A COMPUTER SIMULATION MODEL TO PREDICT AIRPORT CAPACITY ENHANCEMENTS

by

Vijay Bhushan G. Nunna

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Civil Engineering

APPROVED:

A.A. Trani, Ph.D, Chairman

A.G. Hobeika, Ph.D

D.R. Drew, Ph.D

July 30, 1991

Blacksburg, Virginia

A COMPUTER SIMULATION MODEL TO PREDICT AIRPORT CAPACITY ENHANCEMENTS

by

Vijay Bhushan G. Nunna

Dr. A.A. Trani

(Chairman)

(ABSTRACT)

The ever increasing demand on the air transportation system is causing a lot of congestion and delays, leading to large monetary losses and passenger inconvenience. This has prompted the development of many analysis tools to help the understanding of the airport system where some improvements could be performed to enhance the capacity of the airports.

The Center for Transportation Research at Virginia Tech, in line with the FAA's Capacity Enhancements Plan, is developing strategies to alleviate the airport congestion problem by developing a model (REDIM) to design and optimally locate high-speed exit taxiways. The objective of this research is to develop a computer simulation model to predict the airport capacity enhancements due to the above mentioned high-speed exit taxiways and as well as due to other changes in operational procedures, aircraft characteristics, airport environmental conditions, etc.

RUNSIM (RUNway SImulation Model), a discrete event simulation model was developed using SIMSCRIPT II.5 language. This model simulates dual operations on a single runway, with capabilities of simulating FAA standard and REDIM designed high-speed

exits, variable intrail separations, different aircraft mixes, and weights, arrival rates and patterns, etc. Currently it has a 30 aircraft data base to perform the simulation. Its output includes such global statistics as total arrival and departure delays, weighted average ROT and its standard deviation, aircraft exit assignment table, arrival and departure event lists. It has the capability to perform multiple iterations on a single application, which helps in performing statistical analyses on the results for better inference.

ACKNOWLEDGMENTS

I sincerely wish to express my gratitude to my advisor Dr. Trani for the guidance he has given me right through this research and my stay at Virginia Tech. His encouragement, flexibility to meet me at any time and his moral support were the main reasons for the successful completion of this research.

I am greatful to Dr. Hobeika for his financial support and having confidence in me and allowing to work on couple of projects at the Center for Transportation Research.

I am also thankful to Dr. Drew for serving in my graduate committee. I always enjoyed his teaching and wisdom and especially appreciate his free spirit.

I thank my parents, sister, and brother for their love and support in my endeavours. It would not have been possible to pursue my graduate studies in U.S., if not for the love and affection of my uncle Dr. Nagabushanam Nunna and aunt Dr. Sitalakshmi Nunna to whom I am indebted forever.

Last but not the least I express my appreciation to all those who helped me during my graduate studies.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

Transportation infrastructure in recent times has been burdened by high demand compared to the limited capacity available causing heavy delays. Air transportation is also one of the transportation systems that is experiencing delays because of the closing gap between the demand and the available capacity. Figure 1.1 illustrates the relation the average congestion delays during peak hours as a function of ratio of demand to capacity. These delays have economic impact on the users and the suppliers of air transportation. Statistics indicate that nearly $ 3 billion are paid by the air travellers due to the delays in the U.S. with another $ 2.1 billion paid by airlines according to the Federal Aviation Administration [FAA, 1988]. U.S., scheduled air carriers recorded a total of 429.1 billion revenue passenger miles in fiscal year 1989 and over the 12-year forecast period the revenue passenger miles are projected to increase at an average annual rate of 4.9 percent, reaching 765.6 billion in fiscal year 2001 [FAA, 1990]. Airlines have changed their routing system from predominantly linear operations to a system of hub and spokes. The development of connecting hub airports has led to high frequencies in peak hours at major airports and as a result approximately 21 airports are experiencing serious congestion problems. The other side effect of hub and spoke system is the chain effect of delays experienced by the interconnected airports. According to FAA the number of congested airports is going to increase to fifty by the end of the century [FAA,

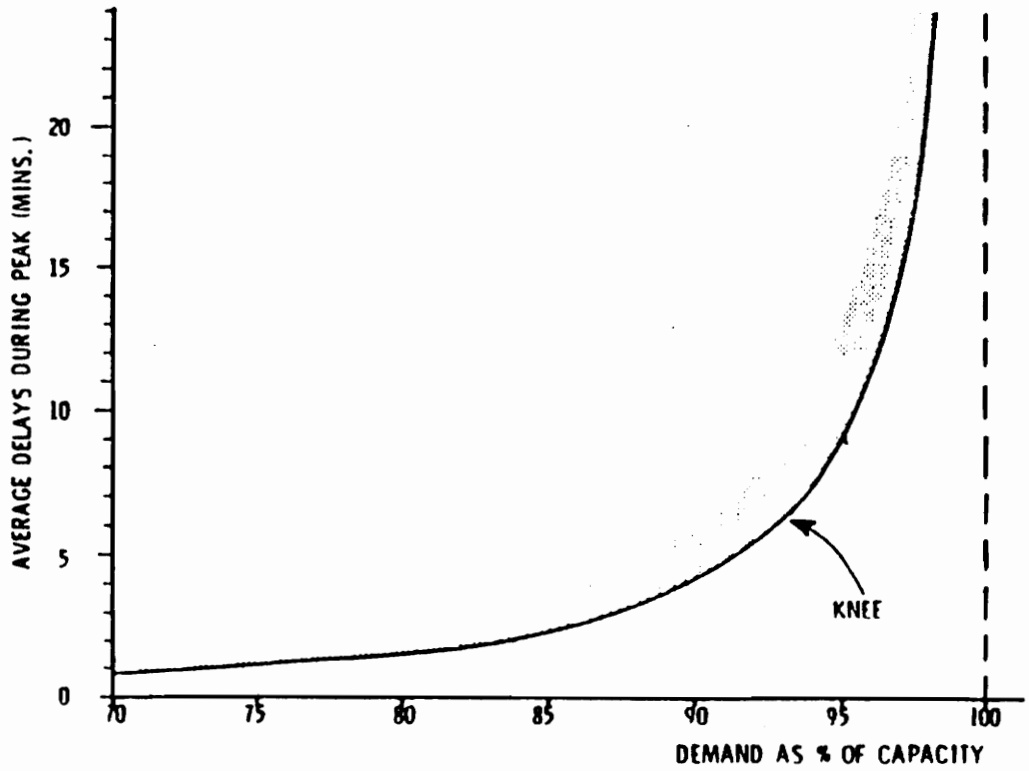

Figure 1.1 Relationship between Average Delay and ratio of Demand/Capacity.

1988] and one-fifth of them will experience more than 50,000 hours of system imposed delays. The construction of new airports to alleviate this problem ia slow and rare process due to the scarcity of land, limited financial resources and, local opposition due to possible environmental pollution, etc,.

1.2 Subject Description

In order to study an airport as a system, it is customary to divide it into two main components:

1) Airside,

2) Landside.

These are again divided into subcomponents (shown in Figure 1.2) which are as follows:

Airside : 1) Airspace and Air Traffic Control (ATC)

2) Runways

3) Taxiways

4) Aprons and Gates

Landside: 1) Terminal Building

2) Parking

3) Ground Access System

Every subcomponent has influence towards the capacity of the airport and each one should complement each other. Capacity is defined as the processing capability of a

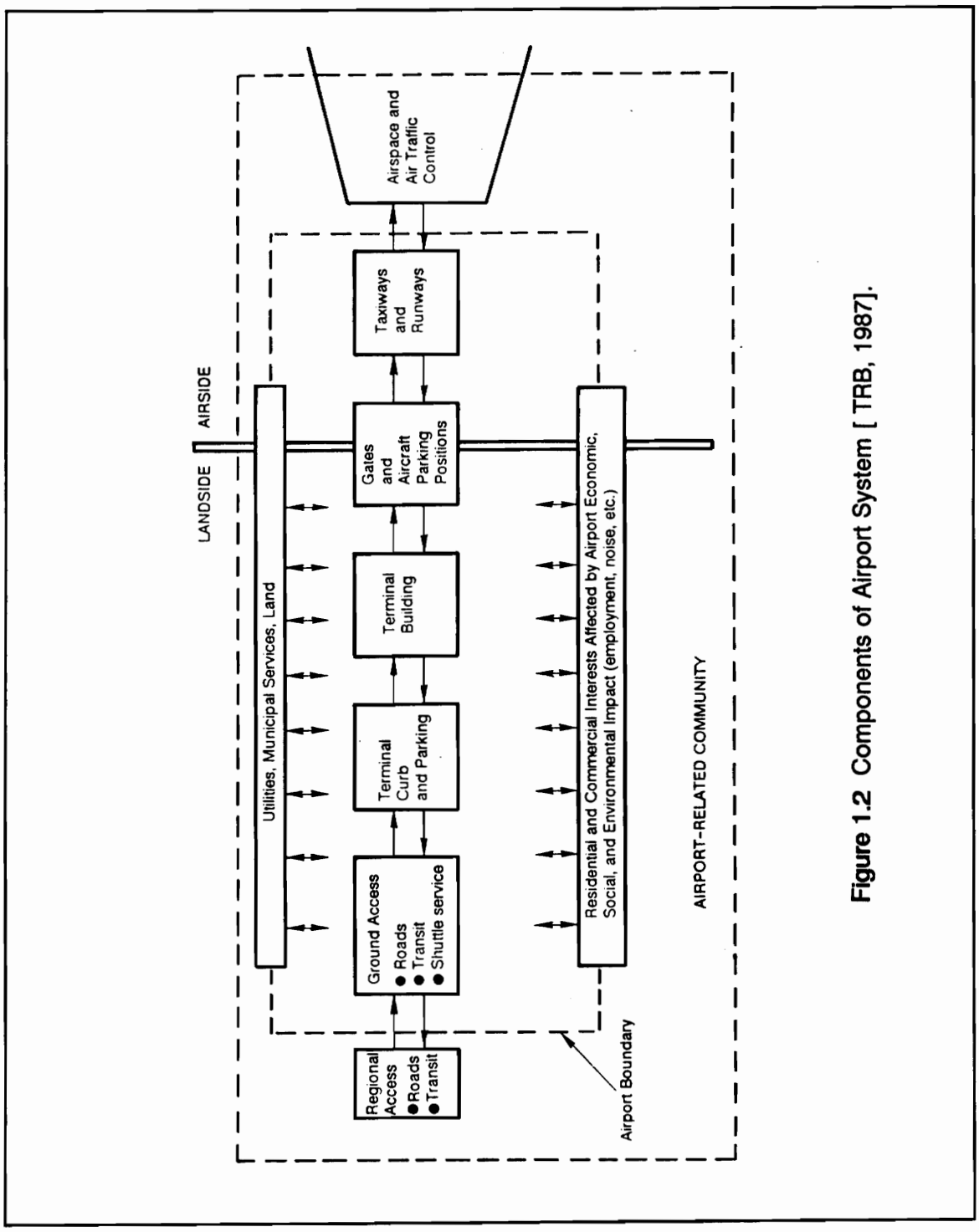

Figure 1.2 Components of Airport System [TRB, 1987].

service facility over some period of time. Traditionally the capacity of the individual subcomponents have been evaluated and the most critical one would dictate the airport capacity. Of the two main components the airside has in general been the critical component which dictated the capacity of the airport (Bangkok airport being a notable exception where the poor ground acces dictates the airport practical capacity). To increase the capacity of the existing air transportation system several topics of interest have been identified by FAA and research is being undertaken ranging in topics from 4D terminal navigation to methods to reduce the runway service time (see Table 1.1). Runway occupancy time (ROT) of aircraft is one of the important factors affecting the capacity of a runway. ROT is the time that an aircraft occupies the runway until a new operation (arrival or departure) can be processed. Some of the most important factors that influence runway capacity are:

1) Intrail separations,
2) Aircraft population mix,
3) Exit locations and their type,

Several studies have suggested that by improving the above factors that there would be an increase in capacity of a single runway by 20 % [Barrer and Diehl, 1988].

1.3 Research Scope

The runway is one of the critical subcomponents of the airport system and if capacity

Table 1.1 Short, Medium and Long Term Projects Affecting Airport Capacity [FAA, 1986]

Time Period	Projects with Highest Effects on Capacity	Projects with Moderate to Significant Effect on Capacity
Near term (1–5 years)	Instruments approaches to converging runways	Microwave landing system
	Independent, closely spaced parallel approaches	Runway configuration management system
	Separate short runways	Enhanced terminal-area radar
	Triple instruments approaches	Wind measuring equipment
		Improved landing and navigation systems and revised air traffic control procedures for rotorcraft
		Improved approach lighting and visual navigation aids
		Improved airport design and configuration
Medium term (6–10 years)	None	Airport surface surveillance guidance and control systems
		Doppler weather radar
		Revised computer algorithms for scheduling and metering arrivals and departures
		Wake vortex detection and avoidance
		Methods of reducing runway occupancy time
Long term (over 10 years)	4D terminal-area navigation	Wake vortex forecasting and avoidance
	Automation of air traffic control in terminal areas	More sensitive and accurate radar
		Low altitude surveillance for rotorcraft and general aviation
		Mode S data link
		Computer-aided decision making in air traffic control
		Advanced wind shear detection
		Improved weather sensors

enhancements to other airside subcomponents are improved, then the runway could become the most critical one. This research focuses on the enhancement of the runway capacity, especially, using optimally located high speed exits.

1.4 Research Objective and Approach

A discrete-event oriented simulation approach is proposed using SIMSCRIPT II.5, a computer simulation language, on a IBM compatible personal computer platform. Computer simulation is a powerful tool to study complex systems which cannot be represented by mathematical formulation. There are time tested analytical tools to evaluate runway capacity performance are being currently used, but they are abstract and mainly useful for long range planning rather than for short range or day to day operational planning. Simulation models reduce the differences between the real world and the abstract world of the model thereby giving better results. To support this contention detailed aircraft and airfield input parameters are planned to be used for the model.

It is proposed to model both the current standard FAA exits and newly proposed rapid runway exit geometries in order to gain appreciation of the ROT gains possible with new exits [Trani, et al, 1990]. These proposed new exit geometries are generated by REDIM (Runway Exit Design Interactive Model), a computer model developed at the Center for Transportation Research, Virginia Tech., which optimizes the location and also generates the geometries of high speed runway exits. This capability of simulating various types of

exits and the ability to modify aircraft arrival and departure patterns make this model suitable to examine runway capacity gains for existing and future ATC systems under realistic airport environmental conditions. The current FAA airspace and airfield simulation model, SIMMOD, does not have the capability of simulating high speed exits realistically, as it assumes for different types of high speed exits some percentage of ROT is spent on those exits while rolling. Also the intrail separations cannot be modified for any simulation runs [USDOT/FAA, 1989].

It is hoped that this research, by demonstrating the effectiveness of the proposed REDIM generated exits, would culminate in adopting the REDIM generated exits into the SIMMOD simulation system making it a more flexible tool.

2. Literature Review

2.1 Introduction

The objective of this literature review is to present some background of past and current research on the influences of airspace separation and runway occupancy time on runway capacity.

Early research on runway capacity began with the development of simple mathematical models to extract only important processes occuring on the runway. Later on graphical and simulation models were developed to incorporate greater details. This chapter discusses the mathematical models first and then graphical and simulation models, respectively.

2.2 Mathematical Models

Mathematical modelling is a convenient and quick method of analyzing any system. The early mathematical models considered a runway as a single channel queuing system with FIFO (First In - First Out) service and the arrivals with poisson probability distribution. In 1948 Brown and Pearcy [Ashford, Wright, 1979] derived an equation for average landing delay, as shown in Eqn. 2.2.1.

$$W = \frac{\rho}{2 \times \mu \times (1 - \rho)} \qquad \ldots(2.2.1)$$

where ρ = load factor = λ/μ

λ = arrival rate (aircraft/unit time)

μ = service rate (aircraft/unit time) = 1 / b

b = mean service time (this could be runway occupancy time or ATC minimum separation rule)

A general rule of the above equation known as the Pollaczek-Khinchum formula is given below:

$$W = \frac{\rho \ (1 + C_b^2)}{2 \times \mu \ (1 - \rho)} \qquad \ldots(2.2.2)$$

where C_b = coefficient of variation of service time = σ_b / b

σ_b = standard deviation of mean service time.

These equations could be used either for arrivals or departures, but are applicable for single operations only. For mixed operations, where arrivals are given priority over departures, the delays for the arrivals is estimated by either Eqn. 2.1 or 2.2 and the average delay to departures is given by Eqn. 2.2.3 [Horonjeff, McKelvey, 1983]:

$$W_d = \frac{\lambda_d \ (\sigma_j^2 + j^2)}{2 \ (1 - \lambda_d j)} + \frac{g \ (\sigma_f^2 + f^2)}{2 \ (1 - \lambda_a f)} \qquad \ldots(2.2.3)$$

where,

W_d = mean delay to departing aircraft, time units.

λ_a = mean arrival rate, aircraft/unit time.

λ_d = mean departure rate, aircraft/unit time.

j = mean interval of time between two successive departures.

σ_j = standard deviation of mean interval of time between two successive departures.

g = mean rate at which gaps between successive arrivals occur.

f = mean interval of time in which no departure can be released.

σ_f = standard deviation of mean interval of time in which no departure can be released.

All previous equations are valid only if the mean arrival or departure rate is less than the mean service rate.

More detailed mathematical models were proposed by Harris [Harris, 1972]. The models developed considered more factors that affect the runway capacity than the models discussed above as these do not account for the length of the common approach path, individual aircraft speeds, and intrail separations. Some of the models proposed by Harris are discussed below:

IFR Landing Intervals Model and Arrival Capacity: This model determines the nominal time separation between two aircraft travelling at speeds V_1, and V_2, which must be

achieved in order to maintain a constant probability of separation violation. Some of the model variations that are considered under this model are:

a) Error Free Case:

"Error free" it implies that a trail aircraft is following another aircraft in the final approach path exactly by a set separation distance without any human or technical error.

$$m(V_2, V_1) = \frac{\delta}{V_2} \qquad (V_2 \geq V_1) \qquad \ldots(2.2.4)$$

or

$$m(V_2, V_1) = \frac{\delta}{V_2} + \gamma\left(\frac{1}{V_2} - \frac{1}{V_1}\right) \qquad (V_2 < V_1) \qquad \ldots(2.2.5)$$

where,

T_i = actual time aircraft i crosses the threshold.

V_i = speed of the aircraft i.

γ = length of the common path.

$m(V_2,V_1)$= Error free minimum time separation over threshold for aircraft 2 following aircraft 1.

δ = minimum safety separation between landing operations.

If aircraft are processed on a FIFO basis, then the expected minimum landing intervals in error free approach is described by the Eqn. 2.2.6

$$\langle m \rangle = \int_0^\infty \int_0^\infty m(V_2, V_1)\ f_v(V_1)\ f_v(V_2)\ dv_1 dv_2 \qquad \ldots(2.2.6)$$

where, $f_v(.)$ is the probability density function describing the speed mix of the arrival aircraft. Hence the landing capacity (λ_m) for an error free system is given by Eqn 2.2.7.

$$\lambda_m = \frac{1}{< m >} \qquad \ldots(2.2.7)$$

b) Interarrival Error Case:

In any real-world system errors are bound to occur, so is with the aircraft following another aircraft in the final approach path in violating the minimum separation rules. The possible sources of errors are when pilots try to achieve the minimum separation, and during ATC manual control when the position of aircraft is not located properly, etc. To account for error in minimum separation a buffer time is added to the minimum separation time i.e., the scheduled interval at the threshold, which is the expected value of $T_2 - T_1$, is simply the sum of the minimum separation and buffer time.

$$< T_2 - T_1 > = m(V_2, V_1) + b(V_2, V_1) \qquad \ldots(2.2.8)$$

where,

$b(V_2, V_1)$ = Buffer time between aircraft 2 and 1.

$<.>$ = Expected value.

Figure 2.1 shows the position of the trail aircraft as it approaches the threshold with and without buffer. If we assume that the actual interarrival times are equal to the expected value plus a zero-mean normally distributed random error, e_o (positive for the second arrival late) with a fixed standard deviation σ_o, then for a given probability of violation p_v:

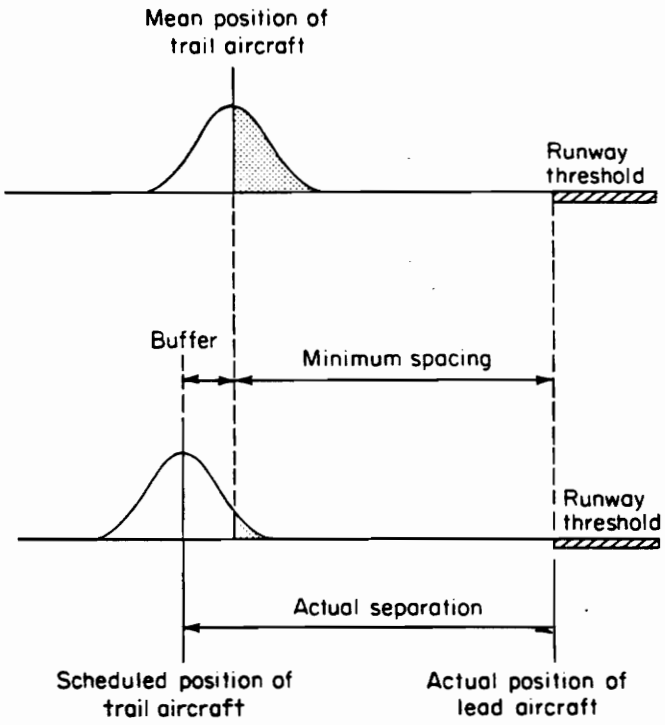

Figure 2.1 Interarrival Separation without and with Buffer [Horonjeff, Mckelvey, 1983].

$$T_2 - T_1 = m(V_2, V_1) + b(V_2, V_1) + e_o \qquad \ldots(2.2.9)$$

If $V_2 \geq V_1$ (Closing Case) then

$$b(V_2, V_1) = \sigma_o \, q(p_v) \qquad \ldots(2.2.10)$$

where $q(p_v)$ is the value for which the cumulative standard normal distribution has the value $(1 - p_v)$.

If $V_2 < V_1$ (Opening Case) then

$$b(V_2, V_1) = \sigma_o \, q(p_v) - \delta \left(\frac{1}{V_2} - \frac{1}{V_1} \right) \qquad \ldots(2.2.11)$$

The above equation is limited to a non-negative value only, with a minimum of zero.

$$\langle l \rangle = l(V_2, V_1) = m(V_2, V_1) + b(V_2, V_1) \qquad \ldots(2.2.12)$$

Eqn 2.2.12 is the scheduled landing interval, with λ_l as the saturation landing capacity given by Eqn 2.2.13.

$$\lambda_l = \frac{1}{\langle l \rangle} \qquad \ldots(2.2.13)$$

c) Runway Occupancy Limitation: In the cases a) and b) runway occupancy was not considered for minimum interval time, to confirm to single occupancy rule. If $< R_j >$ is runway occupancy time of the lead aircraft, then a modified landing interval will be: $\Lambda_{ij} = \max\{ < m >, < R_j > \}$. To account for some errors in runway occupancy time, a probability of runway occupancy violation is considered, hence

$$r_j = \langle R_j \rangle + e_R \qquad e_R \sim N(0, \sigma_R) \qquad \ldots(2.2.14)$$

$$\langle l \rangle = \langle l \rangle + e_l \qquad e_l \sim N(0, \sigma_{ia}) \qquad \ldots(2.2.15)$$

where σ_{ia} is net interarrival error, over threshold. Assume also that the probability of violation is set to some constant multiple, say n, of p_v. Then

$$\Lambda = \max[\langle l \rangle, \langle R_j \rangle + q(nq_v)\sqrt{\sigma_R^2 + \sigma_{ia}^2}] \qquad ..(2.2.16)$$

Capacity for Mixed Operations under IFR: Here the ATC rules are applicable to the mixing of a departure stream into the arrival stream. Let $| T_i, T_{i+1} |$ be the landing times (over threshold) of the i^{th} and $i+1^{st}$ aircraft in the arrival stream and T_j as the enplaning time of the j^{th} departure, which is to be interleaved between i and i+1 arrivals. To confirm to the ATC operational procedures, the following rules apply:

Rule A (Arrival Priority): The arrivals are given priority, and departures are required to wait for a gap in the arrival stream, mathematically, T_i and T_{i+1} are fixed with T_j to be inserted in between, if possible.

Rule B (Single Occupancy): A departure or arrival may not be processed until the previous arrival has safely exited the runway, mathematically, $T_j \geq T_i + R_i$ (also $T_{i+1} \geq T_i + R_i$), where R_i is the ROT for arrival i.

Rule C (Departure/Arrival Spacing): A departure may not be released if an arrival is less than δ_d distance from runway threshold, mathematically, $T_j \leq T_{i+1} - ({}_2t_i+1)$, where$({}_2t_i+1)$ is the i + 1st arrival to travel δ_d distance to reach threshold.

Rule D (Interdeparture Spacing): The departure stream must space itself by some minimum time separation based upon the type of aircraft, mathematically, $T_{j+1} \geq T_j + \Gamma_d$, where Γ_d is the minimum interdeparture spacing.

Figure 2.2 illustrates all the above rules graphically and the Eqn. 2.2.17 summarizes all the above rules.

$$T_{i+1} - T_i > R_i + {}_2t_{i+1} + (n_d - 1)\Gamma_d \quad \ldots(2.2.17)$$

Where n_d is the number of departures that may be released between two arrivals only if a time interval (i/i+1) is present. If p_G is acceptable level of probability of violation, to account for the system errors, then the average interarrival interval, in order to release a departure is given by Eqn. 2.2.18.

$$< T_2 - T_1 > \geq \sigma_G q(p_G) + < R_1 > + <\frac{\delta_d}{V_2}> \quad \ldots(2.2.18)$$

2.3 Graphical Models

The most widely used graphical model to estimate runway capacity was developed by FAA in Advisory Circular AC 150/5060-5 [FAA,1983]. The charts in AC 150/5060-5 can be used to find the hourly capacity of the runway system by using various parameters

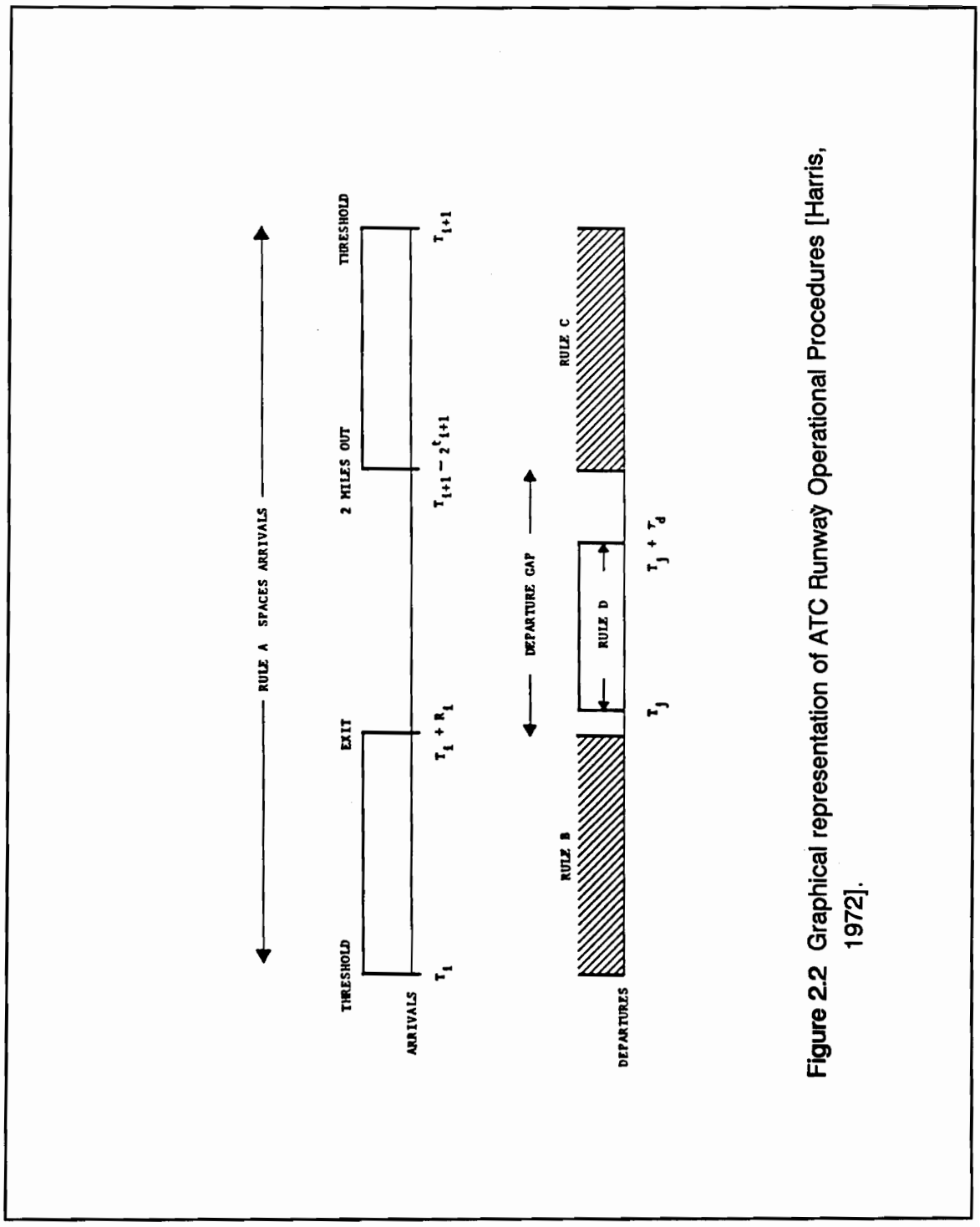

Figure 2.2 Graphical representation of ATC Runway Operational Procedures [Harris, 1972].

affecting runway capacity. These charts are developed from computer simulation. These charts are used to determine the runway hourly capacity through the Eqn. 2.3.1.

$$C = C_b E T \qquad \ldots(2.3.1)$$

where C = hourly capacity of runway-use configuration in operation per hour.

C_b = ideal or base capacity of runway-use configuration.

E = exit adjustment factor for number and location of runway exits.

T = touch-and-go adjustment factor.

Parameters required to use the above equation and the charts are:

a) Prevailing operating condition (IFR or VFR).

b) The mix index MI, which is an indicator of the level of air-carrier-type operation on the runway and it is calculated as given in Eqn. 2.3.2.

$$MI = C + 3D \qquad \ldots(2.3.2)$$

where,

C = percentage of type C aircraft in mix of aircraft using runway.

D = percentage of type D aircraft in mix of aircraft using runway.

c) Percent arrivals (PA).

d) Percent of Touch and Go's.

e) Location and number of exits.

f) Runway system, whether it is single, parallel or intersecting with another one. The runway systems are categorized and for every category an appropriate chart is to be used to evaluate the hourly capacity.

The charts could be also used to find the hourly delay by identifying the hourly demand (HD), peak 15 minute demand (Q). The hourly delay (DTH) is calculated by Eqn. 2.3.3.

$$DTH = \frac{HD\,(PA \times DAHA + (100 - PA)\ DAHD)}{100} \quad \ldots(2.3.3)$$

where DAHA and DAHD are average delays for arriving and departing aircraft respectively.

These FAA charts are mainly helpful for long range planning. They cannot predict the change in capacity or delays due to changes in the sequencing of arrivals, arrival rate. Enough importance is not given to the type of exits that are being used, and the variability of aircraft landing weights.

2.4 Simulation Models

The most comprehensive simulation model used in airport system planning is SIMMOD, the airport and airspace simulation model developed by FAA [DOT/FAA, 1989]. This model can be used in wide variety of scenarios ranging from enroute and terminal area air traffic studies to airport/airline ground operations. Table 2.1 shows a partial list of possible SIMMOD analyses topics.

The SIMMOD simulation engine is a discrete-event simulation written in SIMSCRIPT II.5 simulation language. SIMMOD represents the basic framework of any airport or airspace

system as a series of nodes connected by links. The flight paths, runways and taxiway are depicted by links and their intersections by nodes. In SIMMOD a flight is an aircraft with a unique identifier and a set of data related to it, say, type of flight, starting time and airspace route, and depending on the scenario, arrival at an airport and departure from an airport, etc., are also specified.

SIMMOD uses the Integrated Noise Model (INM) version 3.9 database for aircraft identification and operating characteristics. For the purpose of simulating airspace operations, aircraft are classified into groups, as many aircraft have roughly equivalent characteristics when airborne. Those groups are Heavy, Large, Small, General Aviation, and others (user specified). In airfield operations also the aircraft are categorized into the above groups. The characteristics defined for each group include:

Landing Characteristics: Landing and takeoff roll distances used are based on the observed probabilities which are translated into cumulative distributions. These probabilities are linked to aircraft type and are specified for each airfield. Thus, if the landing rolls are based on observed values then for future scenario simulations the application has to depend on assumed landing roll values. The roll time while landing depends only on the aircraft group to which the aircraft is assigned. The roll times SIMMOD uses for different aircraft group are shown in Table 2.2. The above description implies that irrespective of the cumulative distributions of landing roll the ROT is constant. This is a major deficiency as time and distance are interrelated in the motion of any vehicle.

Table 2.1 Typical SIMMOD Analysis Topics [DOT/FAA, 1989].

Airport Facilities

- Impact of new facilities.
- Expansion or relocation of existing terminal.
- Relocation of gates.

Airfield Design and Procedures

- Revision of terminal routing plan.
- Runway and Taxiway Configuration.
- New runway construction.
- High-speed runway exits.
- Runway and taxiway holding pads.
- Reduction of runway occupancy time.
- Parallel approaches.
- Converging approaches.
- Microwave Landing Systems (MLS).
- Location of navigational aids.
- Apron area operations.
- Queuing strategies and departure rules.

Airspace Design and Procedures

- Revision of separation rules.
- Speed and altitude restrictions.
- Controller tactics.
- Realignment of en route and terminal airspace.
- Etc.

Operations

- Aircraft performance.
- Hub and spoke operations.
- Traffic demand and fleet mix.
- Revised ATC procedures.
- Redistribution of departure scheduled at peak hours.
- Visual and instrumental flight procedures (VFR & IFR).
- Etc.

Other

- Noise abatement procedures.
- Wind conditions (speed, direction, ceiling and visibility).

Takeoff Characteristics: The takeoff roll and times are also categorized similarly as the landing characteristics.

Gate Occupancy Characteristics: The loading and unloading times can be described by cumulative distributions for different groups.

In SIMMOD runway is defined as a list of links from one end to the other and can be used both directions. Runway exits can defined at the end of each link on a runway. The selection of an exit by an arriving aircraft depends an where the aircraft finishes its landing roll. Any exit reached after completion of the roll is a viable exit. A high speed exit is also represented by a link. The difference between the headings of the link and the runway determines the amount of the landing roll that may be completed on the high speed exit, as shown in Table 2.3. The selection of high speed exit on the basis of which factor is not defined clearly. As the landing roll and ROT are not related, the use of an high speed exit during a simulation does not decrease the runway occupancy time. Since the dynamics involved in aircraft landing roll are not considered, the exit speed of the high speed exits and the exiting speed of the aircraft will not correlate.

In response to the FAA and NASA needs the Center for Transportation Research developed REDIM 1.0 [Trani, et. al., 1990] a computer model to expedite turnoff designs and to optimize the location of high speed exits. This model addresses the placement of optimal turnoff locations considering arrivals only as these operations have a logical influence on the turnoff location. The aircraft simulation starts from the time it crosses

Table 2.2 SIMMOD Roll Time Data.
[USDOT/FAA, 1989]

Airline Group Name	Roll time (Sec)
GA	54
SMALL	45
MEDIUM	50
LARGE	50
HEAVY	60
Other	50

Table 2.3 Relationship between High-Speed Exit Heading and Total Roll Time.
[USDOT/FAA, 1989]

Change in heading	% of roll completed on exit
10°	20%
20°	15%
30°	10%
40°	5%

runway threshold until the vehicle's wingtip clears the runway edge. The simulation of aircraft are independent of each other and hence operational procedures such as interarrival spacing, etc., are not involved in the simulation. The model does not consider the variation in landing weights, due to different flight lengths, during simulation and the overall emphasis of the simulation module is to identify the optimal exit location for every individual aircraft of the aircraft population that use the runway under varying runway environmental conditions. These locations are then used as inputs for the optimization module to locate the exits optimally for the whole aircraft population. Hence this model cannot predict the effect of the optimally located exits on the capacity and delay of the runway in use. This shortcoming is being addressed in this complementary research with the development of RUNSIM (Runway Simulation Model).

3. Methodology

The approach followed by this research is to develop a computer based, discrete-event simulation model which will simulate the runway operations. Simulation is an effective way of pretesting proposed systems,and associated operational policies before developing expensive prototypes, field tests or actual implementations. The model is amenable to manipulation that would be impossible, too expensive, or impractical, to perform on the system it portrays. Discrete-event is chosen over continuous simulation because it describes a system in terms of logical relationships that cause changes of state at discrete points in time rather continuously over time, and this research is interested in knowing the behavior of the components of the system being model at important event times only.

3.1 Description of SIMSCRIPT II.5

The widespread use of simulation as an analysis tool has led to the development of a number of languages specifically designed for simulation [Pritsker,1986; Russel, 1987; Pritsker, 1974; Pugh, 1970; Henriksen and Crain, 1983]. The languages provide specific concepts and statements for representing the state of a system at a point in time and moving the system from one state to another.

SIMSCRIPT is a computer language developed by Kiviat, Villaneueva, and Markowitz

[Russel, 1987]. SIMSCRIPT II.5 is the current version of SIMSCRIPT revised by Alasdar [Russel, 1987]. The simulation modelling framework of SIMSCRIPT II.5 is primarily event-oriented. The state of the system is defined by entities, their associated attributes, and by logical groupings of entities referred to as sets. The dynamic structure of the system is described by defining the changes that occur at event times.

In SIMSCRIPT II.5, two types of activities are considered. An entity which remains throughout a simulation is referred to as a permanent entity. A temporary entity is one which is created and destroyed during the execution of the simulation program. The entities might be flights and airports in a simulated air transport system. In a queuing system, each server could be modelled as a permanent entity and the customers as temporary entities. For each type of entity, appropriate names could be given to the attributes that characterize the entity. For example a temporary entity named MAN could be defined by the following statement in **preamble** (described in the latter part of the chapter):

```
Every MAN has a HEIGHT,
          a WEIGHT,
          a AGE
```

The attribute of a particular MAN could be accessed through references as HEIGHT(MAN), WEIGHT(MAN), and AGE(MAN). A particular man is specified by the value of the variable MAN. Thus, the EVERY statement defines a class of objects, each called

MAN, having similar properties. Every MAN, of which there may be many, is portrayed by these attributes.

Sets provide a mean to model the organization of entities in a system. For example, a set containing customers waiting for service could be named as QUEUE and the related statements could be written as:

Every CUSTOMER may belong to QUEUE

The system owns a QUEUE

The first statement above declares that each entity class CUSTOMER could be a member of the set QUEUE. Each set declared in SIMSCRIPT has to be owned by an entity, or a process or the system and hence the second statement accompanied the first. The important points to be noted in a set are:

1) A set is made up of entities, or processes that point to one another, thereby expressing their member relationships i.e., sets are like arrays in that each member elements of which they are composed may be identified and manipulated, but in contrast with static structuring imposed on array elements, the organization of members in sets may be dynamic and changeable.

2) A specific entity can own or belong to any number of sets as long as it has the required pointer attributes.

3) Every program commences execution with empty sets. As a program proceeds, statements are executed that file entities in sets, examine sets, and remove entities from sets.

In SIMSCRIPT II.5 the beginning and end of an activity, with passage of time, is represented by a **process**. An activity within a system is bounded by two instantaneous events; when activity starts, and when it stops. Thus the event is the simplest component of an activity description. The important properties of an event are: 1) it occurs at some instant of time, and 2) the occurrence is instantaneous. Figure 3.1 illustrates an activity delimited by two events and its relationship with processes. A process could be defined as a time-ordered sequence of events and may encompass several activities. The passage of time could be predetermined or indeterminate. Predetermined lapse of time could be service time (deterministic or stochastic) and indeterminate passage of time could be due to delay caused by competition for scarce resources.

Resources are passive elements of a model. A **resource** is used to model an object required by a **process**. For example a teller in a bank is considered as a resource. He is required by customers (entity) to process an activity. When the teller is busy, other customers have to wait until he is free. Included in the resource concept is the automatic queuing of the processes for unavailable resources and their automatic reactivation when the required resources become available. Resources are declared in preamble. A resource, in SIMSCRIPT, is represented as a permanent entity with some predetermined attributes:

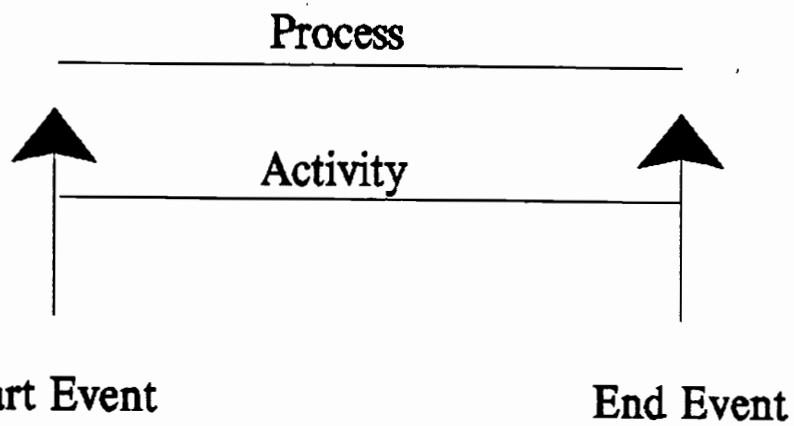

Figure 3.1 Relationship between Events, an Activity and a Process.

U.resource - specifies the number of this resource currently available.

Each resource also has the owner attributes for maintaining two sets:

Q.resource - is the set of processes currently waiting for this resource.

X.resource - is the set of processes currently using this resource.

Because the resources are modelled as permanent entities, they must be created before being used. For example:

```
Create every RUNWAY(1)
let U.RUNWAY(1) = 1
```

The above statements state that there is only one type of runway, and there is only one of them.

Any SIMSCRIPT II.5 simulation model consists of three primary elements [Russel, 1990]: 1) Preamble, 2) Main program module, and 3) Processes. The following paragraphs describe in some detail each one of these:

1) A **PREAMBLE** is used to define the static structure of the model by prescribing the name of permanent and temporary entities, their attributes, set relationships, resources and processes. It is the first section of any SIMSCRIPT II.5 simulation model. It is not part

of a executable program. This section is also used for changing background conditions(whether a variable is real, integer, text or alpha), specifying data structures (defining arrays and their dimensions) other than processes and resources, and listing performance measurements (to collect statistics of the run) to be made. For example:

```
normally mode is integer
```

It implies that the mode of all variables that are not explicitly defined will be integer.

```
define NAME,
        COLOR as text variables
define DATA as a 2-dimensional integer array
```

The principal outputs of simulation experiments are statistical measurements. Such quantities are the average length of a waiting line and the percentage of idle time of a machine are typical examples. Two features **accumulate** and **tally**, allow such information to be gathered during a simulation run, without requiring any other explicit action to be specified within the program. A statement of the form in preamble:

```
tally compute list of names
```

performs computations similar to those of ordinary computations in a routine, but in a global manner, over time, rather locally to an instance of its use. Consider the following

example,

```
tally AVERAGE.WEIGHT as the mean,
    MAXIMUM.WEIGHT as maximum of WEIGHT
```

where, AVERAGE.WEIGHT, MAXIMUM.WEIGHT are global variables collecting statistical data (mean, and maximum respectively) of global variable WEIGHT.

Statistical computations of a different type are made when the word **accumulate** replaces **tally**. These calculations introduce simulation time into average, variance, and standard deviation calculation, weighing the collected observations by the apparent length of the simulation. An example of **accumulate** statement is as follows:

```
Accumulate AVERAGE.QUEUE as the mean,
    and MAXIMUM.QUEUE as the maximum of QUEUE
```

Accumulate and **tally** statements cannot be declared for the same variable. This section is headed by the word **preamble** and terminated by the word **end**.

2) A **MAIN** program is where the execution of any SIMSCRIPT II.5 program begins. In this section, usually, resources are created and initialized before used by processes. A typical statement is as follows:

```
Create every RUNWAY(1)
let U.RUNWAY(1) = 1
```

SIMSCRIPT II.5 requires that an event be awaiting execution before a simulation commences. This is done by activating initial processes in MAIN. For example:

```
Activate a MACHINING (time units)
```

where MACHINING is a process.

Simulation begins when control passes to a system-supplied timing routine. This is done by executing the START SIMULATION statement. A timing routine is the heart of any discrete-event simulation model. This routine ties the entire collection of processes together, as shown in Figure 3.2. Any statement following the START SIMULATION will not be executed until the simulation has terminated. At this point final reports could be produced and a new simulation run could be initiated. This routine is started by using the word **main** and terminated by using the word **end**.

3) A **PROCESS** routine for each process is declared in the preamble. The names of the process object and the process routine are identical. A process routine embodies the logic description of a process, describing the job done by the process object under all circumstances. A routine is declared to be a process routine rather than a callable subprogram by use of the word **process** rather than **routine** in the routine definition.

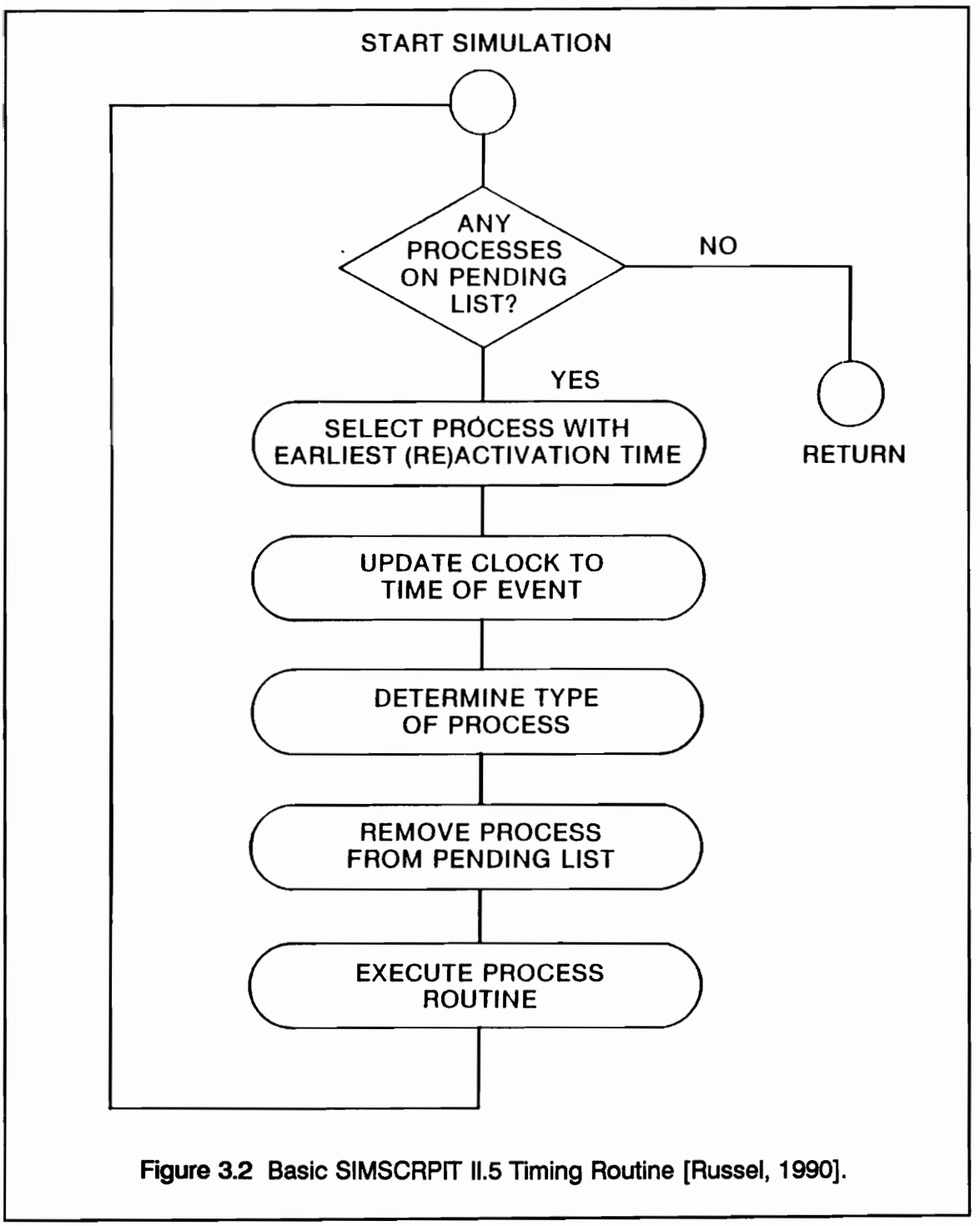

Figure 3.2 Basic SIMSCRPIT II.5 Timing Routine [Russel, 1990].

The general form of the process routine declaration statement is:

process name (optional input argument list)

Each process must be declared in the **processes** section of preamble.

A process requests a quantity of any given resource using a **request** statement as shown below:

Request 1 RUNWAY(1)

If the requested quantity of resources is available, it is given to the process, and the process continues at the statement following the **request** statement. A process that has requested some units of resources may **relinquish** some number of these, but not necessarily all it has. An example of **relinquish** statement is as follows:

Relinquish 1 RUNWAY(1)

The duration of occupation (or usage) of a resource is modelled by a **work** (**wait**) statement. For example:

Work expression time units

Work 5 days

Work (A + B) hours

The **work** statement has to be stated in between the **request** and **relinquish** statements to depict the usage of resource(s).

Other than the above three primary elements (Preamble, Main, Process) of a SIMSCRIPT II.5 simulation model, it may contain one or more subprograms (routines). Subprograms are not executed directly but are subordinate to a higher level routine, where **Main** routine is at the highest level in the hierarchy.

Though SIMSCRIPT II.5 is popular for its discrete event simulation, it has a continuous simulation capability as well. Given an equation for the rate of change of a variable, it calculates the value of the variable and continuously checking a (or some) condition(s). This is the basis for continuous simulation. For example the following SIMSCRIPT II.5 code shows a typical continuous simulation model:

```
Process SHAPE
  Work continuously evaluating 'EQUATIONS'
                                  testing 'QUIT'
End
Routine EQUATIONS Given .SHP
  Let D.ANGLE(.SHP) = -.1 "radians/second
  Let D.X(.SHP) = SPEED(.SHP) * cos.f(ANGLE(.SHP))
  Let D.Y(.SHP) = SPEED(.SHP) * sin.f(ANGLE(.SHP))
End
```

```
Function QUIT
  If time.v > 5 * seconds
    Return with 1
  Endif
  return with 0
End
```

SIMSCRIPT II.5 provides many built in functions to simplify the simulation model especially with probability distribution functions, random number generators, etc. Some of the built in probability distribution functions it has are shown in Table 3.1. Other types of functions and routines are shown in Table 3.2.

SIMSCRIPT II.5 has also the capability of static or dynamic presentation graphics, interactive graphics, and animated displays with help of SIMGRAPHICS, an extension of SIMSCRIPT II.5. The typical applications include [CACI, 1991] :

Data Presentation: Histograms, pie-charts, x-y plots, dials and graphics can be generated to display numerical or statistical information.

Model Representation: Animation graphics can represent the system under simulation and evolve as the simulation progresses.

Run Configuration: Simulation experiments start from some initial conditions, which can

be easily set by manipulating a graphical representation of the system. Interactive reconfiguration at run-time can reduce the iterations needed to achieve stable results, and make simulation programs more general.

SIMGRAPHICS has a built in graphics editor, which helps in creating icons, formatting graphics, and different types of forms like dialog boxes, value boxes, check boxes, buttons and menu bars. When these created icons, forms, etc., are interfaced with the variables of the simulation model the desired graphical presentation output is observed.

4. Model Description

This chapter deals with the description of the simulation model (RUNSIM) developed as part of this research. The description includes the boundaries of the model, assumptions, input parameters, the mathematical computations involved, the algorithms of the simulation which are the heart of the simulation and in the end detailed description of some of the important routines of the source code.

4.1 Model Environment

The process of formulating a simulation model is one which is largely an art. The model should be easily understood, yet sufficiently complex to realistically reflect important characteristics of the real system. The amount of detail included in the model is based on the purpose for which the model is built. Only those elements that could cause significant differences in the decision-making process are considered.

The elements of the airport terminal area and air traffic control procedures included in the simulation model are :

1) In the terminal area air traffic control the boundary for the simulation is the "approach gate", from where the final approach path (common glide path) to runway commences.

2) In the landing area only one runway is modelled.

3) The arrivals exit the simulation immediately after they clear the runway.

4) For departures the aircraft are simulated from the apron at the departure end of the runway till they clear the runway while enplaning.

4.2 Model Assumptions

To simplify the system, and yet capture the essence of the system being modelled the following assumptions were made:

1) The arrivals and departures are generated independent of each other. They are generated by user selected arrival distributions.

2) The arrivals are generated at the approach gate.

3) If the arrivals are greater than the processing capacity of the airport system then the arrivals are queued in a stack at the approach gate.

4) There is no time lag from the time inter-arrival separation at the approach gate is satisfied and the time at which the lagging aircraft departs the stack and enters the final approach path. But this situation is taken care by buffer separation time.

5) For allocating the runway, arrivals are given priority over departures if both events were to occur at the same time.

6) The aircraft has constant velocity in the final approach phase.

7) There is no wind effect.

8) The runway has no gradient.

9) The runway is used only in one direction.

10) Touch and Go operations are not considered. This is so because the model is intended for large commercial airport applications where high speed exits would help to increase the capacity of runway operations under IFR conditions. Touch and Go operations are not allowed under these circumstances.

4.3 Model Parameters.

The output of simulation model is a function of various input parameters. The input parameters of the model and their brief description are as follows:

1) Aircraft Population Mix:

The user can choose the aircraft from a data base of 30 aircraft, shown in Appendix C. The percentage of specific aircraft to be generated for arrivals and departures independently should also be specified.

2) Arrival and Departure Rates:

The inter-arrival distributions for arrivals and departures could be selected independently from a choice of three distributions. They are:

a) Poisson

b) Exponential

c) Uniform

3) Number of Arrivals and Departures:

The number of arrivals and departures should be selected independently for any application to run. Once the model generates all the arrivals and departures the simulation stops only after all the aircraft are processed.

5) Weight Factors for Arriving Aircraft:

One of the important factors that dictate the landing roll of an aircraft is its landing weight. Landing weight at destination is defined as sum of the operating empty weight, the payload, and reserve fuel [Horonjeff, Mckelvy, 1983]. This weight should not exceed the maximum structural landing weight of the aircraft. For example an aircraft within the arrival mix landing at different times would be having different landing weights. This is because of the possible different origins and travel stage lengths. To capture this variation in landing weights a term called "Weight Factor" is introduced. This is a non-dimensional constant which accounts for the fuel consumed during each random flight. The user has to specify mean and standard deviation of the weight factor for every aircraft that was selected for arrivals.

6) ATC Intrail Separations:

Minimum separations are part of air traffic rules for safe aircraft operations. These rules apply only when IFR conditions prevail. Minimum horizontal separations are a function of aircraft type, aircraft speed, availability of radar facilities, and factors such as severity

of wake vortices. To avoid wake turbulence and conform to prevailing FAA's ATC rules the model has default values for inter-arrival, inter-departure, and departure-arrival separations. If future technology is available to decrease the effect of wake turbulence and better air traffic control procedures the intrail separations could be decreased. This will definitely have an effect on the capacity of an airport facility. To study the effect of reduced intrail separations on the operations at airport, the model has feature a to edit these values for a particular run.

The default intrail separations being used by the model are shown in Tables 4.1, 4.2, and 4.3.

7) Buffer Data:

To take care of errors in attaining minimum separation distance a buffer time is required. The size of the buffer depends on the probability of violation that is acceptable [Harris, 1976]. Hence for inter-arrivals, interdepartures, and departure-arrival safe separations their respective probability of violations and standard deviation are specified.

8) Runway Length:

Length of the runway dictates the type of aircraft that could land or take-off. In planning airports, the runways should be long enough to accommodate the aircraft which requires the greatest length. Hence care has to be taken in specifying runway length, by keeping in view the aircraft selected for simulation. If the landing roll of any aircraft is greater than the runway length the simulation will terminate abruptly with an error message.

Table 4.1 Inter-Arrival Separations.

Lead Aircraft	Heavy (miles)	Large (miles)	Small (miles)
Heavy	4.0	5.0	6.0
Large	2.5	2.5	4.0
Small	2.5	2.5	2.5

Table 4.2 Departure-Arrival Separations.

Lead Aircraft (Departure)	Heavy (miles)	Large (miles)	Small (miles)
Heavy	2.0	2.0	2.0
Large	2.0	2.0	2.0
Small	2.0	2.0	2.0

Table 4.3 Inter-Departure Separations.

Lead Aircraft	Heavy (Sec)	Large (Sec)	Small (Sec)
Heavy	60	90	120
Large	60	60	90
Small	60	60	60

9) Runway Width:

Runway width is one of the several parameters dictating an aircraft turnoff time, the wider the runway the more time an aircraft takes to clear the runway. The user has a choice of three FAA standard runway widths, they are 30 m, 45 m, and 61 m wide.

10) Number of Exits:

The user has to specify the number of exits for a given length of runway for every application.

11) Type of Exits:

The user has a set of 10 exits to run an application. The exit types available for simulation are shown in Table 4.4. The exit types provided for simulation are a mix FAA's standard exit and REDIM generated exit types.

12) Exit Locations:

The location of exits has to be provided by the user. The last exit has to be located at the end of runway.

13) Airport Elevation:

Elevation of airport has an effect on the approach velocity of aircraft because of the variation of atmospheric pressure. The relationship between the elevation and the approach velocity is described in Section 4.4.2.

Table 4.4 Exit Types Available for Simulation in RUNSIM.

Exit Type	Max. Exiting Speed (m/s)	Critical Acft. Type
90° FAA Standard	8.00	None
45° FAA Standard	15.00	None
30° FAA Standard	26.00	None
30° Improved, FAA Standard	26.00	None
Wide Throat FAA Standard	17.00	None
30° REDIM	35.00	Heavy
20° REDIM	35.00	Heavy
30° REDIM	35.00	Large
20° REDIM	35.00	Large
30° REDIM	35.00	Small

14) Airport Mean Temperature:

The airport temperature influences the air density which in turn affects the density of air aircraft approach speed. Changes in aircraft approach speed are important as they affect the location of optimal turnoff exits through changes in the aircraft landing roll performance. The relationship between the temperature, density of air, and velocity of approach is described in Section 4.4.2.

15) Final Approach Length: It is the distance from the entry gate to the runway threshold. It should be atleast equal to the lagrest value in the ATC interarrival separation matrix. The shorter it's length the greater the capacity of the runway.

Changes in any one of the above input parameters is going to change the output values such as delays, weighted average runway occupancy time etc.

4.4 Computations

The simulation of this model involves several computational routines. The dynamics of the aircraft in the model is divided into the following phases: 1) Final Approach Phase, 2) Glide Phase, 3) Rolling Phase, 4) Turnoff Phase, and for departures 5) Takeoff Phase. From the above description it is clear that emphasis on computations is on arrivals and this coincides with the objective of this research, to simulate the arrivals and know the effect of high-speed exits on ROT and capacity. The following sections describe the

computations and assumptions made in the dynamics of the aircraft operations.

4.4.1 Final Approach Phase

The final approach path traversed by an arriving aircraft is the first segment for arrivals in the simulation model. As already mentioned in the Section 4.2 the approach velocity (V_{app}) of any arriving aircraft is constant from the entry fix to the runway threshold crossing point (i.e., common approach path). Hence the relationship between V_{app}, final approach travel time (T_{app}), and length of the final approach (L_{app}) is

$$T_{app} = \frac{L_{app}}{V_{app}} \qquad ...(4.4.1.1)$$

T_{app} is the time to traverse from approach gate to threshold of the runway. Each individual aircraft has a specific velocity of approach which is assigned as an attribute of the aircraft.

Estimation of V_{app} for every individual aircraft involves the following calculations:

1) Estimation of atmospheric conditions that affect V_{app}:

The V_{app} of any aircraft depends on the current atmospheric conditions. The following equations were used to calculate the variations in atmospheric conditions from those of an ideal standard atmosphere [Roskam, 1983].

$$\delta = \left(1 - \frac{0.0065}{288.2} \times A_{elev} \right)^{4.2561} \quad ...(4.4.1.2)$$

$$T_{tr} = \frac{(273 + T_{air})}{T_{zero}} \quad ...(4.4.1.3)$$

$$\gamma = \frac{\delta}{T_{tr}} \quad ...(4.4.1.4)$$

$$\rho = 1.225 \quad ...(4.4.1.5)$$

$$g = 9.81 m/sec^2 \quad ...(4.4.1.6)$$

where,

T_{tr} - True temperature ratio.

γ - Equivalent density ratio

ρ - Standard atmosphere sea level density (kg/cu.meter).

g - acceleration due to gravity (m/s^2).

2) Calculation of weight factor (WF):

WF is a candidate value from a normal distribution probability function with mean and standard deviation specified by the user in order to describe the random behavior of aircraft landing weights.

3) Calculation of aircraft weight (W):

The aircraft weight while landing at a particular airport is a value between maximum allowable landing weight (MALW) and operating empty weight (OEW) of an aircraft

and depends on WF as shown below:

$$W = OEW + (MALW - OEW) \times WF \qquad ...(4.4.1.7)$$

4) Calculation of approach velocity (V_{app}) [Roskam, 1983]:

$$V_{stall} = \sqrt{\frac{(2 \times W \times g)}{(\rho \times \gamma \times CL_{max} \times WA)}} \qquad ...(4.4.1.8)$$

$$V_{app} = 1.3 \times V_{stall} \qquad ...(4.4.1.9)$$

where V_{stall} is stalling velocity of the aircraft (m/s), CL_{max} is maximum coefficient of lift (dim), and WA is the wing area of the aircraft (sq. m.).

4.4.2 Glide Phase

After crossing the runway threshold, the aircraft flies over the runway to touchdown with inherent air drag deceleration due to a flare maneuver. The distance covered from threshold to touchdown is estimated by assuming a circular arc flare maneuver flown at constant load factor to transition from a constant rate of descent angle on final approach to flat flight path tangent to the runway. The air distance S_{air} is as shown below [Nicolai, 1976; Torenbeek, 1981; Roskam, 1986]

where V_{flare} is the flare speed (95% of V_{app}), γ is the effective descent flight path, H_{thres} is the threshold crossing altitude, which is a normally distributed variable whose mean

$$S_{air} = \frac{H_{thres}}{\gamma} + \frac{V^2_{flare} \times \gamma}{2 \times g \times (n_{flare} - 1)} \quad ...(4.4.2.1)$$

value depends on the type of aircraft and standard deviation. The flare load factor (n_{flare}) is set to 1.15g's and γ is a normal variate with mean of 3 degrees (0.0523 radians) and a standard deviation of 0.07 of mean to simulate a regular ILS approach flight path. The time consumed in the glide phase (TT_{air}) is a function of the touchdown location (S_{air}), the approach speed (V_{app}), and the touchdown speed (V_{td}) . Assuming a normal distribution for the aircraft touchdown location TT_{air} is given as below:

$$TT_{air} = \frac{2 \times S_{air}}{V_{app} + V_{td}} \quad ...(4.4.2.2)$$

4.4.3 Rolling Phase

The rolling phase of the arriving aircraft is best explained with the help of Figure 4.1. The rolling phase is BCDEF. AB is the glide phase. BC is an initial free roll where the aircraft rolls freely to simulate a pilots' time delay in applying braking mechanisms such as thrust reversers, spoilers or normal wheel braking. CDE is the braking phase. The braking sequence and the time it takes is the main contributor to ROT. To simulate the aircraft realistically the braking phase is again divided to CD and DE, where D is called Decision Point and its distance from the threshold is designated as S_{dp}.

To select the exit a candidate deceleration value (DEC_{land}) for every aircraft is evaluated

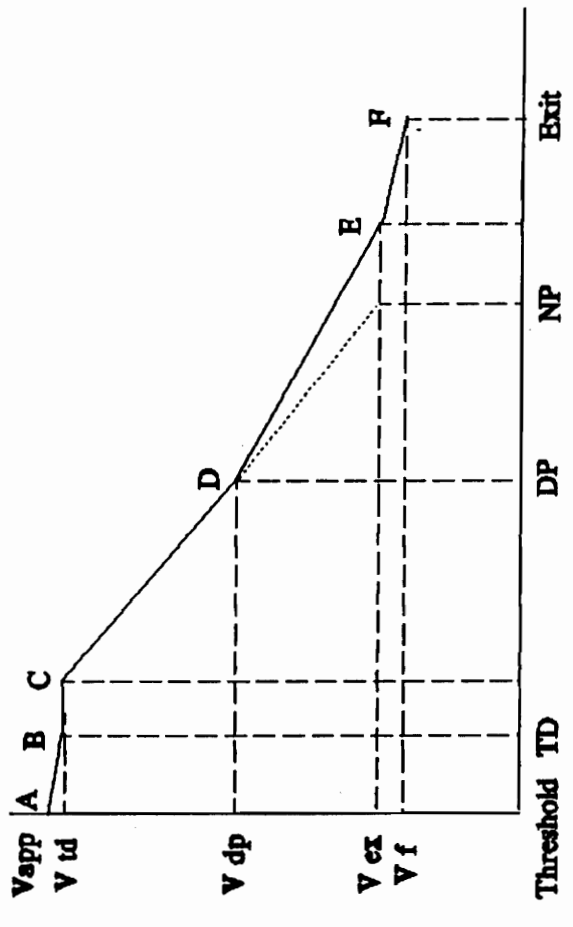

Figure 4.1 Velocity and Distance Profile of a Landing Aircraft on Runway.

using normally distributed probability function with standard deviation as 10% of the mean and within prescribed limits ($\mu \pm 3\sigma$). The nearest exit to where the aircraft can decelerate to achieve the exit speed with constant deceleration DEC_{land} is chosen for exiting. The point where the exit speed is achievable with respect to the threshold is designated as Nominal Point (NP).

Once the exit is selected the aircraft is allowed to decelerate at a constant deceleration DEC_{land} until it reaches point D, the Decision Point. Point D represents a point where the pilot judges and decides whether or not the exit taxiway can be negotiated according to its current aircraft state. The location of this point is a function of the pilots' eye position and his awareness of the runway/exit taxiway configuration. For simplicity a heuristic rule has been used where point D is located at $3V_{td}$ from NP. At D the deceleration is adjusted or a decision is made to change the deceleration so as to achieve the exit speed slightly before the exit is reached. The new deceleration DEC_{dec} at decision point will be to achieve the exit speed at threefourths the distance (i.e., point E) between the Decision Point D and the exit location F. The point E takes into consideration of the sight distance from the pilot and the exit location and the need to adjust the deceleration so as to achieve the exit speed comfortably. The location of E is related to human factors and further research has to be done to know the location and its variability and the current approximation. The aircraft then rolls freely with some inherent deceleration to the exit location F.

The travel times calculated during the rolling phase is shown below:

1) Calculation of travel time from B to C (TT_{BC}):

The TT_{BC} is part of the data base of every individual aircraft.

2) Calculation of travel time from C to D (TT_{CD}):

$$TT_{CD} = \frac{V_{td} - V_{dp}}{DEC_{land}} \qquad ...(4.4.3.1)$$

3) Calculation of travel time from D to E (TT_{DE}):

$$V_{dp} = \sqrt{V^2_{td} - 2 \times (3 \times V_{td}) \times DEC_{land}} \qquad ...(4.4.3.2)$$

$$DEC_{dec} = \frac{V^2_{ex} - V^2_{dp}}{2 \times (3/4) \times DF} \qquad ...(4.4.3.2)$$

$$TT_{DE} = \frac{V_{ex} - V_{dp}}{DEC_{dec}} \qquad ...(4.4.3.4)$$

Where V_{dp} is the velocity at the decision point D, and V_{ex} is the exit speed.

4) Calculation of travel time from E to F (TT_{EF}):

During the final free roll the deceleration due to friction D_{fr} is taken to be 0.375 m/s^2 [Wong, 1983] and because of it the final velocity V_f at exit will be less than V_{ex}. But never less than minimum exit velocity V_{min} (8 m/s).

$$V_f = \sqrt{V^2_{ex} - 2 \times DEC_{fr} \times EF} \qquad ...(4.4.3.5)$$

$$TT_{EF} = \frac{V_{ex} - V_f}{DEC_{fr}} \qquad ...(4.4.3.6)$$

4.4.4. Turnoff Phase

The aircraft in the turnoff phase travels with an initial velocity V_f and rolls freely with some inherent deceleration until it clears the runway. The aircraft is considered that it cleared the runway when its wingtip clears the edge of the runway as shown in Figure 4.2.

The turnoff time (TOT) for the aircraft is computed by using the data stored in a data file. This data is obtained from a continuous simulation program shown in Appendix D [Trani, 1991]. That program uses runway width, exit speed, exit geometry, aircraft wingspan as input variables to simulate the motion of the aircraft along the centerline of the exit and calculates simultaneously the coordinates of the wing tip and the time of travel until it clears the runway edge. Mathematically it is as shown below:

$$TOT_{V_{exit_i}} = f(\ RWY\ WIDTH,\ AIRCRAFT\ WINGSPAN\ ) \qquad ...(4.4.4.1)$$

A population of 34 aircraft representing the entire range (i.e., 4 TERP categories) of aircraft were simulated for every exit type in question. Simulation runs were performed for every exit type from minimum exit velocity V_{min} allowed until the maximum allowable exit speed V_{ex} at some prespecified intervals. Data was collected for each run mentioned above and multiple linear regressional analyses was performed to calculate turnoff time with wing span and runway width as explanatory variables. The intercept and the coefficients for every exit type and velocities are stored in data file (the data is shown in Appendix C). The Runway Simulation Model retrieves this data during the execution of the program and evaluates TOT by interpolation.

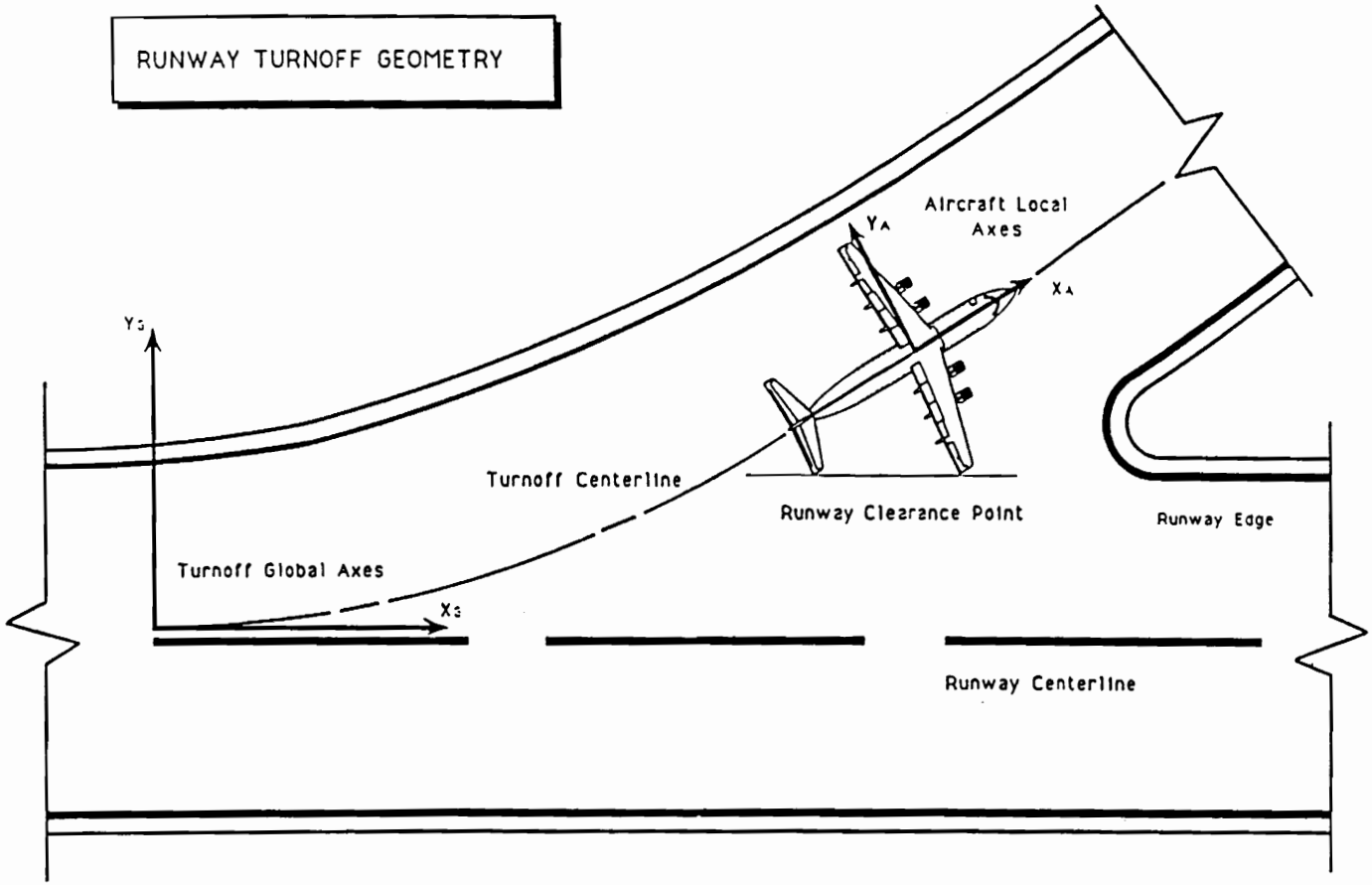

Figure 4.2 Runway Clearance Point [Trani, et al., 1990].

4.4.5 Takeoff Phase

The departing aircraft takeoff time is a constant value for each type of the aircraft. The values are shown in Table 4.5. The reason for the constant takeoff time is that little variation is usually seen in takeoff operation as related to landings.

4.5 Program Logic

The aircraft in the model are generated according to the user selected interarrival distribution for arrivals and departures independently. Though the arrivals and departures are generated independently they interact with each other to process aircraft through the system. Before the program logic is described the modelling of the components of the system in SIMSCRIPT is described below:

1) The final approach is modelled as a **resource**. Currently a maximum of three aircraft can use the final approach { U.FINAL.APPROACH(1) = 3 } at any given time. This number could be changed depending on the length of the final approach and also the minimum intrail separations allowed.

2) The runway is also modelled as a **resource** and only one aircraft can use it at any given time { U.RUNWAY(1) = 1 }.

3) The aircraft are modelled as **temporary entities** which enter the system at their creation time and exit the system after being processed. They are assigned name, type, velocity of approach, type of operation, final approach travel time, runway occupancy time as

Table 4.5 Takeoff Time by Aircraft Category.

Aircraft Type	Takeoff Time (Sec)
Heavy	35
Large	30
Small	25

attributes.

4) The aircraft travel in the final approach is modelled as a **process**.

5) The landing roll for an arrival is modelled as a **process**.

6) The takeoff of an aircraft is also modelled as a **process**.

7) The waiting or queuing at stack and departure ramp are modelled as individual **processes**.

8) The creation of the arrivals and departures are modelled as **events** separately.

The logical flow of the program is shown in Figures 4.3 and 4.4 for arrivals and departures respectively.

4.5.1 Arrivals

The arrival aircraft at the time of creation are randomly assigned a particular aircraft name along with their attributes. The random assignment of aircraft names confirms to the user specified aircraft population and their creation percentages. For SIMSCRIPT the aircraft is identified by a 1-dimensional pointer array assigned to it at its creation time. Immediately after the creation the model checks the system for three conditions. Condition - I, if the final approach and runway is clear. Condition - II, if the final approach is occupied. Condition - III, if the final approach is clear but runway is occupied.

If Condition - I is satisfied, the aircraft departs the departure ramp immediately requests

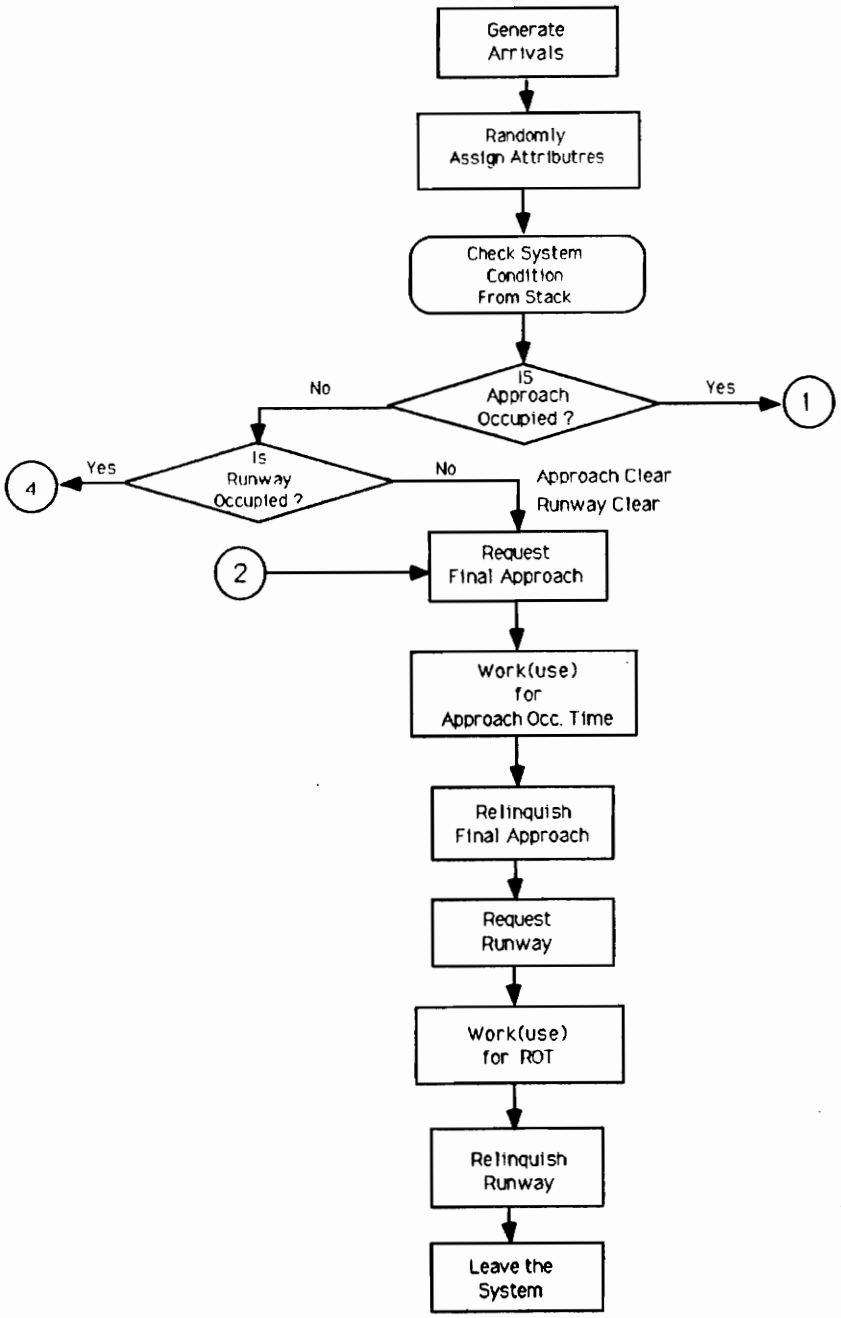

Figure 4.3 Flow Chart for Arrivals.

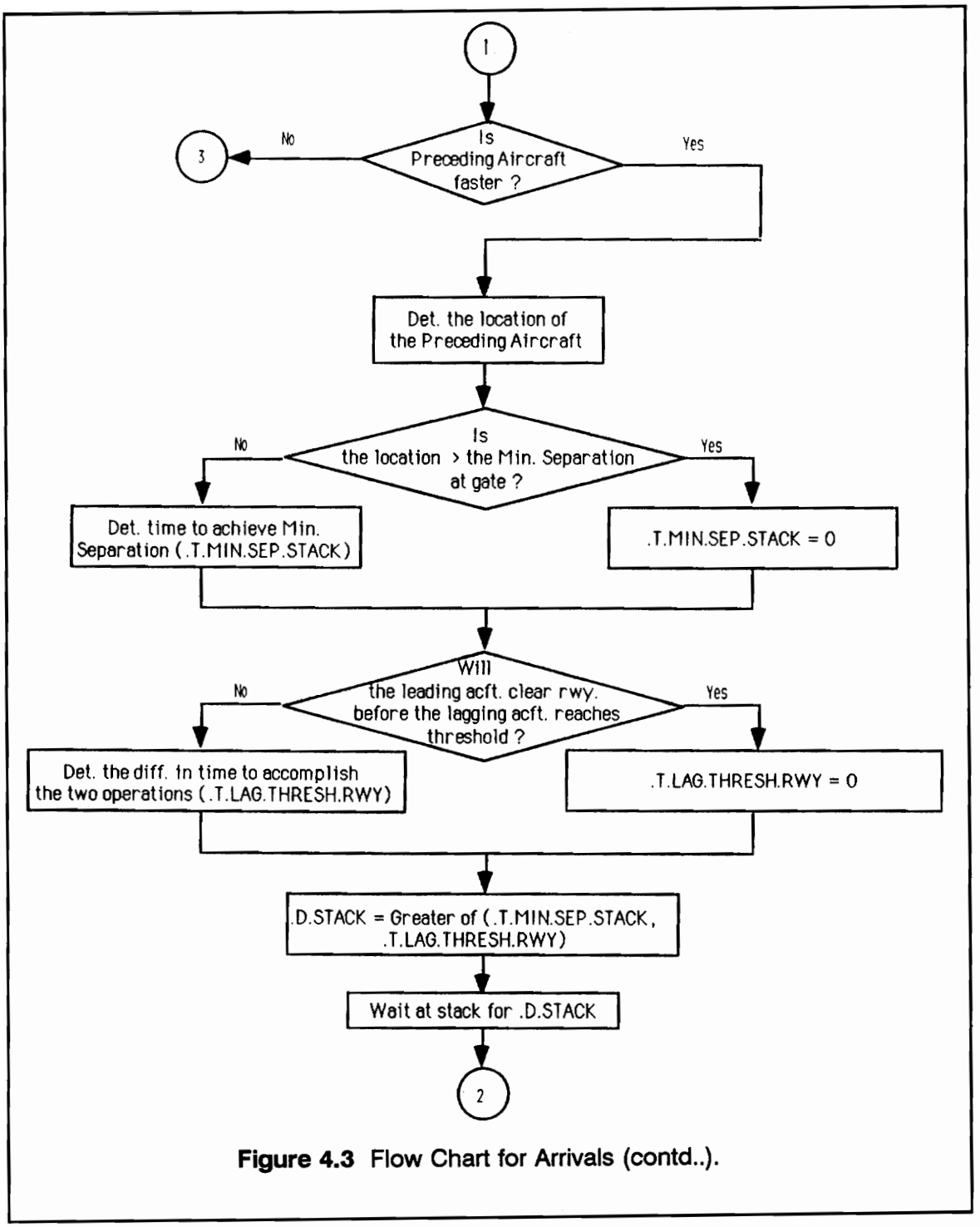

Figure 4.3 Flow Chart for Arrivals (contd..).

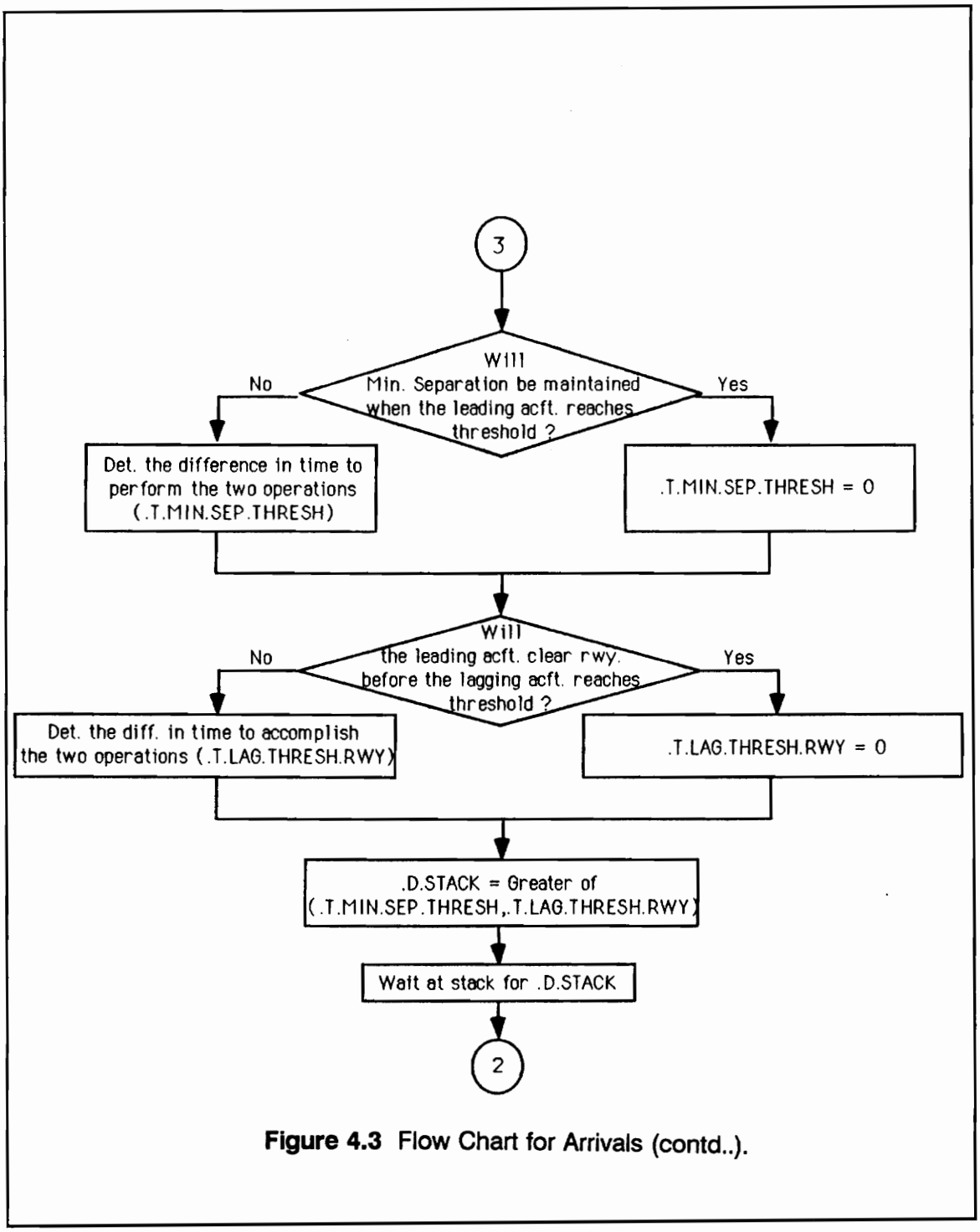

Figure 4.3 Flow Chart for Arrivals (contd..).

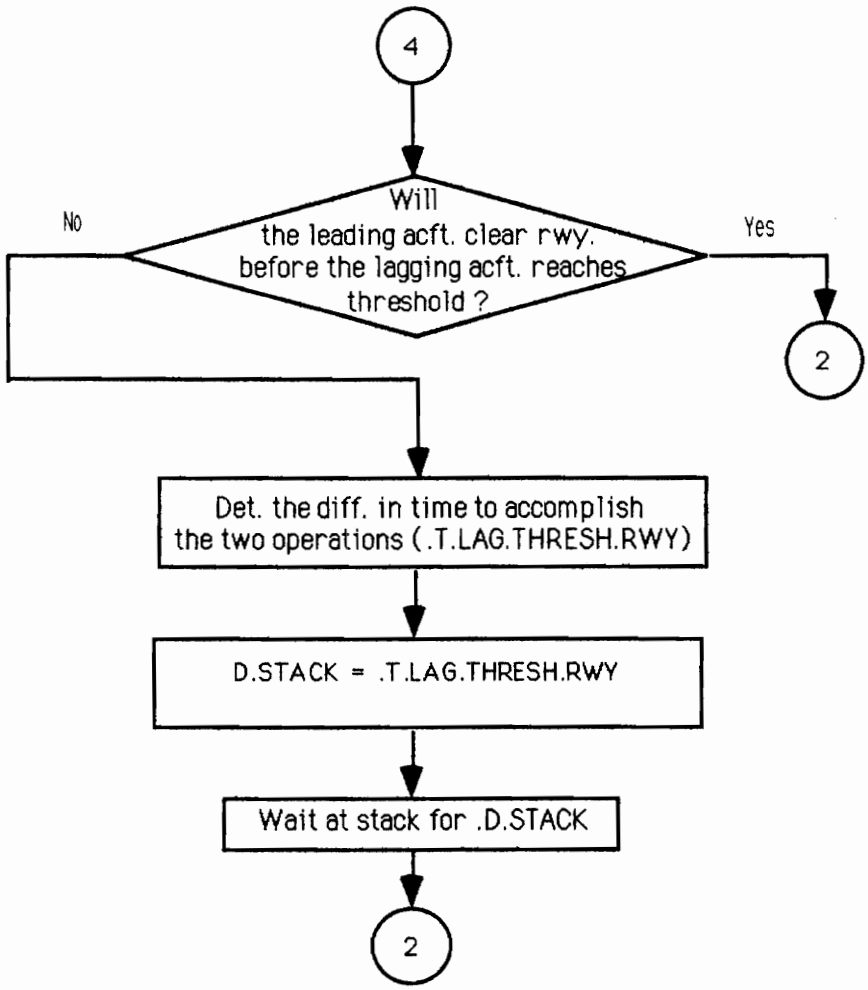

Figure 4.3 Flow Chart for Arrivals (contd..).

for the final approach(process) uses it (works) for approach travel time TT_{app} and leaves (relinquish) the final approach once it reaches the threshold of the runway. At the threshold it enters the runway (requests) and uses (works) for the duration of its runway occupancy time and then it leaves (relinquishes) the runway and exits the system. The program logic is so that runway is always available once it leaves the final approach.

If Condition - II is satisfied, a check is done to find the type of aircraft ahead of it in the final approach. This check is done for the minimum intrail separation criteria set plus a buffer. The speed of the preceding aircraft in the final approach is determined. If the leading aircraft is faster, the separation requirement is checked at the gate because this is where the two aircraft will be closest (opening case). If the leading aircraft is slower the separation distance (includes a buffer) must be met at the threshold (closing case) for the same reason.

If the leading aircraft is faster, the present location of the leading aircraft with respect to the entry gate is determined and checked against the separation distance required. If the location is farther than the required separation then the time to achieve minimum separation at stack (.T.MIN.SEP.STACK) is set to zero. If the location is nearer than the separation, the time to achieve the separation distance for the leading aircraft from its present location is determined and equated to .T.MIN.SEP.STACK. Then a check is done to find if the leading aircraft clears the runway by the time the lagging aircraft in the stack reaches threshold, if it leaves the stack right away. If the preceding aircraft does not clear the runway then the time difference to accomplish the two operations is calculated and

assigned to .T.LAG.THRESH.RWY. If the leading aircraft clears the runway then .T.LAG.THRESH.RWY is set to zero.

The delay at stack (.D.STACK) for the first aircraft in the stack (queue) will be the greater value of .T.MIN.SEP.STACK and .T.LAG.THRESH.RWY. After a delay of D.STACK that particular aircraft enters (requests) the final approach and behaves as if Condition - I has existed.

If the leading aircraft is slower, then a different kind of logic is used. It is checked that if the lagging aircraft enters the final approach, immediately, will the minimum separation (includes a buffer) be satisfied when the leading aircraft reaches the threshold. If the separation is going to be violated then the time difference for the lagging aircraft to reach the minimum separation from the threshold and the leading aircraft to reach the threshold is calculated and assigned to .T.MIN.SEP.THRESH. Later a check is again performed to find .T.LAG.THRESH.RWY. The greater value of .T.MIN.SEP.THRESH and .T.LAG.THRESH.RWY will be .D.STACK. After the passage of .D.STACK time units the first aircraft in the stack will enter the final approach as if Condition - I has existed.

If Condition - III is satisfied, it checks if the time remaining for aircraft on the runway (.T.REM.CLR.RWY) is greater than approach occupancy time (APP.OCC.TIME) of the aircraft in stack. If it is greater, then the aircraft in stack is let to wait for the difference of time in processing the events WAIT.TIME.STACK. If .T.REM.CLR RWY is less than .APP.OCC.TIME, then the aircraft proceeds as if Condition - I has existed.

4.5.2. Departures

The departures generated by the user defined interarrival distribution enter the system at the departure ramp. The aircraft is assigned its attributes pertaining to departure. The attributes assigned to it are the aircraft name, type, operation type (in this case "departure"), takeoff time.

Once it enters the ramp, a system check is performed to know the status of the system. The following four conditions are evaluated:

1) Condition - I, approach and runway clear.

2) Condition - II, approach is clear but runway is occupied.

3) Condition - III, approach is occupied and runway is clear.

4) Condition - IV, both approach and runway are occupied.

If Condition - I is satisfied, the aircraft enters (requests) the runway and uses (work) it for the duration of takeoff time and leaves (relinquishes) the runway and the system.

If Condition - II is satisfied, the time remaining for the aircraft (if it is landing) on the runway to clear the runway (T.REM.CLR.RWY) is determined or .T.MIN.SEP.TAKEOFF is calculated if the aircraft is enplaning for the time to achieve minimum interdeparture separation plus a buffer. If .T.REM.CLR.RWY (or .T.MIN.SEP.TAKEOFF) is greater than zero then the departure aircraft is delayed for that time units and then a system check is performed to know the new status of the system. If the delay was zero, then the aircraft

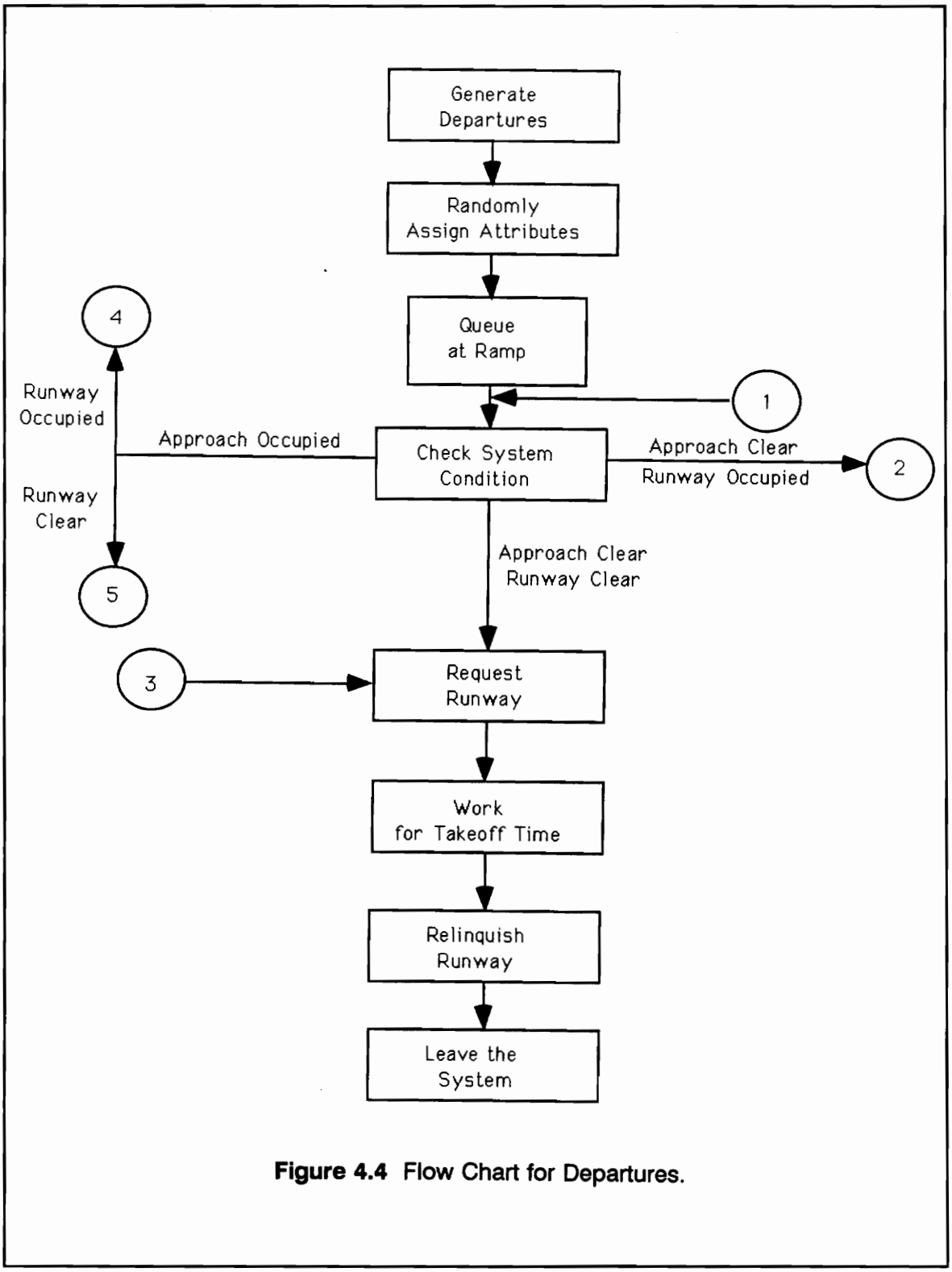

Figure 4.4 Flow Chart for Departures.

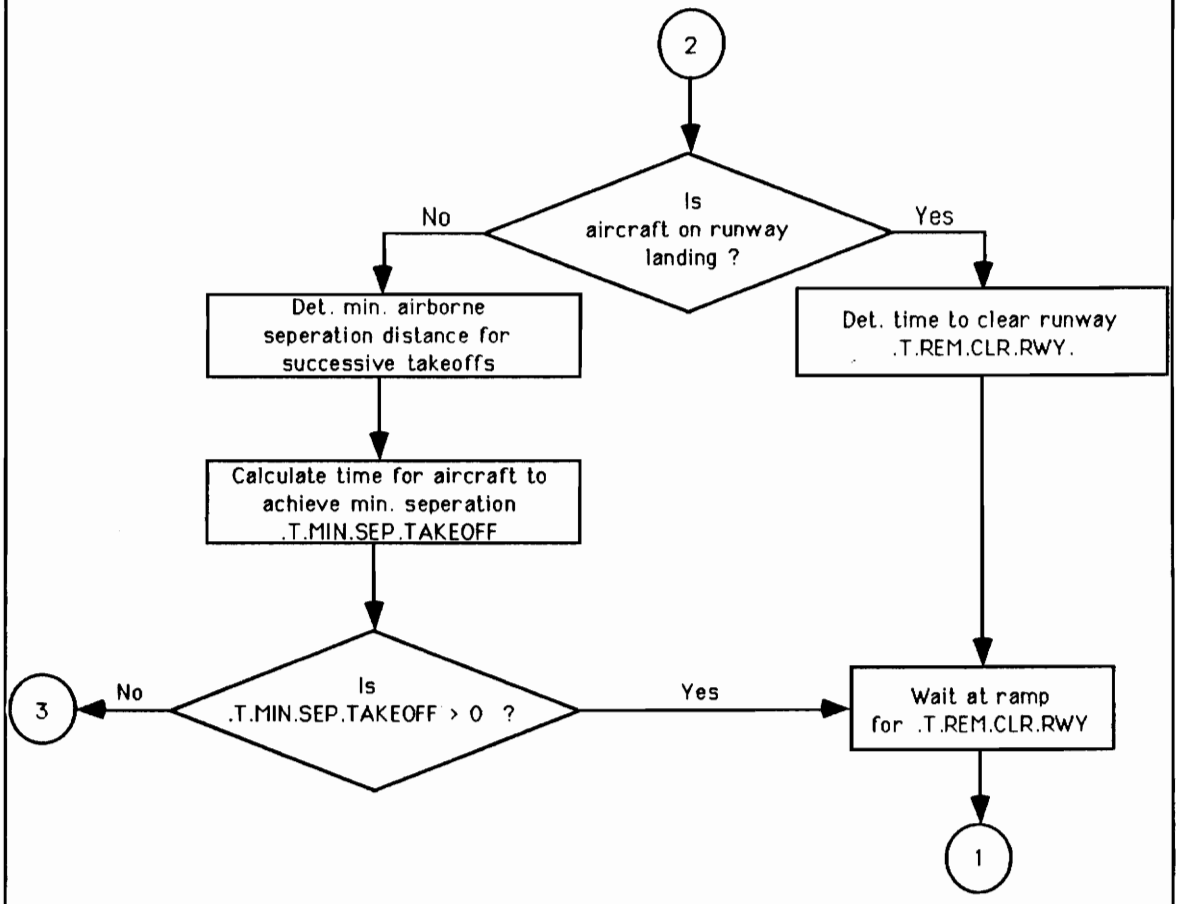

Figure 4.4 Flow Chart for Departures (contd..).

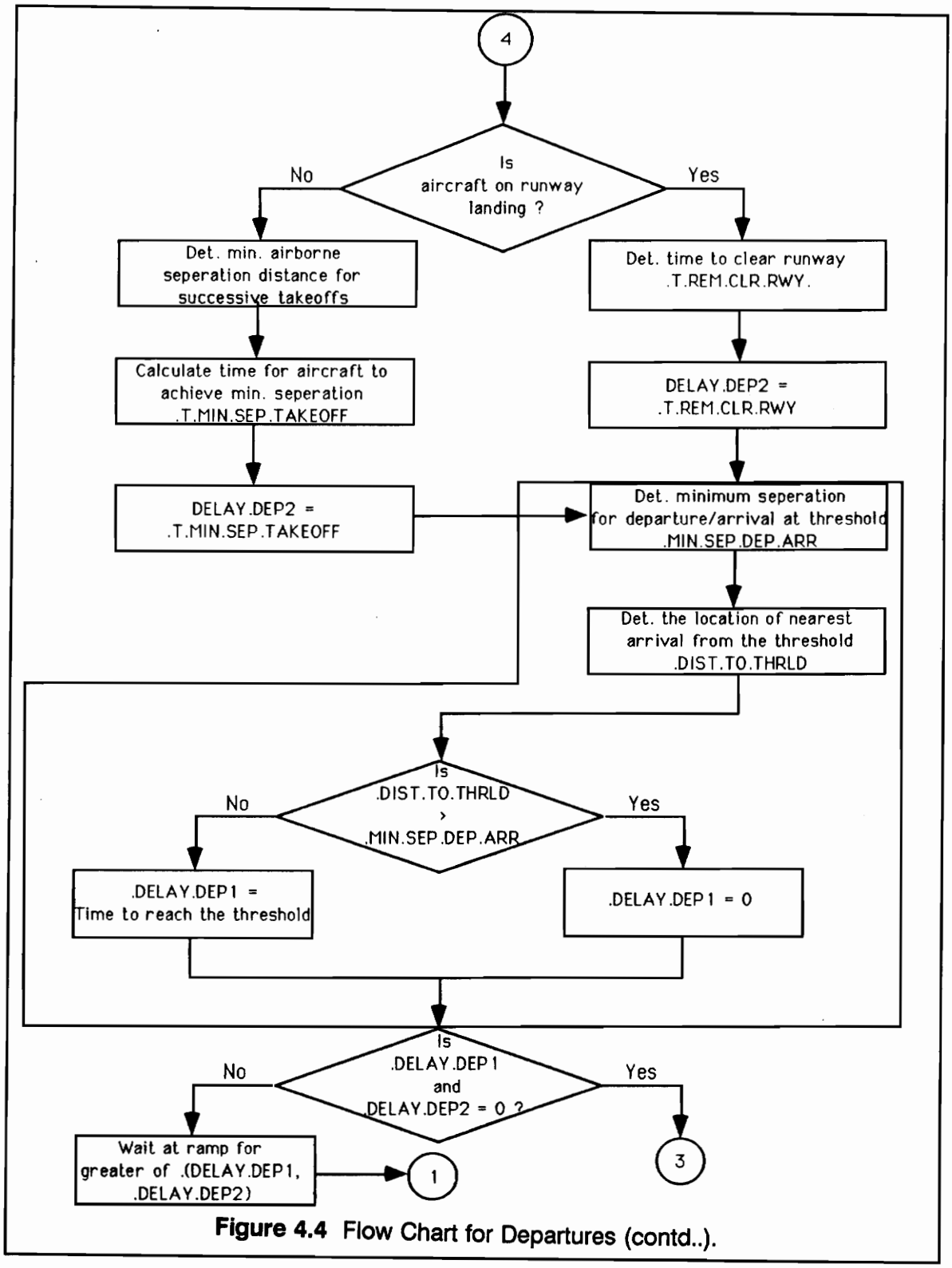

Figure 4.4 Flow Chart for Departures (contd..).

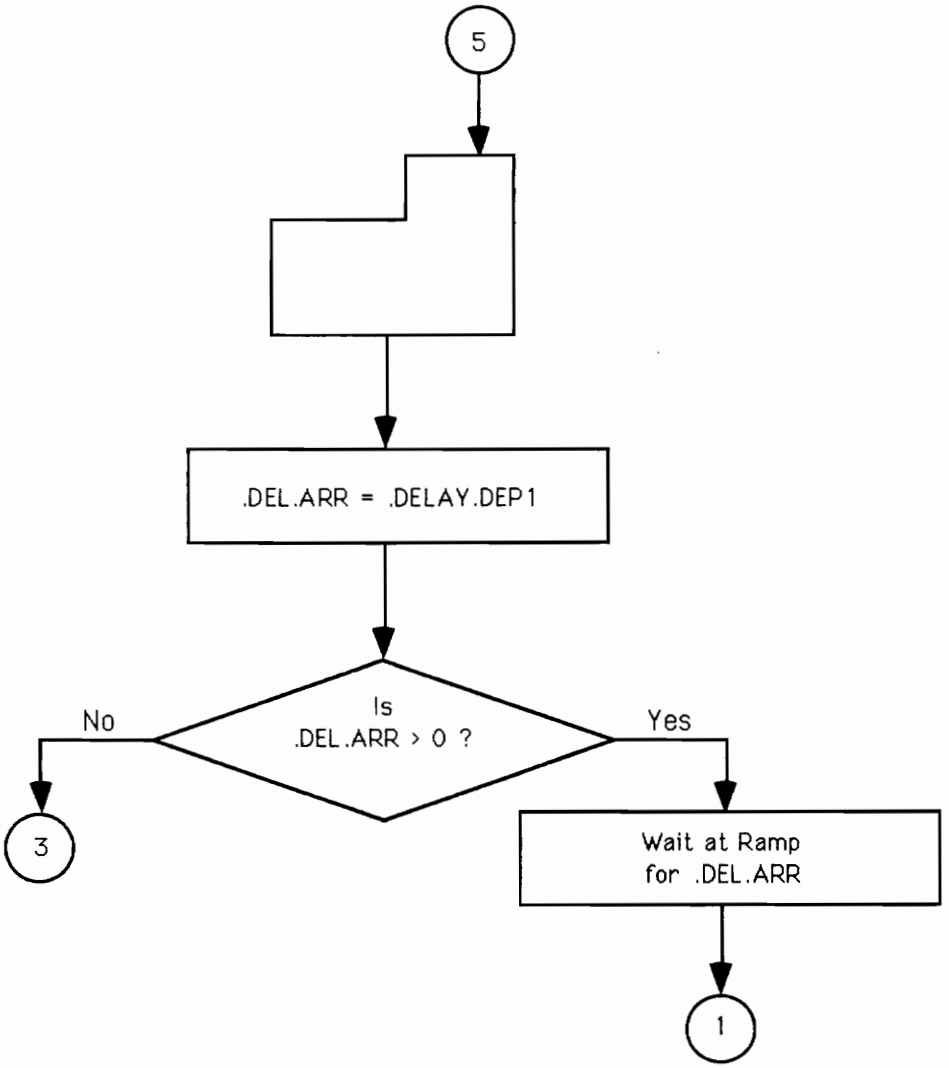

Figure 4.4 Flow Chart for Departures (contd..).

enters the runway and continues as if Condition - I has existed.

If Condition - III is satisfied, the current location of the aircraft in the final approach nearest to the threshold (at a given moment there could be more than one aircraft in the final approach, but the one nearest to threshold is critical) is determined. Its distance to the threshold (.DIST.TO.THRLD) is compared against the minimum departure-arrival separation plus a buffer. If .DIST.TO.THRLD is more than the separation required then the delay due to an arrival (.DEL.ARR) is set to zero and the departure aircraft proceeds as if Condition -I has existed. If .DIST.TO.THRLD is less than the separation, the time required for the arrival aircraft to reach runway from its present location to reach the runway is determined (.DEL.ARR). The departing aircraft is delayed for .DEL.ARR time units and then again a system check is performed to know the status of the system.

If Condition - IV is satisfied, the type of operation currently on the runway is evaluated. If it is a departure .T.MIN.SEP.TAKEOFF is evaluated, as performed in Condition - II, and equated to .DELAY.DEP1. If the aircraft on the runway is an arrival .T.REM.CLR.RWY is evaluated, as performed in Condition - II, and equated to .DELAY.DEP1. .DEL.ARR is evaluated and equated to .DELAY.DEP2. If .DELAY.DEP1 and .DELAY.DEP2 are equal to zero the departing aircraft proceeds as if Condition - I has existed. If .DELAY.DEP1 and .DELAY.DEP2 are not equal to zero the greater value of the above is taken as the delay for the departing aircraft on the ramp and after the passage of that delay time a system check is performed again to know the new state of the system.

enters the runway and continues as if Condition - I has existed.

If Condition - III is satisfied, the current location of the aircraft in the final approach nearest to the threshold (at a given moment there could be more than one aircraft in the final approach, but the one nearest to threshold is critical) is determined. Its distance to the threshold (.DIST.TO.THRLD) is compared against the minimum departure-arrival separation plus a buffer. If .DIST.TO.THRLD is more than the separation required then the delay due to an arrival (.DEL.ARR) is set to zero and the departure aircraft proceeds as if Condition -I has existed. If .DIST.TO.THRLD is less than the separation, the time required for the arrival aircraft to reach runway from its present location to reach the runway is determined (.DEL.ARR). The departing aircraft is delayed for .DEL.ARR time units and then again a system check is performed to know the status of the system.

If Condition - IV is satisfied, the type of operation currently on the runway is evaluated. If it is a departure .T.MIN.SEP.TAKEOFF is evaluated, as performed in Condition - II, and equated to .DELAY.DEP1. If the aircraft on the runway is an arrival .T.REM.CLR.RWY is evaluated, as performed in Condition - II, and equated to .DELAY.DEP1. .DEL.ARR is evaluated and equated to .DELAY.DEP2. If .DELAY.DEP1 and .DELAY.DEP2 are equal to zero the departing aircraft proceeds as if Condition - I has existed. If .DELAY.DEP1 and .DELAY.DEP2 are not equal to zero the greater value of the above is taken as the delay for the departing aircraft on the ramp and after the passage of that delay time a system check is performed again to know the new state of the system.

4.6 Program Description

This section describes some of the important SIMSCRIPT II.5 routines of the model, which is named as RUNSIM (Runway Simulation Model). The subroutines are described in alphabetical order, beginning with preamble and main routines, as they would appear in the source code (Appendix A).

4.6.1 Preamble

As mentioned in Chapter 3., preamble is the first part of a SIMSCRIPT II.5 program where all the global variables, entities, processes, events, and global statistics are declared. The following text is a part of the preamble that shows all the declarations of the model.

```
normally mode is undefined
resources
  every RUNWAY has                                          ...(4.6.1.1)
    a LENGTH.RWY
      define LENGTH.RWY  as a real variable
  every FINAL.APPROACH has                                  ...(4.6.1.2)
    a LENGTH.APP
      define LENGTH.APP as a real variable
processes include
  FINAL.APPROACH.OPERATION,
  LANDING.OPERATION,
  DEPARTURE.OPERATION,
  GENERATOR,
  GENERATOR.DEP,
  WAIT.AT.STACK,
  WAIT.AT.RAMP

events include
  CREATE.AIRCRAFT.ARR,
```

CREATE.AIRCRAFT.DEP

In the above part of the program the mode of all the variables are undefined, so as to enable to define every variable very specifically. RUNWAY (Eqn. 4.6.1.1) and FINAL.APPROACH (Eqn. 4.6.1.2) are declared as resources with their respective attributes. The processes and the events that are declared will be explained in their respective sections.

```
temporary entities
  every AIRCRAFT has                                              ...(4.6.1.3)
    a NAME,          " Name of the aircraft
    a V.APP,         " Velocity of approach
    a TYPE,          " Classification of aircraft by weight
    a APP.OCC.TIME,  " Time to travel the final approach
    a RWY.OCC.TIME,  " Runway Occupancy Time
    a TAKEOFF.TIME,  " Takeoff time i.e. to clear the runway
    a OPERATION      " to know if it is landing or departure

      define NAME as text variable
      define V.APP,
            APP.OCC.TIME,
            RWY.OCC.TIME,
            TAKEOFF.TIME as real variables
      define TYPE,
            OPERATION as integer variables
```

In the above part of the program, Eqn. (4.6.1.3), the AIRCRAFT is defined as a temporary entity with some attributes, which will be described when they are used for the first time in the program.

```
define ARR.ID,
      DEP.ID as 1-dimensional pointer arrays                      ...(4.6.1.4)

define minutes to mean units                                      ...(4.6.1.5)
define minute to mean units
define seconds to mean /60 minutes

tally TOT.ARR.DELAY as the sum of ARR.DELAY
tally TOT.DEP.DELAY as the sum of DEP.DELAY
```

CREATE.AIRCRAFT.DEP

In the above part of the program the mode of all the variables are undefined, so as to enable to define every variable very specifically. RUNWAY (Eqn. 4.6.1.1) and FINAL.APPROACH (Eqn. 4.6.1.2) are declared as resources with their respective attributes. The processes and the events that are declared will be explained in their respective sections.

```
temporary entities
  every AIRCRAFT has                                                    ...(4.6.1.3)
    a NAME,          " Name of the aircraft
    a V.APP,         " Velocity of approach
    a TYPE,          " Classification of aircraft by weight
    a APP.OCC.TIME,  " Time to travel the final approach
    a RWY.OCC.TIME,  " Runway Occupancy Time
    a TAKEOFF.TIME,  " Takeoff time i.e. to clear the runway
    a OPERATION      " to know if it is landing or departure

      define NAME as text variable
      define V.APP,
            APP.OCC.TIME,
            RWY.OCC.TIME,
            TAKEOFF.TIME as real variables
      define TYPE,
            OPERATION as integer variables
```

In the above part of the program, Eqn. (4.6.1.3), the AIRCRAFT is defined as a temporary entity with some attributes, which will be described when they are used for the first time in the program.

```
define ARR.ID,
      DEP.ID as 1-dimensional pointer arrays                            ...(4.6.1.4)

define minutes to mean units                                            ...(4.6.1.5)
define minute to mean units
define seconds to mean /60 minutes

tally TOT.ARR.DELAY as the sum of ARR.DELAY
tally TOT.DEP.DELAY as the sum of DEP.DELAY
```

```
define ARR.DELAY,
       DEP.DELAY as real variables

define ..LANDING to mean 1
define ..DEPARTURE to mean 2
define ..LARGE to mean 3
define ..HEAVY to mean 2
define ..SMALL to mean 1
```

In Eqn. 4.6.1.4, ARR.ID and DEP.ID are declared as one dimensional pointer arrays, these arrays are used to store in the memory each arriving and departing aircraft entity and their attributes. In Eqn. 4.6.1.5, minutes (or minute) is defined as the basic time unit of the simulation instead of days as the default time unit. The tally statements collect the total delay for arrivals and departures respectively. The define statements are used for implied variables for the program to be more.

4.6.2 Main

The model simulates only one runway operation (shown in Eqn. 4.6.2.1) and hence only one runway type and one of them only is created (shown in Eqn. 4.6.2.2). One type of final approach is created (shown in Eqn. 4.6.2.3) with maximum three identical of them (shown in Eqn. 4.6.2.4) could be used simultaneously. This is done so that at any given time with a given length of approach a maximum of three aircraft could follow each other towards runway. If the approach length is long enough to have more aircraft at a time with intrail separations taken into consideration the U.FINAL.APPROACH(1) may be increased to accommodate them.

```
create every RUNWAY(1)                                    ...(4.6.2.1)
```

```
let U.RUNWAY(1) = 1                                                    ...(4.6.2.2)

create every FINAL.APPROACH(1)                                         ...(4.6.2.3)
 let U.FINAL.APPROACH(1) = 3 "To allow a max. of 3 aircraft to use or enter  ...(4.6.2.4)
                        " final approach at any given time
```

The following block (between Eqn's. 4.6.2.5 and 4.6.2.8) is nested in a loop so as to enable the whole simulation program to perform iterations as many times (.NO.REPS) as specified by the user for any application run. If the application has arrivals then only GENERATOR is invoked, similarly for departures GENERATOR.DEP is invoked, if both or any of two are invoked then program control jumps to those processes and returns to line after Eqn. 4.6.2.1 only after all the process and event notices are cleared, i.e., only after a particular simulation iteration has come to an end. Eqn. 4.6.2.2 is invoked to reinitialize all the variables, that affect the simulation, for the next simulation iteration.

```
 for .ITER = 1 to .NO.REPS, do                                         ...(4.6.2.5)
    let RUN.NO. = .ITER
    if TOT.ARR.ACFT gt 0
      activate a GENERATOR now
    always
    if TOT.DEP.ACFT gt 0
       activate a GENERATOR.DEP now
    always
    start simulation                                                   ...(4.6.2.6)
    call setvt.r(7,2)
    call DO.BEEP
    call GLOBAL.STAT.OP giving .ITER
    call REPORT.GEN giving .ITER
    call RE.INITIALIZE                                                 ...(4.6.2.7)
  loop " .ITER = 1 to .NO.REPS                                         ...(4.6.2.8)
```

4.6.3 Routine ASSIGN.ARR.ACFT.NAME

This subroutine identifies the arrival aircraft and assigns its attributes.

```
  reserve .CUM.PERCENT.ARR as (NO.ARR.AIRCRAFT + 1)                    ...(4.6.3.1)
let .CUM.PERCENT.ARR(1) = 0
for .NO = 2 to (NO.ARR.AIRCRAFT + 1)  do
  let .CUM.PERCENT.ARR(.NO) = .CUM.PERCENT.ARR(.NO - 1) +
                              ARR.ACFT.PERCENT(.NO - 1)
loop
```

In Eqn. 4.3.6.1, the NO.ARR.AIRCRAFT represents the total number of aircraft that are chosen to constitute the arriving aircraft population mix . Variable .CUM.PERCENT.ARR is an array to used to store the cumulative arrival percentages of arrival aircraft, starting with zero in the first array to 100 in the last array, and this is done in the program that is part of the loop that is shown above.

In the program block shown below, the variable .ASSIGN is assigned a random integer between 1 and 100. This value is used to find in which cell of the .CUM.PERCENT.ARR it lies, this identification is done in Eqn. 4.6.3.3. After the identification the corresponding .IARR value is used to assign the attributes to the aircraft entity, which are done from Eqn's. 4.6.3.4 to 4.5.3.8. In Eqn. 4.6.3.9, the array .ROT.STAT is used to store the sum of the ROT's of each aircraft to calculate weighted average of ROT and its standard deviation.

```
let .ASSIGN = randi.f(1,100, 5)                                        ...(4.6.3.2)
for .IARR = 1 to NO.ARR.AIRCRAFT  do
  if .ASSIGN gt .CUM.PERCENT.ARR(.IARR) and
                          .ASSIGN le .CUM.PERCENT.ARR(.IARR + 1)       ...(4.6.3.3)
    "Assign the name and the other attributes to the entity.
   NAME(ARR.ID(COUNT.ARR)) = ARR.ACFT.NAME(.IARR)                      ...(4.6.3.4)
   let .DEC = ARR.ACFT.DEC(.IARR)
   let TYPE(ARR.ID(COUNT.ARR)) = ARR.ACFT.TYPE(.IARR)                  ...(4.6.3.5)
   let .FREE.ROLL = ARR.ACFT.FREE.ROLL(.IARR)
   let .WING.SPAN = ARR.ACFT.WING.SPAN(.IARR)
   call DET.VAPP giving .IARR yielding .VELAPP
   let V.APP(ARR.ID(COUNT.ARR)) = .VELAPP                              ...(4.6.3.6)
   call DET.DIST.AIR giving .IARR, .VELAPP yielding .DAIR
   call DET.APP.OCC.TIME giving .VELAPP  yielding .APP.OCC.TIME
   let APP.OCC.TIME(ARR.ID(COUNT.ARR)) = .APP.OCC.TIME                 ...(4.6.3.7)
   call DET.RWY.OCC.TIME giving .DAIR, .DEC, .FREE.ROLL,
```

```
                    .VELAPP, .WING.SPAN, COUNT.ARR yielding .ROT,
                     .EXI
    let RWY.OCC.TIME(ARR.ID(COUNT.ARR)) = .ROT                          ...(4.6.3.8)
    let ROT.STAT(.IARR) = ROT.STAT(.IARR) + .ROT                        ...(4.6.3.9)
    let NO.IDENT.ARR(.IARR) = NO.IDENT.ARR(.IARR) + 1 " Collecting data
                " to know number of times similar aircraft were
                " generated.
    let EXIT.ASSIGNED(COUNT.ARR) = .EXI " storing data on exits assigned
                                 " to the landing aircraft
    let EXIT.ASSIGN(.IARR, .EXI) = EXIT.ASSIGN(.IARR, .EXI) + 1
                " Collecting data on number of times similar aircraft
                " is assigned to different exits
  always
 loop  " till the correct aircraft is identified
```

4.6.4 Routine ASSIGN.DEP.ACFT.NAME

This routine below is very similar to that described in the previous section except for that this routine applies to departing aircraft.

```
reserve .CUM.PERCENT.DEP as (NO.DEP.AIRCRAFT + 1)
let .CUM.PERCENT.DEP(1) = 0
for .NO = 2 to (NO.DEP.AIRCRAFT + 1) do
   let .CUM.PERCENT.DEP(.NO) = .CUM.PERCENT.DEP(.NO - 1) +
                               DEP.ACFT.PERCENT(.NO -1)
loop " creating an array for cumulative percentages of departure aircraft
let .ASSIGN = randi.f(1,100,7) " for random assignment of aircraft names to
                             "  newly created aircraft entity.
 for .ID = 1 to NO.DEP.AIRCRAFT do
    if .ASSIGN gt .CUM.PERCENT.DEP(.ID) and
                .ASSIGN le .CUM.PERCENT.DEP(.ID + 1)
       NAME(DEP.ID(COUNT.DEP)) = DEP.ACFT.NAME(.ID)
       let TYPE(DEP.ID(COUNT.DEP)) = DEP.ACFT.TYPE(.ID)
       call DET.TAKEOFF.TIME given .ID yielding .TAKEOFF.TT
       let TAKEOFF.TIME(DEP.ID(COUNT.DEP)) = .TAKEOFF.TT
    always
 loop
```

4.6.5 Routine CHECK.PRECEDE.ACFT.CLR.RWY

This routine is used to check if the aircraft preceding another one in the final approach will clear the runway by the time the succeeding aircraft reaches the runway threshold. The calculation in latter part Eqn. 4.6.5.1 is used to convert time, which is in minutes and the seconds in hundredth fraction, to minutes and seconds to sixtieth fraction. Notice from Eqn. 4.6.5.2, that if the total time for the preceding aircraft to clear the runway from its present location is less than the time the succeeding aircraft takes to reach the runway threshold from its present location (stack) then the time difference to perform both the operations is stored as zero in (4.6.5.3). If the check in (4.6.5.2) is not satisfied then the difference in time to perform both the operations simultaneously is calculated and assigned to .R.T.LAG.THRLD.RWY in (4.6.5.4).

```
let .T.PRECEDE.ACFT.CLR.APP = APP.OCC.TIME(ARR.ID(COUNT.APP)) -
                              (trunc.f(time.v - T.DEP.STACK(COUNT.APP)) * 60 +
                               frac.f(time.v - T.DEP.STACK(COUNT.APP)) * 60 )      ...(4.6.5.1)
                    " Remaining time for preceding aircraft to clear approach
let .TOT.T.PRECEDE.ACFT.CLR.RWY = (.T.PRECEDE.ACFT.CLR.APP +
                                   RWY.OCC.TIME(ARR.ID(COUNT.APP)))
                    " Total time for the preceding aircraft to travel from
                    " its present position till it clears runway.
if .TOT.T.PRECEDE.ACFT.CLR.RWY le APP.OCC.TIME(ARR.ID(COUNT.APP + 1))  ...(4.6.5.2)

    .R.T.LAG.THRLD.RWY = 0 "Difference in time to perform the two operations    ...(4.6.5.3)
                           "1: Time for precede acft to clear rwy
                           "2: Time for lagging acft to reach threshold
  else
    .R.T.LAG.THRLD.RWY = (.TOT.T.PRECEDE.ACFT.CLR.RWY -
                          APP.OCC.TIME(ARR.ID(COUNT.APP + 1)))                  ...(4.6.5.4)
always
```

4.6.6 Event CREATE.AIRCRAFT.ARR

When this event is activated from GENERATOR process it creates a temporary entity AIRCRAFT (Eqn. 4.6.6.2) and it is assigned to an unique memory location by calling ARR.ID(COUNT.ARR) in Eqn. 4.6.6.2. The memory location is unique to the newly created entity because as each time a new entity (aircraft) is created the global pointer variable COUNT.ARR., is incremented by 1 (Eqn. 4.6.6.1) and this way the entity is stored in ARR.ID with a different identifier. The creation time of every arrival is collected in Eqn. 4.6.6.3. To identify that this aircraft entity is arriving, its attribute is equated to an implied global variable ..LANDING (its value being 1) in Eqn. 4.6.6.4. If Eqn. 4.6.6.5, is satisfied it implies that the newly created aircraft is the first one in the stack and hence it can proceed checking the system from stack.

```
if  COUNT.ARR lt TOT.ARR.ACFT
    COUNT.ARR = COUNT.ARR + 1                              ...(4.6.6.1)
    create an AIRCRAFT called ARR.ID(COUNT.ARR)            ...(4.6.6.2)
    T.ENTER.STACK(COUNT.ARR) = TIME.V                      ...(4.6.6.3)
    let OPERATION(ARR.ID(COUNT.ARR)) = ..LANDING           ...(4.6.6.4)
    call ASSIGN.ARR.ACFT.NAME
    if COUNT.ARR  le (COUNT.APP + 1)                       ...(4.5.6.5)
      call SYS.CHECK.AT.STACK
    always
 else
    return
 always
```

4.6.7 Event CREATE.AIRCRAFT.DEP

This event routine is very similar to the previous one explained except that this deals with

departures.

```
if COUNT.DEP lt TOT.DEP.ACFT
  let COUNT.DEP = COUNT.DEP + 1
  create an AIRCRAFT called DEP.ID(COUNT.DEP)
  let T.ENTER.RAMP(COUNT.DEP) = time.v
  OPERATION(DEP.ID(COUNT.DEP)) = ..DEPARTURE
  call ASSIGN.DEP.ACFT.NAME
  if COUNT.DEP le (COUNT.T.OFF + 1)
    call SYS.CHECK.AT.THRLD
  always
else
  return
always
```

4.6.8 Process DEPARTURE.OPERATION

This process requests the resource runway, in Eqn. 4.6.8.1, and then uses (work) the runway for a period of runway occupancy time of the aircraft which is using the runway. The pointer variable COUNT.T.OFF, in Eqn. 4.6.8.2, is used as a counter to know the sequence number of aircraft that is being or has been processed in the arrival flow. It also helps in retrieving the data related to the aircraft entity that is being processed, as it could be seen in Eqn. 4.6.8.3, where the OCCUPANT.OPERATION variable is given a value to indicate that a departing aircraft is currently using the runway. In Eqn. 4.6.8.4, COUNT.DEP is a pointer variable that indicates the number of departing aircraft that have been created until the current simulation time. The conditional check in Eqn. 4.6.8.4 is done to initiate the first aircraft in the departing queue at the threshold to start checking the system this is a process towards finding a gap for takeoff. The work statement in Eqn. 4.6.8.5 depicts the occupancy of runway by the departing aircraft for the duration of TAKEOFF.TIME. After period of TAKEOFF.TIME the process notice relinquishes, in

Eqn. 4.6.8.6, the runway resource depicting the completion of takeoff of the aircraft.

```
if COUNT.T.OFF  lt  TOT.DEP.ACFT
   request 1 RUNWAY(1)                                                    ...(4.6.8.1)
   if COUNT.T.OFF lt COUNT.DEP
      let COUNT.T.OFF = COUNT.T.OFF + 1                                   ...(4.6.8.2)
      let OCCUPANT.OPERATION = OPERATION(DEP.ID(COUNT.T.OFF))             ...(4.6.8.3)
      let T.DEP.RAMP(COUNT.T.OFF) = TIME.V
      if COUNT.T.OFF ne COUNT.DEP and (COUNT.T.OFF + 1) le COUNT.DEP      ...(4.6.8.4)
         call SYS.CHECK.AT.THRLD
      always " (COUNT.T.OFF + 1) le COUNT.DEP
      work TAKEOFF.TIME(DEP.ID(COUNT.T.OFF)) seconds                      ...(4.6.8.5)
   always "COUNT.T.OFF lt COUNT.DEP
   relinquish 1 RUNWAY(1)                                                 ...(4.6.8.6)
   if .ACFT.ID eq TOT.DEP.ACFT
      END.DEP.SIM = time.v
   always
   if COUNT.T.OFF ne COUNT.DEP and ( COUNT.T.OFF + 1) le COUNT.DEP        ...(4.6.8.3)
      call SYS.CHECK.AT.THRLD
   always
 always " COUNT.T.OFF lt TOT.DEP.ACFT
```

4.6.9 Routine DET.DEL.ARR

This routine determines the delay to a departing aircraft due to an aircraft in final approach. To determine the types of aircraft of the first one in the departing queue and the arriving one nearest to runway threshold, Eqn's. 4.6.9.1 and 4.6.9.3 are used. .DEP.ARR.SEP, in Eqn. 4.6.9.4, is an array which has the departure-arrival separation data. If the leading arriving aircraft is farther than the minimum separation between departure and arrival (.MIN.SEP.ARR.DEP), which is checked in Eqn. 4.6.9.5, then the delay due to the arriving aircraft (.R.DEL.ARR) is zero and if not, .R.DEL.ARR is equal to the time it takes to travel the remaining part of the final approach by the aircraft nearest to threshold.

```
let .TYPE.DEP = TYPE(DEP.ID(COUNT.T.OFF + 1))                                    ...(4.6.9.1)
let .LEAD.ACFT.APP = COUNT.LAND + 1                                              ...(4.6.9.2)
let .TYPE.ARR = TYPE(ARR.ID(.LEAD.ACFT.APP))                                     ...(4.6.9.3)
let .MIN.SEP.ARR.DEP = DEP.ARR.SEP(.TYPE.ARR, .TYPE.DEP) + INTDEP.ARR.BUFF ...(4.6.9.4)
let .TIME.SPENT.APP = trunc.f(time.v - T.DEP.STACK(.LEAD.ACFT.APP)) * 60 +
                      frac.f(time.v - T.DEP.STACK(.LEAD.ACFT.APP)) * 60
                        " seconds
let .TIME.REM.APP = APP.OCC.TIME(ARR.ID(.LEAD.ACFT.APP)) - .TIME.SPENT.APP
                  " Remaining time in final approach before reaching the
                  " runway by the arriving aircraft nearest to the runway
let .DIST.COVERED.APP = .TIME.SPENT.APP * V.APP(ARR.ID(.LEAD.ACFT.APP))
let .DIST.TO.THRLD = LENGTH.APP(FINAL.APPROACH) - .DIST.COVERED.APP
if .DIST.TO.THRLD ge .MIN.SEP.ARR.DEP                                            ...(4.6.9.5)
  .R.DEL.ARR = 0.0 " seconds
else
  .R.DEL.ARR = .TIME.REM.APP  "seconds
always
```

4.6.10 Routine DET.DEL.DEP

This routine determines the time to achieve the minimum interdeparture separation. SEP.T.TAKEOFFS, in Eqn. 4.6.10.1, is an array which has the interdeparture separation data. INT.DEP.BUFF is the buffer time for the interdeparture separation.

```
if COUNT.T.OFF lt TOT.DEP.ACFT
 let .PRECEDE.TYPE = TYPE(DEP.ID(COUNT.T.OFF))
 let .FLLWNG.TYPE = TYPE(DEP.ID(COUNT.T.OFF + 1))
 let .PRECEDE.DEP.STRT.TIME = T.DEP.RAMP(COUNT.T.OFF) " departure time of
                                        "of preceding aircraft
 let .MIN.T.SEP.TAKEOFF = SEP.T.TAKEOFFS(.FLLWNG.TYPE, .PRECEDE.TYPE) +
                          INTDEP.BUFF                                    ...(4.6.10.1)
 let .T.MIN.SEP.TAKEOFF = .MIN.T.SEP.TAKEOFF -
                          (trunc.f(time.v - .PRECEDE.DEP.STRT.TIME) * 60 +
                           frac.f(time.v - .PRECEDE.DEP.STRT.TIME) * 60) +
                           INTDEP.BUFF
                              "time to achieve minimum separation between
                              "successive takeoffs from present time
always
```

4.6.11 Routine DET.DIST.AIR

This routine calculates the distance travelled in the air from the time the arriving aircraft crosses runway threshold until it touches down on the runway. The mathematical calculations were discussed in Section 4.4.2.

```
let .GAMMA.NOM = 0.0523  "Descent flight path angle used to estimate the
                         "airborne portion of the landing path
" New procedure to estimate  gamma as a random variable
let .GAMMA.STD = .GAMMA.NOM * 0.07
let .MAX.GAMMA = .GAMMA.NOM + 3.5 * .GAMMA.STD
let .MIN.GAMMA = .GAMMA.NOM - 3.5 * .GAMMA.STD
" Choose a candidate flight path angle among those permissible
until .GAMMA le .MAX.GAMMA  and .GAMMA ge .MIN.GAMMA do
   let .GAMMA =  normal.f(.GAMMA.NOM, .GAMMA.STD , 9)
loop
" End of new procedure to estimate GAMMA (flight path angle)
let .NFLARE = 1.15   "Landing flare load factor (in g's)
let .ACC.GRA = 9.81  "Acceleration due to gravity (m/s-s)
if ARR.ACFT.TYPE(.DRR) eq ..HEAVY
  .HT.THRLD.NOM = 15.0   "Runway threshold crossimg altitiude(meters)
else
    if ARR.ACFT.TYPE(.DRR) EQ ..LARGE
        .HT.THRLD.NOM = 15.0
    else
        .HT.THRLD.NOM = 10.0
    always
always
" New procedure to estimate HTRLD as a normally distributed random variable
" HT.HTRLD.NOM is the nominal threshold crossing heigth
let .HT.THRLD.STD = .HT.THRLD.NOM * .10
let .MAX.HT.THRLD = .HT.THRLD.NOM + 3.5 * .HT.THRLD.STD
let .MIN.HT.THRLD = .HT.THRLD.NOM - 3.5 * .HT.THRLD.STD
until .HT.THRLD le .MAX.HT.THRLD and .HT.THRLD ge .MIN.HT.THRLD do
 let .HT.THRLD = normal.f(.HT.THRLD.NOM, .HT.THRLD.STD, 4)
loop
let .VFLARE = 0.95 * .VELAPP
let .DAIR = .HT.THRLD / .GAMMA + (.VFLARE ** 2 * .GAMMA) /
                                 ( 2 * .ACC.GRA * (.NFLARE - 1))
                  " Distance traveled in air after crossing threshold
                  " and before touchdown.
return
```

4.6.12 Routine DET.EXIT.SPEEDS

This routine determines the maximum exit speeds of each exit that was chosen by the user for simulation. The array EXIT.CHOICE, shown in Eqn. 4.6.12.1, has the exit numbers identifying the exits chosen by the user. The corresponding maximum exit speeds are stored in the array EXIT.SPEED.

```
for .ES = 1 to NO.EXITS do
  select case EXIT.CHOICE(.ES)                                    ...(4.6.12.1)
    case 1
     let EXIT.SPEED(.ES) = 8.00  " meters/sec
              " 90 deg FAA STANDARD
    case 2
     let EXIT.SPEED(.ES) = 15.00
             " 45 DEG FAA STANDARD
    case 3
     let EXIT.SPEED(.ES) = 26.00
           " 30 DEG FAA STANDARD
    case 4
     let EXIT.SPEED(.ES) = 26.00
            " 30 DEG IMPROVED
    case 5
     let EXIT.SPEED(.ES) = 17.00
           " WIDE THROAT
    case 6
     let EXIT.SPEED(.ES) = 35.00
          " 30 DEG REDIM - HEAVY(CRITICAL ACFT)
    case 7
     let EXIT.SPEED(.ES) = 35.00
          " 20 DEG REDIM - HEAVY(CRITICAL ACFT)
    case 8
     let EXIT.SPEED(.ES) = 35.00
         " 30 DEG REDIM - LARGE (CRITICAL ACFT)
    case 9
     let EXIT.SPEED(.ES) = 35.00
         " 20 DEG REDIM - LARGE(CRITICAL ACFT)
    case 10
     let EXIT.SPEED(.ES) = 35.00
          " 30 DEG REDIM - SMALL (CRITICAL ACFT)
    default
     let EXIT.SPEED(.ES) = 8.00
  endselect " EXIT.CHOICE(.ES)
```

```
loop " until .ES = NO.EXITS
```

4.6.13 Routine DET.RWY.OCC.TIME

This routine calculates the runway occupancy time of a landing aircraft. The deceleration during the landing roll is assumed to be normally distributed with mean and standard deviation of .DEC and .DEC.STD respectively. The deceleration is controlled within limits as shown in Eqn. 4.6.13.1. To calculate the travel time during final free roll, .TT.FINAL.ROLL, the velocity of the aircraft at the exit, .VEL.AT.EXIT, is required. Initially .VEL.AT.EXIT is calculated with .DEC.FINAL.ROLL (deceleration during the final free roll, a default value shown in Eqn. 4.6.13.2) in Eqn. 4.6.13.2. If that value is less than the minimum allowable exiting velocity, .MIN.EXIT.VEL, then .VEL.AT.EXIT assumes the minimum allowable value, i.e., .MIN.EXIT.VEL. Consequently a new or adjusted deceleration during final roll is calculated in Eqn. 4.6.13.4. The calculation of travel time during final free roll (TT.FINAL.ROLL) is calculated in Eqn. 4.6.13.5.

```
let .VEL.TD = (1.15/1.3) * .VELAPP
                    " Velocity at touchdown
let .DEC.STD = 0.10 * .DEC
                    " Standard deviation of the deceleration
let .MAX.DEC = .DEC + 3.0 * .DEC.STD
                    " Upper limit of the deceleration
let .MIN.DEC = .DEC - 3.0 * .DEC.STD
                    " Lower limit of the deceleration
" Choosing a candidate deceleration value between upper and lower limit
until .LAND.DEC le .MAX.DEC  and .LAND.DEC ge .MIN.DEC do          ...(4.6.13.1)
    let .LAND.DEC = normal.f(.DEC, .DEC.STD, 8)
                    " Deceleration after landing
loop
let .INIT.FREE.ROLL.DIST = .TT.FREE.ROLL * .VEL.TD  "Initial
                     " free Roll distance immediately after touchdown
call DET.SELECT.EXIT giving .RCARR, .VEL.TD, .LAND.DEC, .DAIR,
                .INIT.FREE.ROLL.DIST yielding .NOMINAL.PT, .EXIT.CHOSEN
```

```
let .DECISION.PT = (.NOMINAL.PT - 3 * .VEL.TD - .INIT.FREE.ROLL.DIST
                    - .DAIR)      " This is a point w.r.t. threshold
                   " where a decision is made to adjust the deceleration
                   " so as to achieve the exit velocity slightly ahead
                   " of the exit it has selected.
let .VEL.DECISION.PT = sqrt.f(.VEL.TD. ** 2 -
                         2.0 * .LAND.DEC * .DECISION.PT)
                   " Velocity at the decision point
let .DIST.DECISION.EXIT = EXIT.LOCATION(.EXIT.CHOSEN) - .DECISION.PT
                   " Distance between the decision point and the exit
                   " chosen.
let .DEC.AFTER.DECISION = (.VEL.DECISION.PT ** 2 -
                      EXIT.SPEED(.EXIT.CHOSEN) ** 2)/(2.0 * (3/4) *
                        .DIST.DECISION.EXIT)
                   "Deceleration after decision point till the threefourth
                   " of the distance between the decision point and exit
                   "and attaining the exit speed.
let .TT.TD = .DAIR / (( .VELAPP + .VEL.TD ) / 2.0)
                  " Travel time from threshold to touchdown
let .TT.TD.DECISION.PT = (.VEL.TD - .VEL.DECISION.PT) / .LAND.DEC
                  " Travel time from touchdown to decision point
let .TT.AFTER.DECISION = (.VEL.DECISION.PT - EXIT.SPEED(.EXIT.CHOSEN)) /
                         .DEC.AFTER.DECISION
                  " Travel time decision point till it achieves the
                  " exit speed.
let .DEC.FINAL.ROLL = 0.375  " Deceleration during final roll - m/s-s          ...(4.6.13.2)
y let .DIST.FINAL.ROLL = (1/4) * .DIST.DECISION.EXIT
                  " Distance to cover during final roll
let .MIN.EXIT.VEL = 8.0
                 " Minimum allowable exit velocity
let .MOD.EXIT.VEL = (EXIT.SPEED(.EXIT.CHOSEN)) ** 2 -
                    (2.0 * .DEC.FINAL.ROLL * .DIST.FINAL.ROLL)
if .MOD.EXIT.VEL gt 0
  .VEL.AT.EXIT = sqrt.f(.MOD.EXIT.VEL)                                        ...(4.6.13.3)
                  " Velocity of aircraft at the exit
                  " if it decelerates with .DEC.FINAL.ROLL
else
  .VEL.AT.EXIT = .MIN.EXIT.VEL
always
if .VEL.AT.EXIT lt .MIN.EXIT.VEL
  .VEL.AT.EXIT = .MIN.EXIT.VEL " if velocity achieved is less than the
                         " minimum exit prescribed velocity
                         " adjust velocity at exit to that min.
  let .DEC.FINAL.ROLL = ((EXIT.SPEED(.EXIT.CHOSEN)) ** 2 -
                    (.VEL.AT.EXIT) ** 2 ) / ( 2.0 * .DIST.FINAL.ROLL)          ...(4.6.13.4)
                   " Adjusted deceleration during final roll
always
let .TT.FINAL.ROLL = (EXIT.SPEED(.EXIT.CHOSEN) - .VEL.AT.EXIT) /
```

```
                    .DEC.FINAL.ROLL                                    ...(4.6.13.5)
                  " Travel time during final roll
call DET.TOT giving .EXIT.CHOSEN, .VEL.AT.EXIT, .WING.SPAN yielding
                .TT.EXIT
let .RWYOT = (.TT.TD + .TT.FREE.ROLL + .TT.TD.DECISION.PT +
                .TT.AFTER.DECISION + .TT.FINAL.ROLL + .TT.EXIT )
                  " Runway occupancy time
```

4.6.14 Routine DET.SELECT.EXIT

This routine selects the exit that is likely to be taken by a landing aircraft. It also calculates the nominal point of that aircraft on the runway. The selection process of the exit starts from the nearest exit to the farthest exit with respect to the threshold, and this is initiated by the for-do-loop starting in Eqn. 4.6.14.1. If the aircraft can decelerate to the maximum exit velocity at or before the exit location that is being evaluated then that is the exit that will be chosen to exit from runway, as shown in Eqn. 4.6.14.2 and the corresponding .NO.PT will be the nominal point of that aircraft. If none of the exits is suitable for an aircraft then an error message is shown on the screen (Eqn. 4.6.14.3).

```
for .IEX = 1 to NO.EXITS  do                                           ...(4.6.14.1)
  let .FINAL.VEL = EXIT.SPEED(.IEX)
  let .DIST.TRAVELED = (.VTD ** 2 - .FINAL.VEL ** 2)/(2.0 * .LDEC)
                  " Distance covered to decelerate from velocity at
                  " touchdown to the exit speed of the exit in question.
  let .NO.PT = .TDPT + .INFROLL + .DIST.TRAVELED
  if .NO.PT le EXIT.LOCATION(.IEX)                                     ...(4.6.14.2)
    .EXCHOSEN = .IEX
    return
  always
  if .IEX eq NO.EXITS
    call vclears.r
    call vgotoxy.r(18,0)
    call DO.BEEP
    call vbcolor.r(12)
    print 2 lines with NAME(ARR.ID(.IRR)) thus                         ...(4.6.14.3)
```

```
THE ************ AIRCRAFT CANNOT LAND:        INSUFFICIENT RUNWAY LENGTH.
 QUITTING THE SIMULATION.----------TYPE "QUIT" AT THE PROMPT
    call vbcolor.r(1)
  always
 loop
```

4.6.15 Routine DET.TAKEOFF.TIME

This routine determines the takeoff time of an aircraft. The takeoff time is related to three aircraft categories shown in the following block of equations.

```
if DEP.ACFT.TYPE(.RID) eq ..HEAVY
  .TAKEOFF.TTR =  35 "SECONDS
always
if DEP.ACFT.TYPE(.RID) eq ..LARGE
  .TAKEOFF.TTR = 30 "SECONDS
always
if DEP.ACFT.TYPE(.RID) eq ..SMALL
  .TAKEOFF.TTR = 25 "SECONDS
always
```

4.6.16 Routine DET.TOT

This routine determines the turnoff time (.TOT) of the exit that was chosen by an aircraft. Depending on the exit type (EXIT.CHOICE(.EXIT.NO)), and the speed (.ACFT.SPEED) at the entry turnoff point the intercept (.INPT), runway width coefficient (.RW.CF), and aircraft wing span coefficient (.ACFWS.CF) are calculated in Eqn's. 4.6.16.1, 4.6.16.2, and 4.6.16.3 respectively, by interpolation from the data obtained from turnoff time data file (TOT.DAT). The above calculated values are used in the regression equation in Eqn. 4.6.16.4 to calculate .TOT, which is a function of aircraft speed, runway width and

wingspan.

```
open unit 9 for input, name = 'TOT.DAT'
 use unit 9 for input
 let eof.v = 1
 while mode is alpha do
   start new input record
 loop
 until .READ.NO eq EXIT.CHOICE(.EXIT.NO), do
    skip 1 input record
    read .READ.NO
 loop
 read .NO.OF.SPEEDS
 reserve .SPEED.INC,
         .INTERCEPT,
         .RW.COEFF,
         .ACFWS.COEFF as .NO.OF.SPEEDS
 let .STEP = 1
 until .STEP gt .NO.OF.SPEEDS, do
   read .READ.NO, .SPEED.INC(.STEP), .INTERCEPT(.STEP), .RW.COEFF(.STEP),
        .ACFWS.COEFF(.STEP)
   let .STEP = .STEP + 1
 loop
 close unit 9
let .STEP = 1
for .STEP = 1 to ( .NO.OF.SPEEDS - 1), do
   if .ACFT.SPEED ge .SPEED.INC(.STEP) and
                          .ACFT.SPEED le .SPEED.INC(.STEP + 1)
      .FACT = (( .ACFT.SPEED - .SPEED.INC(.STEP)) /
               ( .SPEED.INC(.STEP + 1) - .SPEED.INC(.STEP)))
      let .INPT = .FACT * ( .INTERCEPT(.STEP + 1) - .INTERCEPT(.STEP)) +
                  .INTERCEPT(.STEP)                                          ...(4.6.16.1)
      let .RW.CF = .FACT * ( .RW.COEFF(.STEP + 1) - .RW.COEFF(.STEP)) +
                   .RW.COEFF(.STEP)                                          ...(4.6.16.2)
      let .ACFWS.CF = .FACT * (.ACFWS.COEFF(.STEP + 1) -
                      .ACFWS.COEFF(.STEP)) + .ACFWS.COEFF(.STEP)             ...(4.6.16.3)
      let .TOT = .INPT + .RW.CF * RWY.WIDTH + .ACFWS.CF * .WING.SPAN         ...(4.6.16.4)
      let .STEP = .NO.OF.SPEEDS
  always
loop
```

4.6.17 Routine DET.VAPP

This routine calculates the velocity of approach of an arriving aircraft. The mathematical

calculations are discussed in Section 4.4.1. The aircraft landing weight (.ACFT.WT) is a function of maximum allowable landing weight (ARR.ACFT.MALW), operating empty weight (ARR.ACFT.OEW), weight factor (.WT.FACTOR) of the arriving aircraft. Weight factor of an aircraft is normally distributed with an user defined mean (ARR.ACFT.WTF.MEAN) and standard deviation (ARR.ACFT.WTF.STD), as shown in Eqn. 4.6.17.1.

```
  let .TZERO = 288.2
  let .LAMBDA = .0065
  " Estimate the standard atmosphere values at given elevation
    let .THETAS = 1 - (.LAMBDA/.TZERO) * AIR.ELEV
    let .DELTAS = .THETAS ** 4.2561
  " Estimate true temperature ratio
    let .TR.TEMP.RATIO = (273 + AIR.TEMP) / 288.2
  " Compute the equivalent density ratio(.SIGMA)
    let .SIGMA = .DELTAS / .TR.TEMP.RATIO
  let .RHO = 1.225   " Standard atmosphere sea level density(kg/cu.meter)
  let .ACC.GRA = 9.81 "Acceleration due to gravity(m/sec-sec)
"Determination of the candidate aircraft weight while landing
    if  ARR.ACFT.WTF.STD(.VRR) eq 0
      ARR.ACFT.WTF.STD(.VRR) = .001
    always
    let .WT.FACTOR = normal.f(ARR.ACFT.WTF.MEAN(.VRR), ARR.ACFT.WTF.STD(.VRR),
                              2)                                    ...(4.6.17.1)
    let .ACFT.WT = ARR.ACFT.OEW(.VRR) + (ARR.ACFT.MALW(.VRR) -
                   ARR.ACFT.OEW(.VRR)) * .WT.FACTOR
  "Calculate velocity of approach
   let .V.STALL = sqrt.f(2 * .ACFT.WT * .ACC.GRA /
                         (.RHO * .SIGMA * ARR.ACFT.CL.MAX(.VRR) *
                           ARR.ACFT.WING.AREA(.VRR)))
              "Stalling velocity
   let .VELOCITY = 1.3 * .V.STALL
```

4.6.18 Process FINAL.APPROACH.OPERATION

This process routine simulates the aircraft flight from stack to the runway threshold. The aircraft that is first in the stack requests for entering the final approach (resource) in Eqn.

4.6.17.1. COUNT.APP, in Eqn. 4.6.18.2, is a pointer variable which keeps track of the aircraft sequence number that has entered the final approach. It also helps in retrieving the attributes of the arriving aircraft entity as shown in Eqn. 4.6.18.4. The aircraft occupies the final approach for the duration of its approach travel time (APP.OCC.TIME). After that duration the FINAL.APPROACH.OPERATION process relinquishes, in Eqn. 4.6.18.5, the final approach depicting the aircraft leaving the final approach and immediately landing activity is started by activating the LANDING.OPERATION process. The conditional check in Eqn. 4.6.18.3 is performed to activate the system checking activity at the stack for the first aircraft in the queue at the approach gate.

```
request 1 FINAL.APPROACH(1)                                    ...(4.6.18.1)
let COUNT.APP = COUNT.APP + 1                                  ...(4.6.18.2)
let T.DEP.STACK(COUNT.APP) = time.v
if (COUNT.APP + 1) le COUNT.ARR                                ...(4.6.18.3)
  call SYS.CHECK.AT.STACK
always
let ARR.DELAY = ( T.DEP.STACK(COUNT.APP) -
            T.ENTER.STACK(COUNT.APP))
work APP.OCC.TIME(ARR.ID(COUNT.APP)) seconds                   ...(4.6.18.4)
relinquish 1 FINAL.APPROACH(1) " after reaching threshold      ...(4.6.18.5)
activate a LANDING.OPERATION now
```

4.6.19 Process GENERATOR

This process is the first routine along with GENERATOR.DEP to be activated from main. Depending on the user defined arrival distribution this process activates the CREATE.AIRCRAFT.ARR event routine to create arrivals, and this activation continues till all the arrivals (TOT.ARR.ACFT) are created.

```
for .ll = 1 to TOT.ARR.ACFT do
  if ARR.DIST eq 1
```

```
    .R.ARR.DIST = poisson.f(ARR.DIST.MEAN, 1)
  else
    if ARR.DIST eq 2
      .R.ARR.DIST = exponential.f(ARR.DIST.MEAN, 1)
    else
      if ARR.DIST eq 3
        .R.ARR.DIST = uniform.f(ARR.DIST.MEAN, ARR.DIST.STD, 1)
      always
    always
  always
  activate  a CREATE.AIRCRAFT.ARR now
  wait .R.ARR.DIST seconds
loop " .II = 1 to TOT.ARR.ACFT
```

4.6.20 Process GENERATOR.DEP

This is similar to previous process except for that this is a generator for departures.

```
reserve DEP.ID,
        T.ENTER.RAMP,
        T.DEP.RAMP as TOT.DEP.ACFT
for .II = 1 to TOT.DEP.ACFT  do
"choosing the correct distribution for the interarrivals for departing aircraft
  if DEP.DIST eq 1
    .R.DEP.DIST = poisson.f(DEP.DIST.MEAN, 3)
  else
    if DEP.DIST eq 2
      .R.DEP.DIST = exponential.f(DEP.DIST.MEAN, 3)
    else
      if DEP.DIST eq 3
        .R.DEP.DIST = uniform.f(DEP.DIST.MEAN, DEP.DIST.STD, 3)
      always
    always
  always
  activate a  CREATE.AIRCRAFT.DEP now
  wait .R.DEP.DIST seconds
loop
```

4.6.21 Process LANDING.OPERATION

This process routine simulates the landing portion of an arrival. For an arriving aircraft the runway is always available once it reaches the threshold. In Eqn. 4.6.21.1, COUNT.LAND is a pointer variable used to know the number in the sequence of landings. It also helps in retrieving the attributes of the arriving aircraft entity. After a duration of RWY.OCC.TIME, which is the ROT of the current aircraft, the runway is relinquished. In Eqn. 4.6.21.2, the arriving aircraft (temporary entity) which has exited the runway is destroyed as it is no longer required for simulation.

```
request 1 RUNWAY(1)
let COUNT.LAND = COUNT.LAND + 1                                    ...(4.6.21.1)
let OCCUPANT.OPERATION = OPERATION(ARR.ID(COUNT.LAND))
let T.LANDING(COUNT.LAND) = time.v
let .ACFT.ID = COUNT.LAND
work RWY.OCC.TIME(ARR.ID(COUNT.LAND)) seconds
relinquish 1 RUNWAY(1)
if .ACFT.ID eq TOT.ARR.ACFT
   END.LAND.SIM = time.v
always
destroy this AIRCRAFT called ARR.ID(.ACFT.ID)                      ...(4.6.21.2)
```

4.6.22 Routine PRECEDE.FASTER.ACFT

This routine is used to analyze the "opening case" of an aircraft about to follow a faster aircraft in the final approach path. It determines the delay, if any, due to the faster preceding aircraft. The calculation of buffer distance, in Eqn. 4.6.22.1, is similar to the mathematical model proposed by Harris as discussed in Section 2.2. In Eqn. 4.6.22.2, T.DEP.STACK is an array which has information of the departing times of the aircraft from

the stack. In Eqn. 4.6.22.2, T.DEP.STACK is used to find the time it has travelled since it left the stack, .TT.APP.PRECEDE, by subtracting it from the current simulation time (time.v). .T.MIN.SEP.STACK, in Eqn. 4.6.22.3, is the time to achieve the minimum separation at approach gate (stack). The rest of the calculations and logic is as discussed in Section 4.5.1 of condition-II of the preceding faster aircraft case. To simulate the delay at the stack in order to satisfy the delay either due to minimum separation rule or single occupancy rule of runway the WAIT.AT.STACK is activated in Eqn. 4.6.22.4.

```
let .PREACFT = TYPE(ARR.ID(COUNT.APP))
let .FOACFT = TYPE(ARR.ID(COUNT.APP + 1))
let .BUFFER = INTARR.BUFF - SEP.DIST.ARR( .FOACFT, .PREACFT ) *
       ( 1 / V.APP(ARR.ID(COUNT.APP + 1)) - 1 / V.APP.(ARR.ID(COUNT.APP)))      ...(4.6.22.1)
        " IN SECONDS
if .BUFFER lt 0
    .BUFFER = 0 " buffer cannot be less than zero
always
let .MIN.SEP.BET.ARR = SEP.DIST.ARR( .FOACFT, .PREACFT ) + .BUFFER *
                       V.APP(ARR.ID(COUNT.APP + 1 ))
                    " minimum separation distance plus the buffer dist.
let .TT.APP.PRECEDE = trunc.f(time.v - T.DEP.STACK(COUNT.APP)) * 60 +
                    frac.f(time.v - T.DEP.STACK(COUNT.APP)) * 60              ...(4.6.22.2)
                       "Time elapsed since the preceding aircraft
                       " departed the stack
let .DIST.PRECEDE.STACK = V.APP(ARR.ID(COUNT.APP)) * .TT.APP.PRECEDE
                       "Distance from stack to the preceding
                       "aircraft in the final approach.
" check for minimum separation
if .MIN.SEP.BET.ARR le .DIST.PRECEDE.STACK
  .T.MIN.SEP.STACK = 0     "Time to achieve minimum separation from stack
else
  .T.MIN.SEP.STACK = (.MIN.SEP.BET.ARR - .DIST.PRECEDE.STACK)/
                    V.APP(ARR.ID(COUNT.APP)) "seconds                         ...(4.6.22.3)
always
"check if leading aircraft clears runway before lagging aircraft reaches
"threshold.
call CHECK.PRECEDE.ACFT.CLR.RWY yielding .T.LAG.TRHLD.RWY
if .T.MIN.SEP.STACK ge .T.LAG.TRHLD.RWY
  .DEL.STACK = .T.MIN.SEP.STACK "seconds
else
  .DEL.STACK = .T.LAG.TRHLD.RWY "seconds
```

```
always
let WAIT.TIME.STACK = .DEL.STACK
activate a WAIT.AT.STACK now                                    ...(4.6.22.4)
```

4.6.23 Routine PRECEDE.SLOWER.ACFT

This routine is similar to the previous routine except for that this is for a "closing arrival case", that is a fast aircraft following a slower aircraft in the final approach. In Eqn. 4.6.23.1, the time to achieve the minimum separation at the threshold (.T.MIN.SEP.THRLD) is compared with the time difference to accomplish, the preceding aircraft clearing the runway and the following aircraft reaching the threshold activities simultaneously. This comparison is performed as delay could be greater of either delay due to minimum separation rule or delay due to single occupancy rule of runway, but not both as aircraft can be delayed only at the stack and that too before it enters the final approach.

```
let .PFT = TYPE(ARR.ID(COUNT.APP))
let .FFT = TYPE(ARR.ID(COUNT.APP + 1))
let .MIN.SEP.BET.ARR = SEP.DIST.ARR (.FFT, .PFT) + INTARR.BUFF *
                       V.APP(ARR.ID(COUNT.APP))
let .T.REM.PRECEDE.THRLD = APP.OCC.TIME(ARR.ID(COUNT.APP)) -
                           (trunc.f(time.v - T.DEP.STACK(COUNT.APP)) * 60 +
                            frac.f(time.v - T.DEP.STACK(COUNT.APP)) * 60)
            " Time to reach the threshold by the preceding aircraft
            " from its present position.
let .DIST.LAG.ACFT = .T.REM.PRECEDE.THRLD * V.APP(ARR.ID(COUNT.APP + 1))
            "The distance the lagging aircraft will cover, if released
            "now, in the time the preceding aircraft reaches threshold.
let .DIST.LAG.THRLD = LENGTH.APP(FINAL.APPROACH) - .DIST.LAG.ACFT
            " Distance between the lagging aircraft and threshold after
            " .T.REM.PRECEDE.THRLD
if .MIN.SEP.BET.ARR le .DIST.LAG.THRLD                          ...(4.6.23.1)
  .T.MIN.SEP.THRLD = 0   " Difference of time between the preceding aircraft
                         " reaching threshold and lagging aircraft reaching
                         " minimum separation distance from threshold.
```

```
else
  .T.MIN.SEP.THRLD = .T.REM.PRECEDE.THRLD - (LENGTH.APP (FINAL.APPROACH) -
                    .MIN.SEP.BET.ARR) / V.APP(ARR.ID(COUNT.APP + 1))
always
call CHECK.PRECEDE.ACFT.CLR.RWY  yielding  .T.LAG.THRLD.RWY
if .T.MIN.SEP.THRLD ge .T.LAG.THRLD.RWY
  .D.STACK = .T.MIN.SEP.THRLD "seconds
else
  .D.STACK = .T.LAG.THRLD.RWY "seconds
always
if .D.STACK gt 0.0001
   let WAIT.TIME.STACK =  .D.STACK
   activate a WAIT.AT.STACK now
else
   activate a FINAL.APPROACH.OPERATION now
always
```

4.6.24 Routine RE.INITIALIZE

In this routine all the global variables, attributes of the runway resource (Eqn's. 4.6.24.1 and 4.6.24.2), and final approach resource (Eqn's. 4.6.24.3 and 4.6.24.4) that affect the simulation are reinitialized. Current time (time.v) is set back to zero. In Eqn. 4.6.24.5, all departure aircraft (temporary entities) are destroyed. All statistical arrays are also reintialized. This reintialization is done to facilitate another iteration to start.

```
let N.X.RUNWAY(1) = 0                                          ...(4.6.24.1)
let N.Q.RUNWAY(1) = 0                                          ...(4.6.24.2)
let N.X.FINAL.APPROACH(1) = 0                                  ...(4.6.24.3)
let N.Q.FINAL.APPROACH(1) = 0                                  ...(4.6.24.4)
let ARR.DELAY = 0
let DEP.DELAY = 0
let COUNT.ARR = 0
let COUNT.APP = 0
let COUNT.LAND = 0
let COUNT.DEP = 0
let COUNT.T.OFF = 0
let time.v = 0.0
for .I =  1 to TOT.DEP.ACFT do                                 ...(4.6.24.5)
   destroy the AIRCRAFT called DEP.ID(.I)
loop
```

```
release NO.IDENT.ARR(*),
        EXIT.ASSIGN(*,*),
        ROT.STAT(*)
reset totals of ARR.DELAY,
                DEP.DELAY
```

4.6.25 Routine STATUS.APP.CLR.RWY.OCC

This routine determines the delay, if any, due to the Condition-II that is discussed in Section 4.5.2. In Eqn. 4.6.25.1, it is checked if the aircraft on the runway is an arrival or a departure. Array T.LANDING has the times at which the arrivals cross the runway threshold, and this time is used in Eqn. 4.6.25.2 to calculate the remaining time of runway occupancy by the landing aircraft. To simulate the delay due to single occupancy rule process WAIT.AT.RAMP in Eqn. 4.6.25.3 is activated. If it is determined that there is no delay then the departure of the first aircraft in the departing queue is started by activating the process DEPARTURE.OPERATION in Eqn.4.6.25.4.

```
if OCCUPANT.OPERATION eq ..LANDING                                  ...(4.6.25.1)
  .T.REM.CLR.RWY = RWY.OCC.TIME(ARR.ID(COUNT.LAND)) -
                   (frac.f(time.v -T.LANDING(COUNT.LAND)) * 60 +
                     trunc.f(time.v - T.LANDING(COUNT.LAND)) * 60)  ...(4.6.25.2)
          "Remaining time to clear the runway by the landing aircraft
  if .T.REM.CLR.RWY le 0.00001
    WAIT.TIME.RAMP = 0.005
  else
    WAIT.TIME.RAMP = .T.REM.CLR.RWY
  always
  activate a WAIT.AT.RAMP now                                       ...(4.6.26.3)
else
  call DET.DEL.DEP yielding .T.MIN.SEP.TAKEOFFS
  if .T.MIN.SEP.TAKEOFFS gt 0
    WAIT.TIME.RAMP = .T.MIN.SEP.TAKEOFFS
    activate a WAIT.AT.RAMP now                                     ...(4.6.25.3)
  else
    activate a DEPARTURE.OPERATION now                              ...(4.6.25.4)
  always
```

```
always
```

4.6.26 Routine STATUS.APP.OCC

This routine is part of the Condition-II discussion in Section 4.5.1. This routine determines whether it is an "opening" or "closing" case. This is determined by comparing the velocity of approaches of the aircraft in question in Eqn. 4.6.26.1..

```
let .PRECEDE.SPEED = V.APP(ARR.ID(COUNT.APP))
let .OWN.SPEED = V.APP(ARR.ID(COUNT.APP + 1))
if .PRECEDE.SPEED ge .OWN.SPEED                                    ...(4.6.26.1)
  call PRECEDE.FASTER.ACFT      " preceding aircraft is faster
else
  call PRECEDE.SLOWER.ACFT      " preceding aircraft is slower
always
```

4.6.27 Routine APP.OCC.RWY.CLR

This routine performs the Condition-III discussion of Section 4.5.2.

```
call DET.DEL.ARR yielding .DEL.ARR
if .DEL.ARR gt 0  " minimum separation doesn't exist
  let WAIT.TIME.RAMP = .DEL.ARR
  activate a WAIT.AT.RAMP now
else
  activate a DEPARTURE.OPERATION now   " minimum separation exists
always
```

4.6.28 Routine APP.OCC.RWY.OCC

This routine determines delay, if any, due to the Condition-IV as discussed in Section

4.5.2. The following block of the program determines delay due to runway occupation (.DEL.DEP1).

```
if OCCUPANT.OPERATION eq ..LANDING
  .T.REM.CLR.RWY = RWY.OCC.TIME(ARR.ID(COUNT.LAND)) -
                   (trunc.f(time.v - T.LANDING(COUNT.LAND)) * 60 +
                    frac.f(time.v - T.LANDING(COUNT.LAND)) * 60 )
                      " Remaining time for the landing aircraft
                      " to clear runway.
  let .DEL.DEP1 = .T.REM.CLR.RWY
else
  call DET.DEL.DEP yielding .T.MIN.SEP.TAKEOFFS
  let  .DEL.DEP1 = .T.MIN.SEP.TAKEOFFS
always
```

Eqn. 4.6.28.1 is used to determine delay due to an arriving aircraft in final approach path, .DEL.DEP2. The WAIT.AT.RAMP is adjusted to a higher value, in Eqn. 4.6.28.2, in order the value to be within the accuracy of simulation clock because at very smaller values the clock cannot advance.

```
call DET.DEL.ARR yielding .DEL.DEP2                              (4.86.28.1)
if .DEL.DEP1 le 0 and .DEL.DEP2 le 0
   activate a DEPARTURE.OPERATION now
else
   if .DEL.DEP1 gt .DEL.DEP2
      if .DEL.DEP1 le 0.0001
         WAIT.TIME.RAMP = 0.05                                   (4.6.28.2)
      else
         WAIT.TIME.RAMP =  .DEL.DEP1
      always
      activate a WAIT.AT.RAMP now
   else
      if .DEL.DEP2 le 0.0001
         WAIT.TIME.RAMP = 0.05                                   (4.6.28.2)
      else
         WAIT.TIME.RAMP = .DEL.DEP2
      always
      activate a WAIT.AT.RAMP now
   always
always
```

4.6.29 Routine STATUS.RWY.OCC

This routine performs the discussion under Condition-II of Section 4.5.1.

```
let .T.REM.CLR.RWY = RWY.OCC.TIME(ARR.ID(COUNT.LAND)) -
                     (frac.f(time.v - T.LANDING(COUNT.LAND)) * 60 +
                      trunc.f(time.v - T.LANDING(COUNT.LAND)) * 60)
          " time remaining for the landing aircraft to clear runway
if .T.REM.CLR.RWY  gt APP.OCC.TIME(ARR.ID(COUNT.APP + 1))
  WAIT.TIME.STACK = .T.REM.CLR.RWY - APP.OCC.TIME(ARR.ID(COUNT.APP + 1))
  if WAIT.TIME.STACK le 0.0001
    WAIT.TIME.RAMP = 0.005
  always
  activate a WAIT.AT.STACK now
else
  activate a FINAL.APPROACH.OPERATION now
always
```

4.6.30 Routine SYS.CHECK.AT.STACK

This routine checks the three conditions at the stack which are discussed in Section 4.5.1. Eqn. 4.6.30.1 checks for Condition-II. Eqn. 4.6.30.2 checks for Condition-III. If none of the above conditions are satisfied, then it is implied that Condition-I is satisfied and hence process FINAL.APPROACH.OPERATION is activated in Eqn. 4.6.30.3.

```
let .APPROACH.STATUS = N.X.FINAL.APPROACH(1) " N.X.resource is a system
                                " attribute which gives the no. of
                                " requests the resource is
                                " currently satisfying.
let .RWY.STATUS = N.X.RUNWAY(1)
if .APPROACH.STATUS gt 0                " approach is occupied          ...(4.6.30.1)
  call STATUS.APP.OCC
else
  if .RWY.STATUS gt 0 and OCCUPANT.OPERATION eq ..LANDING               ...(4.6.30.2)
    call STATUS.RWY.OCC
  else
    activate a FINAL.APPROACH.OPERATION now                             ...(4.6.30.3)
  always
```

```
always
```

4.6.31 Routine SYS.CHECK.AT.THRLD

This routine checks the system at runway threshold for four conditions that are discussed in Section 4.5.2.

```
if COUNT.DEP gt COUNT.T.OFF
  let .APP.STATUS = N.X.FINAL.APPROACH(1)
  let .RWY.STATUS = N.X.RUNWAY(1)
  if .APP.STATUS eq 0  and .RWY.STATUS eq 0   "approach and runway clear
    activate a DEPARTURE.OPERATION now
  else
    if .APP.STATUS eq 0 and .RWY.STATUS gt 0  "approach clear and
                                    "runway occupied
      call STATUS.APP.CLR.RWY.OCC
    else
      if .APP.STATUS gt 0 and .RWY.STATUS eq 0 " approach occupied and
                                        " runway clear
        call STATUS.APP.OCC.RWY.CLR
      else
        if .APP.STATUS gt 0 and .RWY.STATUS gt 0  "both approach and
                                        " runway occupied
          call STATUS.APP.OCC.RWY.OCC
        always
      always
    always
  always
always
```

4.6.32 Process WAIT.AT.RAMP

This process simulates the waiting of a departing aircraft at the ramp. After the waiting period, the control is passed to SYS.CHECK.AT.THRLD to again check the system and evaluate the changes in the system and determine delay, if any, due to the new state of the system.

```
wait WAIT.TIME.RAMP  seconds
call SYS.CHECK.AT.THRLD
```

4.6.33 Process WAIT.AT.STACK

This process simulates the waiting of an arriving at the stack. After the duration of WAIT.AT.STACK the aircraft immediately enters the final approach by initiating the FINAL.APPROACH.OPERATION.

```
wait  WAIT.TIME.STACK seconds
activate a FINAL.APPROACH.OPERATION now
```

4.7 MODEL OUTPUT

The model generates the following output:

i) **Global Statistics:** In this, the following data is presented:

a) Total Arrivals Simulated

b) Total Arrival Delay

c) Average Arrival Delay

d) Total Arrival Simulation Time

e) Total Departures Simulated

f) Total Departure Delay

g) Average Departure Delay

h) Total Departure Simulation Time

i) Weighted Average ROT (WAROT)

j) Standard Deviation of WAROT

k) Total Simulation Time

The above data is also stored in a output file called *******G.#, by the program immediately after each iteration, where "*******" is the first seven letters of the application file name given by the user, "G" is for global statistics, and "#" denotes the iteration number.

ii) Arrival Event List: In this, the following data is presented for every arrival:

a) Creation Time

b) Final Approach Entering Time

c) Time of Landing

d) ROT

e) Exit Chosen

f) Delay

This output is stored in a output file called "*******R.#", where "R" stands for arrival event

list data, and the rest being same as discussed before.

iii) Departure Event List: In this, the following data is shown for every departure:

a) Creation Time

b) Time of Takeoff

c) Delay

The output is automatically saved after every iteration in a file called "*******D.#", where "D" stands for departure event list, and the rest being same as discussed before.

iv) Assignment Table: This table displays how many times each aircraft from the whole population of arrival mix is assigned to each exit. The output, if needed has to be saved by the user, but the file name is defined by the program as "*******A.#", where "A" denotes as assignment table file, and the rest of them being same as discussed before.

The output in i) and iv) can be printed from the program, but ii) and iii) can be printed only from HIGH2 subdirectory either in SIMLAB or DOS environment.

5. Model Results and Analysis

5.1 Analysis Description

In order to demonstrate the versatility and the capabilities of the model several scenarios were developed. These scenarios represent possible airport systems under present and future ATC conditions with various exit locations, geometries and under various aircraft operational parameters.

An initial "baseline scenario" was developed using existing ATC conditions and with an aircraft mix representative of a large hub airport facility. Several input parameters were varied from "baseline scenario" to test the sensitivity of the model. Other scenarios analyzed and their corresponding input parameters were:

1) Scenario-I : Number of exits.
2) Scenario-II: Aircraft Landing Weights.
3) Scenario-III: Arrival Intrail Separations.
4) Scenario-IV: REDIM generated exits.
5) Scenario-V: REDIM generated exits and new Intrail Separations.

5.1.1 Base Scenario

Runway 16R, shown in Figure 5.1, at the Seattle-Tacoma (Sea-Tac) International Airport was chosen as the test runway to perform complete analysis. The current aircraft population using this facility and other input parameters used for this analysis are discussed below:

1) Aircraft Population: Sea-Tac International is representative of a large hub facility with a present aircraft mix index of 66. The aircraft population is shown in Table 5.1., and it can be seen that a fairly large presence of commuter operations take place today. The general aviation population is small with only 2% of the total operations.

2) Arrival and Departure Rates: A poisson distribution of interarrivals and interdeparture was assumed. The interarrival time was varied from 125 seconds to 150 seconds to test the sensitivity of the runway delay to demand rate. In this range the total arrival delay is very sensitive to the demand rate because the demand is reaching the arrival capacity of the runway. A 150 seconds interdeparture rate was used for the analysis. The reason for a constant interdeparture rate is that the main emphasis of this analysis is to study effect of different parameters on the arrival capacity and delay during the peak hours of arrival's demand. The model however, is flexible enough to allow any combination of interarrival and interdeparture times.

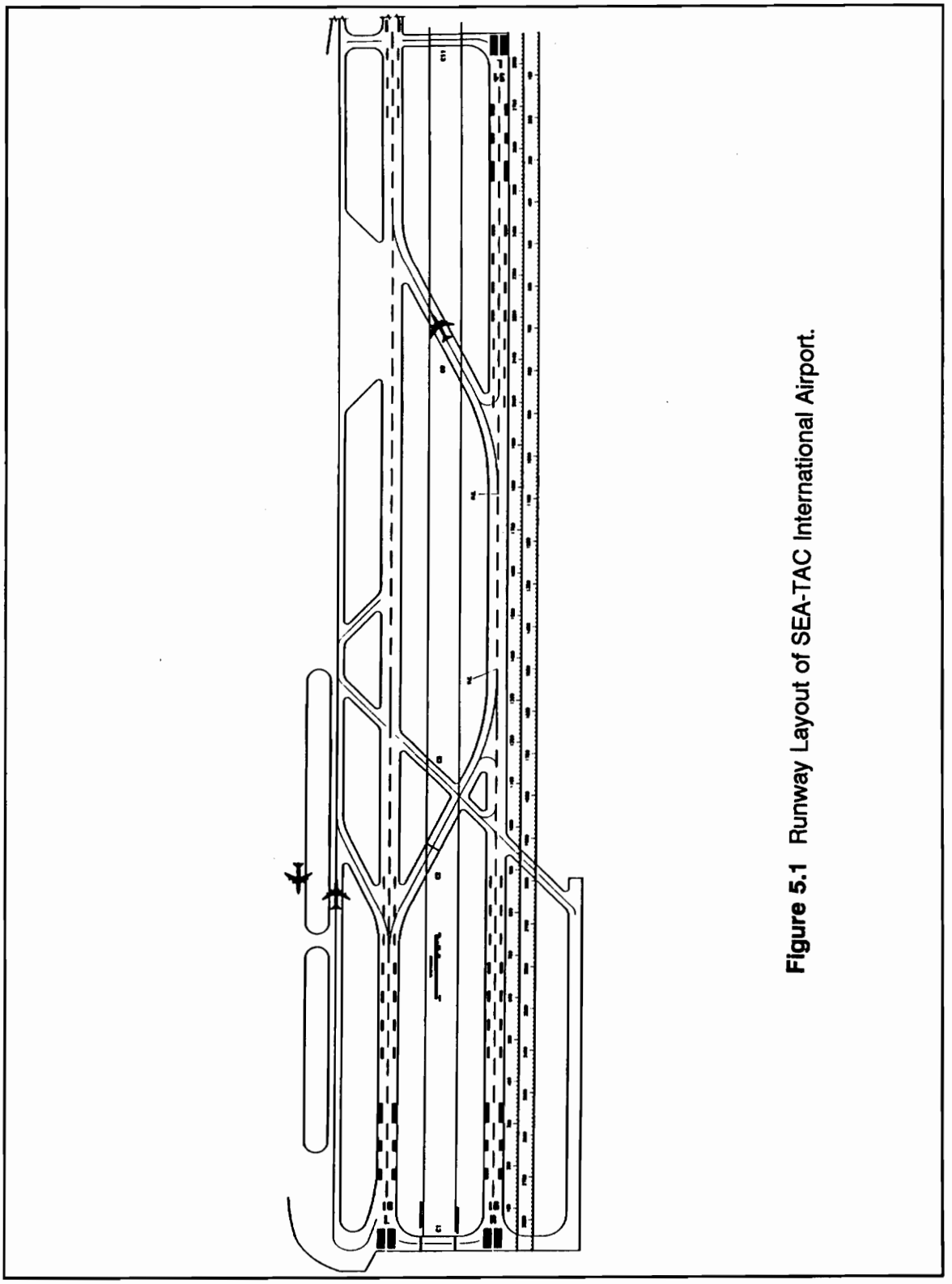

Figure 5.1 Runway Layout of SEA-TAC International Airport.

Table 5.1 Arrival/Departure aircraft and mix.

Aircraft name	% of total arrival/departures
A-300-600	2
A-310-300	3
B-727-200	17
B-737-300	15
B-747-400	2
B-767-300	1
MD-83	11
PA-38-112	1
PA-32-301	2
CE-210P	1
CE-421	4
CE-550	4
CL-601-3A	3
BAe-31	10
EMB-120	2
DHC-8-100	10
SA-227-AT/41	12

3) **Number of Arrivals and Departures**: Due to the stocasticity of the model 1000 arrivals and 500 departures were used per iteration to represent operations over a long period of time.

4) **Weight Factor Data**: This data pertains to landings only. The mean and standard deviation for weight factor of each aircraft specified is shown in Table 5.2. The data used were chosen to model different flight stages the aircraft would have travelled to reach the destination. The significance of this parameter will be evident later in this thesis.

5) **Intrail Separations**: The current FAA set ATC separations were used for the base run, are shown in Tables 4.1, 4.2, and 4.3.

6) **Buffer Data**: The air traffic control interoperation time buffer data used for the analysis is shown in Table 5.3.

7) **Airport Data**: Data containing airport environment and runway exit taxiway characteristics for this baseline analysis is shown in Tables 5.4 and 5.5.

With the above data, and for each interarrival time RUNSIM was run for five iterations to generate data for total delay for arrivals and departures, weighted average runway occupancy time (WAROT) and its standard deviation. The average values of these runs was used for plotting a demand versus average delay graph as shown in Figure 5.2. Figure 5.2 illustrates that as the demand nears the ultimate capacity (i.e., capacity

Table 5.2 Weight Factor Data.

Aircraft name	Mean	Std. Dev.
A-300-600	.40	.10
A-310-300	.40	.10
B-727-200	.50	.10
B-737-300	.50	.10
B-747-400	.50	.10
B-767-300	.40	.10
MD-83	.50	.10
PA-38-112	.70	.10
PA-32-301	.50	.10
CE-210P	.85	.10
CE-421	.85	.10
CE-550	.70	.10
CL-601-3A	.50	.10
BAe-31	.70	.10
EMB-120	.70	.10
DHC-8-100	.50	.10
SA-227-AT/41	.70	.10

Table 5.3 Buffer Data.

Separation	Probability of Violation	Std. Dev. (Sec)
Interarrival	.05	20
Interdeparture	.05	10
Departure-Arrival	.05	10

Table 5.4 Airport Environment Data.

Airport Elevation	130 m. (m.s.l)
Airport Mean Temparature	15° C
Runway Length	2853 m.
Runway Width	45 m.
Final Approach Length	12000 m.
Number of Exits	4

Table 5.5 Runway Exit Taxiway Data.

Exit No.	Exit Type	Location (m)
1.	45° FAA	1006
2.	90° FAA	1142
3.	30° FAA	1749
4.	90° FAA	2853

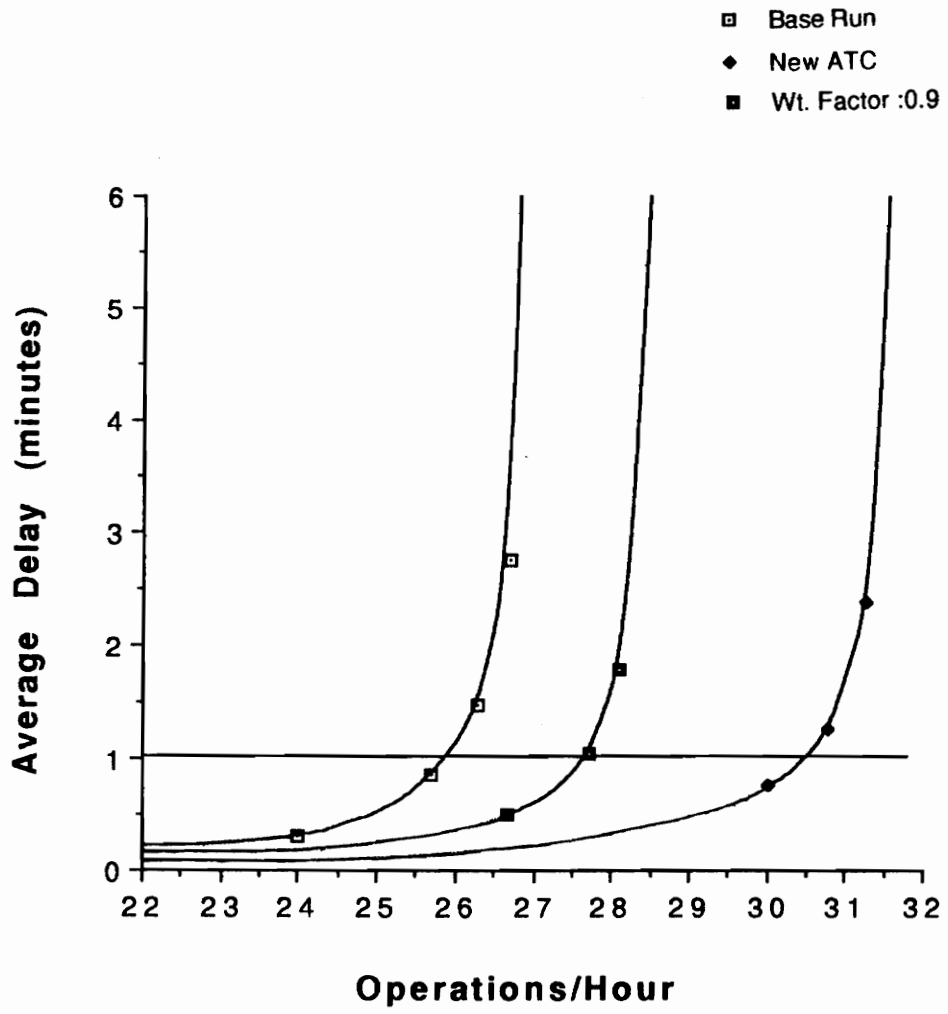

Figure 5.2 Capacity-Delay Curves for different Scenarios.

associated with an infinite delay) the total delay increase very rapidly. For an acceptable average delay of 1 minute, the practical capacity for arrivals is 25.7 arrivals per hour under the baseline aircraft mix. The resulting WAROT of the aircraft population is 51.75 seconds. A sample of the aircraft exit assignment for this scenario is shown in Table 5.6, which is the output of the first iteration. The aircraft exit assignment will be a very useful input for the airport operators for planning the taxiway operations efficiently.

5.1.2 Scenario-I

Here we study effect of addition of exits on the delay output parameters. First, only one exit is added at 1442 meters from the threshold and then another exit is added at 675 meters from the threshold. Separate runs were made to observe variations in output parameters. The new WAROT with one additional exit was 46.62 seconds, and for two it was 45.58 seconds. Compared to the addition of one exit, the gain in WAROT for the second addition is small as a reduced number of aircraft operations (mainly GA's) take advantage of this second exit as shown in Figure 5.3. The arrival capacity-delay curve for both additions were same as that of the base scenario. This implies that the interarrival separation and not the ROT of the aircraft is the critical factor governing the capacity and delay. But the average delay for departures decreased from 1.48 minutes to 1.01 minutes, and this could be attributed due to the availability of more acceptable gaps for departures, an effect of decreased WAROT. The change in the aircraft exit assignment is shown Tables 5.7 and 5.8. Further discussion of the economic implications

Table 5.6 Base Scenario Exit Assignment.

ASSIGNMENT TABLE

AIRCRAFT NAME	EXIT NO. 1	2	3	4
A-300-600	0	1	24	0
A-310-300	0	0	39	0
B-727-200	0	0	164	0
B-737-300	1	47	114	0
B-747-400	0	0	22	0
B-767-300	0	0	10	0
MD-83	0	1	112	0
PA-38-112	11	0	0	0
PA-32-301	17	0	0	0
CE-210P	9	0	0	0
CE-421	35	0	0	0
CE-550	36	0	0	0
CL-601-3A	28	0	0	0
BAe-31	93	0	0	0
EMB-120	15	10	2	0
DHC-8-100	92	0	0	0
SA-227-AT/41	117	0	0	0

Exit No.	Location (m)
1.	1006
2.	1142
3.	1749
4.	2853

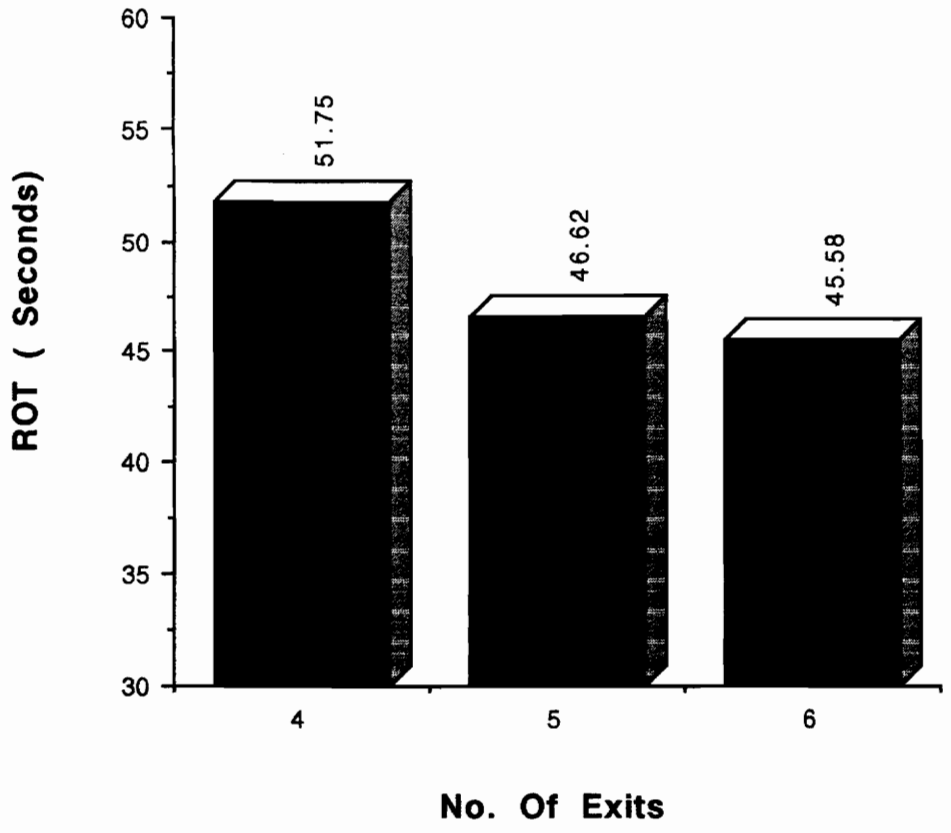

Figure 5.3 Comparision of ROT between varying exits.

Table 5.7 Scenario-I (one exit) Exit Assignment.

ASSIGNMENT TABLE

AIRCRAFT NAME	EXIT NO. 1	2	3	4	5
A-300-600	0	1	24	0	0
A-310-300	0	0	39	0	0
B-727-200	0	0	164	0	0
B-737-300	1	47	114	0	0
B-747-400	0	0	1	21	0
B-767-300	0	0	10	0	0
MD-83	0	1	112	0	0
PA-38-112	11	0	0	0	0
PA-32-301	17	0	0	0	0
CE-210P	9	0	0	0	0
CE-421	35	0	0	0	0
CE-550	36	0	0	0	0
CL-601-3A	28	0	0	0	0
BAe-31	93	0	0	0	0
EMB-120	15	10	2	0	0
DHC-8-100	92	0	0	0	0
SA-227-AT/41	117	0	0	0	0

Exit No.	Location (m)
1.	1006
2.	1142
3	1442
4	1749
5	2853

Table 5.8 Scenario-I (two exits) Exit Assignment.

ASSIGNMENT TABLE

AIRCRAFT NAME	EXIT NO. 1	2	3	4	5	6
A-300-600	0	0	8	17	0	0
A-310-300	0	0	0	39	0	0
B-727-200	0	0	0	164	0	0
B-737-300	0	1	47	114	0	0
B-747-400	0	0	0	6	16	0
B-767-300	0	0	0	10	0	0
MD-83	0	0	1	112	0	0
PA-38-112	11	0	0	0	0	0
PA-32-301	17	0	0	0	0	0
CE-210P	9	0	0	0	0	0
CE-421	1	34	0	0	0	0
CE-550	4	32	0	0	0	0
CL-601-3A	6	22	0	0	0	0
BAe-31	0	72	21	0	0	0
EMB-120	0	5	9	13	0	0
DHC-8-100	3	89	0	0	0	0
SA-227-AT/41	0	117	0	0	0	0

Exit No.	Location (m)
1.	675
2.	1006
3.	1142
4.	1442
5.	1749
6.	2853

will be discussed in Chapter 6., of this thesis.

5.1.3 Scenario-II

This scenario models the effect of different flight stages of the same aircraft population on the capacity. This is done by changing the mean and standard deviation of the weight factor. For this scenario all the means and the standard deviations were changed to 0.9 and 0.01 respectively, to depict the aircraft landing at near maximum allowable landing weights (MALW) and thus simulating short stage length segment operations. This scenario also includes the two extra exits used in Scenario-I. The rest of the data being same as "base scenario", the model is run for different demand rates and the results are plotted in Figure 5.2. Figure 5.2 shows the increase in capacity (practical) to 27.5 as compared to 25.7 operations per hour of the "base scenario". This was attributed to the increased speeds caused by the increase in aircraft weight and also the exit locations were more optimal for these weight factors as these were determined with REDIM at MALW. The WAROT has decreased to 44.19 seconds as compared to 45.58 of Scenario-I (two exits), as shown in Figure 5.4. Due to the increased weight the aircraft exit assignment has shifted downwards, shown by the computer output in Table 5.9, as compared to exit assignment in Table 5.8. Although the assignment of exits has shifted downrange from the threshold the WAROT has still decreased, and this due is to the fact that the aircraft have exited the runway at higher speeds, than in Scenario-I (two exits).

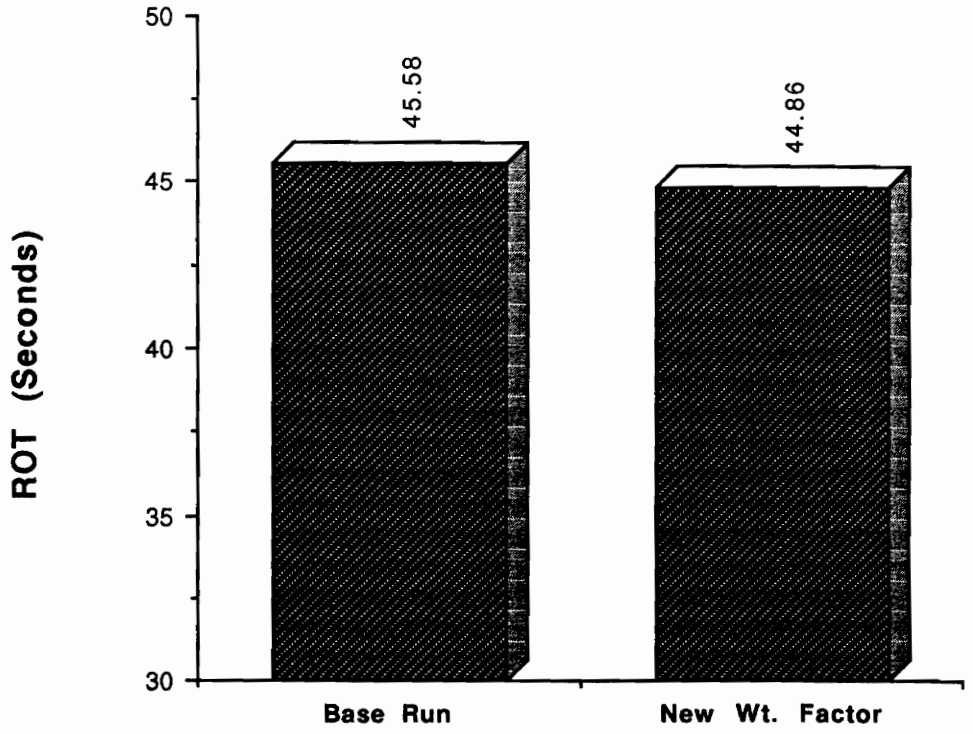

Figure 5.4 Comparision of ROT between different Weight Factors.

Table 5.9 Scenario-II Exit Assignment.

ASSIGNMENT TABLE

AIRCRAFT NAME	EXIT NO. 1	2	3	4	5	6
A-300-600	0	0	0	25	0	0
A-310-300	0	0	0	39	0	0
B-727-200	0	0	0	146	18	0
B-737-300	0	0	0	162	0	0
B-747-400	0	0	0	0	10	12
B-767-300	0	0	0	10	0	0
MD-83	0	0	0	99	14	0
PA-38-112	11	0	0	0	0	0
PA-32-301	17	0	0	0	0	0
CE-210P	9	0	0	0	0	0
CE-421	1	34	0	0	0	0
CE-550	0	36	0	0	0	0
CL-601-3A	0	27	1	0	0	0
BAe-31	0	39	47	7	0	0
EMB-120	0	0	6	21	0	0
DHC-8-100	0	92	0	0	0	0
SA-227-AT/41	0	117	0	0	0	0

Exit No.	Location (m)
1.	675
2.	1006
3.	1142
4.	1442
5.	1749
6.	2853

5.1.4 Scenario-III

The current ATC separation rules set by FAA is the critical parameter governing the delay at most airport facilities. By improving the technology in dealing with wake turbulence, improved radar technology for better air traffic control, the FAA proposes to decrease the interarrival separation to the values shown in Table 5.10. This scenario studies the effect of new ATC separation rules on capacity and delay. The model is run by changing the arrival separation to the new values and keeping the other values same as in "baseline scenario". The Figure 5.2 shows the capacity and delay relationship of this scenario, where the capacity (practical) has increased to 30.5 operations per hour, which is an increase of 5 operations (arrivals) per hour as compared to the present rules. The average delay to departures has increased from 1.48 to 2.75 minutes, this is because of the decreased interarrival spacing whereby effectively the acceptable gaps for departures has been decreased. To achieve reduced delays for both the arrivals and departures the interdeparture and departure-arrival separations should also be decreased.

5.1.5 Scenario-IV

One of the main purposes of developing RUNSIM is to predict the capacity enhancements due to REDIM designed variable geometry high speed exits. In this scenario two case were run one with the runway designed with FAA standard exits but located optimally using REDIM and the second case also has only REDIM designed high speed exits and

Table 5.10 1996 Proposed ATC Separation Rules.

Lead Aircraft	Heavy (miles)	Large (miles)	Small (miles)
Heavy	3.0	4.0	5.0
Large	2.0	2.0	3.0
Small	2.0	2.0	2.0

locations. REDIM model was run with same aircraft population of "baseline scenario" to find the optimal location of exits. With a 90° exit at the end of the runway the other three exits for both the cases were located as shown in Tables 5.11 and 5.12. Table 5.11 also shows the type of exits used for the simulation. The WAROT achieved from the simulation runs for the first case was 37.22 and 29.19 seconds for the second case as compared to 51.75 seconds of the "baseline scenario", whose comparision is shown in Figure 5.5. This was achieved because for the first case all the first three exits are 30° FAA standard exit types with 26m/s exit speed and located optimally and for the second case the exit speed of the REDIM exits was much higher than the standard FAA exits facilitating the aircraft to exit at higher speeds whereby reducing their runway occupancy time. The capacity-delay curve is same as that of the "base scenario". This illustrates that still the interarrival separation is the critical factor. The average delay for departures has decreased to 0.5 minutes as compared to 1.48 minutes of "baseline scenario" for REDIM designed high-speed exits case. Effectively REDIM exits help in increasing the total capacity of the runway by increasing the departure capacity.

5.1.6 Scenario-V

In this scenario, the combined effect of new 1996 ATC interarrival separation and the REDIM exits on the capacity and delay is investigated. The interarrival separations is changed to the values to shown in Table 5.10. The new capacity-delay curve that was generated from the simulation runs was similar to that of scenario-III. The WAROT is

Table 5.11 Scenario-IV (First Case)

Exit No.	Exit Type	Exit Speed (m/s)	Exit Location (m)
1	30° FAA	26	878
2	30° FAA	26	1242
3	30° FAA	26	1607

Table 5.12 Scenario IV (Second Case)

Exit No.	Exit Type (REDIM)	Critical Acft. Type	Exit Speed (m/s)	Location (m)
1.	30°	Small	25	979
2.	30°	Large	35	1679
3.	20°	Heavy	35	2312

* The last exit (90° FAA) for both cases is located at the end of runway.

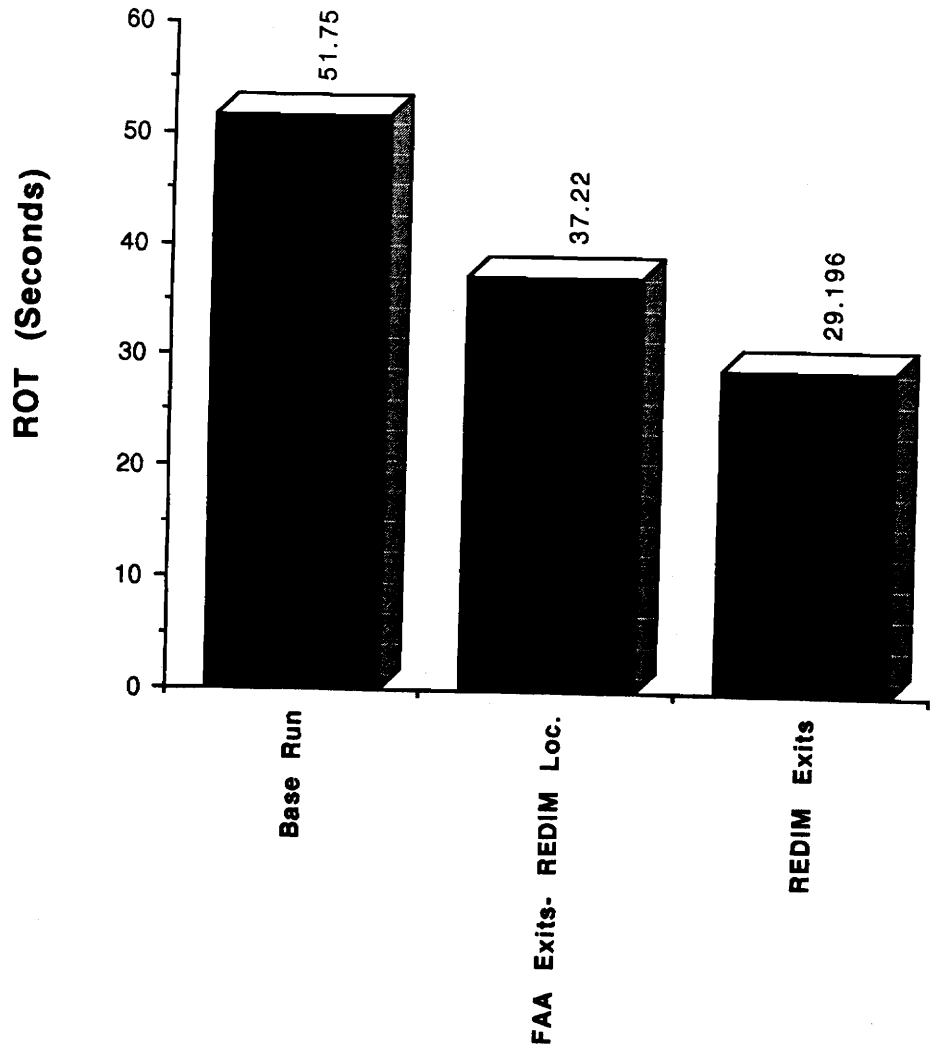

Figure 5.5 Comparision of ROT between FAA and REDIM Exits.

same as that of the scenario-IV. Compared to scenario-III, the average delay for departures has decreased from 2.75 to 0.75 minutes.

Hence for the REDIM exits to be more effective and dominant factor in the capacity of arrivals, the ATC separations have to be further more decreased and this is possible only if new technology advances enough to allow smaller separations.

6. Conclusions and Recommendations

6.1 Conclusions

The simulation approach to investigate the problem of relieving the airport congestion and improving the airport capacity with varied input parameters has proven to be very effective. The model has proved to be very flexible enough to the present needs. The decision to use SIMSCRIPT II.5 as a tool develop the simulation model has definitely paid off. As shown and described in Chapter 2 and 4, English-like syntax is self documenting , and readable which makes the task of future enhancements by other researchers very easy. Like any other high-level simulation languages, its built-in timing routine and other functions and random number generators has reduced both the programming and project time.

With the amount of time and money being wasted due to delays, the want for solutions are congestion problem is more than it was ever before. There is a great potential for the simulation models like RUNSIM, REDIM, and SIMMOD to address these problems and given insight to the system more clearly and help in decision-making for improvements.

As demonstrated in the former chapter, RUNSIM is capable on aiding a planner to develop strategies to improve the capacity of the airport, or atleast from the runway

operations point of view. RUNSIM could be applied in simulating runway for scenarios including sensitivity to the following:

a) Airport types, by varying aircraft weight factors and mixes,

b) Demand rates and patterns (poisson, exponential, and uniform),

c) Intrail separations including buffers,

d) Number and types of exit taxiways,

e) and other airport environmental conditions.

It was proved that the use of REDIM generated high-speed exits can result in moderate gains in total capacity of the airport. This is mainly due to the increase in the number of departure operations, and reductions of their delays. These reductions in delay have economically great effect on other operating costs like fuel consumption. In the top five most busy airports in U.S., currently the total number of enplanements are above 12 millions [FAA, 1990] per year and even if 25% of them are departures during the peak hours, the savings in fuel consumptions ($ 3 million from the case study), user costs, other operational costs, and the reductions in air pollution due to lesser idling of the aircraft at the departure ramps is worth implementing the improvements. The higher exiting speeds by the aircraft also decreases the taxiing time to terminal, whereby effectively increasing the capacity of the taxiway system. Further improvements in the airport capacity can be realized if the FAA proposed 1996 intrail separations are incorporated along the REDIM generated high-speed. It was also found that REDIM designed exits will be very effective in increasing the arrival capacity once the interarrival

separations are decreased to 1.5 miles.

6.2 Recommendations

The following recommendations are a result of the experience gained in developing RUNSIM, by studying and modelling runway operations microscopically, and the analysis done on the present and future airport operations. The proposed recommendations for future research are:

a) Improve the input module of RUNSIM by using the features of SIMGRAPHICS. This would make the model more user-friendly, will enable to incorporate more aircraft models very easily.

b) Improve the output module of RUNSIM by including graphic presentation of the output. This would enhance the understanding of the system that is being studied.

c) Enhance RUNSIM to have the capability of simulating different types of runway configurations.

d) Enhance RUNSIM to incorporate taxiway operations so as to study the effect of REDIM generated high speed exits on the taxiway system in detail. This would permit further evaluation of ground operational procedures allowing aircraft taxi at higher speeds

and effectively increase the taxiway capacity.

e) Incorporate the algorithms of dynamic simulation of landings of RUNSIM into SIMMOD to make the simulation of landings in SIMMOD more realistic. This would also help simulating high speed exits more realistic than it does currently. In SIMMOD, the cumulative probability distributions for landing distances and the constant roll times should be changed by including the more realistic landing modelling performed in RUNSIM. The percentages of roll distances incorporated in SIMMOD to model high-speed should also to be substituted by incorporating the turnoff phase algorithm of RUNSIM and include the data base similar to TOT.DAT (see Appendix C). The suggested could be easily performed as both the models are developed by using SIMSCRIPT II.5, and hence transferring of the model code is easy.

f) Enhance SIMMOD's definition of high speed exits and its modelling and incorporate REDIM designed exits.

g) Gain more knowledge in decreasing the effects of wake turbulence on the trailing aircraft while landing so as to increase the capacity of the airport, as it was demonstrated that most of the time intrail separations dictated the capacity of the airport.

Bibliography

1. Ashford, N. and Wright, P.H., Airport Engineering, John Wiley & Sons, 1979.

2. Barrer, J.N. and Dielh, J.M., "Toward a Goal-Oriented Plan for Identifying Technology to Increase Airfield Capacity," The MITRE Corporation, McLean, Virginia, 1988.

3. CACI Products Company, SIMGRAPHICS User's Guide and Casebook, CACI Products Company, La Jolla, California, 1990.

4. Federal Aviation Administration, FAA Aviation Forecasts Fiscal Years 1990-2001, FAA-APO-90-1, Washington, D.C., March 1990.

5. Federal Aviation Administration, Airport Capacity Enhancements Plan, DOT/FAA/CP 88-4, Washington, D.C., April 1988.

6. Federal Aviation Administration, Airport Capacity and Delay, Advisory Circular AC 150/5060-5, September 1983.

7. Harris, R.M., "Models for Runway Capacity Analysis," MITRE Technical Report No. 4102, The MITRE Corporation, Washington Operations, December 1972.

8. Henriksen, J.O. and Crain, R.C., The GASP/H User's Manual, Second Edition, Wolverine Software, Annandale, Virginia, 1983.

9. Horonjeff, R. and Mckelvey, F.X., Planning and Design of Airports, Third Edition, McGraw-Hill Book Company, New York, 1983.

10. Nicolai, L.M. Fundamentals of Airport Design, METS, Inc., San Jose, California, 1975.

11. Pritsker, A.A.B., Introduction to Simulation and SLAM II, A Halsted Press Book, John Wiley & Sons, Third Edition, 1986.

12. Pritsker, A.A.B., The GASP IV Simulation Language, John Wiley & Sons, 1974.

13. Pugh, A.L., III, DYNAMO II User's Manual, M.I.T. Press, 1970.

14. Roskam, J., Airplane Design Part I, Roskam Aviation and Engineering Corporation Ottawa, Kansas, 1985.

15. Russel, E.C., Building Simulation Models with SIMSCRIPT II.5, CACI Products

Company, La Jolla, California, 1990.

16. Russel, E.C., SIMSCRIPT II.5 Programming Language, CACI Products Company, La Jolla, California, 1990.

17. Torenbeek, E., Synthesis of Subsonic Airplane Design, Van Niemoff, Netherlands, 1981.

18. Trani, A.A., Hobeika, A.G., Sherali, H.D., Kim, B.J., Sadam, C.K., Runway Exit Designs for Capacity Improvements Demonstrations, Phase-I-Algorithm Development, DOT/FAA/RD-90/32,I, Washington, D.C., June 1990.

19. Transportation Research Board, Measuring Airport Landside Capacity, Special Report 215, Washington, D.C., 1987.

20. Wong, J.Y., Theory of Ground Vehicles, John Wiley & Sons, New York, 1958.

21 US Department of Transportation/Federal Aviation Administration, SIMMOD Reference Manual, CACI Products Company, Washington, D.C., September 1989.

Appendix A. RUNSIM Source Code

```
preamble

"Program Name: Runway Simulation Program
"File Name: High2
' Programmer: Vijay B. Nunna
"Description: This program simulates airport runway operations which includes
"             landings from stack at the beginning of the final approach till
"             the aircraft exits the turnoffs. Takeoffs are till the aircraft
"             clears the runway end.
" Date: 3/6/91(original), 7/30/91(Version-I.0)

normally mode is undefined

resources

  every RUNWAY has

    a LENGTH.RWY

      define LENGTH.RWY  as a real variable

  every FINAL.APPROACH has

    a LENGTH.APP

      define LENGTH.APP as a real variable

processes include

  FINAL.APPROACH.OPERATION,

  LANDING.OPERATION,

  DEPARTURE.OPERATION,

  GENERATOR,

  GENERATOR.DEP,

  WAIT.AT.STACK,

  WAIT.AT.RAMP

events include

   CREATE.AIRCRAFT.ARR,

   CREATE.AIRCRAFT.DEP

temporary entities

  every AIRCRAFT has
```

```
  a NAME,          "Name of the aircraft
  a V.APP,         " Velocity of approach
  a TYPE,          " Classification of aircraft by weight
  a APP.OCC.TIME,  " Time to travel the final approach
  a RWY.OCC.TIME,  " Runway Occupancy Time
  a TAKEOFF.TIME,  " Takeoff time i.e. to clear the runway
  a OPERATION      " to know if it is landing or departure

    define NAME as text variable
    define V.APP,
           APP.OCC.TIME,
           RWY.OCC.TIME,
           TAKEOFF.TIME as real variables
    define TYPE,
           OPERATION as integer variables
define ARR.ID,
      DEP.ID as 1-dimensional pointer arrays
define T.ENTER.STACK,
      T.DEP.STACK,
      D.STACK,
      T.LANDING,
      T.ENTER.RAMP,
      T.DEP.RAMP,
      EXIT.SPEED,
      EXIT.LOCATION as real 1-dimensional arrays
define ARR.ACFT.NO,
      DEP.ACFT.NO,
      EXIT.ASSIGNED,
      EXIT.CHOICE,
      NO.IDENT.ARR,
```

```
        ARR.ACFT.TYPE,
        DEP.ACFT.TYPE  as integer 1-dimensional arrays
    define ARR.ACFT.PERCENT,
        ARR.ACFT.MALW,
        ARR.ACFT.OEW,
        ARR.ACFT.CL.MAX,
        ARR.ACFT.FREE.ROLL,
        ARR.ACFT.DEC,
        ARR.ACFT.WING.AREA,
        ARR.ACFT.WING.SPAN,
        ARR.ACFT.WTF.MEAN,
        ARR.ACFT.WTF.STD,
        DEP.ACFT.PERCENT,
        DEP.ACFT.MATW,
        ROT.STAT as real 1-dimensional arrays
    define ACFT.NAME,
        ARR.ACFT.NAME,
        DEP.ACFT.NAME as text 1-dimensional arrays
    define EXIT.ASSIGN as 2-dimensional integer array
    define SEP.DIST.ARR,
        DEP.ARR.SEP,
        SEP.T.TAKEOFFS as real 2-dimensional arrays
    define NO.ARR.AIRCRAFT,
        NO.DEP.AIRCRAFT,
        TOT.ARR.ACFT,
        TOT.DEP.ACFT,
        COUNT.ARR,
        COUNT.DEP,
        COUNT.APP,
        COUNT.LAND,
```

```
        COUNT.T.OFF,
        NO.EXITS,
        RWY.WIDTH,
        RUN.NO,
        OCCUPANT.OPERATION as integer variables
    define ARR.DIST,
        ARR.DIST.MEAN,
        ARR.DIST.STD,
        DEP.DIST,
        DEP.DIST.MEAN,
        DEP.DIST.STD,
        INTARR.BUFF,
        INTDEP.BUFF,
        INTDEP.ARR.BUFF,
        AIR.ELEV,
        AIR.TEMP,
        RWY.LTH,
        WAIT.TIME.RAMP,
        WAIT.TIME.STACK,
        END.LAND.SIM,
        END.DEP.SIM as real variables
    define  APP.FILE.NAME as a text variable
" TITLE ANIMATION
    Normally mode is integer
    Temporary entities include
        MAP
    Graphic entities include
        MAP
    processes include
     ACFTFIG
    dynamic graphic entities include
     ACFTFIG
```

```
   Define ..DISPLAY.VIEW.PORT to mean 2
" TITLE ANIMATION  DECLERATIONS END
  define minutes to mean units
  define minute to mean units
  define seconds to mean /60 minutes
  normally mode is undefined
  tally TOT.ARR.DELAY as the sum of ARR.DELAY
  tally TOT.DEP.DELAY as the sum of DEP.DELAY
  define TOT.ARR.DELAY,
        TOT.DEP.DELAY as real variables
  define ..LANDING to mean 1
  define ..DEPARTURE to mean 2
  define ..HEAVY to mean 1
  define ..LARGE to mean 2
  define ..SMALL to mean 3
 end " preamble
main
 define .NO.REPS,
       .ITER as integer variables
 " title animation starts
 call SET.GRAPHICS
 Create a MAP
 Display MAP with "lhw"
 activate a ACFTFIG now
 show ACFTFIG with "PLANE.ICN"
 start simulation
 let time.v = 0
 let timescale.v = 1 " reinitializing for faster simulation
 call vclears.r
" title animation ends
```

```
use unit 6 for output

let lines.v = 0 " to eliminate page breaks during screen display

create every RUNWAY(1)

let U.RUNWAY(1) = 1

create every FINAL.APPROACH(1)

let U.FINAL.APPROACH(1) = 5 "To allow a max. of 3 aircraft to use or enter
                        " final approach at any given time

" call PROG.STRT.SCRN

call APP.CHOICE

call READ.APP.DATA

call READ.DATA.FILE

call DET.EXIT.SPEEDS

print 1 line thus
STARTING THE SIMULATION

call NO.REPLICATIONS yielding .NO.REPS

for .ITER = 1 to .NO.REPS, do

   let RUN.NO. = .ITER

   if TOT.ARR.ACFT gt 0

     activate a GENERATOR now

   always

   if TOT.DEP.ACFT gt 0

      activate a GENERATOR.DEP now

   always

     call vbcolor.r(1)

     call vclears.r

     call vfcolor.r(5)

     call vgotoxy.r(16,10)

     print 1 line thus
               MODEL IS EXECUTING

     call vfcolor.r(14)

     print 1 line with .ITER, .NO.REPS thus
                     ITERATION ****  OF ****
```

```
      call vfcolor.r(2)

      print 1 line thus
                          PLEASE WAIT.....

    call setvt.r(7,0)

    start simulation

    call setvt.r(7,2)

    call DO.BEEP

    call GLOBAL.STAT.OP giving .ITER

    call REPORT.GEN giving .ITER

    call RE.INITIALIZE

  loop " .ITER = 1 to .NO.REPS

end "main
routine ACFT.DUP.CHECK given .IAD, .ACFT.NO.AD yielding .NEW.ACFT.NO, .DUP

" This routine checks if the newly selected aircraft is already selected
" or not. If so then the user is asked to choose again.
" CALLED FROM: NEW.APP.STRT
" CALLS: DO.BEEP

  define .IAD,

       .IDUP,

       .NEW.ACFT.NO as integer variables

  define .ACFT.NO.AD as a 1-dimensional integer array

  define .DUP as a text variable

  reserve .ACFT.NO.AD as .IAD

  if .IAD gt 1

    for .IDUP = 1 to (.IAD - 1) do

      if .ACFT.NO.AD(.IAD) eq .ACFT.NO.AD(.IDUP)

        call DO.BEEP

        call vgotoxy.r(22,1)

        call vbcolor.r(12)

        print 1 line thus
WARNING: THE AIRCRAFT IS ALREADY SELECTED: CHOOSE ANOTHER ONE

        call vbcolor.r(1)

        call vgotoxy.r(17,44)
```

```
            call vclearl.r
            read .NEW.ACFT.NO
            call vgotoxy.r(22,0)
            call vclearl.r
            let .DUP = "Y"
            return
          always
        loop
      always
      let .NEW.ACFT.NO = .ACFT.NO.AD(.IAD)
      let .DUP = "N"
      return
end " acft.dup.check
routine ACFT.LIST.SCRN yielding .NO.RECORDS

"This routine generates aircraft list on screen for simulation selection
"CALLED FROM: NEW.APP.STRT
    define .NO.RECORDS,
           .I,
           .II as integer variables
    call vclears.r
    call vfcolor.r(14)
    print 1 line thus
                            AIRCRAFT  LIST

    call vfcolor.r(4)
   print 1 line thus
```

```
    call vfcolor.r(15)
    open unit 10 for input, name = "ACFT.DAT"
    use unit 10 for input
    "JUMP OVER THE COMMENTS IN THE DATA FILE
    while mode is alpha do
```

```
    start new record

  loop " mode is alpha do

  start new input record

  read .NO.RECORDS

  reserve ACFT.NAME as .NO.RECORDS

  start new input record

  for .I = 1 to .NO.RECORDS, read ACFT.NAME(.I) as  t 12 , /

  close unit 10

  for .II = 1 to .NO.RECORDS  by 3  do

   write .II, ACFT.NAME(.II), (.II + 1 ), ACFT.NAME(.II + 1 ), (.II + 2 ),
     ACFT.NAME(.II + 2 ) as s 8 , i 2 ,")" , s 1 , t 12 , s 5, i 2,")",
       s 1 , t 12, s 5, i 2, ")", s 1, t 12, /

 loop

 call vfcolor.r(4)

 print 1 line thus
```

```
 call vfcolor.r(15)

 return

end " acft.list.scrn
process ACFTFIG


   define .X1,
          .X2,
          .Y1,
          .Y2,
          .SPEED,
          .TIME,
          .DIRECTION as real variables


   let .X1 = -15.0
   let .Y1 = 10.0
   let .X2 = -5.0
   let .Y2 = 5.0
   let .SPEED = 2.0
   let location.a(ACFTFIG) = location.f(.X1, .Y1)
   call VECTOR given .X1, .Y1, .X2, .Y2, .SPEED yielding .TIME, .DIRECTION
   let velocity.a(ACFTFIG) = velocity.f(.SPEED, .DIRECTION)
   Let orientation.a(ACFTFIG) = .DIRECTION
   Display ACFTFIG
   Wait .TIME units

   let .X1 = -5.0
```

```
    let .Y1 = 5.0
    let .X2 = 0.0
    let .Y2 = 2.5
    let .SPEED = 2.0

    call VECTOR given .X1, .Y1, .X2, .Y2, .SPEED yielding .TIME, .DIRECTION
    Let velocity.a(ACFTFIG)    = velocity.f(.SPEED,.DIRECTION)
    Let orientation.a(ACFTFIG) = .DIRECTION
    Display ACFTFIG
    Wait .TIME units

    let .X1 = 0.0
    let .Y1 = 2.5
    let .X2 = 5.0
    let .Y2 = 0.0
    let .SPEED = 2.0

    call VECTOR given .X1, .Y1, .X2, .Y2, .SPEED yielding .TIME, .DIRECTION
    Let velocity.a(ACFTFIG)    = velocity.f(.SPEED, .DIRECTION)
    Let orientation.a(ACFTFIG) = .DIRECTION
    Display ACFTFIG
    Wait .TIME units

    let .X1 = 5.0
    let .Y1 = 0.0
    let .X2 = 12.0
    let .Y2 = 0.0
    let .SPEED = 2.0

    call VECTOR given .X1, .Y1, .X2, .Y2, .SPEED yielding .TIME, .DIRECTION
    Let velocity.a(ACFTFIG)    = velocity.f(.SPEED, .DIRECTION)
    Let orientation.a(ACFTFIG) = .DIRECTION
    Display ACFTFIG
    Wait .TIME units

"     call readloc.r
"       given 0, 0, 0  yielding DUMMY.X, DUMMY.Y, DUMMY.V
"       let DUMMY.X = DUMMY.X
"       let DUMMY.Y = DUMMY.Y
"       let DUMMY.V = DUMMY.V


  end " acftfig
routine APP.CALL

 define .EXIST.APP,
       .FILE.NAME as text variables
   "     .APP.CHOICE as text variables

 define .ROW,
       .COLUMN as integer variables

 call vclears.r

 skip 3 lines

 print 1 line thus
          EXISTING APPLICATION                FILE NAME
```

```
skip 1 line

open unit 10 for input, name = "APP.FILE"

use unit 10 for input

while data is not ended do

read   .EXIST.APP,
       .FILE.NAME

print 1 line with .EXIST.APP and .FILE.NAME thus
          ************************          ****************

loop " if data has not ended

close unit 10

skip 2 lines

print 1 line thus
Choose the application and enter its file name here:

call vgetxy.r yielding .ROW, .COLUMN

let .ROW = .ROW - 1

let .COLUMN = 55
call vgotoxy.r(.ROW, .COLUMN)

read APP.FILE.NAME

let APP.FILE.NAME = upper.f(APP.FILE.NAME)

call READ.APP.DATA

return

end " APP.CALL
routine APP.CHOICE

define .NUM.CHOICE as integer variable

call vclears.r

skip 3 lines

call vbcolor.r(1)

call vfcolor.r(14)

print 1 line thus
                        INPUT MODULE

call vfcolor.r(4)

print 1 line thus
                        ------------
```

```
call vfcolor.r(15)

skip 8 line

print 1 line thus
Select one option

skip 1 line

print 3 lines thus
                    1) Existing Application

                    2) New Application

skip 2 lines

print 1 line thus
Enter your choice number:

call vgotoxy.r(20,27)

read .NUM.CHOICE

until .NUM.CHOICE ge 1 and .NUM.CHOICE le 2, do

    call DO.BEEP

    call vgotoxy.r(22,1)

    call vbcolor.r(12)

    print 1 line thus
MESSAGE : PLEASE ENTER ONLY CHOICE NO. 1 OR 2 ; TRY AGAIN

    call vbcolor.r(1)

    call vgotoxy.r(20,27)

    call vclearl.r

    read .NUM.CHOICE

    call vgotoxy.r(22,0)

    call vclearl.r

  loop

if .NUM.CHOICE = 1

  call  APP.CALL.

 else

  call NEW.APP.STRT

always

return
```

```
 end "APP.CHOICE
routine ARRIVAL.POP.OP given .ITER

" This routine displays the arrival aircraft used for the simulation .
" CALLED FROM: INPUT.DATA.OP
" CALLS:


 define .ITER,

       .ANO,

       .XLOC,

       .YLOC as integer variables

 define .CHOICE as a text variable

 until .CHOICE eq "Q", do

   call vclears.r

   print 3 lines thus
                 ╔═══════════════════════════╗
                 ‖  ARRIVAL AIRCRAFT POPULATION  ‖
                 ╚═══════════════════════════╝

   skip 1 line

   print 1 line with APP.FILE.NAME thus
 APP. FILE NAME : **************

   print 1 line with TOT.ARR.ACFT thus
 NO. OF ARRIVALS : ********

   print 1 line thus
```

```
   let .XLOC = 7

   let .YLOC = 3

   for .ANO = 1 to NO.ARR.AIRCRAFT,  do

     call vgotoxy.r(.XLOC, .YLOC)

     print 1 line with .ANO, ARR.ACFT.NAME(.ANO), ARR.ACFT.PERCENT(.ANO) thus
 | **) ************ (** %) |

     let .XLOC = .XLOC + 1

     if .XLOC eq 20

       let .XLOC = 7

       let .YLOC = .YLOC + 24

     always ".XLOC eq 20
```

```
   loop " .ANO = 1 to NO.ARR.AIRCRAFT
  call vgotoxy.r(20,0)
   print 1 line thus
```

```
   call vgotoxy.r(22,1)
   call vbcolor.r(5)
   call vfcolor.r(15)
   print 1 line thus
ENTER (PRINT SCREEN) KEY TO PRINT; (Q)UIT:
   call vgotoxy.r(22,43)
   read .CHOICE
   call vbcolor.r(3)
   call vfcolor.r(0)
   let .CHOICE = fixed.f(.CHOICE, 1)
   let .CHOICE = upper.f(.CHOICE)
  loop " .CHOICE eq Q
  if .CHOICE eq "Q"
     call INPUT.DATA.OP giving .ITER
  always " .CHOICE eq Q
  return
end " arrival.pop.op
routine ARRIVALS.EVENT.OP given .ITER

"This routine prints on the screen the arrival event list.
"CALLED FROM: REPORT.GEN
"CALLS:
"OPENS(FOR INPUT) : .NEW.FILE.NAME


   define .NEW.FILE.NAME,
          .EXT,
          .CHOICE,
          .NAME as text variables
   define .ITER,
          .CT.REC,
```

```
        .LINE,
        .EXIT.NO,
        .CLEAN,
        .SKIP.REC as integer variables
    define .CR.TIME,
        .EAP.TIME,
        .LD.TIME,
        .RO.TIME,
        .DEL as real variables
    let .NEW.FILE.NAME = substr.f(APP.FILE.NAME,1,7)
    let .EXT = itot.f(.ITER)
    let .NEW.FILE.NAME = concat.f( .NEW.FILE.NAME,"R.", .EXT)
    let .CHOICE = "P"
    while .CHOICE eq "P", do
      call vclears.r
      print 3 lines with APP.FILE.NAME, .ITER thus
┌──────────────────────────────────────ARRIVAL EVENT LIST──────────────────────────────────────┐
| APP. FILE: ************ ──────────────────────────────── RUN NO.: ****        |
|                                          |
      print 4 lines thus
| AIRCRAFT     CREATION    ENTER FINAL   LANDING   ROT    EXIT    DELAY    |
|  NAME         TIME       APPROACH      TIME          CHOSEN          |
|               Min         Min          Min     Sec          Min    |
├──────────────────────────────────────────────────────────────────────────────────────────────┤
      open unit 21 for input, name = .NEW.FILE.NAME
        use unit 21 for input
        skip 10 records
        for .CT.REC = 1 to COUNT.LAND, do
          read .NAME, .CR.TIME, .EAP.TIME, .LD.TIME, .RO.TIME, .EXIT.NO, .DEL
          print 1 line with .NAME, .CR.TIME, .EAP.TIME, .LD.TIME, .RO.TIME,
                .EXIT.NO, .DEL thus
| ************  *****.**  ******.**   ******.**  ***.**  **    ****.**  |
          let .LINE = .LINE + 1
          if .LINE eq 14
```

```
        close unit 21
        print 1 line thus
```

```
        call vgotoxy.r(22,1)
        print 1 line thus
ENTER CHOICE: (N)EXT SCREEN, (Q)UIT :
        call vgotoxy.r(22,38)
        read .CHOICE
        let .CHOICE = fixed.f(.CHOICE,1)
        let .CHOICE = upper.f(.CHOICE)
        if .CHOICE eq "N"
          let .LINE = 0
          for .CLEAN = 7 to 22 do
            call vgotoxy.r(.CLEAN , 0)
            call vclearl.r
          loop " .CLEAN = 7 to 20
          call vgotoxy.r(7,0)
          open unit 21 for input, name = .NEW.FILE.NAME
            use unit 21 for input
          let .SKIP.REC = .CT.REC + 9
          skip .SKIP.REC records
        else
          let .CT.REC = COUNT.LAND
        always " .CHOICE eq N
      always
    loop " .CT.REC = 1 to COUNT.LAND
    close unit 21
    if .CHOICE eq "Q"
      call REPORT.GEN giving .ITER
      return
    always
```

```
      print 1 line thus
```

```
      call vgotoxy.r(22,1)

      print 1 line thus
ENTER CHOICE: (P)RINT LIST AGAIN; (Q)UIT :

      call vgotoxy.r(22,43)

      read .CHOICE

      let .CHOICE = fixed.f(.CHOICE,1)

      let .CHOICE = upper.f(.CHOICE)

      if .CHOICE eq "Q"

        call REPORT.GEN giving .ITER

        return

      always

    loop " .choice eq P

    return

end " arrivals.event.op
routine ASSIGN.ARR.ACFT.NAME

" This routine assigns the name to the aircraft just created randomly but
" confirming to the user specified population and respective percentages.
" The aircraft is also assigned its other respective attributes too.
"CALLED FROM : CREATE.AIRCRAFT.ARR
"CALLS: DET.APP.OCC.TIME, DET.RWY.OCC.TIME

  define .NO,

         .IARR,

         .ASSIGN,

         .EXI as integer variables

  define .APP.OCC.TIME,

         .ROT,

         .VELAPP,

         .DEC,

         .FREE.ROLL,

         .WING.SPAN,

         .DAIR as real variables
```

```
define .CUM.PERCENT.ARR as a integer 1-dimensional array

reserve .CUM.PERCENT.ARR as (NO.ARR.AIRCRAFT + 1)

let .CUM.PERCENT.ARR(1) = 0

"creating an array of cumulative percentages of arriving aircraft percent of
"creation

for .NO = 2 to (NO.ARR.AIRCRAFT + 1)  do

  let .CUM.PERCENT.ARR(.NO) = .CUM.PERCENT.ARR(.NO - 1) +
                              ARR.ACFT.PERCENT(.NO - 1)

loop

let .ASSIGN = randi.f(1,100, 5)

for .IARR = 1 to NO.ARR.AIRCRAFT  do

  if .ASSIGN gt .CUM.PERCENT.ARR(.IARR) and
                                .ASSIGN le .CUM.PERCENT.ARR(.IARR + 1)

    "Assign the name and the other attributes to the entity.

    NAME(ARR.ID(COUNT.ARR)) = ARR.ACFT.NAME(.IARR)

    let .DEC = ARR.ACFT.DEC(.IARR)

    let TYPE(ARR.ID(COUNT.ARR)) = ARR.ACFT.TYPE(.IARR)

    let .FREE.ROLL = ARR.ACFT.FREE.ROLL(.IARR)

    let .WING.SPAN = ARR.ACFT.WING.SPAN(.IARR)

    call DET.VAPP giving .IARR yielding .VELAPP

    let V.APP(ARR.ID(COUNT.ARR)) = .VELAPP

    call DET.DIST.AIR giving .IARR, .VELAPP yielding .DAIR

    call DET.APP.OCC.TIME giving .VELAPP  yielding .APP.OCC.TIME

    let APP.OCC.TIME(ARR.ID(COUNT.ARR)) = .APP.OCC.TIME

    call DET.RWY.OCC.TIME giving .DAIR, .DEC, .FREE.ROLL,
                        .VELAPP, .WING.SPAN, COUNT.ARR yielding .ROT,
                        .EXI

    let RWY.OCC.TIME(ARR.ID(COUNT.ARR)) = .ROT

    let ROT.STAT(.IARR) = ROT.STAT(.IARR) + .ROT

    let NO.IDENT.ARR(.IARR) = NO.IDENT.ARR(.IARR) + 1 " Collecting data
                " to know number of times  similar aircraft were
                " generated

    let EXIT.ASSIGNED(COUNT.ARR) = .EXI " storing data on exits assigned
                                  " to the landing aircraft
```

```
      let EXIT.ASSIGN(.IARR, .EXI) = EXIT.ASSIGN(.IARR, .EXI) + 1
                  " Collecting data on number of times similar aircraft
                  " is assigned to different exits

    always

  loop  " till the correct aircraft is identified

  return

end  " assign.arr.acft.name
 routine ASSIGN.DEP.ACFT.NAME

 "This routine assigns names to the aircraft created randomly but
 "confirming to user specified population and individual percentages.
 "Other related attributes are also specified to the entity.
 "CALLED FROM: CREATE.AIRCRAFT.DEP
 "CALLS:

 define .ID,

        .NO,

        .ASSIGN  as integer variables

 define .TAKEOFF.TT as a real variable

 define .CUM.PERCENT.DEP as an integer 1-dimensional array

 reserve .CUM.PERCENT.DEP as (NO.DEP.AIRCRAFT + 1)

 let .CUM.PERCENT.DEP(1) = 0

 for .NO = 2 to (NO.DEP.AIRCRAFT + 1) do

    let .CUM.PERCENT.DEP(.NO) = .CUM.PERCENT.DEP(.NO - 1) +
                          DEP.ACFT.PERCENT(.NO -1)

 loop " creating an array for cumulative perentages of departure aircraft

 let .ASSIGN = randi.f(1,100,7) " for random assignment of aircraft names to
                          "  newly created aircraft entity.

  for .ID = 1 to NO.DEP.AIRCRAFT do

     if .ASSIGN gt .CUM.PERCENT.DEP(.ID) and
                  .ASSIGN le .CUM.PERCENT.DEP(.ID + 1)

        NAME(DEP.ID(COUNT.DEP)) = DEP.ACFT.NAME(.ID)

        let TYPE(DEP.ID(COUNT.DEP)) = DEP.ACFT.TYPE(.ID)

        call DET.TAKEOFF.TIME given .ID yielding .TAKEOFF.TT

        let TAKEOFF.TIME(DEP.ID(COUNT.DEP)) = .TAKEOFF.TT

     always

  loop
```

```
  return

end "assign.dep.aircraft.name
routine ASSIGN.TABLE.OP given .ITER

" This routine displays assingment of aircraft to different exits
" CALLED FROM : REPORT.GEN
" CALLS: REPORT.GEN

  define .CHOICE,

       .EXT,

       .NEW.FILE.NAME,

       .FILE.NAME as text variables

  define .ROW,

       .COL,

       .ITER,

       .IEX,

       .IAR,

       .CLR as integer variables


  let .NEW.FILE.NAME = substr.f(APP.FILE.NAME,1,7)

  let .EXT = itot.f(.ITER)

  let .NEW.FILE.NAME = concat.f( .NEW.FILE.NAME,"A.", .EXT)

  let .CHOICE = "V"

  until .CHOICE eq "O", do

    call vclears.r

    if .CHOICE eq "P" or .CHOICE eq "S"

      open unit 13 for output, name = "ASSIGN.RSM"

      use unit 13 for output

      let page.v = 1

      begin report

    always

   if TOT.ARR.ACFT gt 0

    print 2 lines thus
                        ASSIGNMENT TABLE
```

```
    print 1 line thus
AIRCRAFT   :                    EXIT NO.

    write as "NAME       :"

    for .IEX = 1 to NO.EXITS, write .IEX as s 3, i 2

    start new output line

    print 1 line thus
-------------------------------------------------------

    for .IAR = 1  to  NO.ARR.AIRCRAFT do

      write ARR.ACFT.NAME(.IAR) as  t 12

      for .IEX = 1 to NO.EXITS, write EXIT.ASSIGN(.IAR,.IEX) as s 1, i 4

      start new output line

      print 1 line thus
-------------------------------------------------------------------

      call vgetxy.r yielding .ROW, .COL

      if .CHOICE ne "P" and .ROW ge 20 and .COL ge 0

        call vgotoxy.r(22,1)

        call vbcolor.r(5)

        call vfcolor.r(15)

        print 1 line  thus
 HIT ANY KEY TO VIEW REST OF THE ASSIGNMENT TABLE:

        call vgotoxy.r(22,51)

        call rcr.r

        read  as / using unit 5

        call vbcolor.r(3)

        call vfcolor.r(0)

        for .CLR = 5 to 22 do

          call vgotoxy.r(.CLR, 0)

          call vclearl.r

        loop " until all the lines below 6 are cleared

        call vgotoxy.r(5,0)

      always

    loop  " until all the data is printed
```

```
    if .CHOICE eq "P" or .CHOICE eq "S"
      end " report section
    let  .FILE.NAME = .NEW.FILE.NAME
      if .CHOICE eq "P"
         close unit 13
      else
          close unit 13, name = .FILE.NAME
      always
" COMMANDS TO PRINT THE FILE ON PRINTER THROUGH DOS COMMANDS
     if .CHOICE eq "P"
      let GT.1 = "CD HIGH2"
      SYS$DO.X(11)
      let GT.1 = "COPY ASSIGN.RSM LPT1"
      SYS$DO.X(11)
      let GT.1 = "CD.."
      SYS$DO.X(11)
    always
   always
    call vgotoxy.r(22,1)
    call vbcolor.r(3) "cyan background
  always
  if TOT.ARR.ACFT eq 0
    print 2 lines thus
                ARRIVALS ARE NOT SIMULATED
                 RETURN TO (O)UTPUT MENU
  always
    call vbcolor.r(5)
    call vfcolor.r(15)
    print 1 line thus
ENTER - (P)RINT; (S)AVE; (V)IEW AGAIN; (O)UTPUT MENU :
    call vgotoxy.r(22,57)
    read .CHOICE
```

```
    call vbcolor.r(3)

    call vfcolor.r(0)

    let .CHOICE = upper.f(.CHOICE)

    let .CHOICE = fixed.f(.CHOICE,1)

    if .CHOICE eq "O"

      call REPORT.GEN giving .ITER

      return

    always

  loop

  return

end " assign.table.op
routine CHECK.PRECEDE.ACFT.CLR.RWY  yielding .R.T.LAG.THRLD.RWY

" This routine checks if the preceding acft clears runway when the lagging
" aircraft reaches threshold.
" CALLED FROM: PRECEDE.FASTER.ACFT, PRECEDE.SLOWER.ACFT
" CALLS:


  define .T.PRECEDE.ACFT.CLR.APP,

       .TOT.T.PRECEDE.ACFT.CLR.RWY,

       .R.T.LAG.THRLD.RWY as real variables

  let .T.PRECEDE.ACFT.CLR.APP = APP.OCC.TIME(ARR.ID(COUNT.APP)) -
                    (trunc.f(time.v - T.DEP.STACK(COUNT.APP)) * 60 +
                     frac.f(time.v - T.DEP.STACK(COUNT.APP)) * 60 )
                " Remaining time for preceding aircraft to clear approach

  let .TOT.T.PRECEDE.ACFT.CLR.RWY = (.T.PRECEDE.ACFT.CLR.APP +
                                RWY.OCC.TIME(ARR.ID(COUNT.APP)))
                " Total time for the preceding aircraft to travel from
                " its present position till it clears runway.

  if .TOT.T.PRECEDE.ACFT.CLR.RWY le APP.OCC.TIME(ARR.ID(COUNT.APP + 1))

    .R.T.LAG.THRLD.RWY = 0 "Difference in time to perform the two operations
                    "1: Time for precede acft to clear rwy
                    "2: Time for lagging acft to reach threshold

  else

    .R.T.LAG.THRLD.RWY = (.TOT.T.PRECEDE.ACFT.CLR.RWY -
                    APP.OCC.TIME(ARR.ID(COUNT.APP + 1)))

  always

  return
```

```
 end "check.precede.acft.clr.rwy
event CREATE.AIRCRAFT.ARR

" This routine creates arriving aircraft at a specified time.
" CALLED FROM : GENERATOR
" CALLS: ASSIGN.ARR.NAME, SYS.CHECK.AT.STACK


   if  COUNT.ARR lt TOT.ARR.ACFT

     COUNT.ARR = COUNT.ARR + 1

     create an AIRCRAFT called ARR.ID(COUNT.ARR)

     T.ENTER.STACK(COUNT.ARR) = TIME.V

     let OPERATION(ARR.ID(COUNT.ARR)) = ..LANDING

     call ASSIGN.ARR.ACFT.NAME

     if COUNT.ARR  le (COUNT.APP + 1)

       call SYS.CHECK.AT.STACK

     always

   else

     return

   always

   return

 end " create.aircraft.arr
event CREATE.AIRCRAFT.DEP

" The departure aircraft are created in this event routine as scheduled in
" the generator
" CALLED FROM : GENERATOR
" CALLS; ASSIGN.DEP.ACFT.NAME, SYS.CHECK.AT.THRLD

  if COUNT.DEP lt TOT.DEP.ACFT

    let COUNT.DEP = COUNT.DEP + 1

    create an AIRCRAFT called DEP.ID(COUNT.DEP)

    let T.ENTER.RAMP(COUNT.DEP) = time.v

    OPERATION(DEP.ID(COUNT.DEP)) = ..DEPARTURE

    call ASSIGN.DEP.ACFT.NAME

    if COUNT.DEP le (COUNT.T.OFF + 1)

      call SYS.CHECK.AT.THRLD

    always
```

```
  else

    return

  always

    return

end " create.aircraft.dep
routine DEP.POP.OP given .ITER

" This routine displays the departure aircraft used for the simulation .
" CALLED FROM: INPUT.DATA.OP
" CALLS:


  define .ITER,

        .ANO,

        .XLOC,

        .YLOC as integer variables

  define .CHOICE as a text variable

  until .CHOICE eq "Q", do

    call vclears.r

    print 3 lines thus
                ╔═══════════════════════════════════════╗
                ‖ DEPARTURE AIRCRAFT POPULATION ‖
                ╚═══════════════════════════════════════╝

    skip 1 line

    print 1 line with APP.FILE.NAME thus
  APP. FILE NAME : **************


    print 1 line with TOT.DEP.ACFT thus
  NO. OF DEPARTURES : *******

    print 1 line thus
```

```
    let .XLOC = 7

    let .YLOC = 3

    for .ANO = 1 to NO.DEP.AIRCRAFT,  do

      call vgotoxy.r(.XLOC, .YLOC)

      print 1 line with .ANO, DEP.ACFT.NAME(.ANO), DEP.ACFT.PERCENT(.ANO) thus
| **) ************ (** %) |

      let .XLOC = .XLOC + 1
```

```
      if .XLOC eq 20
        let .XLOC = 7
        let .YLOC = .YLOC + 24
      always ".XLOC eq 20
    loop " .ANO = 1 to NO.DEP.AIRCRAFT
    call vgotoxy.r(20,0)
    print 1 line thus
```

```
    call vgotoxy.r(22,1)
    call vbcolor.r(5)
    call vfcolor.r(15)
    print 1 line thus
ENTER (PRINT SCREEN) KEY TO PRINT; (Q)UIT:
    call vgotoxy.r(22,43)
    read .CHOICE
    call vbcolor.r(3)
    call vfcolor.r(0)
    let .CHOICE = fixed.f(.CHOICE, 1)
    let .CHOICE = upper.f(.CHOICE)
  loop " .CHOICE eq "Q"
  if .CHOICE eq "Q"
     call INPUT.DATA.OP giving .ITER
  always " .CHOICE eq "Q"
  return
end " dep.pop.op
  process DEPARTURE.OPERATION
  " The takeoff operation is activated as scheduled and then the aircraft
  " departs the system after its takeoff time.
  " CALLED FROM: SYS.CHECK.AT.THRLD, STATUS.APP.CLR.RWY.OCC,
  "              STATUS.APP.OCC.RWY.CLR, STATUS.APP.OCC.RWY.OCC
  " CALLS:
   define .EXT,
        .NEW.FILE.NAME as text variables
```

```
define .ACFT.ID as a pointer variable

let .NEW.FILE.NAME = substr.f(APP.FILE.NAME,1,7)

let .EXT = itot.f(RUN.NO)

let .NEW.FILE.NAME = concat.f( .NEW.FILE.NAME,"D.", .EXT)

if COUNT.T.OFF  lt  TOT.DEP.ACFT

    request 1 RUNWAY(1)

    if COUNT.T.OFF lt COUNT.DEP

     let COUNT.T.OFF = COUNT.T.OFF + 1

     let OCCUPANT.OPERATION = OPERATION(DEP.ID(COUNT.T.OFF))

     let T.DEP.RAMP(COUNT.T.OFF) = TIME.V

    let TOT.DEP.DELAY = TOT.DEP.DELAY + (T.DEP.RAMP(COUNT.T.OFF) -
                        T.ENTER.RAMP(COUNT.T.OFF))

     if COUNT.T.OFF ne COUNT.DEP and (COUNT.T.OFF + 1) le COUNT.DEP

      call SYS.CHECK.AT.THRLD

     always " (COUNT.T.OFF + 1) le COUNT.DEP

"A REPORT OF DEPARTURE EVENTS LIST IS CREATED BELOW

     if COUNT.T.OFF eq 1

      open unit 15 for output, name = .NEW.FILE.NAME

      use unit 15 for output

      let page.v = 1

      begin report on a new page

       begin heading

        if page is first

          print 2 lines thus
                DEPARTURE EVENT LIST
                ------------------

          skip 1 line

          print 1 line with APP.FILE.NAME, RUN.NO thus
 APPLICATION FILE NAME: ***********              RUN NO.: *****

          skip 1 line

          print 4 lines thus
 AIRCRAFT        CREATION        EMPLANE         DELAY
  NAME           TIME            TIME
```

```
                    Min             Min             Min
-------------------------------------------------------

          always "page is first

         end " heading section

      always " COUNT.T.OFF eq 1

      if COUNT.T.OFF gt 1

        open unit 15 for output, name = .NEW.FILE.NAME, append

         use unit 15 for output

      always " COUNT.T.OFF gt 1

      write NAME(DEP.ID(COUNT.T.OFF)),
            (T.ENTER.RAMP(COUNT.T.OFF)),
             (T.DEP.RAMP(COUNT.T.OFF)), ( T.DEP.RAMP(COUNT.T.OFF) -
              T.ENTER.RAMP(COUNT.T.OFF)) as s 2, t 12, s 10, d(6,2),
               s 12, d(6,2), s 10, d(7,2), /

      if COUNT.T.OFF eq TOT.DEP.ACFT

        end " report section

      always " COUNT.T.OFF eq TOT.DEP.ACFT

      close unit 15

      let .ACFT.ID = COUNT.T.OFF

      work TAKEOFF.TIME(DEP.ID(COUNT.T.OFF)) seconds

    always "COUNT.T.OFF lt COUNT.DEP

    relinquish 1 RUNWAY(1)

    if .ACFT.ID eq TOT.DEP.ACFT

      END.DEP.SIM = time.v

    always

    if COUNT.T.OFF ne COUNT.DEP and ( COUNT.T.OFF + 1) le COUNT.DEP

      call SYS.CHECK.AT.THRLD

    always

  always " COUNT.T.OFF lt TOT.DEP.ACFT

  return

end " departure.operation
routine DEPARTURES.EVENT.OP given .ITER

"This routine prints on the screen the departure event list.
```

```
"CALLED FROM: REPORT.GEN
"CALLS:
"OPENS(FOR INPUT) : .NEW.FILE.NAME

   define .NEW.FILE.NAME,

          .EXT,

          .CHOICE,

          .NAME as text variables

   define .ITER,

          .CT.REC,

          .LINE,

          .CLEAN,

          .SKIP.REC as integer variables

   define .CR.TIME,

          .TOFF.TIME,

          .DEL as real variables

    let .NEW.FILE.NAME = substr.f(APP.FILE.NAME,1,7)

    let .EXT = itot.f(.ITER)

    let .NEW.FILE.NAME = concat.f( .NEW.FILE.NAME,"D.", .EXT)

    let .CHOICE = "P"

    until .CHOICE eq "Q", do

      call vclears.r

      print 3 lines with APP.FILE.NAME, .ITER thus
┌──────────────────────────────────DEPARTURE EVENT LIST──────────────────────────────────┐
| APP. FILE: ************ ──────────────────────────────── RUN NO.: ****        |
├──────────────────────────────────────────────────────────────────────────────────────┤

      print 4 lines thus
|   AIRCRAFT          CREATION        EMPLANE          DELAY       |
|     NAME            TIME          TIME                 |
|                   Min           Min           Min       |
├──────────────────────────────────────────────────────────────────────────────────────┤

      open unit 22 for input, name = .NEW.FILE.NAME

       use unit 22 for input

       skip 10 records

       for .CT.REC = 1 to COUNT.T.OFF, do
```

```
        read .NAME, .CR.TIME, .TOFF.TIME, .DEL

        print 1 line with .NAME, .CR.TIME, .TOFF.TIME, .DEL thus
|   ************      ******.**      *******.**      *******.**      |
        let .LINE = .LINE + 1

        if .LINE eq 14

          close unit 22

          print 1 line thus
└──────────────────────────────────────────────────────────────────────┘

          call vgotoxy.r(22,1)

          print 1 line thus
ENTER CHOICE: (N)EXT SCREEN, (Q)UIT :

          call vgotoxy.r(22,38)

          read .CHOICE

          let .CHOICE = fixed.f(.CHOICE,1)

          let .CHOICE = upper.f(.CHOICE)

          if .CHOICE eq "N"

            let .LINE = 0

            for .CLEAN = 7 to 22 do

              call vgotoxy.r(.CLEAN , 0)

              call vclearl.r

            loop " .CLEAN = 7 to 20

            call vgotoxy.r(7,0)

            open unit 22 for input, name = .NEW.FILE.NAME

              use unit 22 for input

              let .SKIP.REC = .CT.REC + 9

              skip .SKIP.REC records

          else

            let .CT.REC = COUNT.T.OFF

          always " if .CHOICE eq N

        always ".LINE eq 15

      loop " .CT.REC = 1 to COUNT.T.OFF
```

```
        close unit 22

        if .CHOICE eq "Q"

          call REPORT.GEN giving .ITER

          return

        always " .CHOICE eq Q

        print 1 line thus
```

```
        call vgotoxy.r(22,1)

        print 1 line thus
ENTER CHOICE: (P)RINT LIST AGAIN; (Q)UIT :

        call vgotoxy.r(22,43)

        read .CHOICE

        let .CHOICE = fixed.f(.CHOICE,1)

        let .CHOICE = upper.f(.CHOICE)

    loop " .choice eq Q

    if .CHOICE eq "Q"

        call REPORT.GEN giving .ITER

    always " .CHOICE eq Q

    return

end " departures.event.list
routine DET.APP.OCC.TIME given .VELAPH yielding .R.A.O.TIME

" This routine determines the time to traverse the final approach till it
" reaches threshold.
" CALLED FROM: ASSIGN.ARR.ACFT.NAME
" CALLS:

 define .R.A.O.TIME,

        .VELAPH as real variables

 let .R.A.O.TIME = LENGTH.APP(1) / .VELAPH

 return

end
 routine DET.DEL.ARR yielding .R.DEL.ARR

 "This routine is called from STATUS.APP.OCC.RWY.CLR or STATUS.APP.OCC.
 "RWY.OCC to determine the delay if any due to an arriving aircraft.
 "CALLED FROM; STATUS.APP.OCC.RWY.CLR, STATUS.APP.OCC.RWY.OCC
 "CALLS: DET.SEP.DATA
```

```
define .TYPE.DEP,
       .TYPE.ARR,
       .LEAD.ACFT.APP as integer variables
define .MIN.SEP.ARR.DEP,
       .TIME.SPENT.APP,
       .TIME.REM.APP,
       .DIST.COVERED.APP,
       .DIST.TO.THRLD,
       .R.DEL.ARR as real variables
let .TYPE.DEP = TYPE(DEP.ID(COUNT.T.OFF + 1))
let .LEAD.ACFT.APP = COUNT.LAND + 1
let .TYPE.ARR = TYPE(ARR.ID(.LEAD.ACFT.APP))
let .MIN.SEP.ARR.DEP = DEP.ARR.SEP(.TYPE.ARR, .TYPE.DEP) + INTDEP.ARR.BUFF
let .TIME.SPENT.APP = trunc.f(time.v - T.DEP.STACK(.LEAD.ACFT.APP)) * 60 +
                      frac.f(time.v - T.DEP.STACK(.LEAD.ACFT.APP)) * 60
                        " seconds
let .TIME.REM.APP = APP.OCC.TIME(ARR.ID(.LEAD.ACFT.APP)) - .TIME.SPENT.APP
                  " Remaining time in final approach before reaching the
                  " runway by the arriving aircraft nearest to the runway
let .DIST.COVERED.APP = .TIME.SPENT.APP * V.APP(ARR.ID(.LEAD.ACFT.APP))
let .DIST.TO.THRLD = LENGTH.APP(FINAL.APPROACH) - .DIST.COVERED.APP
if .DIST.TO.THRLD ge .MIN.SEP.ARR.DEP
  .R.DEL.ARR = 0.0 " seconds
else
  .R.DEL.ARR = .TIME.REM.APP  "seconds
always
return
end " det.del.arr
routine DET.DEL.DEP yielding .T.MIN.SEP.TAKEOFF

" This routine determines the delay due to preceding aircraft departure
" and time to achieve minimum seperation for inter departures.
" CALLED FROM : STATUS.APP.CLR.RWY.OCC, STATUS.APP.OCC.RWY.OCC
" CALLS : DET.SEP.DATA
```

```
 define .PRECEDE.DEP.STRT.TIME,

       .MIN.T.SEP.TAKEOFF,

       .T.MIN.SEP.TAKEOFF as real variables

 define .PRECEDE.TYPE,

       .FLLWNG.TYPE as integer variables

if COUNT.T.OFF lt TOT.DEP.ACFT

 let .PRECEDE.TYPE = TYPE(DEP.ID(COUNT.T.OFF))

 let .FLLWNG.TYPE = TYPE(DEP.ID(COUNT.T.OFF + 1))

 let .PRECEDE.DEP.STRT.TIME = T.DEP.RAMP(COUNT.T.OFF) " departure time of
                                        "of preceding aircraft

 let .MIN.T.SEP.TAKEOFF = SEP.T.TAKEOFFS(.FLLWNG.TYPE, .PRECEDE.TYPE) +
                          INTDEP.BUFF

 let .T.MIN.SEP.TAKEOFF = .MIN.T.SEP.TAKEOFF -
                          (trunc.f(time.v - .PRECEDE.DEP.STRT.TIME) * 60 +
                           frac.f(time.v - .PRECEDE.DEP.STRT.TIME) * 60) +
                           INTDEP.BUFF
                              "time to achieve minimum seperation between
                              "succesive takeoffs from present time

 always

 return

end  "det.del.dep
 routine DET.DIST.AIR given .DRR, .VELAPP yielding .DAIR

 "This routine calculates the distance the aircraft travels in air from
 " threshold to touchdown.
 "CALLED FROM: ASSIGN.DEP.ACFT.NAME
 "CALLS:

 define .VELAPP,

       .GAMMA,

       .GAMMA.NOM,

       .MAX.GAMMA,

       .MIN.GAMMA,

       .GAMMA.STD,

       .ACC.GRA,

       .NFLARE,

       .VFLARE,
```

```
      .HT.THRLD,
      .HT.THRLD.NOM,
      .MAX.HT.THRLD,
      .MIN.HT.THRLD,
      .HT.THRLD.STD,
      .DAIR as real variables
define .DRR as a integer variable
let .GAMMA.NOM = 0.0523  "Descent flight path angle used to estimate the
                         "airborne portion of the landing path
" New procedure to estimate  gamma as a random variable
let .GAMMA.STD = .GAMMA.NOM * 0.07
let .MAX.GAMMA = .GAMMA.NOM + 3.5 * .GAMMA.STD
let .MIN.GAMMA = .GAMMA.NOM - 3.5 * .GAMMA.STD
" Choose a candidate flight path angle among those permissible
until .GAMMA le .MAX.GAMMA  and .GAMMA ge .MIN.GAMMA do
  let .GAMMA =  normal.f(.GAMMA.NOM, .GAMMA.STD , 9)
loop
" End of new procedure to estimate GAMMA (flight path angle)
let .NFLARE = 1.15   "Landing flare load factor (in g's)
let .ACC.GRA = 9.81  "Acceleration due to gravity (m/s-s)
if ARR.ACFT.TYPE(.DRR) eq ..HEAVY
 .HT.THRLD.NOM = 15.0   "Runway threshold crossimg altitiude(meters)
else
   if ARR.ACFT.TYPE(.DRR) EQ ..LARGE
      .HT.THRLD.NOM = 15.0
   else
      .HT.THRLD.NOM = 10.0
   always
always
" New procedure to estimate HTRLD as a normally distributed random variable
" HT.HTRLD.NOM is the nominal threshold crossing heigth
```

```
 let .HT.THRLD.STD = .HT.THRLD.NOM * .10

 let .MAX.HT.THRLD = .HT.THRLD.NOM + 3.5 * .HT.THRLD.STD

 let .MIN.HT.THRLD = .HT.THRLD.NOM - 3.5 * .HT.THRLD.STD

 until .HT.THRLD le .MAX.HT.THRLD and .HT.THRLD ge .MIN.HT.THRLD do

  let .HT.THRLD = normal.f(.HT.THRLD.NOM, .HT.THRLD.STD, 4)

 loop


" open unit 24 for output, name = "GHT.DAT", append

" use unit 24 for output

" print 1 line with .GAMMA , .HT.THRLD thus
" *****.*****  ******.*****

" close unit 24

" End of new procedure

 let .VFLARE = 0.95 * .VELAPP

 let .DAIR = .HT.THRLD / .GAMMA + (.VFLARE ** 2 * .GAMMA) /
                                  ( 2 * .ACC.GRA * (.NFLARE - 1))
                    " Distance traveled in air after crossing threshold
                    " and before touchdown.
 return

end "det.dist.air
routine DET.EXIT.SPEEDS

" This routine determines the exit speeds of the exits chosen based on the
" type of the exits.
" CALLED FROM: MAIN
" CALLS:

 define .ES as an integer variable

 reserve EXIT.SPEED as NO.EXITS

 for .ES = 1 to NO.EXITS do

   select case EXIT.CHOICE(.ES)

     case 1

      let EXIT.SPEED(.ES) = 8.00  " meters/sec
              " 90 deg FAA STANDARD

     case 2

      let EXIT.SPEED(.ES) = 15.00
             " 45 DEG FAA STANDARD

     case 3
```

```
      let EXIT.SPEED(.ES) = 26.00
            " 30 DEG FAA STANDARD

      case 4

      let EXIT.SPEED(.ES) = 26.00
            " 30 DEG IMPROVED

      case 5

      let EXIT.SPEED(.ES) = 17.00
            " WIDE THROAT

      case 6

      let EXIT.SPEED(.ES) = 35.0
            " 30 DEG REDIM - HEAVY(CRITICAL ACFT)

      case 7

      let EXIT.SPEED(.ES) = 35.0
            " 20 DEG REDIM - HEAVY(CRITICAL ACFT)

      case 8

      let EXIT.SPEED(.ES) = 35.0
            " 30 DEG REDIM - LARGE (CRITICAL ACFT)

      case 9

      let EXIT.SPEED(.ES) = 35.0
            " 20 DEG REDIM - LARGE(CRITICAL ACFT)

      case 10

      let EXIT.SPEED(.ES) = 35.0
            " 30 DEG REDIM - SMALL (CRITICAL ACFT)

      default

      let EXIT.SPEED(.ES) = 8.00

    endselect " EXIT.CHOICE(.ES)

  loop " until .ES = NO.EXITS

  return

end " det.exit.speeds
routine DET.RWY.OCC.TIME  given .DAIR, .DEC, .TT.FREE.ROLL, .VELAPP, .WING.SPAN,
                          .RCARR  yielding .RWYOT, .EXIT.CHOSEN

" This routine determines the runway occupancy time of the aircraft in
" question.It also calls routine DET.SELECT.EXIT to select the exit
" and the occupancy time of that exit.
" CALLED FROM: ASSIGN.ARR.ACFT.NAME
" CALLS: DET.SELECT.EXIT
```

```
define .RCARR,
        .EXIT.CHOSEN  as integer variable
define .DAIR,
        .VELAPP,
        .VEL.TD,
        .DEC,
        .DEC.STD,
        .MAX.DEC,
        .MIN.DEC,
        .LAND.DEC,
        .TT.FREE.ROLL,
        .INIT.FREE.ROLL.DIST,
        .NOMINAL.PT,
        .DECISION.PT,
        .DIST.DECISION.EXIT,
        .VEL.DECISION.PT,
        .DEC.AFTER.DECISION,
        .TT.TD,
        .TT.TD.DECISION.PT,
        .TT.AFTER.DECISION,
        .DEC.FINAL.ROLL,
        .DIST.FINAL.ROLL,
        .VEL.AT.EXIT,
        .MIN.EXIT.VEL,
        .MOD.EXIT.VEL,
        .TT.FINAL.ROLL,
        .TT.EXIT,
        .RWYOT,
        .WING.SPAN  as real variables
```

```
   let .VEL.TD = (1.15/1.3) * .VELAPP
                     " Velocity at touchdown

   let .DEC.STD = 0.07 * .DEC
                     " Standard deviation of the deceleration

   let .MAX.DEC = .DEC + 3.5 * .DEC.STD
                     " Upper limit of the deceleration

   let .MIN.DEC = .DEC - 3.5 * .DEC.STD
                     " Lower limit of the deceleration

   " Choosing a candidate deceleration value between upper and lower limit

   until .LAND.DEC le .MAX.DEC  and .LAND.DEC ge .MIN.DEC do

     let .LAND.DEC = normal.f(.DEC, .DEC.STD, 8)
                     " Deceleration after landing

   loop

   let .INIT.FREE.ROLL.DIST = .TT.FREE.ROLL * .VEL.TD  "Initial
                    " free Roll distance immediately after touchdown

   call DET.SELECT.EXIT giving .RCARR, .VEL.TD, .LAND.DEC, .DAIR,
                  .INIT.FREE.ROLL.DIST yielding .NOMINAL.PT, .EXIT.CHOSEN

   let .DECISION.PT = (.NOMINAL.PT - 3 * .VEL.TD )
                  " This is a point w.r.t. threshold
                  " where a decision is made to adjust the deceleration
                  " so as to achieve the exit velocity slightly ahead
                  " of the exit it has selected.

  " Printing nominal point

"  open unit 22 for output, name = "NOMP.DAT", append

"  use unit 22 for output

"  print 1 line with NAME(ARR.ID(.RCARR)), .NOMINAL.PT, .LAND.DEC thus
"  ************  *******.****  ******.****

"  close unit 22

  " End printing procedure

   let .VEL.DECISION.PT = sqrt.f(.VEL.TD. ** 2 -
                      2.0 * .LAND.DEC * (.DECISION.PT -
                          .DAIR - .INIT.FREE.ROLL.DIST ))
                    " Velocity at the decision point

   let .DIST.DECISION.EXIT = EXIT.LOCATION(.EXIT.CHOSEN) - .DECISION.PT
                  " Distance between the decision point and the exit
                  " chosen.

   let .DEC.AFTER.DECISION = (.VEL.DECISION.PT ** 2 -
                        EXIT.SPEED(.EXIT.CHOSEN) ** 2)/(2.0 * (3/4) *
                          .DIST.DECISION.EXIT)
                 "Deceleration after decision point till the threefourth
```

```
                " of the distance between the decision point and exit
                "and attaining the exit speed.

let .TT.TD = .DAIR / (( .VELAPP + .VEL.TD ) / 2.0)
                " Travel time from threshold to touchdown

let .TT.TD.DECISION.PT = (.VEL.TD - .VEL.DECISION.PT) / .LAND.DEC
                " Travel time from touchdown to decision point

let .TT.AFTER.DECISION = (.VEL.DECISION.PT - EXIT.SPEED(.EXIT.CHOSEN)) /
                        .DEC.AFTER.DECISION
                " Travel time decision point till it achieves the
                " exit speed.

let .DEC.FINAL.ROLL = 0.375  " Deceleration during final roll - m/s-s

let .DIST.FINAL.ROLL = (1/4) * .DIST.DECISION.EXIT
                " Distance to cover during final roll

let .MIN.EXIT.VEL = 8.0
                " Minimum allowable exit velocity

let .MOD.EXIT.VEL = (EXIT.SPEED(.EXIT.CHOSEN)) ** 2 -
                    (2.0 * .DEC.FINAL.ROLL * .DIST.FINAL.ROLL)

if .MOD.EXIT.VEL gt 0

  .VEL.AT.EXIT = sqrt.f(.MOD.EXIT.VEL)
                " Velocity of aircraft at the exit
                " if it decelerates with .DEC.FINAL.ROLL
else

  .VEL.AT.EXIT = .MIN.EXIT.VEL

always

if .VEL.AT.EXIT lt .MIN.EXIT.VEL

   .VEL.AT.EXIT = .MIN.EXIT.VEL " if velocity achieved is less than the
                          " minimum exit prescribed velocity
                          " adjust velocity at exit to that min.

    let .DEC.FINAL.ROLL = ((EXIT.SPEED(.EXIT.CHOSEN)) ** 2 -
                    (.VEL.AT.EXIT) ** 2 ) / ( 2.0 * .DIST.FINAL.ROLL)
                  " Adjusted deceleration during final roll

always

if EXIT.SPEED(.EXIT.CHOSEN) eq .VEL.AT.EXIT

   .TT.FINAL.ROLL = .DIST.FINAL.ROLL / .VEL.AT.EXIT
                "travel time with uniform velocity during final roll

else

  .TT.FINAL.ROLL = (EXIT.SPEED(.EXIT.CHOSEN) - .VEL.AT.EXIT) /
                     .DEC.FINAL.ROLL
                  " Travel time during final roll
always
```

```
    call DET.TOT giving .EXIT.CHOSEN, .VEL.AT.EXIT, .WING.SPAN yielding
                .TT.EXIT

    let .RWYOT = (.TT.TD + .TT.FREE.ROLL + .TT.TD.DECISION.PT +
                .TT.AFTER.DECISION + .TT.FINAL.ROLL + .TT.EXIT )
                    " Runway occupancy time

"   open unit 31 for output, name = "rolldist.dat", append

"   use unit 31 for output

"   print 1 line with .DAIR, .INIT.FREE.ROLL.DIST, 3/4 * .DIST.DECISION.EXIT,
"    1/4 * .DIST.DECISION.EXIT, .DECISION.PT, .NOMINAL.PT thus
" *****.**  *****.**  *****.**  *****.**  ****.**  *****.**

"   close unit 31

"   open unit 30 for output, name = "roll.dat", append

"   use unit 30 for output

"   print 1 line with  NAME(ARR.ID(.RCARR)), .TT.TD, .TT.FREE.ROLL,
"   .TT.TD.DECISION.PT, .TT.AFTER.DECISION, .TT.FINAL.ROLL, .TT.EXIT,
"    .VEL.AT.EXIT, .EXIT.CHOSEN thus
"*********** **.*** **.** **.*** **.*** **.*** **.*** **.*** **

"   close unit 30

  return

end " det.rwy.occ.time
routine DET.SELECT.EXIT given .IRR, .VTD, .LDEC, .TDPT, .INFROLL yielding
                .NO.PT, .EXCHOSEN

" This routine selects the exit for the aircraft in question and also the
" turnoff time.
"CALLED FROM: DET.RWY.OCC.TIME
"CALLS:

  define .VTD,

        .LDEC,

        .TDPT,

        .INFROLL,

        .FINAL.VEL,

        .DIST.TRAVELED,

        .NO.PT as real variables

  define .IRR,

        .IEX,

        .EXCHOSEN as integer variables
```

```
  for .IEX = 1 to NO.EXITS  do

    let .FINAL.VEL = EXIT.SPEED(.IEX)

    let .DIST.TRAVELED = (.VTD ** 2 - .FINAL.VEL ** 2)/(2.0 * .LDEC)
                    " Distance covered to decelerate from velocity at
                    " touchdown to the exit speed of the exit in question.

    let .NO.PT = .TDPT + .INFROLL + .DIST.TRAVELED

    if .NO.PT le EXIT.LOCATION(.IEX)

        .EXCHOSEN = .IEX

         return

    always

    if .IEX eq NO.EXITS

      call vclears.r

      call vgotoxy.r(18,0)

      call DO.BEEP

      call vbcolor.r(12)
      print 2 lines with NAME(ARR.ID(.IRR)) thus
THE ************* AIRCRAFT CANNOT LAND:       INSUFFICIENT RUNWAY LENGTH.
  QUITTING THE SIMULATION.----------TYPE 'QUIT' AT THE PROMPT

      call vbcolor.r(1)
    always

  loop

  return

end " det.select.exit
routine DET.TAKEOFF.TIME given .RID yielding .TAKEOFF.TTR

" This routine determines the takeoff time of the aircraft depending
" on the type it belongs
" CALLED FROM: ASSIGN.DEP.ACFT.NAME
" CALL:

   define .RID as an integer variable

   define .TAKEOFF.TTR as a real variable

   if DEP.ACFT.TYPE(.RID) eq ..HEAVY

      .TAKEOFF.TTR =  35 "SECONDS

   always

   if DEP.ACFT.TYPE(.RID) eq ..LARGE

      .TAKEOFF.TTR = 30 "SECONDS
```

```
  always

  if DEP.ACFT.TYPE(.RID) eq ..SMALL

    .TAKEOFF.TTR = 25 "SECONDS

  always

  return

end " det.takeoff.time
routine DET.TOT given .EXIT.NO, .ACFT.SPEED, .WING.SPAN yielding .TOT
" This routine calculates the turnoff time for a given aircraft and exitning
" speed. It opens TOT.DATA file and reads the data required i.e. intercept,
" runway width coefficient and acft wing span coefficient and then interpolates
" the data to calculate TOT.
" CALLED FROM: DET.RWY.OCC.TIME
" CALLS:


 define .EXIT.NO,

      .READ.NO,

      .NO.OF.SPEEDS,

      .STEP as integer variables

 define .ACFT.SPEED,

      .TOT,

      .FACT,

      .INPT,

      .RW.CF,

      .ACFWS.CF,

      .WING.SPAN as real variables

 define .SPEED.INC,

      .INTERCEPT,

      .RW.COEFF,

      .ACFWS.COEFF as 1-dimensional real arrays


 open unit 9 for input, name = "TOT.DAT"

  use unit 9 for input

  let eof.v = 1

  while mode is alpha do
```

```
  start new input record
loop
until .READ.NO eq EXIT.CHOICE(.EXIT.NO), do
  skip 1 input record
  read .READ.NO
loop
read .NO.OF.SPEEDS
reserve .SPEED.INC,
        .INTERCEPT,
        .RW.COEFF,
        .ACFWS.COEFF as .NO.OF.SPEEDS
let .STEP = 1
until .STEP gt .NO.OF.SPEEDS, do
  read .READ.NO, .SPEED.INC(.STEP), .INTERCEPT(.STEP), .RW.COEFF(.STEP),
       .ACFWS.COEFF(.STEP)
  let .STEP = .STEP + 1
loop
close unit 9
let .STEP = 1
if EXIT.CHOICE(.EXIT.NO) eq 1
  .TOT = .INTERCEPT(.STEP) + .RW.COEFF(.STEP) * RWY.WIDTH +
         .ACFWS.COEFF(.STEP) * .WING.SPAN
 else
   for .STEP = 1 to  (.NO.OF.SPEEDS - 1), do
    if .ACFT.SPEED ge .SPEED.INC(.STEP) and
                  .ACFT.SPEED le .SPEED.INC(.STEP + 1)
      .FACT = (( .ACFT.SPEED - .SPEED.INC(.STEP)) /
              ( .SPEED.INC(.STEP + 1) - .SPEED.INC(.STEP)))
      let .INPT = .FACT * ( .INTERCEPT(.STEP + 1) - .INTERCEPT(.STEP)) +
                  .INTERCEPT(.STEP)
      let .RW.CF = .FACT * ( .RW.COEFF(.STEP + 1) - .RW.COEFF(.STEP)) +
                  .RW.COEFF(.STEP)
      let .ACFWS.CF = .FACT * (.ACFWS.COEFF(.STEP + 1) -
```

```
                    .ACFWS.COEFF(.STEP)) + .ACFWS.COEFF(.STEP)

        let .TOT = .INPT + .RW.CF * RWY.WIDTH + .ACFWS.CF * .WING.SPAN

        let .STEP = .NO.OF.SPEEDS " to quit from the loop

      always

    loop

  always

 return

end  " det.tot
routine DET.VAPP given .VRR yielding .VELOCITY

" This routine calculates the atmospheric constants for the given airport
" conditions and then the velocity of approach for the aircraft in question.
" CALLED FROM: ASSIGN.ARR.ACFT.NAME
" CALLS :

 define .VRR as a integer variable

 define .VELOCITY,

       .TZERO,

       .LAMBDA,

       .THETAS,

       .DELTAS,

       .TR.TEMP.RATIO,

       .SIGMA,

       .RHO,

       .ACC.GRA,

       .WT.FACTOR,

       .ACFT.WT,

       .V.STALL as a real variables

 let .TZERO = 288.2

 let .LAMBDA = .0065

 " Estimate the standard atmosphere values at given elevation

   let .THETAS = 1 - (.LAMBDA/.TZERO) * AIR.ELEV

   let .DELTAS = .THETAS ** 4.2561

 " Estimate true temperature ratio
```

```
    let .TR.TEMP.RATIO = (273 + AIR.TEMP) / 288.2

  " Compute the equivalent density ratio(.SIGMA)

    let .SIGMA = .DELTAS / .TR.TEMP.RATIO

  let .RHO = 1.225   " Standard atmosphere sea level density(kg/cu.meter)

  let .ACC.GRA = 9.81 "Acceleration due to gravity(m/sec-sec)


 "Determination of the candidate aircraft weight while landing

    if  ARR.ACFT.WTF.STD(.VRR) eq 0

      ARR.ACFT.WTF.STD(.VRR) = .001

    always

    until .WT.FACTOR gt 0.0 and .WT.FACTOR le 1.0 , do

    let .WT.FACTOR = normal.f(ARR.ACFT.WTF.MEAN(.VRR), ARR.ACFT.WTF.STD(.VRR),
                              2)

    loop

    let .ACFT.WT = ARR.ACFT.OEW(.VRR) + (ARR.ACFT.MALW(.VRR) -
                   ARR.ACFT.OEW(.VRR)) * .WT.FACTOR

  "Calculate velocity of approach

    let .V.STALL = sqrt.f(2 * .ACFT.WT * .ACC.GRA /
                          (.RHO * .SIGMA * ARR.ACFT.CL.MAX(.VRR) *
                            ARR.ACFT.WING.AREA(.VRR)))

              "Stalling velocity

    let .VELOCITY = 1.3 * .V.STALL

    " New printing procedure for approach velocity

 "  open unit 23 for output, name = "VAPP.DAT", append

 "  use unit 23 for output

 "  print 1 line with .VELOCITY thus
 "  *******.***
 "  close unit 23

 "End of printing procedure

    return

  end "det.vapp
 routine DO.BEEP

 " This beeps when called
 "CALLED FROM: NEW.APP.STRT, NEW.INPUT.DIST
```

```
  define .BEEP as a integer variable


     for .BEEP = 1 to 10, do

       write 7 as A 1, + using unit 6

     loop

  return

end " do.beep
 process FINAL.APPROACH.OPERATION

 " In this process the aircraft enters the final approach as per its
 " T.DEP.STACK and reaches threshold after its APPROACH.OCC.TIME . The
 " no. of aircraft using the FINAL.APPROACH is also kept in track this helps
 " in knowing the aircraft that is preceding  the one in the stack by using
 " counter COUNT.APP.
 " CALLED FROM: SYS.CHECK.AT,STACK, PRECEDE.FASTER.ACFT, PRECEDE.SLOWER.ACFT
 " CALLS: LANDING.OPERATION


   request 1 FINAL.APPROACH(1)

   let COUNT.APP = COUNT.APP + 1

   let T.DEP.STACK(COUNT.APP) = time.v

   if (COUNT.APP + 1) le COUNT.ARR

     call SYS.CHECK.AT.STACK

   always

   let TOT.ARR.DELAY = TOT.ARR.DELAY + ( T.DEP.STACK(COUNT.APP) -
                          T.ENTER.STACK(COUNT.APP))

   work APP.OCC.TIME(ARR.ID(COUNT.APP)) seconds

   relinquish 1 FINAL.APPROACH(1)  "  after reaching threshold

   activate a LANDING.OPERATION now

   return

 end " final.approach.operation
process GENERATOR

  " The initiation of the creation of the aircraft according to user specified
  " distribution is executed in this process.
  " CALLED FROM: MAIN
  " CALLS: CREATE.AIRCRAFT.ARR, CREATE.AIRCRAFT.DEP

     define .R.ARR.DIST as a real variable

     define .II as an integer variable
```

```
    reserve ARR.ID,
            T.ENTER.STACK,
            T.DEP.STACK,
            T.LANDING,
            EXIT.ASSIGNED as TOT.ARR.ACFT
    reserve ROT.STAT,
            NO.IDENT.ARR as  NO.ARR.AIRCRAFT
    reserve EXIT.ASSIGN as NO.ARR.AIRCRAFT by NO.EXITS
    for .II = 1 to TOT.ARR.ACFT do
      if ARR.DIST eq 1
        .R.ARR.DIST = poisson.f(ARR.DIST.MEAN, 1)
      else
        if ARR.DIST eq 2
          .R.ARR.DIST = exponential.f(ARR.DIST.MEAN, 1)
        else
          if ARR.DIST eq 3
          .R.ARR.DIST = uniform.f(ARR.DIST.MEAN, ARR.DIST.STD, 1)
          always
        always
      always
      activate  a CREATE.AIRCRAFT.ARR now
      wait .R.ARR.DIST seconds
    loop "  .II = 1 to TOT.ARR.ACFT
  end "generator
process GENERATOR.DEP
 define .R.DEP.DIST as a  real variable
 define .II as an integer variable

  reserve DEP.ID,
          T.ENTER.RAMP,
          T.DEP.RAMP as TOT.DEP.ACFT
```

```
 for .ll = 1 to TOT.DEP.ACFT  do
"choosing the correct distribution for the interarrivals for departing aircraft
    if DEP.DIST eq 1
     .R.DEP.DIST = poisson.f(DEP.DIST.MEAN, 3)
    else
     if DEP.DIST eq 2
      .R.DEP.DIST = exponential.f(DEP.DIST.MEAN, 3)
     else
      if DEP.DIST eq 3
       .R.DEP.DIST = uniform.f(DEP.DIST.MEAN, ARR.DIST.STD, 3)
      always
     always
    always
    activate a  CREATE.AIRCRAFT.DEP now
    wait .R.DEP.DIST seconds
 loop
 return
end " generator.dep
routine GLOBAL.STAT.OP given .ITER
"This routine displays the global statistics of the simulation run.
"CALLED FROM: REPORT.GEN
"CALLS: REPORT.GEN

  define .WR,
       .ITER,
       .NO.ARR as integer variables
  define .WAROT,
       .IND.WT.ROT,
       .SUM.SQ.IND.WT.ROT,
       .STD.WAROT as  real variables
  define .NEW.FILE.NAME,
       .EXT as text variables
```

```
  " CALCULATION OF WEIGHTED AVERAGE ROT

  if TOT.ARR.ACFT gt 0

    for .WR = 1 to NO.ARR.AIRCRAFT do

     if NO.IDENT.ARR(.WR) eq 0

        .IND.WT.ROT = 0

        let .NO.ARR = .NO.ARR + 1

     else

        .IND.WT.ROT = (ROT.STAT(.WR) / NO.IDENT.ARR(.WR)) *
                                ARR.ACFT.PERCENT(.WR) / 100

     always


     .WAROT = .WAROT + .IND.WT.ROT

     .SUM.SQ.IND.WT.ROT = .SUM.SQ.IND.WT.ROT + (.IND.WT.ROT) ** 2

    loop

     .STD.WAROT = sqrt.f(((NO.ARR.AIRCRAFT - .NO.ARR) * (.SUM.SQ.IND.WT.ROT) -
                    .WAROT ** 2 ) /  ( NO.ARR.AIRCRAFT - .NO.ARR))

   always

   let .NEW.FILE.NAME = substr.f(APP.FILE.NAME,1,7)

   let .EXT = itot.f(.ITER)

   let .NEW.FILE.NAME = concat.f( .NEW.FILE.NAME,"G.", .EXT)

   open unit 12 for output, name = .NEW.FILE.NAME

    use unit 12 for output

     write COUNT.LAND, TOT.ARR.DELAY, COUNT.T.OFF, TOT.DEP.DELAY, .WAROT,
        .STD.WAROT, END.LAND.SIM, END.DEP.SIM, time.v as i 5, s 1, d(10,2),
          s 1, i 5, s 1, d(10,2), s 1, d(8,2), s 1, d(6,2), s 1, d(8,3),
          s 1, d(8,3), s 1, d(9,3)


   close unit 12

   call REPORT.GEN giving .ITER

   return

end  "global.stat.op
routine GLOBAL.STAT.VIEW given .ITER

"This routine displays the global statistics of the simulation run.
"CALLED FROM: REPORT.GEN
"CALLS: REPORT.GEN
```

```
define .CT.LAND,

       .CT.T.OFF,

       .ITER as integer variables

define .TOT.ARR.DEL,

       .TOT.DEP.DEL,

       .ARR.SIM,

       .DEP.SIM,

       .TOT.SIM,

       .WAROT,

       .STD.WAROT as  real variables

define .CHOICE,

       .NEW.FILE.NAME,

       .EXT as text variables

call vclears.r

let .NEW.FILE.NAME = substr.f(APP.FILE.NAME,1,7)

let .EXT = itot.f(.ITER)

let .NEW.FILE.NAME = concat.f( .NEW.FILE.NAME,"G.", .EXT)
" Reading the global statistics from the output file

open unit 12 for input, name = .NEW.FILE.NAME

 use unit 12 for input

 read .CT.LAND, .TOT.ARR.DEL, .CT.T.OFF, .TOT.DEP.DEL, .WAROT, .STD.WAROT,
      .ARR.SIM, .DEP.SIM, .TOT.SIM

close unit 12
" completed the input

let .CHOICE = "A"

until .CHOICE eq "O" do

  if .CHOICE eq "P"

    open unit 12  for output, name = "GLOBAL.RSM"

    use unit 12 for output

  always

  let page.v = 1
```

```
    begin report on a new page

     begin heading

      if page is first

        print 1 line thus
                        GLOBAL STATISTICS

        print 1 line with .ITER thus
                        ----------          ITER. NO. ****


      always

      skip 1 output line

     end "heading section

     print 1 line with  .CT.LAND  thus
Total Arrivals  Simulated : *****

     skip 1 line

     print 1 line with  .TOT.ARR.DEL, (.TOT.ARR.DEL /.CT.LAND)  thus
Total Delay for Arrivals : *****.** Min; Ave. Arrival Delay : ****.*** Min

     skip 1 line

     print 1 line with .ARR.SIM thus
Total Simulation Time for Arrivals : *****.*** Min

     skip 1 line

     print 1 line with .CT.T.OFF thus
Total Departures Simulated : *****

     skip 1 line

     print 1 line with .TOT.DEP.DEL, ( .TOT.DEP.DEL / .CT.T.OFF)  thus
Total Delay for Departures : *****.** Min; Ave. Departure Delay: *****.*** Min

     skip 1 line

     print 1 line with .DEP.SIM thus
Total Simulation Time for Departures: *****.*** Min

     skip 1 line

     print 1 line with .WAROT thus
Weighted Average Runway Occupancy Time : *****.** Secs

     skip 1 line

     print 1 line with .STD.WAROT thus
Std. Dev., of Weighted Average ROT : *****.**

     skip 1 line
```

```
      print 1 line with .TOT.SIM  thus
Total Simulation Time : ****.** Mins

    end "report section

    if .CHOICE eq "P"

      close unit 12

 " COMMANDS FOR USING DOS COMMANDS"

      let GT.1 = "CD HIGH2"

      SYS$DO.X(11)

      let GT.1 = "COPY GLOBAL.RSM LPT1"

      SYS$DO.X(11)

      let GT.1 = "CD.."

      SYS$DO.X(11)

    always

    call vfcolor.r(4)

    call vgotoxy.r(22,1)

    call vbcolor.r(5)

    call vfcolor.r(15)

    print 1 line thus
 ENTER - (P)RINT; (O)UTPUT MENU :

    call vgotoxy.r(22,33)

    read .CHOICE

    let .CHOICE = upper.f(.CHOICE)

    let .CHOICE = fixed.f(.CHOICE,1)

    call vbcolor.r(3)

    call vfcolor.r(0)

  loop " until .CHOICE = "O"

  call REPORT.GEN given .ITER

  return

end  "global.stat.view
routine INPUT.DATA.OP given .ITER
" This routine gives an option to choose view the data used for the simulation.
" CALLED FROM: REPORT.GEN
" CALLS: ARRIVAL.POP.OP, DEP.POP.OP, RWY.EX.OP, INT.SEP.OP, INT.ARR.OP,
```

```
"       REPORT.GEN

 define .ITER,

       .CHOICE as integer variables

 call vclears.r

 print 3 lines thus
                      ╔══════════════════════════════════════════╗
                      ║       APPLICATION DATA        ║
                      ╚══════════════════════════════════════════╝

 skip 2 lines

 print 6 lines thus
                        1) ARRIVAL POPULATION
                        2) DEPARTURE POPULATION
                        3) RUNWAY & EXIT DATA
                        4) INTRAIL SEPARATION
                        5) INTERARRIVAL DISTRIBUTIONS
                        6) RETURN TO OUTPUT MENU


 call vgotoxy.r(22,1)

 print 1 line thus
ENTER CHOICE NO.:

 call vgotoxy.r(22,18)

 read .CHOICE

 select case .CHOICE

   case 1

     call ARRIVAL.POP.OP giving .ITER

   case 2

     call DEP.POP.OP giving .ITER

   case 3

     call RWY.EX.OP giving .ITER

   case 4

     call INT.SEP.OP giving .ITER

   case 5

     call INT.ARR.OP giving .ITER

   case 6

     call REPORT.GEN giving .ITER
```

```
    default

      call REPORT.GEN giving .ITER

   endselect

   return

end " input.data.op
routine INT.ARR.OP given .ITER

" This routine displays the interarrival distributions used for the current
" application run.
" CALLED FROM: INPUT.DATA.OP
" CALLS

   define .ITER as an integer variable

   define .DIST,

          .CHOICE as text variables

    call vclears.r

    print 3 lines thus
                  ╔═══════════════════════════╗
                  ‖  INTERARRIVAL DISTRIBUTIONS  ‖
                  ╚═══════════════════════════╝


    skip 2 lines

    print 2 lines thus
                          ARRIVALS
                          ————————

       select case ARR.DIST

          case 1

            let .DIST = "POISSON"

          case 2

            let .DIST = "EXPONENTIAL"

           case 3

            let .DIST = "UNIFORM"

       endselect

       skip 1 line

       if ARR.DIST lt 3

          print 1 line with .DIST, ARR.DIST.MEAN thus
DISTRIBUTION : ************  MEAN: *****.** Sec
```

```
    else

     print 1 line with  .DIST, ARR.DIST.MEAN, ARR.DIST.STD thus
DISTRIBUTION : ************  LOWER: *****.** Sec  HIGHER  : *****.** Sec

    always "ARR.DIST lt 3

    skip 2 lines


  print 2 lines thus
                        DEPARTURES
                        ----------

    select case DEP.DIST

       case 1

        let .DIST = "POISSON"

       case 2

        let .DIST = "EXPONENTIAL"

       case 3

        let .DIST = "UNIFORM"

    endselect

    skip 1 line

    if DEP.DIST lt 3

      print 1 line with .DIST, DEP.DIST.MEAN thus
DISTRIBUTION : ************  MEAN: *****.** Sec

    else

     print 1 line with  .DIST, DEP.DIST.MEAN, DEP.DIST.STD thus
DISTRIBUTION : ************  LOWER: *****.** Sec  HIGHER  : *****.** Sec

    always

    call vgotoxy.r(22,1)

    call vbcolor.r(5)

    call vfcolor.r(15)

    print 1 line thus
HIT: (PRINT SCREEN) TO PRINT ; (Q)UIT:

    until .CHOICE eq "Q", do

      call vgotoxy.r(22,40)

      read .CHOICE
```

```
        call vbcolor.r(3)

        call vfcolor.r(0)

        let .CHOICE = fixed.f(.CHOICE,1)

        let .CHOICE = upper.f(.CHOICE)

      loop " .CHOICE eq Q

      call INPUT.DATA.OP giving .ITER

      return

end " int.arr.op
routine INT.SEP.OP given .ITER

" This routine displays the min. interarrival separation data used.
" CALLED FROM: INPUT.DATA.OP
" CALLS:

 define .ITER,

       .CT,

       .LOC,

       .IPS as integer variables

 define .CHOICE as a text variable

 while .CHOICE ne "Q", do

   call vclears.r

   let .CT = .CT + 1

   if .CT eq 1

     print 1 line thus
                  MIN. INTER-ARRIVAL SEPERATION (METERS)
   else

     if .CT eq 2

       print 1 line thus
                  MIN. DEPARTURE-ARRIVAL SEPARATION (METERS)

     else

       print 1 line thus
                  MIN. INTER-DEPARTURE SEPARATION (SECONDS)

     always

   always

   call SEP.SCREEN
```

```
  let .LOC = 4

  let .IPS = 0

  for .IPS = 1 to 3  do

    let .LOC = .LOC + 2

    call vgotoxy.r(.LOC,18)

    if .CT eq 1

      print 1 line with SEP.DIST.ARR(.IPS,1), SEP.DIST.ARR(.IPS,2),
            SEP.DIST.ARR(.IPS,3) thus
*****.**          *****.**          *****.**

    else

      if .CT eq 2

        print 1 line with DEP.ARR.SEP(.IPS,1), DEP.ARR.SEP(.IPS,2),
              DEP.ARR.SEP(.IPS, 3) thus
*****.**          *****.**          *****.**

      else

        print 1 line with SEP.T.TAKEOFFS(.IPS,1), SEP.T.TAKEOFFS(.IPS,2),
              SEP.T.TAKEOFFS(.IPS,3) thus
*****.**          *****.**          *****.**

        let .CT = 0

      always

    always

  loop "until .IPS = 3

  call vgotoxy.r(22,1)

  call vfcolor.r(15)

  call vbcolor.r(5)

  print 1 line thus
ENTER (N)EXT SCREEN; (Q)UIT :

  call vgotoxy.r(22,30)

  read .CHOICE

  call vbcolor.r(3)

  call vfcolor.r(0)

  let .CHOICE = fixed.f(.CHOICE, 1)

  let .CHOICE = upper.f(.CHOICE)
```

```
  if .CHOICE eq "Q"

    call INPUT.DATA.OP giving .ITER

  always

 loop

 return

end " int.sep.op
 process LANDING.OPERATION

 " In this process the arriving aircraft touchdowns and exits the runway after
 " selecting the proper exit.  Then the aircraft exits the system.
 " CALLED FROM: FINAL.APPROACH.OPERATION
 " CALLS:


  define .EXT,

       .NEW.FILE.NAME as text variables

  define .ACFT.ID as a pointer variable

  let .NEW.FILE.NAME = substr.f(APP.FILE.NAME,1,7)

  let .EXT = itot.f(RUN.NO)

  let .NEW.FILE.NAME = concat.f( .NEW.FILE.NAME,"R.", .EXT)

   request 1 RUNWAY(1)

   let COUNT.LAND = COUNT.LAND + 1

   let OCCUPANT.OPERATION = OPERATION(ARR.ID(COUNT.LAND))

   let T.LANDING(COUNT.LAND) = time.v

 if COUNT.LAND eq 1

   open unit 16 for output, name = .NEW.FILE.NAME

    use unit 16 for output

      let page.v = 1

      begin report on a new page

       begin heading


         print 2 lines  thus
                    ARRIVAL EVENT LIST
                    ------------


         skip 1 line

         print 1 line with APP.FILE.NAME, RUN.NO thus
```

```
   APPLICATION FILE : **********              RUN NO. : *****

       skip 1 line

       print 4 lines thus
AIRCRAFT     CREATION   ENTER FINAL  LANDING  ROT  EXIT    DELAY
 NAME         TIME      APPROACH     TIME        CHOSEN
             Min        Min        Min   Sec        Min
-------------------------------------------------------

       end " heading section

     always

     if COUNT.LAND gt 1

       open unit 16 for output, name = .NEW.FILE.NAME , append

        use unit 16 for output

      always

     write  NAME(ARR.ID(COUNT.LAND)),
           (T.ENTER.STACK(COUNT.LAND)),
            (T.DEP.STACK(COUNT.LAND)),
             (T.LANDING(COUNT.LAND)),
              RWY.OCC.TIME(ARR.ID(COUNT.LAND)),
               EXIT.ASSIGNED(COUNT.LAND),
                (T.DEP.STACK(COUNT.LAND) -
                   T.ENTER.STACK(COUNT.LAND)) as t 12, s 2, d(8,2), s 5,
                        d(8,2), s 5, d(8,2), s 2, d(6,2), s 2,
                         i 2, s 5, d(7,2), /


  if COUNT.LAND eq TOT.ARR.ACFT

    end " report section

  always " COUNT.LAND eq TOT.ARR.ACFT

  close unit 16

  let .ACFT.ID = COUNT.LAND

  work RWY.OCC.TIME(ARR.ID(COUNT.LAND)) seconds

  relinquish 1 RUNWAY(1)

  if .ACFT.ID eq TOT.ARR.ACFT

    END.LAND.SIM = time.v

  always

  destroy this AIRCRAFT called ARR.ID(.ACFT.ID)

  return

end " landing.operation
```

```
routine NEW.APP.STRT
" This routine is the main input routine and writes the data to an application
" file.
" CALLED FROM: APP.CHOICE
" CALLS:ACFT.LIST.SCRN, DO.BEEP, NEW.INPUT.DIST, YES.NO.MESSAGE,
"       ROW.COL.WARNING

 define .APP.NAME,

        .SURE,

        .YESNO,

        .YSN,

        .YNO,

        .YYNO,

        .DUP as text variables

 define .CUM.PERCENT.DEP,

        .DIST.ARR.MEAN,

        .DIST.ARR.STD,

        .DIST.DEP.MEAN,

        .DIST.DEP.STD,

        .ARR.BUFF,

        .DEP.BUFF,

        .DEP.ARR.BUFF,

        .RWY.ELEV,

        .RWY.TEMP,

        .RWY.LEN,

        .APP.LENGTH,

        .TRANS as real variables

 define .NO.RECORDS,

       .CUM.PERCENT,

       .IA,

       .ID,

       .CT,

       .CD,
```

.IW,

.IWT,

.CROW,

.DIST.ARR,

.DIST.DEP,

.NO.ARR.GEN,

.NO.DEP.GEN,

.RPS,

.SPS,

.TPS,

.IPS,

.JPS,

.KPS,

.MAS,

.NAS,

.MDS,

.NDS,

.MADS,

.NADS,

.ROWSEP,

.COLSEP,

.RADS,

.CADS,

.RDS,

.CDS,

.NO.OF.EXITS,

.RUN.WIDTH,

.CT.EXIT,

.XCD,

.IEXIT,

```
      .LOC,
      .XLOC,
      .LNO,
      .YCD,
      .YTYP,
      .EDIT.EX,
      .NEW.ACFT.NO,
      .TRANS.CHOICE,
      .NN,
      .JJ as  integer variables
define .ACFT.NO.ARR,
      .ACFT.PERCENT.ARR,
      .ACFT.NO.DEP,
      .ACFT.PERCENT.DEP,
      .EXIT.CHOICE as  1-dimensional integer arrays
define .WT.F.MEAN,
      .WT.F.STD,
      .EXIT.LOC as  1-dimensional real arrays
define .ARR.SEP,
      .ARR.DEP.SEP,
      .DEP.SEP as 2-dimensional real arrays
call vclears.r
skip 2 lines
call vfcolor.r(14)
print 1 line thus
                  NEW  APPLICATION
call vfcolor.r(4)
 print 1 line thus
                  ____________________________

call vfcolor.r(15)
skip 1 line
```

```
 print 1 line thus
Name of the application:

 call vgotoxy.r(5,26)

 read .APP.NAME

 let .APP.NAME = upper.f(.APP.NAME)

 skip 1 line

 print 1 line thus
File name of the application:

 call vgotoxy.r(7,32)

 read APP.FILE.NAME

 let APP.FILE.NAME = upper.f(APP.FILE.NAME)

 call vclears.r

 call ACFT.LIST.SCRN yielding .NO.RECORDS

   reserve .ACFT.NO.ARR,

           .ACFT.PERCENT.ARR,

           .WT.F.MEAN,

           .WT.F.STD,

           .ACFT.NO.DEP,

           .ACFT.PERCENT.DEP as .NO.RECORDS

 call vfcolor.r(14)

 print 1 line thus
                        DATA INPUT FOR ARRIVALS

 call vfcolor.r(15)

 call vgotoxy.r(15,0)

 print 1 line thus
Enter No. of Arrivals to be simulated:

 call vgotoxy.r(15,40)

 read .NO.ARR.GEN

 call vgotoxy.r(17,0)

 print 1 line thus
Select the aircraft by entering the number:

 call vgotoxy.r(19,0)
```

```
 print 1 line thus
Enter the percentage of the above aircraft to be simulated:

 call vgotoxy.r(21,0)

 print 1 line thus
Cumulative percentage of aircraft selected for simulation:

 until .CUM.PERCENT ge 100 do

  let .IA = .IA + 1

  call vgotoxy.r(17,44)  " for the cursor to be at 1st question

  read .ACFT.NO.ARR(.IA)

" CHECK FOR DUPLICATION OF AIRCRAFT SELECTION

  let  .DUP = "Y"

  until .DUP eq "N" do

    call ACFT.DUP.CHECK giving .IA, .ACFT.NO.ARR(*) yielding .NEW.ACFT.NO,
                                                          .DUP


 "CHECK FOR CORRECT INPUT WITHIN THE LIMITS

     until .NEW.ACFT.NO ge 1 and .NEW.ACFT.NO le .NO.RECORDS, do

       call DO.BEEP

       call vgotoxy.r(22,1)

       call vbcolor.r(12)

       print 1 line thus
WARNING: THE ENTERED AIRCRAFT NO. IS NOT AVAILABLE, ENTER VALUE AGAIN

       call vbcolor.r(1)

       call vgotoxy.r(17,44)

       call vclearl.r

       read .NEW.ACFT.NO

       call vgotoxy.r(22,0)

       call vclearl.r

     loop "until .ACFT.NO.ARR(.IA) is  > 1 or < 30

     let .ACFT.NO.ARR(.IA) = .NEW.ACFT.NO

  loop

    let .DUP = "Y"
```

```
  call vgotoxy.r(19,60) " for cursor to be at the second question
  read .ACFT.PERCENT.ARR(.IA)
  until (.ACFT.PERCENT.ARR(.IA) + .CUM.PERCENT) le 100, do
    call DO.BEEP
    call vgotoxy.r(22,1)
    call vbcolor.r(12)
    print 1 line thus
WARNING: THE PERCENTAGE CHOSEN EXCEEDS 100 ; ENTER A VALUE AGAIN
    call vbcolor.r(1)
    call vgotoxy.r(19,60)
    call vclearl.r
    read .ACFT.PERCENT.ARR(.IA)
    call vgotoxy.r(22,0)
    call vclearl.r
  loop
  let .CUM.PERCENT = .CUM.PERCENT + .ACFT.PERCENT.ARR(.IA)
  call vgotoxy.r(21,59)
  call vclearl.r
  print 1 line with .CUM.PERCENT thus
*** %
  call vgotoxy.r(17,44)
  call vclearl.r
  call vgotoxy.r(19,59)
  call vclearl.r

loop " until the percentage of aircraft chosen is 100

" input for weight factor data for landing aircraft
call vclears.r
call vfcolor.r(14)
print 1 line thus
                    INPUT WEIGHT FACTOR DATA
```

```
call vfcolor.r(4)
print 1 line thus
-------------------------------------------------------------------------------
call vfcolor.r(15)
print 1 line thus
|    LANDING AIRCRAFT        |        MEAN          |    STD. DEV.        |
call vfcolor.r(4)
print 1 line thus
-------------------------------------------------------------------------------
call vfcolor.r(7)
skip 1 line
 for .IW = 1 to .IA , write ACFT.NAME(.ACFT.NO.ARR(.IW)) as  s 9, t 12, /
 let .SURE = "Y"
 until .SURE = "N" do
   let .CROW = 4
   for .IWT = 1 to .IA  do
     let .CROW = .CROW + 1
     call vgotoxy.r(.CROW, 39)
     call vclearl.r
     read .WT.F.MEAN(.IWT)
     call vgotoxy.r(.CROW, 62)
     read .WT.F.STD(.IWT)
   loop " till selected aircraft were assigned their wt. factor
  call YES.NO.MESSAGE yielding .SURE
loop "until .SURE = N
 call vclears.r

" DATA INPUT FOR DEPARTURES
call ACFT.LIST.SCRN yielding .NO.RECORDS
call vfcolor.r(14)
print 1 line thus
                    DATA INPUT FOR DEPARTURES
```

```
 call vfcolor.r(15)

 skip 1 line

 call vgotoxy.r(15,0)

 print 1 line thus
Enter No. of Departures to be simulated:

 call vgotoxy.r(15,42)

 read .NO.DEP.GEN

 call vgotoxy.r(17,0)

 print 1 line thus
Select the aircraft by entering the number:

 call vgotoxy.r(19,0)

 print 1 line thus
Enter the percentage of the aircraft to be simulated:

 call vgotoxy.r(21,0)

 print 1 line thus
Cumulative percentage of aircraft selected for departures:

 until .CUM.PERCENT.DEP ge 100  do

   let  .ID = .ID + 1

   call vgotoxy.r(17,44)

   read .ACFT.NO.DEP(.ID)

"CHECK FOR DUPLICATE SELECTION OF AIRCRAFT

   let .DUP = "Y"

   until .DUP eq "N" do

      call ACFT.DUP.CHECK giving .ID, .ACFT.NO.DEP(*) yielding .NEW.ACFT.NO,
                                                          .DUP

" CHECK FOR LIMITS OF SELCTION OF AIRCRAFT

      until .NEW.ACFT.NO ge 1 and .NEW.ACFT.NO le .NO.RECORDS, do

        call DO.BEEP

        call vgotoxy.r(22,1)

        call vbcolor.r(12)

        print 1 line thus
WARNING: THE ENTERED AIRCRAFT NO. IS NOT AVAILABLE, ENTER THE VALUE AGAIN

        call vbcolor.r(1)
```

```
      call vgotoxy.r(17,44)

      call vclearl.r

      read .NEW.ACFT.NO

      call vgotoxy.r(22,0)

      call vclearl.r

    loop

      let .ACFT.NO.DEP(.ID) = .NEW.ACFT.NO

  loop " UNTIL BOTH CHECKS ARE SATISFIED

  let .DUP = "Y"

  call vgotoxy.r(19,56)

  read .ACFT.PERCENT.DEP(.ID)

  until (.ACFT.PERCENT.DEP(.ID) + .CUM.PERCENT.DEP) ge 1 and
        (.ACFT.PERCENT.DEP(.ID) + .CUM.PERCENT.DEP) le 100, do

    call DO.BEEP

    call vgotoxy.r(22,1)

    call vbcolor.r(12)

    print 1 line thus
WARNING: THE SELECTED PERCENTAGE EXCEEDS 100; ENTER A VALUE AGAIN

    call vbcolor.r(1)

    call vgotoxy.r(19,56)

    call vclearl.r

    read .ACFT.PERCENT.DEP(.ID)

    call vgotoxy.r(22,0)

    call vclearl.r

  loop


  let .CUM.PERCENT.DEP = .CUM.PERCENT.DEP + .ACFT.PERCENT.DEP(.ID)

  call vgotoxy.r(21,59)

  call vclearl.r

  print 1 line with .CUM.PERCENT.DEP  thus
*** %

  call vgotoxy.r(17,44)
```

```
  call vclearl.r
  call vgotoxy.r(19,56)
  call vclearl.r
loop " until percentage of dep. aircraft is 100.
call vclears.r
call NEW.INPUT.DIST yielding .DIST.ARR, .DIST.ARR.MEAN, .DIST.ARR.STD,
                             .DIST.DEP, .DIST.DEP.MEAN, .DIST.DEP.STD
" INPUT DATA REGARDING THE MINIMUM SEPERATIONS.
reserve .ARR.SEP,
        .ARR.DEP.SEP,
        .DEP.SEP as 3 by 3
 open unit 10 for input, name = "SEP.DAT"
 use unit 10 for input
  "JUMP OVER THE COMMENT LINES IN THE DATA FILE
   while mode is alpha do
     start new  input record
   loop " until mode is alpha

  for .RPS = 1 to 3, do
    start new input record
    read .ARR.SEP(.RPS,1)
    read .ARR.SEP(.RPS,2)
    read .ARR.SEP(.RPS,3)
  loop
  for .SPS = 1 to 3, do
    start new input record
    read .ARR.DEP.SEP(.SPS, 1)
    read .ARR.DEP.SEP(.SPS,2)
    read .ARR.DEP.SEP(.SPS,3)
  loop
  for .TPS = 1 to 3, do
```

```
    start new input record
    read .DEP.SEP(.TPS,1)
    read .DEP.SEP(.TPS,2)
    read .DEP.SEP(.TPS,3)
  loop
 close unit 10
 "END OF READING THE SEPERATION DATA
call vclears.r
call vfcolor.r(14)
print 1 line thus
                MIN. INTERARRIVAL SEPERATION (METERS)
call vfcolor.r(4)
 print 1 line thus
┌──────────────────────────────────────────────────────────────────────────────┐
call vfcolor.r(15)
print 1 line thus
| LEADING »»»»»»»»»»  HEAVY          LARGE          SMALL      |
call vfcolor.r(4)
print 1 line thus
├──────────────────────────────────────────────────────────────────────────────┤
call vfcolor.r(15)
print 7 line thus
| FOLLOWING        1          2          3       |
|                                          |
| HEAVY    1                                  |
|                                          |
| LARGE    2                                  |
|                                          |
| SMALL    3                                  |
 call vfcolor.r(4)
 print 1 line thus
└──────────────────────────────────────────────────────────────────────────────┘
 call vfcolor.r(15)
let .YESNO = "Y"
until .YESNO eq "N"  do
  let .LOC = 4
```

```
  for .IPS = 1 to 3  do

    let .LOC = .LOC + 2

    call vgotoxy.r(.LOC,18)

print 1 line with .ARR.SEP.(.IPS,1), .ARR.SEP(.IPS,2), .ARR.SEP(.IPS,3) thus
*****.**          *****.**          *****.**

  loop "until .IPS = 3

  call YES.NO.MESSAGE yielding .YESNO

  if .YESNO eq "Y"

  call vgotoxy.r(15,0)

    print 2 lines thus
 ENTER THE LOCATION OF THE VALUE TO BE CHANGED - ROW:
                                      COLUMN:

    call vgotoxy.r(15,53)

    read .ROWSEP

    until .ROWSEP ge 1 and .ROWSEP le 3, do

      let .LOC = 15

      call ROW.COL.WARNING giving .LOC  yielding .ROWSEP

    loop " until .ROWSEP is within 3

    call vgotoxy.r(16,53)

    read .COLSEP

    until .COLSEP ge 1 and  .COLSEP le 3, do

       let .LOC = 16

       call ROW.COL.WARNING  giving .LOC  yielding .COLSEP

    loop " until .COLSEP is within 3

    call vgotoxy.r(19,0)

    print 2 lines with .ARR.SEP(.ROWSEP, .COLSEP) thus
 CHANGE THE VALUE FROM : *****.**
                 TO :

    call vgotoxy.r(20,25)

    read .ARR.SEP(.ROWSEP, .COLSEP)

    for .LOC = 15 to 21  do

      call vgotoxy.r(.LOC,0)
```

```
    call vclearl.r

   loop '' until all lines are cleared below the table

  always

loop '' until .YESNO = "N"

call vclears.r

call vfcolor.r(14)

print 1 line thus
        MINIMUM SEPERATION BETWEEN A DEPARTURE AND ARRIVAL(METERS)

call vfcolor.r(4)

 print 1 line thus
┌──────────────────────────────────────────────────────────────────────────────┐

call vfcolor.r(15)

print 1 line thus
| LEADING(DEPARTURE) HEAVY          LARGE          SMALL      |

call vfcolor.r(4)

print 1 line thus
├──────────────────────────────────────────────────────────────────────────────┤

call vfcolor.r(15)

print 7 lines thus
| FOLLOWING(ARRIVAL)  1          2          3      |
|                                                  |
| HEAVY    1                                       |
|                                                  |
| LARGE    2                                       |
|                                                  |
| SMALL    3                                       |

 call vfcolor.r(4)

 print 1 line thus
└──────────────────────────────────────────────────────────────────────────────┘

 call vfcolor.r(15)

let .YSN = "Y"

until .YSN eq "N" do

  let .LOC = 4

  for .JPS = 1 to 3 do

    let .LOC = .LOC + 2

    call vgotoxy.r(.LOC,18)
```

```
print 1 line with .ARR.DEP.SEP(.JPS, 1), .ARR.DEP.SEP(.JPS,2),
            .ARR.DEP.SEP(.JPS,3) thus
*****.**            *****.**            *****.**

 loop " until .JPS = 3

 call YES.NO.MESSAGE yielding .YSN

 if .YSN eq "Y"

  call vgotoxy.r(15,0)

  print 2 lines thus
ENTER THE LOCATION OF THE VALUE TO BE CHANGED - ROW:
                                    COLUMN:

  call vgotoxy.r(15,53)

  read .RADS

  until .RADS ge 1 and .RADS le 3, do

   let .LOC = 15

   call ROW.COL.WARNING giving .LOC yielding .RADS

  loop " until .RADS is within 3

  call vgotoxy.r(16,53)

  read .CADS

  until .CADS ge 1 and .CADS le 3, do

   let .LOC = 16

   call ROW.COL.WARNING giving .LOC yielding .CADS

  loop "until .CADS IS within 3

  call vgotoxy.r(19,0)

  print 2 lines with .ARR.DEP.SEP(.RADS, .CADS) thus
CHANGE THE VALUE FROM : *****.**
                TO :

  call vgotoxy.r(20,25)

  read .ARR.DEP.SEP(.RADS, .CADS)

  for .LOC = 15 to 21 , do

   call vgotoxy.r(.LOC,0)

   call vclearl.r

  loop " to clear all the lines below the table

 always
```

```
loop '' until .YSN eq "N"

call vclears.r

call vfcolor.r(14)

print 1 line thus
                MIN. INTERDEPARTURE SEPERATION (IN SEC'S)


call vfcolor.r(4)

 print 1 line thus
┌──────────────────────────────────────────────────────────────────────────────┐

call vfcolor.r(15)

print 1 line thus
| LEADING »»»»»»»»»»   HEAVY          LARGE          SMALL      |

call vfcolor.r(4)

print 1 line thus
├──────────────────────────────────────────────────────────────────────────────┤

call vfcolor.r(15)

print 7 line thus
| FOLLOWING        1          2          3       |
|                                          |
| HEAVY     1                                  |
|                                          |
| LARGE     2                                  |
|                                          |
| SMALL     3                                  |

 call vfcolor.r(4)

 print 1 line thus
└──────────────────────────────────────────────────────────────────────────────┘

 call vfcolor.r(15)

 let .YNO = "Y"

 until .YNO eq "N" do

   let .LOC = 4

   for .KPS = 1 to 3 do

     let .LOC = .LOC + 2

     call vgotoxy.r(.LOC,18)

 print 1 line with .DEP.SEP(.KPS,1), .DEP.SEP(.KPS,2), .DEP.SEP(.KPS,3) thus
***.**           ***.**           ***.**

   loop "until .KPS = 3
```

```
    call YES.NO.MESSAGE yielding .YNO

   if .YNO eq "Y"

     call vgotoxy.r(15,0)

     print 2 lines thus
ENTER THE LOCATION OF THE VALUE TO CHANGED - ROW:
                                    COLUMN:

     call vgotoxy.r(15,53)

     read .RDS

     until .RDS ge 1 and .RDS le 3, do

       let .LOC = 15

       call ROW.COL.WARNING giving .LOC yielding .RDS

     loop


     call vgotoxy.r(16,53)

     read .CDS

     until .CDS ge 1 and .CDS le 3 , do

       let .LOC = 16

       call ROW.COL.WARNING giving .LOC yielding .CDS

     loop

     call vgotoxy.r(19,0)

     print 2 lines with .DEP.SEP(.RDS,.CDS) thus
CHANGE THE VALUE FROM : *****.**
                TO :

     call vgotoxy.r(20,25)

     read .DEP.SEP(.RDS,.CDS)

     for .LOC = 15 to 21 do

       call vgotoxy.r(.LOC,0)

       call vclearl.r

     loop " to clear all lines below the table

  always

loop " until .YNO = "N"

call SEP.BUFFER yielding .ARR.BUFF, .DEP.BUFF, .DEP.ARR.BUFF
```

```
" INPUT DATA FOR RUNWAY

call vclears.r

call vfcolor.r(14)

print 1 line thus
                        AIRPORT DATA

call  vfcolor.r(4)

print 1 line thus
                        ____________________

call vfcolor.r(15)

skip 1 line

print 1 line thus
1. AIRPORT ELEVATION (ABOVE MSL IN METERS) :

call vgotoxy.r(3,45)

read .RWY.ELEV

skip 1 line

print 1 line thus
2. MEAN TEMPERATURE AT AIRPORT (IN CELSIUS):

call vgotoxy.r(5,45)

read .RWY.TEMP

skip 1 line

print 1 line thus
3. RUNWAY LENGTH (METERS)              :

call vgotoxy.r(7,45)

read .RWY.LEN

skip 1 line

print 1 line thus
4. NO. OF EXITS  (INCLUDE RUNWAY END EXIT) :

call vgotoxy.r(9,45)

read .NO.OF.EXITS

skip 1 line

print 1 line thus
5. RUNWAY WIDTH (ENTER CHOICE NUMBER)     :

skip 1 line
```

```
print 1 line thus
 ( CHOICE : (1) 30 m  :  (2) 45 m  :  (3) 61 m )

call vgotoxy.r(11,45)

read .RUN.WIDTH

call vgotoxy.r(15,0)

print 1 line thus
6. LENGHT OF THE FINAL APPROACH PATH(METERS):

call vgotoxy.r(15,45)

read .APP.LENGTH

let .YYNO = "Y"

until .YYNO eq "N", do

  call YES.NO.MESSAGE yielding .YYNO

  if .YYNO eq "Y"

    call vgotoxy.r(20,1)

    print 1 line thus
              CHOOSE THE LINE NO. TO CHANGE :

    call vgotoxy.r(20,53)

    read .LNO

    until .LNO ge 1 and .LNO le 6, do

      let .LOC = 20

      call ROW.COL.WARNING  giving .LOC yielding .LNO

    loop

    call vgotoxy.r(20,0)

    call vclearl.r

    if .LNO  eq 1

      call vgotoxy.r(3,45)

      call vclearl.r

      read .RWY.ELEV

    always

    if .LNO eq 2

      call vgotoxy.r(5,45)
```

```
      call vclearl.r
      read .RWY.TEMP
    always
    if .LNO eq 3
      call vgotoxy.r(7,45)
      call vclearl.r
      read .RWY.LEN
    always
    if .LNO eq 4
      call vgotoxy.r(9,45)
      call vclearl.r
      read .NO.OF.EXITS
    always
    if .LNO eq 5
      call vgotoxy.r(11,45)
      read .RUN.WIDTH
    always
    if .LNO eq 6
      call vgotoxy.r(15,45)
      call vclearl.r
      read .APP.LENGTH
    always
  always
loop "until .YYNO eq N
 if .RUN.WIDTH eq 1
  .RUN.WIDTH = 30
 else
   if .RUN.WIDTH eq 2
     .RUN.WIDTH = 45
   else
```

```
    .RUN.WIDTH = 61

  always

always

call vclears.r

reserve .EXIT.LOC,

      .EXIT.CHOICE as .NO.OF.EXITS

call vfcolor.r(14)

print 1 line thus
                    EXITS AVAILABLE FOR SIMULATION

call vfcolor.r(4)

print 1 line thus
┌──────────────────────────────────────────────────┬──────────────────────────────────────────────────┐


call vfcolor.r(15)

print 1 line thus
|  NO.   TYPE        SPEED(m/s)     |  NO.   TYPE            SPEED(m/s)   |

call vfcolor.r(4)

print 1 line thus
├──────────────────────────────────────────────────┼──────────────────────────────────────────────────┤


call vfcolor.r(15)

print 5 lines thus
|  1)  90° (STANDARD)   8.00        |  6)   30° (REDIM-HEAVY)   35.00      |
|  2)  45° (STANDARD)   17.00       |  7)   20° (REDIM-HEAVY)   35.00      |
|  3)  30° (STANDARD)   26.00       |  8)   30° (REDIM-LARGE)   35.00      |
|  4)  30° (IMPROVED)   26.00       |  9)   20° (REDIM-LARGE)   35.00      |
|  5)  WIDE THROAT      17.00       |  10)  30° (REDIM-SMALL)   25.00      |

call vfcolor.r(4)

print 1 line thus
└──────────────────────────────────────────────────┴──────────────────────────────────────────────────┘

call vfcolor.r(15)

  skip 1 line

call vfcolor.r(14)

print 1 line thus
                      INPUT RUNWAY EXIT DATA

call vfcolor.r(4)
```

```
 print 1 line thus
┌──────────────────────────────────────────────────────────────────────┬──────────────────────────────────────────────────────────────────────┐


call vfcolor.r(15)

print 1 line thus
| EXIT NO.  LOCATION(MTS)  CHOICE NO.  |  EXIT NO.  LOCATION(MTS)  CHOICE NO. |

call vfcolor.r(4)

 print 1 line thus
└──────────────────────────────────────────────────────────────────────┴──────────────────────────────────────────────────────────────────────┘

call vfcolor.r(15)

let .XLOC = 15

let .YCD = 13

let .YTYP = 31

let .LNO = 6

for .CT.EXIT = 1 to .NO.OF.EXITS  do

  if .CT.EXIT eq 7

     .XLOC = 9

     let .XCD = 0

     let .LNO = 45

     let .YCD =  56

     let .YTYP = 74

   always

  let .XCD = .XLOC + .CT.EXIT

  call vgotoxy.r(.XCD,.LNO)

  print 1 line with .CT.EXIT thus
***)

  call vgotoxy.r(.XCD,.YCD)

  read .EXIT.LOC(.CT.EXIT)

   until .EXIT.LOC(.CT.EXIT) ge 0 and .EXIT.LOC(.CT.EXIT) le .RWY.LEN, do

     call DO.BEEP

     call vgotoxy.r(22,1)

     call vbcolor.r(12)
```

```
    print 1 line with .RWY.LEN thus
WARNING: THE EXIT LOCATION EXCEEDS RUNWAY LENGTH OF ******.**;ENTER AGAIN

    call vbcolor.r(1)

    call vgotoxy.r(.XCD,.YCD)

    call vclearl.r

    read .EXIT.LOC(.CT.EXIT)

    call vgotoxy.r(22,0)

    call vclearl.r

  loop "until exit is within the runway length

  call vgotoxy.r(.XCD,.YTYP)

  read .EXIT.CHOICE(.CT.EXIT)

  until .EXIT.CHOICE(.CT.EXIT) ge 1 and .EXIT.CHOICE(.CT.EXIT) le 10, do
   "the selected is within the choice number given in the exit table

    call DO.BEEP

    call vgotoxy.r(22,1)

    call vbcolor.r(12)

    print 1 line thus
WARNING: THE SELECTED EXIT CHOICE IS UNAVAILABLE: ENTER AGAIN

    call vbcolor.r(1)

    call vgotoxy.r(.XCD,.YTYP)

    call vclearl.r

    read .EXIT.CHOICE(.CT.EXIT)

    call vgotoxy.r(22,0)

    call vclearl.r

 loop " until the exit choice is within the available exit types

loop " until .CT.EXIT = .NO.OF.EXITS

let .YNO = "Y"

until .YNO eq "N" do

  call YES.NO.MESSAGE yielding .YNO

  if .YNO eq "Y"

    call vgotoxy.r(22,1)
```

```
    print 1 line thus
ENTER THE EXIT NUMBER OF THE EXIT TO BE EDITED :

    call vgotoxy.r(22,50)

    read .EDIT.EX

    call vgotoxy.r(22,0)

    call vclearl.r

    print 1 line with .EDIT.EX thus
EDIT THE DATA OF **) EXIT TO - LOCATION:          CHOICE:

    call vgotoxy.r(22,42)

    read .EXIT.LOC(.EDIT.EX)

    call vgotoxy.r(22,62)

    read .EXIT.CHOICE(.EDIT.EX)

    for .XLOC = 15 to 22 do

      call vgotoxy.r(.XLOC,0)

      call vclearl.r

    loop

    let .XLOC = 15

    let .YCD = 13

    let .YTYP = 31

    let .LNO = 6

    for .CT.EXIT = 1 to .NO.OF.EXITS  do

      if .CT.EXIT eq 7

        .XLOC = 9

        let .XCD = 0

        let .LNO = 45

        let .YCD =  56

        let .YTYP = 74

      always

      let .XCD = .XLOC + .CT.EXIT

      call vgotoxy.r(.XCD,.LNO)

      print 1 line with .CT.EXIT thus
```

```
***)

      call vgotoxy.r(.XCD,.YCD)

      print 1 line with .EXIT.LOC(.CT.EXIT) thus
*****.**

      call vgotoxy.r(.XCD,.YTYP)

      print 1 line with .EXIT.CHOICE(.CT.EXIT) thus
**

    loop '' until .CT.EXIT. = .NO.OF.EXITS

  always

 loop '' until .YNO = "N"

 call vclears.r

''SORTING THE EXIT.LOC ARRAY ACCORDING TO INCREASING ORDER OF EXIT LOCATION
'' AND CORRESPONDINGLY THE .EXIT.CHOICE ARRAY

 if .NO.OF.EXITS gt 1

    let .NN = (.NO.OF.EXITS - 1)

    for .CT.EXIT =  1 to .NN do

      let .JJ = .NO.OF.EXITS - .CT.EXIT

      for .LNO = 1 to .JJ do

        if .EXIT.LOC(.LNO) gt .EXIT.LOC(.LNO + 1)

          .TRANS = .EXIT.LOC(.LNO)

          let .EXIT.LOC(.LNO) = .EXIT.LOC(.LNO + 1)

          .EXIT.LOC(.LNO + 1) = .TRANS

          .TRANS.CHOICE  = .EXIT.CHOICE(.LNO)

          let .EXIT.CHOICE(.LNO) = .EXIT.CHOICE(.LNO + 1)

          let .EXIT.CHOICE(.LNO + 1) = .TRANS.CHOICE

        always

      loop

    loop

 always

'' WRITING THE INPUT DATA TO APPLICATION FILE

 skip 10 lines
```

```
print 1 line  with APP.FILE.NAME thus
WRITING THE INPUT DATA TO FILE: ********************

open unit 10 for output, name = APP.FILE.NAME

use unit 10 for output

write APP.FILE.NAME, .APP.NAME as t 15,  s 2, t 20, /

write .IA as i 3, /

for .CT = 1 to .IA, write .ACFT.NO.ARR(.CT), .ACFT.PERCENT.ARR(.CT),
 .WT.F.MEAN(.CT), .WT.F.STD(.CT) as  i 3, s 2, i 3, s 2, 2 d(10,4), /

skip 1 line

write .NO.ARR.GEN as i 5 , /, /

write .DIST.ARR, .DIST.ARR.MEAN, .DIST.ARR.STD as i 1, 2 d(10,4), /

skip 1 line

write .ID as "Departure Data", /, i 3, /

for .CD = 1 to .ID, write .ACFT.NO.DEP(.CD), .ACFT.PERCENT.DEP(.CD)
  as i 3, s 2, i 3, /

skip 1 line

write .NO.DEP.GEN as i 5, /, /

write .DIST.DEP, .DIST.DEP.MEAN, .DIST.DEP.STD as i 1, 2 d(10,4), /

"writing min.seperation data

write as "SEPERATION DATA", /

for .MAS = 1 to 3 do

  for .NAS = 1 to 3 do

    write .ARR.SEP(.MAS, .NAS) as s 1, d(8,2)

  loop " until .NAS = 3

 loop " until .MAS = 3

 skip 1 line

 for .MDS = 1 to 3 do

   for .NDS = 1 to 3 do

     write .DEP.SEP(.MDS,.NDS) as s 1, d(8,2)

   loop " until .NDS = 3

 loop " until .MDS = 3
```

```
 skip 1 line

 for .MADS = 1 to 3 do

   for .NADS = 1 to 3 do

     write .ARR.DEP.SEP(.MADS, .NADS) as s 1, d(8,2)

   loop "until .NADS = 3

 loop " until .MADS = 3

skip 2 lines

write as "BUFFER DATA", /

write .ARR.BUFF,.DEP.BUFF, .DEP.ARR.BUFF as s 2,
      d(6,2), s 2 , d(6,2), s 2, d(6,2)

skip 2 lines

write as "AIRPORT DATA" , /

write .RWY.ELEV, .RWY.TEMP, .RWY.LEN , .RUN.WIDTH, .APP.LENGTH as s 2, d(8,2),
          s 2, d(5,2), s 2, d(8,2), s 2, i 2, s 2, d(8,2), /, /

skip 1 line

write .NO.OF.EXITS as i 2, /

for .IEXIT = 1 to .NO.OF.EXITS do

    write .EXIT.LOC(.IEXIT), .EXIT.CHOICE(.IEXIT) as s 2, d(8,2) , s 2 , i 2, /

 loop " until .IEXIT = .NO.OF.EXITS

 close unit 10

"APPENDING APP.FILE TO INCLUDE THIS NEW APPLICATION INTO THAT LIST

 open unit 10 for output, name = "APP.FIL", append

   use unit 10 for output

   write .APP.NAME, APP.FILE.NAME as t 20, s 2, t 15

 close unit 10


 skip 2 lines

 print 1 line thus
                          DATA ENTRY COMPLETED

  return

end "new.app.strt
routine NEW.INPUT.DIST yielding .SR.DIST.ARR, .SR.DIST.ARR.MEAN,
                          .SR.DIST.ARR.STD, .SR.DIST.DEP,
```

```
                    .SR.DIST.DEP.MEAN, .SR.DIST.DEP.STD

" This routine requests user to choose the interarrival distributions for
" arrivals and departures.
" CALLED FROM: NEW.APP.STRT
" CALLS: SE.DIST.NO, SEL.MEAN.STD, YES.NO.MESSAGE, SEL.EDIT.DM


define .SR.DIST.ARR,

     .SR.DIST.DEP,

     .CLEAR as integer variables

define .SR.DIST.ARR.MEAN,

     .SR.DIST.ARR.STD,

     .SR.DIST.DEP.MEAN,

     .SR.DIST.DEP.STD as real variables

define .FLAG,

     .EDM  as text variables

"INPUT SCREEN FOR INTERARRIVAL DISTRIBUTIONS FOR BOTH ARRIVALS & DEP'S

call vclears.r

skip 3 lines

call vfcolor.r(14)

print 1 line thus
                INTERARRIVAL DISTRIBUTIONS

call vfcolor.r(4)

print 1 line thus
                ----------------

call vfcolor.r(15)

skip 1 lines

print 5 lines thus
                     1) POISSON

                     2) EXPONENTIAL

                     3) UNIFORM

skip 2 lines

call SE.DIST.NO yielding .SR.DIST.ARR

skip 1 line
```

```
call SEL.MEAN.STD giving .SR.DIST.ARR yielding .SR.DIST.ARR.MEAN,
                                                 .SR.DIST.ARR.STD
let .FLAG = "Y"

until .FLAG eq "N", do

  call YES.NO.MESSAGE yielding .FLAG

  if .FLAG eq "Y"

    call SEL.EDIT.DM  yielding  .EDM

    if .EDM eq "D"

      for .CLEAR = 15 to 20 , do " to clear all lines below the choice line

        call vgotoxy.r(.CLEAR,0)

        call vclearl.r

      loop

      call SE.DIST.NO  yielding  .SR.DIST.ARR

      skip 1 line

      call SEL.MEAN.STD giving .SR.DIST.ARR yielding .SR.DIST.ARR.MEAN,
                                                 .SR.DIST.ARR.STD
    else

      call SEL.MEAN.STD giving .SR.DIST.ARR yielding .SR.DIST.ARR.MEAN,
                                                 .SR.DIST.ARR.STD

    always

  always

 loop " until .FLAG for arrivals is N

call vclears.r

skip 3 lines

call vfcolor.r(14)

print 1 line thus
                  INTERDEPARTURE DISTRIBUTIONS

 call vfcolor.r(4)

 print 1 line thus
                  ---------------------

 call vfcolor.r(15)

skip 2 lines

print 5 lines thus
                      1) POISSON
```

```
                    2) EXPONENTIAL

                    3) UNIFORM

skip 2 lines

  call SE.DIST.NO yielding .SR.DIST.DEP

  skip 1 line

  call SEL.MEAN.STD giving .SR.DIST.DEP yielding .SR.DIST.DEP.MEAN,
                                                  .SR.DIST.DEP.STD
  let .FLAG = "Y"

  until .FLAG eq "N", do

    call YES.NO.MESSAGE yielding .FLAG

    if .FLAG eq "Y"

      call SEL.EDIT.DM  yielding  .EDM

      if .EDM eq "D"

        for .CLEAR = 15 to 20 , do " to clear lines below choice line

          call vgotoxy.r(.CLEAR, 0)

          call vclearl.r

        loop

        call SE.DIST.NO yielding .SR.DIST.DEP

        skip 1 line

        call SEL.MEAN.STD giving .SR.DIST.DEP yielding .SR.DIST.DEP.MEAN,
                                                  .SR.DIST.DEP.STD
      else

        call SEL.MEAN.STD giving .SR.DIST.DEP yielding .SR.DIST.DEP.MEAN,
                                                  .SR.DIST.DEP.STD

      always

    always

  loop " until .FLAG for dep is N

  call vclears.r

  return

 end " new.input.dist
routine NO.REPLICATIONS yielding .NO.REPS

" This routine requests the user for number of runs for a given application.
" CALLED FROM : MAIN
```

```
define .NO.REPS as an integer variable

call vclears.r

skip 3 lines

call vfcolor.r(14)

print 1 line thus
                    SIMULATION RUNS

call vfcolor.r(4)

print 1 line thus
                    _________

call vfcolor.r(7)

skip 5 lines

print 1 line thus
Enter No. of replications :

call vgotoxy.r(10,30)

read .NO.REPS

call vclears.r

return

end " no.replications
routine NORMAL.INVERSE.CDF given  .ARR.PROB.VIOL , .DEP.PROB.VIOL,
                  .DEP.ARR.PROB.VIOL yielding .ARR.PERCENTILE,
                    .DEP.PERCENTILE, .DEP.ARR.PERCENTILE

" calculates percentile values from the probability values.
" CALLED FROM: SEP.BUFFER
" CALLS:


   define .ARR.PROB.VIOL,

          .DEP.PROB.VIOL,

          .DEP.ARR.PROB.VIOL,

          .ARR.PERCENTILE,

          .DEP.PERCENTILE,

          .DEP.ARR.PERCENTILE,

          .PROB,

          .SPLIT,

          .A0,
```

```
        .A1,
        .A2,
        .A3,
        .B1,
        .B2,
        .B3,
        .B4,
        .C0,
        .C1,
        .C2,
        .C3,
        .D1,
        .D2,
        .R,
        .Q,
        .ZP as real variables
    define .I as an integer variable
    let .SPLIT = .42
    let .A0 = 2.50662823884
    let .A1 = -18.61500062529
    let .A2 = 41.39119773534
    let .A3 = -25.44106049637
    let .B1 = -8.4735109309
    let .B2 = 23.08336743743
    let .B3 = -21.06224101826
    let .B4 = 3.13082909833
    let .C0 = -2.78718931138
    let .C1 = -2.29796479134
    let .C2 = 4.85014127135
    let .C3 = 2.32121276858
```

```
  let .D1 = 3.54388924762

  let .D2 = 1.63706781897

for .I = 1 to 3 do

  if .I eq 1

    .PROB = ( 1 -.ARR.PROB.VIOL )

  else

    if .I = 2

      .PROB = ( 1 - .DEP.PROB.VIOL )

    else

      .PROB = ( 1 - .DEP.ARR.PROB.VIOL )

    always

  always

  if .PROB lt 0 or .PROB gt 1

    .ZP = 0

  else

    .Q = .PROB - 0.5

    if abs.f(.Q) le .SPLIT

      .R = .Q ** 2

      let .ZP = .Q * ((( .A3 * .R + .A2 ) * .R + .A1) + .A0 ) /
            ((((.B4 * .R + .B3) * .R + .B2 ) * .R + .B1 ) * .R + 1)

    else

      .R = .PROB

      if .Q gt 0

        .R = 1 - .PROB

      always

      let .R = sqrt.f( - log.e.f(.R))

      let .ZP = ((( .C3 * .R + .C2) * .R + .C1 ) * .R + .C0) /
            ((.D2 * .R + .D1) * .R + 1)

      if .Q lt 0

        .ZP =  - .ZP
```

```
        always
      always
    always
    if .I eq 1
      .ARR.PERCENTILE = .ZP
    else
      if .I = 2
        .DEP.PERCENTILE = .ZP
      else
        .DEP.ARR.PERCENTILE = .ZP
      always
    always
  loop " .I = 1 to 3
  return
end " normal.inverse.cdf
routine PRECEDE.FASTER.ACFT

" This routine checks the location of the preceding aircraft from the stack
" than it checked against the minimum seperation criteria and some buffer.
" Accordingly the lagging aircraft is released into the final approach.
" CALLED FROM: STATUS.APP.OCC
" CALLS: PRECEDE.ACFT.CLR.RWY, DET.SEP.DATA

  define .TT.APP.PRECEDE,
        .DIST.PRECEDE.STACK,
        .BUFFER,
        .MIN.SEP.BET.ARR,
        .T.MIN.SEP.STACK,
        .DEL.STACK,
        .T.LAG.TRHLD.RWY  as a real variable
  define .PREACFT,
        .FOACFT as integer variables

  let .PREACFT = TYPE(ARR.ID(COUNT.APP))
  let .FOACFT = TYPE(ARR.ID(COUNT.APP + 1))
```

```
let .BUFFER = INTARR.BUFF - SEP.DIST.ARR( .FOACFT, .PREACFT ) *
        ( 1 / V.APP(ARR.ID(COUNT.APP + 1)) - 1 / V.APP.(ARR.ID(COUNT.APP)))
        " IN SECONDS

if .BUFFER lt 0

    .BUFFER = 0 " buffer cannot be less than zero

always

let .MIN.SEP.BET.ARR = SEP.DIST.ARR( .FOACFT, .PREACFT ) + .BUFFER *
                    V.APP(ARR.ID(COUNT.APP + 1 ))
                  " mimimum separation.distance plus the buffer dist.

let .TT.APP.PRECEDE = trunc.f(time.v - T.DEP.STACK(COUNT.APP)) * 60 +
                  frac.f(time.v - T.DEP.STACK(COUNT.APP)) * 60
                      "Time elapsed since the preceding aircraft
                      " departed the stack

let .DIST.PRECEDE.STACK = V.APP(ARR.ID(COUNT.APP)) * .TT.APP.PRECEDE
                      "Distance from stack to the preceding
                      "aircraft in the final approach.

" check for minimum seperation

if .MIN.SEP.BET.ARR le .DIST.PRECEDE.STACK

  .T.MIN.SEP.STACK = 0     "Time to achieve minimum seperation from stack

else

  .T.MIN.SEP.STACK = (.MIN.SEP.BET.ARR - .DIST.PRECEDE.STACK)/
                  V.APP(ARR.ID(COUNT.APP)) "seconds
always

"check if leading aircraft clears runway before lagging aircraft reaches
"threshold.

call CHECK.PRECEDE.ACFT.CLR.RWY yielding .T.LAG.TRHLD.RWY

if .T.MIN.SEP.STACK ge .T.LAG.TRHLD.RWY

  .DEL.STACK = .T.MIN.SEP.STACK "seconds

else

   open unit 42 for output, name = "runocc.del" , append

   use unit 42 for output

   print 1 line with COUNT.APP , .T.MIN.SEP.STACK, .T.LAG.TRHLD.RWY thus
   ****    *****.****    *****.****

   close unit 42

  .DEL.STACK = .T.LAG.TRHLD.RWY "seconds

always
```

```
  let WAIT.TIME.STACK = .DEL.STACK

  activate a WAIT.AT.STACK now

  return

end "precede.faster.acft
routine PRECEDE.SLOWER.ACFT

" This routine checks if the lagging aircraft is released immediately
" into the final apporach the minimum seperation will be maintained
" when the preceding aircraft is at the threshold and if not appropriately
" the aircraft in the stack is delayed.
" CALLED FROM: STATUS.APP.OCC
" CALLS : CHECK.PRECEDE.ACFT.CLR.RWY, DET.SEP.DATA

  define .PFT,

       .FFT as integer variables

  define .MIN.SEP.BET.ARR,

       .T.REM.PRECEDE.THRLD,

       .DIST.LAG.ACFT,

       .DIST.LAG.THRLD,

       .T.LAG.THRLD.RWY,

       .T.MIN.SEP.THRLD,

       .D.STACK as real variables


  let .PFT = TYPE(ARR.ID(COUNT.APP))

  let .FFT = TYPE(ARR.ID(COUNT.APP + 1))


  let .MIN.SEP.BET.ARR = SEP.DIST.ARR (.FFT, .PFT) + INTARR.BUFF *
                         V.APP(ARR.ID(COUNT.APP))

  let .T.REM.PRECEDE.THRLD = APP.OCC.TIME(ARR.ID(COUNT.APP)) -
                             (trunc.f(time.v - T.DEP.STACK(COUNT.APP)) * 60 +
                              frac.f(time.v - T.DEP.STACK(COUNT.APP)) * 60)
              " Time to reach the threshold by the preceding aircraft
              " from its present position.

  let .DIST.LAG.ACFT = .T.REM.PRECEDE.THRLD * V.APP(ARR.ID(COUNT.APP + 1))
              "The distance the lagging aircraft will cover, if released
              "now, in the time the preceding aircraft reaches threshold.

  let .DIST.LAG.THRLD = LENGTH.APP(FINAL.APPROACH) - .DIST.LAG.ACFT
              " Distance between the lagging aircraft and threshold after
              " .T.REM.PRECEDE.THRLD

  if .MIN.SEP.BET.ARR le .DIST.LAG.THRLD
```

```
  .T.MIN.SEP.THRLD = 0   " Difference of time between the preceding aircraft
                         " reaching threshold and lagging aircraft reaching
                         " minimum seperation distance from threshold.

 else

  .T.MIN.SEP.THRLD = .T.REM.PRECEDE.THRLD - (LENGTH.APP (FINAL.APPROACH) -
                     .MIN.SEP.BET.ARR) / V.APP(ARR.ID(COUNT.APP + 1))

 always

 call CHECK.PRECEDE.ACFT.CLR.RWY  yielding  .T.LAG.THRLD.RWY

 if .T.MIN.SEP.THRLD ge .T.LAG.THRLD.RWY

  .D.STACK = .T.MIN.SEP.THRLD "seconds

 else

   open unit 42 for output, name = "RUNOCC.DEL", append

   use unit 42 for output

    print 1 line with COUNT.APP, .T.MIN.SEP.THRLD, .T.LAG.THRLD.RWY thus
    ****    *****.****    ******.****

    close unit 42

  .D.STACK = .T.LAG.THRLD.RWY "seconds

 always

 if .D.STACK gt 0.0001

   let WAIT.TIME.STACK =  .D.STACK

   activate a WAIT.AT.STACK now

 else

   activate a FINAL.APPROACH.OPERATION now

 always

 return

end "precede.slower.acft
routine PROG.STRT.SCRN
"Currently SIMGRAPHICS produces the title screen and animation. This is no longer in use.
 call vbcolor.r(1)

 call vfcolor.r(7)

 call vclears.r

 skip 3 lines

 call vfcolor.r(14)
```

```
print 3 lines thus
```


```
call  vfcolor.r(7)

skip 3 lines

" print 2 lines thus
 "              Developed by : Mr. Vijay B. Nunna
"                             Dr. Antonio A. Trani

skip 8 lines

call vfcolor.r(5)

print 3 lines thus
              UNIVERSITY CENTER FOR TRANSPORTATION RESEARCH
             VIRGINIA POLYTECHNIC INSTITUTE & STATE UNIVERSITY
                    BLACKSBURG, VIRGINIA


call vfcolor.r(15)

call vgotoxy.r(22,1)

print 1 line thus
Hit RETURN(ENTER) key to continue....

call vgotoxy.r(22,38)

"pause until a key pressed

read as / using unit 5

return

end
routine RE.INITIALIZE

" This routine reinitializes the variables that affect the simulation reruns.

   define .I as an integer variable

   let N.X.RUNWAY(1) = 0

   let N.Q.RUNWAY(1) = 0

   let N.X.FINAL.APPROACH(1) = 0

   let N.Q.FINAL.APPROACH(1) = 0

   let COUNT.ARR = 0

   let COUNT.APP = 0

   let COUNT.LAND = 0
```

```
    let COUNT.DEP = 0

    let COUNT.T.OFF = 0

    let time.v = 0.0

    for .I =  1 to TOT.DEP.ACFT do

      destroy the AIRCRAFT called DEP.ID(.I)

    loop

    release NO.IDENT.ARR(*),

          EXIT.ASSIGN(*,*),

          ROT.STAT(*)

    reset totals of ARR.DELAY and

                DEP.DELAY
    return

end  " re.initialize
routine READ.APP.DATA

  " This routine reads the data specified by the user from a selected
  " application file.
  " CALLED FROM: MAIN
  " CALLS:

  define .IA,

        .ID,

        .IAS,

        .JAS,

        .IADS,

        .JADS,

        .IDS,

        .JDS,

        .EXIT.CT as integer variables

  define .FILE.NAME as a text variable

  open unit 10 for input, name = APP.FILE.NAME

  use unit 10 for input

  let eof.v = 1

  read .FILE.NAME
```

```
start new input record

if APP.FILE.NAME ne .FILE.NAME

  call vclears.r

  call vgotoxy.r(15,0)

  print 2 lines with APP.FILE.NAME thus
  APPLICATION INPUT FILE NAME : **********
  DOES NOT EXIST IN THE DATA FILE. QUITTING FROM SIMULATION

else

 print 1 line thus
 READING THE APPLICATION FILE

 read NO.ARR.AIRCRAFT as i 3, / " No. of arrival aircraft chosen for
                              " simulation

 reserve ARR.ACFT.NO,

       ARR.ACFT.PERCENT,

       ARR.ACFT.WTF.MEAN,

       ARR.ACFT.WTF.STD as NO.ARR.AIRCRAFT

 for .IA = 1 TO NO.ARR.AIRCRAFT do

   read ARR.ACFT.NO(.IA), ARR.ACFT.PERCENT(.IA), ARR.ACFT.WTF.MEAN(.IA),
        ARR.ACFT.WTF.STD(.IA) as i 3, s 2, i 3, 2 d(10,4), /

 loop  "until .ia = no.arr.acft

 start new input record

 read  TOT.ARR.ACFT

 read ARR.DIST, ARR.DIST.MEAN, ARR.DIST.STD as /, /, i 1, 2 d(10,4), /, /, /

 read NO.DEP.AIRCRAFT

 start new input record

  reserve DEP.ACFT.NO,

       DEP.ACFT.PERCENT as NO.DEP.AIRCRAFT

 for .ID = 1 to NO.DEP.AIRCRAFT, read DEP.ACFT.NO(.ID), DEP.ACFT.PERCENT(.ID)
                              as i 3, s 2, i 3, /

 start new input record

 read  TOT.DEP.ACFT

 read DEP.DIST, DEP.DIST.MEAN, DEP.DIST.STD as /, /, i 1, 2 d(10,4), /

 reserve SEP.DIST.ARR,
```

```
      DEP.ARR.SEP,
      SEP.T.TAKEOFFS as 3 by 3
start new input record
for .IAS = 1 to 3 do
  for .JAS = 1 to 3 do
    read SEP.DIST.ARR(.IAS, .JAS) as s 1, d(8,2)
  loop '' until .JAS = 3
loop '' until .IAS = 3
start new input record
for .IDS = 1 to 3 do
  for .JDS = 1 to 3 do
    read SEP.T.TAKEOFFS(.IDS, .JDS) as s 1, d(8,2)
  loop '' until .JDS = 3
loop '' until .IDS = 3
start new input record
for .IADS = 1 to 3 do
  for .JADS = 1 to 3  do
    read DEP.ARR.SEP(.IADS, .JADS) as s 1, d(8,2)
  loop '' until .JADS = 3
loop '' until .IADS = 3
start new input record
read INTARR.BUFF, INTDEP.BUFF, INTDEP.ARR.BUFF as /,
      /, s 2, d(6,2), s 2, d(6,2), s 2, d(6,2)
start new input record
read AIR.ELEV, AIR.TEMP, RWY.LTH, RWY.WIDTH, LENGTH.APP(1) as /, /, s 2,
        d(8,2), s 2, d(5,2), s 2, d(8,2), s 2, i 2, s 2, d(8,2), /, /
skip 1 input record
''read exit details
read NO.EXITS as i 2, /
reserve EXIT.LOCATION,
      EXIT.CHOICE as NO.EXITS
```

```
  for .EXIT.CT = 1 to NO.EXITS , read EXIT.LOCATION(.EXIT.CT),
                                  EXIT.CHOICE(.EXIT.CT) as s 2, d(8,2),
                                    s 2, i 2, /


  always

  close unit 10

  print 1 line thus
COMPLETED READING THE APPLICATION FILE

  list eof.v

  return

end "read.app.data
routine READ.DATA.FILE

  " This routine reads the data from the data file as per the data required
  " by the application file.
  " CALLED FROM: MAIN
  " CALLS:

  define .L.NO as a integer variable

  if TOT.ARR.ACFT gt 0

    reserve ARR.ACFT.NAME,

          ARR.ACFT.MALW,

          ARR.ACFT.OEW,

          ARR.ACFT.CL.MAX,

          ARR.ACFT.DEC,

          ARR.ACFT.FREE.ROLL,

          ARR.ACFT.WING.AREA,

          ARR.ACFT.WING.SPAN,

          ARR.ACFT.TYPE as NO.ARR.AIRCRAFT

   always

  if TOT.DEP.ACFT gt 0

    reserve DEP.ACFT.TYPE,

          DEP.ACFT.NAME as NO.DEP.AIRCRAFT

  always

  print 1 line thus
  READING THE DATA BASE
```

```
open unit 10 for input, name = "ACFT.DATA"
 if TOT.ARR.ACFT gt 0
  for .L.NO = 1 to NO.ARR.AIRCRAFT do
    use unit 10 for input
    while mode is alpha do " to jump over the comments in the data file
      start new input record
    loop " mode alpha
    start new record
    skip ( ARR.ACFT.NO(.L.NO) + 1 ) records
   read ARR.ACFT.NAME(.L.NO), ARR.ACFT.TYPE(.L.NO), ARR.ACFT.MALW(.L.NO),
        ARR.ACFT.OEW(.L.NO), ARR.ACFT.CL.MAX(.L.NO),
         ARR.ACFT.DEC(.L.NO), ARR.ACFT.FREE.ROLL(.L.NO),
          ARR.ACFT.WING.AREA(.L.NO), ARR.ACFT.WING.SPAN(.L.NO)
    rewind 10
  loop " until all arriving aircraft data is read
 always
 if TOT.DEP.ACFT gt 0
  for .L.NO = 1 to NO.DEP.AIRCRAFT  do
    use unit 10 for input
    while mode is alpha do
      start new input record
    loop " while mode is alpha
    start new input record
    skip (DEP.ACFT.NO(.L.NO) + 1) records
    read DEP.ACFT.NAME(.L.NO), DEP.ACFT.TYPE(.L.NO)
    rewind 10
  loop " till all departure aircraft data is read
 always
 close unit 10
 print 1 line thus
COMPLETED THE READING OF THE DATA BASE
 return
```

```
  end "read.data.file
routine REPORT.GEN given .ITER

"This routine displays the output selection menu
"CALLED FROM: MAIN
"CALLS: ASSIGN.TABLE.OP, ARRIVALS.OP, DEPARTURES.OP, EVENT.LIST.OP,
"       RWY.OCC.STAT.OP, GLOBAL.STAT.OP, INPUT.DATA.OP

  define .NUM,

        .ITER as integer variables

  call vbcolor.r(3)

  call vfcolor.r(0)

  call vclears.r

  print 3 lines thus
                  ╔═══════════════════════════════════════════════════════════╗
                  ‖         OUTPUT  MENU            ‖
                  ╚═══════════════════════════════════════════════════════════╝

  skip 2 lines


  print 6 lines thus
                        1. ASSIGNMENT TABLE
                        2. ARRIVAL EVENT LIST
                        3. DEPARTURE EVENT LIST
                        4. GLOBAL STATISTICS
                        5. INPUT DATA
                        6. QUIT (NEXT RUN)

   call vgotoxy.r(22,1)

   call vbcolor.r(5)

   print 1 line thus
 SELECT THE CHOICE NUMBER :

   call vgotoxy.r(22,29)

   read .NUM

   until .NUM ge 1 and .NUM le 6 , do

     call vgotoxy.r(22,29)

     call vclearl.r

     read .NUM

   loop

   call vbcolor.r(3)

   select case .NUM
```

```
    case 1
      call ASSIGN.TABLE.OP giving .ITER
    case 2
      call ARRIVALS.EVENT.OP giving .ITER
    case 3
      call DEPARTURES.EVENT.OP giving .ITER
    case 4
      call GLOBAL.STAT.VIEW giving .ITER
    case 5
      call INPUT.DATA.OP giving .ITER
   case 6
     return
   default
    return
 endselect ".NUM
 return
end " report.gen
routine ROW.COL.WARNING given .LOC yielding .RADS

" This routine prints warning messages until correct value is entered, at
" the row or column locations during editing the seperation data
" CALLED FROM: NEW.APP.STRT
" CALLS:

  define .RADS,
        .LOC as an integer variable
    call DO.BEEP
    call vbcolor.r(12)
    call vgotoxy.r(22,1)
    print 1 line thus
WARNING: THE VALUE ENTERED EXCEEDS THE FIELD; ENTER A VALUE AGAIN
    call vbcolor.r(1)
    call vgotoxy.r(.LOC,53)
    call vclearl.r
```

```
    read .RADS

    call vgotoxy.r(22,0)

    call vclearl.r

  return

end " row.col.warning
routine RWY.EX.OP given .ITER

"This routine displays the runway and exit data.
" CALLED FROM : INPUT.DATA.OP
" CALLS:

 define .ITER,

       .CT,

       .NO,

       .YLOC as integer variables

 define .CHOICE,

       .TYPE as text variable

 let .CHOICE = "A"

 until .CHOICE eq "Q", do

   call vclears.r

   print 3 lines thus
```


```
   print 1 line thus
┌──────────────────────────────────────────────────────────────────────────────┐


   print 1 line with APP.FILE.NAME thus
|  APP. FILE NAME : **********                                                 |

   print 1 line thus
├──────────────────────────────────────────────────────────────────────────────┤

   print 3 lines with RWY.LTH, AIR.ELEV, RWY.WIDTH, AIR.TEMP thus
|  RUNWAY LENGTH : ******.** m       ALTITUDE  : ******.** m                   |
|                                                 |
|  RUNWAY WIDTH  : ******.** m       MEAN TEMP.:  ****.**° C                   |

   print 1 line thus
└──────────────────────────────────────────────────────────────────────────────┘

   print 1 line thus
```

```
                    EXIT DATA
  print 1 line thus
┌──────────────────────────────────────────────────────────────────────────────┐
  print 1 line thus
| NO.  LOCATION     TYPE              NO.  LOCATION    TYPE     |
  print 1 line thus
└──────────────────────────────────────────────────────────────────────────────┘
  let .YLOC = 2
  for .CT = 1 to NO.EXITS do
    select case EXIT.CHOICE(.CT)
      case 1
       let .TYPE = "90°(STANDARD)"
      case 2
        let .TYPE = "45°(STANDARD)"
      case 3
        let .TYPE = "30°(STANDARD)"
      case 4
        let .TYPE = "30°(INPROVED)"
      case 5
        let .TYPE = "WIDE THROAT"
      case 6
        let .TYPE = "30°(REDIM-HEAVY)"
      case 7
        let .TYPE = "20°(REDIM-HEAVY)"
      case 8
        let .TYPE = "30°(REDIM-LARGE)"
      case 9
        let .TYPE = "20°(REDIM-LARGE)"
      case 10
        let .TYPE = "30°(REDIM-SMALL)"
      default
```

```
            let .TYPE = ' '
        endselect
        let .NO = .NO + 1
        call vgotoxy.r((.NO + 13), .YLOC)
        print 1 line with .CT, EXIT.LOCATION(.CT), .TYPE thus
**) ********.**     ****************
        if .NO eq 7
          .NO = 0
          let .YLOC = 40
        always
    loop " .CT = 1 to NO.EXITS
    call vgotoxy.r(22,1)
    call vfcolor.r(15)
    call vbcolor.r(5)
    print 1 line thus
HIT: (PRINT SCREEN) FOR A PRINT; (Q)UIT :
    call vgotoxy.r(22,42)
    read .CHOICE
    call vbcolor.r(3)
    call vfcolor.r(0)
    let .CHOICE = fixed.f(.CHOICE, 1)
    let .CHOICE = upper.f(.CHOICE)
  loop " .CHOICE eq "Q"
  if .CHOICE eq "Q"
      call INPUT.DATA.OP giving .ITER
  always
  return
end " rwy.ex.op
routine SE.DIST.NO yielding .SR.DIST.AD
" This requests the user to choose the distribution fromthe available
" selection and returns the choice.
" CALLED FROM: NEW.INPUT.DIST
" CALLS : DO.BEEP
```

```
   define .SR.DIST.AD as an integer variable

   call vgotoxy.r(14,0)

   print 1 line thus
Enter your choice:

   call vgotoxy.r(14,20)

   call vclearl.r

   read .SR.DIST.AD

   until .SR.DIST.AD ge 1 and .SR.DIST.AD le 3 , do

      call DO.BEEP

      call vgotoxy.r(22,1)

      call vbcolor.r(12)

      print 1 line thus
 WARNING: THE SELECTED CHOICE IS UNAVAILABLE; ENTER YOUR CHOICE AGAIN

      call vbcolor.r(1)

      call vgotoxy.r(14,20)

      call vclearl.r

      read .SR.DIST.AD

      call vgotoxy.r(22,0)

      call vclearl.r

   loop

   return

 end " sel.dist.no
 routine SEL.EDIT.DM yielding .SR.EDM

 "This routine requests user to choose the editing option of either the
 "distribution or the mean/std. dev

        define  .SR.EDM as a text variable

        call vgotoxy.r(22,1)

        call vbcolor.r(5)

        print 1 line thus
DO YOU WANT TO EDIT (D)ISTRIBUTION OR (M)EAN/STD. DEV :

        call vgotoxy.r(22,60)

        read .SR.EDM
```

```
      let .SR.EDM = upper.f(.SR.EDM)

      let .SR.EDM = fixed.f(.SR.EDM,1)

      call vbcolor.r(1)

      call vgotoxy.r(22,0)

      call vclearl.r

   return

 end " sel.edit.dm
routine SEL.MEAN.STD  given .SR.DIST.AD yielding .SR.MEAN, .SR.STD

" This routine requests for mean or lower and upper values of the distribution
" selected.
" CALLED FROM : NEW.INPUT.DIST
" CALLS:

    define .SR.DIST.AD as an integer variable

    define .SR.MEAN,

          .SR.STD as real variables

  if .SR.DIST.AD le 2

     call vgotoxy.r(16,0)

     print 1 line thus
Enter the mean (Sec):

     call vgotoxy.r(16,22)

     call vclearl.r

     read .SR.MEAN

  else

     call vgotoxy.r(16,0)

     print 3 lines thus
Enter the lower value (Sec):

Enter the upper value (Sec):

     call vgotoxy.r(16,29)

     call vclearl.r

     read .SR.MEAN

     call vgotoxy.r(18,29)

     call vclearl.r

     read .SR.STD
```

```
  always

  return

end " sel.mean.std
routine SEP.BUFFER yielding .ARR.BUFF, .DEP.BUFF, .DEP.ARR.BUFF

" Data for buffer time is input here and calculated to be used for
" separations between aircraft.
" CALLED FROM: NEW.APP.STRT
" CALLS: YES.NO.MESSAGE, NORMAL.INVERSE.CDF

 define .ARR.BUFF,

       .ARR.PROB.VIOL,

       .ARR.BUFF.STD,

       .DEP.BUFF,

       .DEP.PROB.VIOL,

       .DEP.BUFF.STD,

       .DEP.ARR.BUFF,

       .DEP.ARR.PROB.VIOL,

       .DEP.ARR.BUFF.STD,

       .ARR.PERCENTILE,

       .DEP.PERCENTILE,

       .DEP.ARR.PERCENTILE as real variables

 define .LINE as an integer variable

 define .SURE as a text variable

 call vclears.r

 call vfcolor.r(14)

 print 1 line thus
                        BUFFER TIME DATA

 call vfcolor.r(4)

 print 1 line thus
┌──────────────────────────────┬──────────────────────────────┬──────────────────────────────┐

 call vfcolor.r(15)

 print 1 line thus
|                              | PROBABILITY OF VIOLATION |   STD. DEV.(SEC'S)   |

 call vfcolor.r(4)
```

```
print 1 line thus
```

```
call vfcolor.r(15)

skip 1 line

print 1 line thus
1) INTER-ARRIVALS

skip 1 line

print 1 line thus
2) INTER-DEPARTURES

skip 1 line

print 1 line thus
3) DEPARTURE-ARRIVAL

skip 1 line

call vfcolor.r(4)

print 1 line thus
```

```
call vfcolor.r(15)

 let .SURE = "Y"
 until .SURE = "N" do
    if .LINE eq 1 or .LINE eq 0
       call vgotoxy.r(5, 39)
       call vclearl.r
       read .ARR.PROB.VIOL
       call vgotoxy.r(5, 62)
       read .ARR.BUFF.STD
    always
    if .LINE eq 2 or .LINE eq 0
       call vgotoxy.r(7, 39)
       call vclearl.r
       read .DEP.PROB.VIOL
       call vgotoxy.r(7, 62)
       read .DEP.BUFF.STD
```

```
      always
      if .LINE eq 3 or .LINE eq 0
         call vgotoxy.r(9, 39)
         call vclearl.r
         read .DEP.ARR.PROB.VIOL
         call vgotoxy.r(9, 62)
         read .DEP.ARR.BUFF.STD
      always
      call YES.NO.MESSAGE yielding .SURE
      if .SURE eq "Y"
        call vgotoxy.r(22,1)
        call vclearl.r
        call vbcolor.r(5)
        print 1 line thus
ENTER THE LINE NO. TO BE EDITED:
        call vgotoxy.r(22,32)
        read .LINE
        call vgotoxy.r(22,0)
        call vclearl.r
        call vbcolor.r(1)
      always
  loop "until .SURE = N
  call NORMAL.INVERSE.CDF giving .ARR.PROB.VIOL, .DEP.PROB.VIOL,
                        .DEP.ARR.PROB.VIOL yielding .ARR.PERCENTILE ,
                          .DEP.PERCENTILE, .DEP.ARR.PERCENTILE
  if .ARR.PERCENTILE lt 0
    .ARR.PERCENTILE = 0
  always
  if .DEP.PERCENTILE lt 0
    .DEP.PERCENTILE = 0
  always
```

```
   let .ARR.BUFF = .ARR.BUFF.STD * .ARR.PERCENTILE

   let .DEP.BUFF = .DEP.BUFF.STD * .DEP.PERCENTILE

   let .DEP.ARR.BUFF = .DEP.ARR.BUFF.STD * .DEP.ARR.PERCENTILE

   call vclears.r

   return

end " sep.buffer
  routine SEP.SCREEN
  " This routine display a template for the speration data to be incorporated
  " within
  " CALLED FROM: INT.SEP.OP
  " CALLS:

    print 1 line thus
┌──────────────────────────────────────────────────────────────────────────────┐


print 1 line thus
| LEADING »»»»»»»»»»   HEAVY          LARGE          SMALL      |

print 1 line thus
├──────────────────────────────────────────────────────────────────────────────┤


print 7 lines thus
| FOLLOWING                                                  |
|                                                        |
| HEAVY                                                    |
|                                                        |
| LARGE                                                    |
|                                                        |
| SMALL                                                    |

 print 1 line thus
└──────────────────────────────────────────────────────────────────────────────┘

  return

end " sep.screen
Routine SET.GRAPHICS

    Let timescale.v = 100

"  Set up a display view port

    Let vxform.v = ..DISPLAY.VIEW.PORT
    Call setworld.r (-15, 15, -15, 15)
"     Call setview.r  (6000, 26000, 1000, 21000)

End "SET.GRAPHICS
 routine STATUS.APP.CLR.RWY.OCC

  " This routine checks if an aircraft is landing or takin off. Depending on
```

```
" the operation the departure is scheduled.if departure is delayed the
" routine SYS.CHECK.AT.THRLD is called  to know the system status after
" the delay.
" CALLED FROM: SYS.CHECK.AT.THRLD
" CALLS:DET.DEL.DEP, SYS.CHECK.AT.THRLD

  define .T.REM.CLR.RWY,

        .T.MIN.SEP.TAKEOFFS as  real variables

  if OCCUPANT.OPERATION eq ..LANDING

    .T.REM.CLR.RWY = RWY.OCC.TIME(ARR.ID(COUNT.LAND)) -
                  (frac.f(time.v -T.LANDING(COUNT.LAND)) * 60 +
                    trunc.f(time.v - T.LANDING(COUNT.LAND)) * 60)
              "Remaining time to clear the runway by the landing aircraft


      if .T.REM.CLR.RWY le 0.00001

        WAIT.TIME.RAMP = 0.005

      else

        WAIT.TIME.RAMP = .T.REM.CLR.RWY

      always

      activate a WAIT.AT.RAMP now

  else

      call DET.DEL.DEP yielding .T.MIN.SEP.TAKEOFFS

      if .T.MIN.SEP.TAKEOFFS gt 0

        WAIT.TIME.RAMP = .T.MIN.SEP.TAKEOFFS

        activate a WAIT.AT.RAMP now

      else

        activate a DEPARTURE.OPERATION now

      always

  always

  return

 end " status.app.clr.rwy.occ
routine STATUS.APP.OCC

" This routine checks the preceding aircraft whether it is faster or slower
" than the one in stack for interarrival seperation criteria.
"CALLED FROM: SYS.CHECK.AT.STACK
"CALLS; PRECEDE.ACFT.FASTER, PRECEDE.ACFT.SLOWER

  define .PRECEDE.SPEED,
```

```
        .OWN.SPEED as real variables

  let  .PRECEDE.SPEED = V.APP(ARR.ID(COUNT.APP))

  let  .OWN.SPEED = V.APP(ARR.ID(COUNT.APP + 1))

  if .PRECEDE.SPEED  ge .OWN.SPEED

    call PRECEDE.FASTER.ACFT       " preceding aircraft is faster

  else

    call PRECEDE.SLOWER.ACFT       " preceding aircraft is slower

  always

  return

end "status.app.occ
 routine STATUS.APP.OCC.RWY.CLR

 " The routine determines if the minimum seperation distance exists between
 " the arriving and departing aircraft.If satisfied the takeoff is set off
 " if not the departure is delayed and again system check is done by calling
 " SYS.CHECK.AT.THRLD.
 " CALLED FROM:
 " CALLS: DET.DEL.ARR, DEPARTURE.OPERATION, SYS.CHECK.AT.THRLD, WAIT.AT.RAMP

     define  .DEL.ARR as a real variable

     call DET.DEL.ARR yielding .DEL.ARR

     if .DEL.ARR gt 0  " minimum seperation does'nt exist

       let WAIT.TIME.RAMP =  .DEL.ARR

       activate a WAIT.AT.RAMP  now

     else

       activate a DEPARTURE.OPERATION now   " minimum seperation exists

     always

     return

   end "app.occ.rwy.clr
routine STATUS.APP.OCC.RWY.OCC

" This routine first checks the type of operation on runway and determines
" delay 1,if any, for a departure. Then it checks if min. seperation is
" present or not between arriving aircraft and accordingly delay 2 due to
" arrival is found. If both the the delays is zero then takeoff is commenced
" if not the departure is delayed by the greater of both delays and
" again system condition is checked.
" CALLED FROM: SYS.CHECK.AT.THRLD
" CALLS: DET.DEL.DEP, DET.DEL.ARR

  define .DEL.DEP1,
```

```
    .DEL.DEP2,

    .T.REM.CLR.RWY,

    .T.MIN.SEP.TAKEOFFS as  real variables

if OCCUPANT.OPERATION eq ..LANDING

  .T.REM.CLR.RWY = RWY.OCC.TIME(ARR.ID(COUNT.LAND)) -
                   (trunc.f(time.v - T.LANDING(COUNT.LAND)) * 60 +
                    frac.f(time.v - T.LANDING(COUNT.LAND)) * 60 )
                        " Remaining time for the landing aircraft
                        " to clear runway. .

  let .DEL.DEP1 = .T.REM.CLR.RWY

else

   call DET.DEL.DEP yielding .T.MIN.SEP.TAKEOFFS

   let  .DEL.DEP1 = .T.MIN.SEP.TAKEOFFS

always

call DET.DEL.ARR yielding .DEL.DEP2

if .DEL.DEP1 le 0 and .DEL.DEP2 le 0

   activate a DEPARTURE.OPERATION now

else

   if .DEL.DEP1 gt .DEL.DEP2

      if .DEL.DEP1 le 0.0001

         WAIT.TIME.RAMP = 0.05

      else

         WAIT.TIME.RAMP =  .DEL.DEP1

       always

      activate a WAIT.AT.RAMP now

    else

      if .DEL.DEP2 le 0.0001

         WAIT.TIME.RAMP = 0.05

      else

         WAIT.TIME.RAMP = .DEL.DEP2

      always

      activate a WAIT.AT.RAMP now
```

```
        always

      always

    return

  end " status.app.occ.rwy.occ
routine STATUS.RWY.OCC

" This routine checks if there is any delay for an arriving aircraft
" due to the presence of a landing aircraft on the runway, and if so it is
" delayed for the same.
" CALLED FROM: SYS.CHECK.AT.STACK
" CALLS:

    define .T.REM.CLR.RWY  as a real variable

    let .T.REM.CLR.RWY = RWY.OCC.TIME(ARR.ID(COUNT.LAND)) -
                         (frac.f(time.v - T.LANDING(COUNT.LAND)) * 60 +
                          trunc.f(time.v - T.LANDING(COUNT.LAND)) * 60)
      " time remaining for the landing aircraft to clear runway

    if .T.REM.CLR.RWY  gt APP.OCC.TIME(ARR.ID(COUNT.APP + 1))

      open unit 40 for output, name = "check.del", append

      use unit 40 for output

      print 1 line with (COUNT.APP + 1), COUNT.LAND thus
      *****    *****

      close unit 40

      WAIT.TIME.STACK = .T.REM.CLR.RWY - APP.OCC.TIME(ARR.ID(COUNT.APP + 1))

       if WAIT.TIME.STACK le 0.00001

         WAIT.TIME.RAMP = 0.005

       always

       activate a WAIT.AT.STACK now

    else

       activate a FINAL.APPROACH.OPERATION now

    always

    return

 end " status.rwy.occ
routine SYS.CHECK.AT.STACK

" This routine checks the system conditions for the aircraft before it enters
" the final approach to satisfy the minimum interarrival seperation criteria.
" Each system condition is done by a seperate subroutine.
" CALLED FROM: CREATE,AIRCRAFT.ARR
" CALLS: STATUS.APP.OCC, STATUS.APP.CLR.RWY.OCC
```

```
 define .APPROACH.STATUS,

       .RWY.STATUS as integer variables

 let .APPROACH.STATUS = N.X.FINAL.APPROACH(1) " N.X.resource is a system
                                  " attribute which gives the no. of
                                  " requests the resource is
                                  " currently satisfying.

 let .RWY.STATUS = N.X.RUNWAY(1)

 if .APPROACH.STATUS gt 0              " approach is occupied

   call STATUS.APP.OCC

 else

   if .RWY.STATUS gt 0 and OCCUPANT.OPERATION eq ..LANDING

     call STATUS.RWY.OCC

   else

     activate a FINAL.APPROACH.OPERATION now

   always

 always

 return

end " sys.check.at.stack
routine SYS.CHECK.AT.THRLD

 "This routine checks the system conditions at runway threshold for departures.
 "Depending on the condition the appropriate routine is called for scheduling
 "departures.
 "CALLED FROM : CREATE.AIRCRAFT.DEP
 "CALLS : DEPARTURE.OPERATION, STATUS.APP.CLR.RWY.OCC, STATUS.APP.OCC.RWY.CLR
 "       STATUS.APP.OCC.RWY.OCC

  define .APP.STATUS,

       .RWY.STATUS as integer variables

 if COUNT.DEP gt COUNT.T.OFF

  let .APP.STATUS = N.X.FINAL.APPROACH(1)

  let .RWY.STATUS = N.X.RUNWAY(1)

  if .APP.STATUS eq 0  and .RWY.STATUS eq 0   "approach and runway clear

    activate a DEPARTURE.OPERATION now

  else

    if .APP.STATUS eq 0 and .RWY.STATUS gt 0  "approach clear and
                                  "runway occupied
```

```
      call STATUS.APP.CLR.RWY.OCC

    else

      if .APP.STATUS gt 0 and .RWY.STATUS eq 0 " approach occupied and
                                          " runway clear

        call STATUS.APP.OCC.RWY.CLR

      else

        if .APP.STATUS gt 0 and .RWY.STATUS gt 0  "both approach and
                                          " runway occupied

          call STATUS.APP.OCC.RWY.OCC

        always

      always

    always

  always

 always

 return

end "sys.check.at.thrld
Routine VECTOR

  Given
     .X1,          "Coordinates of first point
     .Y1,
     .X2,          "Coordinates of second point
     .Y2,
     .SPEED        "Speed of object

  Yielding
     .TIME,        "Time to go from first to second point
     .DIRECTION    "Direction from first to second point (radians)

  Define .X1,
       .Y1,
       .X2,
       .Y2,
       .SPEED,
       .TIME,
       .DIRECTION,
       .DELTA.X and
       .DELTA.Y
  as real variables

  Let .DELTA.X = .X2 - .X1
  Let .DELTA.Y = .Y2 - .Y1
  Let .TIME = sqrt.f(.DELTA.X ** 2 + .DELTA.Y ** 2) / .SPEED
  Let .DIRECTION = arctan.f(.DELTA.Y, .DELTA.X)

End "VECTOR
```

```
process WAIT.AT.RAMP

" This process allows the aircraft to wait for some more time defined by
" WAIT.TIME.RAMP  before the program again checks the system to know the
" status.
" CALLED FROM: STATUS.APP.CLR.RWY.OCC, STATUS.APP.OCC.RWY.CLR,
"              STATUS.APP.OCC.RWY.OCC
" CALLS:

   wait WAIT.TIME.RAMP  seconds

   call SYS.CHECK.AT.THRLD

   return

end " wait.at.ramp
process WAIT.AT.STACK

" This process allows the aircraft to wait for some more time defined by
" WAIT.TIME.STACK before the aircraft is allowed to leave the stack.
" CALLED FROM; PRECED.FASTER.ACFT, PRECEDE.SLOWER.ACFT
" CALLS:


   wait  WAIT.TIME.STACK seconds

   activate a FINAL.APPROACH.OPERATION now

   return

end " wait.at.stack
 routine YES.NO.MESSAGE yielding .YES.NO

 " This routine prints a choice meesage to edit any data in the present screen
 " CALLED FROM: NEW.APP.STRT
 " CALLS:

   define .YES.NO as a text variable

   call vbcolor.r(5)

   call vgotoxy.r(22,1)

   print 1 line thus
 DO YOU WANT TO CHANGE ANY VALUE (Y/N) :

   call vgotoxy.r(22,42)

  read .YES.NO

  let .YES.NO = upper.f(.YES.NO)

  let .YES.NO = fixed.f(.YES.NO,1)

  call vbcolor.r(1)

  call vgotoxy.r(22,0)

  call vclearl.r
```

```
  return

end " yes.no.message
```

Appendix B. RUNSIM Aircraft Data File (ACFT.DAT)

```
* AIRCRAFT DATA IS GIVEN BELOW
* FOR TYPE: 3-HEAVY, 2-LARGE, 1-SMALL
* AIRCRAFT  TYPE MALW  OEW   CL   DEC  FREE  WING   WING
*  NAME                  MAX  M/   ROLL  AREA  SPAN
*            (KG'S) (KG'S)(DIM) S-S  (S)  (Sq.M)  (M)
9  END OF COMMENTS; DATA STARTS

30
A-300-600    1  140000 89669  3.0009 1.90 3.00 260.00 44.84
A-310-300    2  123000 80500  3.1912 2.00 3.00 219.00 43.90
A-320-200    2  64500  39980  2.9941 1.89 3.00 122.40 33.91
FOKKER-100   2  39915  24541  2.5326 2.19 3.00 93.50  28.08
BAe-146-200  2  36740  23882  3.3849 2.43 3.00 77.30  26.34
B-727-200    2  72200  46675  2.5580 1.89 3.00 157.30 32.92
B-737-300    2  51710  31479  2.8425 2.17 3.00 105.40 28.88
B-747-400    1  285765 178661 2.7859 1.36 3.00 511.00 64.94
B-757-200    2  89810  57180  2.8866 1.98 3.00 185.25 38.05
B-767-300    1  129273 83778  2.4480 1.81 3.00 283.30 47.57
MD-83        2  63276  36543  2.9319 1.89 3.00 118.00 32.87
DC-10-30     1  182798 121198 2.6595 1.85 3.00 367.70 50.40
PA-38-112    3  757    502    1.5748 2.04 2.00 11.59  10.36
PA-32-301    3  1640   878    1.6472 2.84 2.00 16.63  11.02
BE-F33A      3  1542   964    2.1300 1.79 2.00 16.81  10.21
CE-210P      3  1772   1052   1.9925 2.45 2.00 16.50  11.20
BE-58        3  2500   1579   1.4859 2.19 2.00 18.52  11.53
BE-300       3  6363   3851   2.0760 2.13 2.00 28.16  16.61
CE-421       3  3266   2298   1.4310 3.23 2.00 19.97  12.53
CE-550       3  5773   3351   1.7680 3.60 3.00 30.00  15.90
LEARJET-31   3  6940   4514   2.4130 3.13 3.00 24.57  13.34
G1159C       2  26535  18098  1.5472 4.37 3.00 89.29  23.72
BAe-125-800  2  10590  6858   2.1704 4.18 3.00 34.75  15.66
CL-601-3A    2  16363  2937   2.1780 4.06 3.00 42.00  19.61
SAAB-340     3  11566  7194   2.5779 1.67 2.00 41.80  21.44
BAe-31       3  6600   4103   2.1346 1.98 2.00 25.20  15.85
EMB-120      3  11250  7070   2.2722 1.61 2.00 39.43  19.28
DHC-8-100    3  15375  10024  3.2894 1.71 2.00 54.35  25.91
FOKKER-50    2  18890  12520  2.5417 1.88 2.00 70.00  29.00
SA-227-AT/41 3  6590   3833   1.8267 2.96 2.00 28.73  16.60
*
```

Appendix C. Turnoff Data File (TOT.DAT)

```
* TURNOFF OCCUPANCY TIME DATA IS GIVEN BELOW
* EXIT  NO OF   SPEED   INTERCEPT    RWY     ACFT
* TYPE  RECORDS                     WIDTH   WING SPAN
*
-999 DATA START;

   1    1
   1          8       3.739    0.073    0.158
   2    3
   2          8      13.239   -0.029    0.173
   2         12       1.386    0.209    0.204
   2         17       8.237    0.072    0.108
   3    5
   3          8      11.344    0.175    0.266
   3         13       7.148    0.175    0.266
   3         18       4.255    0.109    0.170
   3         23       3.749    0.073    0.112
   3         26       3.388    0.061    0.094
   4    4
   4          8      26.745    0.197    0.287
   4         13      22.409    0.196    0.286
   4         23       9.428    0.095    0.141
   4         26       8.237    0.072    0.108
   5    3
   5          8      18.133    0.137    0.220
   5         14      11.973    0.136    0.218
   5         17       8.318    0.103    0.173
   6    5
   6          8      23.214    0.236    0.329
   6         15      14.838    0.235    0.327
   6         22       7.971    0.125    0.177
   6         29       6.346    0.078    0.109
   6         36       5.136    0.058    0.081
   7    5
   7          8      23.811    0.257    0.351
   7         15      15.445    0.257    0.350
   7         22       7.945    0.140    0.196
   7         29       6.473    0.085    0.118
   7         36       5.258    0.064    0.088
   8    5
   8          8      15.576    0.162    0.253
   8         15       7.456    0.159    0.250
   8         22       5.517    0.140    0.196
   8         29       4.259    0.050    0.079
```

8		36	3.441	0.039	0.060
9	5				
9		8	25.291	0.248	0.341
9		15	16.866	0.247	0.339
9		22	8.632	0.136	0.196
9		29	6.924	0.083	0.114
9		36	5.601	0.062	0.085
10	5				
10		8	21.798	0.248	0.343
10		12	18.342	0.247	0.342
10		16	10.912	0.247	0.342
10		20	7.254	0.165	0.236
10		24	6.919	0.110	0.154

Appendix D. Source Code of Aircraft Turnoff Time Evaluation Program

```
Program to estimate the turnoff time of various aircraft and various turnoff angles.

   OPEN "Results" FOR OUTPUT AS # 11

200  CALL title (runwid,extype)

FOR acftype = 1 TO 34


  SELECT CASE acftype

    CASE IS = 1          'Piper Tomahawk (PA-28)

    icat = 1
    acfmass =  757              ' turnoff generating aircraft mass (kilograms)
    acflm = 77.5                 ' Percent mass on main gear (%)
    acfwb = 1.45                 ' Aircraft wheelbase (meters)
    acfws = 10.36                ' Aircraft wingspan (meters)
    tspan = 3.56                    ' Aircraft tail span (meters)
    nttip = 5.68                   ' Distance from main gear to tail plane ref. point
    nwtip = 1.76                   ' Distance from nose gear to wing tip plane

    CASE IS  = 2         'Piper Dakotta II

    icat = 1
    acfmass = 1363              ' turnoff generating aircraft mass (kilograms)
    acflm = 81.73                    ' Percent mass on main gear (%)
    acfwb = 1.98                  ' Aircraft wheelbase (meters)
    acfws = 10.92                    ' Aircraft wingspan (meters)
    tspan = 3.15                  ' Aircraft tail span (meters)
    nttip = 8.56                  ' Distance from main gear to tail plane ref. point
    nwtip = 3.1                   ' Distance from nose gear to wing tip plane

    CASE IS  = 3         'Cessna Skyhawk

    icat = 1
    acfmass = 1090              ' turnoff generating aircraft mass (kilograms)
    acflm = 77.93                    ' Percent mass on main gear (%)
    acfwb = 1.7                   ' Aircraft wheelbase (meters)
    acfws = 10.92                    ' Aircraft wingspan (meters)
    tspan = 2.78                  ' Aircraft tail span (meters)
    nttip = 5.78                  ' Distance from main gear to tail plane ref. point
    nwtip = 1.9                   ' Distance from nose gear to wing tip plane

   CASE IS = 4           'Beechcraft Baron 58P

    icat = 2
    acfmass =  2500             ' turnoff generating aircraft mass (kilograms)
    acflm = 84.73                    ' Percent mass on main gear (%)
    acfwb = 2.72                  ' Aircraft wheelbase (meters)
```

```
acfws = 11.53                    ' Aircraft wingspan (meters)
tspan = 3.34                     ' Aircraft tail span (meters)
nttip = 5.7                      ' Distance from main gear to tail plane ref. point
nwtip = 2.25                     ' Distance from nose gear to wing tip plane


CASE IS  = 5          'Piper Malibu

icat = 1
acfmass = 1772                   ' turnoff generating aircraft mass (kilograms)
acflm = 83.31                    ' Percent mass on main gear (%)
acfwb = 2.44                     ' Aircraft wheelbase (meters)
acfws = 13.66                    ' Aircraft wingspan (meters)
tspan = 3.8                      ' Aircraft tail span (meters)
nttip = 6.1                      ' Distance from main gear to tail plane ref. point
nwtip = 2.5                      ' Distance from nose gear to wing tip plane

CASE IS  = 6          'Piper Navajo (PA-31)

icat = 1
acfmass = 4082                   ' turnoff generating aircraft mass (kilograms)
acflm = 83.31                    ' Percent mass on main gear (%)
acfwb = 3.31                     ' Aircraft wheelbase (meters)
acfws = 12.52                    ' Aircraft wingspan (meters)
tspan = 6.21                     ' Aircraft tail span (meters)
nttip = 8.48                     ' Distance from main gear to tail plane ref. point
nwtip = 3.52                     ' Distance from nose gear to wing tip plane


CASE IS  = 7          'Cessna 402C

icat = 1
acfmass = 3107                   ' turnoff generating aircraft mass (kilograms)
acflm = 83.31                    ' Percent mass on main gear (%)
acfwb = 4.15                     ' Aircraft wheelbase (meters)
acfws = 13.45                    ' Aircraft wingspan (meters)
tspan = 5.43                     ' Aircraft tail span (meters)
nttip = 8.9                      ' Distance from main gear to tail plane ref. point
nwtip = 4.16                     ' Distance from nose gear to wing tip plane

CASE IS  = 8          'Embraer 120 Brasilia

icat = 2
acfmass = 11250                  ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                    ' Percent mass on main gear (%)
acfwb = 6.97                     ' Aircraft wheelbase (meters)
acfws = 19.78                    ' Aircraft wingspan (meters)
tspan = 6.82                     ' Aircraft tail span (meters)
nttip =  17.58                   ' Distance from main gear to tail plane ref. point
nwtip =  7.57                    ' Distance from nose gear to wing tip plane

CASE IS  = 9          'Beech 1900

icat = 2
acfmass = 7303                   ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                    ' Percent mass on main gear (%)
acfwb = 7.17                     ' Aircraft wheelbase (meters)
acfws = 16.61                    ' Aircraft wingspan (meters)
```

```
tspan = 5.63                    ' Aircraft tail span (meters)
nttip = 17.34                   ' Distance from main gear to tail plane ref. point
nwtip = 7.514                   ' Distance from nose gear to wing tip plane

CASE IS  = 10          'BAe ATP

icat = 2
acfmass = 21733             ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                   ' Percent mass on main gear (%)
acfwb = 8.59                  ' Aircraft wheelbase (meters)
acfws = 30.63                 ' Aircraft wingspan (meters)
tspan = 10.87                 ' Aircraft tail span (meters)
nttip = 20.3                ' Distance from main gear to tail plane ref. point
nwtip = 9.75                  ' Distance from nose gear to wing tip plane

CASE IS  = 11          'Aerospatiale ATR 42

icat = 2
acfmass = 15500             ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                   ' Percent mass on main gear (%)
acfwb = 8.901                 ' Aircraft wheelbase (meters)
acfws = 24.57                 ' Aircraft wingspan (meters)
tspan = 7.31                  ' Aircraft tail span (meters)
nttip = 20.57                 ' Distance from main gear to tail plane ref. point
nwtip = 9.49                  ' Distance from nose gear to wing tip plane

CASE IS  = 12          'CASA/Nurtanio 235

icat = 2
acfmass = 14200             ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                   ' Percent mass on main gear (%)
acfwb = 6.83                  ' Aircraft wheelbase (meters)
acfws = 25.81                 ' Aircraft wingspan (meters)
tspan = 10.89                 ' Aircraft tail span (meters)
nttip = 18.36                 ' Distance from main gear to tail plane ref. point
nwtip = 7.259                 ' Distance from nose gear to wing tip plane

CASE IS  = 13          'Shorts 330

icat = 2
acfmass = 10251             ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                   ' Percent mass on main gear (%)
acfwb = 6.37                  ' Aircraft wheelbase (meters)
acfws = 22.76                 ' Aircraft wingspan (meters)
tspan = 5.75                  ' Aircraft tail span (meters)
nttip = 15.48                 ' Distance from main gear to tail plane ref. point
nwtip = 6.74                  ' Distance from nose gear to wing tip plane


CASE IS  = 14          'Saab 340

icat = 2
acfmass = 12020             ' turnoff generating aircraft mass (kilograms)
acflm = 90.88                   ' Percent mass on main gear (%)
acfwb = 7.04                  ' Aircraft wheelbase (meters)
acfws = 21.44                 ' Aircraft wingspan (meters)
tspan = 7.92                  ' Aircraft tail span (meters)
nttip = 15.84                 ' Distance from main gear to tail plane ref. point
```

```
nwtip = 9.86                        ' Distance from nose gear to wing tip plane

CASE IS =15          'BAe 31 "Jetstream"

icat = 2
acfmass = 6600                  ' turnoff generating aircraft mass (kilograms)
acflm = 87.18                       ' Percent mass on main gear (%)
acfwb = 4.6                       ' Aircraft wheelbase (meters)
acfws = 15.85                       ' Aircraft wingspan (meters)
tspan = 6.79                      ' Aircraft tail span (meters)
nttip = 11.615                      ' Distance from main gear to tail plane ref. point
nwtip = 4.82                      ' Distance from nose gear to wing tip plane

CASE IS = 16         'Fairchild Metro SA 227

icat = 2
acfmass =  6350                 ' turnoff generating aircraft mass (kilograms)
acflm = 84.73                       ' Percent mass on main gear (%)
acfwb = 5.984                     ' Aircraft wheelbase (meters)
acfws = 17.37                     ' Aircraft wingspan (meters)
tspan = 4.643                     ' Aircraft tail span (meters)
nttip = 14.75                     ' Distance from main gear to tail plane ref. point
nwtip = 6.273                     ' Distance from nose gear to wing tip plane


CASE IS = 17         'Boeing DeHavilland DHC-8

icat = 2
acfmass =  14923                ' turnoff generating aircraft mass (kilograms)
acflm = 84.73                       ' Percent mass on main gear (%)
acfwb = 7.92                      ' Aircraft wheelbase (meters)
acfws = 25.91                       ' Aircraft wingspan (meters)
tspan = 7.92                      ' Aircraft tail span (meters)
nttip = 19.9                      ' Distance from main gear to tail plane ref. point
nwtip = 8.56                      ' Distance from nose gear to wing tip plane

CASE IS = 18         'Douglas DC-8-73

icat = 2
acfmass =  117000&              ' turnoff generating aircraft mass (kilograms)
acflm = 91                        ' Percent mass on main gear (%)
acfwb = 21.59                       ' Aircraft wheelbase (meters)
acfws = 45.2                      ' Aircraft wingspan (meters)
tspan = 13.47                       ' Aircraft tail span (meters)
nttip = 48.79                     ' Distance from main gear to tail plane ref. point
nwtip = 33.24                       ' Distance from nose gear to wing tip plane

CASE IS = 19         'Douglas DC-9-50

icat = 2
acfmass =  49895&               ' turnoff generating aircraft mass (kilograms)
acflm = 90                        ' Percent mass on main gear (%)
acfwb = 18.01                       ' Aircraft wheelbase (meters)
acfws = 28.47                       ' Aircraft wingspan (meters)
tspan = 11.43                     ' Aircraft tail span (meters)
nttip = 36.02                     ' Distance from main gear to tail plane ref. point
nwtip = 21.48                       ' Distance from nose gear to wing tip plane
```

```
CASE IS = 20          'Douglas DC-10-30

icat = 2
acfmass =  182766&                  ' turnoff generating aircraft mass (kilograms)
acflm = 93                          ' Percent mass on main gear (%)
acfwb = 26.17                       ' Aircraft wheelbase (meters)
acfws = 50.4                        ' Aircraft wingspan (meters)
tspan = 22.56                       ' Aircraft tail span (meters)
nttip = 45.57                       ' Distance from main gear to tail plane ref. point
nwtip = 41.51                       ' Distance from nose gear to wing tip plane

CASE IS = 21          'MD-11

icat = 2
acfmass =  195040&                  ' turnoff generating aircraft mass (kilograms)
acflm = 92                          ' Percent mass on main gear (%)
acfwb = 24.8                        ' Aircraft wheelbase (meters)
acfws = 51.7                        ' Aircraft wingspan (meters)
tspan = 16.36                       ' Aircraft tail span (meters)
nttip = 54.63                       ' Distance from main gear to tail plane ref. point
nwtip = 44.63                       ' Distance from nose gear to wing tip plane


CASE IS = 22          'MD-82

icat = 2
acfmass =  58060&                   ' turnoff generating aircraft mass (kilograms)
acflm = 89                          ' Percent mass on main gear (%)
acfwb = 21.79                       ' Aircraft wheelbase (meters)
acfws = 32.87                       ' Aircraft wingspan (meters)
tspan = 11.82                       ' Aircraft tail span (meters)
nttip = 42.47                       ' Distance from main gear to tail plane ref. point
nwtip = 29.55                       ' Distance from nose gear to wing tip plane

CASE IS = 23          'Fokker F-28

icat = 2
acfmass =  31525                    ' turnoff generating aircraft mass (kilograms)
acflm = 88                          ' Percent mass on main gear (%)
acfwb = 10.37                       ' Aircraft wheelbase (meters)
acfws = 25.07                       ' Aircraft wingspan (meters)
tspan = 8.72                        ' Aircraft tail span (meters)
nttip = 25.92                       ' Distance from main gear to tail plane ref. point
nwtip = 12.25                       ' Distance from nose gear to wing tip plane

CASE IS = 24          'Fokker F-100

icat = 2
acfmass =  38330&                   ' turnoff generating aircraft mass (kilograms)
acflm = 88                          ' Percent mass on main gear (%)
acfwb = 14.07                       ' Aircraft wheelbase (meters)
acfws = 28.08                       ' Aircraft wingspan (meters)
tspan = 8.73                        ' Aircraft tail span (meters)
nttip = 31.88                       ' Distance from main gear to tail plane ref. point
nwtip = 16.37                       ' Distance from nose gear to wing tip plane


CASE IS  = 25          'Airbus A320
```

```
icat = 2
acfmass = 61000&                    ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                       ' Percent mass on main gear (%)
acfwb = 12.63                       ' Aircraft wheelbase (meters)
acfws = 33.91                       ' Aircraft wingspan (meters)
tspan = 12.94                       ' Aircraft tail span (meters)
nttip = 34.43                       ' Distance from main gear to tail plane ref. point
nwtip = 22.5                        ' Distance from nose gear to wing tip plane

CASE IS  = 26          'British Aerospace 146-200

icat = 3
acfmass = 36740&                    ' turnoff generating aircraft mass (kilograms)
acflm = 90.26                       ' Percent mass on main gear (%)
acfwb = 11.2                        ' Aircraft wheelbase (meters)
acfws = 26.34                       ' Aircraft wingspan (meters)
acfwt = 4.7                         ' Aircraft wheeltrack (meters)
tspan = 12.06                       ' Aircraft tail span (meters)
nttip = 26.05                       ' Distance from main gear to tail plane ref. point
nwtip = 12.82                       ' Distance from nose gear to wing tip plane

CASE IS = 27           'BAc/CNIAR 1-11

icat = 2
acfmass =  39462&                   ' turnoff generating aircraft mass (kilograms)
acflm = 89                          ' Percent mass on main gear (%)
acfwb = 12.67                       ' Aircraft wheelbase (meters)
acfws = 28.5                        ' Aircraft wingspan (meters)
tspan = 8.89                        ' Aircraft tail span (meters)
nttip = 28.03                       ' Distance from main gear to tail plane ref. point
nwtip =  15.63                      ' Distance from nose gear to wing tip plane


CASE IS  = 28          'Boeing 707-131

icat = 3
acfmass = 121000&                   ' turnoff generating aircraft mass (kilograms)
acflm = 92!                         ' Percent mass on main gear (%)
acfwb = 18.5                        ' Aircraft wheelbase (meters)
acfws = 44.42                       ' Aircraft wingspan (meters)
acfwt = 6.73                        ' Aircraft wheeltrack (meters)
tspan = 13.8                        ' Aircraft tail span (meters)
nttip = 41.61                       ' Distance from main gear to tail plane ref. point
nwtip = 28.51                       ' Distance from nose gear to wing tip plane

CASE IS = 29           'Boeing 737-300

icat = 2
acfmass =  51710&                   ' turnoff generating aircraft mass (kilograms)
acflm = 89                          ' Percent mass on main gear (%)
acfwb =  12.25                      ' Aircraft wheelbase (meters)
acfws = 28.88                       ' Aircraft wingspan (meters)
tspan = 12.25                       ' Aircraft tail span (meters)
nttip = 28.68                       ' Distance from main gear to tail plane ref. point
nwtip = 16.16                       ' Distance from nose gear to wing tip plane


CASE IS  = 30          'Boeing 727-200
```

```
icat = 3
acfmass = 72200&                    ' turnoff generating aircraft mass (kilograms)
acflm = 92.5                          ' Percent mass on main gear (%)
acfwb = 16.75                         ' Aircraft wheelbase (meters)
acfws = 36.75                         ' Aircraft wingspan (meters)
acfwt = 5.71                          ' Aircraft wheeltrack (meters)
tspan = 14.55                         ' Aircraft tail span (meters)
nttip = 42.11                         ' Distance from main gear to tail plane ref. point
nwtip = 25.3                          ' Distance from nose gear to wing tip plane

CASE IS = 31          'Boeing 757-200

icat = 2
acfmass =  89830&                   ' turnoff generating aircraft mass (kilograms)
acflm = 90                            ' Percent mass on main gear (%)
acfwb = 18.08                         ' Aircraft wheelbase (meters)
acfws = 38.05                         ' Aircraft wingspan (meters)
tspan = 14.5                          ' Aircraft tail span (meters)
nttip = 40.44                         ' Distance from main gear to tail plane ref. point
nwtip = 22.51                         ' Distance from nose gear to wing tip plane

CASE IS = 32          'Boeing 767-200

icat = 2
acfmass =  123733&                  ' turnoff generating aircraft mass (kilograms)
acflm = 92                            ' Percent mass on main gear (%)
acfwb = 22.59                         ' Aircraft wheelbase (meters)
acfws = 47.57                         ' Aircraft wingspan (meters)
tspan = 18.16                         ' Aircraft tail span (meters)
nttip = 49.17                         ' Distance from main gear to tail plane ref. point
nwtip = 27.33                         ' Distance from nose gear to wing tip plane


 CASE IS  = 33          'Boeing 747-400

icat = 4
acfmass = 285750&                   ' turnoff generating aircraft mass (kilograms)
acflm = 94!                           ' Percent mass on main gear (%)
acfwb = 25.6                          ' Aircraft wheelbase (meters)
acfws = 63.3                          ' Aircraft wingspan (meters)
acfwt = 11.05                         ' Aircraft wheeltrack (meters)
tspan = 21.08                         ' Aircraft tail span (meters)
nttip = 62.11                         ' Distance from main gear to tail plane ref. point
nwtip = 41.03                         ' Distance from nose gear to wing tip plane

CASE IS  = 34          'Airbus A300-600

icat = 4
acfmass = 140000&                   ' turnoff generating aircraft mass (kilograms)
acflm = 92.5                          ' Percent mass on main gear (%)
acfwb = 18.6                          ' Aircraft wheelbase (meters)
acfws = 44.8                          ' Aircraft wingspan (meters)
acfwt = 9.61                          ' Aircraft wheeltrack (meters)
tspan = 15.96                         ' Aircraft tail span (meters)
nttip = 46.96                         ' Distance from main gear to tail plane ref. point
nwtip = 32.8                          ' Distance from nose gear to wing tip plane

END SELECT
```

```
' Definition exit parameters for satndard geometries
   IF extype = 6 THEN GOTO 100

   SELECT CASE extype

      CASE IS = 1              '90 Deg. FAA Standard Geometry

            vexit = 8                      ' standard exit speed (m/sec.)
            theta = 90/57.29               ' standard turnoff angle (deg.)
            Rapath = 77                    ' Largest radius of curvature (m.)

      CASE IS  = 2             '45 Deg. FAA Standard

            vexit = 8                      ' standard exit speed (m/sec.)
            theta = 45/57.29               ' standard turnoff angle (deg.)
            Rapath = 244                   ' Standard radius for 45 Deg. exit

      CASE IS  = 3             '30 Deg. FAA Standard Geometry

            vexit = 26.7                   ' standard exit speed (m/sec.)
            theta = 30/57.29               ' standard turnoff angle (deg.)
            Rapath = 550                   ' Radius of curvature (m)

      CASE IS  = 4             '30 Deg. with 1400 ft. spiral

            vexit = 26.7                   ' standard exit speed (m/sec.)
            theta = 30/57.29               ' standard turnoff angle (deg.)
            Rapath = 550                   ' Radius of curvature (m)

      CASE IS  = 5             'Wide Throat Geometry Estimation Parameters

            vexit = 17                     ' standard exit speed (m/sec.)
            theta = 90/57.29               ' standard turnoff angle (deg.)
            Rapath = 550                   ' Radius of curvature (m)


      END SELECT

     ' General Algorithm for Standard Geoemtries

     ' Initialize exit parameters

        xpath = 0
        ypath = 0
        vpath = vexit
        sai = 0
        tpath = 0!
        dt = .05
        count = 0!
        saifus = 0!
        tprint = 0!

        xm = xpath - acfwb
        ym = ypath

        xtip = xm
        ytip = ypath - (acfws/2!)
```

```
    LOCATE 4, 5 : PRINT "Time (s)     Xcord(m)     Ycoord. (m.)  Heading    Speed     Heading"
     LOCATE 6,4: PRINT USING"######.###"; tpath,  xpath, ypath, saifus, vpath, sai


     IF (extype = 4 OR extype = 5) THEN

5      IF (extype = 5) THEN
        dxpath = 1 - .002 *xpath
        ELSE
        dxpath = 1 - .000625 * xpath
        END IF

     xpath1 = xpath + dxpath

     IF (extype = 5) THEN
           ypath1 = 1.3763*10^(.006301*xpath1)
      ELSE
           ypath1 = -.202 + .012 * xpath1 - .0001072 * xpath1^2 + 1.2086E-06 * xpath1^3
      END IF

     dypath = ypath1 - ypath

     ds = SQR(dxpath^2 + dypath^2)
     dt = ds / vpath

     IF (sai < theta*57.29) THEN
        dsai = ATN (dypath / dxpath)
     ELSE
        dsai = 0!
     END IF


      IF (vpath < 8) THEN
           accpath = 0!
       ELSE
           accpath = -.375
       END IF

        ' Integration of nose gear parameters

        xpath = xpath1
        ypath = ypath1
        vpath = vpath + dt*accpath
        sai = sai + dsai
        tpath = tpath + dt
        tprint = tprint + dt

        saifus = (ypath - ym) / (xpath - xm)          ' Fuselage instantaneous heading angle

        xm = xm + dt * vpath * COS (saifus)           ' Main gear centerline trajectory
        ym = ym + dt * vpath * SIN (saifus)


        xwtip = xpath - nwtip * COS (saifus) + (acfws/2) * SIN (saifus)
        ywtip = ypath - nwtip * SIN (saifus) - (acfws/2) * COS (saifus)

        xttip = xpath - nttip * COS (saifus) + (tspan/2) * SIN (saifus)
        yttip = ypath - nttip * SIN (saifus) - (tspan/2) * COS (saifus)
```

```
      said = sai*57.29
      thetad = theta*57.29

    IF tprint > .5 THEN

     PRINT USING"#######.###"; tpath,  xpath, ypath,said, vpath, sai
      tprint = 0!
     END IF

   IF  (ywtip < runwid/2!  OR  yttip < runwid/2!)  GOTO 5

   PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND

   CLS

   ELSE
        GOTO 3
   END IF


3       'Estimation of Rate Variables

        dspath = vpath*dt
        dsai = dspath/Rapath

      ' Estimation of heading rate of change

        IF (sai < theta) THEN
           saidot = dsai/dt
         ELSE
            saidot = 0!
        END IF

      ' Estimation of deceleration rate of change

        IF (vpath < 8) THEN
           accpath = 0!
        ELSE
           accpath = -.375
        END IF

      ' Estimation of downrange position changes

        xdot = dspath*COS(sai)/dt
        ydot = dspath*SIN (sai)/dt

      ' Integration of nose gear parameters

        xpath = xpath + dt*xdot
        ypath = ypath + dt*ydot
        vpath = vpath + dt*accpath
        sai = sai + dt* saidot
        tpath = tpath + dt

        saifus = (ypath - ym) / (xpath - xm)          ' Fuselage instantaneous heading angle

        xm = xm + dt * vpath * COS (saifus)           ' Main gear centerline trajectory
        ym = ym + dt * vpath * SIN (saifus)
```

```
        xwtip = xpath - nwtip * COS (saifus) + (acfws/2) * SIN (saifus)
        ywtip = ypath - nwtip * SIN (saifus) - (acfws/2) * COS (saifus)

        xttip = xpath - nttip * COS (saifus) + (tspan/2) * SIN (saifus)
        yttip = ypath - nttip * SIN (saifus) - (tspan/2) * COS (saifus)

        said = sai*57.29
        thetad = theta*57.29

        count = count + 1

      IF count = 10 THEN

       PRINT USING"#######.###"; tpath,  Rapath, xpath, ypath,said, vpath
        count = 0!
       END IF

    IF  (ywtip < runwid/2!  OR  yttip < runwid/2!)  GOTO 3

       'PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND

    CLS

    PRINT "Geometry Parameters"
    PRINT " "
    PRINT " Aircraft Type                          = ", actype
    PRINT " Turnoff Angle                           = ", thetad
    PRINT  " Exit Speed                              =", vexit
    PRINT  " Turnoff Time                           =" , tpath
    PRINT " Aircraft Wingspan                      = ", acfws
    PRINT  " Distance from NG to Wingtip plane     =", nwtip
    PRINT  " Distance from NG to Tailtip plane     =" , nttip
    PRINT  " Final Fuselage Exit Angle              =" , said

       WRITE # 11, acftype, thetad, vexit, tpath, acfws, nwtip, nttip, said

      'PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND
      CLS

NEXT acftype

100    CLOSE # 11

      GOTO 500


100   ' Define constants of the model (only for REDIM computations)

    vars=1                                  '1 - variable mu, 2 - constant mu
    n=1
    g= 9.81                                  ' gravity force (m/sec-sec)
    pi = 3.141592654#                            ' pi
    froll = .03                             ' free roll friction coefficient (dimensionless)
    dt = .05                               ' simulation step size (seconds)
    dx = 5
    dy = 5
    rho = 1.125
```

```
countl=0!
jerkmax = .55                          ' Maximum jerk in m/sec-sec-sec
anmax = 1.2

pcount = 0

   'Estimation of the aircraft cornering capabilities (inertia limitations in the turn)
   ' Estimate the yaw inertia (IZZ) of the aircraft in question

   acfyaw =  10 ^ ( 1.7215 * LOG (acfmass)  / LOG (10) - 1.673 )

IF vars = 1 THEN
   CALL skid(icat, iwet, ymumax, vexit, sf)
ELSE
   CALL fixskid(iwet,ymumax)
END IF


' Initialization of variables (aircraft generating the turnoff geometry)

xpath = 0
ypath = 0
vpath = vexit
saig = 0
castor = 0
tpath = 0
tpath1 = 0
Rapath = 5000

xm = xpath - acfwb
ym = ypath

xtip = xm
ytip = ypath - (acfws/2!)

  LOCATE 4, 5 : PRINT "Time (s)    Radius (m)    Xcord(m)    Ycoord. (m.)    Heading    Speed"
  LOCATE 6,4: PRINT USING"######.###"; tpath,  Rapath, xpath, ypath, saifus, vpath


' Estimation of rate variables (in the turn alone)

rdot1 = acfmass * acfwb
rdot0 = (acflm / 100!) * ( 1 - acflm /100!)

2  IF vpath <= 10! THEN                    ' Clip the value of deceleration to zero
      acpath = 0!                          ' if speed is reduced below taxiing value
   ELSE
     acpath = (-1) * g * froll
   END IF

    'Estimation of scrubbing coefficient and steering algorithm

   CALL steer ( Rapath, vpath, saig, saidotg, ypath, theta)
   CALL scrub (Rapath, acfmass, ymusc)

    ymuc = vpath ^ 2 / (Rapath * g)
    cenacc= ymuc*g
    ymui = ymumax - ymusc - ymuc
```

```
rdot2 = -(ymui * g * Rapath ^ 2 / ( acfyaw * vpath))
rdot = rdot0 * rdot1 * rdot2

' Estimation of the normal acceleration (an) and jerk-related parameters

 an = vpath^2/Rapath
 rdotjerk = - Rapath^2 * jerkmax / vpath^2
 jerk = an * vpath / lpath
 rdotan = - Rapath^2 * anmax / (lpath * vpath)

 xjerk = xpath
 yjerk = jerkmax * lpath^3/ (6*vpath^3)

 IF ABS(rdot) > ABS(rdotjerk) THEN
    rdot = rdotjerk
 END IF

' Integrate aircraft generating exit parameters

lpath = lpath + dt * vpath
saig = saig + dt * saidotg
Rapath = Rapath + dt * rdot
xpath = xpath + dt * vpath * COS (saig)
ypath = ypath + dt * vpath * SIN (saig)
vpath = vpath + dt * acpath
tpath = tpath + dt
count = count + 1

saifus = (ypath - ym) / (xpath - xm)              ' Fuselage instantaneous heading angle

    xm = xm + dt * vpath * COS (saifus)              ' Main gear centerline trajectory
    ym = ym + dt * vpath * SIN (saifus)

    xwtip = xpath - nwtip * COS (saifus) + (acfws/2) * SIN (saifus)
    ywtip = ypath - nwtip * SIN (saifus) - (acfws/2) * COS (saifus)

    xttip = xpath - nttip * COS (saifus) + (tspan/2) * SIN (saifus)
    yttip = ypath - nttip * SIN (saifus) - (tspan/2) * COS (saifus)

saifusd = saifus*57.29
said = saig*57.29


 IF count = 10 THEN

  PRINT USING"#######.###"; tpath,  Rapath, xpath, ypath, saifusd, vpath
   count = 0!
  END IF

IF  (ywtip < runwid/2!  OR  yttip < runwid/2!)  GOTO 2

  'PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND
CLS

PRINT " REDIM Geometry Parameters
PRINT " "
PRINT " Turnoff Angle                              = ", theta
PRINT  " Exit Speed                                 =", vexit
```

```
    PRINT  " Turnoff Time                                =" , tpath
    PRINT " Aircraft Wingspan                           = ", acfws
    PRINT  " Distance from NG to Wingtip plane         =", nwtip
    PRINT  " Distance from NG to Tailtip plane         =" , nttip
    PRINT  " Final Fuselage Exit Angle                  =" , said

     'PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND

500     PRINT " Would you like another run - "
        PRINT " "
        PRINT "    1       - no "
        PRINT "    2       - yes "

         INPUT "rerun - ", rerun
        IF (rerun = 2) THEN
           GOTO 200
        END IF


      STOP
      END

' _____ Subroutines _____


    SUB steer ( Rapath, vpath, saig, saidotg, ypath, theta) STATIC

    tpol = 2                  ' straight ahead after (theta) heading


      IF tpol = 1 THEN
            IF saig < 1.41 THEN
               IF ypath < (distt/2!) THEN
                  saidotg = vpath / Rapath
               ELSE
                  saidotg = (-1) * vpath / Rapath
               END IF
            ELSE
               saidotg = (-1) * vpath / Rapath
            END IF
       ELSE
            IF saig < (theta/57.29) THEN
               saidotg = vpath / Rapath
            ELSE
               saidotg = 0
            END IF
       END IF

      END SUB

    SUB scrub (Rapath, acfmass, ymusc) STATIC

       IF Rapath < 50! THEN
          ymusc = .015
       ELSE
          IF acfmass > 165000& THEN
             k1 = - 43.8987 + .0002706 * acfmass
             k2 = .1807 - 4.88E-06 * acfmass
```

```
      ELSE
         k1 = 5.1349 - .0000266 * acfmass
         k2 = - 1.8417 + 7.37E-06 * acfmass
      END IF

      ymusc = k1 * Rapath ^ k2

   END IF

END SUB

SUB title (runwid, extype) STATIC

PRINT "
PRINT ' '
INPUT "Runway Width (meters) = ", runwid

PRINT "Enter the Exit Type"
PRINT "
PRINT "          1    -    90 Deg FAA Standard"
PRINT "          2    -    45 Deg FAA Standard"
PRINT "          3    -    30 Deg FAA Standard"
PRINT "          4    -    30 Deg FAA Standard with 1400 ft. Spiral"
PRINT "          5    -    Wide Throat Geometry"
PRINT "          6    -    REDIM Geometry"

 INPUT "Exit Type = ", extype

IF (extype = 6) THEN

   INPUT "Exit Speed (m/sec.) = ", vexit
   INPUT "Safety Factor for =|  (%) = ", sf
   INPUT "Turnoff Angle (degrees) = ", theta

END IF


CLS
END SUB


SUB skid(icat,iwet,ymumax,vexit, sf) STATIC

   IF iwet = 2 THEN

      SELECT CASE icat

      CASE IS = 1
         ymumax = .6919 - .0448*vexit + .0022*vexit^2 - 5.392E-05*vexit^3 + 4.847E-07*vexit^4

      CASE IS = 2
         ymumax = .6313 - .0406*vexit + .0021*vexit^2 - 5.413E-05*vexit^3 + 5.175E-07*vexit^4

      CASE IS = 3
         ymumax = .5725 - .0373*vexit + .0019*vexit^2 - .0000487*vexit^3 + 4.642E-07*vexit^4

      CASE IS = 4
         ymumax = .5153 - .0326*vexit + .0016*vexit^2 - 3.885E-05*vexit^3 + 3.579E-07*vexit^4
```

```
      END SELECT

   ELSE

      SELECT CASE icat

      CASE IS = 1
         k = .8515

      CASE IS = 2
         k = .773

      CASE IS = 3
         k = .6945

      CASE IS = 4
         k = .616

       END SELECT

       mumax = 1.001 - .0574*vexit + .0023*vexit^2 - .0000545*vexit^3 + 6.911E-07*vexit^4 - 3.567E-09*vexit^5
       ymumax = k * mumax

   END IF

ymumax = ymumax / (1 + sf/100 )          ' Sf is the safety factor applied to the ultimate =|


END SUB


   SUB fixskid(iwet,ymumax) STATIC
      IF iwet = 1 THEN
         ymumax = .35                          ' maximum side nose mu coefficient (dim.)
       ELSE
         ymumax = .2
      END IF

   END SUB

  SUB box STATIC

   LINE (10,10) - (600,380), , B

   END SUB


   SUB small.box STATIC
     LINE (20,20) - (400,180), , B
   END SUB
```

```
' Program to estimate the turnoff time of various aircraft and various turnoff angles.

   OPEN "Results" FOR OUTPUT AS # 11

200  CALL title (runwid,extype)

FOR acftype = 1 TO 34


  SELECT CASE acftype

    CASE IS = 1          'Piper Tomahawk (PA-28)

    icat = 1
    acfmass =  757              ' turnoff generating aircraft mass (kilograms)
    acflm = 77.5                 ' Percent mass on main gear (%)
    acfwb = 1.45                 ' Aircraft wheelbase (meters)
    acfws = 10.36                ' Aircraft wingspan (meters)
    tspan = 3.56                   ' Aircraft tail span (meters)
    nttip = 5.68                   ' Distance from main gear to tail plane ref. point
    nwtip = 1.76                   ' Distance from nose gear to wing tip plane

    CASE IS  = 2         'Piper Dakotta II

    icat = 1
    acfmass = 1363              ' turnoff generating aircraft mass (kilograms)
    acflm = 81.73                  ' Percent mass on main gear (%)
    acfwb = 1.98                 ' Aircraft wheelbase (meters)
    acfws = 10.92                  ' Aircraft wingspan (meters)
    tspan = 3.15                 ' Aircraft tail span (meters)
    nttip = 8.56                 ' Distance from main gear to tail plane ref. point
    nwtip = 3.1                  ' Distance from nose gear to wing tip plane

    CASE IS  = 3         'Cessna Skyhawk

    icat = 1
    acfmass = 1090              ' turnoff generating aircraft mass (kilograms)
    acflm = 77.93                  ' Percent mass on main gear (%)
    acfwb = 1.7                  ' Aircraft wheelbase (meters)
    acfws = 10.92                  ' Aircraft wingspan (meters)
    tspan = 2.78                 ' Aircraft tail span (meters)
    nttip = 5.78                 ' Distance from main gear to tail plane ref. point
    nwtip = 1.9                  ' Distance from nose gear to wing tip plane

   CASE IS = 4           'Beechcraft Baron 58P

    icat = 2
    acfmass =  2500             ' turnoff generating aircraft mass (kilograms)
    acflm = 84.73                  ' Percent mass on main gear (%)
    acfwb = 2.72                 ' Aircraft wheelbase (meters)
    acfws = 11.53                  ' Aircraft wingspan (meters)
    tspan = 3.34                 ' Aircraft tail span (meters)
```

```
nttip = 5.7                    ' Distance from main gear to tail plane ref. point
nwtip = 2.25                   ' Distance from nose gear to wing tip plane

CASE IS  = 5          'Piper Malibu

icat = 1
acfmass = 1772              ' turnoff generating aircraft mass (kilograms)
acflm = 83.31                  ' Percent mass on main gear (%)
acfwb = 2.44                ' Aircraft wheelbase (meters)
acfws = 13.66               ' Aircraft wingspan (meters)
tspan = 3.8                 ' Aircraft tail span (meters)
nttip = 6.1                 ' Distance from main gear to tail plane ref. point
nwtip = 2.5                 ' Distance from nose gear to wing tip plane

CASE IS  = 6          'Piper Navajo (PA-31)

icat = 1
acfmass = 4082              ' turnoff generating aircraft mass (kilograms)
acflm = 83.31                  ' Percent mass on main gear (%)
acfwb = 3.31                ' Aircraft wheelbase (meters)
acfws = 12.52               ' Aircraft wingspan (meters)
tspan = 6.21                ' Aircraft tail span (meters)
nttip = 8.48                ' Distance from main gear to tail plane ref. point
nwtip = 3.52                ' Distance from nose gear to wing tip plane

CASE IS  = 7          'Cessna 402C

icat = 1
acfmass = 3107              ' turnoff generating aircraft mass (kilograms)
acflm = 83.31                  ' Percent mass on main gear (%)
acfwb = 4.15                ' Aircraft wheelbase (meters)
acfws = 13.45               ' Aircraft wingspan (meters)
tspan = 5.43                ' Aircraft tail span (meters)
nttip = 8.9                 ' Distance from main gear to tail plane ref. point
nwtip = 4.16                ' Distance from nose gear to wing tip plane

CASE IS  = 8          'Embraer 120 Brasilia

icat = 2
acfmass = 11250             ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                  ' Percent mass on main gear (%)
acfwb = 6.97                ' Aircraft wheelbase (meters)
acfws = 19.78               ' Aircraft wingspan (meters)
tspan = 6.82                ' Aircraft tail span (meters)
nttip =  17.58              ' Distance from main gear to tail plane ref. point
nwtip =  7.57               ' Distance from nose gear to wing tip plane

CASE IS  = 9          'Beech 1900

icat = 2
acfmass = 7303              ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                  ' Percent mass on main gear (%)
acfwb = 7.17                ' Aircraft wheelbase (meters)
acfws = 16.61               ' Aircraft wingspan (meters)
tspan = 5.63                ' Aircraft tail span (meters)
nttip = 17.34               ' Distance from main gear to tail plane ref. point
```

```
nwtip = 7.514                          ' Distance from nose gear to wing tip plane

CASE IS  = 10          'BAe ATP

icat = 2
acfmass = 21733                  ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                          ' Percent mass on main gear (%)
acfwb = 8.59                         ' Aircraft wheelbase (meters)
acfws = 30.63                        ' Aircraft wingspan (meters)
tspan = 10.87                        ' Aircraft tail span (meters)
nttip = 20.3                         ' Distance from main gear to tail plane ref. point
nwtip = 9.75                         ' Distance from nose gear to wing tip plane

CASE IS  = 11          'Aerospatiale ATR 42

icat = 2
acfmass = 15500                  ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                          ' Percent mass on main gear (%)
acfwb = 8.901                        ' Aircraft wheelbase (meters)
acfws = 24.57                        ' Aircraft wingspan (meters)
tspan = 7.31                         ' Aircraft tail span (meters)
nttip = 20.57                        ' Distance from main gear to tail plane ref. point
nwtip = 9.49                         ' Distance from nose gear to wing tip plane

CASE IS  = 12          'CASA/Nurtanio 235

icat = 2
acfmass = 14200                  ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                          ' Percent mass on main gear (%)
acfwb = 6.83                         ' Aircraft wheelbase (meters)
acfws = 25.81                        ' Aircraft wingspan (meters)
tspan = 10.89                        ' Aircraft tail span (meters)
nttip = 18.36                        ' Distance from main gear to tail plane ref. point
nwtip = 7.259                        ' Distance from nose gear to wing tip plane

CASE IS  = 13          'Shorts 330

icat = 2
acfmass = 10251                  ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                          ' Percent mass on main gear (%)
acfwb = 6.37                         ' Aircraft wheelbase (meters)
acfws = 22.76                        ' Aircraft wingspan (meters)
tspan = 5.75                         ' Aircraft tail span (meters)
nttip = 15.48                        ' Distance from main gear to tail plane ref. point
nwtip = 6.74                         ' Distance from nose gear to wing tip plane


CASE IS  = 14          'Saab 340

icat = 2
acfmass = 12020                  ' turnoff generating aircraft mass (kilograms)
acflm = 90.88                          ' Percent mass on main gear (%)
acfwb = 7.04                         ' Aircraft wheelbase (meters)
acfws = 21.44                        ' Aircraft wingspan (meters)
tspan = 7.92                         ' Aircraft tail span (meters)
nttip = 15.84                        ' Distance from main gear to tail plane ref. point
nwtip = 9.86                         ' Distance from nose gear to wing tip plane
```

```
CASE IS  =15            'BAe 31 "Jetstream"

icat = 2
acfmass = 6600                  ' turnoff generating aircraft mass (kilograms)
acflm = 87.18                     ' Percent mass on main gear (%)
acfwb = 4.6                     ' Aircraft wheelbase (meters)
acfws = 15.85                     ' Aircraft wingspan (meters)
tspan = 6.79                    ' Aircraft tail span (meters)
nttip = 11.615                    ' Distance from main gear to tail plane ref. point
nwtip = 4.82                    ' Distance from nose gear to wing tip plane

CASE IS = 16            'Fairchild Metro SA 227

icat = 2
acfmass =  6350                 ' turnoff generating aircraft mass (kilograms)
acflm = 84.73                     ' Percent mass on main gear (%)
acfwb = 5.984                   ' Aircraft wheelbase (meters)
acfws = 17.37                   ' Aircraft wingspan (meters)
tspan = 4.643                   ' Aircraft tail span (meters)
nttip = 14.75                   ' Distance from main gear to tail plane ref. point
nwtip = 6.273                     ' Distance from nose gear to wing tip plane


CASE IS = 17            'Boeing DeHavilland DHC-8

icat = 2
acfmass =  14923                ' turnoff generating aircraft mass (kilograms)
acflm = 84.73                     ' Percent mass on main gear (%)
acfwb = 7.92                    ' Aircraft wheelbase (meters)
acfws = 25.91                     ' Aircraft wingspan (meters)
tspan = 7.92                    ' Aircraft tail span (meters)
nttip = 19.9                    ' Distance from main gear to tail plane ref. point
nwtip = 8.56                    ' Distance from nose gear to wing tip plane

CASE IS = 18            'Douglas DC-8-73

icat = 2
acfmass =  117000&              ' turnoff generating aircraft mass (kilograms)
acflm = 91                      ' Percent mass on main gear (%)
acfwb = 21.59                     ' Aircraft wheelbase (meters)
acfws = 45.2                    ' Aircraft wingspan (meters)
tspan = 13.47                     ' Aircraft tail span (meters)
nttip = 48.79                   ' Distance from main gear to tail plane ref. point
nwtip = 33.24                     ' Distance from nose gear to wing tip plane

CASE IS = 19            'Douglas DC-9-50

icat = 2
acfmass =  49895&               ' turnoff generating aircraft mass (kilograms)
acflm = 90                      ' Percent mass on main gear (%)
acfwb = 18.01                     ' Aircraft wheelbase (meters)
acfws = 28.47                     ' Aircraft wingspan (meters)
tspan = 11.43                   ' Aircraft tail span (meters)
nttip = 36.02                   ' Distance from main gear to tail plane ref. point
nwtip = 21.48                     ' Distance from nose gear to wing tip plane

CASE IS = 20            'Douglas DC-10-30
```

```
icat = 2
acfmass =  182766&                    ' turnoff generating aircraft mass (kilograms)
acflm = 93                            ' Percent mass on main gear (%)
acfwb = 26.17                          ' Aircraft wheelbase (meters)
acfws = 50.4                          ' Aircraft wingspan (meters)
tspan = 22.56                          ' Aircraft tail span (meters)
nttip = 45.57                         ' Distance from main gear to tail plane ref. point
nwtip = 41.51                          ' Distance from nose gear to wing tip plane

CASE IS = 21          'MD-11

icat = 2
acfmass =  195040&                    ' turnoff generating aircraft mass (kilograms)
acflm = 92                            ' Percent mass on main gear (%)
acfwb = 24.8                          ' Aircraft wheelbase (meters)
acfws = 51.7                          ' Aircraft wingspan (meters)
tspan = 16.36                          ' Aircraft tail span (meters)
nttip = 54.63                         ' Distance from main gear to tail plane ref. point
nwtip = 44.63                          ' Distance from nose gear to wing tip plane


CASE IS = 22          'MD-82

icat = 2
acfmass =  58060&                     ' turnoff generating aircraft mass (kilograms)
acflm = 89                             ' Percent mass on main gear (%)
acfwb = 21.79                          ' Aircraft wheelbase (meters)
acfws = 32.87                          ' Aircraft wingspan (meters)
tspan = 11.82                          ' Aircraft tail span (meters)
nttip = 42.47                         ' Distance from main gear to tail plane ref. point
nwtip = 29.55                          ' Distance from nose gear to wing tip plane

CASE IS = 23          'Fokker F-28

icat = 2
acfmass =  31525                      ' turnoff generating aircraft mass (kilograms)
acflm = 88                             ' Percent mass on main gear (%)
acfwb = 10.37                          ' Aircraft wheelbase (meters)
acfws = 25.07                          ' Aircraft wingspan (meters)
tspan = 8.72                          ' Aircraft tail span (meters)
nttip = 25.92                         ' Distance from main gear to tail plane ref. point
nwtip = 12.25                          ' Distance from nose gear to wing tip plane

CASE IS = 24          'Fokker F-100

icat = 2
acfmass =  38330&                     ' turnoff generating aircraft mass (kilograms)
acflm = 88                             ' Percent mass on main gear (%)
acfwb = 14.07                          ' Aircraft wheelbase (meters)
acfws = 28.08                          ' Aircraft wingspan (meters)
tspan = 8.73                          ' Aircraft tail span (meters)
nttip = 31.88                         ' Distance from main gear to tail plane ref. point
nwtip = 16.37                          ' Distance from nose gear to wing tip plane


CASE IS  = 25          'Airbus A320

icat = 2
```

```
acfmass = 61000&                    ' turnoff generating aircraft mass (kilograms)
acflm = 90.77                         ' Percent mass on main gear (%)
acfwb = 12.63                       ' Aircraft wheelbase (meters)
acfws = 33.91                       ' Aircraft wingspan (meters)
tspan = 12.94                       ' Aircraft tail span (meters)
nttip = 34.43                       ' Distance from main gear to tail plane ref. point
nwtip = 22.5                        ' Distance from nose gear to wing tip plane

CASE IS  = 26          'British Aerospace 146-200

icat = 3
acfmass = 36740&                    ' turnoff generating aircraft mass (kilograms)
acflm = 90.26                         ' Percent mass on main gear (%)
acfwb = 11.2                        ' Aircraft wheelbase (meters)
acfws = 26.34                       ' Aircraft wingspan (meters)
acfwt = 4.7                         ' Aircraft wheeltrack (meters)
tspan = 12.06                       ' Aircraft tail span (meters)
nttip = 26.05                       ' Distance from main gear to tail plane ref. point
nwtip = 12.82                       ' Distance from nose gear to wing tip plane

CASE IS = 27           'BAc/CNIAR 1-11

icat = 2
acfmass =  39462&                   ' turnoff generating aircraft mass (kilograms)
acflm = 89                          ' Percent mass on main gear (%)
acfwb = 12.67                       ' Aircraft wheelbase (meters)
acfws = 28.5                        ' Aircraft wingspan (meters)
tspan = 8.89                        ' Aircraft tail span (meters)
nttip = 28.03                       ' Distance from main gear to tail plane ref. point
nwtip =  15.63                      ' Distance from nose gear to wing tip plane


CASE IS  = 28          'Boeing 707-131

icat = 3
acfmass = 121000&                   ' turnoff generating aircraft mass (kilograms)
acflm = 92!                           ' Percent mass on main gear (%)
acfwb = 18.5                        ' Aircraft wheelbase (meters)
acfws = 44.42                       ' Aircraft wingspan (meters)
acfwt = 6.73                        ' Aircraft wheeltrack (meters)
tspan = 13.8                        ' Aircraft tail span (meters)
nttip = 41.61                       ' Distance from main gear to tail plane ref. point
nwtip = 28.51                       ' Distance from nose gear to wing tip plane

CASE IS = 29           'Boeing 737-300

icat = 2
acfmass =  51710&                   ' turnoff generating aircraft mass (kilograms)
acflm = 89                            ' Percent mass on main gear (%)
acfwb =  12.25                        ' Aircraft wheelbase (meters)
acfws = 28.88                       ' Aircraft wingspan (meters)
tspan = 12.25                       ' Aircraft tail span (meters)
nttip = 28.68                       ' Distance from main gear to tail plane ref. point
nwtip = 16.16                         ' Distance from nose gear to wing tip plane


CASE IS  = 30          'Boeing 727-200
```

```
icat = 3
acfmass = 72200&                    ' turnoff generating aircraft mass (kilograms)
acflm = 92.5                          ' Percent mass on main gear (%)
acfwb = 16.75                         ' Aircraft wheelbase (meters)
acfws = 36.75                         ' Aircraft wingspan (meters)
acfwt = 5.71                          ' Aircraft wheeltrack (meters)
tspan = 14.55                         ' Aircraft tail span (meters)
nttip = 42.11                         ' Distance from main gear to tail plane ref. point
nwtip = 25.3                          ' Distance from nose gear to wing tip plane

CASE IS = 31          'Boeing 757-200

icat = 2
acfmass =  89830&                   ' turnoff generating aircraft mass (kilograms)
acflm = 90                            ' Percent mass on main gear (%)
acfwb = 18.08                         ' Aircraft wheelbase (meters)
acfws = 38.05                         ' Aircraft wingspan (meters)
tspan = 14.5                          ' Aircraft tail span (meters)
nttip = 40.44                         ' Distance from main gear to tail plane ref. point
nwtip = 22.51                         ' Distance from nose gear to wing tip plane

CASE IS = 32          'Boeing 767-200

icat = 2
acfmass =  123733&                  ' turnoff generating aircraft mass (kilograms)
acflm = 92                            ' Percent mass on main gear (%)
acfwb = 22.59                         ' Aircraft wheelbase (meters)
acfws = 47.57                         ' Aircraft wingspan (meters)
tspan = 18.16                         ' Aircraft tail span (meters)
nttip = 49.17                         ' Distance from main gear to tail plane ref. point
nwtip = 27.33                         ' Distance from nose gear to wing tip plane


 CASE IS  = 33          'Boeing 747-400

icat = 4
acfmass = 285750&                   ' turnoff generating aircraft mass (kilograms)
acflm = 94!                           ' Percent mass on main gear (%)
acfwb = 25.6                          ' Aircraft wheelbase (meters)
acfws = 63.3                          ' Aircraft wingspan (meters)
acfwt = 11.05                         ' Aircraft wheeltrack (meters)
tspan = 21.08                         ' Aircraft tail span (meters)
nttip = 62.11                         ' Distance from main gear to tail plane ref. point
nwtip = 41.03                         ' Distance from nose gear to wing tip plane

 CASE IS  = 34          'Airbus A300-600

icat = 4
acfmass = 140000&                   ' turnoff generating aircraft mass (kilograms)
acflm = 92.5                          ' Percent mass on main gear (%)
acfwb = 18.6                          ' Aircraft wheelbase (meters)
acfws = 44.8                          ' Aircraft wingspan (meters)
acfwt = 9.61                          ' Aircraft wheeltrack (meters)
tspan = 15.96                         ' Aircraft tail span (meters)
nttip = 46.96                         ' Distance from main gear to tail plane ref. point
nwtip = 32.8                          ' Distance from nose gear to wing tip plane

END SELECT
```

```
' Definition exit parameters for satndard geometries
   IF extype = 6 THEN GOTO 100

   SELECT CASE extype

      CASE IS = 1              '90 Deg. FAA Standard Geometry

            vexit = 8                        ' standard exit speed (m/sec.)
            theta = 90/57.29                 ' standard turnoff angle (deg.)
            Rapath = 77                      ' Largest radius of curvature (m.)

      CASE IS  = 2             '45 Deg. FAA Standard

            vexit = 8                        ' standard exit speed (m/sec.)
            theta = 45/57.29                 ' standard turnoff angle (deg.)
            Rapath = 244                     ' Standard radius for 45 Deg. exit

      CASE IS  = 3             '30 Deg. FAA Standard Geometry

            vexit = 26.7                     ' standard exit speed (m/sec.)
            theta = 30/57.29                 ' standard turnoff angle (deg.)
            Rapath = 550                     ' Radius of curvature (m)

      CASE IS  = 4             '30 Deg. with 1400 ft. spiral

            vexit = 26.7                     ' standard exit speed (m/sec.)
            theta = 30/57.29                 ' standard turnoff angle (deg.)
            Rapath = 550                     ' Radius of curvature (m)

      CASE IS  = 5             'Wide Throat Geometry Estimation Parameters

            vexit = 17                       ' standard exit speed (m/sec.)
            theta = 90/57.29                 ' standard turnoff angle (deg.)
            Rapath = 550                     ' Radius of curvature (m)


      END SELECT

     ' General Algorithm for Standard Geoemtries

      ' Initialize exit parameters

         xpath = 0
         ypath = 0
         vpath = vexit
         sai = 0
         tpath = 0!
         dt = .05
         count = 0!
         saifus = 0!
         tprint = 0!

         xm = xpath - acfwb
         ym = ypath

         xtip = xm
         ytip = ypath - (acfws/2!)
```

```
    LOCATE 4, 5 : PRINT "Time (s)     Xcord(m)     Ycoord. (m.)  Heading   Speed     Heading"
     LOCATE 6,4: PRINT USING"#######.###"; tpath,  xpath, ypath, saifus, vpath, sai


     IF (extype = 4 OR extype = 5) THEN

5      IF (extype = 5) THEN
        dxpath = 1 - .002 *xpath
        ELSE
        dxpath = 1 - .000625 * xpath
        END IF

     xpath1 = xpath + dxpath

     IF (extype = 5) THEN
          ypath1 = 1.3763*10^(.006301*xpath1)
      ELSE
          ypath1 = -.202 + .012 * xpath1 - .0001072 * xpath1^2 + 1.2086E-06 * xpath1^3
      END IF

     dypath = ypath1 - ypath

     ds = SQR(dxpath^2 + dypath^2)
     dt = ds / vpath

     IF (sai < theta*57.29) THEN
        dsai = ATN (dypath / dxpath)
     ELSE
         dsai = 0!
     END IF


      IF (vpath < 8) THEN
          accpath = 0!
       ELSE
          accpath = -.375
       END IF

        ' Integration of nose gear parameters

        xpath = xpath1
        ypath = ypath1
        vpath = vpath + dt*accpath
        sai = sai + dsai
        tpath = tpath + dt
        tprint = tprint + dt

       saifus = (ypath - ym) / (xpath - xm)            ' Fuselage instantaneous heading angle

       xm = xm + dt * vpath * COS (saifus)             ' Main gear centerline trajectory
       ym = ym + dt * vpath * SIN (saifus)


       xwtip = xpath - nwtip * COS (saifus) + (acfws/2) * SIN (saifus)
       ywtip = ypath - nwtip * SIN (saifus) - (acfws/2) * COS (saifus)

       xttip = xpath - nttip * COS (saifus) + (tspan/2) * SIN (saifus)
       yttip = ypath - nttip * SIN (saifus) - (tspan/2) * COS (saifus)
```

```
      said = sai*57.29
      thetad = theta*57.29

    IF tprint > .5 THEN

     PRINT USING"#######.###"; tpath,  xpath, ypath,said, vpath, sai
      tprint = 0!
     END IF

  IF  (ywtip < runwid/2!  OR  yttip < runwid/2!)  GOTO 5

  PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND

  CLS

  ELSE
       GOTO 3
  END IF


3       'Estimation of Rate Variables

        dspath = vpath*dt
        dsai = dspath/Rapath

     ' Estimation of heading rate of change

        IF (sai < theta) THEN
           saidot = dsai/dt
         ELSE
            saidot = 0!
        END IF

     ' Estimation of deceleration rate of change

        IF (vpath < 8) THEN
           accpath = 0!
        ELSE
           accpath = -.375
        END IF

     ' Estimation of downrange position changes

        xdot = dspath*COS(sai)/dt
        ydot = dspath*SIN (sai)/dt

     ' Integration of nose gear parameters

        xpath = xpath + dt*xdot
        ypath = ypath + dt*ydot
        vpath = vpath + dt*accpath
        sai = sai + dt* saidot
        tpath = tpath + dt

        saifus = (ypath - ym) / (xpath - xm)          ' Fuselage instantaneous heading angle

        xm = xm + dt * vpath * COS (saifus)           ' Main gear centerline trajectory
        ym = ym + dt * vpath * SIN (saifus)
```

```
        xwtip = xpath - nwtip * COS (saifus) + (acfws/2) * SIN (saifus)
        ywtip = ypath - nwtip * SIN (saifus) - (acfws/2) * COS (saifus)

        xttip = xpath - nttip * COS (saifus) + (tspan/2) * SIN (saifus)
        yttip = ypath - nttip * SIN (saifus) - (tspan/2) * COS (saifus)

        said = sai*57.29
        thetad = theta*57.29

        count = count + 1

      IF count = 10 THEN

       PRINT USING"########.###"; tpath,  Rapath, xpath, ypath,said, vpath
        count = 0!
       END IF

     IF  (ywtip < runwid/2!  OR  yttip < runwid/2!)  GOTO 3

       'PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND

     CLS

     PRINT "Geometry Parameters"
     PRINT " "
     PRINT " Aircraft Type                          = ", actype
     PRINT " Turnoff Angle                           = ", thetad
     PRINT  " Exit Speed                              =", vexit
     PRINT  " Turnoff Time                           =" , tpath
     PRINT " Aircraft Wingspan                      = ", acfws
     PRINT  " Distance from NG to Wingtip plane     =", nwtip
     PRINT  " Distance from NG to Tailtip plane     =" , nttip
     PRINT  " Final Fuselage Exit Angle              =" , said

       WRITE # 11, acftype, thetad, vexit, tpath, acfws, nwtip, nttip, said

      'PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND
      CLS

NEXT acftype

100    CLOSE # 11

      GOTO 500


100   ' Define constants of the model (only for REDIM computations)

   vars=1                                  '1 - variable mu, 2 - constant mu
   n=1
   g= 9.81                                  ' gravity force (m/sec-sec)
   pi = 3.141592654#                            ' pi
   froll = .03                             ' free roll friction coefficient (dimensionless)
   dt = .05                               ' simulation step size (seconds)
   dx = 5
   dy = 5
   rho = 1.125
```

```
countl=0!
jerkmax = .55                          ' Maximum jerk in m/sec-sec-sec
anmax = 1.2

pcount = 0

   'Estimation of the aircraft cornering capabilities (inertia limitations in the turn)
   ' Estimate the yaw inertia (IZZ) of the aircraft in question

   acfyaw =  10 ^ ( 1.7215 * LOG (acfmass)  / LOG (10) - 1.673 )

IF vars = 1 THEN
   CALL skid(icat, iwet, ymumax, vexit, sf)
ELSE
   CALL fixskid(iwet,ymumax)
END IF


' Initialization of variables (aircraft generating the turnoff geometry)

xpath = 0
ypath = 0
vpath = vexit
saig = 0
castor = 0
tpath = 0
tpath1 = 0
Rapath = 5000

xm = xpath - acfwb
ym = ypath

xtip = xm
ytip = ypath - (acfws/2!)

  LOCATE 4, 5 : PRINT "Time (s)    Radius (m)    Xcord(m)    Ycoord. (m.)    Heading    Speed"
  LOCATE 6,4: PRINT USING"######.###"; tpath,  Rapath, xpath, ypath, saifus, vpath


' Estimation of rate variables (in the turn alone)

rdot1 = acfmass * acfwb
rdot0 = (acflm / 100!) * ( 1 - acflm /100!)

2  IF vpath <= 10! THEN                        ' Clip the value of deceleration to zero
      acpath = 0!                              ' if speed is reduced below taxiing value
   ELSE
     acpath = (-1) * g * froll
   END IF

    'Estimation of scrubbing coefficient and steering algorithm

   CALL steer ( Rapath, vpath, saig, saidotg, ypath, theta)
   CALL scrub (Rapath, acfmass, ymusc)

    ymuc = vpath ^ 2 / (Rapath * g)
    cenacc= ymuc*g
    ymui = ymumax - ymusc - ymuc
```

```
rdot2 = -(ymui * g * Rapath ^ 2 / ( acfyaw * vpath))
rdot = rdot0 * rdot1 * rdot2

' Estimation of the normal acceleration (an) and jerk-related paramaters

 an = vpath^2/Rapath
 rdotjerk = - Rapath^2 * jerkmax / vpath^2
 jerk = an * vpath / lpath
 rdotan = - Rapath^2 * anmax / (lpath * vpath)

 xjerk = xpath
 yjerk = jerkmax * lpath^3/ (6*vpath^3)

 IF ABS(rdot) > ABS(rdotjerk) THEN
    rdot = rdotjerk
 END IF

' Integrate aircraft generating exit parameters

lpath = lpath + dt * vpath
saig = saig + dt * saidotg
Rapath = Rapath + dt * rdot
xpath = xpath + dt * vpath * COS (saig)
ypath = ypath + dt * vpath * SIN (saig)
vpath = vpath + dt * acpath
tpath = tpath + dt
count = count + 1

saifus = (ypath - ym) / (xpath - xm)              ' Fuselage instantaneous heading angle

   xm = xm + dt * vpath * COS (saifus)              ' Main gear centerline trajectory
   ym = ym + dt * vpath * SIN (saifus)

   xwtip = xpath - nwtip * COS (saifus) + (acfws/2) * SIN (saifus)
   ywtip = ypath - nwtip * SIN (saifus) - (acfws/2) * COS (saifus)

   xttip = xpath - nttip * COS (saifus) + (tspan/2) * SIN (saifus)
   yttip = ypath - nttip * SIN (saifus) - (tspan/2) * COS (saifus)

saifusd = saifus*57.29
said = saig*57.29


IF count = 10 THEN

 PRINT USING"#######.###"; tpath,  Rapath, xpath, ypath, saifusd, vpath
  count = 0!
 END IF

IF  (ywtip < runwid/2!  OR  yttip < runwid/2!)  GOTO 2

 'PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND
CLS

PRINT " REDIM Geometry Parameters
PRINT " "
PRINT " Turnoff Angle                              = ", theta
PRINT  " Exit Speed                                 =", vexit
```

```
    PRINT  " Turnoff Time                              =" , tpath
    PRINT " Aircraft Wingspan                          = ", acfws
    PRINT  " Distance from NG to Wingtip plane        =", nwtip
    PRINT  " Distance from NG to Tailtip plane        =" , nttip
    PRINT  " Final Fuselage Exit Angle                 =" , said

     'PRINT "click to continue" : WHILE MOUSE (0) <> 1 :WEND

500    PRINT " Would you like another run - "
       PRINT " "
       PRINT "    1       - no "
       PRINT "    2       - yes "

        INPUT "rerun - ", rerun
       IF (rerun = 2) THEN
          GOTO 200
       END IF


      STOP
      END

' ____________________________ Subroutines ____________________

    SUB steer ( Rapath, vpath, saig, saidotg, ypath, theta) STATIC

    tpol = 2                  ' straight ahead after (theta) heading


      IF tpol = 1 THEN
            IF saig < 1.41 THEN
               IF ypath < (distt/2!) THEN
                  saidotg = vpath / Rapath
               ELSE
                  saidotg = (-1) * vpath / Rapath
               END IF
            ELSE
               saidotg = (-1) * vpath / Rapath
            END IF
       ELSE
             IF saig < (theta/57.29) THEN
                saidotg = vpath / Rapath
             ELSE
                saidotg = 0
             END IF
       END IF

      END SUB

    SUB scrub (Rapath, acfmass, ymusc) STATIC

       IF Rapath < 50! THEN
          ymusc = .015
       ELSE
          IF acfmass > 165000& THEN
             k1 = - 43.8987 + .0002706 * acfmass
             k2 = .1807 - 4.88E-06 * acfmass
```

```
      ELSE
         k1 = 5.1349 - .0000266 * acfmass
         k2 = - 1.8417 + 7.37E-06 * acfmass
      END IF

      ymusc = k1 * Rapath ^ k2

   END IF

END SUB

SUB title (runwid, extype) STATIC

PRINT "
PRINT ' '
INPUT "Runway Width (meters) = ", runwid

PRINT "Enter the Exit Type"
PRINT "
PRINT "          1    -    90 Deg FAA Standard"
PRINT "          2    -    45 Deg FAA Standard"
PRINT "          3    -    30 Deg FAA Standard"
PRINT "          4    -    30 Deg FAA Standard with 1400 ft. Spiral"
PRINT "          5    -    Wide Throat Geometry"
PRINT "          6    -    REDIM Geometry"

 INPUT "Exit Type = ", extype

IF (extype = 6) THEN

   INPUT "Exit Speed (m/sec.) = ", vexit
   INPUT "Safety Factor for =|  (%) = ", sf
   INPUT "Turnoff Angle (degrees) = ", theta

END IF


CLS
END SUB


SUB skid(icat,iwet,ymumax,vexit, sf) STATIC

   IF iwet = 2 THEN

      SELECT CASE icat

      CASE IS = 1
         ymumax = .6919 - .0448*vexit + .0022*vexit^2 - 5.392E-05*vexit^3 + 4.847E-07*vexit^4

      CASE IS = 2
         ymumax = .6313 - .0406*vexit + .0021*vexit^2 - 5.413E-05*vexit^3 + 5.175E-07*vexit^4

      CASE IS = 3
         ymumax = .5725 - .0373*vexit + .0019*vexit^2 - .0000487*vexit^3 + 4.642E-07*vexit^4

      CASE IS = 4
         ymumax = .5153 - .0326*vexit + .0016*vexit^2 - 3.885E-05*vexit^3 + 3.579E-07*vexit^4
```

```
      END SELECT

   ELSE

      SELECT CASE icat

      CASE IS = 1
         k = .8515

      CASE IS = 2
         k = .773

      CASE IS = 3
         k = .6945

      CASE IS = 4
         k = .616

       END SELECT

       mumax = 1.001 - .0574*vexit + .0023*vexit^2 - .0000545*vexit^3 + 6.911E-07*vexit^4 - 3.567E-09*vexit^5
       ymumax = k * mumax

   END IF

ymumax = ymumax / (1 + sf/100 )          ' Sf is the safety factor applied to the ultimate =|


 END SUB


    SUB fixskid(iwet,ymumax) STATIC
      IF iwet = 1 THEN
         ymumax = .35                              ' maximum side nose mu coefficient (dim.)
       ELSE
         ymumax = .2
      END IF

    END SUB

  SUB box STATIC

   LINE (10,10) - (600,380), , B

   END SUB


   SUB small.box STATIC
     LINE (20,20) - (400,180), , B
   END SUB
```

Vita

The author was born in December 16, 1967, in Kurnool, Andhra Pradhesh, India. He received his undergraduate degree in Bachelor of Engineering in Civil Engineering from Osmania University, India in June 1989. He started pursuing his graduate studies at the University of Oklahoma from Fall 1989 and then transferred to Virginia Tech in Spring 1990. He completed M.S., in Civil Engineering, specializing in Transportation Engineering in July 1991. He accepted a job in a consulting firm to pursue his career in Transportation Engineering with plans to do Ph.D., after gaining some experience.