SIMULATION)

RELATIVE CLOCK 120 ABSOLUTE CLOCK 120

BLOCK COUNTS

BLOCK	CURRENT	TOTAL
1	0	10
2	3	10
3	0	7
4	0	7
5	1	7
6	0	6
7	0	6
8	0	1
9	0	1

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
3	.883	7	15.142	5	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
2	3	1.091	10	1	9.9	13.099	14.555

FIGURE 4-C-5: OUTPUT (HOURS 1 AND 2 OF THIRD EIGHT HOUR DAY)

RELATIVE CLOCK 120 ABSOLUTE CLOCK 240

BLOCK COUNTS

BLOCK	CURRENT	TOTAL
1	0	10
2	5	10
3	0	8
4	0	8
5	1	8
6	0	8
7	0	8
8	0	1
9	0	1

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
3	1.000	9	13.333	6	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
2	5	4.000	13		.0	36.923	36.923

FIGURE 4-C-6: OUTPUT (HOURS 3 AND 4 OF THIRD EIGHT HOUR DAY)

```
RELATIVE CLOCK          240  ABSOLUTE CLOCK          480
BLOCK COUNTS
BLOCK CURRENT     TOTAL
    1       0        19
    2       8        19
    3       0        16
    4       0        16
    5       1        16
    6       0        16
    7       0        16
    8       0         1
    9       0         1


FACILITY      AVERAGE          NUMBER          AVERAGE          SEIZING    PREEMPTING
              UTILIZATION      ENTRIES         TIME/TRAN        TRANS. NO. TRANS. NO.
       3      1.000              17            14.117           10


QUEUE      MAXIMUM     AVERAGE     TOTAL      ZERO      PERCENT     AVERAGE      $AVERAGE
           CONTENTS    CONTENTS    ENTRIES    ENTRIES   ZEROS       TIME/TRANS   TIME/TRANS
    2          8       6.254         24                   .0        62.541       62.541
```

FIGURE 4-C-7: OUTPUT (HOURS 5 THROUGH 8 OF THIRD EIGHT HOUR DAY)

```
CPU TIME USED:

ASSEMBLY:       .504 SECONDS

EXECUTION:     1.072 SECONDS


**** ON AT  04:30.26
**** OFF AT 04:30.45
**** ELAPSED TIME        19.346  SEC.
**** CPU TIME USED         5.559  SEC.
**** STORAGE USED       201.406  PAGE-SEC.
**** CARDS READ           37
**** LINES PRINTED
**** PAGES PRINTED
**** CARDS PUNCHED         1
**** DRUM READS
**** APPROX. COST OF THIS RUN     $.52
```

FIGURE 4-C-8: TIME AND COST DATA

Figure 4-C-1 shows the Block Symbol Table. All model Blocks which were given symbolic names by the programmer are listed in this table, together with the corresponding Block Number assignment made by the simulator. Hence KTRL1 is synonymous with 1, and KTRL2 is synonymous with 8.

Figure 4-C-2 shows the assembled version of the original punchcards constituting the model. Note that each Block has been assigned a unique number by the simulator. (Consistent with the Block Symbol Table, the second GENERATE block is numbered 8.) The Block Number is simply the "Location" of the Block as already discussed on page 7. Note in Figure 4-C-2 that the Block Operands have been shifted to the right and intervening commas have been removed.

Figure 4-C-3 indicates output from the first eight hour day simulation. In addition to Facility 3 and Queue 2 statistics, information concerning the RELATIVE CLOCK, ABSOLUTE CLOCK, and BLOCK COUNTS is included. Definitions for these three concepts are now given:

ABSOLUTE CLOCK: The Absolute Clock registers the point reached in simulated time since the last time a model was completely restored to its original state, i.e. since the last time a CLEAR card was encountered by the simulator. When a model is first submitted, the Absolute Clock is given an initial value of zero. The value of the Absolute Clock is not available during the course of a simulation.

RELATIVE CLOCK: The Relative Clock records the point reached in simulated time since the last time a model was partially restored to its original state, i.e. since the last time a RESET card was encountered by the simulator. When a model is first submitted, the Relative Clock is given an initial value of zero. The value of the Relative Clock is available as a simulation proceeds (and so can be used to help control the course of a simulation). The Relative Clock is addressed as C1. C1 is one of the GPSS Standard Numerical Attributes.

Hence there are really two simulation clocks in GPSS, although this was not indicated as such in Section 3.

BLOCK COUNTS: Each Transaction which is active in a model is viewed by the simulator as being "at exactly one Block in the model. The Block which any particular Transaction is "at" of course changes, in general, as a simulation proceeds.

The Current Block Count indicates the number of Transactions at the various Blocks in a model when the simulation stopped. In Figure 4-C-3 we see, for example, that there was 1 Transaction at Block 5 when the simulation stopped.

The Total Block Count indicates the total number of Transactions which have ever been at (or are still at) the various Blocks in a model when the simulation stopped. In Figure 4-C-3, for example, we see that a total of 27 Transactions had been at Block 5 during the course of the simulation. Of the 27, 26 had moved on to subsequent Blocks by the end of the simulation, and 1 was still at Block 5 when the simulation stopped.

Current Block Counts and Total Block Counts are available as a simulation proceeds and can consequently be used in structuring model logic. The Counts are addressed as Wj and Nj, respectively, where j is the number of the Block in Question. Hence, referring again to Figure 4-C-3, W5 is 1 and N5 is 27. Wj and Nj are members of the set of GPSS Standard Numerical Attributes.

The modeler usually does not know what Block Number will be assigned by the simulator to a Block for which the Nj or Wj is to be referenced dynamically as a simulation proceeds. This difficulty is overcome by giving a symbolic name to the Block, then referencing the Block Counts as W$@ or N$@, where @ represents the symbolic name chosen. Referring first to Figure 4-C-1 and then Figure 4-C-3, we see for example that W$KTRL2 was 0 at the end of the simulation. Similarly, N$KTRL2 was 1 when the simulation terminated.

The following table shows the effect of the Control Cards RESET and CLEAR on the Block Counts:

Control Card	Effect on Current Block Counts	Effect on Total Block Counts
RESET	Not Altered	Set equal to Current Block Counts
CLEAR	Set to Zero	Set to Zero

These effects are consistent with the "partial model restoration" and "complete model restoration" roles of the RESET and CLEAR cards, respectively.

When a model is first submitted, all Block Counts are given an initial value of zero.

Figure 4-C-4 is output for the second eight hour day simulation. Note the differences in simulation results for this "second" eight hour day.

Figures 4-C-5 through 4-C-7 display results for the three time slices which make up the "third" eight hour day. Third day results cannot be considered in the same light as first and second day results because the customer interarrival time has been decreased. For each time slice in the third day, note:

1) Relative Clock vs. Absolute Clock,

2) The growth pattern in Facility Utilization,

3) The increasing Queue Contents, and

4) The increasing value of W2 (the number of Transactions currently at the QUEUE Block).

These latter three phenomena are consistent with the fact that on the third day, customers are arriving faster than the barber can handle them.

20. Barber Shop: Third Model

Statement of the Problem

Modify the model shown in Figure 3-A to eliminate the explicit waiting line included as part of that model.

Approach Taken in Building the Model

The model involves more than a repetition of Figure 3-A with the QUEUE/DEPART complementary block pair eliminated. If the QUEUE block were eliminated in Figure 3-A, then the next sequential block after the GENERATE block would be the SEIZE block, which can refuse entry to Transactions. This can affect the apparent operation of the GENERATE block, because the time of arrival of the next Transaction to the GENERATE block is not scheduled until its predecessor Transaction succeeds in exiting to the next sequential block (see page 9 of these notes and page 15, IUM). The requested interarrival time distribution (18±6 in this case) can consequently be distorted.

This problem is resolved by creating only one Transaction at the GENERATE block (using a Limit Count of 1). This Transaction is created when the simulation starts and moves immediately into an ADVANCE block, where a holding time of 18±6 is specified. The Transaction being held at this ADVANCE block can be thought of as a customer on his way to the barber shop. Departure from the ADVANCE is equivalent to arrival at the shop. The next block after the ADVANCE is a SPLIT, where one offspring is created and routed back to the ADVANCE to be the next customer on his way to the shop. The parent Transaction goes on to SEIZE the barber in his turn, etc. In conclusion, the SPLIT and ADVANCE preceding it are used to simulate the GENERATE block; "holding time" is being used as a substitute for the GPSS capability of structuring "interarrival time".

Table of Definitions

Definitions used are the same as those for the model of Figure 3-A.

Other Model Documentation

Figure	Information Exhibited
5-A	Block Diagram
5-B	Program Listing
5-C	Selected Program Output; Time and Cost Data

Discussion of Block Diagram and Output

The Block Diagram corresponds to the considerations described in the Approach Taken in Building the Model. Nothing new is offered in the Output, except that in the Table of Block Counts it is worthwhile to compare N2, N3, and N4. Computer Time and Cost Data can be compared with that in Figure 3-C.

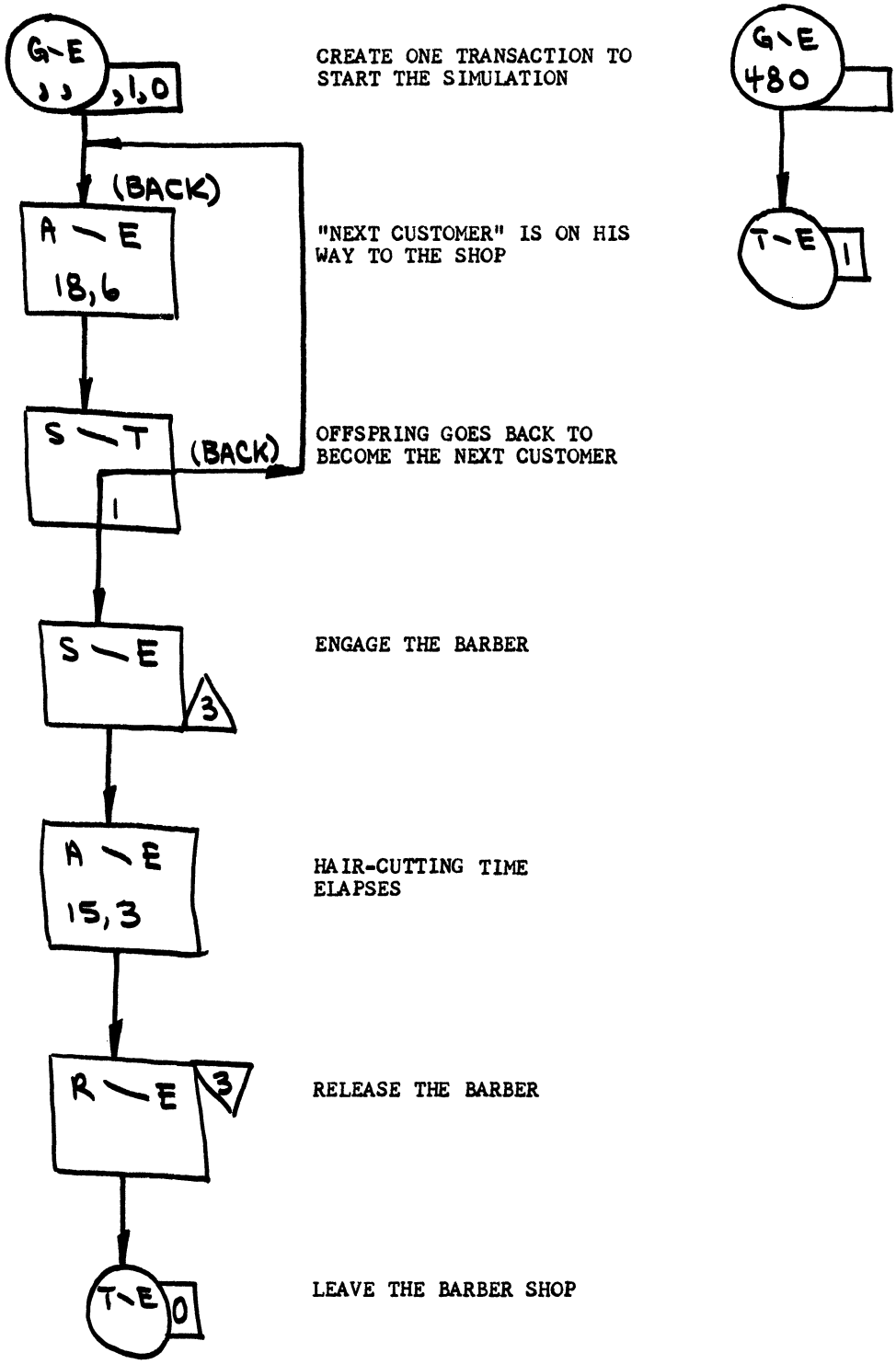


FIGURE 5-A: BLOCK DIAGRAM
(BARBER SHOP: THIRD MODEL)

```
$SIGNON 505k PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
      SIMULATE
*
* THE BARBER SHOP PROBLEM - ONE BARBER, ONE TYPE OF CUSTOMER
* THIS MODEL DOES NOT MAINTAIN WAITING LINE STATISTICS
*
      GENERATE    ,,,1,0      CREATE ONE XACT TO START THE RUN
BACK  ADVANCE     18,6        THE 'NEXT CUSTOMER' IS ON HIS WAY TO THE SHOP
      SPLIT       1,BACK      OFFSPRING GOES BACK TO BECOME NEXT CUSTOMER
      SEIZE       3           ENGAGE THE BARBER WHEN HE BECOMES AVAILABLE
      ADVANCE     15,3        CUSTOMER GETS HIS HAIR CUT
      RELEASE     3           RELEASE THE BARBER
      TERMINATE   0           LEAVE THE BARBER SHOP
*
      GENERATE    480         CREATE A TIMER AFTER 8 HOURS OF SIMULATED TIME
      TERMINATE   1           SHUT OFF THE RUN
      START       1           CARRY OUT THE SIMULATION
      END                     RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                         SIGN-OFF
```

FIGURE 5-B: PROGRAM LISTING
(BARBER SHOP: THIRD MODEL)

```
 1      GENERATE                    1      0
 2      ADVANCE     18      6
 3      SPLIT       1       2
 4      SEIZE       3
 5      ADVANCE     15      3
 6      RELEASE     3
 7      TERMINATE   0
*
 8      GENERATE    480
 9      TERMINATE   1
        START       1


RELATIVE CLOCK          480  ABSOLUTE CLOCK          480
BLOCK COUNTS
BLOCK CURRENT    TOTAL
    1      0         1
    2      1        29
    3      0        56
    4      0        28
    5      1        28
    6      0        27
    7      0        27
    8      0         1
    9      0         1


FACILITY        AVERAGE          NUMBER         AVERAGE          SEIZING    PREEMPTING
                UTILIZATION      ENTRIES        TIME/TRAN        TRANS. NO. TRANS. NO.
       3          .887             28            15.214            4


                                 **** ON AT  04:30.49
                                 **** OFF AT 04:31.06
     CPU TIME USED:              **** ELAPSED TIME       16.54   SEC.
                                 **** CPU TIME USED       4.546  SEC.
 ASSEMBLY:      .292 SECONDS     **** STORAGE USED      151.573  PAGE-SEC.
                                 **** CARDS READ         22
 EXECUTION:     .275 SECONDS     **** LINES PRINTED
                                 **** PAGES PRINTED
                                 **** CARDS PUNCHED       1
                                 **** DRUM READS
                                 **** APPROX. COST OF THIS RUN    $.41
```

FIGURE 5-C: SELECTED OUTPUT; TIME AND COST DATA
(BARBER SHOP: THIRD MODEL)

21. Unconditionally Routing a Transaction to a Non-sequential Block

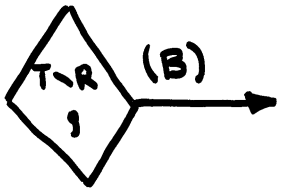
***We already know of one possibility for diverting Transactions to some non-sequential block in a GPSS model. In particular, offspring Transactions exiting a SPLIT block are routed to the block whose symbolic name appears as the B operand. The offspring may, of course, be sent along with the parent to the next sequential block by giving it a symbolic name and using the name as the SPLIT block's B operand.

***The GPSS block which makes it possible to unconditionally route a Transaction to a non-sequential block is the TRANSFER block.

The TRANSFER Block:

Purpose: Unconditionally route Transactions to a non-sequential block.

Shape and Operands:



Operand	Significance
A	The A operand is not specified when the TRANSFER block is used to unconditionally divert Transactions to some non-sequential point in a model. The presence of the comma in the TRANSFER block diamond suggests this "default case" for the A operand.
B	Specifies the symbolic name of the non-sequential block to which Transactions entering the TRANSFER block are routed.

Note

***The TRANSFER block used as described above is said to be operating in "Unconditional Transfer Mode". There are alternative modes in which the TRANSFER block can be used. Discussion of these alternatives will be deferred until later.

22. Barber Shop: Fourth Model

Statement of the Problem

Two types of customers arrive at a one-chair barber shop: those who called ahead, and those who didn't. The barber serves both types of customer on a first-come, first-served basis within type, but gives preference to the call-ahead type. The interarrival time is the same for both types, 36± 12 minutes, uniformly distributed. The barber gives one haircut every 15 minutes, on the average. The time per haircut ranges between 12 and 18 minutes, however, with equal likelihood Model the barber shop in GPSS, then simulate the operation of the shop through eight hours of operation. Determine the percent idle time of the barber, the average waiting time of each type of customer, and the maximum length to which the waiting line for each type of customer grew.

Approach Taken in Building the Model

The queue discipline followed by GPSS (First-come, first-served within priority level) can be taken advantage of here by giving call-ahead customers a higher priority level than customers who didn't call ahead. The PR values can be established by providing one GENERATE Block for each of the two customer types and making use of the GENERATE E operand. Separate waiting line statistics will be automatically provided if two Queues are established, one for each customer type. Finally, Transactions coming from the two GENERATE Blocks can be put into a common channel (to compete for the barber) with the aid of the Transfer Block.

Table of Definitions

Time Unit: 1 Minute

GPSS Entity	Interpretation
Transaction	A customer in the Main Segment PR=1: Didn't-call-ahead customer PR=2: Call-ahead customer A "Timer" in the Timer Segment
Facility 3	The Barber
Queue 1	The Waiting Line for Didn't-call-ahead Customers
Queue 2	The Waiting Line for Call-ahead Customers

Other Model Documentation

Figure	Information Exhibited
6-A	Block Diagram
6-B	Program Listing
6-C	Selected Program Output; Time and Cost Data

Discussion of Block Diagram

Separate GENERATE blocks are used to create the two types of customers. The GENERATE block E Operand is used to establish the priority levels. After joining Queue 1, didn't-call-ahead customers proceed directly to compete for the barber; after joining Queue 2, call-ahead customers

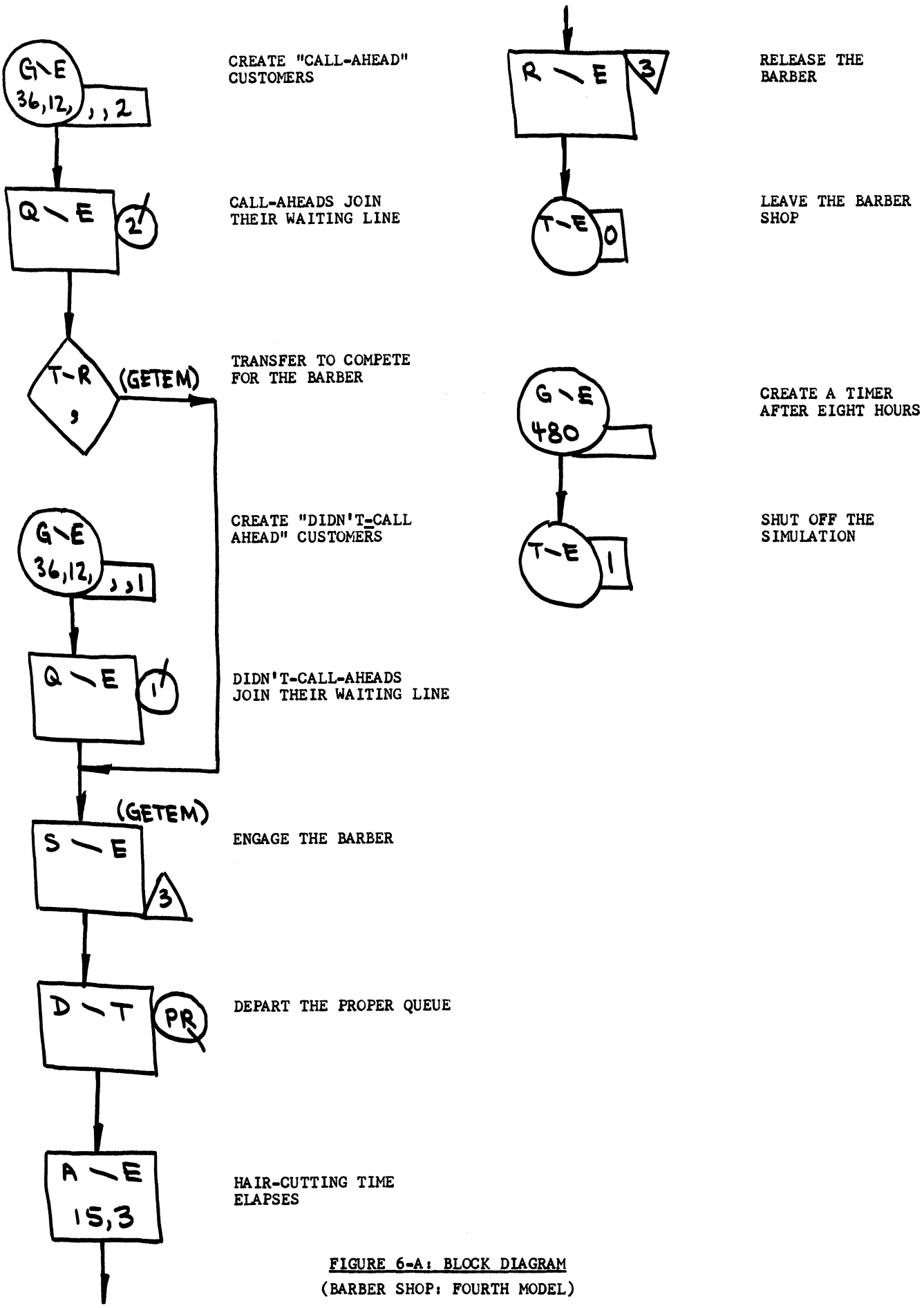


FIGURE 6-A: BLOCK DIAGRAM
(BARBER SHOP: FOURTH MODEL)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
      SIMULATE
*
* THE BARBER SHOP PROBLEM - ONE BARBER, TWO TYPES OF CUSTOMERS -
* THOSE WHO 'CALLED AHEAD' AND THOSE WHO DIDN'T
*
      GENERATE   36,12,,,2  CREATE 'CALL AHEAD' CUSTOMERS WITH PR = 2
      QUEUE      2          QUEUE UP IN WAITING LINE
      TRANSFER   ,GETEM     TRANSFER TO COMPETE FOR THE BARBER
      GENERATE   36,12,,,1  CREATE 'DIDN'T CALL AHEAD' CUSTOMERS WITH PR = 1
      QUEUE      1          QUEUE UP IN WAITING LINE
GETEM SEIZE      3          ENGAGE THE BARBER WHEN HE BECOMES AVAILABLE
      DEPART     PR         DEPART THE PROPER QUEUE ACCORDING TO CUSTOMER TYPE
      ADVANCE    15,3       CUSTOMER GETS HIS HAIR CUT
      RELEASE    3          RELEASE THE BARBER
      TERMINATE  0          LEAVE THE BARBER SHOP
*
      GENERATE   480        CREATE A TIMER AFTER 8 HOURS OF SIMULATED TIME
      TERMINATE  1          SHUT OFF THE SIMULATION
      START      1             SIMULATE WITH THE MODIFIED MODEL
      END                   RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                       SIGN-OFF
```

FIGURE 6-B: PROGRAM LISTING

(BARBER SHOP: FOURTH MODEL)

```
  1     GENERATE   36     12              2
  2     QUEUE      2
  3     TRANSFER          6
  4     GENERATE   36     12              1
  5     QUEUE      1
  6     SEIZE      3
  7     DEPART     PR
  8     ADVANCE    15     3
  9     RELEASE    3
  10    TERMINATE  0
*
  11    GENERATE   480
  12    TERMINATE  1
        START      1

RELATIVE CLOCK          480  ABSOLUTE CLOCK          480
BLOCK COUNTS
BLOCK CURRENT    TOTAL    BLOCK CURRENT    TOTAL
    1      0       14       11      0        1
    2      0       14       12      0        1
    3      0       14
    4      0       12
    5      0       12
    6      0       26
    7      0       26
    8      1       26
    9      0       25
   10      0       25

FACILITY        AVERAGE        NUMBER        AVERAGE        SEIZING    PREEMPTING
                UTILIZATION    ENTRIES       TIME/TRAN      TRANS. NO. TRANS. NO.
        3          .804           26          14.846          4

QUEUE       MAXIMUM     AVERAGE     TOTAL      ZERO      PERCENT      AVERAGE      $AVERAGE
            CONTENTS    CONTENTS    ENTRIES    ENTRIES   ZEROS        TIME/TRANS   TIME/TRANS
     1          1          .127        12          4       33.3          5.083        7.625
     2          1          .079        14          9       64.2          2.714        7.599

                                    **** ON AT  04:31.10
                                    **** OFF AT 04:31.27
                                    **** ELAPSED TIME      16.876  SEC.
        CPU TIME USED:              **** CPU TIME USED      4.644  SEC.
                                    **** STORAGE USED     156.35   PAGE-SEC.
    ASSEMBLY:      .331 SECONDS     **** CARDS READ        25
                                    **** LINES PRINTED
    EXECUTION:     .331 SECONDS     **** PAGES PRINTED
                                    **** CARDS PUNCHED      1
                                    **** DRUM READS
                                    **** APPROX. COST OF THIS RUN    $.42
```

FIGURE 6-C: SELECTED OUTPUT; TIME AND COST DATA

(BARBER SHOP: FOURTH MODEL)

TRANSFER to the block named GETEM to join in the competition. The rest of the model is similar to the preceding barber shop models. Notice that there is only one DEPART block, however, which is used by both types of customers. The A Operand of the DEPART Block is PR, which means the PR value of a Transaction specifies the number of the Queue which is to be DEPARTed. Call-ahead customers consequently DEPART Queue 2 (consistent with the logic used in building the model), and didn't-call-aheads DEPART Queue 1.

Discussion of Output

Information available in the output includes:

1) There were 14 call-ahead and 12 didn't-call-ahead customers, respectively, who entered the shop.

2) Call-aheads were able to get their haircut without waiting 64.2% of the time. This was possible for the didn't-call-aheads only 33.3% of the time.

3) Call-aheads who had to wait for the barber waited about as long (7.5 minutes) as those who didn't call ahead (7.6 minutes).

4) There never was more than 1 call-ahead waiting; there never was more than 1 didn't-call-ahead waiting.

5) At the time the simulation shut off, a Transaction was engaging the Facility (Transaction Number 4). There were no customers waiting (the number of Queue entries equals the number of Facility entries).

23. Storages in GPSS

***A Storage is the GPSS term for the capability of "providing service" to a Transaction. A Storage can in general accommodate more than one Transaction at a time.

***It is convenient to think of "service" in units. In this sense, a Storage is a capability for providing, in general, more than one unit of service at a time. Contrast Storages with Facilities, which by definition can provide only one unit of service at a time (can accommodate only one Transaction at a time).

***A given Transaction may demand only one unit of service from a Storage, or it may demand more than one unit of service.

***The programmer defines the number of units of service which any particular Storage can provide simultaneously. This is known as the Storage capacity.

***Similar to Facilities and Queues, Storages are numbered to provide a basis for differentiating among them.

***The number of Storages which can exist in a model depends on the amount of computer storage available

***Do not confuse the GPSS concept of "Storage" with the general concept in computing of "a storage location".

***It is sometimes convenient to think of a "unit of service" and a "unit of space" as being analogous.

***Some possible interpretations of Storages:

A Storage is a book service counter at a college bookstore; the capacity of the Storage is the number of girls working behind the counter; a Transaction is a shopper; each shopper requires one unit of service (can be serviced by one girl).

A Storage is a hotel; the capacity of the Storage is the number of beds in the hotel; a Transaction is the head of a family; the number of units of space demanded (beds) is equal to the number of members in the family.

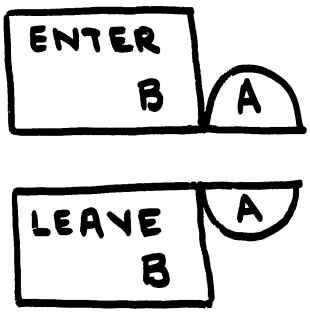
24. Complementary Block Pair for Storages

Purpose: Change the recorded status of Storages, i.e. appropriately modify the record of the units of space which are currently occupied.

Cause certain statistics relative to Storages to be automatically maintained.

Prevent Transactions from occupying a Storage which does not currently have sufficient units of unoccupies space to accommodate the corresponding Transaction requirements (only one block of the complementary pair has this effect).

Shape and Operands:



Operand	Significance
A	Specifies the number of the Storage which the Transaction is to ENTER or is to LEAVE.
B	Specifies the number of units of service (space) which the Transaction demands (when ENTERing) or frees up (when LEAVE-ing). If the B operand is not specified, a value of 1 is automatically assumed.

Nature of Operation: For each Storage used in a model, there is a storage location known as Rj and another storage location known as Sj, where j is the number of the particular Storage in question. The value of Rj is the number of units of space in the Storage which are currently unoccupied; the value of Sj is, of course, the number of units of currently occupied space. The reader can now predict how the ENTER/LEAVE block pair operate.

The ENTER Block

When a Transaction attempts to gain entry to the ENTER block, the Rj value of the appropriate Storage is checked against the B operand's value. One of two conditions results:

Rj $\geq$ B:

The Transaction gains entry to the ENTER block;
Storage j statistics are updated;
The Transaction exits the ENTER block and moves on to the next sequential block.

Rj $<$ B:

The Transaction is refused entry to the ENTER block;
The Transaction waits "ahead of" the ENTER block until Rj $\geq$ B, at which time it competes with other Transactions (if any) also waiting for the condition Rj $\geq$ B for the privilege of moving into the ENTER block. (See page 14, these notes, for a statement concerning the outcome of the competition.)

The LEAVE Block

When a Transaction encounters a LEAVE block in its path:

The Transaction enters the LEAVE block;
Storage j statistics are updated;
The Transaction exits the LEAVE block, if possible, and moves to the next sequential block in the model.

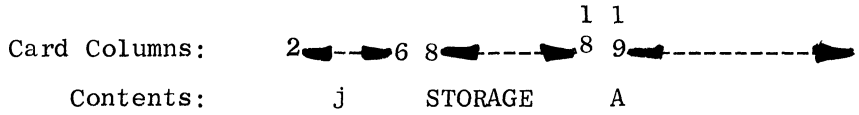
Note

***A Transaction can LEAVE a Storage that it did not previously ENTER. (Contrast this situation with that for Facilities, page 15 these notes.)

***Sj (the current contents of Storage j) cannot be decremented to a value less than zero; if an attempt is made to do this, Running Error Stop No. 425 occurs

Storage Capacity Definition Card

***The capacity of each Storage in a model is defined with a punchcard that takes this form:



where:

j is the number of the Storage whose capacity is being defined, and

A is the capacity of the Storage.

25. Storage Statistics Automatically Maintained by GPSS

***In addition to the Standard Numerical Attributes Rj and Sj, for each Storage in a model there are five other Standard Numerical Attributes:

1) SAj: The average contents of Storage j, that is, the average number of units of space occupied to date.

2) SCj: The number of entries to date for Storage j; not to be confused with the number of Transactions which have succeeded in gaining entry to the ENTER Block.

3) SMj: The maximum contents to date of Storage j, that is, the greatest number of units of space occupied at one time.

4) SRj: The fractional utilization of Storage j, in parts per thousand.

5) STj: The average time to date that each Transaction used Storage j (integerized).

26. Barber Shop: Model Five

Statement of the Problem

Customers arrive at a three-chair barber shop with an average time between arrivals of 6 minutes. The interarrival time actually varies between 2 and 10 minutes, with equal likelihood for each possibility. The three barbers are uniform in the sense that each requires an average of 15 minutes to give a haircut. The time per haircut ranges between 12 and 18 minutes for each barber, however, with equal likelihood. Model the barber shop in GPSS, then simulate the operation of the shop through eight hours of operation. Determine the percent idle time of the barbers (considered as a group), the average waiting time of the customers, and the maximum number of customers who at one and the same time were waiting for the barber. Assume that customer are always willing to have their hair cut by the next barber who becomes free.

Approach Taken in Building the Model

The service capability in this barber shop can be represented as a Storage with a capcity of three units. Even though there are three barbers operating in parallel, the model can be built with a single sequence of blocks. No differentiation is made among the barbers because they provide service at a uniform rate.

Table of Definitions

Time Unit: 1 Minute

GPSS Entity	Interpretation
Transaction	A Customer in the Main Segment
	A "Timer" in the Timer Segment
Queue 1	The Waiting Line
Storage 1	The Three Barbers

Other Model Documentation

Figure	Information Exhibited
7-A	Block Diagram
7-B	Program Listing
7-C	Selected Program; Time and Cost Data

Discussion of Block Diagram

Figure 7-A can be compared directly with Figure 3-A. The only real difference between the Figures is that the SEIZE/RELEASE pair in Figure 3-A has been replaced with an ENTER/LEAVE pair in Figure 7-A. Of course the A, B Operands in the GENERATE blocks differ, reflecting the difference between the two problems in interarrival times. Also, Queue 3 is the waiting line in Figure 3-A, while 1 has arbitrarily been chosen as the number of the Queue in Figure 7-A. Notice that the Storage capacity definition appears as part of the block diagram.

Discussion of Output

About 20% of the customers had to wait for a barber for an average of 3 minutes. In Barber Shop: First Model (page 25), about 40% of the customers had to wait for an average of 3.7 minutes. Yet "expected interarrival time per barber" was the same in both models. Can an inference be made from this information (remember that duration of the simulation was small in both cases)?

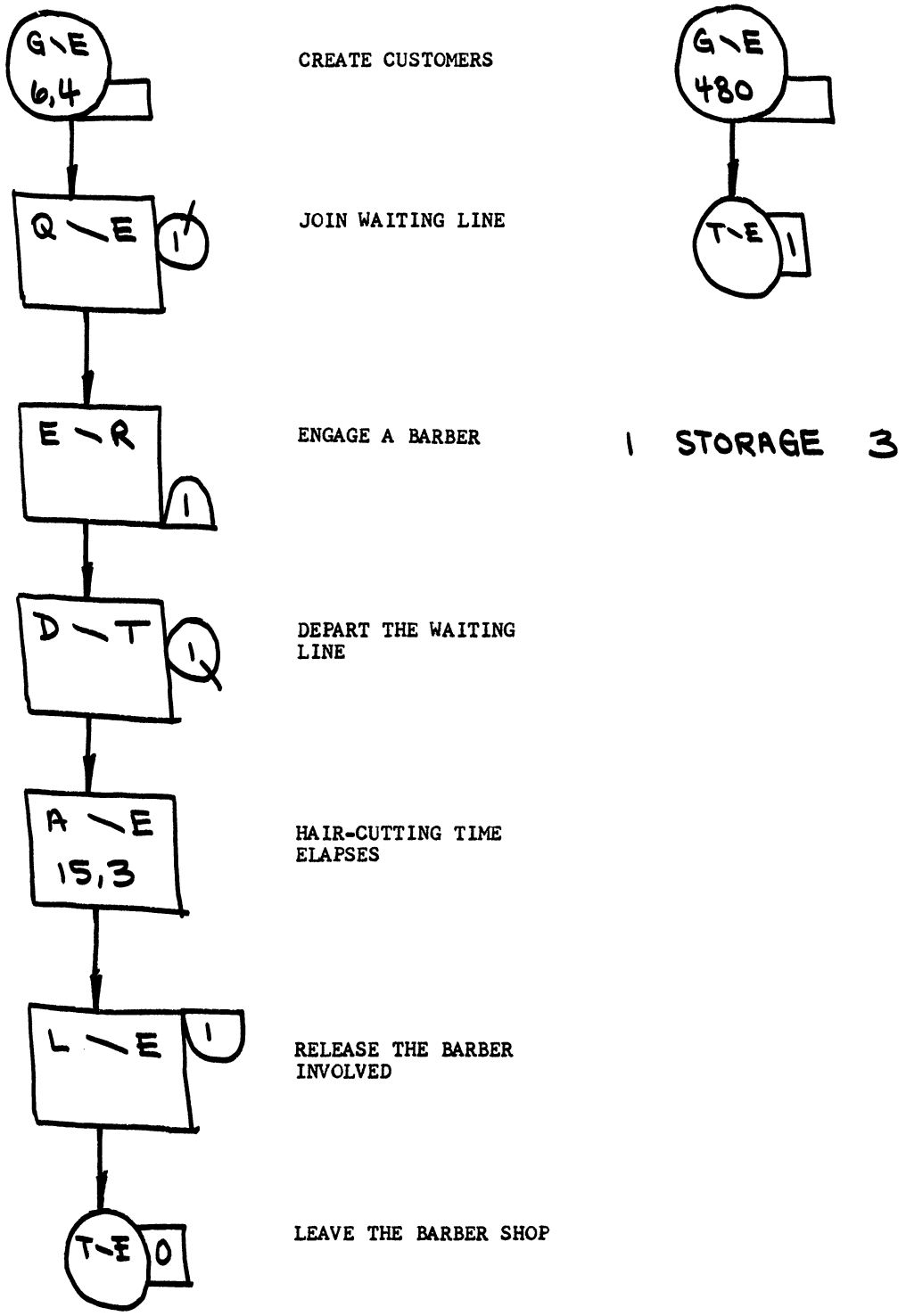


FIGURE 7-A: BLOCK DIAGRAM
(BARBER SHOP: FIFTH MODEL)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
      SIMULATE
*
* THE BARBER SHOP PROBLEM - THREE BARBERS, ONE TYPE OF CUSTOMER
* THIS MODEL MAINTAINS WAITING LINE STATISTICS
*
      GENERATE   6,4          CREATE CUSTOMER ARRIVALS
      QUEUE      1            CUSTOMERS QUEUE UP IF NECESSARY
      ENTER      1            ENGAGE AN AVAILABLE BARBER
      DEPART     1            CUSTOMER LEAVES THE QUEUE
      ADVANCE    15,3         CUSTOMER GETS HIS HAIR CUT
      LEAVE      1            FREE THE BARBER
      TERMINATE  0            LEAVE THE BARBER SHOP
*
      GENERATE   480          CREATE A TIMER AFTER 8 HOURS OF SIMULATED TIME
      TERMINATE  1            SHUT OFF THE RUN
    1 STORAGE    3
      START      1            CARRY OUT THE SIMULATION
      END                     RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                         SIGN-OFF
```

FIGURE 7-B: PROGRAM LISTING

(BARBER SHOP: FIFTH MODEL)

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	3	2.174	.724	71	14.704	1	3

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
1	2	.089	71	57	80.2	.605	3.071

```
     CPU TIME USED:

ASSEMBLY:       .293 SECONDS

EXECUTION:      .465 SECONDS
```

```
**** ON AT  04:31.31
**** OFF AT 04:31.55
**** ELAPSED TIME        23.813   SEC.
**** CPU TIME USED         4.75   SEC.
**** STORAGE USED        161.19   PAGE-SEC.
**** CARDS READ           23
**** LINES PRINTED
**** PAGES PRINTED
**** CARDS PUNCHED         1
**** DRUM READS            1
**** APPROX. COST OF THIS RUN     $.43
```

FIGURE 7-C: SELECTED OUTPUT; TIME AND COST DATA

(BARBER SHOP: FIFTH MODEL)

27. A Spare Parts Problem

Statement of Problem

A machine contains a part which is subject to periodic failure. Whenever the part fails, the machine must be turned off, then the failed part is removed, a good spare part is installed (if available), and the machine is turned on again. Failed parts can be repaired on the premises and can then be re-used.

The lifetime of a part has been observed to be 100±75 hours, uniformly distributed. Time required to remove a failed part is negligible, but 2±1 hours elapse while a spare part is being put into the machine. 100±50 hours elapse before failed parts have been fixed and are available for re-use. Assume that the capacity for repairing failed parts is unlimited.

For the following interpretation of a Transaction, model this situation with a block diagram, then run the corresponding GPSS program and determine the number of parts required for the machine to be operational at least 92% of the time.

Interpret a Transaction as a part. The number of Transactions circulating in the model will then equal the number of parts available to the system. The particular location of a Transaction in the model indicates the status of the part, i.e. a good part operational in the machine, a failed part being repaired, or a good part which is "on the shelf".

Approach Taken in Building the Model

The model can be built as a single sequence of blocks. The order of the blocks is consistent with the time sequence of events in the real situation. Good parts on the shelf are represented by Transactions in a Queue. A Facility is used (somewhat artificially) as a "logic switch" to permit movement of a good part into the installation phase only when this is logically possible, that is, when the machine is "down" and therefore can accept another part. Another Facility is used to represent the machine itself. The utilization of this latter Facility then represents the fractional "up time" of the machine. Note that if installation time were negligible, only one Facility would be required in this approach.

Table of Definitions

Time Unit: 1 Hour

GPSS Entity	Interpretation
Transaction	A Part
Facility 1	The machine When the Facility is engaged the machine is operational
Facility 2	The "logic switch" When the Facility is engaged a spare part is not needed
Queue 1	The pool of good parts ready for use

Other Model Documentation

Figure	Information Exhibited
8-A	Block Diagram
8-B	Program Listing
8-C	Selected Program Output; Time and Cost Data

Discussion of the Block Diagram

Two parts are introduced to the model shown by using the Limit Count feature of the GENERATE block. It is assumed that at time zero the machine is down and cannot be turned on until installation of a part has taken place. As has been the case so far in these notes, a separate Timer Segment is used. Notice that the GENERATED block has been given the symbolic name KNTRL (for "control"). This has been done to facilitate running the model for several configurations (total parts in the model) with a single submit. Hence, as indicated in the block diagram, after the first START card, the CLEAR card is used to zero-out model statistics and remove the two Transactions from wherever they may be in the model. Then the block named KNTRL is re-defined as a GENERATE block with a Limit Count of 3. The next START card then instructs the simulator to run the model again. The second run is of course made with a model having three parts available to it. This means that slightly modified versions of the same basic model are being studied on a sequential basis with a single submit.

Discussion of the Program Output

The "Answer" to the problem posed is available in the output via Facility 1 utilization statistics. It is the model itself which is of interest, of course, for our purposes. In any event, based on the limited run shown, the machine is "up" about 98% of the time even when only two parts are available to it.

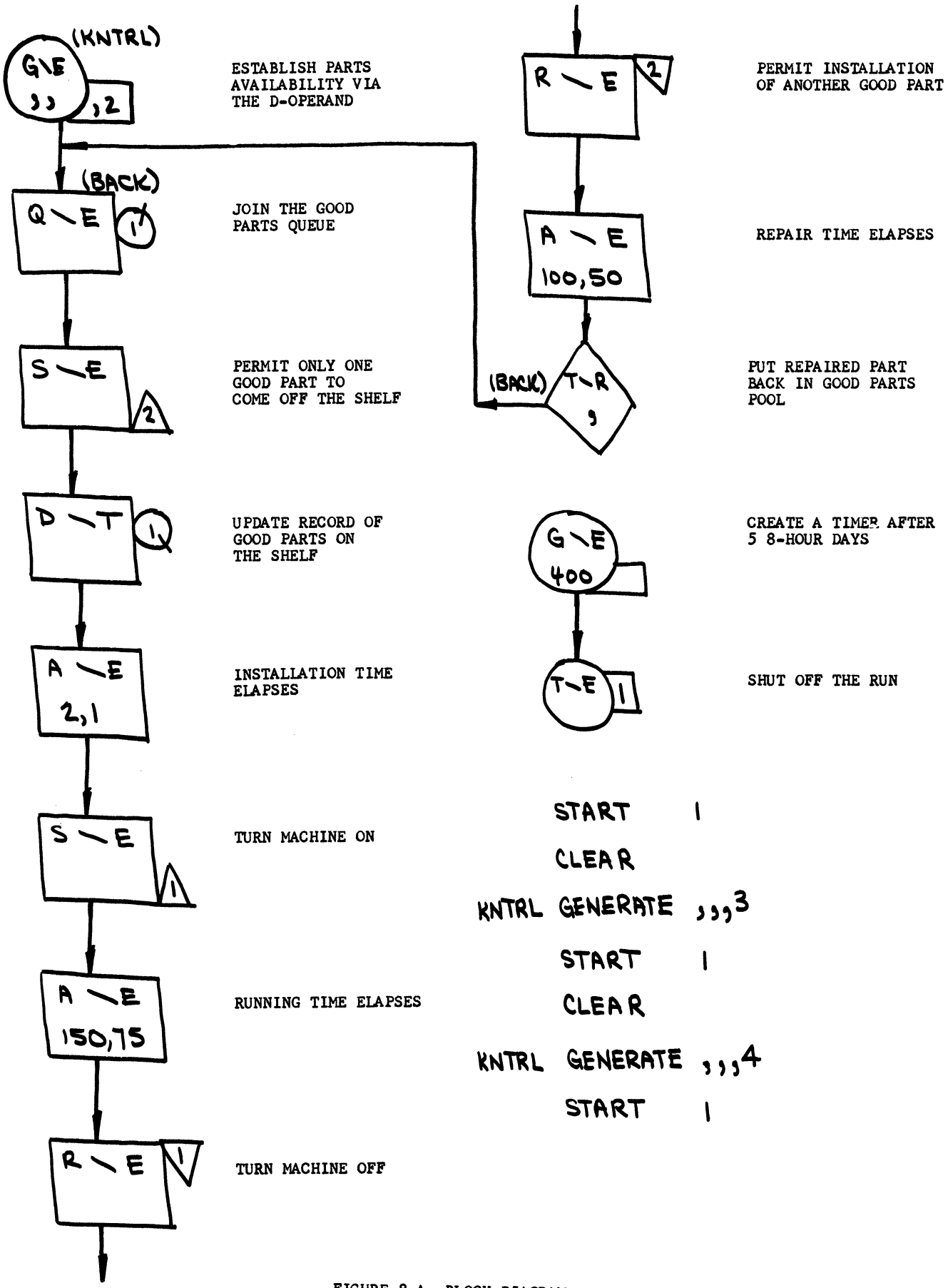


FIGURE 8-A: BLOCK DIAGRAM
(A SPARE PARTS PROBLEM)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
      SIMULATE
*
* THE SPARE PARTS PROBLEM, WHERE A TRANSACTION IS A PART
*
 KNTRL GENERATE   ,,,2       THE FIRST MODEL HAS TWO PARTS AVAILABLE
 BACK  QUEUE      1          JOIN THE GOOD PARTS QUEUE
       SEIZE      2          KEEP OTHER SPARES ON THE SHELF FOR NOW
       DEPART     1          LEAVE THE GOOD PARTS QUEUE
       ADVANCE    2,1        INSTALLATION TIME ELAPSES
       SEIZE      1          TURN MACHINE ON
       ADVANCE    150,75     RUNNING TIME ELAPSES
       RELEASE    1          TURN MACHINE OFF
       RELEASE    2          IT IS NOW FEASIBLE TO INSTALL ANOTHER PART
       ADVANCE    100,50     REPAIR TIME ELAPSES
       TRANSFER   ,BACK      PUT REPAIRED PART BACK ON THE SHELF
*
       GENERATE   400        CREATE A TIMER AFTER 5 8-HOUR DAYS
       TERMINATE  1          SHUT OFF THE RUN
       START      1          CARRY OUT THE SIMULATION
*
       CLEAR                 COMPLETELY RESTORE THE PRECEDING MODEL
 KNTRL GENERATE   ,,,3       MODIFY THE MODEL FOR THE CASE OF 3 PARTS
       START      1          CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
*
       CLEAR                 COMPLETELY RESTORE THE PRECEDING MODEL
 KNTRL GENERATE   ,,,4       MODIFY THE MODEL FOR THE CASE OF 4 PARTS
       START      1          CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
       END
$SIGH
```

FIGURE 8-B: PROGRAM LISTING
(A SPARE PARTS PROBLEM)

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.979	3	130.666	1	
2	.997	3	133.000	1	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
1	1	.409	3	1	33.3	54.666	82.000

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.979	4	98.000	1	
2	.997	4	99.750	1	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
1	2	1.389	5	1	19.9	111.199	139.000

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.984	3	131.333	4	
2	.997	3	133.000	4	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS
1	3	2.557	6	1	16.6	170.500	204.599

CPU TIME USED:

ASSEMBLY: .462 SECONDS

EXECUTION: .593 SECONDS

**** ON AT 04:32.07
**** OFF AT 04:32.33
**** ELAPSED TIME 26.28 SEC.
**** CPU TIME USED 5.084 SEC.
**** STORAGE USED 176.866 PAGE-SEC.
**** CARDS READ 33
**** LINES PRINTED
**** PAGES PRINTED
**** CARDS PUNCHED 1
**** DRUM READS 10
**** APPROX. COST OF THIS RUN $.47

FIGURE 8-C: SELECTED OUTPUT; TIME AND COST DATA

(A SPARE PARTS PROBLEM)

28. The Spare Parts Problem Revisited

Statement of the Problem

Build another model for the spare parts problem of Section 27, this time interpreting a Transaction as a "workman" who sees to it that appropriate action is taken in the model when and as required. This approach raises a question as to how the total number of parts in the model should be represented, and how a distinction can be made between good and failed parts. Suppose that a Storage is used to represent this information. Then the Storage capacity can be interpreted as synonymous with the number of parts in the model, with Rj taken as the number of good parts "on the shelf".

Approach Taken in Building the Model

A workman is created, takes a part off the shelf, installs it, and turns the machine on. When the part fails, the workman turns the machine off, then sends a co-worker back to take the next part off the shelf (if possible) and have it installed, etc. Meantime the original workman sees to it that the failed part just removed is repaired. The repaired part is then put back on the shelf. The workman then is removed from the model, leaving the co-worker to carry on.

Table of Definitions

Time Unit: 1 Hour

GPSS Entity	Interpretation
Transaction	A Workman
Facility 1	The machine
	When the Facility is "engaged," the machine is operating.
Storage 1	Storage 1 is the storage representing the parts situation.

Other Model Documentation

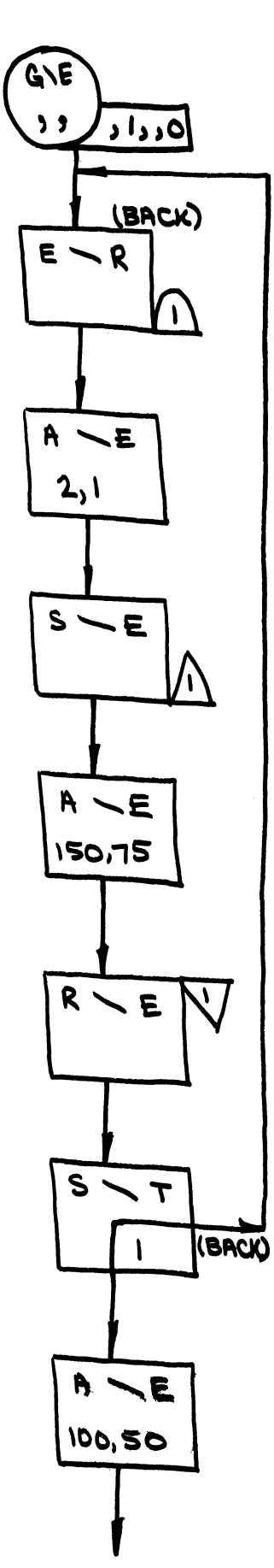
Figure	Information Exhibited
9-A	Block Diagram
9-B	Program Listing
9-C	Selected Program Output; Time and Cost Data

Discussion of the Block Diagram

Figure 9-A shows the Control Card sequence with which the situation can be modeled for the cases of 2, 3, and 4 parts with a single submit. The Block Diagram itself is very straightforward, corresponding exactly to the model description given above.

Discussion of the Program Output

Results are identical with those of Figure 8-C. Note, for example, that Facility 1 average utilizations and average Time/Transaction for the three situations studied are exactly the same in Figures 9-C and 8-C. Can you explain this in terms of your understanding of pseudo-random numbers in conjunction with a comparison of the Block Diagrams (Figures 8-A and 9-A)?



CREATE A WORKMAN TO START THE SIMULATION

TAKE A SPARE PART OFF THE SHELF

INSTALLATION TIME ELAPSES

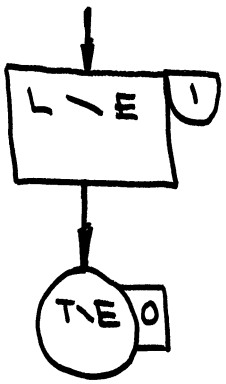
TURN THE MACHINE ON

RUNNING TIME ELAPSES

TURN THE MACHINE OFF

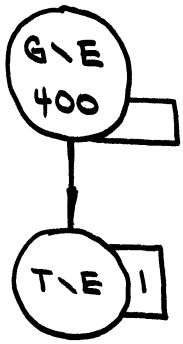
ARRANGE FOR INSTALLATION OF ANOTHER SPARE

REPAIR TIME ELAPSES



SET RECORD INDICATING REPAIRED PART IS BACK ON THE SHELF

REMOVE THIS WATCHMAN FROM THE MODEL



CREATE A TIMER AFTER 5 8-HOUR DAYS

SHUT OFF THE RUN

1	STORAGE	2	
	START	1	
	CLEAR		
1	STORAGE	3	
	START	1	
	CLEAR		
1	STORAGE	4	
	START	1	

FIGURE 9-A: BLOCK DIAGRAM
(THE SPARE PARTS PROBLEM REVISITED)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
        SIMULATE
*
* THE SPARE PARTS PROBLEM, WHERE A TRANSACTION IS A WORKMAN
*
        GENERATE    ,,,1,,0     CREATE A WORKMAN TO START THE RUN
 BACK   ENTER       1           TAKE A SPARE PART OFF THE SHELF
        ADVANCE     2,1         INSTALLATION TIME ELAPSES
        SEIZE       1           TURN MACHINE ON
        ADVANCE     150,75      RUNNING TIME ELAPSES
        RELEASE     1           TURN MACHINE OFF
        SPLIT       1,BACK      SEND ANOTHER WATCHMAN TO CHECK FOR SPARES
        ADVANCE     100,50      REPAIR TIME ELAPSES
        LEAVE       1           PUT REPAIRED PART BACK ON THE SHELF
        TERMINATE   0           REMOVE THIS WATCHMAN FROM THE MODEL
*
        GENERATE    400         CREATE A TIMER AFTER 5 8-HOUR DAYS
        TERMINATE   1           SHUT OFF THE RUN
      1 STORAGE     2           THE FIRST MODEL HAS TWO PARTS AVAILABLE
        START       1           CARRY OUT THE SIMULATION
*
        CLEAR                   COMPLETELY RESTORE THE PRECEDING MODEL
      1 STORAGE     3           MODIFY THE MODEL FOR THE CASE OF 3 PARTS
        START       1           CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
*
        CLEAR                   COMPLETELY RESTORE THE PRECEDING MODEL
      1 STORAGE     4           MODIFY THE MODEL FOR THE CASE OF 4 PARTS
        START       1           CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
        END
$SIGH
```

FIGURE 9-B: PROGRAM LISTING

(THE SPARE PARTS PROBLEM REVISITED)

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.979	3	130.666	1	

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	2	1.584	.792	3	211.333	2	2

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.979	4	98.000	3	

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	3	1.602	.534	4	160.250	2	3

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.984	3	131.333	4	

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	4	1.432	.358	3	191.000	1	3

CPU TIME USED:

ASSEMBLY: .466 SECONDS

EXECUTION: .609 SECONDS

```
**** ON AT  04:32.38
**** OFF AT 04:33.00
**** ELAPSED TIME        22.46   SEC.
**** CPU TIME USED        5.102  SEC.
**** STORAGE USED       177.59   PAGE-SEC.
**** CARDS READ          33
**** LINES PRINTED
**** PAGES PRINTED
**** CARDS PUNCHED       1
**** DRUM READS          44
**** APPROX. COST OF THIS RUN    $.47
```

FIGURE 9-C: SELECTED OUTPUT; TIME AND COST DATA
(THE SPARE PARTS PROBLEM REVISITED)

29. Modification of a Transaction's Priority Level

***We have already seen that a Transaction's Priority Level is given a value when the Transaction is created at a GENERATE Block or a SPLIT Block.

***A Transaction's PR value can also be modified by causing the Transaction to move through a Block known as the PRIORITY Block.

The PRIORITY Block:

Purpose: Modify the PR value of each Transaction passing through the Block.

Shape and Operands:

Operand	Significance
A	Specifies the Priority Level which the Transaction is to be given.
B	Optional operand; if used, the B operand consists of the word BUFFER.

Nature of Operation:

B operand not used: When a Transaction enters the PRIORITY Block its PR value is given the value of the A operand, the simulator then immediately attempts to move the Transaction to the next sequential block.

B operand used: Again, the PR value of an entering Transaction is given the value of the A operand. The simulator does not then immediately attempt to move the Transaction to the next sequential block. Rather, the Transaction is put into the last position in its priority class on the Current Events Chain. The simulator then re-starts its scan of the Current Events Chain at the front of the chain.

Note: Remember that the Current Events Chain is "only" used by the simulator for bookkeeping. Hence the Transaction is still "at" the PRIORITY Block, even though for bookkeeping purposes its position on the Current Events Chain may have changed.

Figure 10 is a simple Block Diagram example designed to shed some light on this BUFFER option. In Figure 10-A, the second Transaction to SEIZE Facility 1 will be the Transaction coming from the GENERATE Block labeled ONE. This occurs despite the facts that 1) At the time of the second SEIZE, the seizing Transaction has a PR value of 1, and 2) A higher-prioritied Transaction (PR=2)

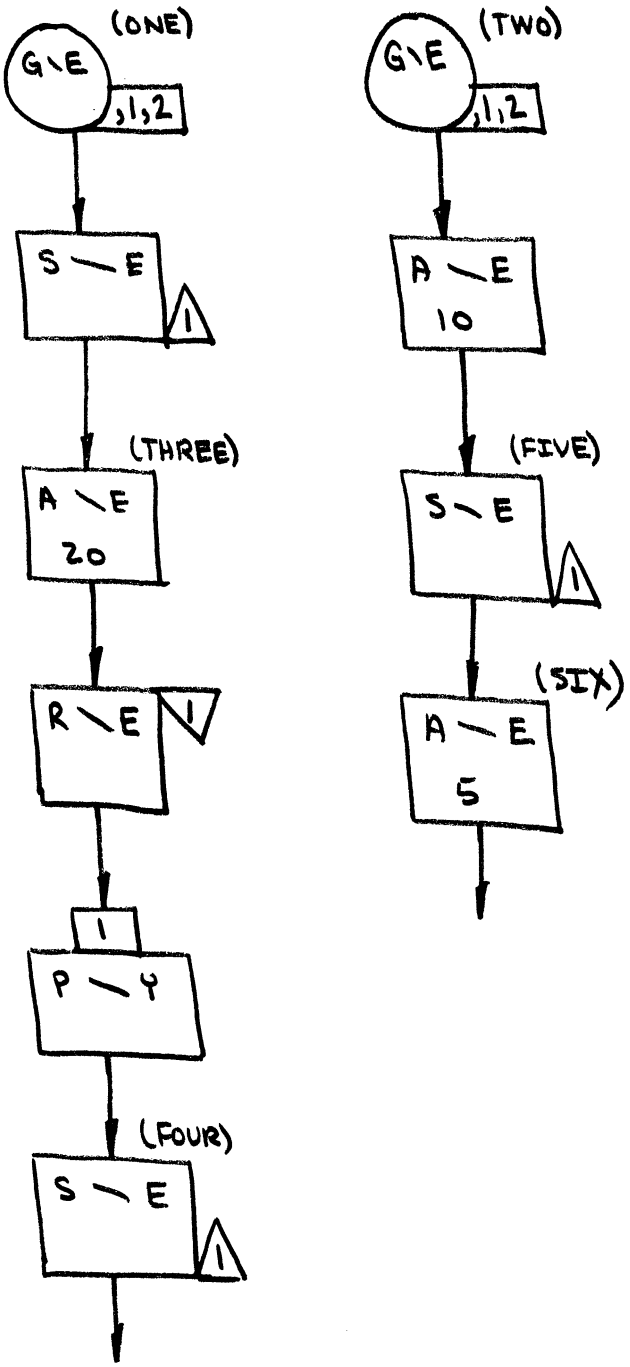


FIGURE 10-A:
MISLEADING USE OF THE PRIORITY BLOCK

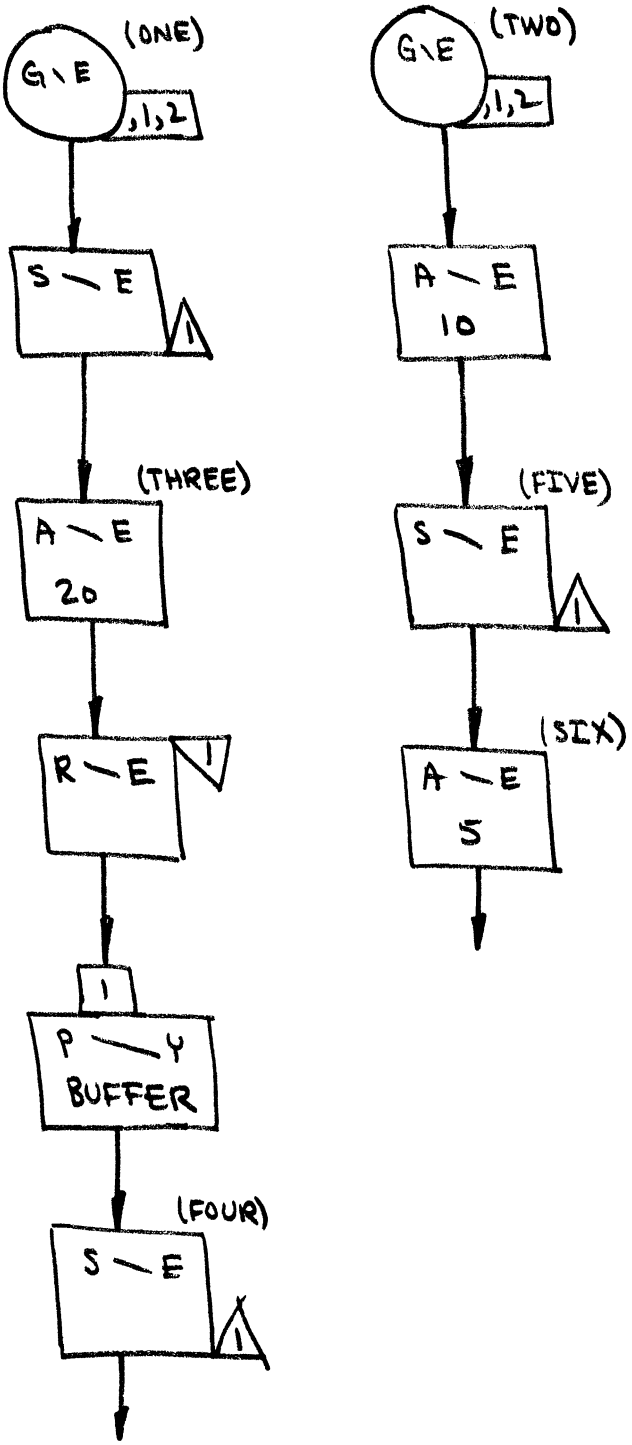


FIGURE 10-B:
CORRECT USE OF THE PRIORITY BLOCK IN THE SAME CONTEXT

has been waiting 10 time units to SEIZE Facility 1 itself. This result occurs because the simulator eventually carries out these steps:

1) A Transaction exits the ADVANCE Block THREE.

2) It moves into the RELEASE Block where the Facility is RELEASED.

3) It's PR is changed from 2 to 1 at the PRIORITY Block.

4) The Transaction then moves into the SEIZE Block FOUR, SEIZE-ing the Facility it had just released.

In short, the "other Transaction" was never given a chance to SEIZE Facility 1 between the time it was RELEASED and then re-SEIZED.

In contrast, in Figure 10-B the Transaction from GENERATE Block TWO is the second Transaction to SEIZE Facility 1, per this eventual simulator sequence:

1) A Transaction exits the ADVANCE Block THREE.

2) It moves into the RELEASE Block where the Facility is RELEASED.

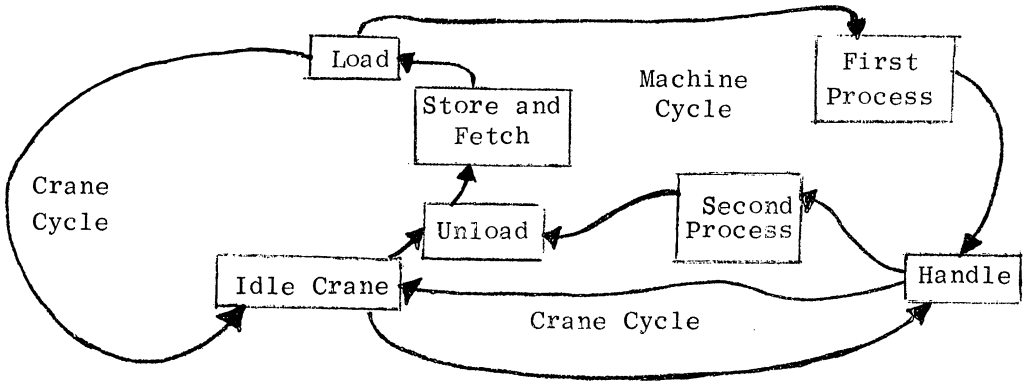
3) It's PR is changed from 2 to 1 at the PRIORITY Block.

4) The simulator then re-starts the scan at the front of the Current Events Chain. There it finds the "other Transaction", and moves it forward to its next scheduled Block (the SEIZE Block FIVE), where it SEIZES the now-available Facility 1, then moves into the ADVANCE Block SIX.

5) As the scan progresses to the back of the Current Events Chain, the simulator encounters the Transaction at the PRIORITY Block and attempts to move it forward in the model. The SEIZE Block refuses entry, however. Hence the Transaction must remain at the PRIORITY Block "for the time being."

30. An Equipment Balancing Problem

Consider a machine shop which machines castings produced in a foundry. Several machines are served by one overhead crane which fetches the castings (one at a time) from the foundry, helps in the required intermediate handling of each casting part way through the machining process, and also takes finished pieces (one at a time) to a warehouse. This process can be illustrated for the case of one machine in this manner:



Assume that the times required for the various operations are uniformly distributed according to this table:

Operation	Time, Minutes
Load	10 ± 6
First Process	80 + 20
Handle	15 + 7
Second Process	110 + 30
Unload	20 + 5
Store and Fetch	20 + 10

In the event of competition for the crane, the priority followed is that handling takes precedence over unloading. Per the diagram shown above, the crane always reloads a machine as its next job after unloading it.

Model this system in GPSS in such a way that the number of machines serviced by a single crane can be readily varied. Then run the model to determine a) the percent idle time of the crane, and b) the percent idle time of the machines considered as a group both as a function of the number of machines serviced per crane.

Approach Taken in Building the Model

A Transaction is interpreted as a casting in the process of being finished. There is, then, one casting in motion for each machine in the machine group using the services of a single crane. The casting goes through the basic cycle from the non-machined phase to the finished phase. Then it is cycled back (in the model), where it once again plays the role of an unfinished casting. The priority with which the casting claims the crane depends upon its in-process stage, per the problem statement. It is a simple matter to modify the priority level of the corresponding Transaction so that the "first-come, first-served within priority class" feature of GPSS can be used.

Table of Definitions

Time Unit: 1 Hour

GPSS Entity	Interpretation
Transaction	Main Segment: An In-process Casting
	Timer Segment: A Timer
Facility 1	The Crane
	When the Facility is "engaged", the crane is being used.
Storage 1	The group of machines using the crane
	Each engaged unit of capacity corresponds to an idle machine.

Other Model Documentation

Figure	Information Exhibited
11-A	Block Diagram
11-B	Program Listing
11-C	Selected Program Output; Time and Cost Data

Discussion of Block Diagram

Consistent with the approach taken in building the model, the block diagram closely follows the chronology presented in the problem statement. It is assumed that at time zero, one casting has been fetched from the warehouse for each machine. Hence the process starts at the loading phase. Priority levels of 0 and 1 have been used to distinguish between the two priorities that apply in the problem. The number of machines serviced by a single crane is varied by the combined action of 1) changing the Limit Count on the GENERATE block, symbolically named KNTRL, and 2) correspondingly changing the capacity of the Stroage representing the group of machines using the crane. Referring to the example in Figure 10, what might happen if the Priority Block's B operand "BUFFER" were eliminated as part of the model?

Discussion of Program Output

Output for configurations of 1, 2, and 3 machines per crane is shown. As expected, machine utilization (the complement of Storage 1 utilization) decreases as crane utilization (Facility 1 utilization) increases. Machine utilization is still 91%, however, even for the case of three machines per crane. If cost figures were included as part of this problem (hourly cost of an idle crane and of an idle machine), then the machine/crane ratio minimizing cost due to idleness could be determined with this model. The cost implications could be considered by desk calculation with computer output in hand, or they could be built in as part of the model. Possibly the "best" model would be one which would use programmer--supplied logic to search for the optimum machine/crane ratio.

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
       SIMULATE
*
* THE EQUIPMENT BALANCING PROBLEM
*
 KNTRL GENERATE    ,,,1        CREATE ONE TRANSACTION FOR EACH MACHINE
       ENTER       1           MACHINE IS IDLE UNTIL CRANE BECOMES AVAILABLE
       SEIZE       1           OBTAIN CRANE TO LOAD MACHINE
       LEAVE       1           MACHINE IS NO LONGER IDLE
 LOAD  ADVANCE     10,6        LOADING TIME ELAPSES
       RELEASE     1           RELEASE THE CRANE
       ADVANCE     80,20       'FIRST PROCESS' TIME ELAPSES
       PRIORITY    1           SET HIGH PRIORITY FOR IMMINENT HANDLING
       ENTER       1           MACHINE IS IDLE UNTIL CRANE BECOMES AVAILABLE
       SEIZE       1           OBTAIN CRANE FOR HANDLING
       LEAVE       1           MACHINE IS NO LONGER IDLE
       ADVANCE     15,7        HANDLING TIME ELAPSES
       RELEASE     1           RELEASE THE CRANE
       ADVANCE     110,30      'SECOND PROCESS' TIME ELAPSES
       PRIORITY    0,BUFFER    SET LOW PRIORITY FOR IMMINENT CRANE NEED
       ENTER       1           MACHINE IS IDLE UNTIL CRANE BECOMES AVAILABLE
       SEIZE       1           OBTAIN CRANE FOR UNLOAD-STORE-LOAD CYCLE
       LEAVE       1           MACHINE IS NO LONGER IDLE
       ADVANCE     20,5        UNLOADING TIME ELAPSES
       ADVANCE     20,10       STORE AND FETCH TIME ELAPSES
       TRANSFER    ,LOAD       TIE BACK IN AT THE LOADING TIME BLOCK
*
       GENERATE    2550        CREATE A TIMER AFTER ABOUT 100 CASTINGS/MACHINE
       TERMINATE   1           SHUT OFF THE RUN
     1 STORAGE     1           ONE MACHINE PER CRANE
       START       1           CARRY OUT THE SIMULATION
*
       CLEAR                   COMPLETELY RESTORE THE PRECEDING MODEL
 KNTRL GENERATE    ,,,2        MODIFY MODEL FOR THE CASE OF 2 MACHINES PER CRANE
     1 STORAGE     2           TWO MACHINES PER CRANE
       START       1           CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
*
       CLEAR                   COMPLETELY RESTORE THE PRECEDING MODEL
 KNTRL GENERATE    ,,,3        MODIFY MODEL FOR THE CASE OF 3 MACHINES PER CRANE
     1 STORAGE     3           THREE MACHINES PER CRANE
       START       1           CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
       END                     RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                          SIGN-OFF
```

FIGURE 11-B: PROGRAM LISTING
(EQUIPMENT BALANCING PROBLEM)

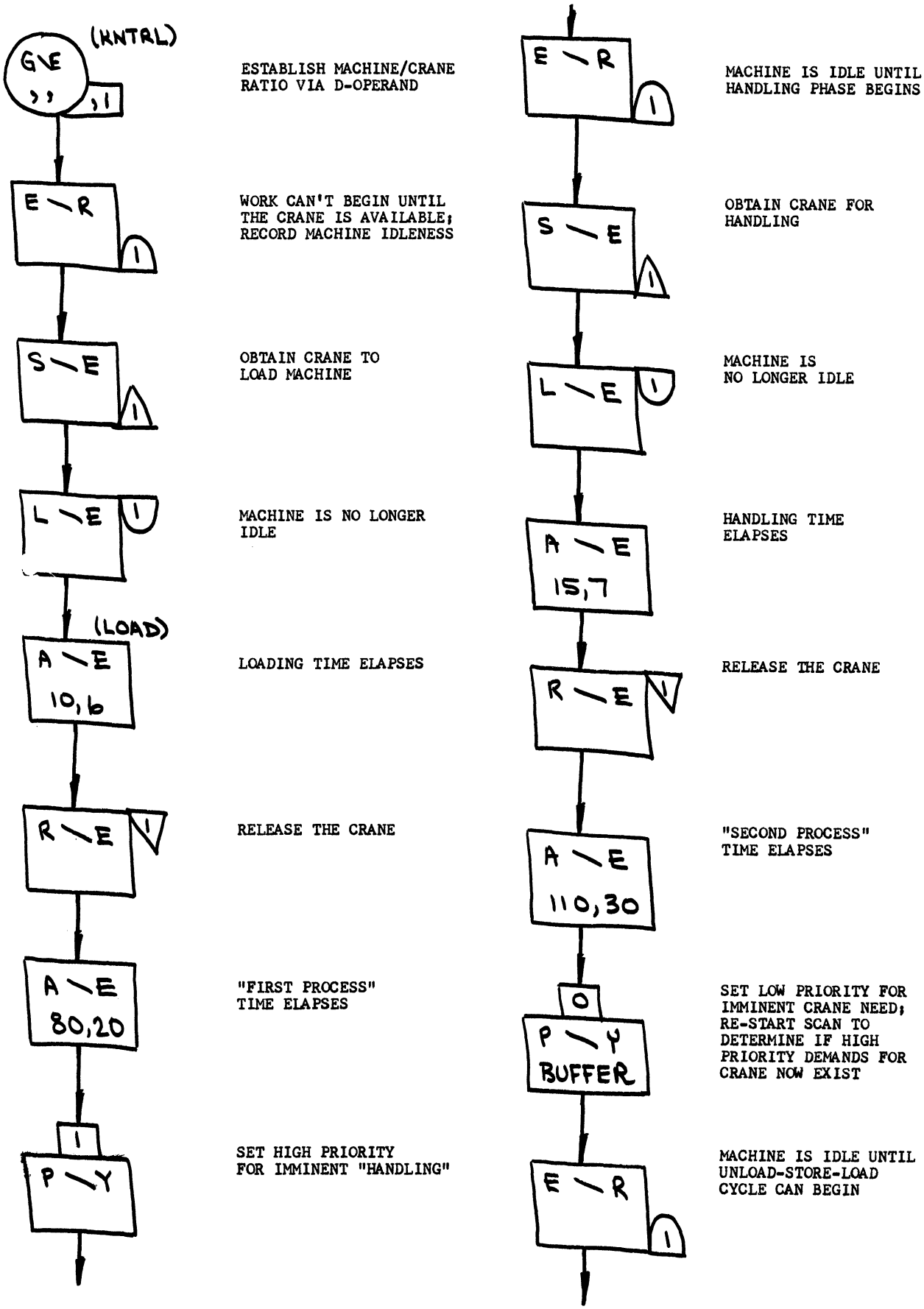
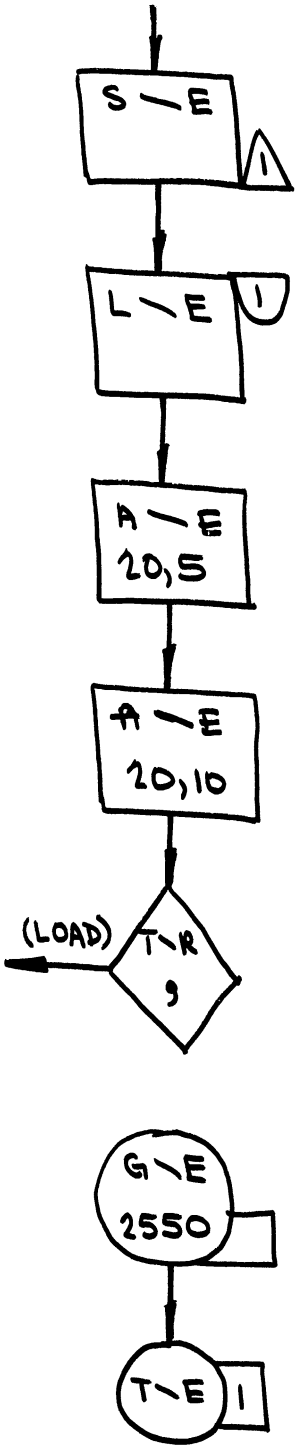


FIGURE 11-A: BLOCK DIAGRAM (CONTINUED ON NEXT PAGE)

(EQUIPMENT BALANCING PROBLEM)



OBTAIN CRANE FOR UNLOAD-STORE-LOAD CYCLE

1 STORAGE 1

START 1

CLEAR

MACHINE IS NO LONGER IDLE

KNTRL GENERATE ,,,2

1 STORAGE 2

START 1

UNLOADING TIME ELAPSES

CLEAR

KNTRL GENERATE ,,,3

STORE AND FETCH TIME ELAPSES

1 STORAGE 3

START 1

TIE BACK IN AT THE LOADING TIME BLOCK

CREATE A TIMER AFTER 2550 WORKING HOURS ELAPSE

TURN OFF THE RUN

FIGURE 11-A: BLOCK DIAGRAM (CONTINUED FROM PRECEDING PAGE)
(EQUIPMENT BALANCING PROBLEM)

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.263	21	31.952		

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	1	.000	.000	21	.000		1

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.476	38	32.000		

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	2	.069	.034	38	4.684		1

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.685	56	31.232	1	

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	3	.271	.090	56	12.357		2

CPU TIME USED:

ASSEMBLY: .655 SECONDS

EXECUTION: 1.075 SECONDS

```
**** ON AT  01:40.12
**** OFF AT 01:40.42
**** ELAPSED TIME        30.356  SEC.
**** CPU TIME USED        5.911  SEC.
**** STORAGE USED       212.376  PAGE-SEC.
**** CARDS READ          46
**** LINES PRINTED       216
**** PAGES PRINTED       20
**** CARDS PUNCHED       1
**** DRUM READS
**** APPROX. COST OF THIS RUN    $.68
```

FIGURE 11-C: SELECTED OUTPUT; TIME AND COST DATA
(EQUIPMENT BALANCING PROBLEM)

31. Addressing of Data in GPSS

***The following table summarizes several schemes for addressing data in terms of the information supplied in the model, the relationship of the information to the data, and descriptive terminology which is sometimes used:

Information Supplied	General Terminology	GPSS Terminology
The Data Itself	Zero-level Addressing	Direct Specification
Address of the Data	First-level Addressing	Indirect Specification
Address of the Address of the Data	Second-level Addressing	Indirect Addressing

***Direct Specification

The data to be used is provided directly; it is not necessary to "go to a storage location" to fetch the data.

***Indirect Specification

The data is the value of some "SNAj", where "SNA" is any of the Standard Numerical Attributes (such as discussed for Facilities, Queues, Storages, and Transactions and summarized on pp. 28-31, IUM) and "j" is, as usual, the index number of the Standard Numerical Attribute.

For example:

Name of Variable	Value of Variable
Q3	The current contents of Queue 3
S1	The current contents of Storage 1
FR10	The utilization of Facility 10, in parts per thousand

Note that the value of j itself is directly specified.

*** Indirect Addressing

The data itself will again be the value of "SNAj", but the value of j is indirectly specified. In particular, in indirect addressing the value of a Transaction's Parameter is always interpreted as the value of j.

Notation Used

An asterisk (*) is placed between the Standard Numerical Attribute and the index number of the Parameter whose value is interpreted as j.

For example:

Name of Variable	Value of Variable
Q*8	The current contents of the Queue whose number is the value of Parameter 8. [1]
P*7	The value of the Parameter whose number is the value of Parameter 7

[1] A natural question is: Parameter 8 of which Transaction. (Recall that there may be many Transactions in a model, each with its own P8 value.) Note that data is only referenced when a Transaction enters a block and "causes the system to be updated." With a single exception (involving the UNLINK block, not yet discussed), whenever reference is made to "the value of Parameter j", it is the Transaction which "causes the updating" whose j-th Parameter value will be used.

***Further Examples

Indirect Specification:

Indirect Addressing:

The example of indirect specification shown above indicates how a Transaction can itself carry information as to which Queue it is to QUEUE up in, which Facility it is to SEIZE, which Storage it is to ENTER, or whatever. It is clear that this capability permits great flexibility in the modeling process.

Suppose that a Transaction with a P6 value of 3 moves into the LEAVE block shown as an example of indirect addressing. What will be the value of S3 when the Transaction exits the block?

32. Modification of Transaction Parameter Values

***A Transaction can have the value of one of its Parameters modified by causing it to move through a block known as the ASSIGN block.

The ASSIGN Block:

Purpose: Modify the value of one Parameter of each Transaction passing through the block

Shape and Operands:

Operand	Significance
A	Specifies the number of the Parameter whose value is to be modified.
B	Specifies the value to be used in the modification process.
C	Optional operand; if used, the C operand specifies the number of a Function. The Function will be used to modify the B operands value multiplicatively to produce a number which, in turn, is used to modify the Parameter value.

***Modes of Parameter Modification

The ASSIGN block can carry out Parameter modification in any one of three possible modes:

Modification Mode	How Specified
Increment Mode	Last Character of the A operand is a +
Decrement Mode	Last Character of the A operand is a -
Replacement Mode	No Special Last Character is Provided as Part of the A operand

33. Savevalue Concept in GPSS

***A particular Transaction's Parameter values can be referenced only when that Transaction enters a block, thereby triggering an activity which happens to require making reference to one or more Pj values.

***It is impossible for one Transaction to trigger an activity which requires a direct reference to some other Transaction's Pj values. Hence the use of Pj values is relatively restricted.

***There is, however, in addition to Standard Numerical Attributes and Logical Attributes, a series of storage locations whose contents can be referenced from any point in a model. These storage locations are known as Savevalues.

***Savevalues may consist of 2 bytes (16 bits; i.e. halfword locations) or 4 bytes (32 bits; i.e. fullword locations). The values taken on by Savevalues are signed integers. (See page 5 in these notes for the range of decimal values that can be represented with either 16 or 32 bits, making allowance for algebraic sign.)

***GPSS automatically initializes the contents of Savevalues as zero (unless directed otherwise by the programmer).

***Two types of Savevalues are available:

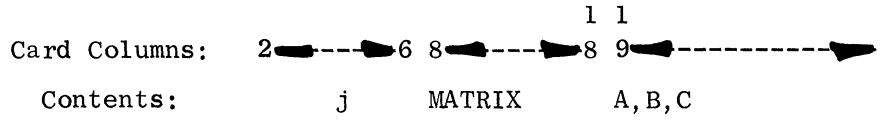
"Vector" Savevalues (called SAVEVALUEs), where
Xj is the mnemonic for fullword SAVEVALUE number j, and
XHj is the mnemonic for halfword SAVEVALUE number j.

Note: No "dimensioning" is required for "vector" type Savevalues.

Matrix Savevalues (called MSAVEVALUEs), where
MXj(a,b) is the mnemonic for the fullword storage location occupying row a and column b in MSAVEVALUE number j, and

MHj(a,b) is the mnemonic for the halfword storage location occupying row a and column b in MSAVEVALUE number j.

Note: Dimensioning is required for matrix Savevalues. This is accomplished by use of the MATRIX definition card:



where:

j is the number of the matrix Savevalue being dimensioned,

A is X or H, depending on whether a fullword or halfword MSAVEVALUE is being dimensioned, respectively,

B is the number of rows in the matrix, and

C is the number of columns in the matrix

***The number of different "vector" and matrix Savevalues available depends on the amount of computer storage.

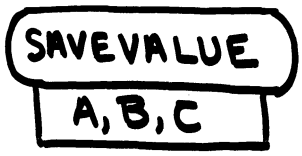
34. Modification of SAVEVALUE and MSAVEVALUE Values

***A Transaction can cause the value of a Savevalue to be modified by causing it to move through a SAVEVALUE block or a MSAVEVALUE block.

The SAVEVALUE Block:

Purpose: Modify the value of one SAVEVALUE when a Transaction passes through the block

Shape and Operands:

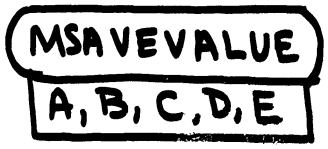


Operand	Significance
A	Specifies the number of the SAVEVALUE whose value is to be modified.
B	Specifies the value to be used in the modification process.
C	Optional operand; the C operand must be H if a halfword SAVEVALUE is being modified, otherwise it is not used.

The MSAVEVALUE Block:

Purpose: Modify the value of one MSAVEVALUE when a Transaction passes through the block

Shape and Operands:



Operand	Significance
A	Specifies the number of the MSAVEVALUE whose value is to be modified.
B	Specifies the matrix row in which the MSAVEVALUE involved is located.
C	Specifies the matrix column in which the MSAVEVALUE involved is located.
D	Specifies the value to be used in the modification process.
E	Optional operand; The E operand must be H if a halfword MSAVEVALUE is being modified; otherwise it is not used.

***Modes of Savevalue Modification

The SAVEVALUE and MSAVEVALUE blocks can carry out Savevalue modification in any one of three possible modes. The options here and their manner of specification are identical to those for Parameter modification.

35. Initialization of SAVEVALUE and MSAVEVALUE Values

***SAVEVALUEs can be assigned initial values by use of the INITIAL card, laid out in this format:

		1 1
Card Columns:	8 ------ 8	9 ----------
Contents:	INITIAL	A,B

where:

A is Xj or XHj, for fullword or halfword SAVEVALUEs, respectively, and j is the number of the SAVEVALUE being initialized;

B is the initial value.

***MSAVEVALUEs can be assigned initial values by use of the INITIAL card, laid out exactly as above, where the A operand is expressed in this manner:

A is MXj(k,n) or MHj(k,n), for fullword or halfword MSAVEVALUEs, respectively, where j is the number of the MSAVEVALUE being initialized, and k and n are the row and column numbers, respectively, of the matrix element involved.

***Multiple Initializations with One INITIAL Card

Rather than using a separate INITIAL card for each Savevalue being initialized, this format may be followed:

		1 1
Card Columns:	8 ------ 8	9 ----------
Contents:	INITIAL	A,B/A,B/A,B/ ... /A,B

where:

A is Xj, or XHj, or MXj(k,n), or MHj(k,n), and

B is the corresponding initial value.

***Initialization of Consecutive Savevalues with a Common Value

It is possible to achieve some economy of representation when consecutive Savevalues are to be initialized with a common value. For example:

```
INITIAL     X3-9,4
INITIAL     MH1-3(2,5),7/MH2(1-2,2),-5
```

are legal in GPSS. Details concerning these possibilities can be found on pages 123-125 in the User's Manual.

36. Logic Switches in GPSS

*** Logic switches are characteristic of control elements in a real-world system (lock on a door; traffic light; "Go to Next Window" sign--or its absence--at a bank teller's window, etc.).

*** Logic switches can be in either one of two positions, known as "set" and "rest".

*** Analogous to Facilities, Storages, Queues, Savevalues, and Functions, Logic Switches are known by number: and the total number of them in a model depends on computer storage (Table 1, page 47, IUM).

*** GPSS automatically initializes Logic Switches in "reset" position (unless directed otherwise by the programmer).

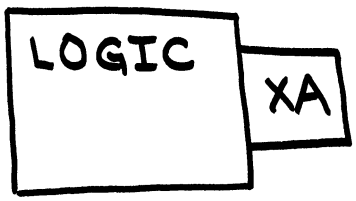
37. Modification of Logic Switch Settings

*** When a Transaction moves through a LOGIC Block, the setting of one Logic Switch is modified.

The LOGIC Block:

Purpose: Modify the setting of a Logic Switch each time a Transaction moves through the Block.

Shape and Operands:



Operand	Significance
X	X is one of the single letters R, S, or I, depending on whether the Logic Switch is to be Reset, Set, or Inverted (reversed), respectively. X is termed an auxiliary operator. It is punched in the Operation Field of the punchcard, in card column 14.
A	Specifies the number of the Logic Switch whose setting will be modified (if modification is necessary).

38. Initialization of Logic Switch Settings

*** Logic Switches can be initialized in the "set" position by use of the INITIAL card, laid out in this format:

	2 ------- 6	8 ------- 8	9 -------
card columns:		1	1
contents:		INITIAL	A/A/A/.../A

where:

A is LSj, and

j is the number of the Logic Switch which is to be initialized in the "set" position.

Example: INITIAL LS3/LS7/LS18

The example shows an INITIAL card which will cause Logic Switches 3, 7, and 18 to be initialized in the "set" position.

39. Logical Information Automatically Maintained by GPSS

*** During the course of a simulation, GPSS automatically maintains logical information concerning the status of Facilities, Storages, and Logic Switches used in the model.

*** Logical information is Boolean-valued information (a Boolean value can only be True or False).

*** Logical information is referenced in a model by specifying two items:

1) A mnemonic indication of the particular logical information being referenced, and

2) The index number of the particular Facility, Storage, or Logic Switch in question.

*** The various mnemonics that are available for Facilities, Storages, and Logic Switches are now listed:

Facility Logical Attributes

Mnemonic	Value
NU	True if the referenced Facility is not in use; False otherwise
U	True if the referenced Facility is in use; False otherwise

Storage Logical Attributes

Mnemonic	Value
SE	True if the referenced Storage is empty; False otherwise
SNE	True if the referenced Storage is not empty; False otherwise
SF	True if the referenced Storage is full; False otherwise
SNF	True if the referenced Stroage is not full; False otherwise

Logic Switch Logical Attributes

Mnemonic	Value
LR	True if the referenced Logic Switch is reset; False otherwise
LS	True if the referenced Logic Switch is set; False otherwise

*** The manner in which the index number of the particular Facility, Storage or Logic Switch in question is provided will be discussed in Sections 43 and 62, where the use of logical attributes in model building is illustrated.

40. Implicit Tests in the Path of Flow of Transactions

***We have already seen that two tests relative to the flow of Transactions are conducted automatically in GPSS

***These tests may be viewed as "implicit tests", since in a sense they do not have to be explicitly requested by the programmer

***The two implicit tests are:

1) The testing of Fj by a Transaction attempting to gain entry to a SEIZE Block.

2) The testing of Rj by a Transaction attempting to gain entry to an ENTER Block.

41. Explicit Tests in the Path of Flow of Transactions

***GPSS makes available to the programmer a variety of tests which can be explicitly introduced into models whenever desired.

***These tests may be put into two categories on the basis of the information used in the testing process:

1) Numerical Information

   -- Tests are conducted with the TEST Block

   -- Tests involve a comparison of two Standard Numerical Attributes (such as Rj, QCj, FRj, or whatever).

2) Logical Information

   -- Tests are conducted with GATE Block

   -- Tests involve interrogation of one of the Logical Attributes of a Facility, Storage, or Logic Switches (such as NU, SNE, LS, or whatever).

***Alternatively, these explicit tests may be put into two categories on the basis of the mode in which the test is conducted:

1) Refusal Mode

   -- No change in the direction of flow of the Transaction is permitted.

   -- If the condition being tested is not satisfied, the Transaction must wait at the TEST Block or GATE Block.

2) Transfer Mode

   -- If the condition being tested is satisfied, the Transaction moves from the TEST Block or GATE Block to the next sequential Block.

   -- If the condition being tested is not satisfied, the Transaction moves from the TEST Block or GATE Block and is routed to a non-sequential Block specified by the programmer.

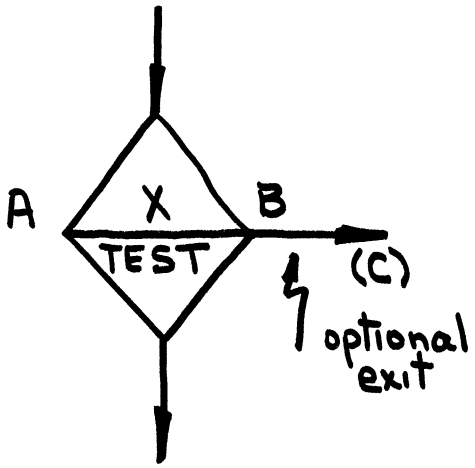
***In Section 42 we take up the TEST Block. Then, in Section 43, the GATE Block is discussed.

42. Tests Involving Numerical Information

The TEST Block:

Purpose: Test a relation between two Standard Numerical Attributes, then use the test result to control the movement of the Transaction which initiated the test

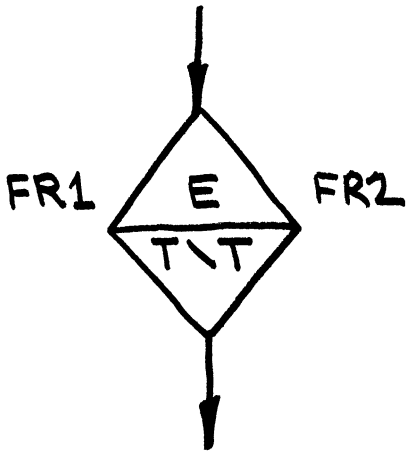
Shape and Operands:



Operand	Significance
X	X is one of the relational operators: G; GE; E; NE; LE; or E representing "greater than," "greater than or equal to", "equal to", "not equal to", "less than or equal to", or "equal to", respectively. X is termed an auxiliary operator. It is punched in the Operation Field of the punchcard, in card column 13 (or 13 and 14).
A	Specifies one of the two Standard Numerical Attributes involved in the test.
B	Specifies the other Standard Numerical Attribute involved in the test.
C	Optional operand controlling the mode of the test: -- if C is not specified, the test is conducted in Refusal Mode; in this case, the non-sequential exit (Optional exit) is not shown. -- if C is specified, it is the symbolic name of the non-sequential Block to which Transactions will be routed if the relation tested is not satisfied

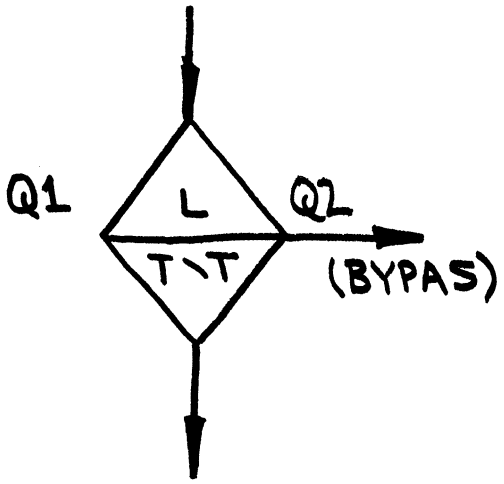
Examples:

1) Suppose FR2 = 371, FR7 = 126, and a Transaction encounters this Block:



Then the relation is not satisfied and the Transaction is not permitted to move further (until conditions change so that the relation being tested is satisfied).

2) Suppose Q1 = 5, Q2 = 3, and a Transaction encounters this Block:



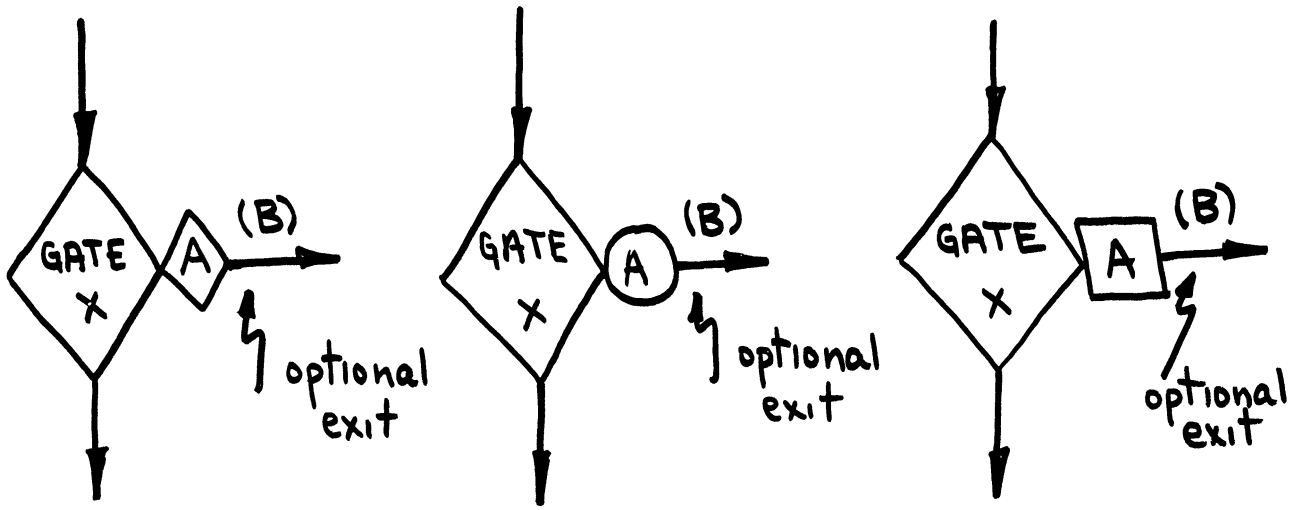
Then the relation is not satisfied and the Transaction is routed to the non-sequential Block named BYPAS. (The test might be used, for example, to route a Transaction to whichever one of two waiting lines is the shortest.)

43. Tests Involving Logical Information

The GATE Block:

Purpose: Examine a Logical Attribute to determine whether it is currently True or False, using the result to control the movement of the Transaction which initiated the test

Shape and Operands



Facility Test | Storage Test | Logic Switch Test

Operand	Significance
X	X is one of the Logical Attribute ninemonics: Facility Test: NU or U Storage Test: SE; SNE; SF; or SNF Logic Switch Test: LR or LS X is termed on auxiliary operator. It is punched in the Operation Field of the punchcard, in card column 13 (or 13 and 14; or 13, 14, and 15).
A	Specifies the number of the Facility, Storage, or Logic Switch which is being tested
B	Optional operand controlling the mode of the test: if B is not specified, the test is condusted in Refusal Mode; in this case, the non-sequential exit (optional exit) is not shown if B is specified, it is the symbolic name of the non-sequential Block to which Transactions will be routed if the Logical Attribute examined is False

44. Barber Shop: Model Six

Statement of the Problem

Modify the Block Diagram of Figure 3-A so that it will satisfy these conditions:

1) No more customers will be permitted to enter the shop after eight hours of simulated time have elapsed.

2) The simulation will not shut off until all customers in the shop at the time of closing have been serviced.

Approach Taken in Building the Model

Combined use is made of the GATE, TEST, and LOGIC Blocks, and the Block Counts (Current Count and Total Count) available as Wj and Nj. In particular, the model features this logic:

1) Transactions are not allowed into the model (i.e. out of the GENERATE Block) unless Logic Switch 1 is reset. At time 480, a Transaction is created in the Timer Segment and sets Logic Switch 1. Hence condition 1) above is satisfied.

2) The simulation can shut off at or after time 480 when the number of Transactions entering the shop equals the number leaving the shop. A TEST Block operating in Refusal Mode in the Timer Segment can be used to prevent the Timer Transaction from turning the simulation off until two appropriately selected Nj's in the main segment are equal.

Table of Definitions

Time Unit: 1 Minute

GPSS Entity	Interpretation
Transaction	Customer in Main Segment
	Timer in Timer Segment
Facility 3	The Barber
Logic Switch 1	Logic Switch whose status is used to control entry of customers to the shop

Other Model Documentation

Figure	Information Exhibited
12-A	Block Diagram
12-B	Program Listing
12-C	Selected Program Output; Time and Cost Data

Discussion

The Block Diagram should be studied in conjunction with the model description. It should be readily evident how the approach described has been implemented in the model. Notice in the Figure 12-C output that the barber was done at times 493 and 501 on the first and second days, respectively. Output from the third and fourth days is not shown.

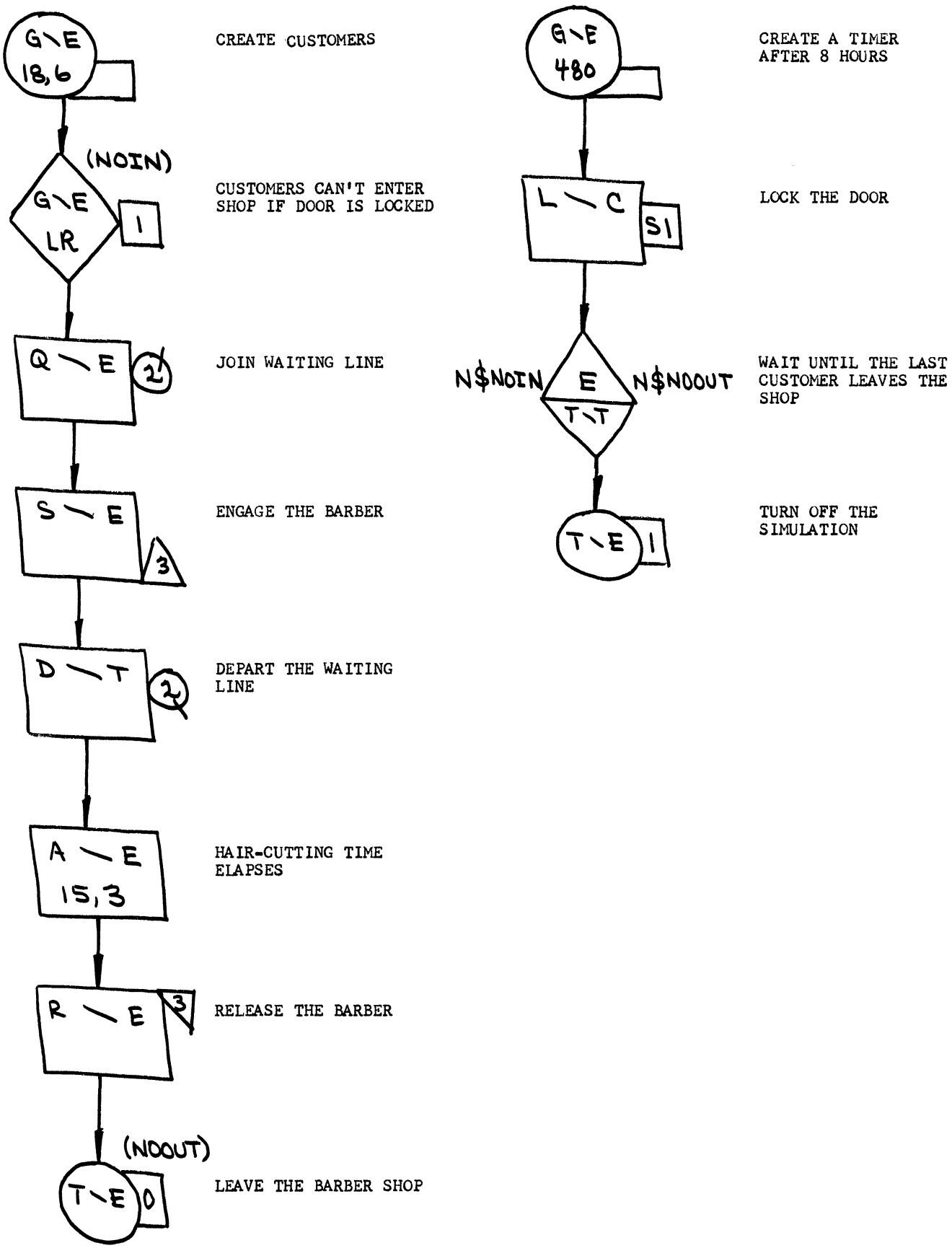


FIGURE 12-A: BLOCK DIAGRAM
(BARBER SHOP: SIXTH MODEL)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
       SIMULATE
*
* THE BARBER SHOP PROBLEM - ONE BARBER, ONE TYPE OF CUSTOMER
* THIS MODEL FINISHES PROCESSING 'IN SHOP' CUSTOMERS AFTER CLOSING TIME
*
       GENERATE   18,6          CREATE CUSTOMER ARRIVALS
 NOIN  GATE LR    1             DOOR LOCKED => CAN'T GET INTO THE SHOP
       QUEUE      2             CUSTOMERS QUEUE UP IF NECESSARY
       SEIZE      3             ENGAGE THE BARBER WHEN HE BECOMES AVAILABLE
       DEPART     2             CUSTOMER LEAVES THE QUEUE
       ADVANCE    15,3          CUSTOMER GETS HIS HAIR CUT
       RELEASE    3             RELEASE THE BARBER
 NOOUT TERMINATE  0             LEAVE THE BARBER SHOP
*
       GENERATE   480           GENERATE A TIMER AFTER 8 HOURS OF SIMULATED TIME
       LOGIC S    1             THE TIMER 'LOCKS THE DOOR'
       TEST E     N$NOIN,N$NOOUT  WAIT UNTIL CUSTOMERS IN = CUSTOMERS OUT
       TERMINATE  1             SHUT OFF THE RUN
       START      1             CARRY OUT THE SIMULATION
       CLEAR
       START      1
       CLEAR
       START      1
       CLEAR
       START      1
       END                      RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                           SIGN-OFF
```

FIGURE 12-B: PROGRAM LISTING
(BARBER SHOP: SIXTH MODEL)

```
 1      GENERATE   18     6
 2      GATE LR    1
 3      QUEUE      2
 4      SEIZE      3
 5      DEPART     2
 6      ADVANCE    15     3
 7      RELEASE    3
 8      TERMINATE  0
*
 9      GENERATE   480
 10     LOGICS     1
 11     TEST E     N2     N8
 12     TERMINATE  1
        START      1
```

FIGURE 12-C-1: ASSEMBLED PROGRAM
(BARBER SHOP: SIXTH MODEL)

```
RELATIVE CLOCK          493  ABSOLUTE CLOCK          493
BLOCK COUNTS
BLOCK CURRENT    TOTAL     BLOCK CURRENT    TOTAL
    1       1       28        11       0        1
    2       0       27        12       0        1
    3       0       27
    4       0       27
    5       0       27
    6       0       27
    7       0       27
    8       0       27
    9       0        1
   10       0        1

FACILITY      AVERAGE          NUMBER          AVERAGE          SEIZING     PREEMPTING
              UTILIZATION      ENTRIES         TIME/TRAN        TRANS. NO.  TRANS. NO.
       3         .821             27            15.000

QUEUE      MAXIMUM     AVERAGE     TOTAL      ZERO      PERCENT      AVERAGE      $AVERAGE
           CONTENTS    CONTENTS    ENTRIES    ENTRIES   ZEROS        TIME/TRANS   TIME/TRANS
    2           1        .083         27         16       59.2          1.518        3.727
```

FIGURE 12-C-2: OUTPUT (FIRST EIGHT HOUR DAY SIMULATION)

```
RELATIVE CLOCK          501  ABSOLUTE CLOCK          501
BLOCK COUNTS
BLOCK CURRENT    TOTAL     BLOCK CURRENT    TOTAL
    1       1       29        11       0        1
    2       0       28        12       0        1
    3       0       28
    4       0       28
    5       0       28
    6       0       28
    7       0       28
    8       0       28
    9       0        1
   10       0        1

FACILITY      AVERAGE          NUMBER          AVERAGE          SEIZING     PREEMPTING
              UTILIZATION      ENTRIES         TIME/TRAN        TRANS. NO.  TRANS. NO.
       3         .904             28            16.178

QUEUE      MAXIMUM     AVERAGE     TOTAL      ZERO      PERCENT      AVERAGE      $AVERAGE
           CONTENTS    CONTENTS    ENTRIES    ENTRIES   ZEROS        TIME/TRANS   TIME/TRANS
    2           1        .175         28         11       39.2          3.142        5.176
```

FIGURE 12-C-3: OUTPUT (SECOND EIGHT HOUR DAY SIMULATION)

```
          CPU TIME USED:

     ASSEMBLY:       .442 SECONDS

     EXECUTION:     1.156 SECONDS

**** ON AT  04:35.04
**** OFF AT 04:35.29
**** ELAPSED TIME        25.013  SEC.
**** CPU TIME USED         5.726  SEC.
**** STORAGE USED        206.116  PAGE-SEC.
**** CARDS READ           31
**** LINES PRINTED
**** PAGES PRINTED
**** CARDS PUNCHED        1
**** DRUM READS           85
**** APPROX. COST OF THIS RUN     $.52
```

FIGURE 12-C-4: TIME AND COST DATA

(OUTPUT FROM THIRD AND FOURTH DAYS NOT SHOWN)

45. On the Importance of Controlling the Event Sequence During a Simulated Clock Instant

Consider Figure 12-A. The simulator turns off the run at the simulated clock instant that the last customer leaves the shop. It does this by conducting a test and establishing that N$NOIN equals N$NOOUT. But this implies that the test is conducted in real time after the last customer left. Remember that the model is updated sequentially in real time. Hence, even though the two activities "last customer is caused to leave the shop" and "test is conducted to see if last customer has left the shop" occur at the same instant of simulated time, one activity necessarily precedes the other in real time. The importance of the sequence in which the two activities occur should be clear. Consider the two possible sequences and their consequences:

Sequence One

First Activity:	Last customer is caused to leave.
Second Activity:	Test of N$NOIN and N$NOOUT is conducted. N$NOIN = N$NOOUT. Model shuts off and simulation is completed successfully.

Sequence Two

First Activity:	Test of N$NOIN and N$NOOUT is conducted. N$NOIN $\neq$ N$NOOUT. Model does not shut off.
Second Activity:	Last customer is caused to leave.

Unless the simulator now "goes back" at the current instant of simulated time and repeats the test, the model is invalid. Without such a re-testing, the simulation clock will be advanced to time 960 (when a "second" Timer is scheduled to come out of the Timer GENERATE Block). When the test is conducted at time 960, the model shuts off. The outputted statistics are then distorted (for example, Facility utilization would be just somewhat more than half of what it should be). Such a distortion, or distortions, may be far from self-evident in a model of a complex system.

This then again raises the question: How does GPSS do things internally? The matter was touched upon in Section 17. And the PRIORITY Block example (with and without the BUFFER option) in Section 29 illustrated the importance of "event sequence during a simulated clock instant" in another specific context. The reader should now re-study Sections 17 and 29.

As it turns out, Sequence Two above is the one followed by the simulator for the Figure 12-A model. The model ran successfully because the Sequence Two Second Activity caused a status change (a change in the condition of a Facility, Storage, or Logic Switch) which in turn caused GPSS to automatically re-start the scan of the Current Events Chain. In returning to the front of the chain, GPSS really returned to the Transaction blocked at the TEST and attempted again (for the second time during the simulated clock instant) to move the Transaction forward. This time the attempt was successful, and the simulation was completed.

Between this section and Section 29, then, we have two cases where re-starting the Current Events Chain scan has been of critical importance for model validity. In one case, by using the PRIORITY Block BUFFER option, we explicitly directed the simulator to re-start the scan; in the other case, GPSS re-started the scan of its own accord. With these two examples as motivation, proceed to take a more detailed look at how GPSS orders its affairs internally:

Study page 16 ("Basic GPSS/360 Chains") and pages 96-104 and 106-107 (from "PRIORITY BLOCK TO CHANGE PRIORITY LEVEL OF TRANSACTIONS" up to "Example 3") in the User's Manual. The various logical flowcharts included there should be given careful attention in particular.

As a last point here, note that programmer logic could have been used to force the simulator to follow Sequence One above. This could be done simply by assigning Transactions in the Main Segment a higher priority level than the Transaction in the Timer Segment. Then the Timer Transaction would be at the end of the Current Events Chain while it waited for the TEST condition to be satisfied. This in turn means that the test would be conducted as the last activity in any simulated clock instant at or after time 480.

46. Loops in GPSS

*** The logic of a model may require that one or more Transactions be repeatedly recycled through a given segment of the model. Such recycling could be accomplished in a variety of ways. One approach would be to use a Transaction Parameter to count the number of times the Transaction has been through the given segment, or loop. A simple example illustrates this approach. Suppose a Transaction is to join whichever one of Queues 1 through 6 has the least contents currently. Then the model segment shown in Figure 13-A can be used to implement the logic involved in selecting the Queue. Note the presence of the Initialize-Decrement-Test pattern in Figure 13-A. (Of course, an Initialize-Increment-Test pattern could have been used as well; and, for either pattern, the test could precede the Decrement or Increment step.)

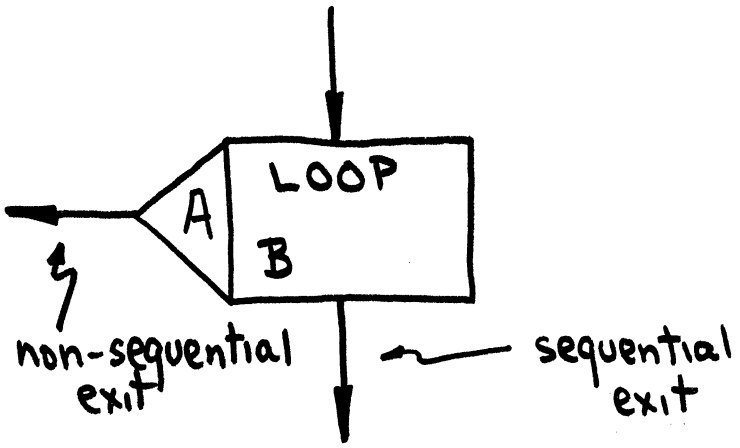
*** In GPSS, the Decrement-Test sequence can be represented with a single Block, the LOOP Block, when three conditions are satisfied:

1) A Parameter is being used as the LOOP counter,
2) The size of the decrement is 1, and
3) Looping is to continue until the Looping Parameter has been decremented to zero.

The Loop Block:

Purpose: Control the number of times a Transaction is recycled through a model segment before it is permitted to move on to the next portion of the model.

Shape and Operands:



Operand	Significance
A	Specifies the number of the Parameter which is to be decremented by 1, then tested against zero.
B	Specifies the symbolic name of the non-sequential Block to which the Transaction is routed when the Parameter's value is non-zero

Nature of Operation:

1) Transaction enters LOOP Block

2) The specified Parameter's value is decremented by 1, then tested against zero.

3) a. If the resulting value is non-zero, the Transaction is routed to the non-sequential Block.

   b. Otherwise, the Transaction proceeds to the next sequential Block.

Example: Construct a model segment to determine how many of the Facilities between 7 and 11, inclusive, have a current utilization less than .25. Use P1 as the Looping Parameter, store the answer in P2, and use P3 to hold the number of the Facility currently being tested. Figure 13-B shows a model segment which carries out the determination.

47. Random Numbers in GPSS

***There are eight random number generators which can be referenced in GPSS. A random number is specified as RNj, where j is one of the digits 1 through 8. Hence RN3 is synonymous with a random number returned by the third random number generator.

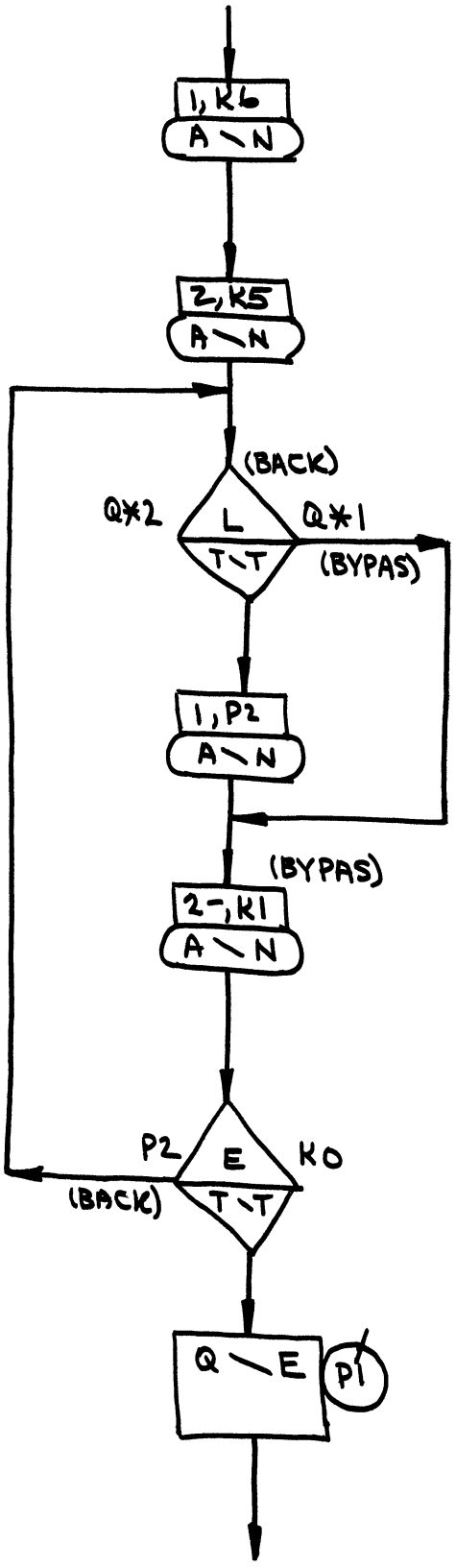
***The value of RNj depends upon the context in which RNj is used. There are two possibilities:

RNj may take on an integer value over the closed interval from 0 to 999.

RNj may take on a floating point value over the closed interval from 0.00000 to 0.99999.

***The floating point value is taken on only when RNj is the argument of a GPSS Function (to be discussed in Section 50).

***In any event, RNj takes on the value of a sample which is drawn from a population uniformly distributed over whichever one of the two closed intervals is appropriate.



INITIALIZE P1
(P1 HOLDS NUMBER OF
"SHORTEST QUEUE SO FAR")

INITIALIZE P2
(P2 IS THE LOOPING PARAMETER;
IT ALSO HOLDS THE NUMBER OF
THE "OTHER QUEUE" INVOLVED IN
EACH TEST)

SHOULD THE VALUE OF
P1 BE CHANGED?

UPDATE THE VALUE OF P1

DECREMENT THE LOOPING
PARAMETER (SIMULTANEOUSLY:
PREPARE TO TEST THE NEXT
QUEUE)

IS THE TESTING PROCESS
COMPLETE?

JOIN THE SHORTEST QUEUE
(OR, IN CASE OF TIES, JOIN
ONE OF THE SHORTEST QUEUES)

FIGURE 13-A: A LOOP TO TEST FOR THE SHORTEST QUEUE
(LOOP BLOCK NOT USED)

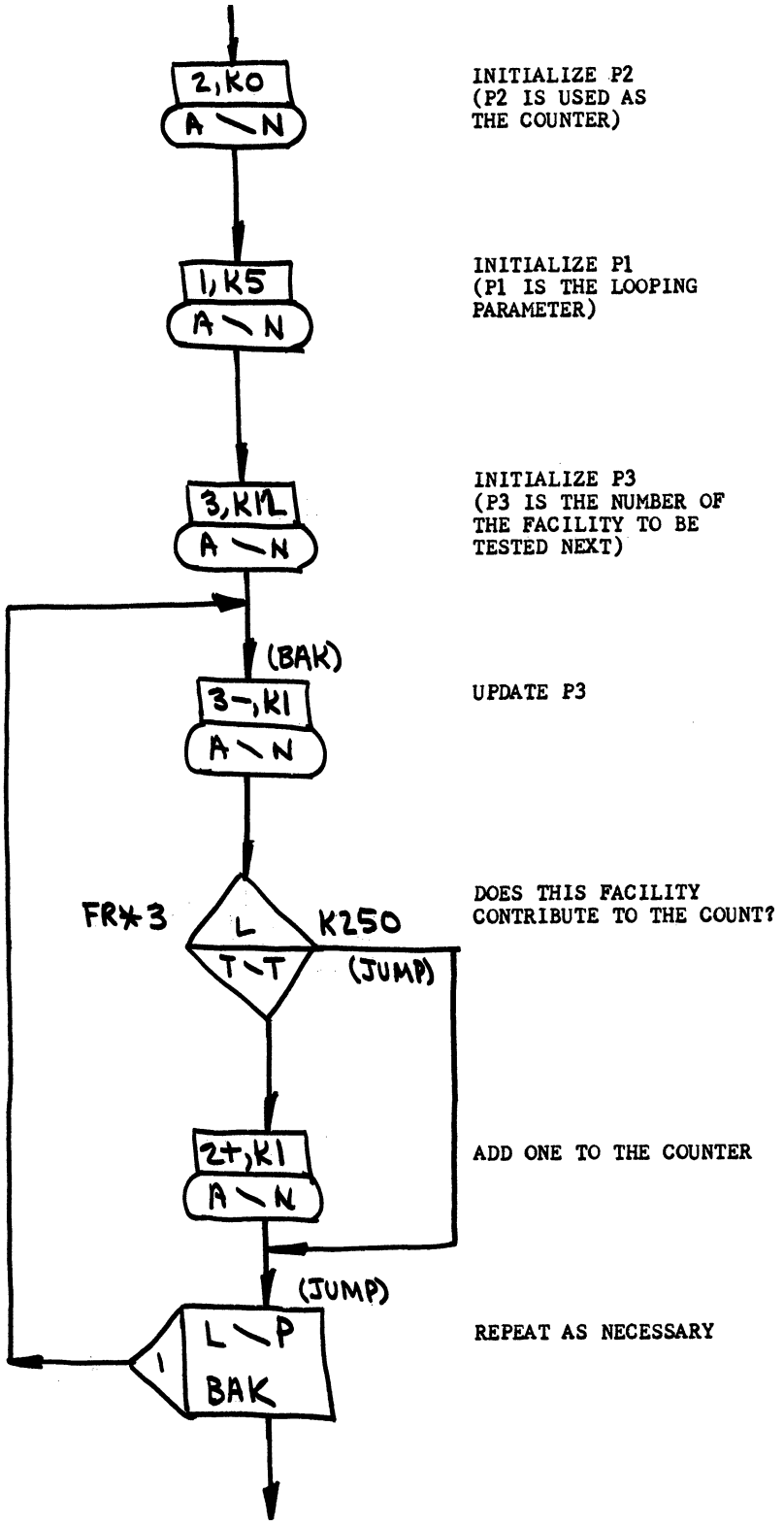


FIGURE 13-B: A LOOP TO COUNT FACILITIES WITH FRj LESS THAN 250
(LOOP BLOCK USED)

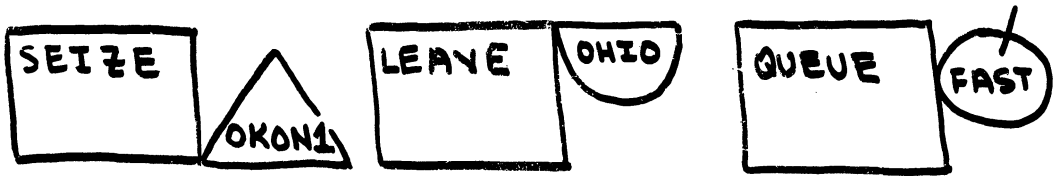
48. Symbolic Naming of Entities in GPSS

*** Indirect specification and indirect addressing both ultimately involve fetching data from some storage location. Depending on the context of the fetch, the content of the storage location may then be used as a Queue number, or the number of units of space required in a Storage, or whatever.

***It is also possible to give certain GPSS entities symbolic names, leaving to the simulator (assembly program) the task of associating numbers with those names. As the model is assembled, the assembly program then provides the appropriate number whenever a symbolic name is encountered. It does this by using a dictionary which it builds up during the assembly process.

***The GPSS entities so far discussed which can be given symbolic index numbers are Facilities, Storages, Queues, and Blocks (we saw on page 7 of these notes that symbolic names can be given to GPSS blocks).

***Here are some examples using symbolic names:



***As mentioned on page 7 of these notes, symbolic names consist of from 3 to 5 alphanumeric characters, with the restriction that the first 3 characters must be alphabetic.

49. User-Defined Standard Numerical Attributes

*** The Standard Numerical Attributes associated with Transactions, Facilities, Storages, Queues, and Blocks are pre-defined variables which are an inherent part of GPSS. There are also other pre-defined Standard Numerical Attributes which have not yet been described in this monograph.

*** GPSS also permits the user to define certain of his own Standard Numerical Attributes. There are three types of such user-defined SNA's:

1) Functions
2) Arithmetic Variables
3) Boolean Variables

*** When Functions, Arithmetic Variables, and Boolean Variables have been defined in a given model, they are considered a part of that model's set of Standard Numerical Attributes.

*** We now take up a discussion of user-defined Functions.

50. Functions in GPSS

*** There are five types of Functions which can be defined in GPSS. Only two of those types will be discussed now.

*** For our purposes here, a Function is defined by supplying to the simulator five pieces of information:

1) The number of the Function

2) The argument of the Function

3) Whether the Function is discrete or continuous

4) The number of ordered pairs of points being supplied to describe the Function

5) The ordered pairs of points themselves.

*** Analogous to the control cards and the Storage Capacity Definition Card, definition of Functions involves only cards; there is no Block involved in the definition.

*** The first four pieces of information described above are entered into a single punchcard having this format:

Card Columns:	2 ◄--► 6	8 ◄-----► 8	9 ◄--------► 1 1
Contents:	j	FUNCTION	A,B

where:

j is the number of the Function being defined,

A is the argument of the Function; the argument can be any Standard Numerical Attribute, such as P2, or Q7 or RN1, or whatever.

B is one of the alphabetic characters C (for continuous) or D (for discrete), followed without intervening blanks by n, where n is the number of ordered pairs of points being supplied on following cards.

*** The fifth piece of information appears on one or more (if required) punchcards taking this form:

Card columns:	1 ◄ - - - - - - - - - - - - - - - - - - ► 7 1
Contents:	$X_1,Y_1/X_2,Y_2/X_3,Y_3/ \ . \ . \ . \ /X_n,Y_n$

where:

X_i,Y_i is the i-th ordered pair, and

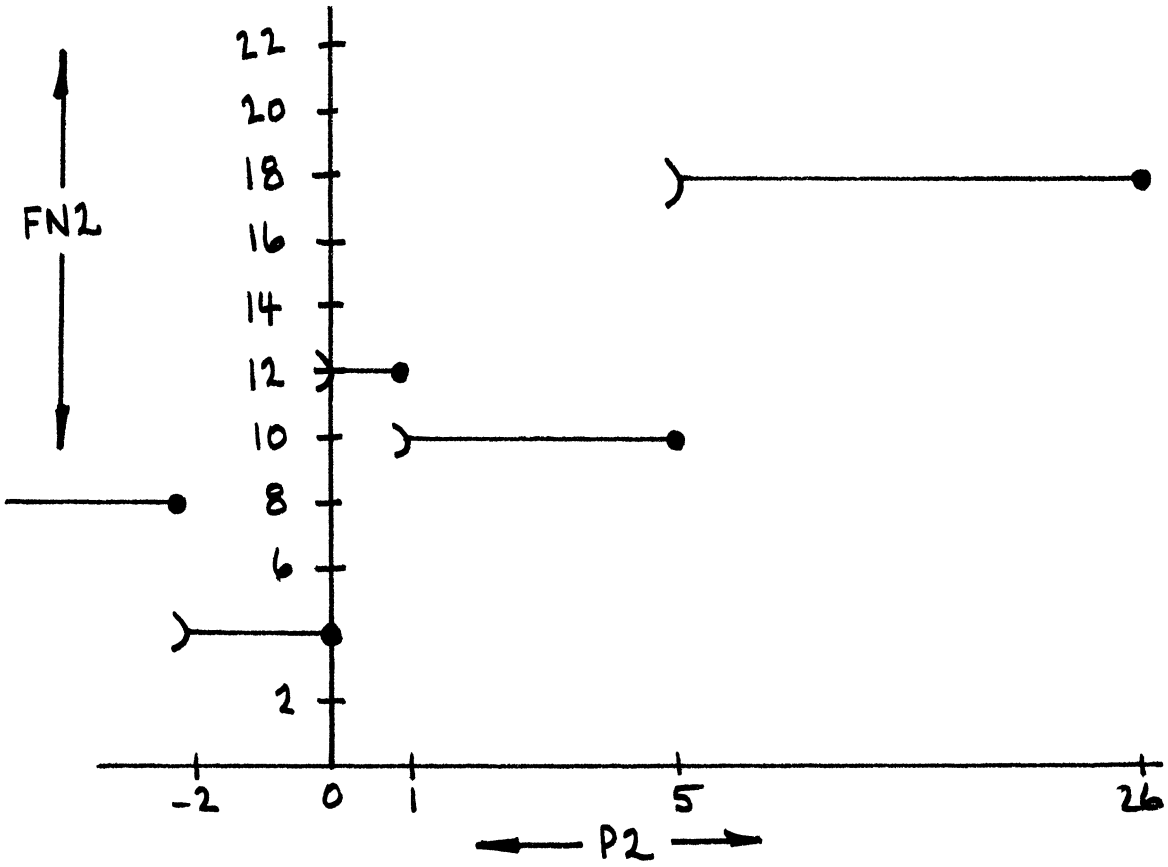
$X_1 < X_2 < X_3 < \ . \ . \ . < X_n$

*** Two examples are now given:

Example 1:

```
          2    FUNCTION     P2, D5
-2,8/0,4/1,12/5,10/26,18
```

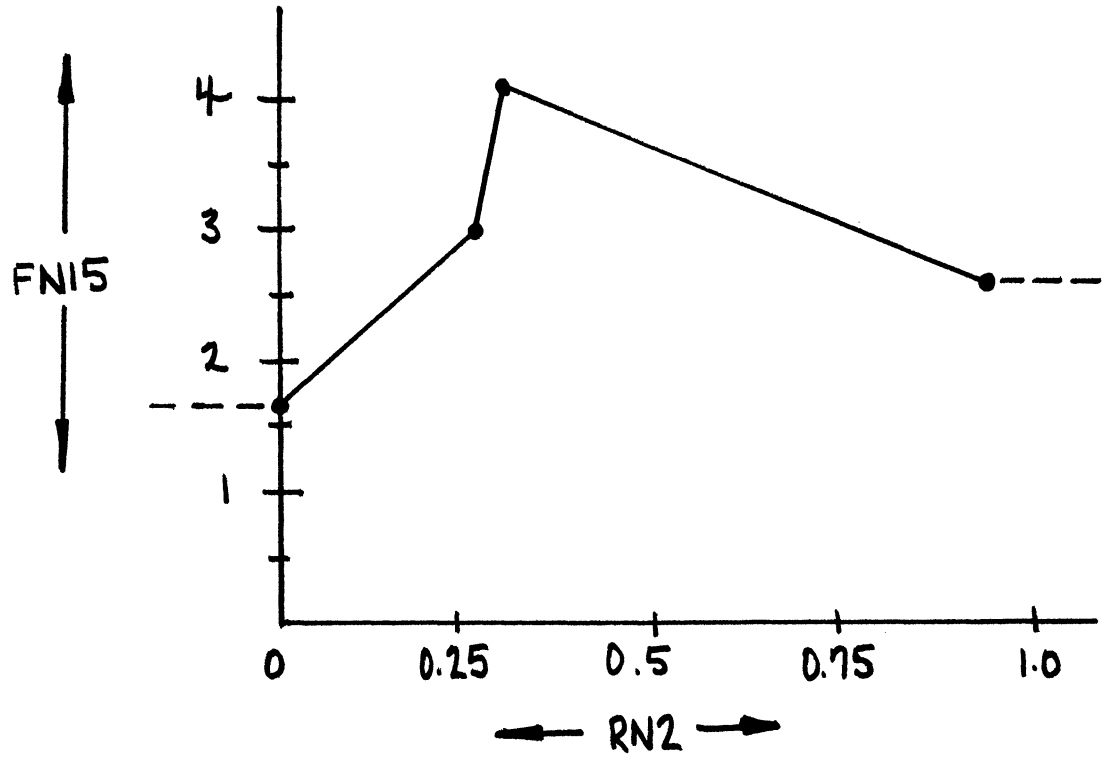
Geometric Interpretation of Function 2:



Example 2:

```
15    FUNCTION    RN2, C4
0,1.65/.32,3/.38,4.1/.992,2.665
```

Geometric Interpretation of Function 15:



Example 1 shows the definition of Function 2 as a discrete, 5-point function. In Example 2, Function 15 is shown being defined as a continuous, 4-point function. Function 2 uses the value of Parameter 2 as its argument (for the answer to the question: Parameter 2 of which Transaction?, see the footnote at the bottom of page 33, these notes). Function 15 uses the value returned by random number generator 2 as its argument (each time the value of Function 15 is required, a fresh value of RN2 will be fetched).

As the geometric interpretations shown above suggest, the simulator views discrete Functions as step functions and continuous Functions as consisting of a connected series of straight line segments. Hence the argument of a discrete Function is always mapped into one of the Y_i values supplied in defining the Function. In contrast, the argument of a continuous Function is (in general) determined by linear interpolation between an adjacent pair of Y_i values. These ideas are now spelled out in more detail:

***When the value of a Function is determined, one of four cases can arise:

1) The argument is less than or equal to the first member of the first ordered pair supplied in defining the Function. In this event, the Function takes on the value of the second member of the first ordered pair. This is true whether the Function is discrete or continuous.

2) The argument is greater than or equal to the first member of the last ordered pair supplied in defining the Function. In this event, the Function takes on the value of the second member of the last ordered pair. Again, this is true whether the Function is discrete or continuous.

3) The argument equals the first member of the k-th ordered pair, k = 2,3,4, ... , n-1, where n is the number of ordered pairs supplied. The Function then takes on the value of the second member of the k-th ordered pair. This happens whether the Function is discrete or continuous.

4) The argument takes on a value greater than the first member of the k-th ordered pair, but less than the first member of the (k+1)-st ordered pair, k = 1,2,3, ... , n-1. Then, if the Function is:

    a) discrete, it takes on the value of the second member of the (k+1)-st ordered pair.

    b) continuous, it takes on the value determined by linear interpolation between the bracketing ordered pair of points.

***A Function is referred to in GPSS as FNj, where j is the number of the Function in question.

***Analogous to Facilities, Storages, Queues, and Blocks, Functions can be given symbolic names. Hence we might name a Function EXPO. Note, however, that whereas FN15 is a valid reference to the value of Function 15, FNEXPO is illegal in GPSS. Whenever an entity has been given a symbolic name and one of its Standard Numerical Attributes is being referenced, a dollar sign ($) must stand between the SNA and the symbolic name of the index number. Consequently FN$EXPO is a valid reference to the value of the Function named EXPO. Similarly, FR$BARB1, Q$TRUBL, SC$RUTE7, and so on, could all be valid references to values of Standard Numerical Attributes.

51. Constructing Functions to Sample from Various Probability Distributions

***It is assumed that the reader is familiar with the concept of mapping a random number, drawn from the population uniformly distributed over the unit interval from 0 to 1, into a number representing a random draw from some other population (the target population) whose distribution is known. In GPSS terminology, RNj is a random number from the unit interval just described; and, after it is appropriately defined, FNj is the random draw from the target population. The steps involved in defining FNj are essentially the same whether the target population consists of a finite or an infinite number of values. The steps will now be summarized and illustrated with examples for each of the two cases:

Case 1) When the target population is composed of an infinite number of values:

1) Develop the cumulative distribution function F(x) from the probability density function f(x).

2) Form the inverse of the cumulative distribution function, $F^{-1}(x)$.

3) For selected values of F(x), use the inverse function to tabulate the corresponding values of X.

4) Present the resulting ordered pairs to the simulator in the usual manner for defining a function.

Example

Set up a GPSS Function with which it is possible to sample from the population whose probability density function is $f(x) = 2x$, $0 \leqslant x \leqslant 1$.

1) A straightforward integration reveals that the cumulative distribution function is

$$F(x) = x^2, \quad 0 \leqslant x \leqslant 1.$$

2) The inverse is formed by solving for x:

$$x = (F(x))^{1/2}$$

3) The inverse is used to produce this tabulation:

F(x) (=RN1)	x (=FN3)
0	0
.01	.1
.04	.2
.09	.3
.16	.4
.25	.5
.36	.6
.49	.7
.64	.8
.81	.9
1.0	1.0

4) The Function is defined in GPSS as:

```
   3   FUNCTION    RN1,C11
0,0/.01,.1/.04,.2/.09,.3/.16,.4/.25,.5
.36,.6/.49,.7/.64,.8/.81,.9/1,1
```

Comments

1) The Function has been arbitrarily designated as Function 3.

2) The argument used, RN1, could just as well have been any one of the other RNj, j = 2,3,4,...,8.

3) The Function has been defined as a continuous Function so that FN3 can take on values other than the 0, .1, .2, etc. used in its definition.

4) Remember that FNj values are integerized except when FNj is the B Operand in the GENERATE or ADVANCE Block, or when j is the C Operand in the ASSIGN Block. This means that if FN3 is used in any other context, its value will necessarily be zero.

5) Steps 1) and 2), although posing no problem in this example, may not be analytically feasible, depending on f(x). In this case, numeric integration and/or root finding techniques can be invoked, but at best only awkwardly in GPSS.

Case 2) When the target population taken on a finite number of values:

1) List the values of the random variable X in a column in ascending order.

2) In an adjacent column, list the probability p(X) associated with each value of X.

3) In a third column, enter row-by-row the sum of the column two entries down to and including the entry in each particular row. (That is, form the cumulative distribution P (X).)

4) Now use the third and first column entries as the first and second members, respectively, of the ordered pairs with which the GPSS Function is defined.

Example

Set up a GPSS Function which can be used to sample from the population whose values are 0, 1, and 2 with probabilities 1/4, 1/2, and 1/4, respectively.

1, 2, and 3) The tabulation described is shown below:

X	p(X)	P(X)
0	.25	.25
1	.50	.75
2	.25	1.0

4) The Function is defined in GPSS as:

```
   2   FUNCTION    RN6,D3
.25,0/.75,1/1,2
```

Comments

Use of 2 as the Function number and RN6 as the Function argument is arbitrary. The Function is defined as discrete so that it can only take on the values 0, 1, and 2, no matter the context in which it is used.

***Two concepts which arise in queuing theory involve Poisson arrival rates and exponential service times. The probability distributions associated with these concepts are the Poisson and exponential distributions, respectively. These theoretical distributions are of interest because the assumptions underlying them are sometimes satisfied in practice. Hence their use in the queuing area is often justified. It is therefore of interest to learn how to sample from these distributions when building GPSS models.

As it turns out, only one Function need be defined to structure Poisson arrivals from GENERATE Blocks and exponential service times at ADVANCE Blocks. This may seem strange at first, because the Poisson and exponential distributions are dissimilar; in fact, the former is discrete and the latter is continuous. Nevertheless, it is the distribution of time between arrivals which must be specified at GENERATE Blocks and, when arrival rates are Poisson, interarrival times follow the exponential distribution. Our attention turns, then, to the exponential distribution.

The exponential random variable is distributed according to the density function:

$$f(x) = (1/\emptyset)e^{-x/\emptyset}, \qquad x > 0$$

where $\emptyset$ is the distribution parameter and is, in fact, the mean of the distribution. (For our purposes, then, $\emptyset$ signifies the mean interarrival time at a GENERATE Block, or the mean service time at an ADVANCE Block.)

We now follow the steps outlined above to make it possible to sample from this distribution in GPSS:

1) Integrating f(x), the cumulative distribution function is:

$$F(x) = 1 - e^{-x/\emptyset}$$

2) The inverse is formed by solving for x:

$$x = -\emptyset \ln \left[1-F(x)\right]$$

3) The inverse is used to produce this tabulation:

F(x) (=RN1)	$-\ln \left[1-F(x)\right]$ (=FN$EXPO)
0	0
.1	.104
.2	.222

intermediate values are shown in 4) below

.999	7
.9998	8

Notice that in forming the table, the factor Ø has not been used in the second column. Recall from Section 6 that when a GENERATE or ADVANCE Block B Operand is FNj, the Function value will be used multiplicatively, without integerizing, to modify the A Operand (then the integerized product is used as interarrival time or service time, respectively). Hence, when Poisson arrivals or exponential servers are to be modeled, the desired effect is achieved by providing Ø as the A Operand and the Function shown above as the B Operand. Ø is then particular to the various GENERATE and/or ADVANCE Blocks. This leaves the Function in such general form that it can be referenced from various points in a model, independent of the Ø involved.

4) The Function is defined in GPSS as:

```
                    EXPO  FUNCTION   RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
```

Comments

1) Poisson arrivals are often specified as rates. Remember, it is not this rate but the corresponding interarrival time which is to be provided as the GENERATE Block A Operand. For example, if an average of 10 people per hour arrive at a ticket office, the average interarrival time is six minutes.

2) Interarrival time depends, of course, on the time unit chosen for the model. Interarrival times of six minutes and 360 seconds are fully equivalent.

3) If the interarrival time is too small as expressed in the time unit chosen, the distribution which results by sampling from the exponential may bear little resemblance to the exponential itself. As an example, if the A Operand in a GENERATE Block is 1, then inspection of FN$EXPO shows that more than 60% of the time, an interarrival time of zero results after integerizing. This potential problem can be avoided by choosing a small time unit for the model, resulting in a larger A Operand value. This remark also applies, of course, to use of the exponential function in ADVANCE Blocks.

52. Arithmetic Variables

*** An Arithmetic Variable is any arbitrary arithmetic combination of Standard Numerical Attributes.

*** Constants are considered to be Standard Numerical Attributes. They are written as Kj, where j is the constant. Only integer constants are permissible in GPSS. The user can supply decimal points in only two contexts:

1) In specifying the ordered pairs of points constituting Function definition, and

2) In specifying the A operand of the TRANSFER Block used in Statistical Transfer Mode, a topic not yet discussed.

*** The arithmetic operators that may be used in forming the combination are +, -, *, /, and @, where the first four operators are well known and the character @ denotes modulus division. (In modulus division, the quotient is discarded and the remainder is retained as the result of the division. Examples: 10@3 = 1; 4@2 = 0.)

*** Pairs of parentheses may be used to group terms in the process of defining Arithmetic Variables. The usual chronology holds for evaluation of expressions.

*** Arithmetic Variables can only take on integer values.

*** Each Arithmetic Variable used in a model is defined by including a Variable Definition Card as part of the model. The card takes this form:

card columns:	2◄--►6	8◄---►8 (1)	9◄-----► (1)
contents	j	VARIABLE or FVARIABLE	arithmetic combination constituting the definition

where:

j is the number of the Variable being defined.

When VARIABLE is used in the Operation Field, integerizing occurs step-by-step as the value of the Variable is built up. When FVARIABLE appears in the Operation Field, floating point arithmetic is used to evaluate the variable up to the last step, at which time integerizing occurs. Note: @ is an illegal operator when FVARIABLE is used in the Operation Field.

The arithmetic combination constituting the definition must be punched in consecutive card columns and must not extend beyond card column 71. There is no continuation card concept in defining Variables.

*** Arithmetic Variables are referenced as Vj, independent of whether they have been defined as VARIABLES or FVARIABLES. As is the case with Functions, Arithmetic Variables must be defined in a model before the first reference to them is made. The value of an Arithmetic Variable is computed each time a reference is made to that Variable.

53. GPSS Program Output

*** A distinction can be made between two types of GPSS program output:

Type 1) Output which occurs after the simulation has stopped (that is, after the Termination Counter has been decremented to zero), and

Type 2) Output which occurs during the course of the Simulation.

Thus far we are familiar only with the first type of output. After mentioning some of the features of this first type, several possibilities in the second category will be described in this Section.

*** Type 1 output:

-- consists of fixed content information: Relative and Absolute Clock values; Block Counts; Facility, Queue, and Storage Statistics; Logic Switch Settings; SAVEVALUE and MSAVEVALUE values; and other items not yet discussed.

-- can be suppressed by specifying NP as the optional B Operand in the START card.

-- is of fixed format. (There is a GPSS Output Editor which can be used to overcome this fixed format feature. Details concerning its use are available in the Users Manual. An example of its use is given later in this monograph.)

*** Type 2 output:

-- there are two possibilities in this category:

1) Snap Interval Output

2) PRINT Block Output

Snap Interval Output

In addition to the Termination Counter, GPSS maintains a Snap Interval Counter. Its initial value is specified by the START Card C Operand. Each time the Termination Counter is decremented, the Snap Interval Counter (SIC) is decremented by a like amount. When the SIC has been decremented to zero or less, the full set of fixed-content, fixed-format information described above is printed out. The SIC is then automatically re-initialized, and the process continues.

Example:

For this START Card:

```
START   200,,75
```

three full sets of fixed-content, fixed-format information will result (two sets on the basis of the SIC being decremented to zero or less; one set on the basis of the TC being decremented to zero).

Note: Decrementing the SIC to zero does not result in zeroing-out any model statistics. Consequently, SIC-triggered statistics are cumulative. They cover a time span starting at a Relative Clock value of zero and ending at whatever the Relative Clock happens to read when the particular SIC printout occurs.

There is also an optional D Operand for the START Card. "A 1 in Field D of the START Card indicates that each statistical printout (either at a snap interval or at the end of a run) will also include a transaction printout of the current events chain, future events chain, interrupt transactions, user chain, and transactions in matching status." (page 190, User's Manual)

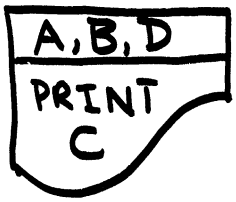
PRINT Block Output

Transactions themselves can trigger output by moving into a special Block known as the PRINT Block.

The PRINT Block:

Purpose: Initiate printout of a selected, limited nature each time a Transaction moves through the Block.

Shape and Operands:



Operand	Significance
C	Mnemonic code specifying the nature of the output; can be any one of these:

Mnemonic	Output
S	Storage statistics
Q	Queue statistics
F	Facility statistics
X	Savevalues (Fullword)
XH	Savevalues (Halfword)
MX	Matrix Savevalues (Fullword)
MH	Matrix Savevalues (Halfword)
LG	Logic Switch Status

The user also specifies the particular Storages, or Queues, or whatever, for which statistics are to be printed. This is done by indicating the inclusive lower and upper limits of the range of index numbers.

A	Specifies the lower limit of the range of output
B	Specifies the upper limit of the range of output
D	Optional operand; used for page control; when D is blank, a skip occurs to the top of the next page; when D is any single alphabetic character, the skip is suppressed.

Examples:

1) PRINT 3,5,F,S

   Print statistics for Facility 3 through Facility 5, saving paper in the process.

2) PRINT 5,5,MH

   Print out Matrix Halfword Savevalue 5 at the top of the next page.

Further Comment:

Consult page 187 in the User's Manual for further C Operand mnemonics available to the user.

54. A Problem in Stochastic Inventory Control

Statement of the Problem

In this problem we study a procedure for maintaining an adequate supply of spare parts of a particular type for a machine which uses the part and must be shut off from time to time because of part failure. In contrast to the Spare Parts Problem already studied, failed parts cannot be repaired but are discarded. Hence the supply of spares is gradually diminished and must be replenished periodically by placing orders with the parts supplier.

We are given:

1) The observed distribution of the part's lifetime (10 days on the average, negative exponentially distributed)

2) The observed distribution of the time that elapses between placing an order for additional parts with the supplier and arrival of that order at the site where the machine is located (30±10 days, uniformly distributed)

3) An inventory control procedure, known as Trigger Level Ordering, whereby an order for ROQ (re-order quantity) units is placed with the supplier each time the sum of spare parts in stock and parts on the way to the site (but not yet arrived) falls to ROP (re-order point).

Construct a GPSS model to determine, as a function of ROP and ROQ, the fraction of the time that the machine is down because no spare parts are available. Assume that time required to remove a failed part and install a spare is negligible. Build the model so that the (ROP, ROQ) combinations (2,3), (3,3), (4,3), (2,4), (3,4), and (4,4) can be investigated sequentially with a single submit.

Approach Taken in Building the Model

The model is set up in three segments:

1) A Machine Segment, in which a workman sees to it that the machine is on whenever possible,

2) An Inventory Segment, in which a worker places an order with the supplier whenever necessary, then puts the parts in inventory when the order arrives at the factory, and

3) A Timer Segment, which turns off the simulation after about 100 failures have occurred

Halfword MSAVEVALUE 1 carries pertinent information in Row 1 and in Columns 1 to 5,

Column	Information
1	Total number of parts on hand as spares
2	Number of parts on the way from the supplier
3	Re-order point
4	Re-order quantity
5	Fractional downtime of the machine

as described in the Table of Definitions. Appropriate MH1 values can then be supplied with an INITIAL card. After a given (ROP, ROQ) combination has been investigated, the CLEAR, INITIAL, START sequence can be used to provide for immediate investigation of the next combination.

Table of Definitions

Time Unit: 1 Hour

GPSS Entity	Interpretation
Transaction	A Shop Worker in the Machine Segment
	An Office Worker in the Inventory Segment
	P1: Re-order Point
	P2: Re-order Quantity
	A Timer in the Timer Segment
Facility 1	The machine
Function EXPO	The inverse cumulative distribution function corresponding to the exponential probability density function
Matrix 1 (Halfword)	Information Matrix
	Row 1 is the only row used

Column	Information
1	Total parts on hand as spares
2	Parts on way from the supplier
3	Re-order Point
4	Re-order Quantity
5	Fractional machine downtime

Variable LEVEL	The sum of parts on hand and parts on the way

Block Diagram

Figure 14-A shows a Block Diagram for the model.

Discussion of the Block Diagram

Machine Segment:

It is assumed that at time zero the machine is ready to be turned on. The documented Block Diagram should be self-explanatory.

Inventory Segment:

The "Office Worker" keeps an eye on the inventory situation at the Block named WATCH. When it is time to place an order, he updates the record of parts on the way before sending an offspring back to re-establish the watch. Notice that when the run starts, the Transaction involved has the ROP and ROQ copied into its P1 and P2, respectively. (This is done to reduce execution time by eliminating repeated reference to MH1(1,3) and MH1(1,4), which are invariant for a given run.) When the offspring is created at the SPLIT Block, it then automatically carries the ROP and ROQ in its P1 and P2, respectively.

Timer Segment:

The Timer Transaction stores the statistic of interest in MH1(1,5) then causes the Information Matrix to be printed and shuts off the run. The NP option is used for the B Operand of the START to eliminate unnecessary printing at the conclusion of the run.

The reader should try to defend the presence of the BUFFER Block (discussed in the Users Manual as referenced in Section 45 in these notes) in the Inventory Segment before reading the following explanation.

Without the BUFFER Block, the model could be invalid via this time sequence of events:

1) The Machine Segment Transaction is in delay status on the Current Events Chain (i.e. in the model it is the TEST Block because there are no spare parts on hand).

2) The simulator advances the Simulation clock to the time of arrival of the next order. The corresponding inventory segment Transaction is moved from the Future Events Chain to the back of the Current Events Chain (because all Transactions in the model have the same priority level).

3) Now the scan starts at the front of the Current Events Chain:

   a) Machine Segment Transaction can't move forward because MH1(1,1) hasn't been updated yet, even though an order is arriving at this simulated clock instant.

   b) Inventory Segment Transaction at Block WATCH is processed. Whether it continues to be delayed or not is of no consequence in this analysis.

   c) Inventory Segment Transaction just transferred to the Current Events Chain is processed: MH1(1,1) is incremented; MH1(1,2) is decremented; and the Transaction is destroyed. (Remember that this analysis assumes the BUFFER Block is not in the model.) This completes the scan of the Current Events Chain.

      Model validity is now violated because the machine should be turned back on at this clock instant but hasn't been and won't be.

4) The simulation clock is now advanced in the usual way and invalid statistics are accumulated for the unwary user.

With the BUFFER Block, Step 3-C above becomes: "MH1(1,1) is incremented; MH1(1,2) is decremented; and the simulator is directed via the BUFFER to re-start the scan of the Current Events Chain. When the delayed Machine Segment Transaction is processed this time (i.e. for the second time during this simulated clock instant), MH1(1,1) exceeds zero and the machine is turned on again. The model is consistent with the real-world situation.

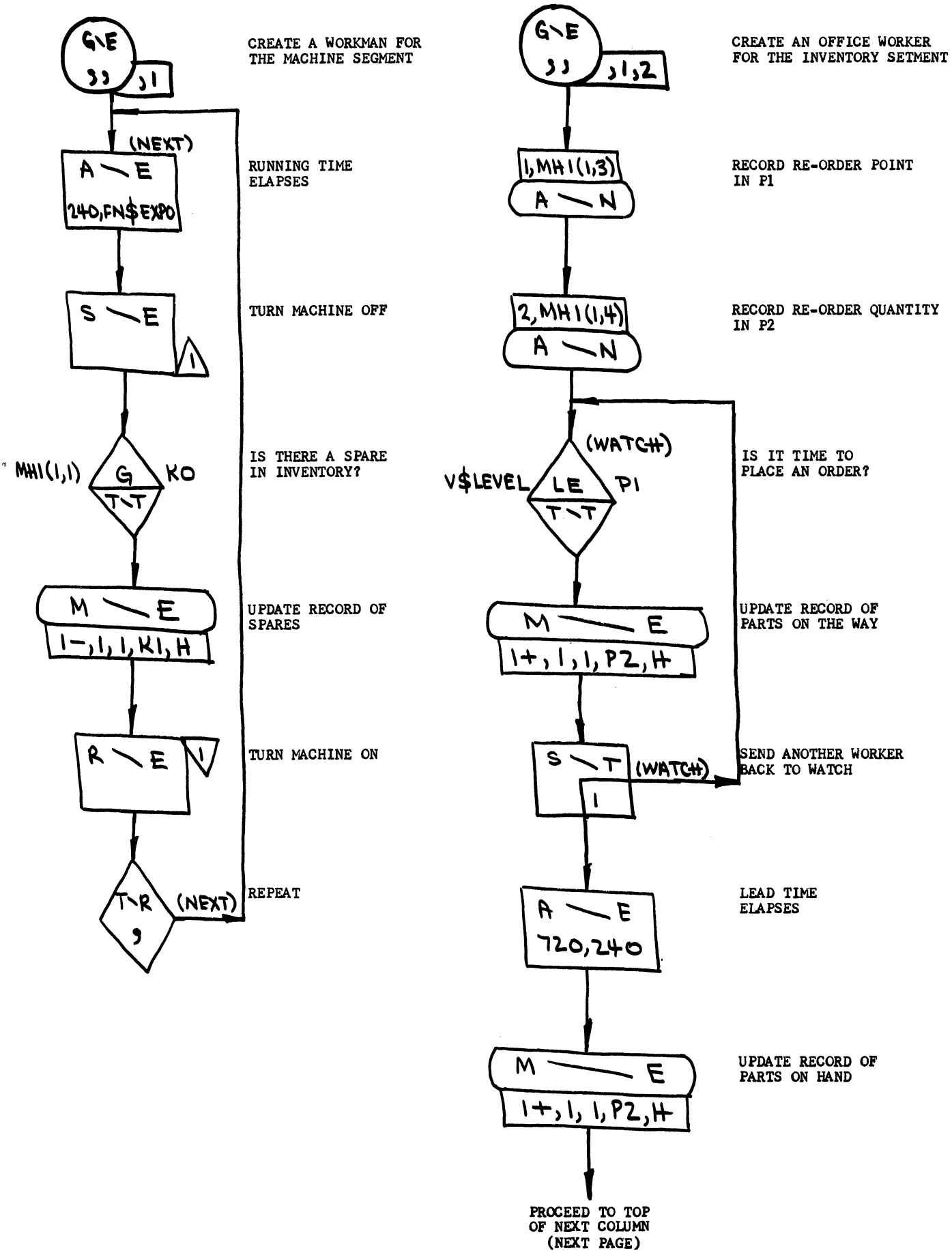


FIGURE 14-A: BLOCK DIAGRAM (CONTINUED ON NEXT PAGE)

(STOCHASTIC INVENTORY CONTROL)

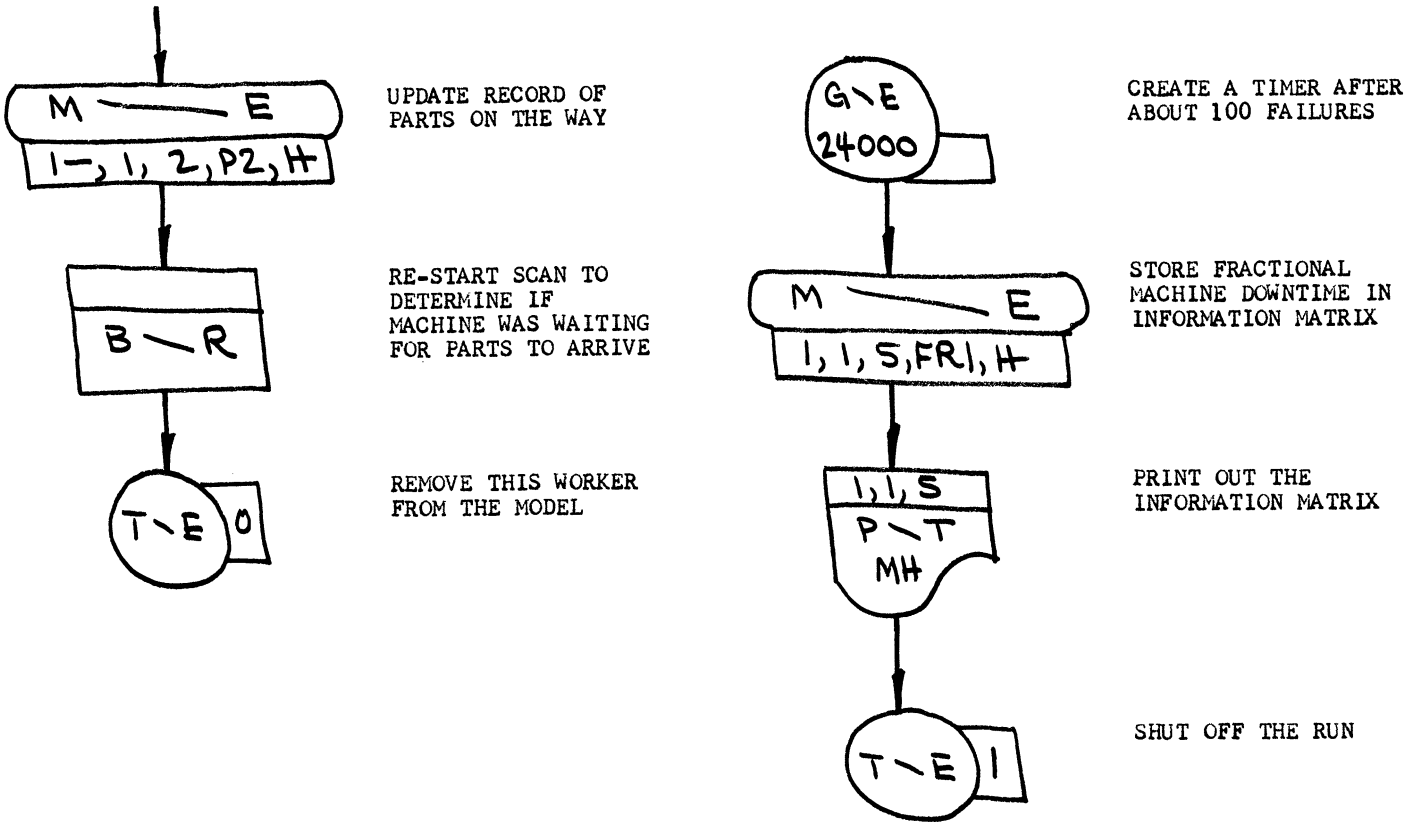


FIGURE 14-A: BLOCK DIAGRAM (CONTINUED FROM PRECEDING PAGE)
(STOCHASTIC INVENTORY CONTROL)

Now test your understanding of this by describing how Transaction priority levels can be utilized in this model, thereby making it possible to eliminate the BUFFER Block. (Answer: Give the Inventory Segment Transactions a higher priority level than the one in the Machine Segment. Then in Step 2 above, the Inventory Segment Transaction is put at the front of the Current Events Chain and MH1 (1,1) is subsequently updated before the Machine Segment Transaction tests Mh1 (1,1) against zero.)

Other Model Documentation

Figure	Information Exhibited
14-B	Program Listing
14-C	Selected Program Output; Time and Cost Data

Discussion

Based on the limited sample shown here, the best (ROP, ROQ) is (4,3), corresponding to 94.9% machine utilization. Compare this with the (4,4) machine utilization of 90.8%. Is the trend consistent? Justify your conclusion.

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
       SIMULATE
*
* THE INVENTORY CONTROL PROBLEM
* THIS MODEL WORKS WITH ONE (ROP,ROQ) COMBINATION AT A TIME.
*
 EXPO   FUNCTION   RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
     1 MATRIX     H,1,5
 LEVEL VARIABLE   MH1(1,1)+MH1(1,2)
*
* MACHINE SEGMENT
*
       GENERATE   ,,,1          CREATE A WORKMAN FOR THE MACHINE SEGMENT
 NEXT  ADVANCE    240,FN$EXPO RUNNING TIME ELAPSES
       SEIZE      1             TURN MACHINE OFF
       TEST G     MH1(1,1),K0   IS THERE A SPARE IN INVENTORY?
       MSAVEVALUE 1-,1,1,K1,H   REMOVE SPARE FROM INVENTORY
       RELEASE    1             TURN MACHINE ON
       TRANSFER   ,NEXT         REPEAT
*
* INVENTORY SEGMENT
*
       GENERATE   ,,,1,,2       CREATE OFFICE WORKER FOR THE INVENTORY SEGMENT
       ASSIGN     1,MH1(1,3)    P1 = REORDER POINT
       ASSIGN     2,MH1(1,4)    P2 = REORDER QUANTITY
 WATCH TEST LE    V$LEVEL,P1 IS IT TIME TO PLACE AN ORDER?
       MSAVEVALUE 1+,1,2,P2,H   UPDATE RECORD OF PARTS ON THE WAY
       SPLIT      1,WATCH       SEND ANOTHER WORKER BACK TO WATCH
       ADVANCE    720,240    LEAD TIME ELAPSES
       MSAVEVALUE 1+,1,1,P2,H   UPDATE RECORD OF PARTS ON HAND
       MSAVEVALUE 1-,1,2,P2,H   UPDATE RECORD OF PARTS ON THE WAY
       BUFFER                   CHECK FOR DELAYED XACT AT MACH. SEG. TEST BLOCK
       TERMINATE  0             REMOVE THIS WORKER FROM THE MODEL
*
* TIMER SEGMENT
*
       GENERATE   24000         CREATE A TIMER AFTER ABOUT 100 FAILURES
       MSAVEVALUE 1,1,5,FR1,H   STORE FRACTIONAL MACHINE DOWNTIME
       PRINT      1,1,MH,S      PRINT OUT THE DATA MATRIX
       TERMINATE  1             TURN OFF THE SIMULATION
       INITIAL    MH1(1,1),3/MH1(1,2),0/MH1(1,3),2/MH1(1,4),3
       START      1,NP          CARRY OUT THE SIMULATION
       CLEAR
       INITIAL    MH1(1,1),3/MH1(1,2),0/MH1(1,3-4),3
       START      1,NP          CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
       CLEAR
       INITIAL    MH1(1,1),3/MH1(1,2),0/MH1(1,3),4/MH1(1,4),3
       START      1,NP          CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
       CLEAR
       INITIAL    MH1(1,1),3/MH1(1,2),0/MH1(1,3),2/MH1(1,4),4
       START      1,NP          CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
       CLEAR
       INITIAL    MH1(1,1),3/MH1(1,2),0/MH1(1,3),3/MH1(1,4),4
       START      1,NP          CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
       CLEAR
       INITIAL    MH1(1,1),3/MH1(1,2),0/MH1(1,3-4),4

       START      1,NP          CARRY OUT THE SIMULATION WITH THE MODIFIED MODEL
       END
$SIGH
```

FIGURE 14-B: PROGRAM LISTING
(STOCHASTIC INVENTORY CONTROL)

```
MATRIX HALFWORD SAVEVALUE    1

        COL. 1     2     3     4     5
 ROW  1      0     3     2     3   190


MATRIX HALFWORD SAVEVALUE    1

        COL. 1     2     3     4     5
 ROW  1      3     3     3     3    75


MATRIX HALFWORD SAVEVALUE    1

        COL. 1     2     3     4     5
 ROW  1      2     3     4     3    51


MATRIX HALFWORD SAVEVALUE    1

        COL. 1     2     3     4     5
 ROW  1      0     4     2     4   108


MATRIX HALFWORD SAVEVALUE    1

        COL. 1     2     3     4     5
 ROW  1      4     0     3     4    70


MATRIX HALFWORD SAVEVALUE    1

        COL. 1     2     3     4     5
 ROW  1      2     4     4     4    92
```

```
    CPU TIME USED:

ASSEMBLY:       .995 SECONDS

EXECUTION:     3.872 SECONDS
```

```
**** ON AT  04:34.19
**** OFF AT 04:35.00
**** ELAPSED TIME        40.58    SEC.
**** CPU TIME USED         8.964  SEC.
**** STORAGE USED       383.153  PAGE-SEC.
**** CARDS READ          65
**** LINES PRINTED
**** PAGES PRINTED
**** CARDS PUNCHED        1
**** DRUM READS         224
**** APPROX. COST OF THIS RUN     $.84
```

FIGURE 14-C: SELECTED OUTPUT; TIME AND COST DATA
(STOCHASTIC INVENTORY CONTROL)

55. An Alternative Approach to the Inventory Control Problem

Statement of the Problem

Construct an alternative model for the problem of Section 54. This time build the model so that the effect of the six (ROP, ROQ) combinations can be assessed simultaneously by providing six machines operating in parallel, with each machine being "kept in spares" via one of the (ROP, ROQ) combinations.

Approach Taken in Building the Model

Figure 14-A is not simply repeated six times with minor modifications. Rather, first level addressing is used so that the basic model shown in Figure 14-A applies simultaneously to each of the six machines. An extension of Figure 14-A involving only four additional Blocks is then required for the model. The six machines involved are numbered from 1 to 6, and the corresponding (ROP, ROQ) combination used to keep the respective machines in spares is:

Machine	Applicable (ROP, ROQ)
1	(2,3)
2	(3,3)
3	(4,3)
4	(2,4)
5	(3,4)
6	(4,4)

Table of Definitions

Time Unit: 1 Hour

GPSS Entity	Interpretation
Transaction	A Shop Worker in the Machine Segment P1: Number of the machine An Office Worker in the Inventory Segment P1: Number of the machine P2: Applicable ROP P3: Applicable ROQ A Timer in the Timer Segment P1: P1 has a triple role: 1) Serves as a Looping Parameter 2) Points to the row in MH1 in which the complement of machine utilization is to be stored 3) Is the number of the Facility (machine) whose utilization (complement of "uptime") is to be stored in the Information Matrix.

Facility j	Machine j j = 1,2,3, ..., 6
Function EXPO	The inverse cumulative distribution function corresponding to the exponential probability density function
Matrix 1 (Halfword)	Information Matrix Rows 1,2,3, ..., 6 carry information about machines 1,2.3, ..., 6, respectively

Column	Information
1	Total parts on hand as spares
2	Parts on way from the supplier
3	Re-order Point
4	Re-order Quantity
5	Fractional machine downtime

Variable LEVEL	The sum of parts on hand and parts on the way for each of the six (ROP, ROQ) combinations

Block Diagram

Figure 15-A shows the model Block Diagram.

Discussion of the Block Diagram

Machine Segment:

Six Transactions circulate in the Machine Segment, one for each of the six machines. A SPLIT Block is used to achieve two results in one step:

1) Create five Transactions (after the GENERATE Block has first created one), thereby bringing the segment up to full complement, and

2) Tag the six Transactions with the machine numbers (by using the SPLIT Block's serialization option).

The SEIZE/RELEASE pair then reference the appropriate machine via P1. Also, the "parts on hand" status is determined from the proper row in the Information Matrix by using P1 as the row number.

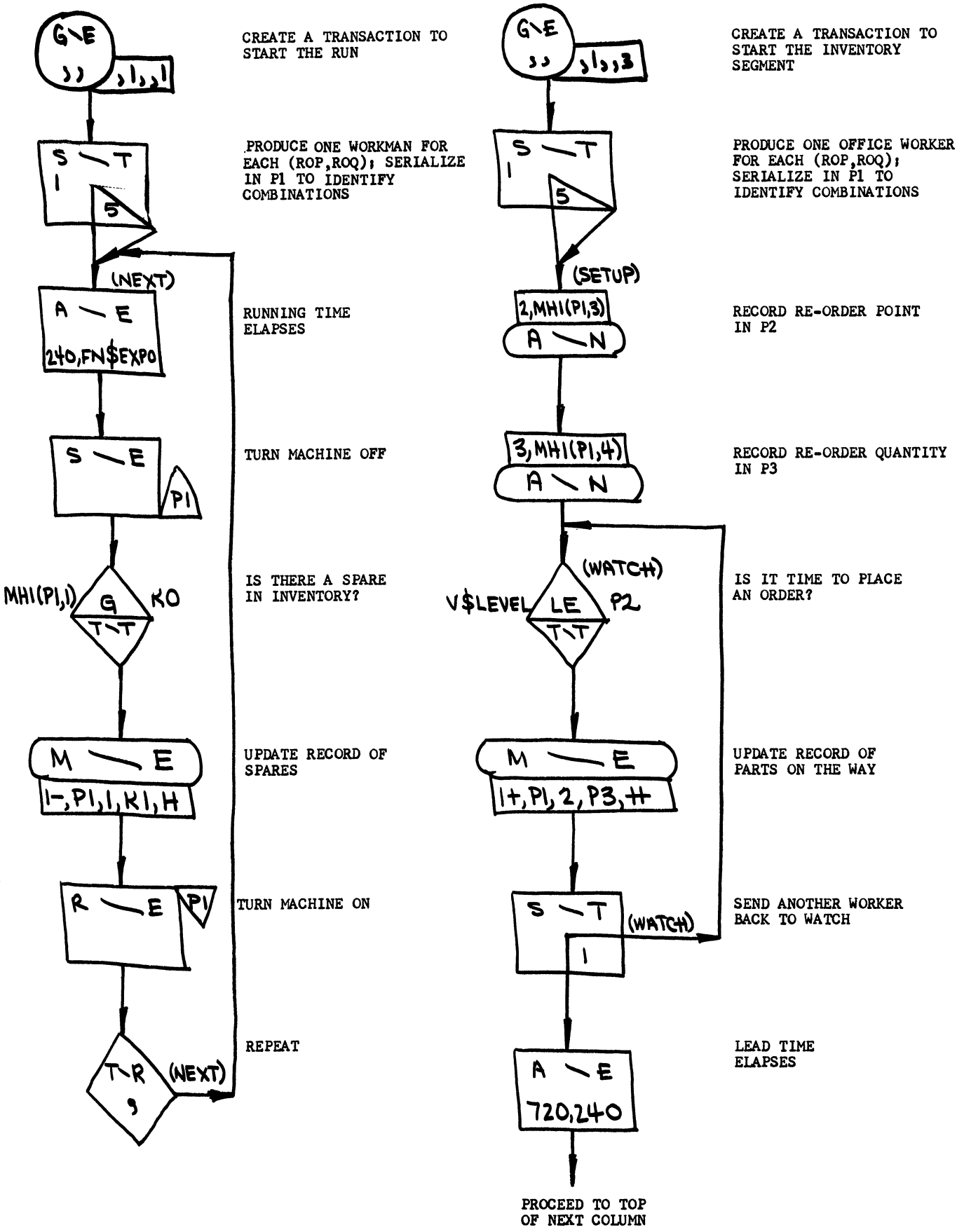
Inventory Segment:

A SPLIT Block is used for the same dual purpose as in the machine Segment. Again, P1 references the proper row in the Information Matrix.

The Variable LEVEL (see Program Listing, Figure 15-B) also makes use of P1 so that "on hand plus on the way" will be computed for the appropriate machine when the Variable is referenced.

Timer Segment:

The Timer Transaction loops to load column 5 in rows 1-6 of MH1 with the complement of machine utilizations. Note how P1 of the Transaction serves three purposes as described in the Table of Definitions.



(NEXT PAGE)

FIGURE 15-A: BLOCK DIAGRAM (CONTINUED ON NEXT PAGE)

(INVENTORY CONTROL - ALTERNATE APPROACH)

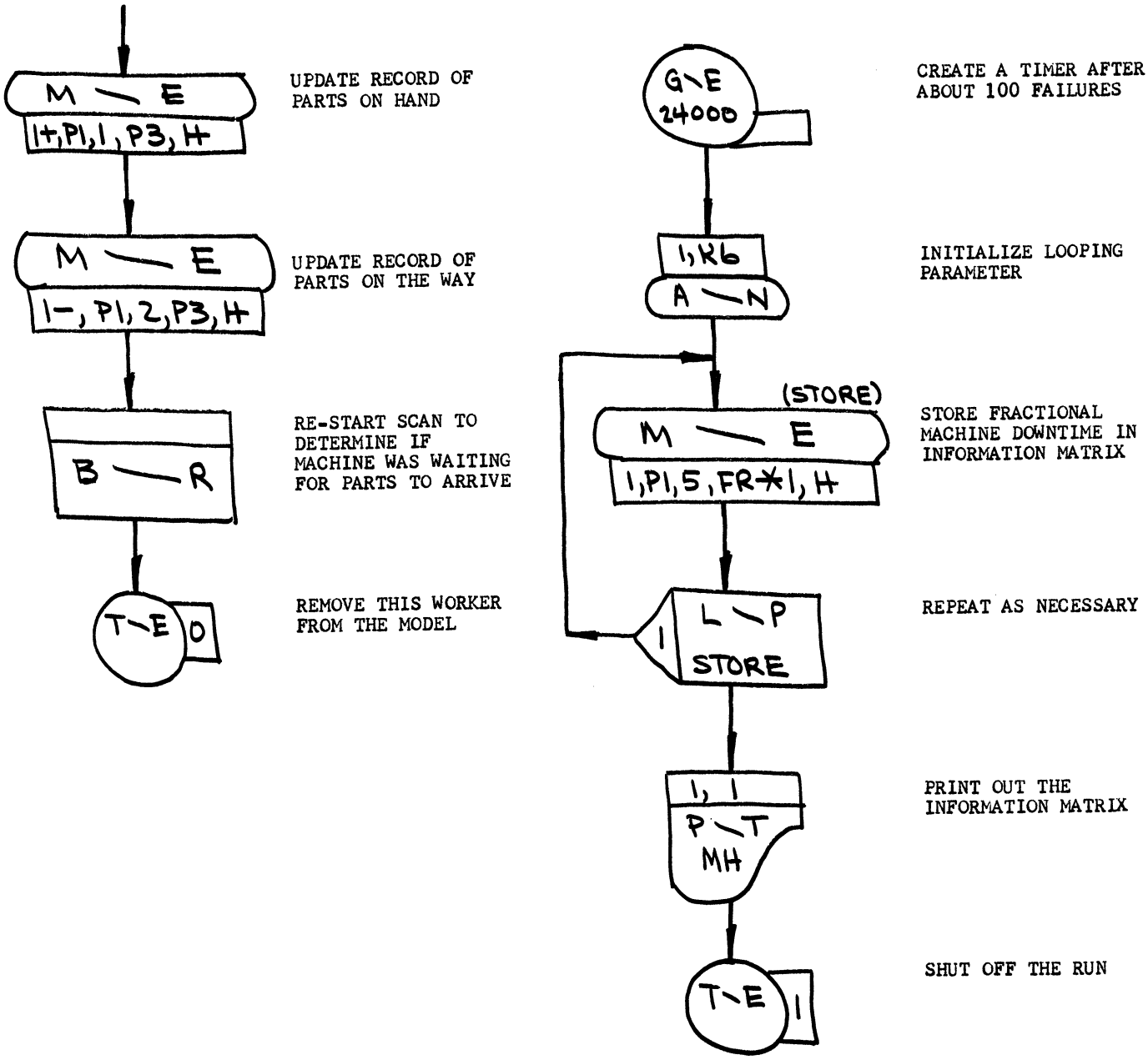


FIGURE 15-A: BLOCK DIAGRAM (CONTINUED FROM PRECEDING PAGE)
(INVENTORY CONTROL - ALTERNATE APPROACH)

Other Model Documentation

Figure	Information Exhibited
15-B	Program Listing
15-C	Selected Program Output; Time and Cost Data

Discussion of Output

The (ROP, ROQ) combination of (4,4) appears to be the best, based on the sample taken. Contrast this with the output in Figure 14-C.

This table shows time data in seconds, for this model and the model of Section 54:

	Assembly	Execution
Sequential Model	.995	3.872
Simultaneous Model	.889	10.184

The Assembly Time difference is essentially the same and will not be discussed. The simultaneous model executed much more slowly than its sequential counterpart. Some light can be shed on this sluggishness:

> At each simulated clock instant, the test at WATCH is conducted a minimum of six times in the Simultaneous model (the case when no parts come in and no order is placed for any of the six machines). If an order comes in at the clock instant, the Current Events Chain scan is re-started (via the BUFFER), requiring that the test be conducted another six times. Furthermore, more than one order might arrive in a given clock instant (either two or more for a given machine, or an order for each of two or more machines, or ...). Compare this with the sequential model, where the test is conducted once (if no parts come in and no order is placed), or twice (if an order comes in but no order is placed), etc. Now, a bit of reflection shows that the simulation clock is moved forward almost as many different times in the simultaneous simulation as in the sequential run covering all six combinations. The relatively slow execution of the simultaneous model should now be plausible.

How might the simulataneous model be made to run faster? One possility is to replace the BUFFER Block approach with the priority level concept as discussed at the end of Section 54. This replacement was in fact made and the slightly modified model was run. These are the resulting times:

Assembly	Execution
.949	9.327

This is not much help. In Section 63 this matter of execution time economy is considered further and the capability of influencing it significantly is introduced.

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
       SIMULATE
*
* THE INVENTORY CONTROL PROBLEM
* THIS MODEL INVESTIGATES SIX (ROP,ROQ) COMBINATIONS SIMULTANEOUSLY.
*
 EXPO  FUNCTION   RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
     1 MATRIX     H,6,5
 LEVEL VARIABLE   MH1(P1,1)+MH1(P1,2)
*
* MACHINE SEGMENT
*
       GENERATE   ,,,1,,1       CREATE A MASTER XACT FOR THE MACHINE SEGMENT
       SPLIT      5,NEXT,1      ONE XACT FOR EACH (ROP,ROQ), SERIALIZE IN P1
 NEXT  ADVANCE    240,FN$EXPO RUNNING TIME ELAPSES
       SEIZE      P1            TURN MACHINE OFF
       TEST G     MH1(P1,1),K0  IS THERE A SPARE IN INVENTORY?
       MSAVEVALUE 1-,P1,1,K1,H  REMOVE SPARE FROM INVENTORY
       RELEASE    P1            TURN MACHINE ON
       TRANSFER   ,NEXT         REPEAT
*
* INVENTORY SEGMENT
*
       GENERATE   ,,,1,,3       CREATE A MASTER XACT FOR THE INVENTORY SEGMENT
       SPLIT      5,SETUP,1     ONE XACT FOR EACH (ROP,ROQ), SERIALIZE IN P1
 SETUP ASSIGN     2,MH1(P1,3)   P2 = REORDER POINT
       ASSIGN     3,MH1(P1,4)   P3 = REORDER QUANTITY
 WATCH TEST LE    V$LEVEL,P2 IS IT TIME TO PLACE AN ORDER?
       MSAVEVALUE 1+,P1,2,P3,H  UPDATE RECORD OF PARTS ON THE WAY
       SPLIT      1,WATCH       SEND ANOTHER WORKER BACK TO WATCH
       ADVANCE    720,240     LEAD TIME ELAPSES
       MSAVEVALUE 1+,P1,1,P3,H  UPDATE RECORD OF PARTS ON HAND
       MSAVEVALUE 1-,P1,2,P3,H  UPDATE RECORD OF PARTS ON THE WAY
       BUFFER                   CHECK FOR DELAYED XACT AT MACH. SEG. TEST BLOCK
       TERMINATE  0             REMOVE THIS WORKER FROM THE MODEL
* TIMER SEGMENT
*
*
       GENERATE   24000         CREATE A TIMER AFTER ABOUT 100 FAILURES
       ASSIGN     1,K6          INITIALIZE LOOPING PARAMETER
 STORE MSAVEVALUE 1,P1,5,FR*1,H STORE RESULTS IN ROW P1 OF MH1
       LOOP       1,STORE       REPEAT UNTIL ALL RESULTS ARE STORED
       PRINT      1,1,MH        PRINT OUT THE DATA MATRIX
       TERMINATE  1             TURN OFF THE SIMULATION
       INITIAL    MH1(1-6,1),3/MH1(1-6,2),0/MH1(1,3),2/MH1(2,3),3
       INITIAL    MH1(3,3),4/MH1(4,3),2/MH1(5,3),3/MH1(6,3),4
       INITIAL    MH1(1-3,4),3/MH1(4-6,4),4
       START      1,NP          CARRY OUT THE SIMULATION
       END
$SIGH
```

FIGURE 15-B: PROGRAM LISTING

(INVENTORY CONTROL - ALTERNATE APPROACH)

```
MATRIX HALFWORD SAVEVALUE    1

          COL.  1     2     3     4     5
ROW  1          4     0     2     3   145
     2          3     3     3     3   111
     3          3     3     4     3    68
     4          0     4     2     4   152
     5          1     4     3     4    89
     6          1     4     4     4    46
```

```
                                  **** ON AT  04:33.36
                                  **** OFF AT 04:34.15
                                  **** ELAPSED TIME        38.103  SEC.
      CPU TIME USED:              **** CPU TIME USED       15.208  SEC.
                                  **** STORAGE USED       709.553  PAGE-SEC.
ASSEMBLY:      .889 SECONDS       **** CARDS READ          56
                                  **** LINES PRINTED
EXECUTION:   10.184 SECONDS       **** PAGES PRINTED
                                  **** CARDS PUNCHED        1
                                  **** DRUM READS         116
                                  **** APPROX. COST OF THIS RUN    $1.35
```

FIGURE 15-C: SELECTED OUTPUT; TIME AND COST DATA

(INVENTORY CONTROL - ALTERNATE APPROACH)

56. Determination of Transit Times

*** The transit time of a Transaction is the number of simulated time units that elapse while the Transaction moves between two points in a simulation model, say from Point A to Point B.

*** Transit time is easily computed by "saving the clock" when the Transaction reaches Point A, then subtracting this saved value from the clock reading when the Transaction arrives at Point B.

*** The "clock" referred to above cannot necessarily be the Relative Clock, because it is set back to zero whenever a RESET Card is encountered. If this happens while a Transaction is somewhere between A and B, the transit time subsequently computed will clearly be in error. This problem does not arise with the Absolute Clock.

*** Yet, as stated in Section 19, the value of the Absolute Clock is not available during a simulation run. GPSS consequently provides a roundabout way for the programmer to get at the difference between two Absolute Clock readings. This involves two steps:

1) Having the Absolute Clock saved at Point A, either:

   a) In a special storage location associated with each Transaction and known as Mark Time, or

   b) In one of the Transaction's Parameters, then

2) Triggering automatic computation of the time difference at Point B by using the Standard Numerical Attribute whose value is the transit time. The Standard Numerical Attribute that applies will be either one of two possibilities, per the two different ways the Absolute Clock could have been saved at Point A.

We now discuss the specifics involved.

*** Saving the Absolute Clock at Point A is accomplished with the MARK Block:

The MARK Block:

Purpose: Cause the Absolute Clock reading to be stored in a Transaction's Mark Time or in one of its Parameters when the Transaction moves through the Block

Shape and Operands:

Operand	Significance
A	Optional operand; If A is not used, the clock reading is placed in Mark Time; If A is used, it specifies the number of the Parameter in which the clock reading will be placed.

Note: The Mark Time storage is initialized with the Absolute Clock's value when a Transaction is created at a GENERATE Block. (Offspring at a SPLIT Block are automatically given the same Mark Time as their parent.) Hence, when Point A is the point of entry to the model, the MARK Block does not have to be used.

*** Triggering automatic computation of the time difference at Point B is accomplished either by:

1) Using M1 if the Absolute Clock has been saved in Mark Time, or

2) Using MPj if the Absolute Clock has been saved in Parameter j.

*** M1 and MPj are members of the set of Standard Numerical Attributes.

57. Tabulation of Frequency Distributions

*** Simulation involves drawing random samples from populations. It is often of interest to use these samples to estimate properties of the source population: mean, standard deviation, probability density function, cumulative distribution function, and so on. One step toward this end may involve computing these very properties for the sample itself. GPSS simplifies this computation by providing an entity known as user-defined Tables. The user can define one or more unique Tables for each model random variable of interest, specifying the size of a typical frequency class and the total number of frequency classes in the sample space. Table entries are then made under the control of programmer-supplied logic. At the end of a simulation run (or during the course of a simulation, as discussed in Section 53), such Table statistics as mean, standard deviation, relative frequency, and cumulative frequency are printed for each user-defined Table.

*** As suggested above, construction of a GPSS Table is a two-step process:

1) The user defines the existence of the Table, specifying:

   a) Table number,

   b) The random variable being tabulated,

   c) Inclusive upper limit of the lowest frequency class (lower limit of the lowest frequency class is $-\infty$),

   d) Frequency class width (the same for all frequency classes, with the exception of the lowest and the highest), and

   e) The total number of frequency classes (upper limit of the highest frequency class is $+\infty$).

2) The user then builds into his model one or more appropriate Blocks, known as TABULATE Blocks, which reference the Table. Each time a Transaction moves through such a Block, the current value of the random variable mentioned in 1-b above is entered into the Table.

We proceed to detail the specifics involved in each of the two steps.

*** The existence of a Table is declared to the simulator with a Table Definition Card following this format:

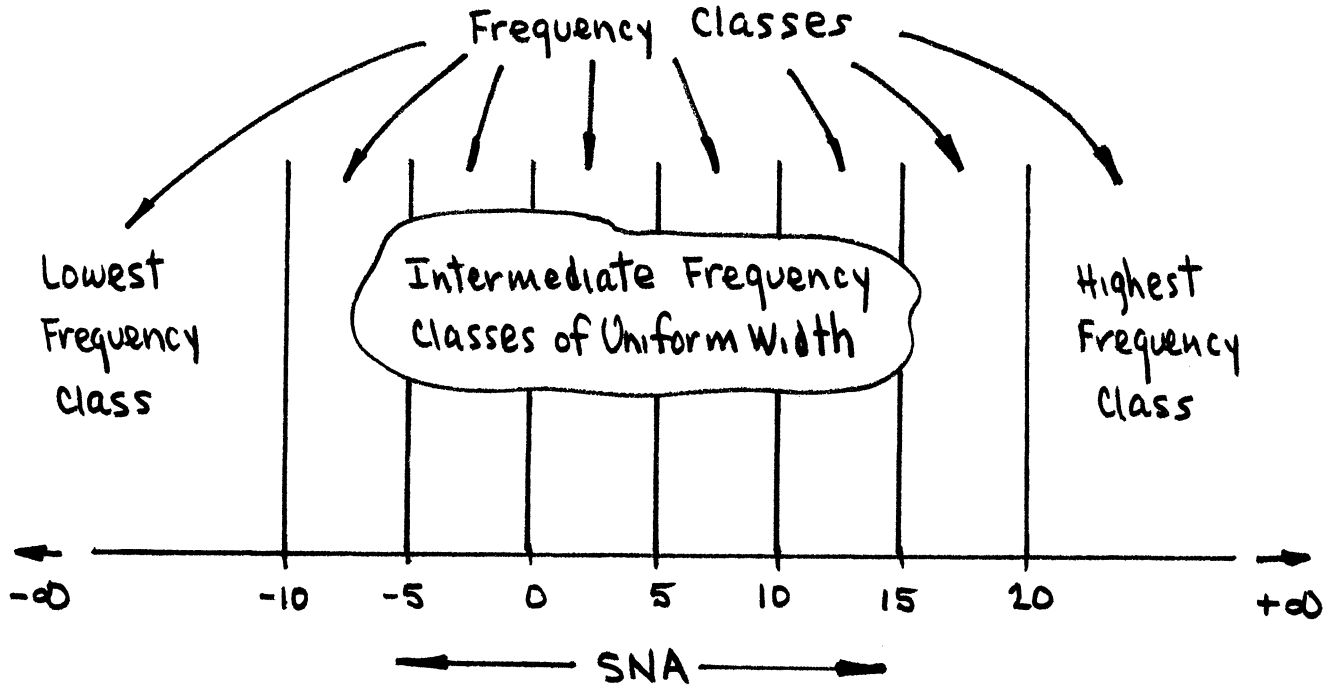
card columns:	2 ... 6	8 ... 18	19 ...
content:	j	TABLE	A,B,C,D,E

where:

j is the number (or symbolic name) of the Table being defined, and the Operands have this significance:

Operand	Significance
A	Specifies the Standard Numerical Attribute (random variable) to be tabulated; or, alternatively, specifies "table mode" (to be discussed below)
B	Specifies the inclusive upper limit of the lowest frequency class
C	Specifies the width of all frequency classes (except the two bounding classes)
D	Specifies the total number of frequency classes
E	Optional operand; used only when the A Operand is the "table mode" mnemonic TR. (to be discussed below)

The frequency class concept can be depicted diagrammactically this way:



Suppose that transit time is to be the Standard Numerical Attribute entered in the above Table, and that the Table is Table 3. Then the Table Definition Card would be:

```
3  TABLE  M1,-10,5,8
```

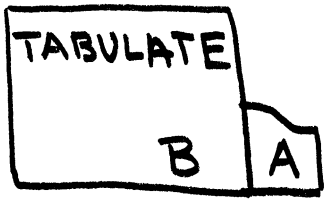
These entries should be checked against the definitions given above. (The M1 might have been MPj, depending on the considerations set forth in Section 56.)

*** Table entries are made with use of the TABULATE Block.

The TABULATE Block:

Purpose: Each time a Transaction moves through a TABULATE Block, an entry is made in a Table.

Shape and Operands:



Operand	Significance
A	Specifies the number (or symbolic name) of the Table in which the entry is to be made.
B	Optional operand; specifies the weight of the entry, i.e. effectively, "how many times" that entry is to count. If B is not used, a weight of 1 is assumed. If B is an integer exceeding 1, the first character in the D operand of the corresponding Table Definition Card must be a W (actually, any alphabetic character).

*** In addition to the Standard Numerical Attributes, there are three other items that can be tabulated via their use as the A Operand in the Table Definition Card. These three possibilities are known as "table modes." They are:

1) Difference Mode: when the last character in the A Operand is a -, then not the quantity itself but the difference between the quantity and its value when the Table was last referenced is tabulated. The first reference to the Table, then, does not result in a Table entry.

2) Interarrival Mode: when the A Operand is IA, the tabulated quantity is the time interval since the last reference to the Table.

3) Arrival Rate Mode: when the A Operand is RT, a reference to the Table does not produce an entry in the table. Rather, a Table entry is made once every certain number of time units. The table entry is the value of an "Arrival Count" maintained by the simulator. Its value is initially zero. Each time the Table is referenced, the Arrival Count is incremented by the TABULATE Block's B Operand. When it is time for the next Table entry, the value of the Arrival Count is tabulated. Then the Arrival Count is set back to zero and the process continues.

   The Table Definition Card E Operand specifies the time interval that is to elapse between table entries.

*** As indicated above, Tables are established with the Table Definition Card, then maintained with the TABULATE Block. There is one exception to this generalization. That is, there is one type of Table in which entries are made without use of the TABULATE Block. It is only necessary for the programmer to establish the Table with a Table Definition Card. It is then automatically maintained by the simulator. The quantity tabulated is the time spent by various Transactions in Queues. (Note that Queue statistics include average waiting times but give no information as to the distribution of these times.) The Table Definition Card takes this form:

card columns:	2–6	8–18	19–----
contents:	j	QTABLE	A,B,C,D

where:

j is the number (or symbolic name) of the Table,

A is the number (or symbolic name) of the Queue whose waiting time distribution is to be tabulated, and

B,C,D have the usual Table Definition Card significance.

58. Table Statistics Available During the Course of a Simulation

***Several properties of Tables can be referenced as a simulation proceeds, making information available that can be used in the logic of a model. The three Table Standard Numerical Attributes are:

1) TBj: the mean value of entries made in Table j to date

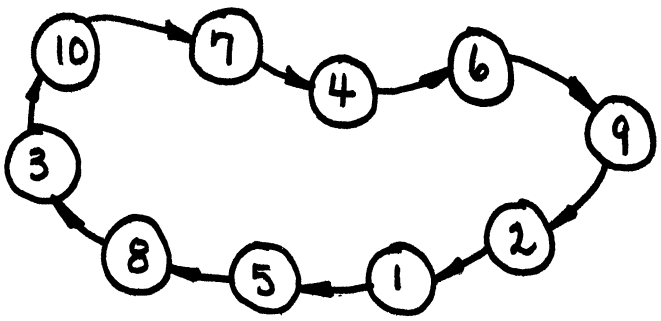
2) TCj: the number of entries made in Table j to date

3) TDj: the standard deviation of entries made in Table j to date

59. The Traveling Salesman Problem

Statement of the Problem

A Traveling Salesman must visit cities numbered from 1 to 10. He can visit them in any order, but must wind up at the city in which he started, having visited the other 9 cities exactly one time each. The cost of traveling between two cities equals the square of the difference of the city numbers. For example, if the route is as shown, the cost of completing the circuit is $180.



The objective is to determine that route which minimizes the total cost of completing a circuit through the ten cities. Build a GPSS model which will generate possible routes randomly and compute the associated cost. Tabulate the costs in an appropriate Table. Each time a lower-cost route is found, save the cost and the route itself. Print out both the cost table and the record of lower-cost routes after 250 routes have been randomly produced.

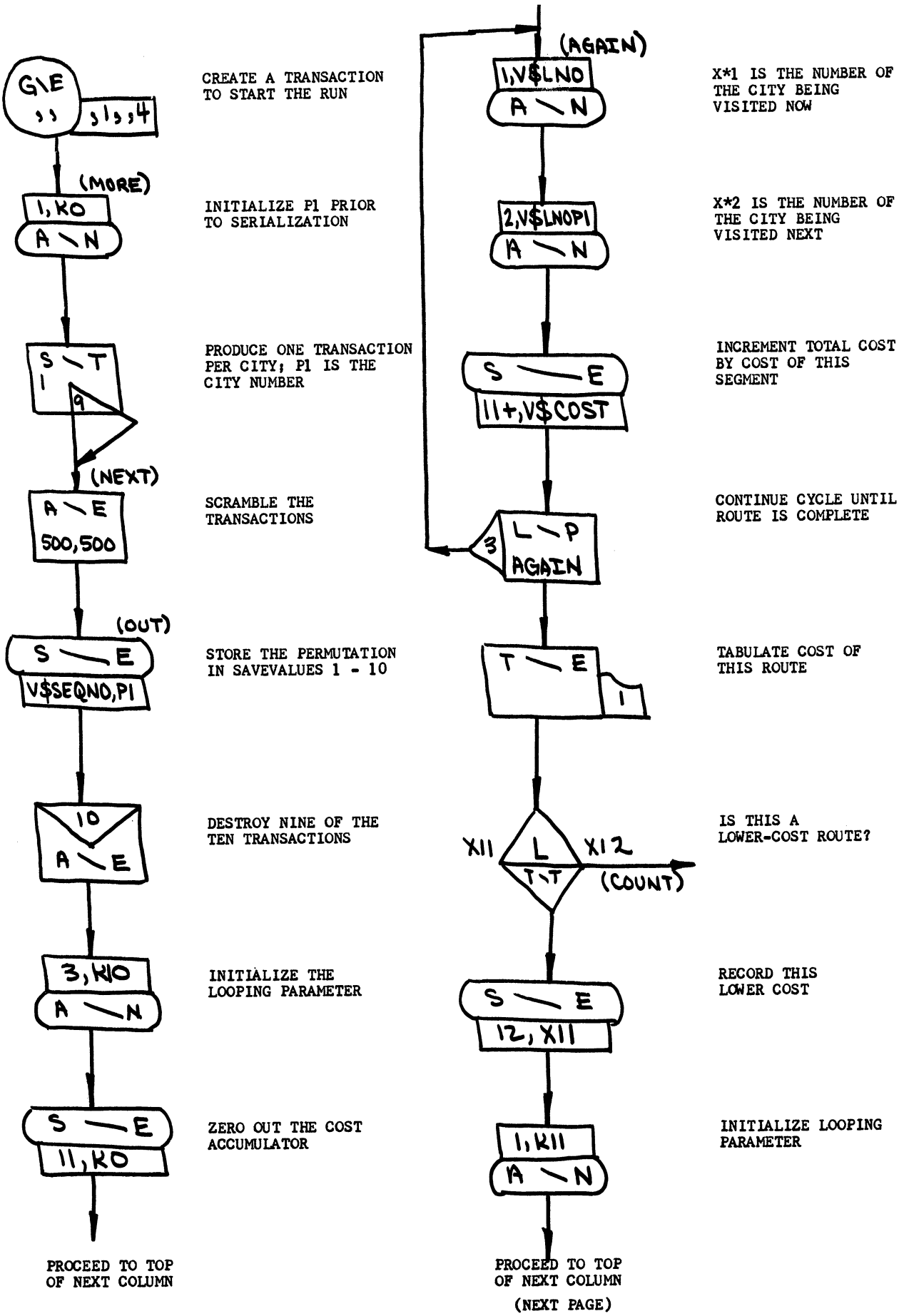
Approach Taken in Building the Model

The problem of generating random routes is essentially the problem of permuting the integers from 1 to 10 in random fashion. The permutation is accomplished by sending a single Transaction into a SPLIT block, where 9 offspring are produced with serialization from 1 to 10 in P1. The 10 Transactions are then scrambled by sending them into an ADVANCE block, where holding time can be any of the values from 0 to 1000. As each of the 10 Transactions exits the ADVANCE block, the corresponding P1 value is stored in Savevalues 1 through 10, respectively. All but one of the 10 Transactions is then destroyed. One Transaction goes on to compute the cost of the route, make the Table entry, and check to see if this is a "lower-cost" case. After lower-cost updating as may be required, the one Transaction is routed back to the original SPLIT block to randomly generate a next route. This is continued until 250 routes (not necessarily all different) have been produced. Then the cost Table and the Matrix of "better routes" are printed out and the simulation ends.

Table of Definitions

Time Unit: No Interpretation Necessary

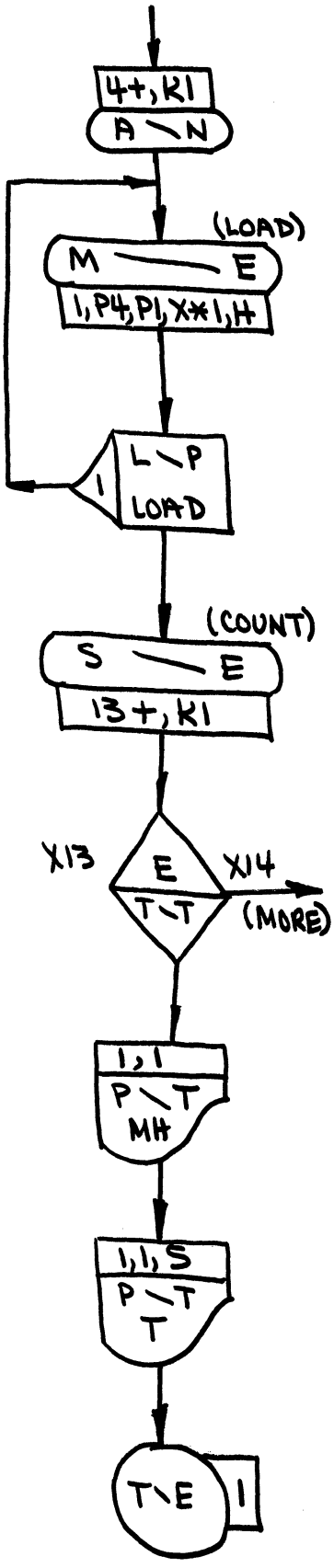
GPSS Entity	Interpretation
Transaction	A Brute Force Experimenter P1: A city number (an integer from 1 to 10) P2: The number of a Savevalue (an integer from 1 to 10) in which a city number is found P3: Looping Parameter controlling computation of route cost P4: A Counter, used to count the total number of routes generated
Savevalue j j = 1,2,...,10	Location of the permutation of the integers from 1 to 10
Savevalue 11	An Accumulator, used to accumulate the cost of a route
Savevalue 12	Lowest cost to date, given an initially high value
Matrix 1	The n-by-11 Matrix in which each better route and the associated cost are saved, where n equals the number of better routes that are encountered during the simulation. Columns 1 through 10 contain the permutation representing the route. Column 11 is the corresponding cost. n is unknown so a deliberately large value is chosen for it.
Table 1	The Table in which cost information is entered.
Variable COST	The cost of traveling from the j-th city visited to the j+1-st city visited
Variable LNO	The number of the Savevalue in which is found the number of the city currently being visited.
Variable LNOP1	The number of the Savevalue in which is found the number of the next city to be visited. Takes on the values 2, 3, 4, ... , 9, 10, 1 as the Variable LNO takes on the values from 1 to 10, respectively.
Variable SEQNO	A Variable taking on the values from 1 to 10, representing the chronological order in which Transactions exit the ADVANCE Block which scrambles them



(NEXT PAGE)

FIGURE 16-A: BLOCK DIAGRAM (CONTINUED ON NEXT PAGE)

(THE TRAVELING SALESMAN PROBLEM)



UPDATE NUMBER OF LOWER-COST ROUTES FOUND

RECORD PART OF THE ROUTE (OR COST) IN MH1

CONTINUE CYCLE UNTIL RECORD IS COMPLETE

UPDATE COUNT OF TOTAL ROUTES PRODUCED THUS FAR

ARE MORE ROUTES TO BE PRODUCED?

PRINT OUT THE INFORMATION MATRIX

PRINT OUT THE TABULATED COSTS

SHUT OFF THE SIMULATION

FIGURE 16-A: BLOCK DIAGRAM (CONTINUED FROM PRECEDING PAGE)
(THE TRAVELING SALESMAN PROBLEM)

Block Diagram

The Block Diagram is shown in Figure 16-A.

Discussion of the Block Diagram

Most features of the model will be evident based on description of the approach taken in building the model. Some points are worth making though. Note that the Transaction routed back to begin producing the next route has its P1 zeroed out at MORE. This initializes the P1 prior to using it for serialization in the SPLIT block. The step is redundant the first time it is carried out. Also notice that the Variables SEQNO, LNO, and LNOP1 make use of modulus arithmetic so that these Variables will take on the values 1 to 10, 1 to 10, and 2 through 10 to 1, respectively, independent of the number of routes which have been produced. The Variable COST uses Indirect Addressing and appears as it does because of the lack of an exponentiation operator.

Other Model Documentation

Figure	Information Exhibited
16-B	Program Listing
16-C	Selected Program Output; Time and Cost Data

Discussion of Output

Output Table 1 shows that the average route cost is $178.60. Row 7 in Matrix 1 indicates that the lowest cost route encountered during the simulation was $62, about 2.4 standard deviations below the mean. Table 1 shows that routes costing $75 or less came up only two times. Rows 6 and 7 in Matrix 1 show the details of those two routes.

Perhaps the chief value of this model is its demonstration of several techniques, it particular the use of modulus arithmetic, and utilization of the ADVANCE Block to produce a random permutation.

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
       SIMULATE
*
* THE TRAVELING SALESMAN PROBLEM
*
     1 MATRIX     H,10,11
     1 TABLE      X11,50,25,12
 COST  VARIABLE   (X*1-X*2)*(X*1-X*2)
 LNO   VARIABLE   N$AGAIN@K10+K1
 LNOP1 VARIABLE   P1@K10+K1
 SEQNO VARIABLE   N$OUT@K10+K1
*
       GENERATE   ,,,1,,4          CREATE AN XACT TO START THE MODEL
 MORE  ASSIGN     1,K0             INITIALIZE P1 PRIOR TO SERIALIZATION
       SPLIT      9,NEXT,1         ONE XACT PER CITY, P1 = CITY NUMBER
 NEXT  ADVANCE    500,500          SCRAMBLE THE ORDER OF THE CITIES
 OUT   SAVEVALUE  V$SEQNO,P1       STORE THE PERMUTATION IN SAVEVALUES 1 - 10
       ASSEMBLE   10               DESTROY NINE OF THE TEN TRANSACTIONS
       ASSIGN     3,K10            ROUTE CONSISTS OF TEN SEGMENTS
       SAVEVALUE  11,K0            ZERO OUT THE COST ACCUMULATOR
 AGAIN ASSIGN     1,V$LNO          X*1 IS NUMBER OF CITY BEING VISITED NOW
       ASSIGN     2,V$LNOP1        X*2 IS NUMBER OF CITY BEING VISITED NEXT
       SAVEVALUE  11+,V$COST       INCREMENT TOTAL COST BY COST OF THIS SEGMENT
       LOOP       3,AGAIN          REPEAT AS NECESSARY
       TABULATE   1                TABULATE THE COST OF THIS ROUTE
       TEST L     X11,X12,COUNT    IS THIS A LOWER-COST ROUTE?
       SAVEVALUE  12,X11           RECORD THIS LOWER COST
       ASSIGN     1,K11            PREPARE TO STORE ROUTE AND ITS COST
       ASSIGN     4+,K1            UPDATE NUMBER OF LOWER-COST ROUTES FOUND
 LOAD  MSAVEVALUE 1,P4,P1,X*1,H    TRANSFER PERMUTATION AND COST TO MATRIX 1
       LOOP       1,LOAD           REPEAT AS NECESSARY
 COUNT SAVEVALUE  13+,K1           UPDATE TOTAL ROUTES PRODUCED TO DATE
       TEST E     X13,X14,MORE     ARE MORE ROUTES TO BE PRODUCED?
       PRINT      1,1,MH           PRINT OUT THE INFORMATION MATRIX
       PRINT      1,1,T,S          PRINT OUT THE TABULATED COSTS
       TERMINATE  1                SHUT OFF THE SIMULATION
       INITIAL    X12,999/X14,250
       START      1,NP             CARRY OUT THE SIMULATION
       END                         RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                              SIGN-OFF
```

FIGURE 16-B: PROGRAM LISTING
(THE TRAVELING SALESMAN PROBLEM)

```
MATRIX HALFWORD SAVEVALUE    1

        COL.  1    2    3    4    5    6    7    8    9   10   11
ROW  1         1    9    5    2    3    7    8   10    6    4  140
     2         7    8    4    5   10    9    2    3    1    6  124
     3         7    4    8    9   10    5    2    6    1    3  122
     4         7    6    4    2    3    5    1    8    9   10   90
     5        10    6    1    2    3    4    7    5    9    8   78
     6         2    1    5    3    4    8   10    9    7    6   64
     7         3    1    4    6    5    8   10    9    7    2   62
     8         0    0    0    0    0    0    0    0    0    0    0
     9         0    0    0    0    0    0    0    0    0    0    0
    10         0    0    0    0    0    0    0    0    0    0    0
```

```
TABLE   1
ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION          SUM OF ARGUMENTS
            250                 178.599                      48.812                  44650.000

      UPPER       OBSERVED      PER CENT      CUMULATIVE      CUMULATIVE      MULTIPLE      DEVIATION
      LIMIT       FREQUENCY     OF TOTAL      PERCENTAGE      REMAINDER       OF MEAN       FROM MEAN
         50               0          .00              .0           100.0          .279         -2.634
         75               2          .79              .7            99.1          .419         -2.122
        100              10         3.99             4.7            95.1          .559         -1.610
        125              29        11.59            16.3            83.6          .699         -1.098
        150              33        13.19            29.5            70.4          .839          -.585
        175              43        17.19            46.7            53.2          .979          -.073
        200              52        20.79            67.5            32.4         1.119           .438
        225              40        15.99            83.5            16.4         1.259           .950
        250              19         7.59            91.1             8.8         1.399          1.462
        275              15         5.99            97.1             2.8         1.539          1.974
        300               7         2.79           100.0              .0         1.679          2.487
REMAINING FREQUENCIES ARE ALL ZERO
```

```
    CPU TIME USED:

ASSEMBLY:       .707 SECONDS

EXECUTION:    11.911 SECONDS
```

```
**** ON AT  23:36.42
**** OFF AT 23:37.26
**** ELAPSED TIME        43.266   SEC.
**** CPU TIME USED       16.826   SEC.
**** STORAGE USED        769.75   PAGE-SEC.
**** CARDS READ          43
**** LINES PRINTED
**** PAGES PRINTED
**** CARDS PUNCHED
**** DRUM READS          69
**** APPROX. COST OF THIS RUN     $1.44
```

FIGURE 16-C: SELECTED OUTPUT; TIME AND COST DATA
(THE TRAVELING SALESMAN PROBLEM)

60. The Lathe Department Problem

Statement of the Problem

A certain job shop has been experiencing long delays in jobs going through the lathe department because of inadequate capacity. The foreman contends that five machines are required, as opposed to the three machines that he now has. However, because of pressure from management to hold capital expenditures down, only one additional machine will be authorized unless there is solid evidence that a second one is necessary.

This shop does three kinds of jobs, namely: government jobs; commercial jobs; and standard products. Whenever a lathe finishes a job, it starts a government job if one is waiting; if not, it starts a commercial job if any are waiting; if not, it starts on a standard product if any are waiting. Jobs of the same type are taken on a first-come, first-served basis.

Although much overtime work is required currently, management wants the lathe department to operate on an 8-hour, 5-day-a-week basis. The probability distribution of the time required by a lathe for a job appears to be approximately exponential with a mean of ten hours. Jobs come into the shop according to a Poisson input process, but at a mean rate of seven per week for government jobs, five per week for commercial jobs, and two per week for standard products. These figures are expected to remain the same for the indefinite future.

Jobs are promised to be finished 24 working hours after their arrival at the lathe department. Lateness penalties of $300, $150, and $30 per day apply to government, commercial, and standard jobs, respectively. The penalty for fractional days late is allocated proportionately. The incremental capitalized cost of providing each lathe (including the operator, etc.) is $100 per working day.

Develop and use a GPSS model to simulate this situation and make recommendations on the basis of the results.

Approach Taken in Building the Model

Two lathe departments are set up in parallel in a Lathe Department Segment, with the departments having 4 and 5 lathes, respectively. Only one GENERATE block is used to feed jobs to the model. When a job arrives, its type is determined by sampling from a function and the priority level is assigned accordingly. Then the SPLIT block is used so that each arrival to the model becomes an arrival to each of the two parallel configurations. Hence, the GENERATE/SPLIT sequence guarantees that the two departments experience the same job arrival rate. The machining operation is then simulated in straightforward fashion. If the finished job was in the shop more than 24 working hours, its lateness time is entered in the appropriate one of six tables (one table per job type per department).

Also notice that between establishing job priority and SPLITing, machining time is assigned to P2. This is done before the SPLIT for two reasons:

1) The time required by any particular job in both of the two parallel departments will be the same. Consequently each department experiences the same corresponding machine time requirements, making the simulation results easier to compare directly.

2) Computer time is saved. If the ASSIGN Block were positioned after the SPLIT, twice as many Transactions would move through it as do for the configuration shown.

In a Timer Segment, the cost consequences of lateness time experience are computed and stored in a two-by-four matrix. The two rows represent the two shops. The first three columns represent standard, commercial, and government jobs, respectively. The fourth column represents combined experience for the three job types. Matrix entires are cost penalties in dollars per week. Entries in the fourth column can then be used directly for comparison of the two cases, keeping in mind that marginal cost of providing the additional lathe capacity must still be taken into account. The importance of making the critical comparison on the basis of steady state costs is recognized. The model is built with use of the RESET Control Card in mind, so that the path followed through transient conditions toward steady state operation can be traced.

Table of Definitions

Time Unit: 1 Minute

GPSS Entity	Interpretation
Transaction	Lathe Department Segment: A job P1: 1 or 2 for departments with 4 or 5 lathes, respectively Timer Segment: Timer and Accountant P1: 1 or 2 for departments with 4 or 5 lathes, respectively P2: Job Type Code Value — Job Type 1 — Standard 2 — Commercial 3 — Government P3: Table Number for Lateness Data; Ranges from 1 to 6 in value
Function EXPO	The inverse cumulative distribution function corresponding to the exponential probability density function
Function SETPR	Function for sampling from the job type distribution
Matrix 1 (Fullword)	The two-by-four matrix in which cost penalties (dollars per week) are entered, broken down by lathe department and job type, and also appearing cumulatively by lathe department Rows 1 and 2 represent departments 1 and 2, respectively. Columns 1 through 4 are standard, commercial, government, and cumulative cost penalties, respectively.

Table j j=1,2,...,6	Lateness Time Tables The correspondence between Table number, lathe department, and job type is shown in this table:

		Job Type		
		1	2	3
Lathes:	4	1	2	3
	5	4	5	6

Variable j j=1,2,3	Dollar cost penalty per day for standard, commercial, and government jobs, respectively
Variable COUNT	Counter used in Timer Segment to determine when the cost table is complete and ready for printing
Variable LATE	Lateness time, minutes
Variable LOSS	Lateness penalty, dollars per week
Variable TABLE	Variable used in Lathe Department Segment to determine number of Table in which to enter lateness time
Variable TABNO	Variable used in Timer Segment to determine number of Table in which lateness time information can be found

Block Diagram

The Block Diagram appears in Figure 17-A.

Discussion of Block Diagram

The Lathe Department Segment requires no further special discussion. In the Timer Segment, a single Transaction prints out the relative and absolute clock and Queue statistics for the two lathe departments. Then the single Transaction double-SPLITs and each of the resulting six Transactions produces an entry in one of the six Matrix 1 cells corresponding to Rows 1-2, Columns 1-3. Each of the 6 Transactions also makes an incremental contribution to Column 4, where cumulative costs are developed. The Transaction which completes building up Matrix 1 prints out a copy of the Matrix and then turns off the simulation.

Other Model Documentation

Figure	Information Exhibited
17-B	Program Listing
17-C	Selected Program Output; Time and Cost Data

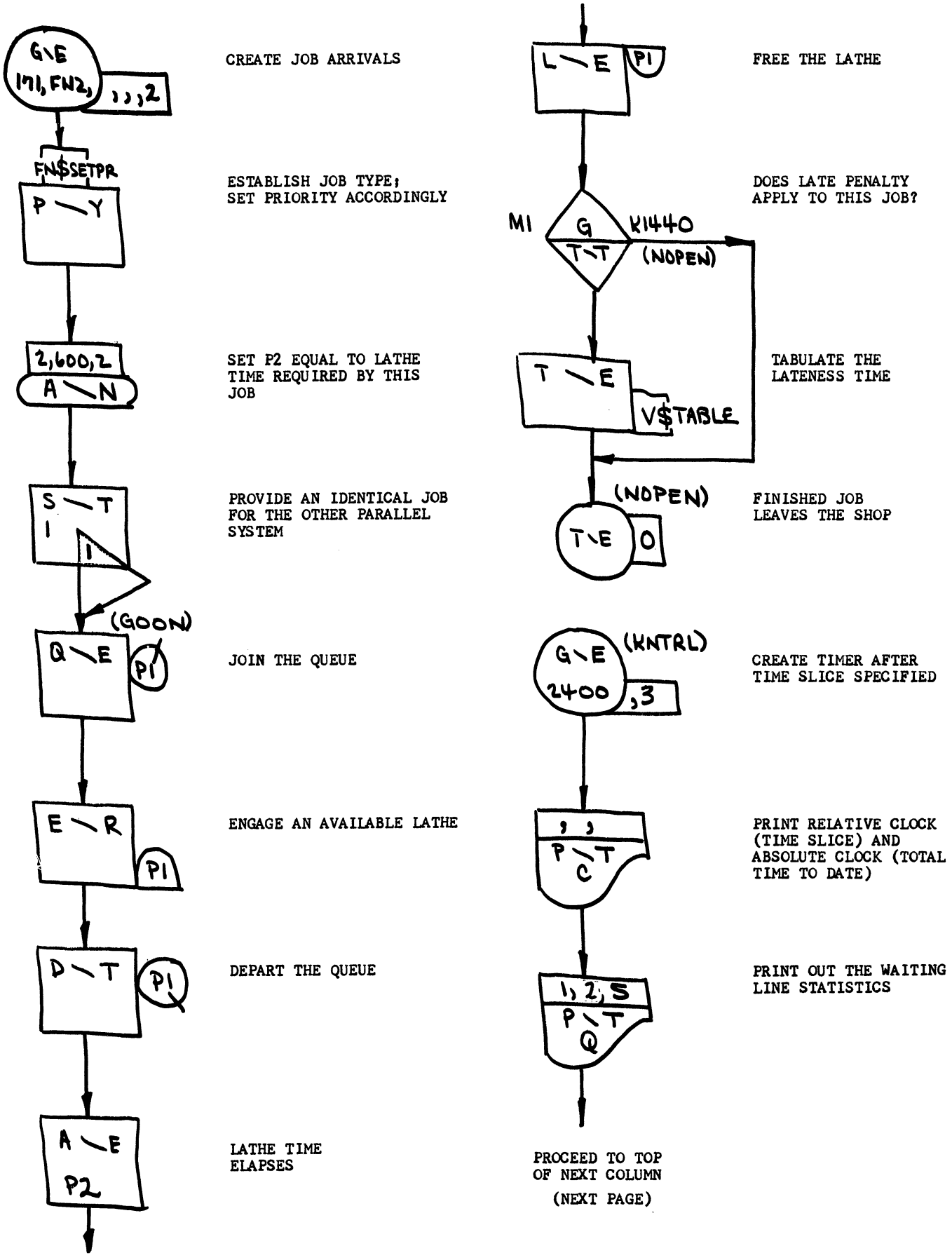


FIGURE 17-A: BLOCK DIAGRAM (CONTINUED ON NEXT PAGE)

(THE LATHE DEPARTMENT PROBLEM)

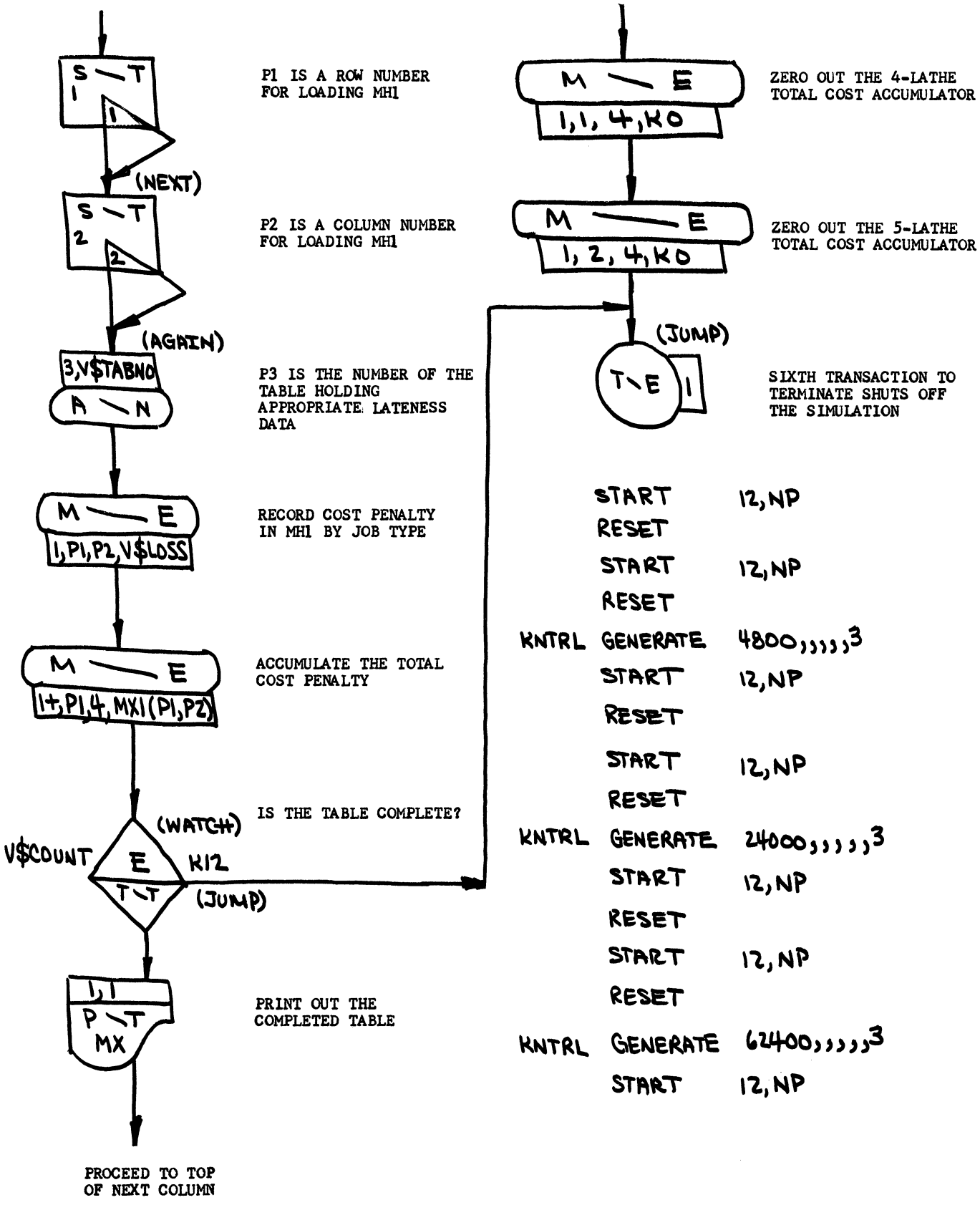


FIGURE 17-A: BLOCK DIAGRAM (CONTINUED FROM PRECEDING PAGE)
(THE LATHE DEPARTMENT PROBLEM)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS PAR=8
       SIMULATE
*
* LATHE DEPARTMENT SEGMENT
*
 SETPR FUNCTION   RN1,D3
.5,3/.857,2/1,1
      2 FUNCTION   RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
      1 STORAGE    4
      2 STORAGE    5
      1 MATRIX     X,2,4
      1 VARIABLE   K30
      2 VARIABLE   K150
      3 VARIABLE   K300
 COUNT VARIABLE    N$WATCH@K6+K1
 TABLE VARIABLE    PR+K3*(P1-K1)
 TABNO VARIABLE    P2+K3*(P1-K1)
 LATE  VARIABLE    M1-K1440
 LOSS  FVARIABLE   (TB*3*TC*3/K480)/(C1/K2400)*V*2
      1 TABLE      V$LATE,0,240,6
      2 TABLE      V$LATE,0,240,6
      3 TABLE      V$LATE,0,240,6
      4 TABLE      V$LATE,0,240,6
      5 TABLE      V$LATE,0,240,6
      6 TABLE      V$LATE,0,240,6
*
       GENERATE   171,FN2,,,,2     GENERATE JOB ARRIVALS
       PRIORITY   FN$SETPR         ESTABLISH PRIORITY BY JOB TYPE
       ASSIGN     2,600,2          P2 = LATHE TIME FOR THIS JOB
       SPLIT      1,GOON,1         TWO CONFIGURATIONS ARE STUDIED IN PARALLEL
 GOON  QUEUE      P1               QUEUE UP FOR A LATHE
       ENTER      P1               GO ON THE LATHE
       DEPART     P1               DEPART THE QUEUE
       ADVANCE    P2               LATHE TIME ELAPSES
       LEAVE      P1               FREE THE LATHE
       TEST G     M1,K1440,NOPEN   DOES LATE PENALTY APPLY TO THIS JOB?
       TABULATE   V$TABLE          TABULATE THE LATENESS TIME
 NOPEN TERMINATE  0                THIS JOB IS DONE
*
* TIMER SEGMENT
*
 KNTRL GENERATE   2400,,,,,3      CREATE TIMER AFTER SPECIFIED TIME SLICE
       PRINT      ,,C             PRINT TIME SLICE FOR THIS RUN AND TOTAL TIME
       PRINT      1,2,Q,S         PRINT OUT THE WAITING LINE STATISTICS
       SPLIT      1,NEXT,1        P1 IS A ROW NUMBER FOR LOADING MH1
 NEXT  SPLIT      2,AGAIN,2       P2 IS A COLUMN NUMBER FOR LOADING MH1
 AGAIN ASSIGN     3,V$TABNO       P3 = NUMBER OF TABLE HOLDING 'LATENESS' DATA
       MSAVEVALUE 1,P1,P2,V$LOSS     RECORD COST PENALTY IN MH1 BY JOB TYPE
       MSAVEVALUE 1+,P1,4,MX1(P1,P2)    ACCUMULATE TOTAL COST PENALTY
 WATCH TEST E     V$COUNT,K6,JUMP    IS THE TABLE COMPLETE?
       PRINT      1,1,MX          PRINT OUT THE COMPLETED TABLE
       MSAVEVALUE 1,1,4,KC        ZERO OUT THE 4-LATHE TOTAL COST ACCUMULATOR
       MSAVEVALUE 1,2,4,KC        ZERO OUT THE 5-LATHE TOTAL COST ACCUMULATOR
 JUMP  TERMINATE  1
       START      6,NP            CARRY OUT THE SIMULATION
*
       RESET                      PARTIALLY RESTORE THE PRECEDING MODEL
       START      6,NP            CONTINUE THE SIMULATION
*
       RESET                      PARTIALLY RESTORE THE PRECEDING MODEL
 KNTRL GENERATE   4800,,,,,3      CHANGE SIZE OF TIME SLICE
       START      6,NP            CONTINUE THE SIMULATION
*
       RESET                      PARTIALLY RESTORE THE PRECEDING MODEL
       START      6,NP            CONTINUE THE SIMULATION
*
       RESET                      PARTIALLY RESTORE THE PRECEDING MODEL
 KNTRL GENERATE   24000,,,,,3     CHANGE SIZE OF TIME SLICE
       START      6,NP            CONTINUE THE SIMULATION
*
       RESET                      PARTIALLY RESTORE THE PRECEDING MODEL
       START      6,NP            CONTINUE THE SIMULATION
*
       RESET                      PARTIALLY RESTORE THE PRECEDING MODEL
 KNTRL GENERATE   62400,,,,,3     CHANGE SIZE OF TIME SLICE
       START      6,NP            FINISH THE SIMULATION.
       END                        RETURN CONTROL TO THE OPERATING SYSTEM
$SIGH                             SIGN-OFF
```

FIGURE 17-B: PROGRAM LISTING

(THE LATHE DEPARTMENT PROBLEM)

Discussion of Output

The simulation was run for the equivalent of 52 working weeks. Statistics were printed our for these seven time slices during the run:

Slice	Time Spanned
1	Week 1
2	Week 2
3	Weeks 3 and 4
4	Weeks 5 and 6
5	Weeks 7 through 16
6	Weeks 17 through 26
7	Weeks 26 through 52

The influence of randomness both in arrival times and job types is evident in the results. Conclusions are based on comparison of MH1 (1,4) and MH1 (2,4), representing average weekly cost penalties for the 4-lathe and 5-lathe cases, respectively. The MH1 (2,4) figure must be increased by $500 for comparative purposes to reflect the higher capitalized cost because of the additional lathe. It appears that based on available information, neither case shows a clear advantage over the other. Before final conclusions are reached, however, this model should be studied further to determine how long a run is required before steady state is reached. Alternative approaches to modeling this situation should also be considered. Whether the GENERATE/PRIORITY sequence is consistent with the problem statement, for example, is a question that might be raised.

```
RELATIVE CLOCK          2400  ABSOLUTE CLOCK          2400
TERMINATIONS TO GO         6

QUEUE       MAXIMUM      AVERAGE      TOTAL       ZERO      PERCENT      AVERAGE      $AVERAGE
            CONTENTS     CONTENTS     ENTRIES     ENTRIES   ZEROS        TIME/TRANS   TIME/TRANS
   1            1         .000          12          12      100.0          .000         .000
   2            1         .000          12          12      100.0          .000         .000
 $AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

MATRIX FULLWORD SAVEVALUE    1

            COLUMN   1          2          3          4
 ROW  1              0          0          0          0
      2              0          0          0          0
```

```
RELATIVE CLOCK          2400  ABSOLUTE CLOCK          4800
TERMINATIONS TO GO         6

QUEUE       MAXIMUM      AVERAGE      TOTAL       ZERO      PERCENT      AVERAGE      $AVERAGE
            CONTENTS     CONTENTS     ENTRIES     ENTRIES   ZEROS        TIME/TRANS   TIME/TRANS
   1            3         .296          12           8      66.6         59.333       178.000
   2            1         .008          12          11      91.6          1.750        21.000
 $AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

MATRIX FULLWORD SAVEVALUE    1

            COLUMN   1          2          3          4
 ROW  1              0          0        426        426
      2              0          0        426        426
```

```
RELATIVE CLOCK          4800  ABSOLUTE CLOCK          9600
TERMINATIONS TO GO         6

QUEUE       MAXIMUM      AVERAGE      TOTAL       ZERO      PERCENT      AVERAGE      $AVERAGE
            CONTENTS     CONTENTS     ENTRIES     ENTRIES   ZEROS        TIME/TRANS   TIME/TRANS
   1            1         .004          25          24      95.9          .839        21.000
   2            1         .000          25          25      100.0         .000          .000
 $AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

MATRIX FULLWORD SAVEVALUE    1

            COLUMN   1          2          3          4
 ROW  1              0        219         18        237
      2              0        219         18        237
```

```
RELATIVE CLOCK          4800  ABSOLUTE CLOCK         14400
TERMINATIONS TO GO         6

QUEUE       MAXIMUM      AVERAGE      TOTAL       ZERO      PERCENT      AVERAGE      $AVERAGE
            CONTENTS     CONTENTS     ENTRIES     ENTRIES   ZEROS        TIME/TRANS   TIME/TRANS
   1            2         .106          21          12      57.1         24.285       56.666
   2            1         .000          21          21      100.0          .000          .000
 $AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

MATRIX FULLWORD SAVEVALUE    1

            COLUMN   1          2          3          4
 ROW  1              0         50          0         50
      2              0         50          0         50
```

FIGURE 17-C: SELECTED OUTPUT (CONTINUED ON NEXT PAGE)

(THE LATHE DEPARTMENT PROBLEM)

```
RELATIVE CLOCK          24000  ABSOLUTE CLOCK          38400
TERMINATIONS TO GO          6

QUEUE       MAXIMUM      AVERAGE      TOTAL        ZERO       PERCENT       AVERAGE       $AVERAGE
            CONTENTS     CONTENTS     ENTRIES      ENTRIES    ZEROS         TIME/TRANS    TIME/TRANS
   1            12        2.999         155           43        27.7         464.393       642.687
   2             7         .750         155           95        61.2         116.212       300.216
 $AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

MATRIX FULLWORD SAVEVALUE    1

            COLUMN   1          2          3          4
 ROW  1              101        276        577        954
      2               25         65        353        443
```

```
RELATIVE CLOCK          24000  ABSOLUTE CLOCK          62400
TERMINATIONS TO GO          6

QUEUE       MAXIMUM      AVERAGE      TOTAL        ZERO       PERCENT       AVERAGE       $AVERAGE
            CONTENTS     CONTENTS     ENTRIES      ENTRIES    ZEROS         TIME/TRANS    TIME/TRANS
   1            11        2.213         148           55        37.1         359.026       571.354
   2             5         .452         146          103        70.5          74.458       252.813
 $AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

MATRIX FULLWORD SAVEVALUE    1

            COLUMN   1          2          3          4
 ROW  1               29        312        330        o71
      2                1        251        323        575
```

```
RELATIVE CLOCK          62400  ABSOLUTE CLOCK         124800
TERMINATIONS TO GO          6

QUEUE       MAXIMUM      AVERAGE      TOTAL        ZERO       PERCENT       AVERAGE       $AVERAGE
            CONTENTS     CONTENTS     ENTRIES      ENTRIES    ZEROS         TIME/TRANS    TIME/TRANS
   1            19        5.034         412           63        15.2         762.580       900.237
   2            11         .827         402          230        57.2         128.405       300.110
 $AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

MATRIX FULLWORD SAVEVALUE    1

            COLUMN   1          2          3          4
 ROW  1              270        606        304       1180
      2               12        112        262        386
```

```
          CPU TIME USED:

     ASSEMBLY:      1.678 SECONDS

     EXECUTION:    10.275 SECONDS
```

```
**** CN AT  01:43.18
**** OFF AT 01:44.24
**** ELAPSED TIME        66.696   SEC.
**** CPU TIME USED        16.094   SEC.
**** STORAGE USED        911.83    PAGE-SEC.
**** CARDS READ           85
**** LINES PRINTED       357
**** PAGES PRINTED        14
**** CARDS PUNCHED         1
**** DRUM READS
**** APPROX. COST OF THIS RUN     $1.68
```

FIGURE 17-C: SELECTED OUTPUT; TIME AND COST DATA (CONTINUED FROM PRECEDING PAGE)
(THE LATHE DEPARTMENT PROBLEM)

61. The Pre-empt Capability

*** When two or more Transactions are competing to SEIZE a Facility, the competition can be resolved in either of two ways:

1) First-come, first-served within priority class (the GPSS "default case" queue discipline), or

2) According to any arbitrary rule which can be modeled using the LINK/UNLINK pair (to be discussed in Section 63).

This implies that Transactions requiring a Facility are prepared to wait until it becomes available before they make their bid to obtain it. This need not always be true. Some Transactions may have such an urgent need for a Facility that they have the right to pre-empt it, i.e. take it away from some other Transaction if it is already in use. This pre-empt, or "interrupt" capability is provided by GPSS in the form of the PREEMPT/RETURN Complementary Block Pair.

We now pose and answer several questions which have probably occurred to the reader:

1) What happens to the pre-empted Transaction? For simplicity, assume that it was at an ADVANCE Block, corresponding to being "in the process of using" the Facility. Then its remaining ADVANCE time is computed and saved, and the Transaction is put into temporarily inactive status (on an Interrupt Chain). It remains in contention for the Facility, which it will eventually re-engage until its requirements vis-a-vis that Facility are satisfied.

2) Can a pre-empting Transaction have a pre-empt put on it (relative to the Facility in question) by another Transaction? The answer is "no" or "yes", depending on the wishes of the modeler. In particular, an available option is that a pre-empting Transaction can itself be pre-empted by a Transaction with a higher PR value.

3) If a pre-empting Transaction is pre-empted, what happens to it? Assume it was at an ADVANCE Block. Then its remaining ADVANCE time is computed and saved, and the Transaction is put into temporarily inactive status (on the Interrupt Chain). It is chained on a last-in, first-out basis (i.e. pushdown list), meaning it will eventually re-engage and finish using the Facility before the Transaction it had pre-empted does so.

4) What if a Facility is not in use and a Transaction pre-empts it? There is no problem that arises here. An idle Facility can be pre-empted (the very meaning of the word "pre-empt" admittedly does not seem to apply in this case). But whereas a Transaction which has SEIZED an idle Facility can be pre-empted, a Transaction which has PREEMPTED an idle Facility cannot itself be pre-empted (unless the available option mentioned in 2) above is used).

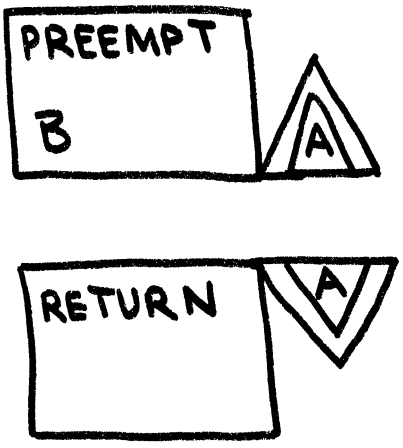
Now the PREEMPT/RETURN Block details are discussed.

The PREEMPT/RETURN Blocks:

Purpose: Enable a Transaction to engage a Facility at the expense of some other Transaction which had been using it (if any).

Enable the pre-empting Transaction to lift the pre-empt which has been in effect on the Facility.

Shape and Operands:



Operand	Significance
A	Speficies the number of the Facility being PREEMPTED or RETURNED
B	Optional Operand; If B is not used, the pre-empt occurs if and only if the Facility is not already pre-empted. If B is PR, the pre-empt takes place only if the PR of the would-be pre-empter exceeds that of the Transaction which already has the Facility (if any)

Further Comment:

There are actually additional optional Operands (C,D, and E) available for use with the PREEMPT Block. These provide the modeler with a variety of options in determining the fate of the pre-empted Transaction. The user who wants to investigate these options is referred to the User's Manual.

62. Boolean Variables

*** A Boolean Variable can assume either one of two values: True, or False. However, True is coded as 1 and False is coded as 0.

*** The most basic units used in defining Boolean Variables are of two types:

Type 1): A three-element sequence consisting of

SNA-RO-SNA

where:

SNA is any Standard Numerical Attribute, and

RO is any one of the relational operators:

'G'
'L'
'E'
'NE'
'LE'
'GE'

The relational operators represent greater than, less than, etc. Note that the apostrophes are a necessary part of the relational operator in GPSS in the context being described. Also note that no apostrophes are involved when G, L, E, NE, LE, and GE are used with the TEST Block.

Examples:

Q1'L'Q2
FR*1'GE'C1
R4'E'K2

Type 2): A single element consisting of

LOj

where:

LO is a logical operator referencing a Facility, Storage, or Logic Switch, and having the specific form and making the assertion shown below:

	Logical Operator	Assertion
Facilities:	FU	Facility is in Use
	FNU	Facility is not in Use
	FI	Facility is Pre-empted
	FNI	Facility is not Pre-empted
	FN	Facility is either in Use or Pre-empted
Storages:	SE	Storage is Empty
	SNE	Storage is not Empty
	SF	Storage is Full
	SNF	Storage is not Full

	LS	Logic Switch is Set
Logic Switches:	LR	Logic Switch is Reset

and j, as usual, is the index number of the Facility, Storage, or Logic Switch for which the assertion is made.

Examples: FNU3
SE8
LS21

*** Note that each of the two types of basic units has a value at any particular time of either True or False. Hence, Q1'L'Q2 is either True or False (Q1 is less than Q2 or it is not.) Also, FNU3 is either True or False (The assertion that Facility 3 is not in use is either True or False.)

*** Any two of the basic units described above can be joined by a Boolean operator in the pattern:

BU-BO-BU

where:

BU is one of the basic units, and

BO is one of these two Boolean operators:

Boolean Operator	Prose Description	Effect
*	"and"	BU*BU takes the value True only if each of BU units is True; Otherwise it takes the value False.
+	"or"	BU+BU takes the value True if either one or both of the BU units is True; otherwise it takes the value False.

Examples: Q1'L'Q2*FR*1'GE'C1
R4'E'K2+FNU3
SE8*LS21

Two basic units joined by a Boolean operator are termed a compound unit. Compound units can themselves be joined to basic units or to other compound units by Boolean operators to form what are called complex units. No matter how complicated this might become, the value of the resulting complex unit will be Boolean-valued: either True or False

*** A Boolean variable is any configuration that can only be either True or False. Hence a basic unit as described above is a Boolean Variable; so is a compound unit; so is a complex unit.

*** In any complex unit, the final result is developed first by evaluating the effect of the relational operators, then considering the subsequent effect of the Boolean operators. That is, the relational operators enjoy a higher precedence level than do the Boolean operators. It should be clear that this is strictly necessary, not arbitrary. Within precedence level no distinctions are made; evaluation is, as usual, on a left-to-right basis. Pairs of parentheses are permitted in constructing complex units.

*** Each Boolean Variable used in a model is defined by including a Variable Definition Card as part of the model. The card takes this form:

card columns:	2–6	8–18	19–
contents:	j	BVARIABLE	basic, compound, or complex unit constituting the definition

where:

j is the number of the Boolean Variable being defined.

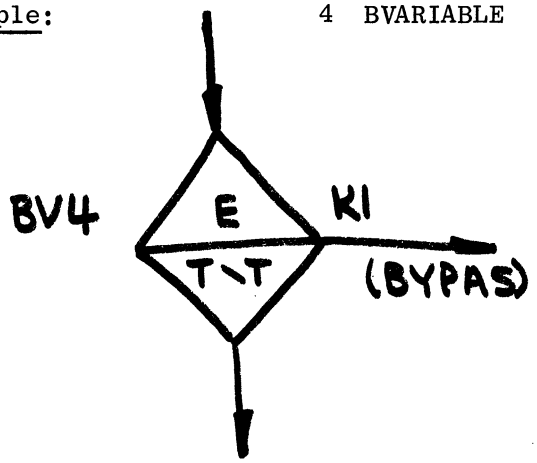
The basic, compound, or complex unit constituting the definition must be punched in consecutive card columns and must not extend beyond card column 71. There is no continuation card concept in defining Boolean Variables.

*** Boolean Variables are referenced as BVj. The value of a Boolean Variable is computed each time a reference is made to that Variable.

*** Boolean Variables can be used in models in connection with the TEST Block.

Example:

```
4  BVARIABLE  FNU2*FNU3*FNU8
```



In the example shown, if one or more of Facilities 2, 3, and 8 is in use, the testing Transaction will be routed to the non-sequential Block BYPAS. Otherwise, the Transaction proceeds to the next sequential Block.

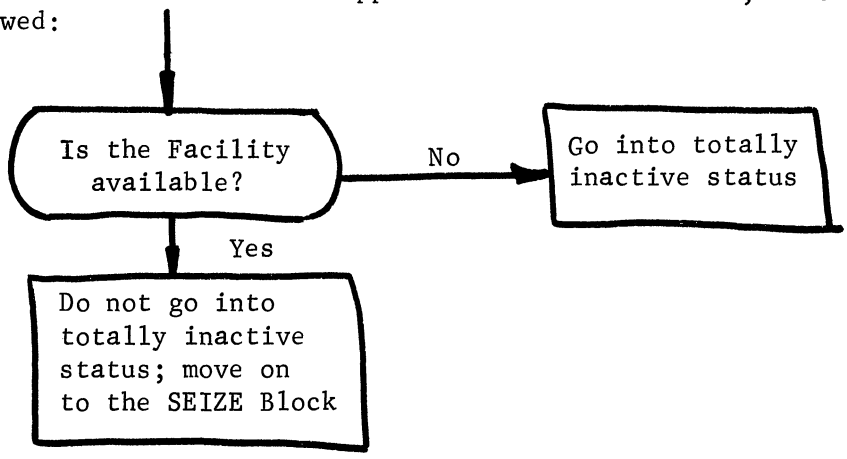
63. User Chains and Their Use for Queue Discipline and Model Efficiency

Transactions experiencing a blocking condition in a model reside on the Current Events Chain. They are either scan-active or scan-inactive depending on the type of blocking condition. In either case, the scan encounters them one or more times at each instant of simulated clock time. If a blocked Transaction is scan-inactive (meaning its Scan Status Indicator in On), the simulator moves immediately to the next Transaction on the Current Events Chain. The only time "lost" here is that required to test the Scan Status Indicator. Scan-active Transactions are fully processed, however, even though the logic of a given situation may make it clear (to the modeler) that the blocking condition is still in effect, meaning the processing is unnecessary and is wasteful of execution time. It should be clear that if blocked Transactions could be made totally inactive in a model by taking them off the Current Events Chain, then execution time economies would result.

Consider a specific example. Suppose that n Transactions are queued up for a Facility. When the Facility eventually becomes available, only one Transaction can SEIZE it. Even so, after the SEIZE occurs, the simulator continues at each scan to examine each of the other n-1 Transactions waiting for that facility. It would be much better if all n Transactions were taken off the Current Events Chain and put into totally inactive status until the Facility again becomes available. At that time, exactly one of the Transactions could be brought back onto the Current Events Chain and permitted to SEIZE the Facility. Which one was returned to the Current Events Chain from totally inactive status would be up to the programmer, i.e. would depend on whatever queue discipline the programmer wanted to invoke. Hence there is the possibility for a twofold benefit: decreased running time for a given model; and the structuring of arbitrary queue discipline.

Transactions can be put into totally inactive status in GPSS. During such totally inactive status they reside on chains known as User Chains. There may be one or more User Chains in a model. Analogous to the other GPSS entities, User Chains are designated by number (or by symbolic name). The ability to put a Transaction onto a User Chain, or remove one or more Transactions from a User Chain, is provided with the LINK/UNLINK Complementary Block Pair.

Before the Block details are taken up, consider the logic typically involved in deciding whether a Transaction should be put into totally inactive status. Appealing again to the "Transactions waiting for a Facility" example given above, when a new Transaction approaches the SEIZE Block, this pattern should be followed:

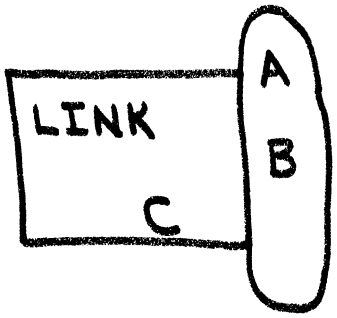


The key question is, how is the incoming Transaction to know whether the Facility (or whatever it is that might have to be waited for) is available? In this case, the necessary test could certainly be conducted with a GATE Block. GPSS, however, provides an equivalent but more direct way for a Transaction to determine whether inactive status is for it or not. In particular, each User Chain has associated with it a two-state indicator known as a Link Indicator. The Link Indicator is either Set or Reset ("on" or "off"). Hence the indicator can in effect signal to the Transaction whether it is logically sound to remain in active status, or whether to go the route of inactive status. The simulator does not automatically maintain the Link Indicator. As we are about to see, it must be set and reset by the programmer according to the logical conditions prevailing. The Link Indicator is automatically interrogated in conjunction with the LINK/UNLINK Blocks, which will now be discussed.

The LINK Block:

Purpose: Make it possible to remove a Transaction from active status in a model by LINKing it onto a USER CHAIN.

Shape and Operands:



Operand	Significance
A	Specifies the number (or symbolic name) of the User Chain onto which a Transaction which moves into the LINK Block may be linked.
B	Specifies whether the Transaction, if linked, is to be put on the front (LIFO) or the back (FIFO) of the User Chain, or is to be merged into the User Chain in increasing order according to the value of one of its Parameters (Pj).
C	Optional Operand: If C is not used, each Transaction moving into the LINK Block will be unconditionally linked onto the User Chain If the C Operand is used, it is the symbolic name of the Block to which the Transaction is routed if it remains in active status.

If the Link Indicator (LI) is:

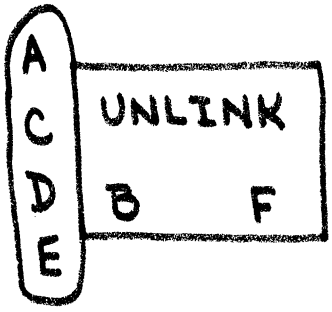
1) set, the Transaction is put on the User Chain and the LI is not affected.

2) reset, the Transaction remains in active status but the LI is put into set position.

The UNLINK Block:

Purpose: Make it possible for a Transaction in active status to trigger the UNLINKing of one or more Transactions from a User Chain, thereby bringing them back into active status

Shape and Operands:



Operand	Significance
A	Specifies the number (or symbolic name) of the User Chain from which one or more Transactions will be unlinked, if possible
B	Specifies the symbolic name of the Block to which the Transaction or Transactions being brought back into active status will be routed
C	Specifies the total number of Transactions which are to be unlinked, if possible; if the C Operand is ALL, all Transactions on the User Chain will be unlinked

The D and E Operands are used together to decide which Transactions (if any) are to be unlinked. These are the five possible D-E combinations and their significance:

1) D: blank; E: blank Unlink Transactions from the front of the User Chain.

2) D: BACK; E: blank Unlink Transactions from the back of the User Chain.

3) D: Pj; E: Any Standard Numerical Attribute starting at the front of the User Chain, unlink Transactions whose Pj value matches the value of the specified Standard Numerical Attribute

4) D: Pj; E: blank Starting at the front of the User Chain, unlink Transactions whose Pj value matches the Pj value of the Transaction that triggered the unlinking

5) D: BVj; E: blank Starting at the front of the User Chain, evaluate BVj Transaction by Transaction, unlinking each Transaction (up to the number specified) for which BVj is TRUE.

F — Optional operand;

If F is not used, the Transaction which triggered the unlinking moves on to the next sequential Block in the model

If F is used, it is the symbolic name of a Block in the model. The Transaction which triggered the unlinking is routed to this Block if no Transactions could be unlinked. Otherwise (if at least one Transaction was unlinked) the triggering Transaction moves to the next sequential Block.

UNLINK Block's effect on Link Indicator:

If there are no Transactions on a User Chain when it is referenced from an UNLINK Block, the Link Indicator is moved into the reset position. This is the only influence the UNLINK can have on the Link Indicator.

Control of the Link Indicator can be summarized this way:

1) The LI can be set (but never reset) via the LINK Block.

2) The LI can be reset (but never set) via the UNLINK Block.

Examples:

1) Show how to use the LINK/UNLINK pair in conjuction with the SEIZEing and RELEASEing of Facility 8. Transactions waiting for the Facility when it is engaged should be kept on User Chain 3. Queue discipline is to be first-come, first served independent of the priority levels of the waiting Transactions.

   Figure 18-A shows a model segment which meets the request. The reader should carefully think through the role of the Link Indicator as this sequence occurs:

   The simulation starts. (Link Indicators are initially in the reset position.)

   a) A first Transaction flows into the segment.

   b) A second Transaction flows into the segment before the first has released the Facility.

   c) The first Transaction releases the Facility.

   d) The second Transaction releases the Facility before a third Transaction comes along.

   e) A third Transaction flows into the segment.

   Figure 19-A shows an alternative model segment which meets the request.

2) Show how to use the LINK/UNLINK pair to model "random" queue discipline. That is, the next Transaction permitted to SEIZE Facility 8 should be chosen at random from those waiting on User Chain 3.

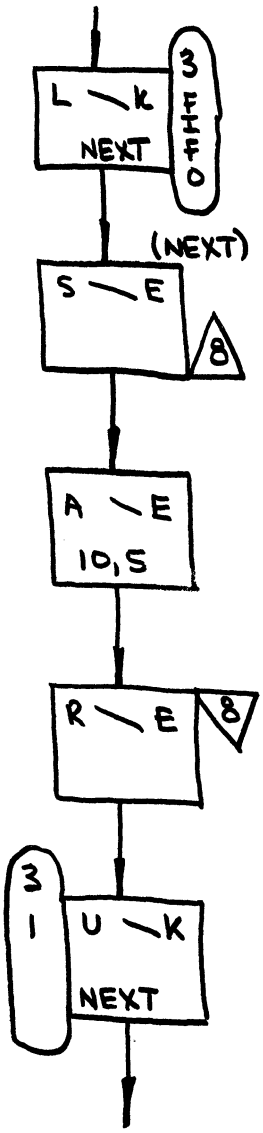


FIGURE 18-A:
FIRST-COME, FIRST-SERVED QUEUE DISCIPLINE
(MODELED WITH USER CHAIN)

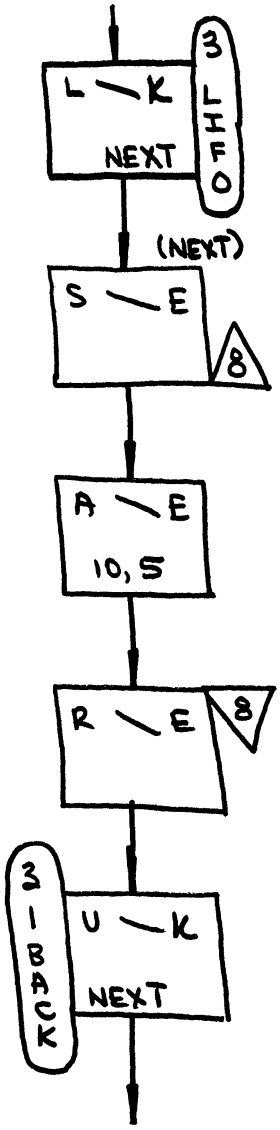


FIGURE 18-B:
FIRST-COME, FIRST-SERVED QUEUE DISCIPLINE
(MODELED WITH USER CHAIN - ALTERNATE APPROACH)

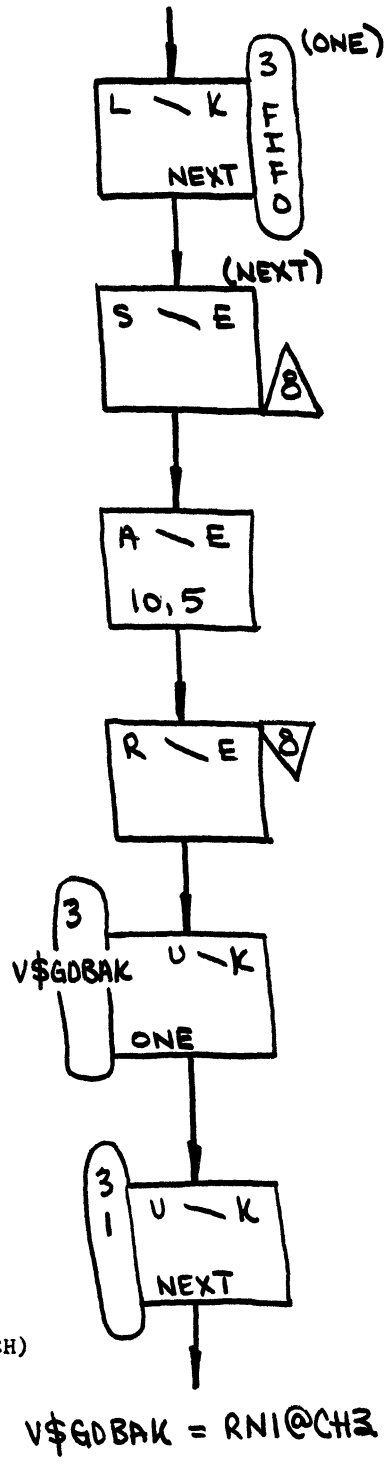


FIGURE 18-C:
RANDOM SERVICE AS A QUEUE DISCIPLINE

Hint: There is a Standard Numerical Attribute, CHj whose value is the current count of Transactions on User Chain j. Suppose a Variable, call it GOBAK, is defined as RN1@CH3. Then GOBAK takes on random values over the closed set of integers from 0 to CH3 minus 1 (i.e. from 0 to m-1, where m is the number of Transactions on the User Chain). For example, suppose CH3 is 5. Then RN1@CH3 will be one of the values 0, 1, 2, 3, or 4, because of the nature of modulus division. In the model, when a Transaction is to be removed from the User Chain, we simple use RN1@CH3 to determine how many Transactions to move from the front to the back of the chain before finally coming to the Transaction which is allowed to SEIZE the Facility.

Figure 18-C shows a model segment which implements this logic.

Execution Time Savings

As a measure of the savings in execution time that might result when the LINK/UNLINK pair is used for this purpose, the models shown in Figures 18-D and 18-E were computer-implemented. First the models shown were run. Then the LINK/UNLINK Blocks were removed and the models were re-run. (The names NEXT and TRUBL were put in place of the names GRAB and TUFF, too.) These execution times resulted:

		Execution Time, Seconds	
		With LINK/UNLINK	Without LINK/UNLINK
Model:	18-D	.569	1.470
	18-E	.596	2.775

Two things are clear from the table:

1) Use of the LINK/UNLINK Block saved execution time.

2) The time savings in 18-E was more worthwhile than that in 18-D. Transactions delayed at TEST Blocks are more costly in execution time than those delayed at SEIZE, ENTER or GATE Blocks. The simulator does not process the latter Transactions because their scan status indicator is on (see Figure 23, page 99, Users Manual). This is not the case for Transactions delayed at TEST Blocks. Hence, at each simulated clock instant the simulator uses real time attempting in vain (for the 18-E model) to move each delayed Transaction through the TEST Block.

   The 18-E example is of course a contrived one. Nevertheless, situations that are analogous, if less trivial, are frequently encountered in the modeling process.

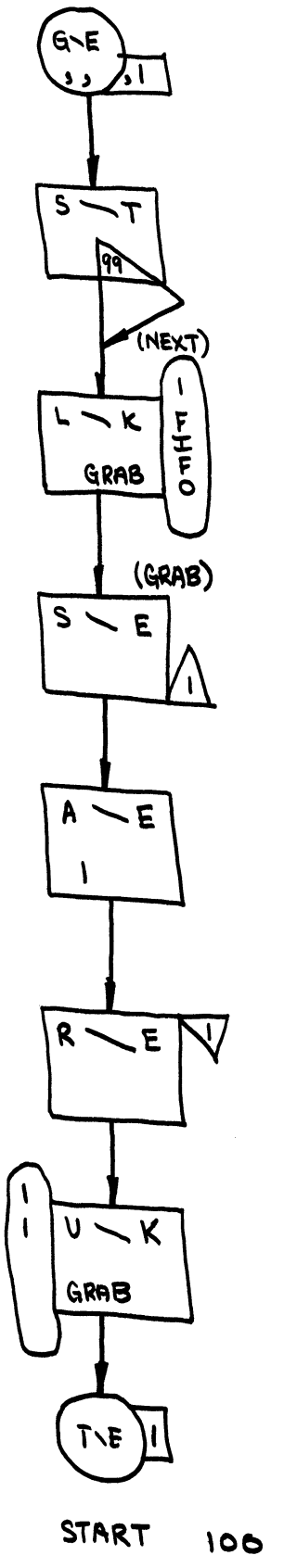


FIGURE 18-D:
A FIRST MODEL TO TEST USER CHAIN EFFICIENCY

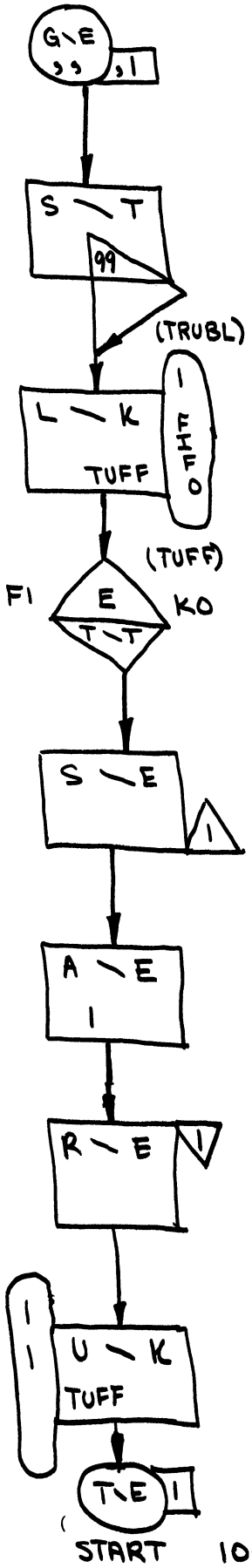


FIGURE 18-E:
A SECOND MODEL TO TEST USER CHAIN EFFICIENCY

64. User Chain Statistics Available During the Course of a Simulation

***Several properties of User Chains can be referenced as a simulation proceeds, making information available that can be used in the logic of a model. The five User Chain Standard Numerical Attributes are:

1) CAj: the integerized average number of Transactions on User Chain j

2) CCj: the number of different times Transactions have been put on User Chain j

3) CHj: the number of Transactions currently on User Chain j

4) CMj: the maximum number of Transactions on User Chain j to date

5) CTj: the integerized average time spent by Transactions on User Chain j

65. The Job Shop Problem: First-Come, First-Served Scheduling Rule

General Comments on Job Shop Scheduling

A "job" is an order for a piece of work whose completion requires that a series of operations be performed on a specified sequence of machines. A job shop is a shop in which the work is initiated and brought to completion. In general, more than one job is being worked on at a time; in fact, a stream of jobs is usually moving through the shop at all times. No two jobs are the same, except by chance (i.e. the series of operations and the sequence in which they are to be performed varies from job to job).

The problem lies in scheduling the jobs so that all due dates are met or, failing this, the total lateness time is minimized. Construction of a schedule involves using a rule to resolve the conflict resulting whenever two or more jobs simultaneously compete for a given machine. Effectiveness of alternative rules can be assessed by constructing models simulating shop operation, then comparing the schedules which result when the models are computer-implemented.

There are two problem modes:

1) The Static Problem: The problem is said to be static when all jobs are on hand at time zero.

2) The Dynamic Problem: The dynamic problem results when some jobs are on hand at time zero but new jobs are admitted to the shop from time to time.

Some of the scheduling rules (queue disciplines) that might be used to resolve conflicts in constructing schedules are:

1) Job Slack: The job which has the least job slack gets the machine next. Job slack is defined as the due date, minus the current time, minus the total operation time remaining for the job.

2) Job Slack per Operation: The job which has the least job slack per operation gets the machine next. Job slack per operation is job slack divided by the remaining number of operations for that job.

3) Job Slack Ratio: The job which has the least job slack ratio gets the machine next. Job slack ratio is job slack divided by the time remaining until the job's due date.

4) Shortest Imminent Operation: The job which will tie up the machine for the least amount of time gets the machine next.

5) First come, first served.

6) Random selection.

Rules 5 and 6 do not take available information concerning the jobs on hand into account and can be expected to be relatively poor scheduling rules as a result. Notice though that all six rules above are relatively inflexible. A flexible approach allows for the possibility of deviating from rigid adherence to the scheduling rule from time to time, as conditions may require. Three possibilities for more flexible scheduling are now suggested:

1) Alternate Operation: Schedule the next job on the machine according to shortest imminent operation unless one or more of the other jobs waiting for the same machine is critical, i.e. has negative job slack. If so, schedule the most critical job next.

2) Re-do with Adjusted Due Dates: If the schedule is complete but one or more jobs are finished late whereas others are finished early, re-do the simulation after decreasing the due date of each late job and possibly increasing the due date of each early job. On the second run, the scheduling rule will tend to "tighten up" the jobs previously late, hopefully completing them on time without forcing other jobs to be late. (This approach makes no sense, of course, unless the rule being tested uses the due date.)

3) Pre-empting: Under certain conditions, the job already using a machine may be pre-empted by a job which comes to the machine at a later time. In this case, some appropriate "inconvenience time" should be built into the model, since removing a partially-completed job from a machine, then bringing the job back on the machine later, will cost some time.

A particular Job Shop problem will now be considered.

Statement of the Problem

A certain Job Shop consisting of four machines has ten jobs on hand at time zero. The data concerning these jobs is contained in the following matrix:

	COL. 1	2	3	4	5	6	7	8	9	10	11
ROW 1	4	35	84	1	4	2	3	11	7	10	7
2	4	39	90	4	1	2	3	12	5	13	9
3	4	23	53	4	3	1	2	6	6	6	5
4	4	41	89	3	4	2	1	6	11	10	14
5	4	40	97	2	1	3	4	8	9	9	14
6	4	41	93	2	3	4	1	10	13	10	8
7	4	35	82	1	3	4	2	7	11	9	8
8	4	46	103	3	4	1	2	7	12	13	14
9	4	33	77	4	1	2	3	8	9	8	8
10	4	42	103	2	1	4	3	11	5	14	12

Job Information Matrix

Rows 1,2,3,...,10 carry information about jobs 1,2,3,...,10, respectively.

Column 1: Number of operations
Column 2: Total operation time, hours
Column 3: Due date, working hours after time zero
Columns 4 - 7: Numbers of the machines, appearing in the sequence in which the job requires them
Columns 8 - 11: Operation times on the machines in columns 4 - 7, respectively

Construct a GPSS model to simulate the operation of the shop using a first-come, first-served scheduling rule. The model should produce:

1) A Starting Time and Completion Time Matrix, which is a record of the time at which each job goes onto each machine, the time at which each job is completed, and the corresponding number of hours the job was early or late.

2) A Job Completion Table, which records the distribution of job earliness or lateness hours.

Assume that no other jobs are admitted to the shop. Also assume that the operation times are deterministic.

Approach Taken in Building the Model

The mechanics of the model are quite straight forward, once the "data management" considerations are complete. It is convenient to work directly with the Job Information Matrix shown above, using it as a Matrix Savevalue in the model and loading the information with INITIAL Cards. Then the necessary data can be drawn from the Matrix as it is needed during the simulation. Although the LINK/UNLINK Blocks do not have to be used for the first-come, first-served queue discipline, they will be included, making it easier to extend or modify the model. Finally, the model will be designed to schedule an arbitrary number of jobs (all on hand at time zero) through a Job Shop consisting of an arbitrary number of machines.

Table of Definitions

Time Unit: 1 Hour

GPSS Entity	Interpretation
Transaction	A job P1: Not used P2: Job number P3: Number of remaining operations P4: Imminent machine number P5: Imminent operation time P6: Not used P7: Column Index pointing to immenent machine number in Job Information Matrix P8: Column Index pointing to imminent operation time in Job Information Matrix
Facility j	Machine j j = 1,2,3,...,MTOT, where MTOT represents the number of machines in the shop
Matrix 1 (Halfword)	Job Information Matrix Rows 1,2,3,...,JTOT carry information about jobs 1,2,3,...,JTOT, respectively, where JTOT represents the number of jobs

Column	Information
1	Number of operations
2	Total operation time
3	Due date
4	Numbers of the machines, appearing in the sequence in which the job requires them
5	
:	
3+MTOT	
4+MTOT	Operation times on the machine in columns 4,5,...,3+MTOT, respectively
5+MTOT	
:	
3+2*MTOT	

Matrix 2 (Halfword)	Starting Time and Completion Time Matrix MH2(i,j) is the time job i went onto machine j, for i = 1,2,3,...,JTOT, and j = 1,2,3,...,MTOT MH2(i,MTOT+2) is the hours early or late that job i was completed
Savevalue JTOT (Halfword)	Total number of jobs
Savevalue MTOT (Halfword)	Total number of machines
Table DISTN	Job Completion Table
User Chain j	Waiting Line for Machine j j=1,2,3,...,MTOT
Variable HOURS	Earliness hours or lateness hours relative to job completion; a negative value means that the job was completed late
Variable JOBS	Total jobs in shop minus one; (used as A operand in SPLIT block)
Variable MACP1	Total machines in shop plus one
Variable MACP2	Total machines in shop plus two
Variable SHFT1	Total machines in shop plus three
Variable TIME	Current value of clock minus one

Block Diagram

The Block Diagram appears in Figure 19-A

Discussion of the Block Diagram

The LINK-SEIZE-ADVANCE-RELEASE-UNLINK sequence in the Block Diagram is basic. Inserted in this sequence is the MSAVEVALUE Block to record Starting Time in MH2. C1 is not read directly into MH2 as Starting Time because the first Transaction arrives at the model at time 1, not time 0. In GPSS, no Transactions arrive at a model at time 0 since an internal problem might then result in evaluating time integrals. The Variable TIME=C1-K1 compensates for this problem which, in general, is unfortunate but can be conveniently neglected.

Ahead of the basic sequence are four Blocks which arrange to copy the up-to-date imminent machine number and imminent operation time from MH1 into P4 and P5, respectively. Then the entire string is put into a loop, where looping takes place relative to the total number of operations each job requires.

Prior to the loop segment, a copy of the Job Information Matrix is printed and P7 and P8 initializations take place. After the loop, job completion data is entered in MH2 and the Table DISTN. Final output occurs automatically at the end of the run.

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
*       SIMULATE
        INITIAL     XH$MTOT,4/XH$JTOT,10
      1 MATRIX      H,10,11
      2 MATRIX      H,10,6
        INITIAL     MH1(1-10,1),4/MH1(1,2),35/MH1(2,2),39/MH1(3,2),23
        INITIAL     MH1(4,2),41/MH1(5,2),40/MH1(6,2),41/MH1(7,2),35
        INITIAL     MH1(8,2),46/MH1(9,2),33/MH1(10,2),42/MH1(1,3),84
        INITIAL     MH1(2,3),90/MH1(3,3),53/MH1(4,3),89/MH1(5,3),97
        INITIAL     MH1(6,3),93/MH1(7,3),82/MH1(8,3),103/MH1(9,3),77
        INITIAL     MH1(10,3),103/MH1(1,4),1/MH1(2-3,4),4/MH1(4,4),3
        INITIAL     MH1(5-6,4),2/MH1(7,4),1/MH1(8,4),3/MH1(9,4),4
        INITIAL     MH1(10,4),2/MH1(1,5),4/MH1(2,5),1/MH1(3,5),3
        INITIAL     MH1(4,5),4/MH1(5,5),1/MH1(6-7,5),3/MH1(8,5),4
        INITIAL     MH1(9-10,5),1/MH1(1-2,6),2/MH1(3,6),1/MH1(4,6),2
        INITIAL     MH1(5,6),3/MH1(6-7,6),4/MH1(8,6),1/MH1(9,6),2
        INITIAL     MH1(10,6),4/MH1(1-2,7),3/MH1(3,7),2/MH1(4,7),1
        INITIAL     MH1(5,7),4/MH1(6,7),1/MH1(7-8,7),2/MH1(9-10,7),3
        INITIAL     MH1(1,8),11/MH1(2,8),12/MH1(3-4,8),6/MH1(5,8),8
        INITIAL     MH1(6,8),10/MH1(7-8,8),7/MH1(9,8),8/MH1(10,8),11
        INITIAL     MH1(1,9),7/MH1(2,9),5/MH1(3,9),6/MH1(4,9),11
        INITIAL     MH1(5,9),9/MH1(6,9),13/MH1(7,9),11/MH1(8,9),12
        INITIAL     MH1(9,9),9/MH1(10,9),5/MH1(1,10),10/MH1(2,10),13
        INITIAL     MH1(3,10),6/MH1(4,10),10/MH1(5,10),9/MH1(6,10),10
        INITIAL     MH1(7,10),9/MH1(8,10),13/MH1(9,10),8/MH1(10,10),14
        INITIAL     MH1(1,11),7/MH1(2,11),9/MH1(3,11),5/MH1(4-5,11),14
        INITIAL     MH1(6-7,11),8/MH1(8,11),14/MH1(9,11),8/MH1(10,11),12
DISTN   TABLE       V$HOURS,-20,5,10
HOURS   VARIABLE    MH1(P2,3)-(C1-K1)
JOBS    VARIABLE    XH$JTOT-K1
MACP1   VARIABLE    XH$MTOT+K1
MACP2   VARIABLE    XH$MTOT+K2
SHFT1   VARIABLE    XH$MTOT+K3
TIME    VARIABLE    C1-K1
        GENERATE    ,,,1              CREATE A MASTER XACT FOR THE MODEL
           PRINT    1,1,MH,S          PRINT OUT THE JOB FILE
        ASSIGN      7,K3              INITIALIZE COL INDEX FOR MACHINE NUMBER
        ASSIGN      8,V$SHFT1         INITIALIZE COL INDEX FOR MACHINE TIME
        SPLIT       V$JOBS,NEXT,2     CREATE 1 XACT PER JOB, P2 = JOB NUMBER
NEXT    ASSIGN      3,MH1(P2,1)        P3 = NUMBER OF OPERATIONS REQUIRED
RETUR   ASSIGN      7+,K1             UPDATE COL INDEX FOR MACHINE NUMBER
        ASSIGN      8+,K1             UPDATE COL INDEX FOR MACHINE TIME
        ASSIGN      4,MH1(P2,P7)      P4 = IMMINENT MACHINE NUMBER
        ASSIGN      5,MH1(P2,P8)      P5 = IMMINENT MACHINE TIME
        LINK        P4,FIFO,START     JOB WAITS ON CHAIN IF MACHINE IS BUSY
START   SEIZE       P4                JOB GOES ON THE MACHINE
        MSAVEVALUE  2,P2,P4,V$TIME,H      RECORD TIME JOB WENT ON THE MACHINE
        ADVANCE     P5                MACHINE TIME ELAPSES
        RELEASE     P4                JOB GOES OFF THE MACHINE
        UNLINK      P4,START,1        PUT WAITING JOB ON THE MACHINE
        LOOP        3,RETUR           SEND THIS JOB TO ITS NEXT MACHINE
        MSAVEVALUE  2,P2,V$MACP1,V$TIME,H      RECORD TIME JOB IS FINISHED
        MSAVEVALUE  2,P2,V$MACP2,V$HOURS,H     RECORD HOURS EARLY OR LATE BY JOB
        TABULATE    DISTN             TABULATE HOURS EARLY OR LATE
        TERMINATE   1                 FINISHED JOB LEAVES THE SHOP
        START       10
        END
$SIGH
```

FIGURE 19-B: PROGRAM LISTING

(JOB SHOP PROBLEM: FIRST-COME, FIRST-SERVED)

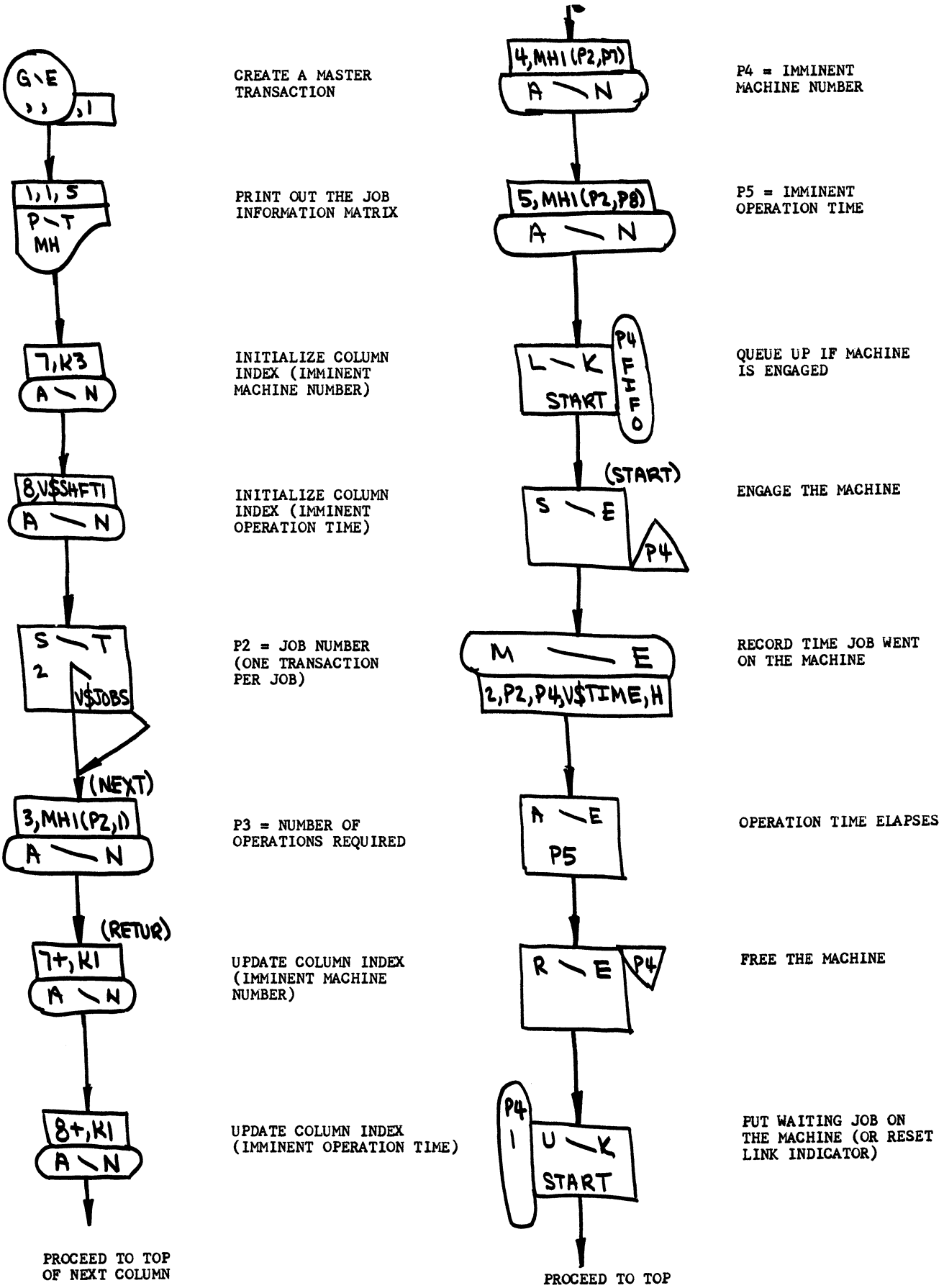
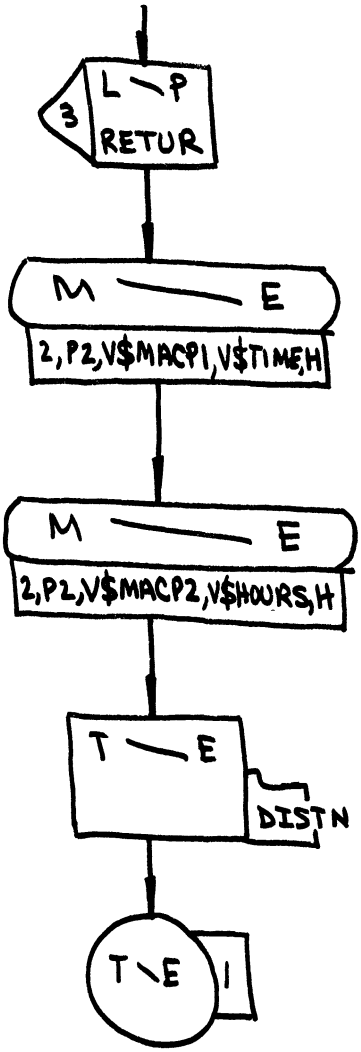


FIGURE 19-A: BLOCK DIAGRAM (CONTINUED ON NEXT PAGE)

(JOB SHOP PROBLEM: FIRST-COME, FIRST-SERVED)



PROCEED TO NEXT OPERATION

RECORD TIME JOB IS FINISHED

RECORD HOURS EARLY OR LATE BY JOB

TABULATE HOURS EARLY OR LATE

FINISHED JOB LEAVES THE SHOP

FIGURE 19-A: BLOCK DIAGRAM (CONTINUED FROM PRECEDING PAGE)
(JOB SHOP PROBLEM: FIRST-COME, FIRST-SERVED)

Other Model Documentation:

Figure	Information Exhibited
19-B	Program Listing
19-C	Selected Program Output; Time and Cost Data

Discussion of Program Output

The results can be interpreted from the output with reference to the Table of Definitions. In particular, the Table DISTN shows that the average job was completed 2.3 hours ahead of its due date, with a standard deviation of 13.5 hours. MH2 shows that of the 10 jobs, 3 failed to meet their due date and 7 were finished ahead of schedule. One job was 25 hours late (job 3) and another was 24 hours early (job 2). A better scheduling rule could hopefully improve this situation.

RELATIVE CLOCK 104 ABSOLUTE CLOCK 104

USER CHAIN	TOTAL ENTRIES	AVERAGE TIME/TRANS	CURRENT CONTENTS	AVERAGE CONTENTS	MAXIMUM CONTENTS
1	7	11.857		.798	2
2	8	11.625		.894	3
3	7	11.428		.769	3
4	9	24.111		2.086	4

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.836	10	8.699		
2	.932	10	9.699		
3	.846	10	8.799		
4	.990	10	10.299		

TABLE DISTN

ENTRIES IN TABLE	MEAN ARGUMENT	STANDARD DEVIATION	SUM OF ARGUMENTS
10	2.299	13.496	23.000

UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
-20	1	9.99	9.9	90.0	-8.695	-1.652
-15	0	.00	9.9	90.0	-6.521	-1.281
-10	0	.00	9.9	90.0	-4.347	-.911
-5	1	9.99	19.9	80.0	-2.173	-.540
0	1	9.99	29.9	70.0	-.000	-.170
5	5	50.00	79.9	20.0	2.173	.200
10	0	.00	79.9	20.0	4.347	.570
15	0	.00	79.9	20.0	6.521	.941
20	1	9.99	89.9	10.0	8.695	1.311
OVERFLOW	1	9.99	100.0	.0		
AVERAGE VALUE OF OVERFLOW		24.00				

MATRIX HALFWORD SAVEVALUE 1

	COL. 1	2	3	4	5	6	7	8	9	10	11
ROW 1	4	35	84	1	4	2	3	11	7	10	7
2	4	39	90	4	1	2	3	12	5	13	9
3	4	23	53	4	3	1	2	6	6	6	5
4	4	41	89	3	4	2	1	6	11	10	14
5	4	40	97	2	1	3	4	8	9	9	14
6	4	41	93	2	3	4	1	10	13	10	8
7	4	35	82	1	3	4	2	7	11	9	8
8	4	46	103	3	4	1	2	7	12	13	14
9	4	33	77	4	1	2	3	8	9	8	8
10	4	42	103	2	1	4	3	11	5	14	12

MATRIX HALFWORD SAVEVALUE 2

	COL. 1	2	3	4	5	6
ROW 1	0	63	74	37	81	3
2	27	32	57	0	66	24
3	48	73	42	12	78	-25
4	55	45	0	26	69	20
5	18	0	48	89	103	-6
6	82	8	18	56	90	3
7	11	78	31	66	86	-4
8	69	86	6	44	100	3
9	32	55	66	18	74	3
10	41	18	89	75	101	2

CPU TIME USED:

ASSEMBLY: 1.032 SECONDS

EXECUTION: 1.126 SECONDS

**** ON AT 01:41.50
**** OFF AT 01:42.26
**** ELAPSED TIME 36.063 SEC.
**** CPU TIME USED 6.289 SEC.
**** STORAGE USED 233.46 PAGE-SEC.
**** CARDS READ 61
**** LINES PRINTED 292
**** PAGES PRINTED 21
**** CARDS PUNCHED 1
**** DRUM READS
**** APPROX. COST OF THIS RUN $.77

FIGURE 19-C: SELECTED OUTPUT; TIME AND COST DATA
(JOB SHOP PROBLEM: FIRST-COME, FIRST-SERVED)

66. The Job Shop Problem: Job Slack per Operation Scheduling Rule

Statement of the Problem

Build a model analogous to the one presented in Section 65, using Job Slack per Operation (JSPO) as the scheduling rule. In particular, the model should be able to schedule an arbitrary number of jobs (all on hand at time zero) through a Job Shop consisting of an arbitrary number of machines. Implement the model to construct a schedule for 10 jobs, using the data presented in Section 65.

Approach Taken in Building the Model

The model of Figures 19-A and 19-B can be easily extended to meet the requirements of the JSPO scheduling rule. The changes involved are twofold:

1) The remaining number of operations must be available. This is actually carried in P3 for looping purposes, but in the model at hand will be kept up-to-date in column 1 of MH1 and used from that position to compute JSPO.

2) When it is time to select the next job for a machine, this sequence is followed:

   A) All jobs waiting for the machine are unhooked from the appropriate User Chain, their JSPO is computed, and they are then put back on the machine in order of increasing JSPO.

   B) Then the job at the front of the User Chain is unhooked and sent to the machine.

   Notice that JSPO should not be computed until the machine becomes available because the value of the clock at that time must be used in the computation.

Table of Definitions

GPSS Entity	Interpretation
Transaction	A job P1: Not used P2: Job number P3: Number of remaining operations P4: Imminent machine number P5: Imminent operation time P6: Job's "Job Slack per Operation" P7: Column Index pointing to imminent machine number in Job Information Matrix P8: Column Index pointing to imminent operation time in Job Information Matrix
Facility j	Machine j j = 1,2,3,...,MTOT, where MTOT represents the number of machines in the shop

Matrix 1 (Halfword)

Job Information Matrix

Rows 1,2,3,...,JTOT carry information about jobs 1,2,3,...,JTOT, respectively, where JTOT represents the number of jobs

Column	Information
1	Number of operations
2	Total operation time
3	Due date
4 5 : 3+MTOT	Numbers of the machines, appearing in the sequence in which the job requires them
4+MTOT 5+MTOT 3+2*MTOT	Operation times on the machines in columns 4,5,...,3+MTOT, respectively

Matrix 2 (halfword)

Starting Time and Completion Time Matrix

MH2(i,j) is the time job i went onto machine j, for i = 1,2,3,...,JTOT, and j = 1,2,3,...,MTOT

MH2(i,MTOT+1) is the time job i was completed

MH2(i,MTOT+2) is the hours early or late that job i was completed

Savevalue JTOT (Halfword)

Total number of jobs

Savevalue MTOT (Halfword)

Total number of machines

Table DISTN

Job Completion Table

User Chain j

Waiting Line for Machine j

j = 1,2,3,...,MTOT

Variable HOURS

Earliness hours or lateness hours relative to job completion; a negative value means that the job was completed late

Variable JOBS

Total jobs in shop minus one; (used as A Operand in SPLIT block)

Variable JSPO

Job Slack per Operation

Variable MACP1

Total machines in shop plus one

Variable MACP2

Total machines in shop plus two

Variable SHFT1

Total machines in shop plus three

Variable SLACK

Job slack

Variable TIME

Current value of clock minus one

Block Diagram

Figure 20-A shows the Block Diagram of the model.

Discussion of the Block Diagram

Features of the block diagram will be evident in light of the discussion in Section 65 and the remarks above. One elaboration may be worthwhile relative to Steps 2-A and 2B (Approach Taken in Building the Model). Notice how the Transaction which leaves the machine and initiates the unhooking of waiting Transactions waits for the chain to be re-formed before it then sends the Transaction at the head of the chain to the SEIZE Block. Such waiting is a logical necessity (if it went immediately to the second UNLINK Block before chain re-formation, the Transaction would find an empty chain).

Other Model Documentation

Figure	Information Exhibited
20-B	Program Listing
20-C	Selected Program Output; Time and Cost Data

Discussion of Program Output

The Table DISTN shows that the average job was completed 2.3 hours ahead of its due date, with a standard deviation of 9.1 hours (vs. 2.3 and 13.5, respectively via the first-in, first-out rule). Hence, for this limited sample at least, the JSPO rule does tend to get jobs finished closer to their due dates than does FIFO. Of the 10 jobs, 5 were actually late, 1 finished exactly on time, and 4 were early. Total lateness time is 22 hours, vs. 35 hours via FIFO.

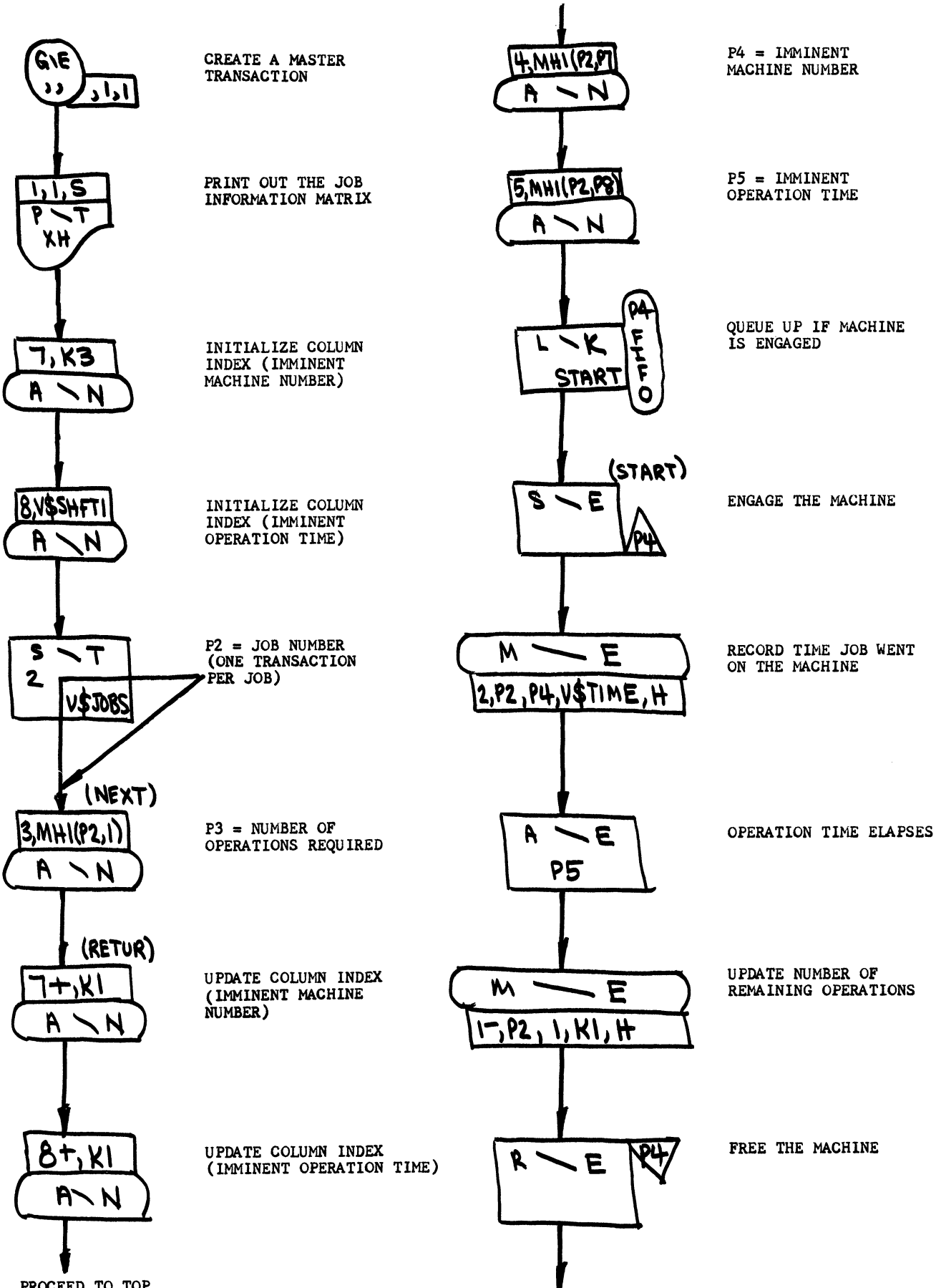


FIGURE 20-A: BLOCK DIAGRAM (CONTINUED ON NEXT PAGE)

(NEXT PAGE)

(JOB SHOP PROBLEM: JOB SLACK PER OPERATION)

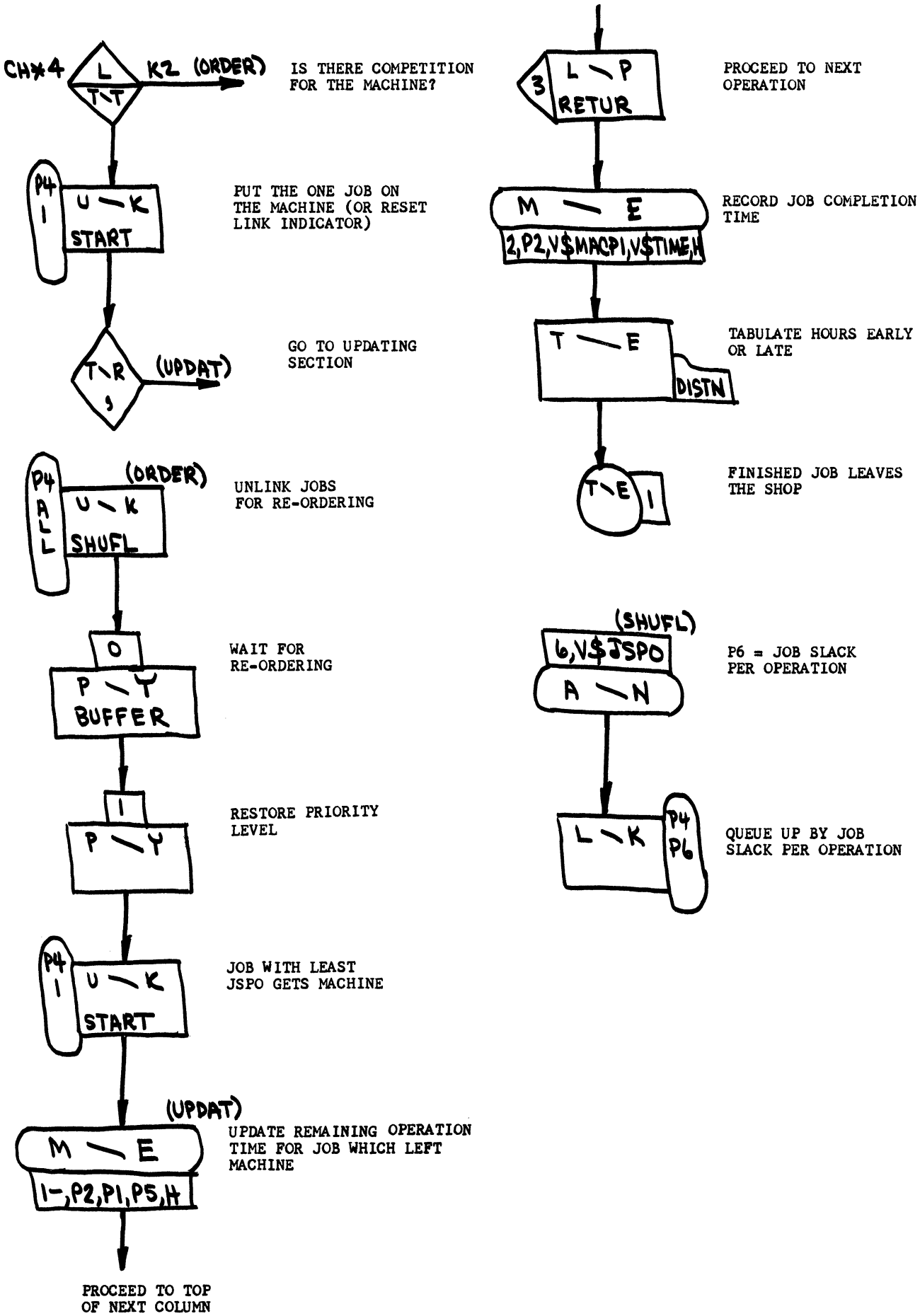


FIGURE 20-A: BLOCK DIAGRAM (CONTINUED FROM PRECEDING PAGE)

(JOB SHOP PROBLEM: JOB SLACK PER OPERATION)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
*       SIMULATE
        INITIAL     XH$MTOT,4/XH$JTOT,10
      1 MATRIX      H,10,11
      2 MATRIX      H,10,6
        INITIAL     MH1(1-10,1),4/MH1(1,2),35/MH1(2,2),39/MH1(3,2),23
        INITIAL     MH1(4,2),41/MH1(5,2),40/MH1(6,2),41/MH1(7,2),35
        INITIAL     MH1(8,2),46/MH1(9,2),33/MH1(10,2),42/MH1(1,3),84
        INITIAL     MH1(2,3),90/MH1(3,3),53/MH1(4,3),89/MH1(5,3),97
        INITIAL     MH1(6,3),93/MH1(7,3),82/MH1(8,3),103/MH1(9,3),77
        INITIAL     MH1(10,3),103/MH1(1,4),1/MH1(2-3,4),4/MH1(4,4),3
        INITIAL     MH1(5-6,4),2/MH1(7,4),1/MH1(8,4),3/MH1(9,4),4
        INITIAL     MH1(10,4),2/MH1(1,5),4/MH1(2,5),1/MH1(3,5),3
        INITIAL     MH1(4,5),4/MH1(5,5),1/MH1(6-7,5),3/MH1(8,5),4
        INITIAL     MH1(9-10,5),1/MH1(1-2,6),2/MH1(3,6),1/MH1(4,6),2
        INITIAL     MH1(5,6),3/MH1(6-7,6),4/MH1(8,6),1/MH1(9,6),2
        INITIAL     MH1(10,6),4/MH1(1-2,7),3/MH1(3,7),2/MH1(4,7),1
        INITIAL     MH1(5,7),4/MH1(6,7),1/MH1(7-8,7),2/MH1(9-10,7),3
        INITIAL     MH1(1,8),11/MH1(2,8),12/MH1(3-4,8),6/MH1(5,8),8
        INITIAL     MH1(6,8),10/MH1(7-8,8),7/MH1(9,8),8/MH1(10,8),11
        INITIAL     MH1(1,9),7/MH1(2,9),5/MH1(3,9),6/MH1(4,9),11
        INITIAL     MH1(5,9),9/MH1(6,9),13/MH1(7,9),11/MH1(8,9),12
        INITIAL     MH1(9,9),9/MH1(10,9),5/MH1(1,10),10/MH1(2,10),13
        INITIAL     MH1(3,10),6/MH1(4,10),10/MH1(5,10),9/MH1(6,10),10
        INITIAL     MH1(7,10),9/MH1(8,10),13/MH1(9,10),8/MH1(10,10),14
        INITIAL     MH1(1,11),7/MH1(2,11),9/MH1(3,11),5/MH1(4-5,11),14
        INITIAL     MH1(6-7,11),8/MH1(8,11),14/MH1(9,11),8/MH1(10,11),12
DISTN   TABLE       V$HOURS,-20,5,10
HOURS   VARIABLE    MH1(P2,3)-(C1-K1)
JOBS    VARIABLE    XH$JTOT-K1
JSPO    VARIABLE    V$SLACK/MH1(P2,1)
MACP1   VARIABLE    XH$MTOT+K1
MACP2   VARIABLE    XH$MTOT+K2
SHFT1   VARIABLE    XH$MTOT+K3
SLACK   VARIABLE    MH1(P2,3)-(C1-K1)-MH1(P2,2)
TIME    VARIABLE    C1-K1
        GENERATE    ,,,1,1          CREATE A MASTER XACT FOR THE MODEL
        PRINT       1,1,MH,S        PRINT OUT THE JOB FILE
        ASSIGN      7,K3            INITIALIZE COL INDEX FOR MACHINE NUMBER
        ASSIGN      8,V$SHFT1       INITIALIZE COL INDEX FOR MACHINE TIME
        SPLIT       V$JOBS,NEXT,2   CREATE 1 XACT PER JOB, P2 = JOB NUMBER
NEXT    ASSIGN      3,MH1(P2,1)     P3 = NUMBER OF OPERATIONS REQUIRED
RETUR   ASSIGN      7+,K1           UPDATE COL INDEX FOR MACHINE NUMBER
        ASSIGN      8+,K1           UPDATE COL INDEX FOR MACHINE TIME
        ASSIGN      4,MH1(P2,P7)    P4 = IMMINENT MACHINE NUMBER
        ASSIGN      5,MH1(P2,P8)    P5 = IMMINENT MACHINE TIME
        LINK        P4,FIFO,START   JOB WAITS ON CHAIN IF MACHINE IS BUSY
START   SEIZE       P4              JOB GOES ON THE MACHINE
        MSAVEVALUE  2,P2,P4,V$TIME,H     RECORD TIME JOB WENT ON THE MACHINE
        ADVANCE     P5              MACHINE TIME ELAPSES
        MSAVEVALUE  1-,P2,1,K1,H    UPDATE NUMBER OF REMAINING OPERATIONS
        RELEASE     P4              JOB GOES OFF THE MACHINE
        TEST L      CH*4,K2,ORDER   CH*4 < 2 => NO DECISION IN CHOICE OF NEXT JOB
        UNLINK      P4,START,1      SEND JOB TO SEIZE OR RESET LINK INDICATOR
        TRANSFER    ,UPDAT          XFER TO UPDATE MACHINE HOURS LEFT FOR THIS JOB
ORDER   UNLINK      P4,SHUFL,ALL    UNLINK ALL XACTS TO REORDER THEM VIA JSPO RULE
        PRIORITY    0,BUFFER        XACT WAITS UNTIL RE-ORDERING IS FINISHED
        PRIORITY    1               'NEUTRAL' PRIORITY LEVEL OF XACT IS RESET
        UNLINK      P4,START,1      SEND JOB WITH LEAST JSPO TO SEIZE MACHINE
UPDAT   MSAVEVALUE  1-,P2,2,P5,H    UPDATE REMAINING MACHINE TIME FOR THIS JOB
        LOOP        3,RETUR         SEND THIS JOB TO ITS NEXT MACHINE
        MSAVEVALUE  2,P2,V$MACP1,V$TIME,H     RECORD TIME JOB IS FINISHED
        MSAVEVALUE  2,P2,V$MACP2,V$HOURS,H    RECORD HOURS EARLY OR LATE BY JOB
        TABULATE    DISTN           TABULATE HOURS EARLY OR LATE
        TERMINATE   1               FINISHED JOB LEAVES THE SHOP
SHUFL   ASSIGN      6,V$JSPO        P6 = JOB'S 'JOB SLACK PER OPERATION'
        LINK        P4,P6           LINK JOB IN ORDER OF INCREASING JSPO
        START       10
        END
$SIGH
```

FIGURE 20-B: PROGRAM LISTING

(JOB SHOP PROBLEM: JOB SLACK PER OPERATION)

RELATIVE CLOCK 105 ABSOLUTE CLOCK 105

USER CHAIN	TOTAL ENTRIES	AVERAGE TIME/TRANS	CURRENT CONTENTS	AVERAGE CONTENTS	MAXIMUM CONTENTS
1	18	4.777		.819	2
2	21	5.190		1.038	3
3	13	5.692		.704	3
4	33	6.181		1.942	4

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
1	.828	10	8.699		
2	.923	10	9.699		
3	.838	10	8.799		
4	.980	10	10.299		

TABLE DISTN

ENTRIES IN TABLE	MEAN ARGUMENT	STANDARD DEVIATION	SUM OF ARGUMENTS
10	2.299	9.128	23.000

UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
-20	0	.00	.0	100.0	-8.695	-2.442
-15	0	.00	.0	100.0	-6.521	-1.895
-10	0	.00	.0	100.0	-4.347	-1.347
-5	3	29.99	29.9	70.0	-2.173	-.799
0	3	29.99	59.9	40.0	-.000	-.251
5	1	9.99	69.9	30.0	2.173	.295
10	1	9.99	79.9	20.0	4.347	.843
15	1	9.99	89.9	10.0	6.521	1.391
20	0	.00	89.9	10.0	8.695	1.938
OVERFLOW	1	9.99	100.0	.0		
AVERAGE VALUE OF OVERFLOW		21.00				

MATRIX HALFWORD SAVEVALUE 1

	COL. 1	2	3	4	5	6	7	8	9	10	11
ROW 1	4	35	84	1	4	2	3	11	7	10	7
2	4	39	90	4	1	2	3	12	5	13	9
3	4	23	53	4	3	1	2	6	6	6	5
4	4	41	89	3	4	2	1	6	11	10	14
5	4	40	97	2	1	3	4	8	9	9	14
6	4	41	93	2	3	4	1	10	13	10	8
7	4	35	82	1	3	4	2	7	11	9	8
8	4	46	103	3	4	1	2	7	12	13	14
9	4	33	77	4	1	2	3	8	9	8	8
10	4	42	103	2	1	4	3	11	5	14	12

MATRIX HALFWORD SAVEVALUE 2

	COL. 1	2	3	4	5	6
ROW 1	0	59	69	37	76	8
2	36	69	82	0	91	-1
3	41	54	31	12	59	-6
4	54	44	0	26	68	21
5	18	0	48	89	103	-6
6	81	8	18	65	89	4
7	11	82	37	56	90	-8
8	68	90	6	44	104	-1
9	27	36	57	18	65	12
10	47	18	91	75	103	0

CPU TIME USED:

ASSEMBLY: 1.242 SECONDS

EXECUTION: 1.395 SECONDS

**** ON AT 01:42.32
**** OFF AT 01:43.11
**** ELAPSED TIME 39.166 SEC.
**** CPU TIME USED 6.773 SEC.
**** STORAGE USED 260.913 PAGE-SEC.
**** CARDS READ 73
**** LINES PRINTED 321
**** PAGES PRINTED 21
**** CARDS PUNCHED 1
**** DRUM READS 20
**** APPROX. COST OF THIS RUN $.85

FIGURE 20-C: SELECTED OUTPUT; TIME AND COST DATA

(JOB SHOP PROBLEM: JOB SLACK PER OPERATION)

67. Synchronizing the Movement of Transactions in a Model

***In the course of building a model, the modeler occasionally wants to do one or the other (or both) of two things:

1) Hold Transactions at a particular point in a model until a certain number of them have arrived at that point, then permit them all to go on.

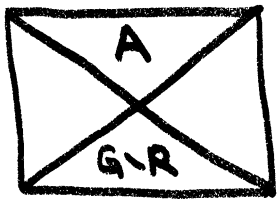
2) Hold a Transaction at a particular point, say Point A, until another Transaction reaches some other point, say Point B, at which time the two Transactions are permitted to continue their movement.

GPSS provides the first capability with the GATHER Block and the second capability with the MATCH Block.

The GATHER Block:

Purpose: Detain Transactions arriving at the Block until a certain number of them (termed the Gather Count) from the same Assembly Set are gathered there, at which time all of them move on to the next sequential Block.

Shape and Operands:



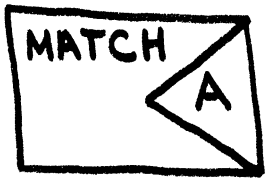
Operand	Significance
A	Specifies the Gather Count.

Comments: The GATHER Block can gather Transactions from more than one Assembly Set Concurrently. Also, the number of fatherings that may take place relative to a given Assembly Set during a simulation are unlimited.

The MATCH Block:

Purpose: Hold a Transaction at the point of the MATCH Block until another Transaction from the same Assembly Set has reached another particular MATCH Block elsewhere in the model, at which time both Transactions are permitted to move on.

Shape and Operands:



Operand	Significance
A	Specifies the symbolic name of the other particular MATCH Block involved in the Transaction synchronization.

Comments: The MATCH Block can be in the process of MATCHING Transactions from an arbitrary number of Assembly Sets concurrently.

Use of the MATCH Block will be illustrated in Section 70 in the simulation of a construction project.

68. Further Possibilities for Conditionally Directing Transactions to Non-Sequential Blocks

*** We have seen that both the GATE and TEST Blocks can be used to conditionally control the paths followed by Transactions (Transfer Mode usage).

*** Only one symbolic Block name is specified when the GATE or TEST Block is used in Transfer Mode.

*** GPSS provides another testing capability in which two symbolic Block names must be specified; (one of the two names can be specified, however, by "default" when one of the Blocks is the next sequential Block).

*** This testing capability, requiring use of the TRANSFER Block, can be structured via the mode of the test to produce any one of these three effects:

1) .ddd Mode: ("d" represents an arbitrary decimal digit) probabilistic selection of a "next Block" from two possibilities

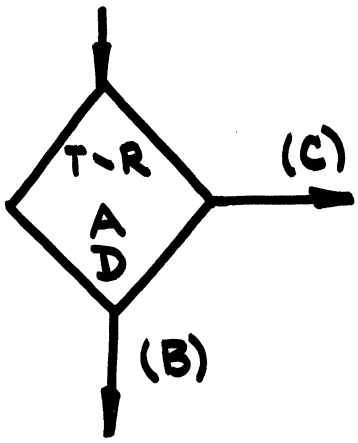
2) BOTH Mode: "no blockage" selection of a "next Block" from two possibilities

3) ALL Mode: "no blockage" selection of a "next Block" from an arbitrarily large number of possibilities

The TRANSFER Block in Conditional Transfer Mode:

Purpose: Selection of a "next Block" according to one of the three possibilities described above.

Shape and Operands:



Operand	Significance
A	Specifies the mode of the testing; in a specific case, A will be 1) .ddd, or 2) BOTH, or 3) ALL
B	Specifies one of the two required symbolic Block Names (may be deleted if the Block involved is the next sequential Block)

C	Specifies the other of the two required symbolic Block names
D	The D operand is used only when the A operand is ALL; in that case, the D operand specifies a Block indexing factor.

Nature of Operation:

The manner in which the TRANSFER Block operates in each of the three conditional transfer modes is now described:

1) .ddd Mode: Selection of the next Block is probabilistic. Suppose that in a specific case .ddd is .370. Then for each Transaction entering the TRANSFER Block, the probability is .370 that it will be routed to the Operand C Block (i.e. to the Block whose symbolic name is specified as the C operand), and .630 that it will be routed to the Operand B Block.

2) BOTH Mode: This sequence of steps is followed:

   A) The Transaction is routed to the Operand B Block unless it is refused entry there, in which case

   B) It is routed to the Operand C Block, unless it is also refused entry there, in which case

   C) It remains at the TRANSFER Block. Subsequent attempts will then be made to move the Transaction whenever the Current Events Chain scan is re-started (either at the same instant of simulated clock time, or at future readings of the simulated clock).

3) ALL Mode: This mode can perhaps best be explained by example. Suppose that a Transaction moves into the TRANSFER Block for which the punchcard equivalent is:

   ```
   TRANSFER   ALL,BLOK1,BLOKN,4
   ```

   Suppose further that BLOKI has the value 12 in the model, and BLOKN has the value 24. This sequence of steps is followed:

   A) The Transaction is routed to the Operand B Block unless it is refused entry there, in which case

   B) It is routed to the Operand C Block, unless it is also refused entry there, in which case

   C) It is routed to the "Operand B+2*D" Block (Block number 20), unless it is also refused entry there, in which case

   D) It is routed to the "Operand B+3*D" Block (Block number 24, i.e. BLOKN, the Operand C Block, which is the last candidate for testing), unless it is also refused entry there, in which case

E) It remains at the TRANSFER Block. Subsequent attempts are made to move the Transaction whenever the Current Events Chain scan is re-started.

Now it should be evident why the D Operand is termed a Block indexing factor. Note that the modeler must be certain there is some integer, call it I, such that "Operand B+I*D" = "Operand C".

Further Comment:

Another transfer mode, known as SIM (for simultaneous) is available for the TRANSFER Block. It is used to test whether a series of conditions are simultaneously satisfied. (The same effect can be accomplished by use of the TEST Block in conjunction with Boolean Variables.) The interested reader should consult the User's Manual.

69. GPSS Macros

***A Macro is any string of Blocks which a modeler may have occasion to repeat several times in a model. By defining the string of Blocks as a GPSS Macro, the user can arrange to have the entire string automatically inserted at any point in the model. The insertion is requested with a single card. The effect is to save both repetitive coding and keypunching time.

***There may be more than one Macro used in a model. Each Macro is given its own number (or symbolic name).

***Although a particular Macro always consists of the same Blocks, the Block Operands can depend on where the Macro is inserted in the model. Such call-dependent Operands are specified at the calling point. This is analogous to providing arguments when calling a subroutine.

***Using a Macro involves two things:

1) Defining the Macro. The Macro is defined by putting the string of Blocks between two control cards, the STARTMACRO and ENDMACRO cards, which take this form:

		1 1
card columns:	2---6	8---8 9---
contents:	j STARTMACRO	
	ENDMACRO	

where:

j is the number (or symbolic name) of the Macro being defined

These shapes will be used to represent the two control cards in block diagrams:

The Blocks making up the Macro take their usual appearance, except for Operands which are to be supplied in the call. There may be up to ten of these call-supplied Operands. In the Macro they are designated as #A, #B, #C, ... corresponding to the first argument, second argument, third argument, ... respectively, provided in the call. The #A, #B, etc. are literally entered as such wherever required in the Macro definition. In the assembly phase, GPSS then replaces them with the corresponding calling arguments. An example is given later.

2) Calling the Macro. The Macro is called with the MACRO card, which takes this form:

```
                                        1 1
card columns:      2---6 8---------8 9----------------------
  contents:          j   MACRO        A,B,C, ...
    where:
```

j is the number (or symbolic name) of the Macro, and A, B, C, ... are Standard Numerical Attributes to be used in the positions occupied by #A, #B, #C, ... respectively, in the Macro definition.

The hexagon is introduced to indicate the point of a Macro call in a block diagram. Block information appears as shown:

where j and #A, #B, #C, ... are as defined above.

Example:

Figure 21-A shows an example of a particular Macro. Notice that three of the Block Operands are to be supplied in the call. Figure 21-B shows how the Macro might be called from some point in a model. Figure 21-C illustrates how the Macro will appear when it is inserted at the point of the Figure 21-B call.

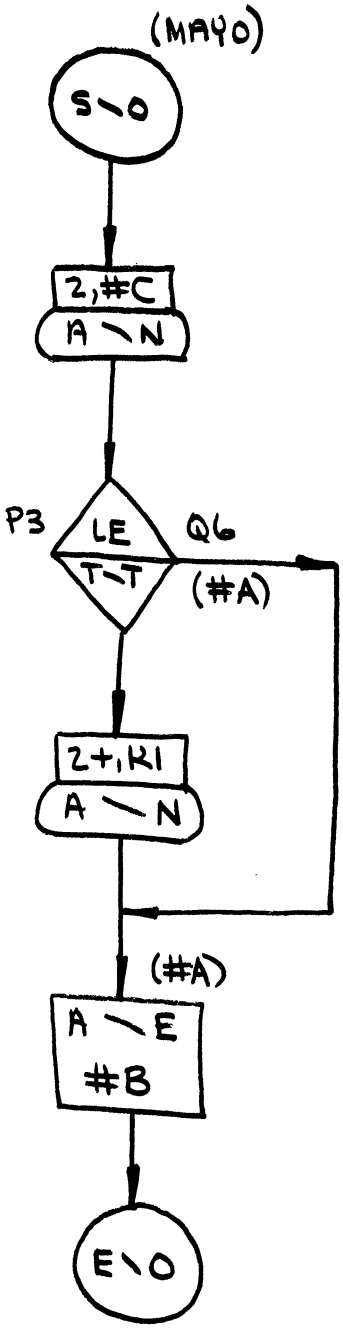


FIGURE 21-A: EXAMPLE OF A MACRO

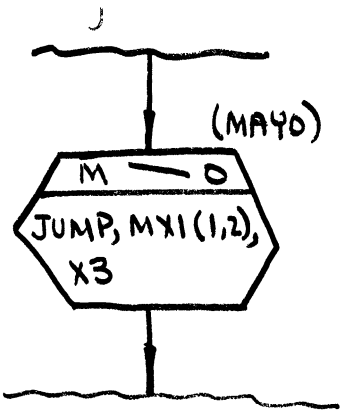


FIGURE 21-B: EXAMPLE OF A MACRO CALL

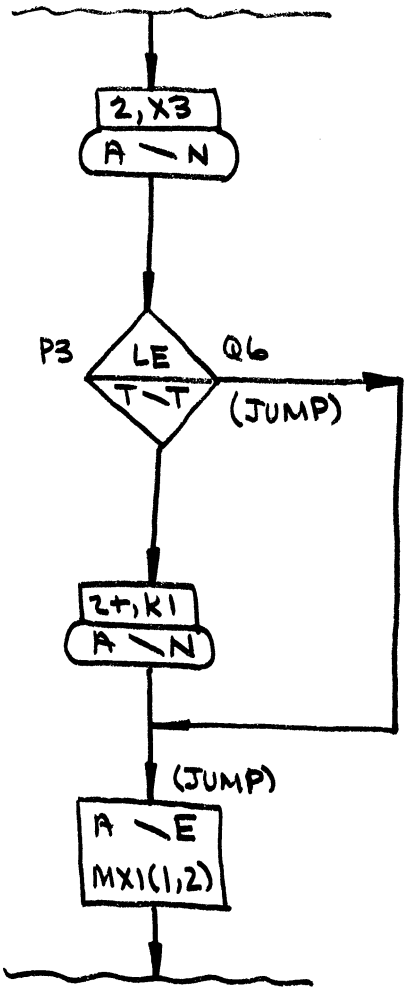
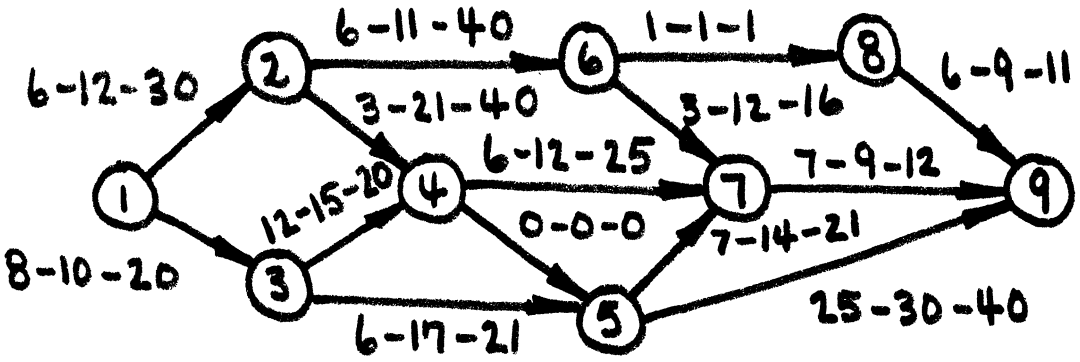


FIGURE 21-C: APPEARANCE OF THE MACRO AS INSERTED AT THE CALLING POINT

70. Simulation of a Construction Project

Statement of the Problem

The following network represents a series of sub-projects which must be carried out to complete an overall project:



A pair of circles (nodes) connected by a directed line segment is used to depict any particular sub-project. For example, node 1 is connected to node 3, depicting what is called sub-project 1-3. Each directed line segment is labeled with three numbers representing estimates of the time required to complete that particular sub-project. The three numbers express "optimistic time", "expected time", and "pessimistic time", respectively, thought to apply to the corresponding sub-project. For example, the expected time required to carry out sub-project 3-5 is 17 time units. If all goes well, the sub-project might be completed in as few as 6 time units. If problems are encountered, it might require as many as 21 time units.

The network also displays the precedence constraints on the various sub-projects. In many cases, one or more sub-projects must be completed before another one or more can be started. Consider these examples:

1) Sub-project 1-2 must be completed before either 2-4 or 2-6 can be started. (2-4 and 2-6, once started, proceed independently of each other.)

2) 2-4 and 3-4 must be finished before either 4-7 or 4-5 can be started. (Again, once they are started, 4-7 and 4-5 proceed independently of each other.)

3) Sub-projects 6-7, 4-7, and 5-7 must be finished before 7-9 can be begun.

Notice that the overall project is complete as soon as sub-projects 8-9, 7-9, and 5-9 are finished.

If sub-projects times are assumed to be known (by working, say, with only the expected time), a simple algorithm called Critical Path Method (CPM) gives the length of time required for the total project. The sub-projects which are especially obstructive are also singled out by CPM. They

---

R. Van Slyke, "Monte Carlo Methods and the PERT Problem", Operations Research, 11, 839-860 (1963). The network chosen here is identical to the one used in this article.

constitute what is called the "critical path" (or critical paths), because for them there is no "slack time", i.e. each critical sub-project must be started as soon as possible. Any arbitrary delay in a critical sub-project directly contributes to an increase in project completion time. Even without prior knowledge of CPM, the interested reader can quickly establish that project completion time for the above network is 63 times units, and the critical path consists of this sequence of sub-projects: 1-2; 2-4; 4-5; 5-9.

The CPM approach is generalized by PERT (Program Evaluation and Review Technique), which introduces uncertainty in the sub-project times by viewing them as random variables, then working with expected values and variances after assuming that particular probability density functions apply (usually the Beta).

The intention here is to take a Monte Carlo approach, determining sub-project times by sampling from assumed probability distributions. After building a GPSS model representing the project, it will be "carried out" a number of times, say 20, and the average project completion time, standard deviation, and frequency distribution will be tabulated. For this purpose, assume that sub-project time follows a triangular distribution based on the optimistic, expected, and pessimistic times shown in the network. Compare the results against the values of 69 (mean project time) and 9.6 (standard deviation) predicted by PERT on the assumption that the triangular distribution represents sub-project time distribution adequately.

Approach Taken in Building the Model

A Transaction is interpreted here as a sub-project foreman. Each sub-project has its own foreman. The number of foremen in the model depends, then, on how many sub-projects are underway, but may exceed the number underway. The reason for this is that after completing a sub-project, some foremen may have to wait for one or more other sub-projects to be completed before work is initiated on the next sub-project(s). This is of course due to the precedence relations imposed on the overall project.

Use of the MATCH and ASSEMBLE Blocks is indicated here. The MATCH concept is applied, for example, in noting that sub-projects 4-5 and 4-7 cannot begin until 3-4 and 2-4 are complete. If 3-4 is completed first, say, then that foreman waits in a MATCH Block until the foreman from 2-4 reaches the MATCH Block's mate, signaling that 2-4 is finished. The two foremen then immediately begin 4-7 and 4-5, respectively. The activities leading to and from node 5 provide an analogous situation for the MATCH Block. The general conclusion is that when an equal number of Transactions flow into and out of a project network node, the situation can be modeled in straightforward fashion with MATCH Blocks.

Node 7 is a situation in which three Transactions flow into a point, but after all three have arrived, only one continues on. This network situation has an exact GPSS correspondence in the ASSEMBLE Block. Node 9 is another such situation.

Finally, it is clear that when more Transactions flow from a node than into it, the SPLIT Block applies. Node 2 is such a node.

With use of the MATCH, ASSEMBLE, and SPLIT Blocks in mind, modeling the project network is straightforward. A further point is worth making, though. Sampling from an arbitrary triangular distribution requires using a handful of GPSS Blocks. The Block sequence must be provided whenever sub-project time is to be determined. Rather than punching these cards repeatedly, it is convenient to prepare a general block sequence as a GPSS MACRO, then call this MACRO as required, passing the time arguments to it in the call. The MACRO approach has been taken in this model.

Table of Definitions

Time Unit: 1/10-th the unit used in the network

GPSS Entity	Interpretation
Transaction	Sub-project Foreman P1: Optimistic sub-project time; Later, actual time if actual time is less than "expected time" P2: Expected sub-project time P3: Pessimistic sub-project time; Later, actual time if actual time is greater than "expected time" P4: Coded indication of whether actual time is less than expected (P4=1) or greater than expected (P4=3)
Function 1	The inverse cumulative distribution function corresponding to the density function $f(x) = 2x, 0 \leq x \leq 1$
Logic Switch 1	Control Switch used to signal that the overall project has been completed and the next pass (i.e. carrying out the overall project again) can now be started
Table 1	Project Completion Time Table
Variable 20	Variable whose value is the probability that actual sub-project time exceeds expected time; expressed in parts per thousand
Variable LDIF	Difference between expected time and optimistic time
Variable RDIF	Difference between pessimistic time and expected time
Variable SCALE	Variable which maps the model's time unit back into the time unit of the problem as stated

MACRO Arguments	Interpretation
#A	Symbolic name of the ADVANCE Block in the MACRO where sub-project time elapses
#B	Optimistic sub-project time
#C	Expected sub-project time
#D	Pessimistic sub-project time
#E	Symbolic name of an ASSIGN Block which is one of the Blocks in the MACRO

Block Diagrams

Figure 22-A shows the Block Diagram for the MACRO named STIME.
Figure 22-B shows the primary Block Diagram.

Discussion of the Block Diagrams

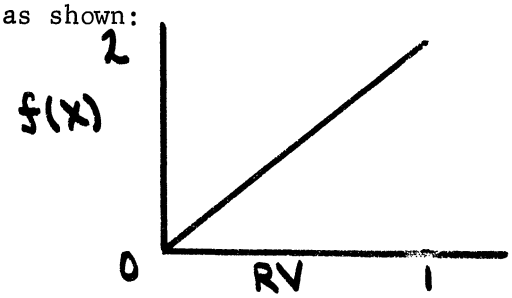
When a Transaction enters the MACRO segment, it first has Parameters 1,2, and 3 loaded with optimistic, expected, and pessimistic sub-project times, respectively, as provided in the call via zero-level addressing. Then the TRANSFER Block operating in ".ddd mode", shown as .V20, is used to determine which side of expected time the actual time is to lie on. For example, if the three times are 20-30-50, the actual time should exceed expected time 2/3-rds of the time.

V20 computes the 2/3rds as (50-30)/(50-20), then multiplies by 1000 (remember, Variables can only assume integer values). The "decimal point" is then provided directly as part of the TRANSFER Block's A Operand.

After the TRANSFER, Parameter 3 is marked accordingly, then actual time is determined either by:

1) Adding to optimistic time a fraction of the time span between optimistic and expected, or
2) Subtracting from pessimistic time a fraction of the time span between expected and pessimistic.

The "fraction" referred to is a random variable, call it RV, taking on values over the closed interval from 0 to 1 and following the triangular distribution as shown:



where f(x) is the linear density function.

It is a simple matter to show that the corresponding inverse cumulative distribution function is:

$$RV = +\sqrt{RN1}$$

where RN1 is interpreted as the GPSS representation of the cumulative distribution. This inverse function appears as Function 1 in the model.

The last Block in the MACRO is the ADVANCE, where sub-project time elapses according to the value registered in P1 or P3. Indirect addressing is used to determine which Parameter applies.

Figure 22-B shows the primary Block Diagram, which should be interpreted in light of the "Approach" discussion above. The Block Diagram format used until now has been abandoned in an attempt to make 22-B correspond more directly to the original project network. Remarks adjacent to the Blocks have been deliberately eliminated because of the increased complexity that would result. (Consult the Program Listing for remarks)

Other Model Documentation

Figure	Information Exhibited
22-C	Program Listing
22-D	Selected Program Output; Time and Cost Data

Discussion of Output

Per the entries in Table 1, average project time was 67.7 time units; standard deviation was 7.5 time units. This compares with PERT results of 69 and 9.6. In the article referenced, Van Slyke ran his model 10,000 times and obtained results of approximately 71.7 and 7.6 under assumption of the rectangular distribution. His model was coded for the IBM 7090. The Computer language used is not mentioned.

The motivation for Van Slyke's work lies in the fact that many different paths in a network could turn out to be critical, depending on sub-project times actually realized. Hence the tentative critical path established by PERT may not turn out to be critical after all. He introduces the concept of a "criticality index," which is the probability that any particular sub-project will lie on the critical path. Experimental approximations of these criticality indices are available as part of his model output. The interested reader is referred to the Van Slyke article.

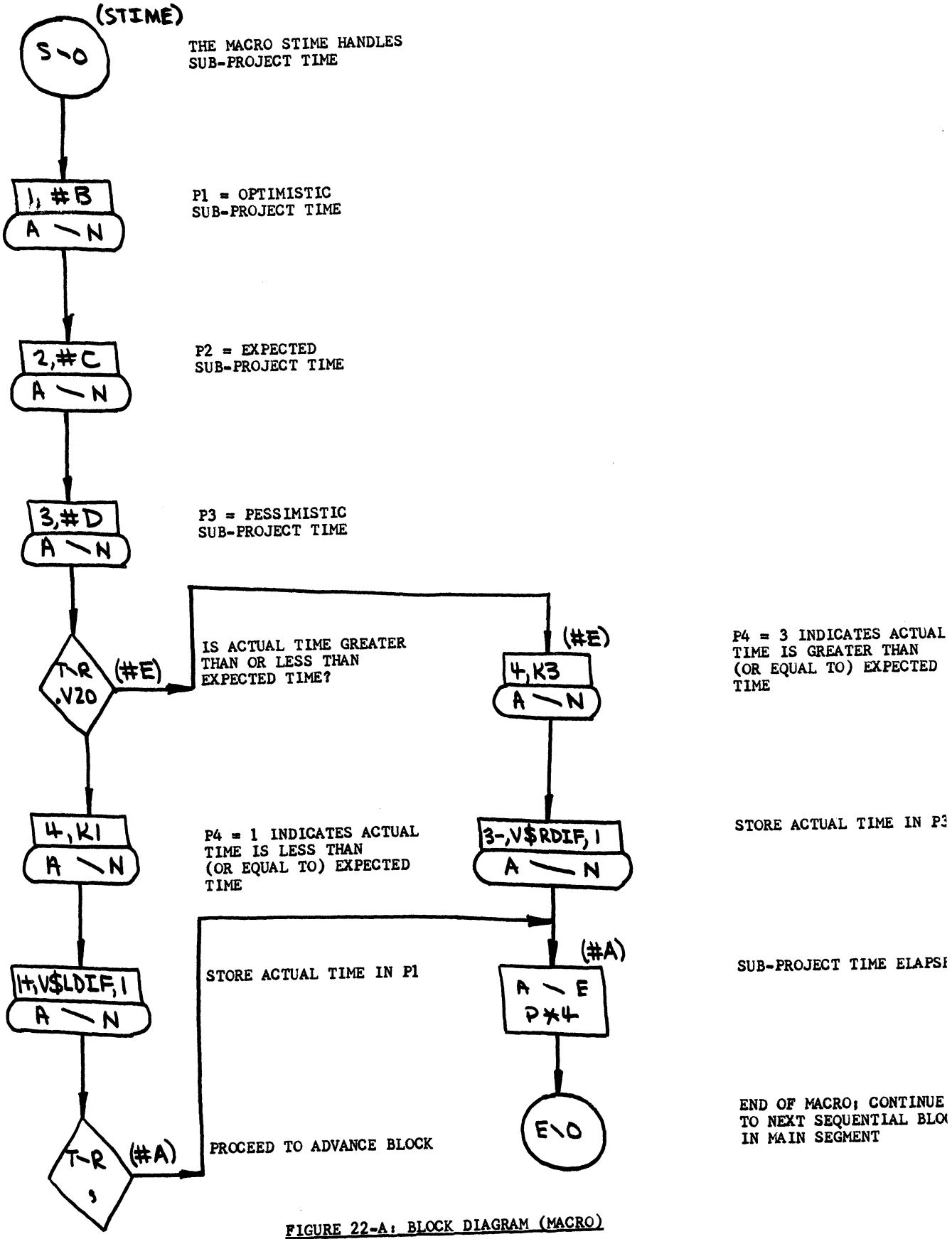


FIGURE 22-A: BLOCK DIAGRAM (MACRO)
(CONSTRUCTION PROJECT)

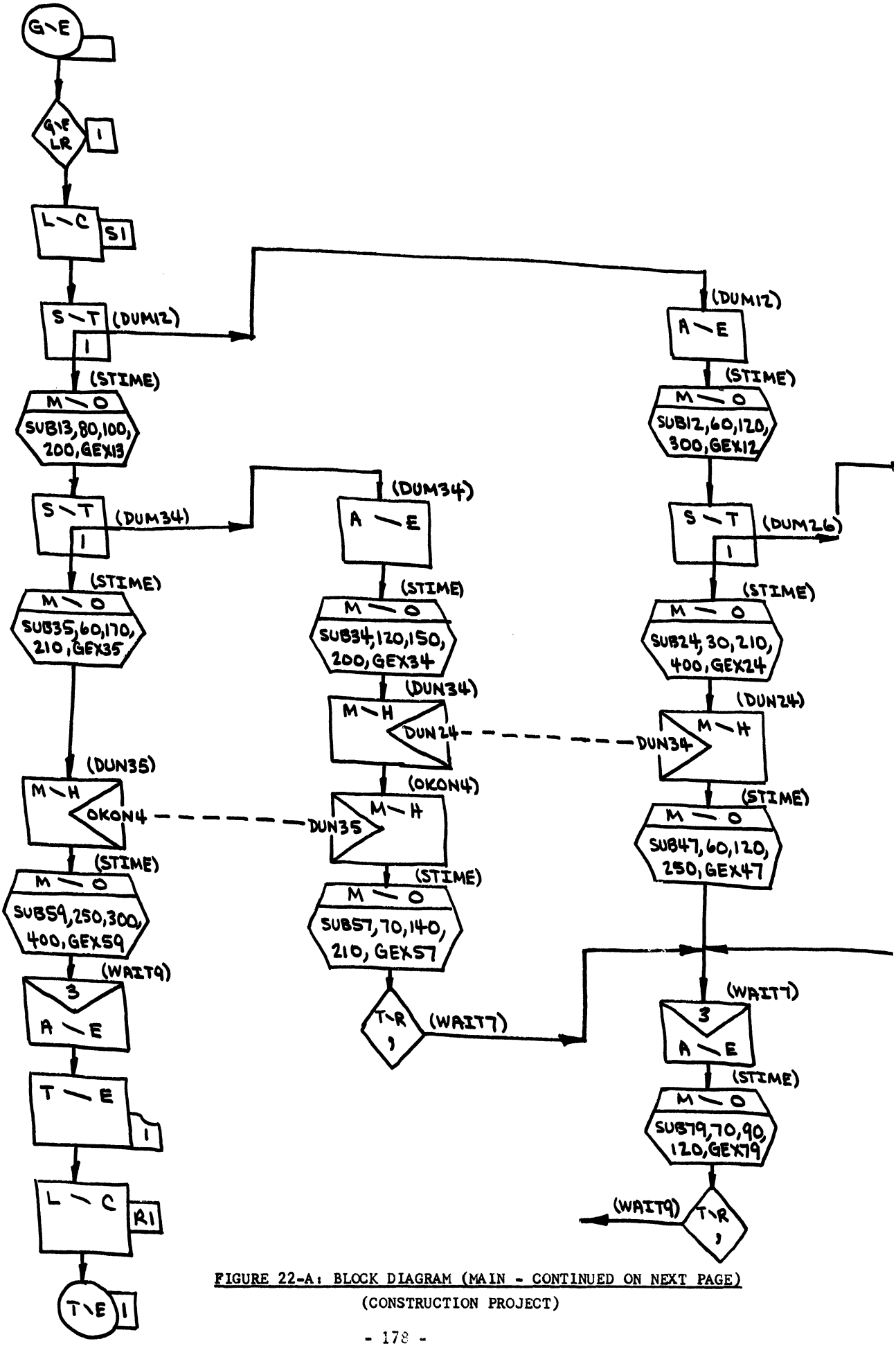


FIGURE 22-A: BLOCK DIAGRAM (MAIN - CONTINUED ON NEXT PAGE)

(CONSTRUCTION PROJECT)

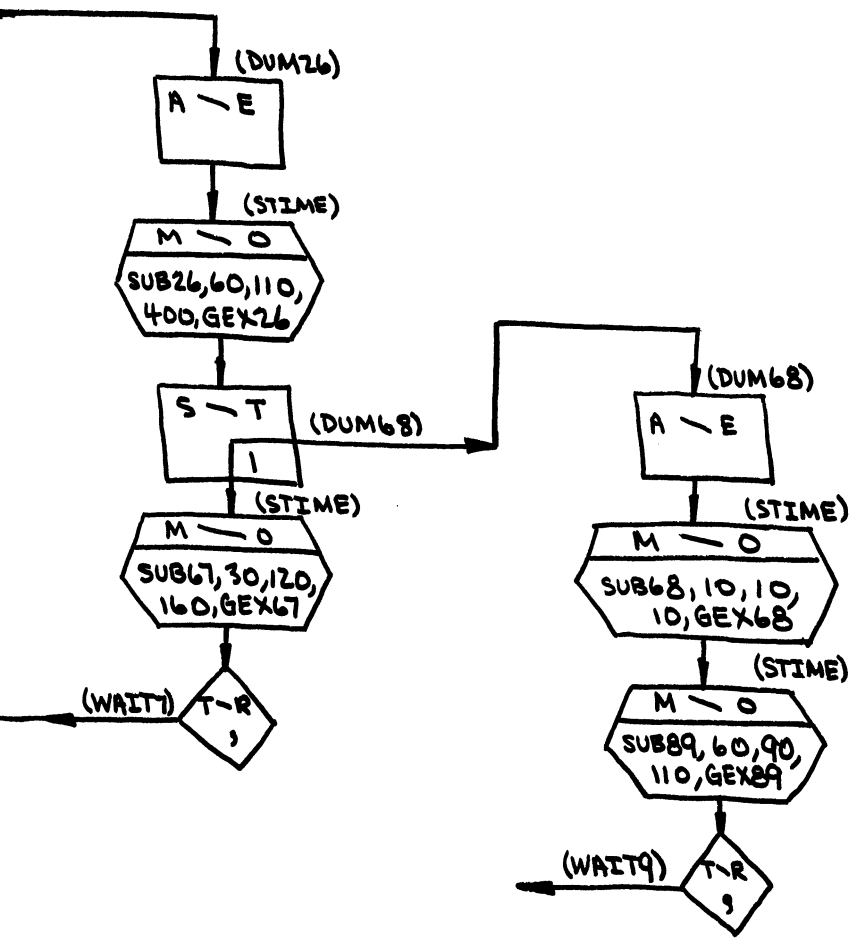


FIGURE 22-A: BLOCK DIAGRAM (MAIN - CONTINUED FROM PRECEDING PAGE)
(CONSTRUCTION PROJECT)

```
$SIGNON 505W PW=ATHENS 'THOMAS J. SCHRIBER'
$RUN *GPSS
       REALLOCATE TAB,1,LOG,1,STO,1,BLO,180
*      SIMULATE
     1 TABLE      V$SCALE,50,5,10
    20 VARIABLE   K1000*(P3-P2)/(P3-P1)
 LDIF  VARIABLE   P2-P1
 RDIF  VARIABLE   P3-P2
 SCALE VARIABLE   M1/K10
     1 FUNCTION   RN1,C11
0,0/.01,.1/.04,.2/.09,.3/.16,.4/.25,.5/.36,.6/.49,.7/.64,.8/.81,.9/1,1
 STIME STARTMACRO                 'STIME' HANDLES SUB-PROJECT TIME
       ASSIGN     1,#B            P1 = OPTIMISTIC SUB-PROJECT TIME
       ASSIGN     2,#C            P2 = EXPECTED SUB-PROJECT TIME
       ASSIGN     3,#D            P3 = PESSIMISTIC SUB-PROJECT TIME
       TRANSFER   .V20,,#E        SUB-PROJECT TIME <= OR >= EXPECTED TIME?
       ASSIGN     4,K1            P4 = 1 => ACTUAL TIME <= EXPECTED TIME
       ASSIGN     1+,V$LDIF,1     P1 = ACTUAL TIME
       TRANSFER   ,#A
 #E    ASSIGN     4,K3            P4 = 3 => ACTUAL TIME >= EXPECTED TIME
       ASSIGN     3-,V$RDIF,1     P3 = ACTUAL TIME
 #A    ADVANCE    P*4             SUB-PROJECT '#A' IS ACCOMPLISHED
       ENDMACRO                   SUB-PROJECT TIME SETUP HAS NOW BEEN HANDLED
       GENERATE                   RE-DO PROJECT WHEN PRECEDING PASS IS DONE
       GATE LR    1               SWITCH 1 SET => PRECEDING PASS NOT YET DONE
       LOGIC S    1               THIS PASS IS BEGINNING, SET SWITCH 1
       SPLIT      1,DUM12         SEND OFFSPRING TO SUB-PROJECT 1-2
 STIME MACRO      SUB13,80,100,200,GEX13    ACCOMPLISH SUB-PROJECT 1-3
       SPLIT      1,DUM34         SEND OFFSPRING TO SUB-PROJECT 3-4
 STIME MACRO      SUB35,60,170,210,GEX35    ACCOMPLISH SUB-PROJECT 3-5
 DUN35 MATCH      OKON4           WAIT UNTIL 3-4 AND 2-4 ARE COMPLETE
 STIME MACRO      SUB59,250,300,400,GEX59   ACCOMPLISH SUB-PROJECT 5-9
 WAIT9 ASSEMBLE   3               WAIT UNTIL 7-9 AND 8-9 ARE COMPLETE
       TABULATE   1               RECORD OVERALL PROJECT TIME IN TABLE
       LOGIC R    1               THIS PASS IS FINISHED, RESET SWITCH 1
       TERMINATE  1               UPDATE RECORD OF TIMES PROJECT HAS BEEN DONE
 DUM34 ADVANCE                    DUMMY TRANSFER POINT AHEAD OF THE MACRO
 STIME MACRO      SUB34,120,150,200,GEX34   ACCOMPLISH SUB-PROJECT 3-4
 DUN34 MATCH      DUN24           WAIT UNTIL 2-4 IS COMPLETE
 OKON4 MATCH      DUN35           WAIT UNTIL 3-5 IS COMPLETE
 STIME MACRO      SUB57,70,140,210,GEX57    ACCOMPLISH SUB-PROJECT 5-7
       TRANSFER   ,WAIT7          GO TO WAIT UNTIL 4-7 AND 6-7 ARE COMPLETE
 DUM12 ADVANCE                    DUMMY TRANSFER POINT AHEAD OF THE MACRO
 STIME MACRO      SUB12,60,120,300,GEX12    ACCOMPLISH SUB-PROJECT 1-2
       SPLIT      1,DUM26         SEND OFFSPRING TO SUB-PROJECT 2-6
 STIME MACRO      SUB24,30,210,400,GEX24    ACCOMPLISH SUB-PROJECT 2-4
 DUN24 MATCH      DUN34           WAIT UNTIL 3-4 IS COMPLETE
 STIME MACRO      SUB47,60,120,250,GEX47    ACCOMPLISH SUB-PROJECT 4-7
 WAIT7 ASSEMBLE   3               WAIT UNTIL 5-7 AND 6-7 ARE COMPLETE
 STIME MACRO      SUB79,70,90,120,GEX79     ACCOMPLISH SUB-PROJECT 7-9
       TRANSFER   ,WAIT9          GO TO WAIT UNTIL 5-9 AND 8-9 ARE COMPLETE
 DUM26 ADVANCE                    DUMMY TRANSFER POINT AHEAD OF THE MACRO
 STIME MACRO      SUB26,60,110,40,GEX26     ACCOMPLISH SUB-PROJECT 2-6
       SPLIT      1,DUM68         SEND OFFSPRING TO SUB-PROJECT 6-8
 STIME MACRO      SUB67,30,120,160,GEX67    ACCOMPLISH SUB-PROJECT 6-7
       TRANSFER   ,WAIT7          GO TO WAIT UNTIL 5-7 AND 4-7 ARE COMPLETE
 DUM68 ADVANCE                    DUMMY TRANSFER POINT AHEAD OF THE MACRO
 STIME MACRO      SUB68,100,100,100,GEX68   ACCOMPLISH SUB-PROJECT 6-8
 STIME MACRO      SUB89,60,90,110,GEX89     ACCOMPLISH SUB-PROJECT 8-9
       TRANSFER   ,WAIT9          GO TO WAIT UNTIL 5-9 AND 7-9 ARE COMPLETE
       START      20              COMPLETE THE PROJECT 20 DIFFERENT TIMES
       END
$SIGH
```

FIGURE 22-B: PROGRAM LISTING
(CONSTRUCTION PROJECT)

TABLE 1

ENTRIES IN TABLE	MEAN ARGUMENT	STANDARD DEVIATION	SUM OF ARGUMENTS
20	67.699	7.453	1354.000

UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
50	0	.00	.0	100.0	.738	-2.374
55	0	.00	.0	100.0	.812	-1.703
60	3	14.99	14.9	85.0	.886	-1.033
65	7	34.99	49.9	50.0	.960	-.362
70	5	25.00	74.9	25.0	1.033	.308
75	1	4.99	79.9	20.0	1.107	.979
80	3	14.99	94.9	5.0	1.181	1.650
85	1	4.99	100.0	.0	1.255	2.321

REMAINING F

CPU TIME USED:

ASSEMBLY: 2.169 SECONDS

EXECUTION: 2.366 SECONDS

**** ON AT 01:40.49
**** OFF AT 01:41.44

**** ELAPSED TIME	54.81	SEC.
**** CPU TIME USED	8.793	SEC.
**** STORAGE USED	409.42	PAGE-SEC.
**** CARDS READ	65	
**** LINES PRINTED	528	
**** PAGES PRINTED	17	
**** CARDS PUNCHED	1	
**** DRUM READS	34	
**** APPROX. COST OF THIS RUN		$1.12

FIGURE 22-C: SELECTED OUTPUT; TIME AND COST DATA
(CONSTRUCTION PROJECT)

71. The Output Editor

Output occurring at the end of a GPSS simulation (via a blank START Card B Operand) is of fixed form and content. Output occurring during the course of a simulation (PRINT Block output) permits the user to be partially selective in terms of content but still provides no flexibility in terms of form. There is a third type of output made possible via the GPSS/360 Output Editor. Such output, occurring at the end of a simulation, enables the user not only to choose the output content but also to design the format in which it will appear. In addition, the output can be labeled to facilitate its interpretation in the context of the system being modeled. Finally, with the Output Editor various statistics can be displayed graphically as well as in tabular form, adding to the ease of interpretation.

The Output Editor is described in Appendix C of the User's Manual. As indicated there, its use permits these flexibilities:

1) Select the output statistics of interest.

2) Title appropriate sections of the output and/or insert comments.

3) Control spacing, page skipping, and order of output.

4) Display GPSS/360 Standard Numerical Attribute values in graphic format.

Details of the Output Editor will not be chronicled here. An indication of its capabilities is given, however, in the next Section, were the Equipment Balancing Problem of Section 30 is re-modeled to study various machine:crane ratios in parallel. At the end of the simulation, the Output Editor is used to produce Average Waiting Times (machines waiting for a crane) and Crane Utilizations in graphic form. This model was contributed by Mr. James O. Henriksen, a Master's candidate in The Graduate School of Business Administration.

72. The Equipment Balancing Problem Revisited

Statement of the Problem

Build another GPSS model for the Equipment Balancing Problem (Section 30, page 65), incorporating these features into the model:

1) Rather than simulating various Machine:Crane ratios on a sequential basis, the model should study these nine ratios in one parallel model:

   2:1; 3:1; 4:1; 4:2; 6:2; 8:2; 6:3; 9:3; 12:3

2) In addition to tabulating data for Crane Utilizations and Average Machine Waiting Times, produce the utilizations and average waiting times in graphic form. All Crane Utilization information should appear on one graph; Average Machine Waiting Time prior to the "Load-Unload" phase should be on another graph; and Average Machine Waiting Time prior to the "Handle" phase should be on a third graph.

3) Turn off the simulation after 10,000 hours of simulated time.

Approach Taken in Building the Model

The approach taken is essentially that described on pages 65-66, except that appropriate SPLIT Blocks are used to set up the nine parallel configurations. Also, there must be 18 different Queues (Queues at two different points in each of the 9 configurations). Queue numbers are computed and assigned as Parameter values. Storages are used to represent cranes in each configuration and there must, of course, be nine of these. Queues and Storages are referenced, of course, with first-level addressing. In short, the pre-Output Editor portion of the model poses no problems. Details of the Output Editor portion should be reviewed after Users Manual Appendix C has been studied.

Table of Definitions

Time Unit: 1 Hour

GPSS Entity	Interpretation
Transaction	A Casting
	P1: Code indicating the Machine:Crane ratio in the configuration through which the casting moves

Value	Ratio
1	2:1
2	3:1
3	4:1
4	4:2
5	6:2
6	8:2
7	6:3
8	9:3
9	12:3

Alternatively, number of the "Load-Unload" Queue

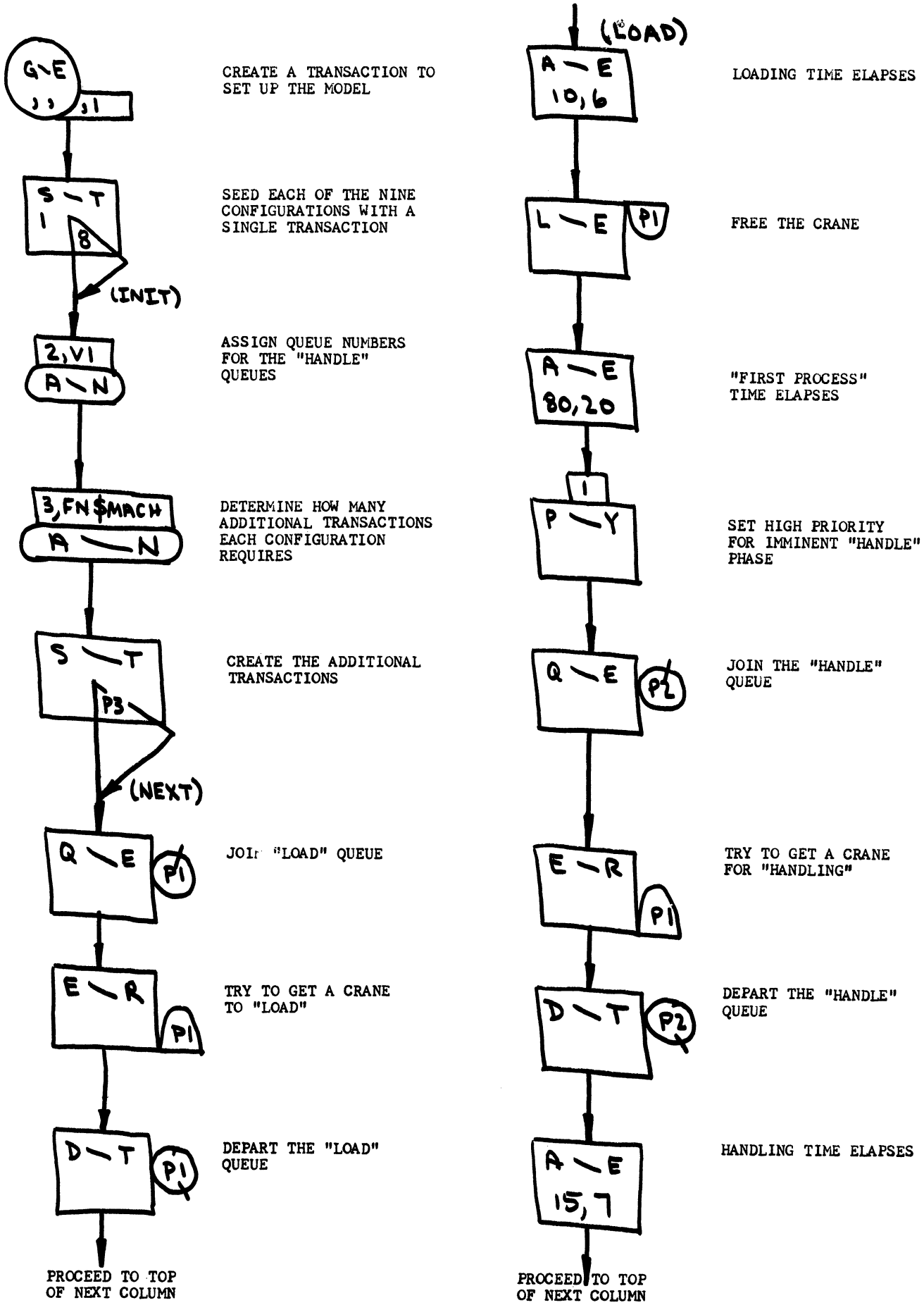
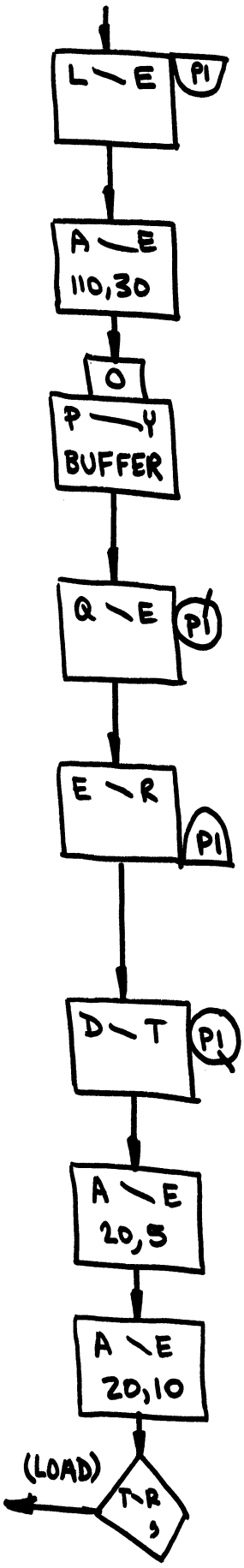


FIGURE 23-A: BLOCK DIAGRAM (CONTINUED ON NEXT PAGE)

(EQUIPMENT BALANCING PROBLEM REVISITED)



FREE THE CRANE

"SECOND PROCESS"
TIME ELAPSES

SET LOW PRIORITY FOR
IMMINENT "UNLOAD-LOAD"
PHASE; RESTART SCAN TO
DETERMINE IF CRANE JUST
FREED IS UNDER HIGH
PRIORITY REQUEST

JOIN THE "UNLOAD-LOAD"
QUEUE

TRY TO GET A CRANE
TO "UNLOAD-LOAD"

DEPART THE "UNLOAD-LOAD"
QUEUE

UNLOADING TIME ELAPSES

STORE AND FETCH
TIME ELAPSES

GO TO THE LOADING
TIME BLOCK

FIGURE 23-A: BLOCK DIAGRAM (CONTINUED FROM PRECEDING PAGE)

(EQUIPMENT BALANCING PROBLEM REVISITED)

```
*       FN$MACH CONTROLS THE MACHINE:CRANE RATIO
*
*
MACH    FUNCTION    N$INIT,L9
1,1/2,2/3,3/4,3/5,5/6,7/7,5/8,8/9,11
*
*
1       VARIABLE    P1+9
*
*
*       STORAGE CAPACITIES ARE THE NUMBERS OF CRANES
*
*
        STORAGE     S1-S3,1         1 CRANE
        STORAGE     S4-S6,2         2 CRANES
        STORAGE     S7-S9,3         3 CRANES
*
*
*       MACHINE-CRANE SEGMENT
*
*
        SIMULATE
        GENERATE    ,,,1            INITIAL PARENT TRANSACTION
        SPLIT       8,INIT,1        SET UP THE NINE PARALLEL MODELS
INIT    ASSIGN      2,V1            SET UP PAR 2 FOR "HANDLE" QUEUE
        ASSIGN      3,FN$MACH       SET UP MACHINE TO CRANE RATIO
        SPLIT       P3,NEXT         CREATE THE JOB XACTS FOR EACH MODEL
NEXT    QUEUE       P1              JOIN "LOAD-UNLOAD" QUEUE
        ENTER       P1              TRY TO GET A CRANE
        DEPART      P1              DEPART THE "LOAD-UNLOAD" QUEUE
LOAD    ADVANCE     10,6            LOADING TIME ELAPSES
        LEAVE       P1              FREE THE CRANE
        ADVANCE     80,20           FIRST PROCESS TIME ELAPSES
        PRIORITY    1               "HANDLE" PHASE HAS HIGHER PRIORITY
        QUEUE       P2              JOIN "HANDLE" QUEUE
        ENTER       P1              TRY TO GET A CRANE
        DEPART      P2              DEPART THE "HANDLE" QUEUE
        ADVANCE     15,7            HANDLING TIME ELAPSES
        LEAVE       P1              FREE UP THE CRANE
        ADVANCE     110,30          SECOND PROCESS TIME ELAPSES
        PRIORITY    0,BUFFER        "LOAD-UNLOAD" PHASE OF LOWER PRIORITY
        QUEUE       P1              JOIN "LOAD-UNLOAD" QUEUE
        ENTER       P1              TRY TO GET A CRANE
        DEPART      P1              DEPART THE "LOAD-UNLOAD" QUEUE
        ADVANCE     20,5            UNLOADING TIME ELAPSES
        ADVANCE     20,10           STORE + FETCH TIME ELAPSES
        TRANSFER    ,LOAD           GO RELOAD THE MACHINE
```

FIGURE 23-B: PROGRAM LISTING (CONTINUED ON NEXT PAGE)

(EQUIPMENT BALANCING PROBLEM REVISITED)

```
*
*       TIMER SEGMENT
*
*
      GENERATE    ,,10000,1,3      GENERATE THE TIMER XACT
      TERMINATE   1                TURN OFF THE MODEL
*
*
      START       1                BEGIN SIMULATION
*
*
*       SPECIAL OUTPUT SPECIFICATIONS
*
*
      REPORT
      EJECT                        A FRESH PAGE

      GRAPH       QT,1,9           AVERAGE TIMES IN QUEUES 1-9
      ORIGIN      49,6             POSITION THE HISTOGRAM
      X           ,3,10,,,,NO      SET UP THE X-AXIS
      Y           0,5,6,7          SET UP THE Y-AXIS
17    STATEMENT   50,106,2:1           3:1          4:1          4:2     1
     6:2          8:2          6:3          9:3          12:3
48    STATEMENT   53,44,AVERAGE WAITING TIMES IN "LOAD-UNLOAD" PHASE
53    STATEMENT   55,34,FOR MACHINE:CRANE RATIOS INDICATED
      ENDGRAPH
      GRAPH       QT,10,18         AVERAGE TIMES IN QUEUES 10-18
      ORIGIN      49,6             POSITION THE HISTOGRAM
      X           ,3,10,,,,NO      SET UP THE X-AXIS
      Y           0,5,6,7          SET UP THE Y-AXIS
17    STATEMENT   50,106,2:1           3:1          4:1          4:2     1
     6:2          8:2          6:3          9:3          12:3
50    STATEMENT   53,39,AVERAGE WAITING TIMES IN "HANDLE" PHASE
53    STATEMENT   55,34,FOR MACHINE:CRANE RATIOS INDICATED
      ENDGRAPH
      GRAPH       SR,1,9           CRANE UTILIZATION
      ORIGIN      49,6             POSITION THE HISTOGRAM
      X           ,3,10,,,,NO      SET UP THE X-AXIS
      Y           .40,.10,6,7      SET UP THE Y-AXIS
17    STATEMENT   50,106,2:1           3:1          4:1          4:2     1
     6:2          8:2          6:3          9:3          12:3
62    STATEMENT   53,17,CRANE UTILIZATION
53    STATEMENT   55,34,FOR MACHINE:CRANE RATIOS INDICATED
      ENDGRAPH
QUE   TITLE       ,ACTUAL QUEUE STATISTICS
QUE   INCLUDE     1-18/1,2,3,7,8   SELECT DESIRED COLUMNS OF OUTPUT
      SPACE       2                SKIP 2 LINES
*  NOTE: STATISTICS APPEAR IN THE SAME ORDER AS THOSE ON THE HISTOGRAMS
      EJECT
STO   TITLE       ,ACTUAL STORAGE STATISTICS (CRANE USAGE)
STO   INCLUDE     1-9/1,2,3,4,6    SELECT DESIRED COLUMNS OF OUTPUT
      SPACE       2                SKIP 2 LINES
*  NOTE: STATISTICS APPEAR IN THE SAME ORDER AS THOSE ON THE HISTOGRAMS
      END
```

FIGURE 23-B: PROGRAM LISTING (CONTINUED FROM PRECEDING PAGE)

(EQUIPMENT BALANCING PROBLEM REVISITED)

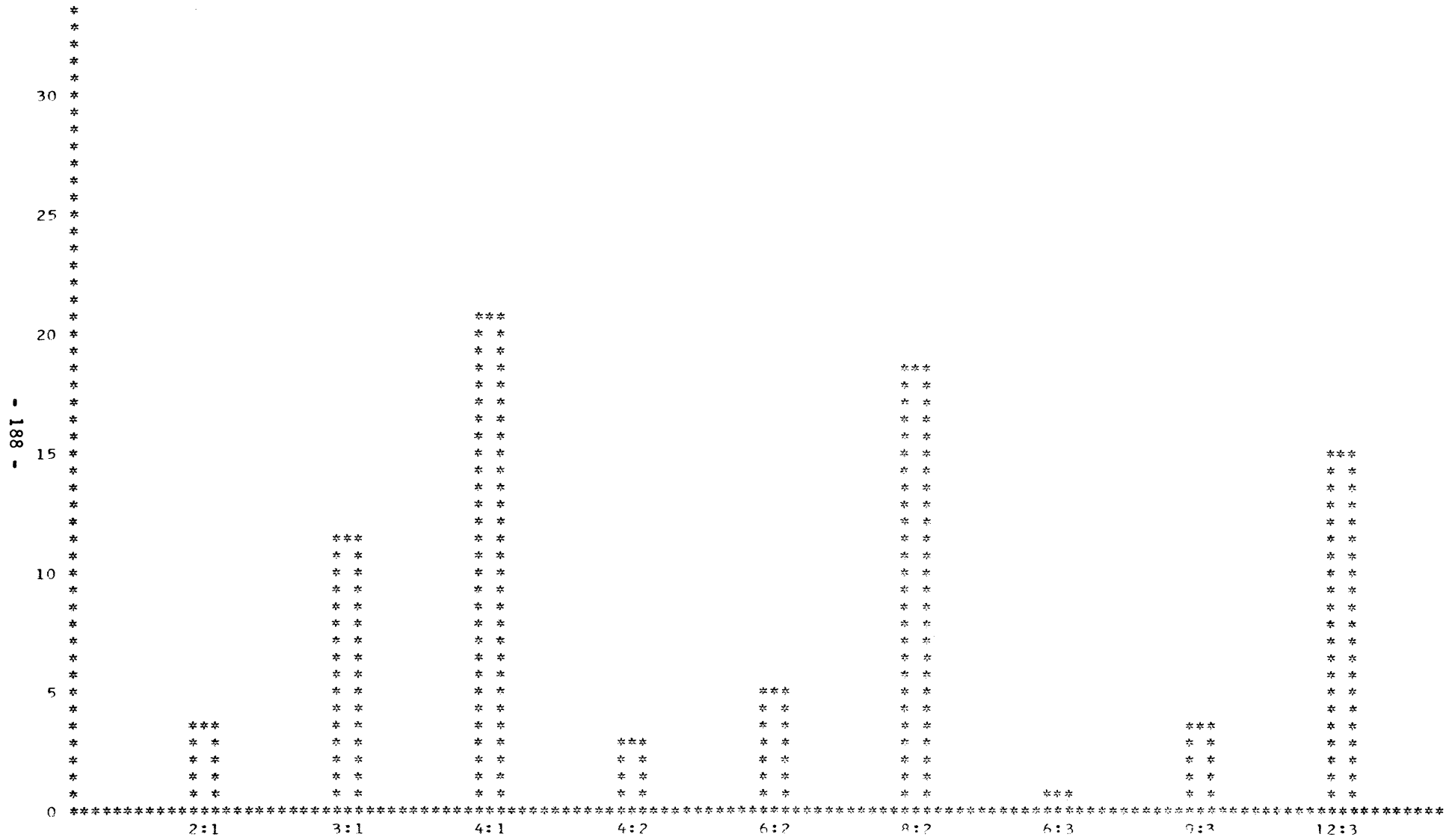


FIGURE 23-C: GRAPHICAL OUTPUT - AVERAGE WAITING TIMES IN "LOAD-UNLOAD" PHASE
(EQUIPMENT BALANCING PROBLEM REVISITED)

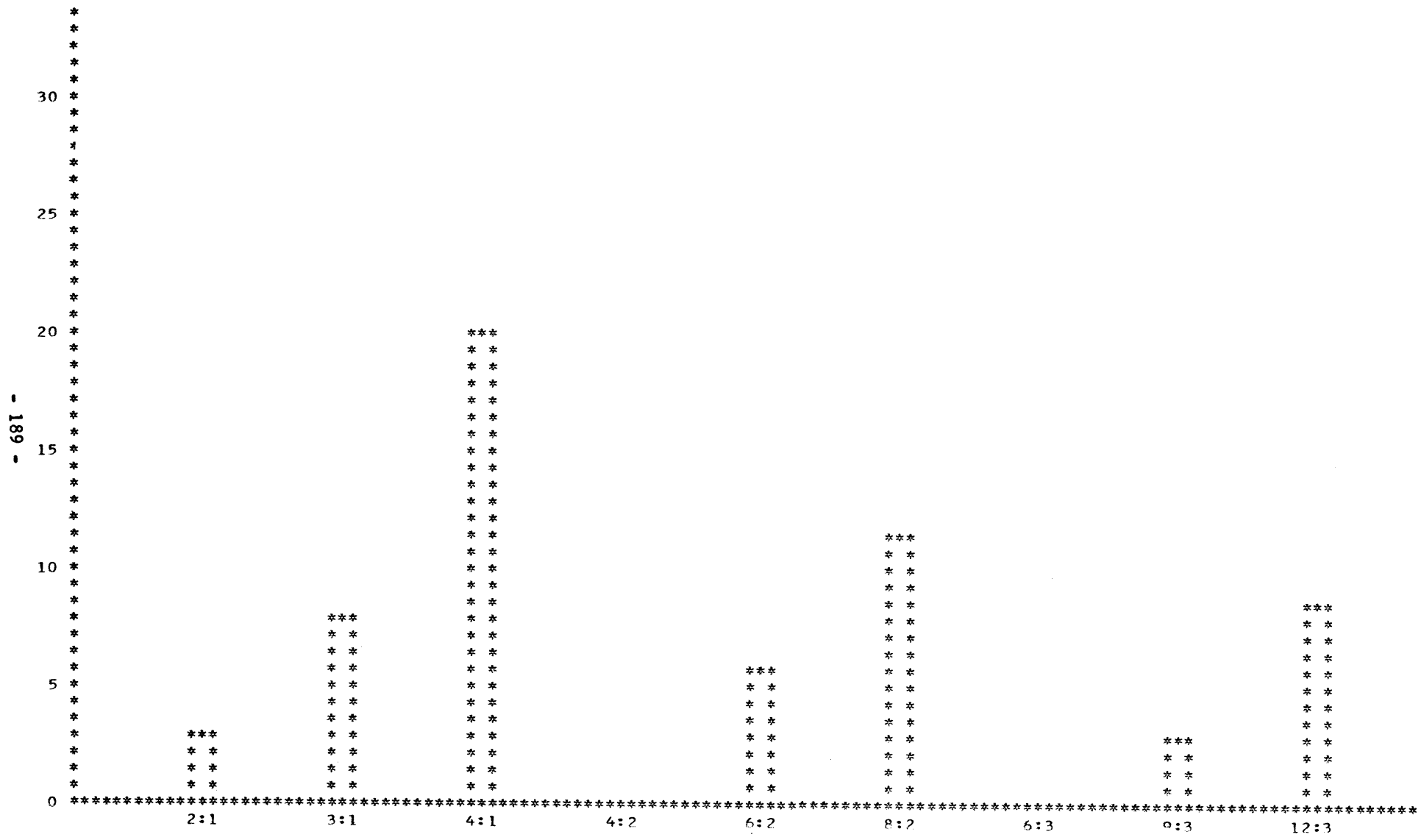


FIGURE 23-D: GRAPHICAL OUTPUT - AVERAGE WAITING TIMES IN "HANDLE" PHASE
(EQUIPMENT BALANCING PROBLEM REVISITED)

P2: Number of the "Handle" Queue; Values are 10, 11, 12, ... , 18 for P1's of 1,2,3,...,9, respectively

P3: Value is the number of additional Transactions to create so there will be the right number of castings moving through each configuration

Function MACH	Function with which the correct P3 value is determined and assigned
Storages 1 - 9	Storages having capacities of 1,1,1,2,2,2,3,3,3, respectively, representing the cranes for each of the nine ratios being studied. Used to determine crane availability and utilization
Variable 1	Variable with which the correct P2 value is determined and assigned

Other Model Documentation

Figure	Information Exhibited
23-A	Block Diagram
23-B	Program Listing
23-C	Graphical Output
23-D	Graphical Output
23-E	Graphical Output
23-F	Tabular output; Time and Cost Data

Discussion

In the Barber Shop: Fifth Model (Section 26) it was observed that for a given customer arrival rate per barber, customers wait less on average when there are three barbers in the shop than when there is one. This effect, consistent as it is with intuition, is observed again in Figure 23-E where crane utilization rises progressively for Machine:Crane ratios of 3:1, 6:2, and 9:3; for example. Similar observations can be made with Figures 23-C and 23-D. The model reveals more information than that in Section 30, since a greater number of ratios are studied and cases having 2 or 3 cranes servicing various machine groups are included.

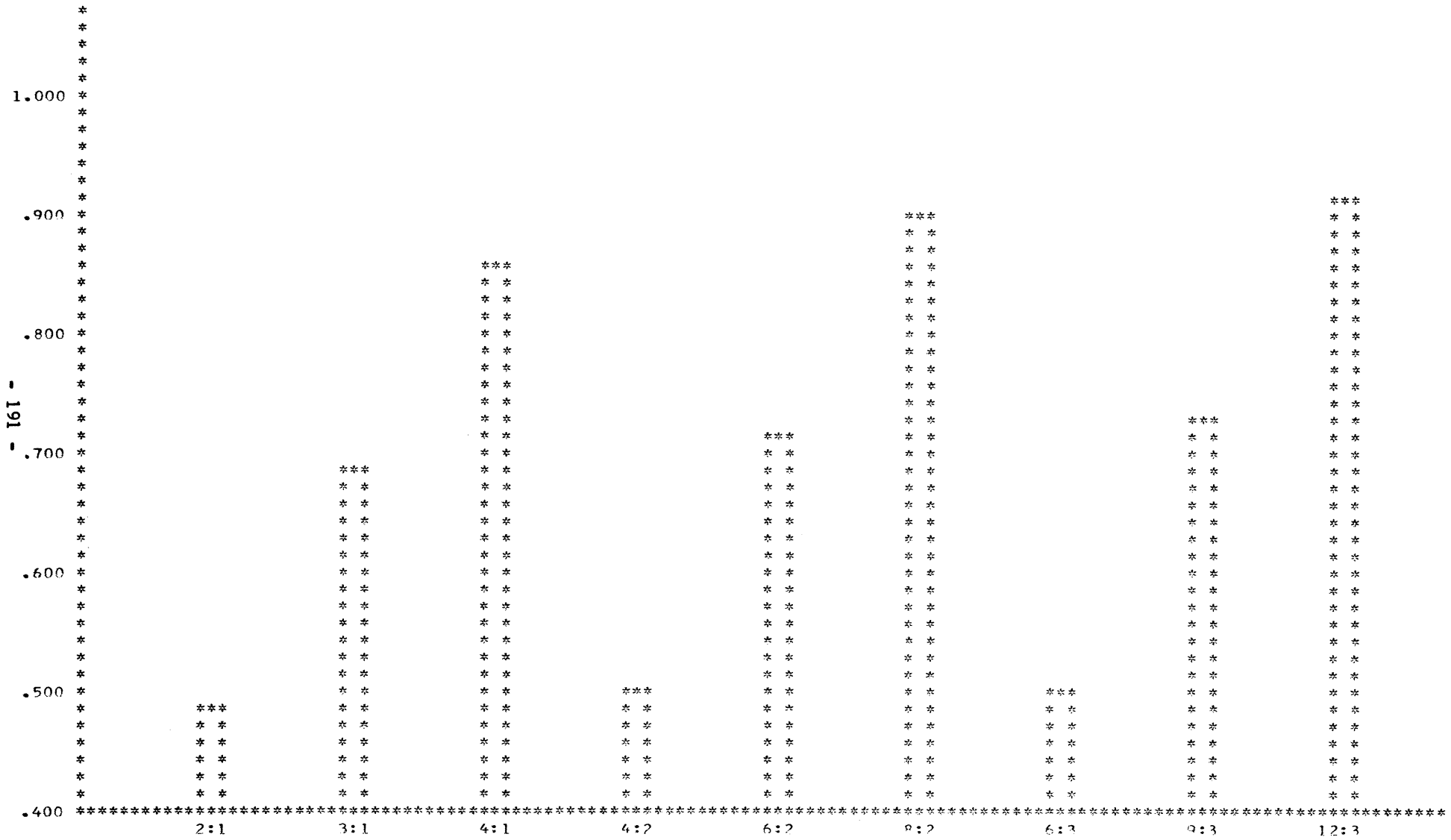


FIGURE 23-E: GRAPHICAL OUTPUT - CRANE UTILIZATION
(EQUIPMENT BALANCING PROBLEM REVISITED)

ACTUAL QUEUE STATISTICS

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	AVERAGE TIME/TRANS	*AVERAGE TIME/TRANS
1	1	.034	4.435	23.066
2	2	.132	12.054	24.109
3	3	.294	21.318	33.431
4	2	.049	3.225	14.705
5	4	.128	5.707	16.753
6	6	.545	19.225	28.144
7	3	.027	1.187	12.130
8	6	.168	4.867	15.216
9	9	.673	15.416	22.016
10	1	.026	3.415	15.470
11	2	.087	8.064	21.974
12	2	.283	20.830	25.294
13	1	.009	.611	15.500
14	3	.148	6.651	14.190
15	3	.343	12.329	15.779
16	2	.013	.593	6.523
17	3	.108	3.160	8.958
18	4	.403	9.357	12.843

NOTE: STATISTICS APPEAR IN THE SAME ORDER AS THOSE ON THE HISTOGRAMS

ACTUAL STORAGE STATISTICS (CRANE USAGE)

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	AVERAGE TIME/TRAN
1	1	.494	.494	31.929
2	1	.689	.689	31.470
3	1	.862	.862	31.578
4	2	1.009	.504	32.892
5	2	1.455	.727	32.348
6	2	1.824	.912	32.527
7	3	1.525	.508	32.727
8	3	2.221	.740	32.201
9	3	2.751	.917	31.771

NOTE: STATISTICS APPEAR IN THE SAME ORDER AS THOSE ON THE HISTOGRAMS

CPU TIME USED:

ASSEMBLY: 1.185 SECONDS

EXECUTION: 21.301 SECONDS

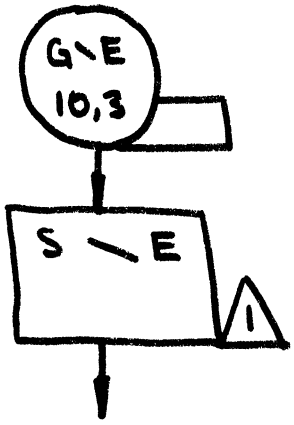
```
**** ON AT  04:15.12
**** OFF AT 04:17.53
**** ELAPSED TIME        160.973   SEC.
**** CPU TIME USED        29.046   SEC.
**** STORAGE USED       1475.24    PAGE-SEC.
**** CARDS READ            7
**** LINES PRINTED       684
**** PAGES PRINTED        23
**** CARDS PUNCHED
**** DRUM READS
**** APPROX. COST OF THIS RUN      $2.78
```

FIGURE 23-F: TABULATED OUTPUT; TIME AND COST DATA
(EQUIPMENT BALANCING PROBLEM REVISITED)

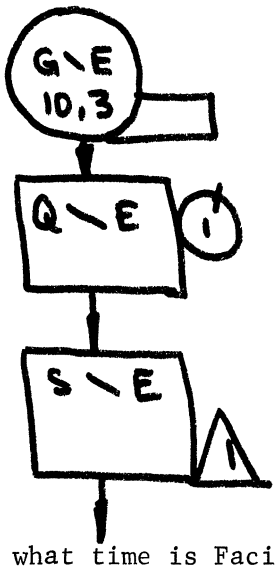
Problems for Practice

1. Discuss the difference between these two block diagram segments:

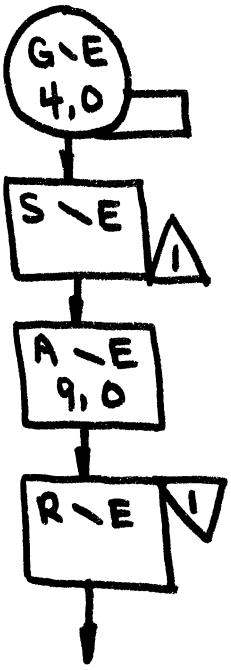
a.)



b.)



2. Consider this GPSS/360 block diagram:



a.) At what time is Facility 1 first seized by a Transaction? (What is the value of C1?)

b.) When is Facility 1 seized for the second time?

c.) When is Facility 1 seized for the third time?

d.) How much time elapses between generation of successive Transactions at the GENERATE Block?

3. In a certain GPSS/360 model, each time Facility 2 is released, a new Transaction is to enter a model within 10±4 time units. Show a block diagram segment which will provide for creation of a new Transaction at the various proper times.

4. The manufacture of a certain line of widgets requires a relatively lengthy assembly process, followed by a short firing time in an oven. Since ovens are expensive to maintain, several assemblers share a single oven, which holds only one widget at a time. An assembler cannot begin assembling a new widget until the old one has been removed from the oven. Cost and time studies have shown the following characteristics of the process:

Assembly time:	30±5 minutes, uniformly distributed
Firing time:	8±2 minutes, uniformly distributed
Raw Material:	$2 per widget
Assembler Salary:	$2 per hour

Cost of Oven: $10 per hour

Sale Price: $5 per widget

Set up a GPSS/360 block diagram model with which to investigate this process for the purpose of determining the most efficient number of assemblers to be assigned to an oven. Your intention should be to study various possibilities on a sequential basis, using Control Cards so that more than one possibility can be studied with a single batch session. Show the Control Card configuration you would use. (Note: the cost information cannot be conveniently built into the model until Arithmetic Variables have been studied in Section 52. For the time being, indicate how you would use the cost information in conjunction with the simulation results to compute the "answer" to the problem by hand.)

5. Transactions are to be introduced to a model each day of simulated time at 1 a.m., 4 a.m. 10 a.m., and 2 p.m. Under these alternative assumptions, produce a GPSS/360 block diagram segment which provides for creation of the Transactions at the proper times:

   a) Transactions are to be introduced exactly on time.

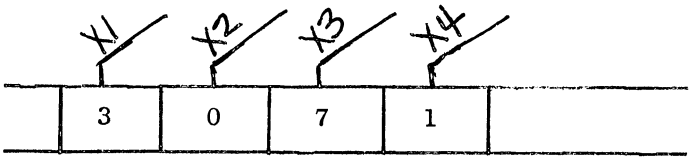
   b) Transactions are not always introduced exactly on time but are delayed 10±10 minutes, uniformly distributed.

6. In a garment factory, 400 sewing machines are operated 8 hours a day, 5 days a week. The Management is interested in the proper number of mechanics and standby machines to have available in order to minimize costs. Mechanics wages are $2.50 per hour, additional machines may be rented for $2 per day, and downtime costs $10 per hour in lost profits. Average running time between machine breakdowns is 16±5 hours, uniformly distributed. Service times are 30±15 minutes, uniformly distributed.

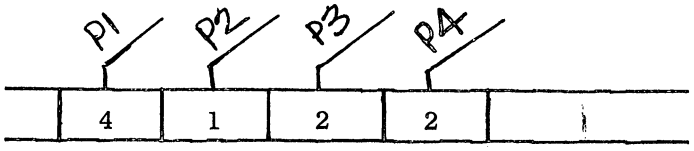
   Simulate this situation and establish the number of mechanics and standby machines to have on hand to minimize expected costs. (See the Note at the end of Problem 4.)

   (after Bowman and Fetter, Analysis for Production and Operations Management)

7. Suppose that SAVEVALUEs 1 through 4 have the values shown

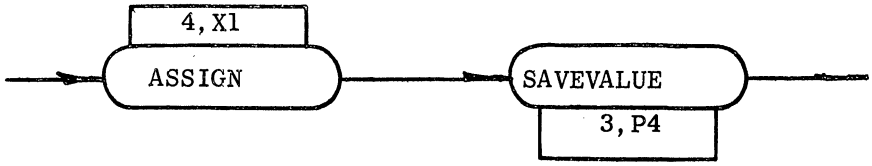


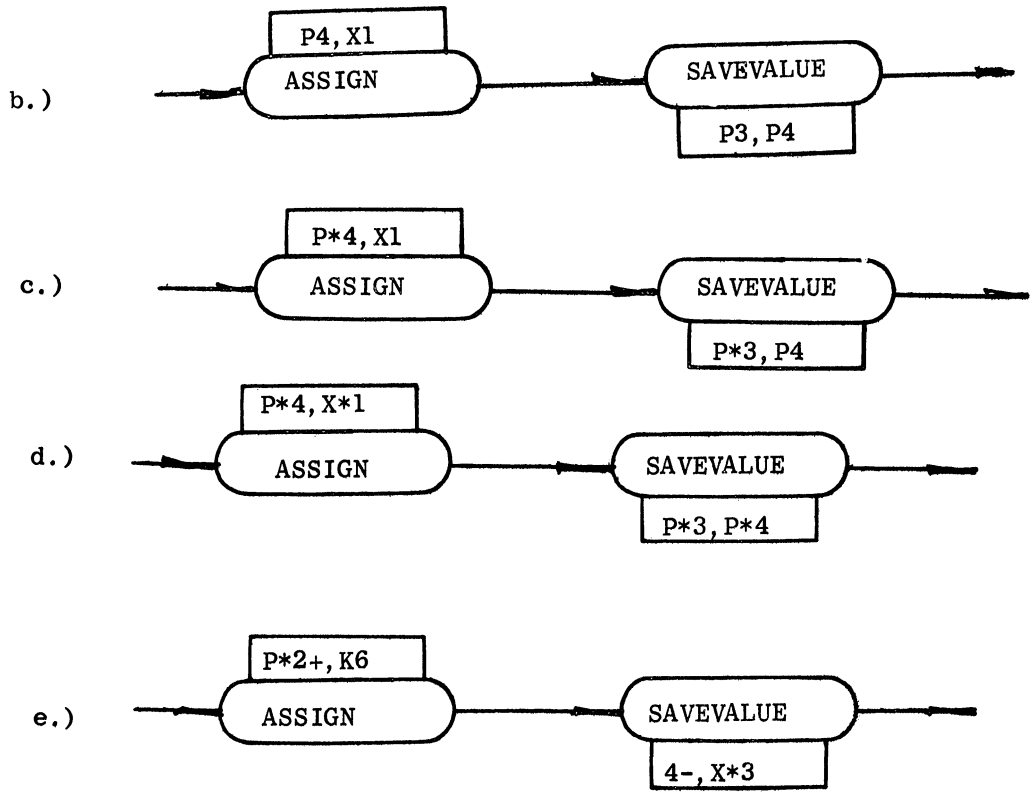
when a transaction with these parameter values



passes through any one of the block diagram segments appearing below. In each case, state which SAVEVALUE and/or Parameter values are changed, and indicate the resulting changed values.

a.)





8. A certain inventory control system uses a reorder point/reorder quantity replenishment rule. In a GPSS/360 model of the system, X2 is to contain the reorder point and X3 is to contain the reorder quantity. The system is to be simulated for all combinations of reorder point and reorder quantity between 2000 and 10,000 units, inclusive, in steps of 2000 units. When a given combination has been established, 250 simulated time units are to elapse before the next combination is made effective. Set up a GPSS/360 model segment to achieve this effect, complete with its own GENERATE and TERMINATE blocks. (Suggestion: construct a block diagram segment through which a Xact repeatedly loops, establishing the various combinations for the proper time durations in the process.)

9. Two queues in a system are numbered 3 and 5, respectively. Set up a block diagram segment in such a way that a transaction traveling through the segment will be routed to the queue of shorter length. How does your model behave when the two queues are of equal length?

10. The conventional design for checkout counters at a supermarket is for several counters to operate in parallel with a line forming at each counter. Model this situation in GPSS/360 for the case of six checkout counters, showing how transactions are routed to the appropriate Queue according to this criterion:

> The customer selects the Queue for which it is true that the sum of the Queue length and the average number of items per basket in that Queue is smaller (or possibly equal to) the analogous sum corresponding to the other Queues.

(Problem 10 continued on next page)

Assume that the number of items each shopper has in his basket is recorded as the value of P10 of the corresponding Transaction. Simply use "A,B" for the checkout time distribution.

11. A certain model is being designed to study various policies for maintaining pumps in a chemical plant. Each pump has six valves, and whenever one of these valves fails, the pump must be shut down for repair. Valve life is negative exponentially distributed, with a mean lifetime of 500 hours. Assume that a Transaction represents a pump, with Parameter values interpretable as the lifetime of various valves. Show how a loop or loops can be used in constructing a GPSS block diagram segment which will have this effect:

    a) Assign as the values of Transactions Parameters 1 through 6 the operating time before failure of valves 1 through 6, respectively, by sampling from the valve life distribution.

    b) Route the Transaction to an ADVANCE block where the time delay will be the parameter value representing the smallest valve operating time before failure.

12. Show how to use a Logic Switch to simulate a Facility. What are the advantages in doing this? The potential disadvatages? (Page 132, User's Manual)

13. a) Logic Switches 10,9,8,...,3,2,1 are to be put into Set position in that chronological order. Show a block diagram segment using the LOOP Block which accomplishes this effect.

    b) Logic Switches 1,2,3,...,8,9,10 are to be put into Set position in that chronological order. Show a block diagram segment using the LOOP Block which accomplishes this effect. Then show another segment not using the LOOP Block which accomplishes the effect.

    c) The Logic Switches whose numbers are in Savevalues (fullword) 1,2,3,...,10 are to be put into Set position. Show a block diagram segment which brings about the desired effect.

14. A certain machine uses Part A and Part B, both of which are subject to periodic failure. Whenever one of the parts fails, the machine is shut down, the failed part is removed and replaced if a spare is available, and the machine is turned on again. Failed parts are repaired and can then be used again. These data are available for the two types of parts:

	Lifetime	Repair Time*	Installation Time
Part A	235±75 hours	336±50 hours	2±1 minutes
Part B	375±95 hours	280±50 hours	2±1 hours

Assume that the above times are uniformly distributed.

Construct a GPSS block diagram model of this system to determine how the percentage running time of the machine depends on the number of Part A and Part B

---

* Actually, mean time lapse before the part is again available. Failed parts are sent to a central repair department where long delays are encountered.

parts available to the system. Show how various "Part A spares, Part B spares" combinations could be studied on the basis of a single submit by using appropriate Control Cards.

15. A particular work center in a job shop can be represented as a single-server queueing system where jobs arrive according to a Poisson process with a mean rate of 4.5 per day. The arriving jobs are of three distinct types, with the time required to perform each type of job having exponential distributions with means of 0.15, 0.20, and 0.25 working days, respectively. The practice has been to work on arriving jobs on a FIFO (first in, first out) basis. However it is important that jobs of the first type do not have to wait very long, whereas this is only moderately important for jobs of the second type, and relatively unimportant for jobs of the third type. The three types arrive with a mean rate of 1.5, 2.0, and 1.0 per day, respectively. All three types have been experiencing rather long delays on the average, and it has been proposed that the jobs be selected according to an appropriate priority discipline instead.

    Model this situation in GPSS, then compare the expected waiting time for each of the three types of jobs if the queue discipline were

    1) First come, first served

    2) According to relative job importance

    (after Hillier and Lieberman, Introduction to Operations Research)

16. Re-do Problem 5, assuming that delays have been observed to occur with these relative frequencies:

No. of Minutes Late:	0	5	10	15	20	25	30
Relative Frequency:	.3	.3	.2	.1	.05	.025	.025

17. Attendants at a gas station are paid at the rate of $15 each per 8-hour day. Cars arrive for service in Poisson fashion at an average rate of one car every 3 minutes. Each car is serviced by a single attendant, and the time per car is exponentially distributed with mean 8 minutes. When the number of cars waiting for service exceeds two, arrivals become discouraged and drive on (lost business).

    Set up a GPSS/360 model to determine the optimum number of attendants to have on duty. In particular, build the model in such a way that it will simultaneously gather pertinent statistics for the cases of 4, 5, 6, and 7 attendants.

18. Plans are currently being developed for a new factory. One department has been allocated a large number of automatic machines of a certain type, and it is now desired to determine how many machines should be assigned to each operator for servicing (loading, unloading, adjusting, setup, etc.). For the purpose of this analysis, the following information has been provided.

    The running time (time between completing service and requiring service again) of each machine is exponentially distributed with a mean of 120 minutes. The service time has an exponential distribution with a mean of 6 minutes. The net cost to the company of each operator is $3 per hour. It is estimated that the cost of an idle machine is $30 per hour.

An operator must attend to his own machines; he cannot give help to or receive help from other operators. The problem is to determine how many machines should be assigned to each operator.

Set up a GPSS model to simultaneously estimate the average cost per hour per machine for each of these four "machines per operator" possibilities: 4; 5; 6; and 7.

(Note: for an analytic solution to this problem, see Hillier and Lieberman, Introduction to Operations Research, pp. 348-351.)

19. A material used in production by a particular company can only be purchased in quantities of 2000 or any integral multiple of 12000. Ordering costs (marginal) are estimated at $4 per order, and the annual charge for storage and carrying costs is 30% of the average value of inventory. Cost of the material is $1 per unit.

Experience with delivery times has been:

% of Total Experience:	5	10	20	40	15	5	3	2
Delivery Time, Days:	3	4	5	6	7	8	9	10

Demand per day is a variable according to this distribution:

Demand:	0	400	800	1200	1600	2000	2400	2800	3200
Days/Year:		50	40	25	15	10	5	3	2

If this item is unavailable at any time, the company estimates the probable loss to be $100 per day. Back orders are always placed, however, and are accepted by the company.

The reorder point - reorder quantity approach is to be used as the replenishment rule. Set up a GPSS model to estimate the average daily cost resulting from various reorder point - reorder quantity combinations that might be used. The model should be set up in terms of two segments: a Demand Segment and an Inventory Segment.

(after Bowman and Fetter: Analysis for Production and Operations Management)

20. Re-do Problem 4, building your model to include these features:

    a) Study various Assembler:Oven ratios on a simultaneous basis and

    b) Design the model so that it will consist of two segments. Segment 1 will collect only those statistics necessary for the investigation; Segment 2 will utilize the Segment 1 statistics to carry out pertinent computations and print out values of interest. All unnecessary printing is to be suppressed.

21. Re-do Problem 6, building your model to include features corresponding to those described in Problem 20, Parts a) and b).

22. Show a block diagram segment delaying Transactions in Queue 7 until Facility 5 is not in use, Storage 3 is not full, and Logic Switch 2 is Reset.

23. Re-do Problem 15, permitting one level of pre-empting to occur according to relative job importance.

24. A modeler wants to gather statistics showing what percentage of the time Queue 3 is exactly of length zero, exactly of length one, exactly of length two,...,exactly of length five, or of length six or greater. Prepare a GPSS block diagram (or block diagram segment) showing how the desired statistics can be obtained.

    Note: There are at least two random variables associated with queues:

    1. The random variable which takes on values equal to the length of the waiting line, and

    2. The random variable which takes on values equal to the time spent by various transactions in the waiting line.

    It is sometimes of interest to gather information about the probability distribution followed by one or the other of these random variables. Note that the standard Queue printout (if not suppressed) indicates only the expected value (actually, the experimental approximation to the expected value) of the two probability distributions at hand.

    The QTABLE card can be used to gather statistics on the time distribution.

    In the above problem, you are simply asked how to gather statistics on the distribution of Queue length.

25. At a particular service station in a model, a server has been instructed to pick, as the next customer, the customer who has been in the system the greatest length of time. Set up a GPSS/360 block diagram segment according to which the Transaction which SEIZEs a Facility will be the Transaction having spent the longest time in the system. (One approach involves use of the LINK/UNLINK Blocks, although their use is not strictly necessary for modeling this process.)

26. Develop a GPSS/360 block diagram segment showing how Transactions which are eventually to be serviced in Storage 5 can be put on a User Chain and then can be sent to ENTER Storage 5, one by one, as it becomes capable of receiving new Transactions. Assume that each Transaction will use either one or two units of the Storage's capacity, as recorded in Transaction Parameter 1. Queue discipline to be used is first in, first out, except that whenever only one unit of Storage capacity is available and the Transaction at the front of the User Chain requires two units, the waiting line will be polled on a front-to-back basis to find the first Transaction which needs but one capacity unit, and it will be permitted to ENTER the Storage immediately.

27. (Barber Shop Problem Extended to Non-Uniform Service Rates) A three-chair barber shop is manned by exponential barbers who provide service at these rates:

	Mean Service Rate, Minutes
Barber 1:	23
Barber 2:	18
Barber 3:	15

Customers arrive according to a Poisson Pattern with a mean arrival rate of nine customers per hour. Set up a GPSS block diagram model of the system.

28. Develop a GPSS/360 block diagram segment showing how transactions which are eventually to be serviced in Facility 5 can be put on a User Chain and then can be sent to SEIZE Facility 5, one by one, as it becomes available. This queue discipline is to be used by the server in selecting the next customer:

    If the queue is of an odd length, the customer in the middle of the queue is to be selected.

    If the queue is of an even length, the customer at the back of the queue is to be selected.

29. Show a block diagram segment in which Transactions will contribute to the contents of Queue 25 until Facility 1 is available or Storage 60 is not full. (See the User's Manual, page 86.)

30. In a barber shop, 33% of the customers consider getting a shoeshine after their haircut is finished. Of the 33%, however, 20% proceed without a shoeshine if the shoeshine boy is busy with someone else. Prepare a block diagram segment corresponding to this situation.

31. One Transaction from an Assembly Set will eventually find its way to BLOKA. Another Transaction from the same Assembly Set will eventually find its way to either BLOKB or BLOKC. Only after the two Transactions have reached their respective Blocks are they to move on in the model.

    a) Show a block diagram segment for this situation, assuming the first Transaction reaches BLOKA before the second reaches either BLOKB or BLOKC.

    b) Re-do the segment without assuming that either Transaction will arrive at its respective Block before the other. (See the User's Manual, page 96.)

32. Show how to tabulate the Transit Time of Transactions between BLOKA and BLOKB, inclusive, in a model.

33. Re-do Problem 14, making use of the PREEMPT and RETURN Blocks.

34. A greeting card manufacturer is faced by the problem of how many Christmas cards he should stock to meet his demand. The cards are all produced before orders start to arrive. During the time orders for Christmas cards are arriving, the capacity of the plant is completely occupied by other lines. Each card costs 5¢ to produce and is sold for 7¢ in wholesale lots. Cards not sold must be carried in inventory until the next year at an average cost of 1.3¢ per card. The manufacturer does have the option of printing up sheets of cards but not cutting or folding them. The annual cost of storing a card in sheet form is 0.5¢. However, only 60% of the dealers will wait for the cards to be made up if they cannot be supplied immediately; the rest cancel their orders. Previous experience indicates that any number of sales between one and two million is equally likely.

    Simulate this situation and determine how many complete cards should be stocked and how many should be held in sheet form.

Suggest ways to expand the details of this problem to make it more realistic. Avoid including details to which the problem outcome is not sensitive.

(after Bowman and Fetter, Analysis for Production and Operations Management)

35. A steel company, which operates its own fleet of ships to import iron ore, is considering the construction of port facilities to support a new plant. Both the number of unloading berths and the type of installation in each berth must be decided in such a way as to minimize total unloading costs. A maximum of three berths can be built; and it is required that the same type of unloading installation be in each berth that is built. The choice of unloading installation lies among types A, B, and C, for which this information is available:

Installation	Fixed Cost[1]	Operating Cost[1]	Capacity[1]
A	$ 840	$ 840	3600 tons
B	1350	1350	5800 tons
C	1500	1600	6400 tons

[1] per day per berth

Fixed costs include such items as the amortization of the original cost of the installation over its expected life, general maintenance, etc.; they apply to all days, whether the equipment is used or not. Operating costs are incurred only during the time intervals that the unloading equipment is actually in use.

Ships to be unloaded each carry 8000 tons of ore, and are considered to arrive in a Poisson fashion throughout the year with a mean arrival rate of five ships a week. Service times for a given type of installation are considered to be exponential, with mean service rate corresponding to the average capacity column in the table. Time spent in the unloading system (waiting time plus unloading time) is considered to cost the company $2000 per ship per day. The system will operate 7 days per week.

Set up a single GPSS/360 model to simulate the operation of all feasible system configurations (i.e. berth type and number of berths) simultaneously. On this basis, it will not be necessary to use RESET and CLEAR cards, except for the possibility of using a RESET card after the various system configurations have reached steady state.

(after Sasiene, Yaspan, and Friedman: Operations Research Methods and Problems)

36. Shoppers enter a supermarket every 36 ± 18 time units. If no shopping basket is available, they leave without shopping. If one of the 150 baskets is available, they spend 440 ± 220 or 2640 ± 1820 time units shopping, depending on whether they are express or regular shoppers, respectively. Twenty five percent of the shoppers are express shoppers. When shopping is complete, the express shoppers are checked out via one express checkout counter and the regular shoppers are checked out via any one of seven regular checkout counters. Checkout time at the express counter is 360 ± 180 seconds. Shoppers leave the store immediately after checking out.

Produce a block diagram model of this system, then implement the model using GPSS/360. The model should be designed to gather these statistics:

1. Percentage of the shoppers to whom no shopping basket is available.

2. Maximum length and average length of queues forming at the checkout counters. (Assume that only one queue forms for the regular checkout counters, considered as a storage.)

3. Transit time of express shoppers.

4. Transit time of regular shoppers.

5. Distribution of queue length at the checkout counters. The choice of interval width is yours to make.

6. Rate of arrival at the regular checkout counter queue. The time span chosen for the rate of arrival measurement should be 36 time units.

7. Time between arrivals at the express checkout counter queue.

Simulate the system until 250 transactions have been terminated and obtain a printout of the statistics requested. Then reset the model (using the RESET control card) and continue the simulation until another 250 transactions have been terminated, again obtaining a printout of the statistics. Reset and repeat one more time. Does each pertinent statistic appear to be approaching a stable value?

37. Esto Oil Company is considering bidding on a contract to supply fuel oil to a port in the United Kingdom. Esto's plan is to tap its production facilities in Africa and transport the oil in its own tankers. The fuel is to be delivered to the customer's tank facilities in port at the United Kingdom. The capacity of these facilities is 1,250 units of oil and the customer's invitation to bid specifies that the supply must not be less than 250 units more than 5 percent of the time.

Demand on the tank facilities in the U.K. is uniformly distributed over the interval from 65 to 135 units, inclusive.

Esto management wants to know how many of its standard 625 unit tankers it must employ on a full-time basis to maintain the required service level. Management also wants to know by what amount the demand in the U.K. could increase (assuming uniformly distributed demand over a 71 unit interval) before the customer's indicated "5% specification" could no longer be met without adding another ship to the run.

System constraints are:

1. Berth Facilities: There is one berth in the U.K. and two berths in Africa. 24 ± 6 hours are required to load a ship berthed in Africa, and 24 ± 2 hours are required to unload a ship berthed in the U.K.

2. Travel Time: 8 ± two days are required for a ship to make the trip from Africa to Britain and 7 ± 2 days are required for the return trip.

3. Weather: A ship cannot enter or leave a berth during a storm. Storms are seasonal with this pattern:

	Probability of a Storm	
	April 1 to Sept. 30	Rest of Year
Africa	0.2	0.1
U.K.	0.1	0.2

If a storm is in progress when a ship arrives at the U.K., its remaining duration can be approximated to be 4 + 3 hours. If a storm is in progress when a ship arrives at the African facility, its remaining duration is approximately 5 + 2 hours.

Esto's production facilities in Africa do not pose a constraint.

Esto's plan is that ships which cannot unload in the U.K. because the customer's tank facilities are loaded will anchor in the harbor and wait, constituting a floating inventory.

Construct and run a GPSS/360 model for simulating Esto's situation for the purpose of providing management with the requested information.

(after McMillan and Gonazlez: Systems Analysis A Computer Approach to Decision Models)

38. A chemical company has a series of high-pressure injection valves operating under similar conditions and wishes to determine a proper maintenance policy. The pump valves are subject to failure, and their routine maintenance costs a certain number of man-hours per year. Each pump has three intake valves and three exhaust valves. When a valve fails, it is necessary to shut down the pump and prepare it for maintenance. Each set of valves is covered by a manifold which must be removed after shutdown in order to expose either the three intake valves or the three exhaust valves. There is no down-time cost as the firm has stand-by pumps to be used during maintenance on the valves.

The company is interested in and wants to evaluate four maintenance procedures which it considers practical:

1. Repair a valve only when it fails.
2. Repair all three exhaust valves if one exhaust valve fails, or all three intake valves if one intake valve fails.
3. Repair all six valves (three exhaust and three intake valves) whenever a pump must be shut down to repair one valve.
4. Repair the valve that fails plus all valves which have been in use more than the estimated average service life (560 hours).

These data have been supplied by the company for the analysis:

Maintenance Cost, Expressed in Mechanic's Time

Operation	Time, Hours
Shut down, prepare for maintenance	1/2
Remove manifold (either intake or exhaust)	2/3
Disassemble one valve	1/3
Overhaul one valve	5/4
Assemble one valve	1/3
Replace manifold (either intake or exhaust)	2/3

The cumulative probability distribution for valve service life, shown in the following table, was constructed from empirical data supplied by the company:

Valve Age at Failure Hours	Cumulative Probability Per Cent
100	5
300	22
500	45
700	65
900	83
1100	91
1300	97

Set up appropriate GPSS/360 models to simulate each of the four maintenance procedures described above. Run the models to determine which maintenance procedure corresponds to minimum mechanic time requirement per pump.

(after Bowman and Fetter: Analysis for Production Management)

39. Companies which own, service and operate automatic coin-operated vending machines (soft drinks, coffee, ice cream, etc.) have the problem of determining how often to service their machines. This servicing operation includes replenishing the supply of the commodity sold, gathering the coins from the change box, and performing any minor maintenance which may be necessary. Service which is scheduled too often results in needlessly high servicing cost, and service scheduled too infrequently results in lost customers because of lack of change, lack of commodity in the machine, or machine breakdowns.

As a specific problem, suppose that a particular building contains ten vending machines which dispense coke in paper cups. The building is visited periodically by a service man who services all 10 machines. A service visit comprises these activities:

1) Refill machine with coke syrup and soda
2) Remove coins from change box
3) Make repairs (if necessary)
4) Put nickels and dimes in the machine to be used as change for quarters which might be deposited

Each machine can hold 750 cokes and up to 200 pre-loaded dimes and 200 pre-loaded nickels. Assume that each machine will accept any number of quarters, nickels, and dimes during usage. A quarter will not be accepted if change cannot be made for it; and all nickels and dimes in the machine are available for the change-making process. Customer interarrival times at the various machines, which are located throughout the building, are: (Poisson arrival process)

Machine Number:	1	2	3	4	5	6	7	8	9	10
Interarrival Time, Mins.:	5	5	5	7	7	8	9	10	12	15

Customers have either 2 nickels, a dime, or a quarter to put in the machine. The distribution, independent of machine, is:

Coins Available	Relative Frequency
Quarter	.45
Dime	.40
Nickels	.15

All machine failures are assumed to be minor enough so that they can be repaired in a short time and on-the-spot by the service man. Failures are negative exponentially distributed with a mean operating time between failures of 100 hours. Assume that zero time is required to service each machine. The cost of one lost customer is estimated to be 5¢, and the cost of one service visit is $30.

Model this situation in GPSS, then determine how frequently the machines should be serviced to minimize the costs mentioned above. Also use the model to determine how small the values of nickels and dimes (n and d) can be before their influence on cost of operation becomes a factor (the cost of having pre-loaded nickels and dimes in the machine is deliberately neglected in the problem). How would you make this problem more realistic? (Problem suggested by R. M. Denise, W '67.)