10.11 General Purpose Simulation System (GPSS)

Introduction : The General Purpose Simulation System (GPSS) is a process-oriented language designed for discrete event modeling. It was originally designed by Geoffrey Gordon in 1961 at IBM Corporation. It is a very dynamic language as the new versions keep being introduced every three to four years. The language as it exists today is the result of more than 45 years of evolution. It is now a multi-vendor language and a large number of its versions are available. By 1972, there were more than 10 versions of GPSS available in the market. GPSS started with the advent of mainframe computers, has continued its developments to meet the developing computer configurations and computing environments. GPSS has become a very popular simulation language firstly because of IBM's strong influence on the computer industry, and secondly because of its power of expression and conciseness. A short easily understood GPSS model would require many pages of FORTRAN programming to accomplish the same task.

Since, the GPSS language has in-built mechanisms to collect statistics, analyze them, produce tabulated outputs, and execute a number of mundane tasks, the GPSS user can devote more time and attention to the important issues involved in the modeling.

GPSS/H developed by James Henriksen, in 1977, is a compiled language with the interpretive approach of GPSS V and is reported to be five times faster in execution time. Compared to GPSS V, GPSS/H has many improvements like:

- A real value clock for recording the simulation time.
- Ability to read and write external files.
- Enhanced control statements.
- Increased number of mathematical functions.
- Capability of generating unlimited number of independent random number series.
- Capability of generating random variants form different probability distributions.

Common modern versions of GPSS are GPSS/H, GPSS V, GPSS/PC, GPSSR/DC and GPSS/VX. It is possible to add animation to the results of simulation. A number of functions performed by GPSS can be found embedded in other language such as GPSS-FORTRAN, APL-FORTRAN and PL11-GPSS.

10.12 GPSS Output Information

GPSS is a discrete event modeling language and allows the user to study the structure of the system, to follow the flow of entities through the system, to measure the effect of starving or blocking the time-shared components of the system, to ascertain the utilization of the elements in the system and to evaluate the performance of the system. The output of the GPSS program furnishes information on following aspects.

1. The flow of entities through the system as a whole or through its various paths.
2. The maximum and average queue lengths building at various locations in the system.
3. The distribution with mean and spread of waiting times at various locations in the system.
4. The distribution with mean and spread of the time taken by the entities to pass through the system as well as its various paths.
5. The capacity utilization of various elements of the system.
6. The distribution of the occupancy of the storage elements provided in the system.
7. The statistical variations introduced in the system as well as the information about the state of the system at various points in time.

10.13 GPSS Block Diagrams

The development of a simulation model in GPSS is a block-by-block construction. A set of standard block is arranged in the form of a block diagram that represents the flow of entities through the various paths of the system. Each block represents a step in the action of the system and links, joining the blocks, represent the sequence of events that can occur.

To build a block diagram, it is essential to have a complete description of the system. The meanings of the blocks used in the system must be clearly defined. Each block must be assigned the *block time*, *i.e.*, the time which the execution of the block will take.

In total, a set of 25 specific block types have been designed, which can be used in the construction of a block diagram. Each block type can be used any number of times in a block diagram, but the total number of blocks should not exceed 2047. On the completion of the block diagram, each block is assigned a number, between 1 and 2047, called the *block number*.

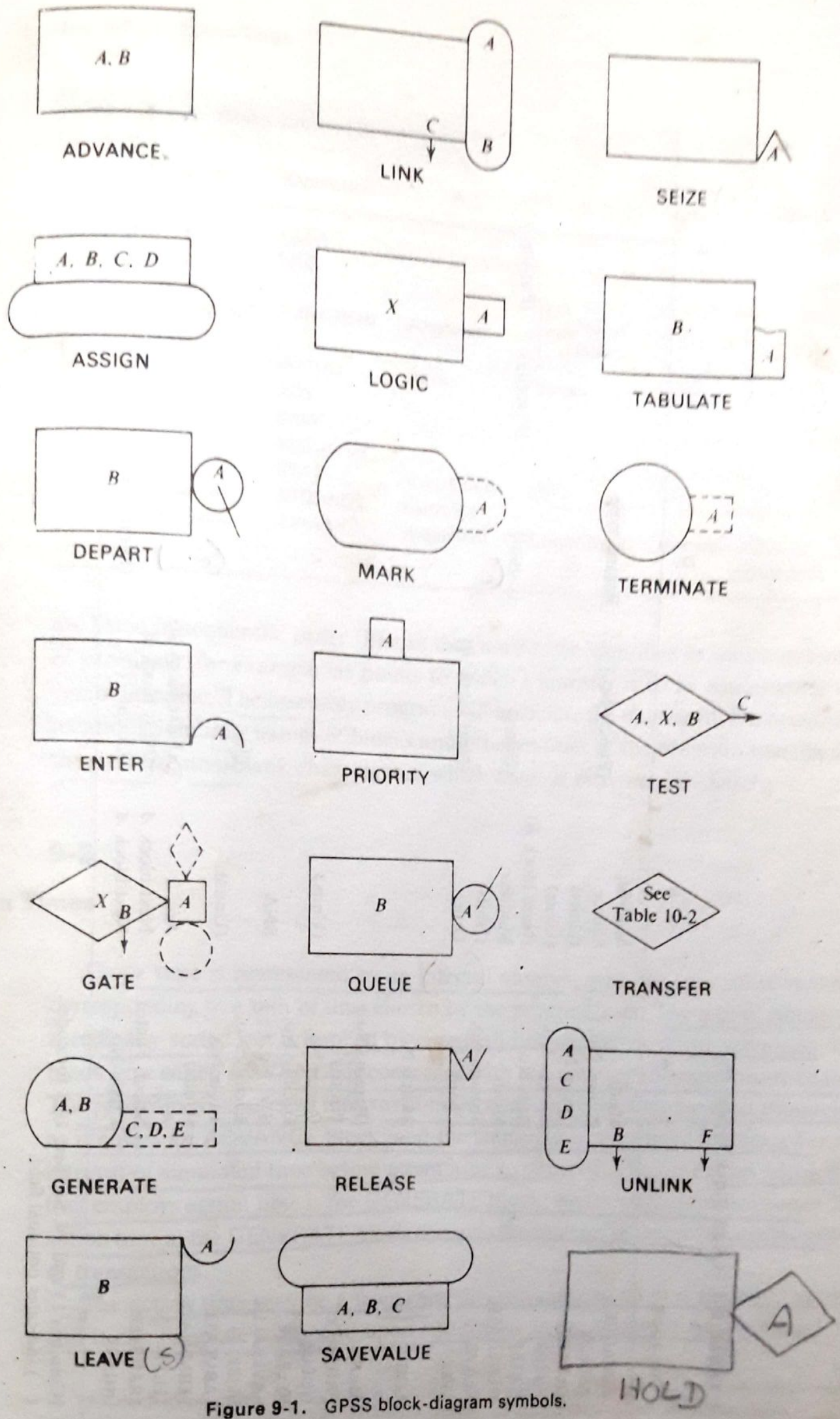


Figure 9-1. GPSS block-diagram symbols.

eneral Description

The system to be simulated in GPSS is described as a block diagram in which the blocks represent the activities, and lines joining the blocks indicate the sequence in which the activities can be executed. Where there is a choice of activities, more than one line leaves a block and the condition for the choice is stated at the block.

The use of block diagrams to describe systems is, of course, very familiar. However, the form taken by a block diagram description usually depends upon the person drawing the block diagram. To base a programming language on this descriptive method, each block must be given a precise meaning. The approach taken in GPSS is to define a set of 48 specific *block types*, each of which represents a characteristic action of systems. The program user must draw a block diagram of the system using only these block types.

Each block type is given a name that is descriptive of the block action and is represented by a particular symbol. Figure 9-1 shows the symbols used for the block types that will be described in this and the next chapter. To assist the reader, the block diagrams drawn in this chapter will name the block types, although this is not usually done by a programmer familiar with GPSS. Coding instructions for all the described block types are similarly brought together in Table 9-1. Table 9-2 describes the control statements. Each block type has a number of data fields. As the blocks are described, the fields will be referred to as field A, B, C, and so on, reflecting the order in which they are specified.

Moving through the system being simulated are entities that depend upon the nature of the system. For example, a communication system is concerned with the movement of messages, a road transportation system with motor vehicles, a data processing system with records, and so on. In the simulation, these entities are called *transactions*. The sequence of events in real time is reflected in the movement of transactions from block to block in simulated time.

Transactions are created at one or more GENERATE blocks and are removed from the simulation at TERMINATE blocks. There can be many transactions simultaneously moving through the block diagram. Each transaction is always positioned at a block and most blocks can hold many transactions simultaneously. The transfer of a transaction from one block to another occurs instantaneously at a specific time or when some change of system condition occurs.

A GPSS block diagram can consist of many blocks up to some limit prescribed by the program (usually set to 1,000). An identification number called a *location* is given to each block, and the movement of transactions is usually from one block to the block with the next highest location. The locations are assigned automatically by an assembly program within GPSS so that, when a problem is coded, the blocks

...called the *block number*.

10.14 Characteristics of the Blocks

10.14.1 *Block Time*

The block time also called action time is an integer giving the number of units required to execute the action represented by the block. The unit of time is decided by the programmer and is not entered in the block. But all block times throughout the block diagram are in same units of time. The block time is often a random variable, and is specified by the numbers giving *mean* and *spread* about the mean. In case the block time is constant, the spread is zero. In a simple rectangular distributions of time, the value of the block time will lie between the range, mean ± spread, with equal probability given to each integer in the range.

In many situations, the block time is not defined by the rectangular distribution. It may be some other probability density function like exponential, Poisson or normal etc. In such a case a number specifying the distribution is also inserted in the block.

10.14.2 *Alternative Actions*

At many points in a system, alternative actions are possible. More than one line may be leading into a block and more than one line may be leaving out of the block. The convention regarding alternative actions is that except the BRANCH block, not more than two lines may lead from the block and any number of lines may lead into the block. The BRANCH block type' allows upto 127 alternative paths.

10.14.3 *Selection Factor*

All block types other than the terminating block, can have two exits, referred to as *exit 1* and *exit 2*, which lead to exit blocks 1 and 2. The choice of which exit is to be followed is given by a number called the selection factor, S $(0 < S < 1)$ which is entered in the block. S is the probability of selecting exit 2 and $1 - S$ is the probability of selecting exit 1.

The choice of exit may also depend upon the availability of the blocks at exit 1 or exit 2, at the point of time when decision is to be made. The convention is that if the exit block 1 is available, then exit 1 is chosen, otherwise exit 2 is chosen. This mode of selection is indicated by setting selection factor, $S = 1$.

10.14.4 *Transactions*

In each system represented by a block diagram, some entities pass through the system. In a petrol pump system they may be vehicles, in a production system they may be work pieces and in supermarket they may be customers etc. Those entities are referred to as Transactions. In a simulation, the transactions are created, which move through the block diagram in the same way, as the entities pass through the actual system being simulated.

10.14.5 *Items of Equipment*

In each system worth the name, there are physical equipments, which perform some operations on the transactions. Machine tools in a production shop perform machining operations on work pieces (transactions). A toll booth on a road is an equipment, when vehicles are transactions. A worker at an assembly line workstation is an item of equipment, while the components being assembled are transactions. The items of equipment may operate upon transactions individually or may handle groups. The item of equipment generally has a limited capacity and can cause congestion or waiting lines of transactions.

10.14.6 *Stores and Facilities*

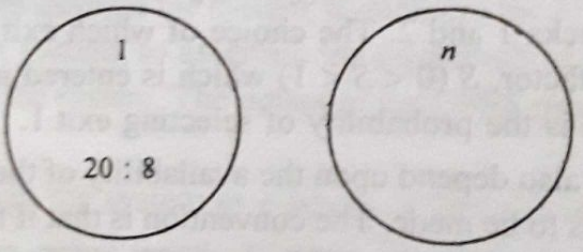
Items of equipment can be *classified* into *stores* and *facilities*, depending upon the capacity for handling the transactions. An item of equipment that can handle only one transaction at a time is called a *facility* while the items of equipment, which can handle a large number of transaction at the same time is called a *store*. There is always a defined capacity of a store. In a production line, a workstation handling one work piece at a time is a facility, while the transfer line between the workstations, is a store. In a library the book issue counter is a facility, while the books stacking area is a store. In a traffic system a toll booth is a facility, while road is a store. In a block diagram, there can be 511 facilities and 511 stores.

10.15 Block Types

The detailed description of all the block types is beyond the scope of this book. Here, only a brief description of some block types is given, to help understand the concept of GPSS block diagram.

ORIGINATE: The ORIGINATE block type is concerned with the creations of transactions and feeding them into the simulation. Generation of arrival of customers at a general store, generation of passengers arrival at a railway ticket window, generation of inter-arrival times of vehicles entering a service station, are the exemption of creating transactions. Transactions are created according to the given block time. Fig. 10.2 (*a*) shows an ORIGINATE block numbered 1 with block time of 20 ± 8 seconds. The time interval between transactions will be an integer between 12 and 28, with all integers having equal probability.

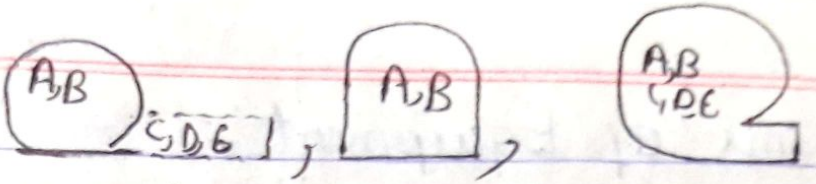
TERMINATE: The TERMINATE block type is concerned with the removal of the transaction from the simulation or the destruction of the transactions. At this block the entity represented by the transaction moves out from the system. TERMINATE block has no block time and only the number of the block is given as in Fig. 10.2 (*b*).



(*a*) ORIGINATE (*b*) TERMINATE

Block Types:

① Generate :



→ used for creation of transactions.

→ employs action time.

→ the transaction are generated accoureding to the given block time (mean I spread).

GENERATE A, B, C, D, E.

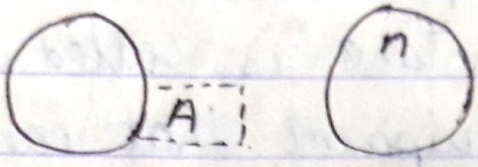
A & B represent the mean & spread of the time.

C is the time at which first transaction is to enter the system.

D is the max^{m} no. of transaction to be created.

E is the priority.

(2) TERMINATE:

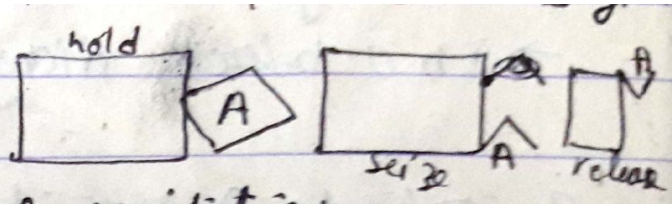


⇒ is concerned with the removal of the transactions.

→ the entity represented by the transaction moves out from the system.

→ no block time and only the number of the block is giv

③ HOLD, SEIZE & RELEASE:



→ are concerned with the use of facilities.

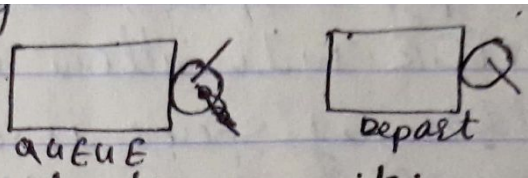
→ HOLD block allows a transaction which enter the block to engage the facility for the prescribed action time.

→ SEIZE block allows a transaction to engage a

a facility if it is available.

→ The RELEASE block allows the transaction to disengage the facility.

(4) QUEUE & DEPART:

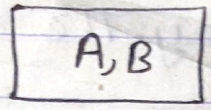


→ Queue block is placed at the position where congestion (over crowding) or in flow of transaction is expected. Used in waiting lines.

→ Queue block collect statistics like the max^m queue length, average queue length.

→ DEPART block is used in connection with QUEUE block.

→ The depart block enable the transfer to leave the queue block.

5 ADVANCE: 

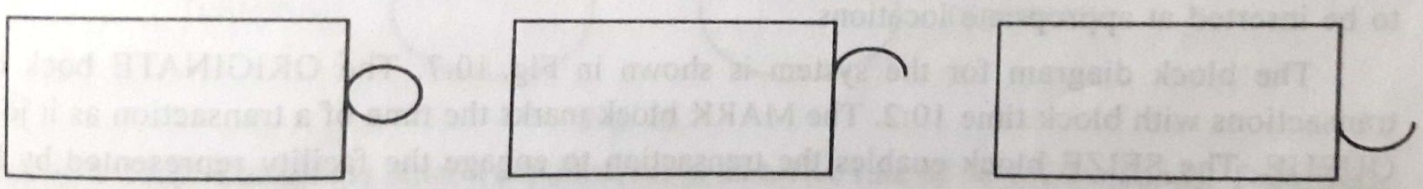
→ it is concerned with the flow of transactions.
→ it represent any action required time
→ provided with zero block time.

ADVANCE: The ADVANCE block is concerned with the flow of transactions. It represents any action requiring time but does not involve any equipment, and hence cannot cause any congestion or waiting line. The ADVANCE block, provided with a zero block time, is used as a buffer at the exit of a block using some equipment. The symbol of ADVANCE block is shown in Fig. 10.3 (*a*).

Fig. 10.4

STORE, ENTER and LEAVE: These three blocks are associated with stores. The symbols for these blocks are given in Fig. 10.5. The flag on the side of each block is for indicating the number of store. The STORE block allows a transaction to occupy space in the store. A transaction can remain in STORE for as much time as associated with the block.

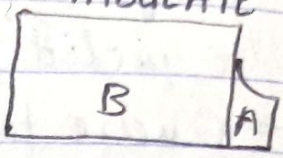
Like SEIZE and RELEASE blocks allow the transaction to engage and disengage a facility, the ENTER and LEAVE blocks allow a transaction to occupy a space or vacate a space in a store. ENTER will allow a transaction to enter the store, if space is available. The LEAVE block will remove a transaction from the store. While SEIZE and RELEASE are associated with the same transaction, the ENTER and LEAVE may be assigned with different transaction.



(*a*) STORE (*b*) ENTER (*c*) LEAVE

from the store.

7 MARK & TABULATE:



→ These blocks are used to measure the time spent by a transaction in the system or part of the system.

→ These block are inserted at a points betⁿ which the time spent by the transaction is to be measured.

→ The program make the note of the current clock time, when transaction enters MARK block and again when it arrives the TABULATE block.

→ The difference betⁿ the two times is the time spent by the transaction in the system & is entered in the table

TRANSFER: The TRANSFER block is used when the transactions are moved in a non-sequential manner. This is like the GO TO statement used in FORTRAN and other programming languages. Though there are several forms of TRANSFER block, two most commonly employed are described below:

(*a*) **The unconditional TRANSFER block**

The form of this block is TRANSFER, (Label). The label is the label of the block to which the transaction is to be transferred.

TRANSFER, NEXT

TRANSFER, DOWN

TRANSFER, UP

Are some of the examples.

(*b*) **The conditional TRANSFER block**

The form of this block is TRANSFER, xyz, block 1, block 2

xyz is a maximum three digit decimal number like .005, .25, .5, which gives the fraction of time, the transaction is to be transferred to block 2. The remaining time transfer will be to block 1.

TRANSFER .25, NEXT 1, NEXT 2 will transfer the transaction 25% of the time to block with label NEXT 2 and remaining 75% of the time to block with label NEXT 1. The symbol for the TRANSFER block is shown in Fig. 10.10.

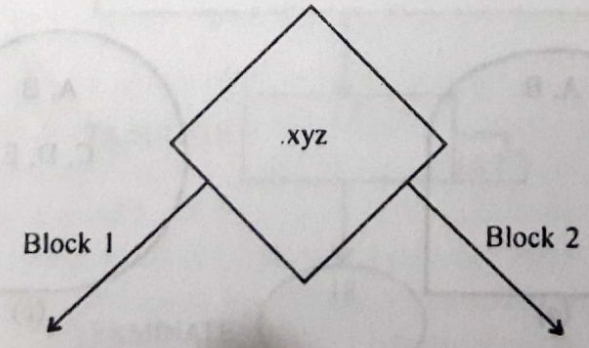


Fig. 10.10: TRANSFER Block

10.16 SIMULATE, START and END Statements

Every GPSS program will have these statements.

SIMULATE statement is generally the first statement of the program, but it is not necessary to be always the first. The general form of SIMULATE statement is

SIMULATE *n*

The operand *n* is optional. It is used to limit the running time of the program. SIMULATE 5 means that the program can run upto a maximum of 5 minutes. SIMULATE 5s means that maximum run time is 5 seconds.

START: Every GPSS program must have a START statement, the form of which is START *n*, where *n* must be a non-zero integer, which gives the maximum number of runs. It is like a counter, the value of which goes on depreciating with each completed run.

START 5 will limit the number of runs to 5.

THE END statement is always the last statement in a program.

10.17 Merits of GPSS

GPSS is the most widely used simulation language, for the following reasons:

1. GPSS is written in machine language and is very fast from the execution point of view. The simulation which may requires hours of CPU time in FORTRAN, can be executed in minute in GPSS.
2. GPSS being block-based is very fast in developing the simulation models. A model which may requires months of writing program in FORTRAN or other general-purpose languages, can be written in days in GPSS, once it is rightly understood.
3. The programs written in GPSS can be very easily modified, while it requires a lot of effort and time when modifying the program written in a general-purpose languages.
4. GPSS is much more user friendly as compared to the general-purpose languages.
5. GPSS is a multi-vendor language and is being continually updated.
6. It is widely available.
7. GPSS allows animation in its models.
8. The programs written in GPSS are small, comprise of much less computer code lines as compared to general-purpose languages.

Simulation of a Manufacturing Shop

To illustrate the features of the program described so far, consider the following simple example. A machine tool in a manufacturing shop is turning out parts at the rate of one every 5 minutes. As they are finished, the parts go to an inspector, who takes 4 ± 3 minutes to examine each one and rejects about 10% of the parts. Each part will be represented by one transaction, and the time unit selected for the problem will be 1 minute.

A block diagram representing the system is shown in Fig. 9-2. The usual

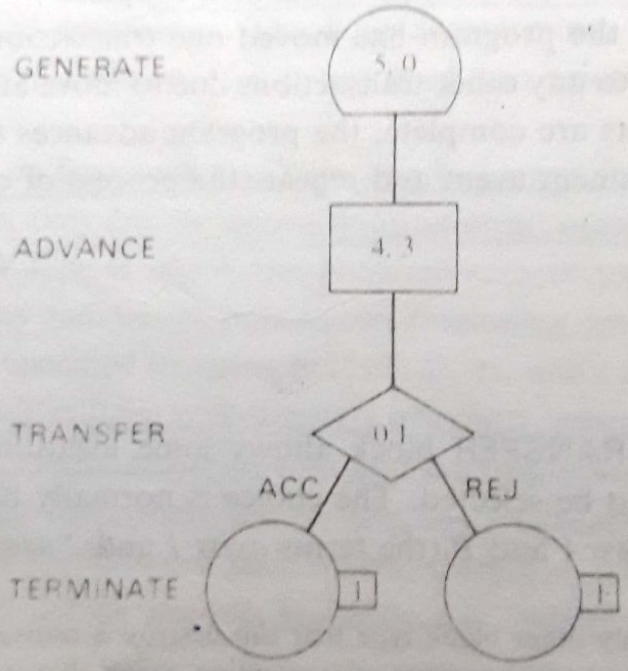
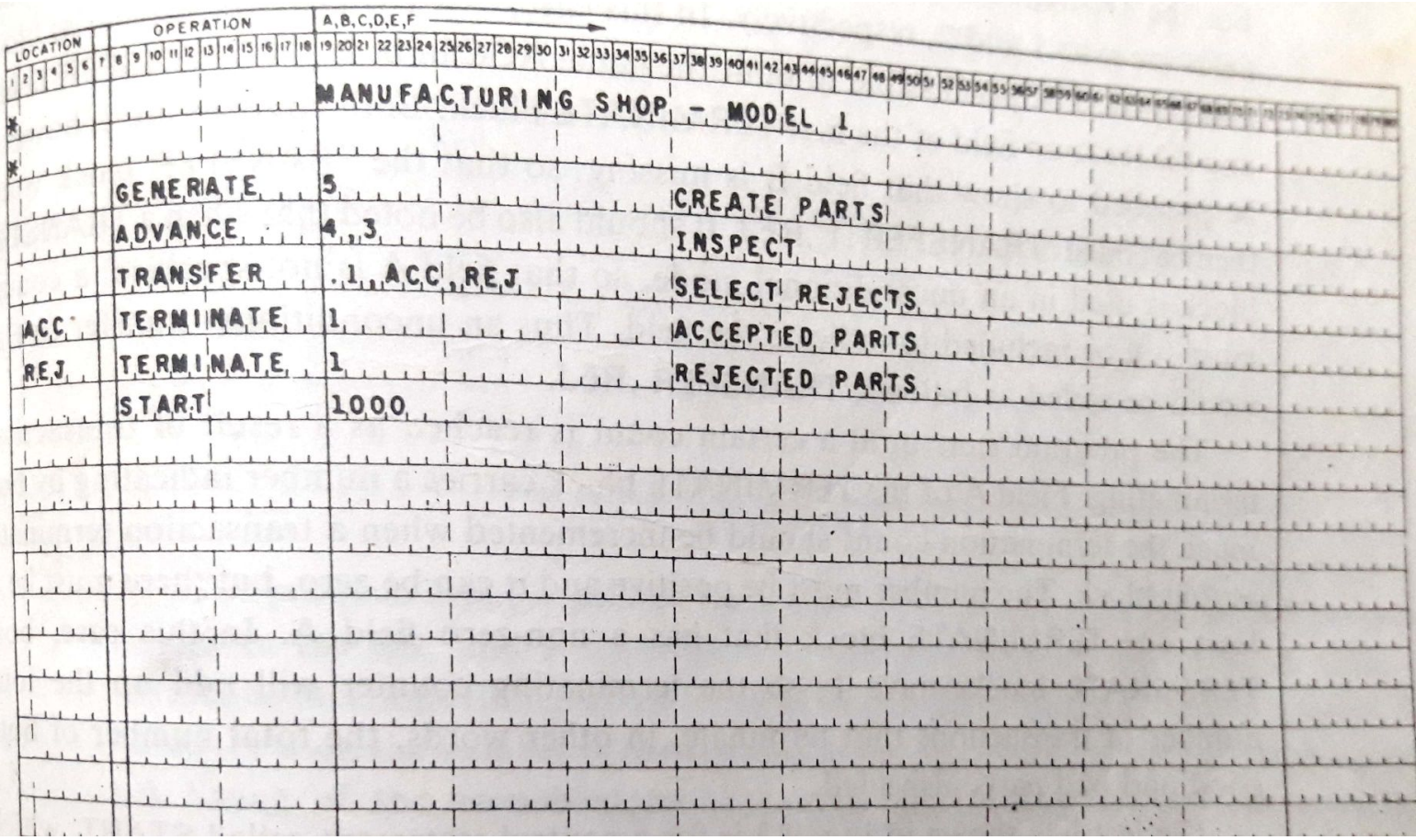


Figure 9-2. Manufacturing shop - model 1.



convention used in drawing blocks is to place the block location (where needed) at the top of the block; the action time is indicated in the center in the form $T = a, b$, where a is the mean and b is the modifier; and the selection factor is placed at the bottom of the block.

A GENERATE block is used to represent the output of the machine by creating one transaction every five units of time. An ADVANCE block with a mean of 4 and modifier of 3 is used to represent inspection. The time spent on inspection will therefore be any one of the values 1, 2, 3, 4, 5, 6, or 7, with equal probability given to each value. Upon completion of the inspection, transactions go to a TRANSFER block with a selection factor of 0.1, so that 90% of the parts go to the next location (exit 1) called ACC, to represent accepted parts and 10% go to another location (exit 2) called REJ to represent rejects. Since there is no further interest in following the history of the parts in this simulation, both locations reached from the TRANSFER block are TERMINATE blocks.

The problem can be coded in a fixed format, as shown in Fig. 9-3. Column 1 is only used for a comment card. An * in column 1 results in the statement being printed in the output only. A field from columns 2 to 6 contains the location of the block where it is necessary for it to be specified. The GPSS program will automatically assign sequential location numbers as it reads the statements, so it is not

```
BLOCK
NUMBER   *LOC    OPERATION  A,B,C,D,E,F,G,H,I        COMMENTS              STATEMENT
                  SIMULATE                                                  NUMBER
                *                                                             1
                * MANUFACTURING SHOP - MODEL 1                               2
                *                                                             3
1                 GENERATE   5                       CREATE PARTS           4
2                 ADVANCE    4,3                     INSPECT                5
3                 TRANSFER   .1,ACC,REJ              SELECT REJECTS         6
4                 TERMINATE  1                       ACCEPTED PARTS         7
5                 TERMINATE  1                       REJECTED PARTS         8
                *                                                             9
                  START      1000                    RUN 1000 PARTS         10
                                                                            11

                             CROSS-REFERENCE
                             BLOCKS
SYMBOL                NUMBER          REFERENCES

ACC                     4               7
REJ                     5               7
```

```
        SIMULATE
*
* MANUFACTURING SHOP - MODEL 1
*
 1      GENERATE    5
 2      ADVANCE     4,3
 3      TRANSFER    .100,,4,5
 4      TERMINATE   1
 5      TERMINATE   1
*
        START       1000

 RELATIVE CLOCK              5005  ABSOLUTE CLOCK          5005
 BLOCK COUNTS
 BLOCK  CURRENT     TOTAL
     1        0      1001
     2        1      1001
     3        0      1000
     4        0       888
     5        0       112
```

Figure 9-4. Output for model 1.

Assuming that there is only one inspector, it is necessary to represent the inspector by a facility, to simulate the fact that only one part at a time can be inspected. The block diagram will then appear as shown in Fig. 9-5. A SEIZE block and a RELEASE block have been added to simulate the engaging and disengaging of the inspector.

Coding and results for this model are given in Fig. 9-6. A line of output is given for each facility, showing how many times it was seized, the average utilization made of the facility, and the average time transactions held the facility. In this case, the results show that the inspector was busy for 73.5% of his time.

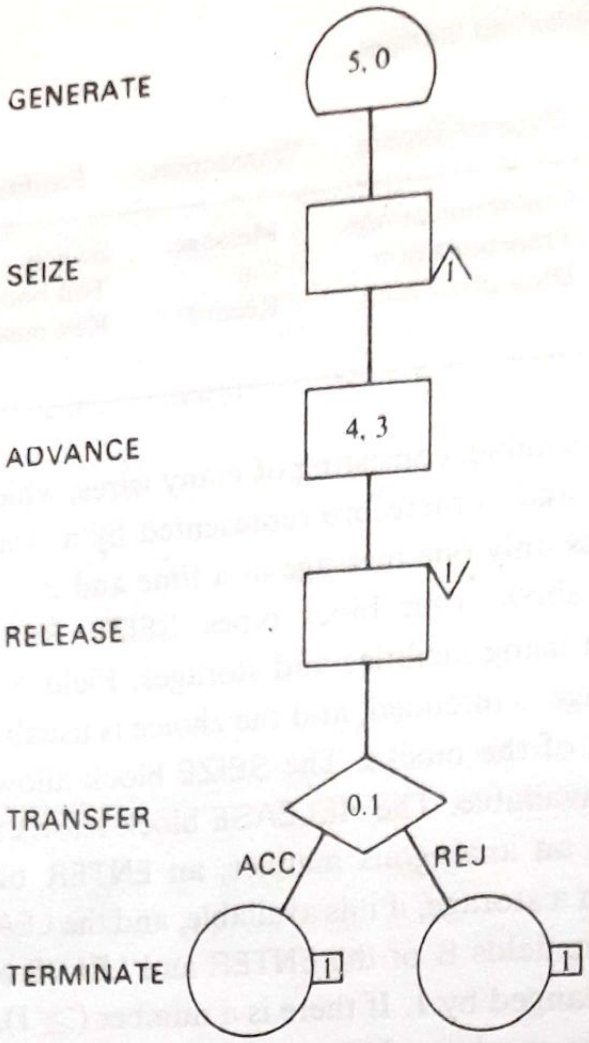


Figure 9-5. Manufacturing shop - model 2.

```
          * MANUFACTURING SHOP - MODEL 2
          *
1                 GENERATE    5             CREATE PARTS
2                 SEIZE       1             GET INSPECTOR
3                 ADVANCE     4,3           INSPECT
4                 RELEASE     1             FREE INSPECTOR
5                 TRANSFER    .1,ACC,REJ    SELECT REJECTS
6           ACC   TERMINATE   1             ACCEPTED PARTS
7           REJ   TERMINATE   1             REJECTED PARTS
          *
                  START       1000          RUN 1000 PARTS

RELATIVE CLOCK          5435  ABSOLUTE CLOCK          5435
BLOCK COUNTS
BLOCK CURRENT    TOTAL
    1       1     1001
    2       0     1000
    3       0     1000
    4       0     1000
    5       0     1000
    6       0      900
    7       0      100

                  ****************************************
                  *                                      *
                  *            FACILITIES                *
                  *                                      *
                  ****************************************

FACILITY   NUMBER  AVERAGE    -AVERAGE UTILIZATION DURING-
           ENTRIES TIME/TRAN  TOTAL  AVAIL.  UNAVAIL. CURRENT   PERCENT
                              TIME   TIME     TIME    STATUS  AVAILABILITY
1          1000    3.995     .735                             100.0
```

Figure 9-6. Coding and results for model 2.

Q There are 15 people in a class learning GPSS. They all started writing a prg at a same time. It takes each people 12 ± 3 mins to write the program. Only 25% of the prg will run correctly for first time. When prg has a error it takes 8 ± 3.5 minutes to do the de-bugging. For the second debug program will run successfully 35% of the time. How long does it take for whole class to finish writing a program?

⇒

```
          SIMULATE
          GENERATE    15
          ADVANCE     12,3
          TRANSFER    .25,AGAIN,DONE
AGAIN     ADVANCE     8,3.5
          TRANSFER    .65,DONE,AGAIN
DONE      TERMINATE   1
          START       15
          END.
```