

**THE DESIGN OF GRAPHICAL OUTPUT INTERFACE FOR THE RUNWAY EXIT DESIGN
INTERACTIVE MODEL**

by

Chalapathirao Kishan Sadam

Project Report submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Master of Engineering
in
Civil Engineering

APPROVED:

Dr. Antoine G. Hobeika, Chairman

Dr. Donald R. Drew

Dr. Antonio A. Trani

January 31, 1990
Blacksburg, Virginia

THE DESIGN OF GRAPHICAL OUTPUT INTERFACE FOR THE RUNWAY EXIT DESIGN

INTERACTIVE MODEL

by

Chalapathirao Kishan Sadam

Dr. Antoine G. Hobeika, Chairman

Civil Engineering

(ABSTRACT)

The acceptance rate of a runway at an airport can be considerably increased by proper location of high-speed exit taxiways along the runway downrange. The location, geometry and the exit angle of these high-speed exit taxiways play a crucial role in the successful negotiation of the turnoff and in reducing the runway occupancy times. This reduction in ROT increases the acceptance rate of a runway and thereby increasing the capacity of the airport. Research was done to approximate the radius of curvature with the aid of a compound curve rather than a single curve for the exit taxiway.

In the past studies, researchers did not lay much emphasis on the presentation of the results of the model developed. This project has been envisioned to present the output provided by the "Runway Exit Design Interactive Model" to the analyst in such a manner that improvement, analysis and the design of a new runway can be done with the aid of interactive screens provided in the output phase. The analyst may perform sensitivity analysis, compare the centerline turnoff geometries and view the change in runway occupancy times for each aircraft, for different turnoffs and for different runway surface conditions.

A general description of the model, the output interface and the exit taxiway geometry built with a compound curve is also included.

Acknowledgements

I would like to thank Dr. Antoine G. Hobeika, my major advisor, for his time, patience and effort devoted to the conception, development and completion of this research project. I would like to acknowledge Dr. Donald R. Drew, for his friendship and guidance throughout the completion of my degree program in Civil Engineering. In addition, I would like to express my gratitude to Dr. Antonio A. Trani for his guidance in the research project and especially with the writing of the project report.

I would like to extend my special thanks to my colleagues and
for helping me with some computer techniques.

I would like to express my gratitude to my parents for giving me an opportunity to study in a foreign country, for all their moral and financial support through all these years away from home. Thanks to all the employees of the University Center for Transportation Research, Virginia Tech for their friendship and cooperation during my stay there.

Table of Contents

1.0 INTRODUCTION	1
1.1 Air Transport in The United States	4
1.2 Airport Capacity and Delay	5
1.3 Research Objective and Scope	14
1.4 Model Overview	17
2.0 AIRPORT CONFIGURATION	18
2.1 Runways	18
2.2 Taxiways	19
2.3 Airport Classification System	21
2.4 Runway Configurations	24
2.4.1 Single Runways	24
2.4.2 Parallel Runways	24
2.4.2.1 Dual-Lane Runway	25
2.4.3 Intersecting Runways	25
2.4.4 Open-V Runways	27
2.5 Taxiway System	27
2.5.1 Requirements of the Taxiway System	29
2.5.2 Dimensional Design Criteria	30
2.5.3 Parallel Taxiway	33
2.5.4 Taxiway Intersections	33

2.5.5	Entrance Taxiways	33
2.5.6	Bypass Taxiways	35
2.5.7	Holding Bays	35
2.5.8	Dual Parallel Taxiways	35
2.5.9	Exit Taxiways	36
2.5.9.1	Configuration	38
2.5.10	Exits for Small Airplanes	41
2.5.11	Apron Taxiways and Terminal Taxilanes	41
2.5.12	Taxiway Safety Area and Shoulders	43
3.0	MODEL DESCRIPTION	44
3.1	Input Module	44
3.1.1	Data Input Method	45
3.1.2	Procedures in Input Module	46
3.1.2.1	Start a New Problem	46
3.1.2.2	Edit Data	47
3.1.2.3	Edit Master File	47
3.2	Dynamic Module	47
3.2.1	Air Distance	48
3.2.2	Free Roll Distances	48
3.2.3	Braking Distance	49
3.2.4	Turnoff Algorithm	50
3.2.5	Exit Assignment Algorithm	50
3.2.6	Airport Environmental Variables	50
3.2.7	Aircraft Characteristics	51
3.3	Optimization Module	52
3.4	Output Module	53

4.0 OUTPUT STRUCTURE	55
4.1 View the Output	55
4.1.1 Evaluate an Existing Runway	57
4.1.1.1 ROT / Reliability Table of Results	57
4.1.1.2 Turnoff Locations and their Geometries	60
4.1.1.3 Go to Output Menu	61
4.1.2 Improve an Existing Runway	61
4.1.2.1 ROT Table of Results	66
4.1.2.2 Turnoff Locations and their Geometries	67
4.1.2.3 Aircraft Statistics and their ROT's	71
4.1.2.4 Go to Output Menu	72
4.1.3 Design of a New Runway	72
4.1.3.1 ROT Table of Results	74
4.1.3.2 Turnoff Locations and their Geometries	77
4.1.3.3 Aircraft Statistics and their ROT's	78
4.1.3.4 Go to Output Menu	79
4.2 Print the Report	79
4.3 Help	80
 5.0 RESULTS AND FURTHER RESEARCH	 81
 Bibliography	 89
 Appendix A. PROGRAM CODE	 91
 Vita	 159

List of Figures

Figure 1.1.	Distribution of passenger trip lengths on scheduled domestic airline flights in the United States, Nov. 1980.	6
Figure 1.2.	Distribution of scheduled domestic airline flights in the United States, Nov, 1980.	7
Figure 1.3.	Hypothetical Aircraft Landing Distance Distribution.	12
Figure 1.4.	Typical Projected Turnoff Locations for a Hypothetical Runway.	13
Figure 1.5.	Sample Turnoff Geometries for Different Aircraft Categories.....	15
Figure 1.6.	Aircraft Landing Events Modeled in the Analysis.	16
Figure 2.1.	Plan View of Runway.....	20
Figure 2.2.	Typical Runway Configurations.	26
Figure 2.3.	Taxiway and Taxilane sections.	28
Figure 2.4.	Common Taxiway Intersections.	34
Figure 2.5.	30-Degree High Speed Exit Taxiway.	39
Figure 2.6.	90-Degree Exit Taxiway.	40
Figure 2.7.	45-Degree Exit Taxiway.	42
Figure 4.1.	Output Menu.....	56
Figure 4.2.	Flow Pattern for Runmode '1'.....	160
Figure 4.3.	View Menu for Runmode '1'.	58
Figure 4.4.	ROT / Reliability Table of Results.....	59
Figure 4.5.	Turnoff Locations along the Runway Downrange.....	62
Figure 4.6.	Standard FAA 30-Degree Turnoff.	63

Figure 4.7.	Standard FAA 45-Degree Turnoff.	64
Figure 4.8.	Standard FAA 90-Degree Turnoff.	65
Figure 4.9.	Flow Pattern for Runmode '2'.....	160
Figure 4.10.	View Menu for Runmode '2' and '3'.....	68
Figure 4.11.	ROT Table of Results.....	69
Figure 4.12.	Comparision of Centerline Geometries.....	70
Figure 4.13.	Flow Pattern for Runmode '3'.....	160
Figure 4.14.	Table of Turnoff Geometry Coordinates.	73
Figure 4.15.	Variable Turnoff Geometry.....	75
Figure 4.16.	Aircraft Statistics and their ROT's.	76
Figure 5.1.	Radius of Curvature and Entrance Curves for Taxiways.....	83
Figure 5.2.	Comparison of Radius of Curvatures.....	84
Figure 5.3	Turnoff Time Vs. Turnoff Angle Sensitivity (Boeing 747-400 Aircraft).....	86
Figure 5.4	Turnoff Time Vs. Turnoff Angle Sensitivity (Boeing 727-200 Aircraft).....	87

List of Tables

Table 1.1.	Domestic Intercity Passenger-Miles by Mode of Travel and Class of service, 1959-1980, Millions.	2
Table 1.2.	Household Trips in The United States, 1977.	3
Table 1.3.	Arrival Runway Occupancy Times.	9
Table 1.4.	Arrival in-trail separation Criteria.	10
Table 2.1.	FAA Aircraft Approach Category Classification.	22
Table 2.2.	FAA Airplane Design Group Classification for Geometric Design for Airports	23
Table 2.3.	Runway Dimensional Standards, FAA.	31
Table 2.4.	Taxiway Dimensional Standards, FAA.	32
Table 2.5.	Taxiway Exit Geometry Recommended by FAA, ft.	37

1.0 INTRODUCTION

Today, the nation is facing a rising crisis in aviation. In the last decade alone a boom in airline passenger traffic has severely strained the airport and airways system. In fact in 1987, the world airlines have crossed the one-billion passenger mark [Aviation Week and Space Technology, 1989]. To a certain extent this upsurge has been stimulated by reduced fares, wider varieties of routes and types of service, and special incentives offered by airlines since deregulation. But there are also underlying socio-economic reasons--economic growth, shifting demographic patterns, changing life-styles, and increasing reliance on air as the mode of travel for trips of more than 200 miles [Transportation Research Board, 1988].

But the growth in demand has given rise to congestion and delay throughout the nation's air travel network, principally at the major metropolitan airports that serve as connecting points, or hubs, for converging traffic routes. The Federal Aviation Administration (FAA) has estimated that 11 major airports now experience severe chronic delay in operations as a result of traffic congestion and that the number could reach 29 by 1996 and perhaps 47 by early in the next century [FAA, 1987]. Delays at these key airports are not simply local problems; the effects ripple outward to other airports with flights connecting to these hubs and ultimately to the entire network [Transportation Research Board, 1988].

Table 1.1 Domestic Intercity Passenger-Miles by Mode of Travel and Class of Service, 1959-1980, Millions

	1950	1960	1970	1975	1980
Total	458,832	748,121	1,161,335	1,331,093	1,496,787
Total common carriers	55,989	67,521	135,335	167,093	233,387
Airlines	7,954	30,557	104,146	131,728	200,087
Railroads	26,781	17,064	6,179	9,765	6,400
Motor buses	21,254	19,900	25,300	25,600	26,900
Air share, %	14.2	45.3	76.8	78.8	85.7
Private automobile	402,843	680,600	1,026,000	1,164,000	1,263,400
Common-carrier share of total, %	12.2	9.0	11.7	12.6	15.6
Air share of total, %	1.7	4.1	9.0	9.9	13.4

SOURCE: Air Transportation Association of America.

Table 1.2 Household Trips in the United States, 1977

Round trip length, mi	Percent of trips	Percent by travel mode				
		Auto	Bus	Train	Air	Other
200-299	34	93	4	1	1	1
300-399	17	90	4	1	3	1
400-599	17	82	5	1	9	3
600-799	8	70	4	1	21	4
800-999	5	62	4	1	29	4
1000-1999	10	48	3	1	42	6
2000 and over	9	25	2	1	67	5
Total	100	76	4	1	16	3

SOURCE: Department of Commerce.

1.1 Air Transport in The United States

The growth of economy of the United States has been closely associated with the development of transportation. The amount of money spent for transportation services by all modes has correlated closely with the level of economic activity as measured by the Gross National Product (GNP). The domestic inter-city travel expenditures has averaged 5 to 6 percent of the GNP. In fact, the civil aviation contributions to the GNP have consistently grown at a faster pace than the economy as a whole [Horonjeff et al, 1983].

Inter-city travel can be considered to be of two parts: private automobile travel, which accounts for nearly 85 percent of the total, and common carrier travel (bus, rail and air), which accounts for the remaining portion. A phenomenal growth in air travel has been observed during the past few decades as can be seen from Table 1.1.

Generally, it is perceived by many that air transportation is the predominant mode for long-distance trips. But it is interesting to note that in 1980 more than one-half of all trips by air were not more than 500 mi, and almost 87 percent did not exceed 1000 mi. This is illustrated in Fig.1.1. Further as observed from Fig. 1.2, it is quite evident that 67 percent of the flights scheduled in the United States were not more than 500 mi, and 90 percent were not more than 100 mi [Horonjeff et al, 1983].

Even though automobile still remains the principal means of short-haul travel, it is beneficial to compare air transport with the other competing modes of inter-city travel. From Table 1.2 it is evident that the majority of round trips in excess of 400 mi are made by air. But a substantial amount of air travel for one way trips in excess of 100 mi is predicted, provided efficient and economical short-haul aircraft and adequate facilities can be developed. This potential, together with the severe airspace and airfield congestion occurring at terminals in large metropolitan areas, is stimulating the development of new smaller-capacity aircraft and of satellite airports to service the short-haul markets [Horonjeff et al, 1983]. In fact, as the

commuters are growing in size, the average transport capacity has increased by 15 seats in the past decade.

1.2 Airport Capacity and Delay

Airport congestion and delays has attracted considerable attention in recent years due to the rapid growth of air transportation services. Current statistics indicate that approximately two billion dollars are paid by air travelers due to system imposed delays in the United States [FAA, 1987]. These delays are likely to increase as air travel demand builds up from 416 billion passenger revenue mile flown in 1988 to an estimated 750 billion passenger revenue mile by the end of the century or equivalent to an average annual growth of roughly 6 percent [Aviation Week & Space Technology, 1989]. The problem can be better visualized when one considers that the current air transportation system has been operating under the same airport infrastructure since the early seventies. A better perspective about the magnitude of the current delay problem can be obtained by observing the predictions given by the Federal Aviation Administration for the year 1996 when thirty two airport facilities in the United States will have more than 20,000 hours of annual system imposed delays. In fact, eleven of these will exceed the 50,000 hour mark [United States Department of Transportation, 1988].

In an airport, the following airport components play a vital role in defining the capacity of the airport: (1) the airspace, (2) the runway, (3) the taxiway and (4) the apron-gate. But the capacity of the airport is restricted by the characteristics of the least efficient component. For certain airport configurations, the interrelationship among these four components is significant, but it is certainly beneficial to study each component independently and then select the most restrictive one as that defining the capacity of the facility. In addition to the above mentioned components, airport capacity is also affected by many external factors viz.,

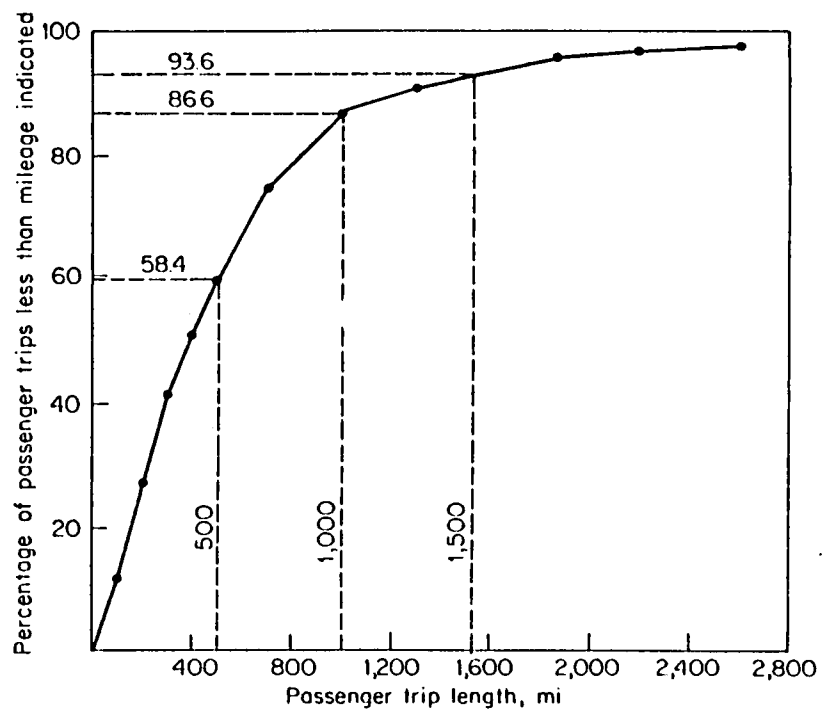


Figure 1.1. Distribution of passenger trip lengths on scheduled domestic airline flights in the United States, November 1980.
Source: Civil Aeronautics Board.

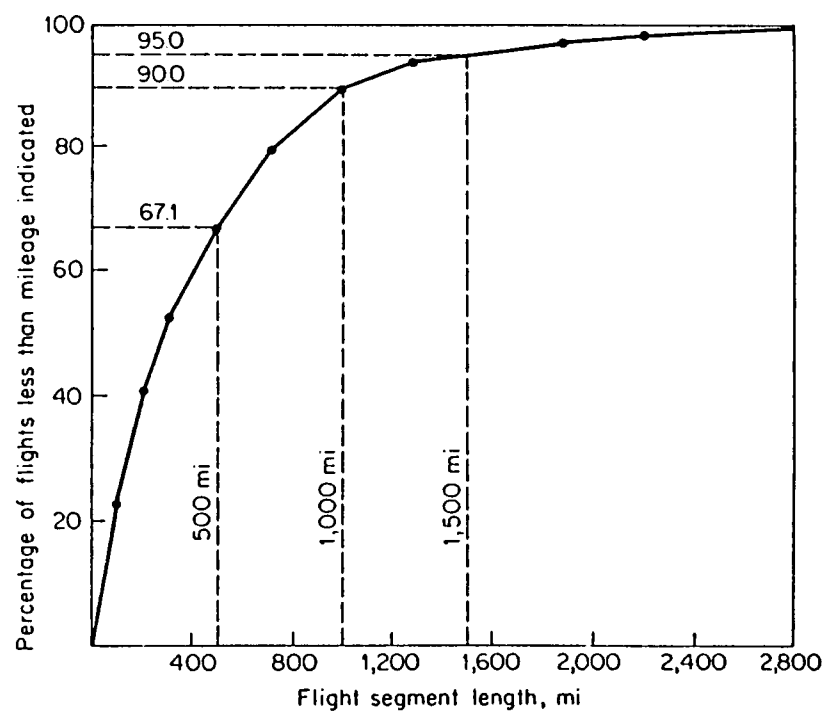


Figure 1.2 Distribution of scheduled domestic airline flights in the United States, November 1980.
Source: Civil Aeronautics Board.

meteorological conditions, airfield configurations, aircraft characteristics and also on the performance of air traffic control system.

Today, system developers are working to increase the airport capacity without compromising the systems safety. Research and development programs at FAA and NASA are addressing several issues of the airfield problems, foremost of which are the improvement of operational use of runways, provision of efficient flow control, spacing and management of aircraft in the terminal airspace, upgrading of the computer / communication technology usage, and the resolution of the effects of wake vortex and aircraft noise.

Improving the operational use of runways, through a reduction of runway occupancy times using high-speed exits is one of the research activities being investigated by the FAA and NASA. The efficiency of the runway component is dictated primarily by the runway occupancy time (ROT) and its variability from aircraft to aircraft. ROT is the time span that an aircraft spends on the runway or its vicinity until a new arrival or departure can be processed. To be precise, ROT is the time span from threshold to the clearance of the runway. Table 1.3 illustrates typical values of ROT and its variability for several aircraft classes using current and future technologies according to a recent study conducted at the MITRE Corporation [Barrer and Dielh, 1988]. The study quantified the potential increases to runway capacity resulting from improvements to the Air Traffic Control System (ATC) performance parameters (i.e., reducing in-trail landing separations, better planned runway exits, improved ground-based radar surveillance capability, etc.) and concluded that gains of up to 20% in the capacity of a single runway are possible if these control actions were to be implemented. Other studies support similar gains if advanced systems are used [Lebron, 1987; Simpson et al, 1988]. On the airspace component a critical parameter directly related to capacity is the arrival in-trail separation (AIT). AIT values are dictated by safety criteria to avoid the effects of the wake vortex generated by a leading aircraft during landing. Current and future arrival in-trail separation values taken from the same study are shown in Table 1.4. Values of equivalent arrival in-trail separation have been added to show that even with the projected fixed distance separation standards, large in-trail separation times remain for some

Table 1.3 Arrival Runway Occupancy Times

TERP Category	ROT Mean Value (seconds)	
	Present	Future (1996)
A	43	35
B	45	37
C	46	40
D	50	45

SOURCE: Barrer & Dielh, 1988.

Table 1.4 Arrival in-trail separation Criteria.

Lead Aircraft	Current Values (nautical miles) (seconds)			Future Goal (1996) (nautical miles) (seconds)		
	Small	Large	Heavy	Small	Large	Heavy
Small	2.5	2.5	2.5	2.0	2.0	2.0
	82	64	56	65	52	45
Large	4.0	2.5	2.5	3.0	2.0	2.0
	131	64	56	98	52	45
Heavy	6.0	5.0	4.0	5.0	4.0	3.0
	196	129	90	163	103	68

Assumed speeds for three aircraft classes:

Small - 110 knots
Large - 145 knots
Heavy - 160 knots

SOURCE: Barrer and Deilh, 1988.

operational scenarios (i.e., a small aircraft operating behind a heavy jet). Moreover, one must recognize that challenges remain to be solved in terms of the technology available to support some to these control actions. For example, the wake vortex separation criteria envisioned in the MITRE study would necessitate of a completely revised scheme in how aircraft control systems are designed or through sophisticated approach sequences which might not only include curved paths but also real-time advisories of the progress of preceding landing aircraft. The first point is probably the most debatable since the vortex wake generated by a heavy aircraft requires a significant amount of time to decay to harmless levels and the reduction of the current separation criteria could result in substantial increases in vortex wake penetration [Rossow and Tinlin, 1988]. The second approach seems more promising with the introduction of the microwave landing system (MLS) coupled with area navigation procedures (RNAV) and its well publicized multipath approach capabilities although more flight testing is necessary to validate potential MSL-RNAV procedures for heavy aircraft operations in the "near" terminal control area [Branstetter et al, 1988].

Research is being done by various organizations to address the air space and runway capacity issues in order to improve the level of service of existing and future facilities. However, it is imperative to emphasize that a reduction in the in-trail separation rules will have to be followed by a corresponding reduction in ROT times if significant improvements in the airport capacity are to be achieved. The use of highly advanced avionics coupled with the installation of modern navigation facilities (i.e., microwave landing system, four-dimensional navigation, etc.) and the strategic location of turnoffs seem to promise some reductions in the in-trail separation, ROT and their corresponding variabilities to warrant changes in runway capacity.

In order to improve the airport capacity, several alternatives were suggested. The one that is being addressed in this project is the use of high-speed turnoffs in airport runways to reduce the aircraft time spent on the runway. Although certain major airports have adopted the 30-degree high-speed FAA turnoff standard and newly proposed spiral design [Greiner, 1988], it should be clear that there is room to improve the level of service of a runway facility,

SEARCHING FOR OPTIMAL TURNOFF LOCATIONS AND GEOMETRIES

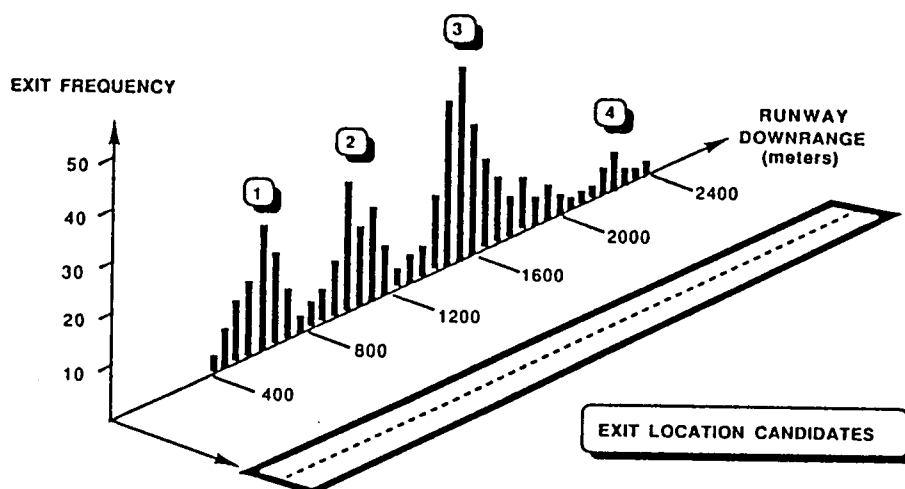
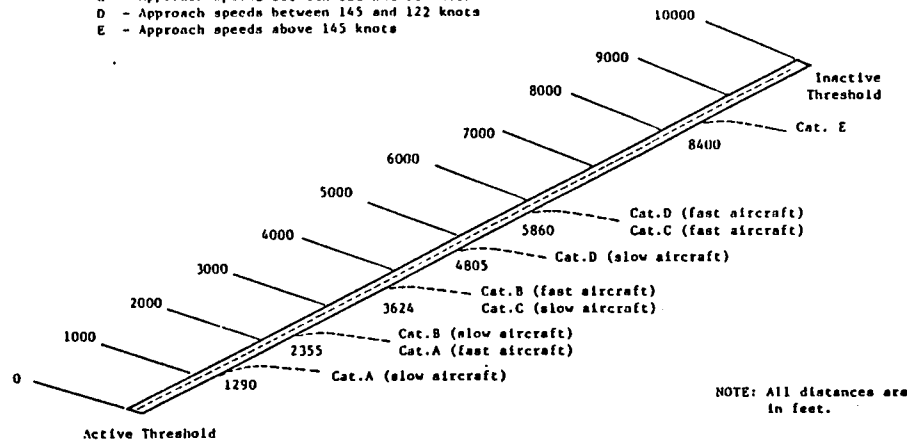


Figure 1.3 Hypothetical Aircraft Landing Distance Distribution.

TERMINAL OPERATIONS AIRCRAFT CATEGORIES (TERPS)

- A - Aircraft with approach speeds below 61 knots
- B - Approach speeds between 91 and 62 knots
- C - Approach speeds between 121 and 92 knots
- D - Approach speeds between 145 and 122 knots
- E - Approach speeds above 145 knots



TYPICAL OUTPUT OF HIGH SPEED TURNOFFS FOR VARIOUS AIRCRAFT TYPES

Figure 1.4 Typical Projected Turnoff Locations for a Hypothetical Runway.

i.e., decreasing the runway occupancy time or increasing the runway acceptance rate, by tailoring the turnoff geometry and location to an existing or forecasted population of aircraft. That is, the population of aircraft operating in a particular airfield should dictate not only the location but also the characteristic turnoff geometries associated with that runway.

1.3 Research Objective and Scope

The goal of this project is to develop a user friendly graphical output interface for the computer simulation model which estimates the optimal location of high-speed turnoffs at an airport facility. These optimally located high-speed turnoffs reduce the runway occupancy time (ROT) which in turn increases the capacity of the runway. The model incorporates environmental factors, like the airfield elevation, runway configuration, weather conditions etc., and operational factors such as aircraft mix (i.e., terminal operation aircraft types) and aircraft piloting technique (i.e., aircraft touchdown speed dispersion values) to determine the potential location of high speed turnoffs. In the past studies, the models that were developed did not consider the above mentioned factors. Also, none of the models developed earlier gave the user any flexibility in choosing the Output needed from those models.

Research performed by various groups has revealed that images and graphic are expecially helpful in problem structuring and in presenting correctly the results desired by the analyst. It has been proven that diagrams and graphs are useful in representing problems, serving as external memory aids, revealing inconsistencies in decision-maker knowledge, and in leading to a better understanding of novel problems and hence to improved task performance and greater decision quality [Pracht and Courtney, 1988]. The project "Design of Graphical Output Interface for the Runway Exit Design Interactive Model" is envisioned to develop a graphical interface, which is user friendly and at the same time allowing the analyst to perform sensitivity analysis.

HIGH SPEED TURNOFFS FOR AIRCRAFT CATEGORIES

Note: All turnoff geometries starting at the origin for comparison purposes.

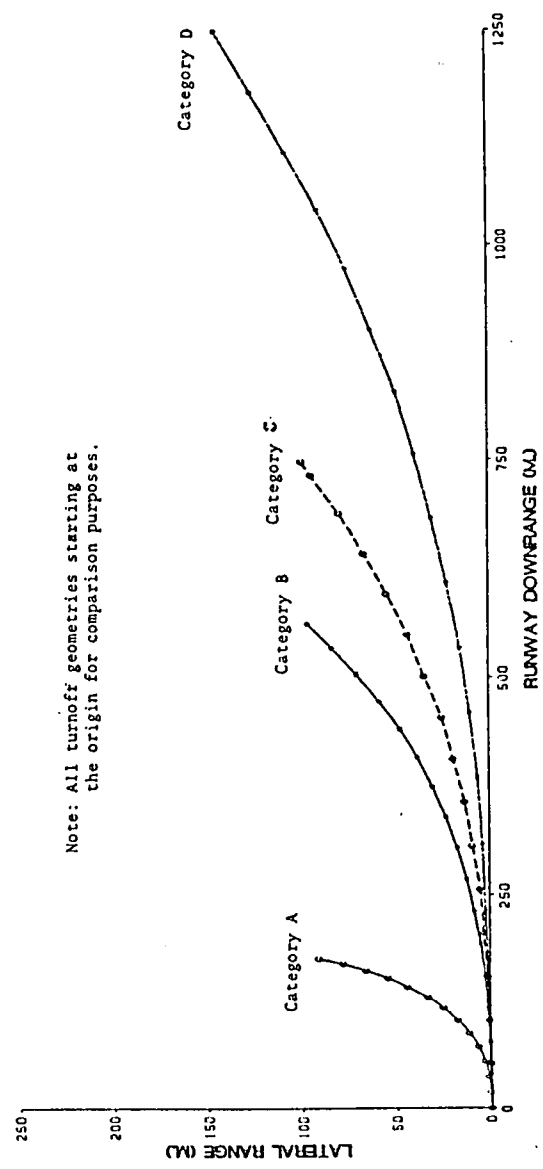
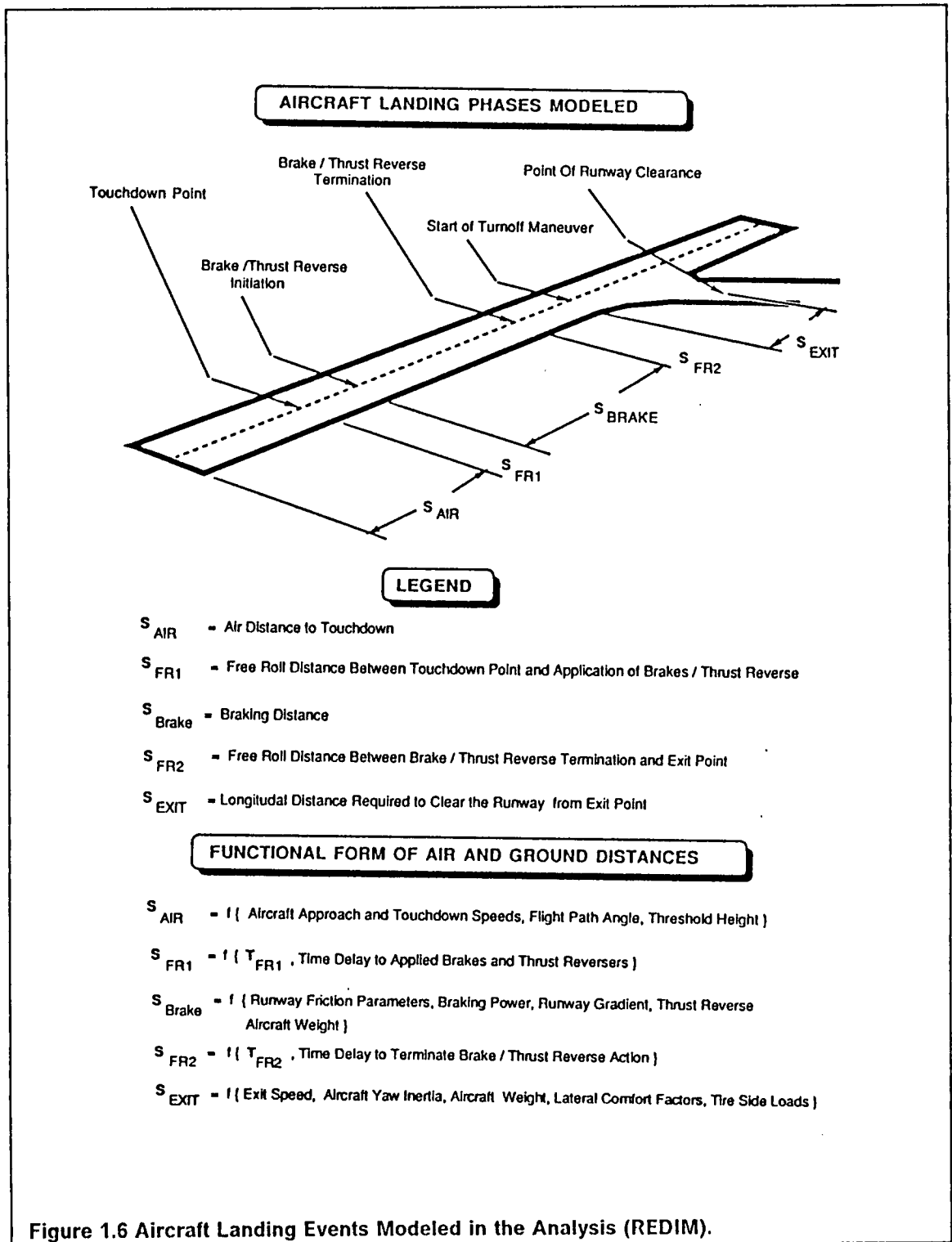


Figure 1.5. Sample Turnoff Geometries for Different Aircraft Categories.



1.4 Model Overview

The project, "The Design of Graphical Output Interface for the Runway Exit Design Interactive Model", forms the output module of the major project "Runway Exit Design Interactive Model", REDIM. The model incorporates several specific airfield variables that affect the landing performance of the aircraft as well as other important operational constraints (e.g. aircraft mix) that have a direct impact on the selection of turnoff location and their geometry. The model comprises of five modules: (1) an input module, (2) a dynamic simulation module to estimate the ROT times for individual aircraft, (3) a optimization module to determine candidate turnoff locations to comply with a desired reliability threshold value, (4) an aircraft data module containing relevant aircraft performance and geometric parameters and (5) an output module to show graphically the suggested runway turnoff configuration and display some measures of effectiveness desired by the analyst.

Through its interactive screens, the model displays the results and allows the analyst to perform sensitivity analysis. The analyst may also view the existing turnoff geometries and compare them with the high-speed turnoff geometry generated by the model. In fact, the analyst may compare the centerline geometries of each of the turnoffs generated through successive overlays of each turnoff on a common plot. Thus helping to better understand the geometrical differences among the proposed turnoffs.

2.0 AIRPORT CONFIGURATION

Norman Ashford and Paul H.Wright [Ashford and Wright, 1979] have defined airport configuration as the general arrangement of the various parts or components of the airport system. Horonjeff et al have defined airport configuration as the number and orientation of runways and the location of the terminal area relative to the runways. The number of runways depends on the volume of traffic, and the orientation depends on the direction of the wind, the size of the area available for airport development, obstructions in the vicinity of the airport, population centers, etc.

2.1 *Runways*

The runway system at an airport consists of the structural pavement, the shoulders, the blast pad, and the runway safety area. The structural pavement supports the aircraft with respect to structural load, and provides enough space for maneuvering safely during ground operations. The shoulder adjacent to the end of the structural pavement resists jet blast erosion and accommodates maintenance and emergency equipment. The blast pad is an area designed to prevent erosion of the surfaces adjacent to the ends of runways which are subjected to sustained or repeated jet blast. The runway safety area is an area which is cleared, drained, and graded, and which includes the structural pavement, shoulders, blast pad, and stopway, if provided. This area should be capable of supporting emergency and

maintenance equipment as well as providing support for aircraft should they veer off the pavement for one reason or another [Horonjeff et al, 1983]. The plan view of runway elements is shown in Fig. 2.1.

The runways and the connecting taxiways are to be so arranged that they

- provide adequate separation in the air traffic pattern
- cause the least interference and delay in the landing, taxiing, and takeoff operations
- provide adequate taxiways so landing aircraft can leave the runways as quickly as possible and follow routes as short as possible to the terminal area.

2.2 Taxiways

The taxiways should be arranged to provide access from the runways to the terminal area and service hangars without interfering with aircraft taxiing to takeoff. Parallel one-way taxiways should be provided where traffic is expected to move simultaneously in both directions. Taxiways should be located along runways, such that landing aircraft can leave the runway as quickly as possible to clear them for use by other aircraft. These taxiways are commonly referred to as exit taxiways or turnoffs.

In Chapter 1, it was mentioned that an airport comprises of four components: (1) the airspace, (2) the runway, (3) the taxiway and (4) the apron gate. It was also stressed that the capacity of the airport is restricted by the characteristics of the least efficient component. During peak traffic periods when a continuous supply of aircraft is available, the capacity of a runway is dependent to a large degree on how quickly landing aircraft can be vacated from the runway. This will become more critical as approach separation minima are reduced in the

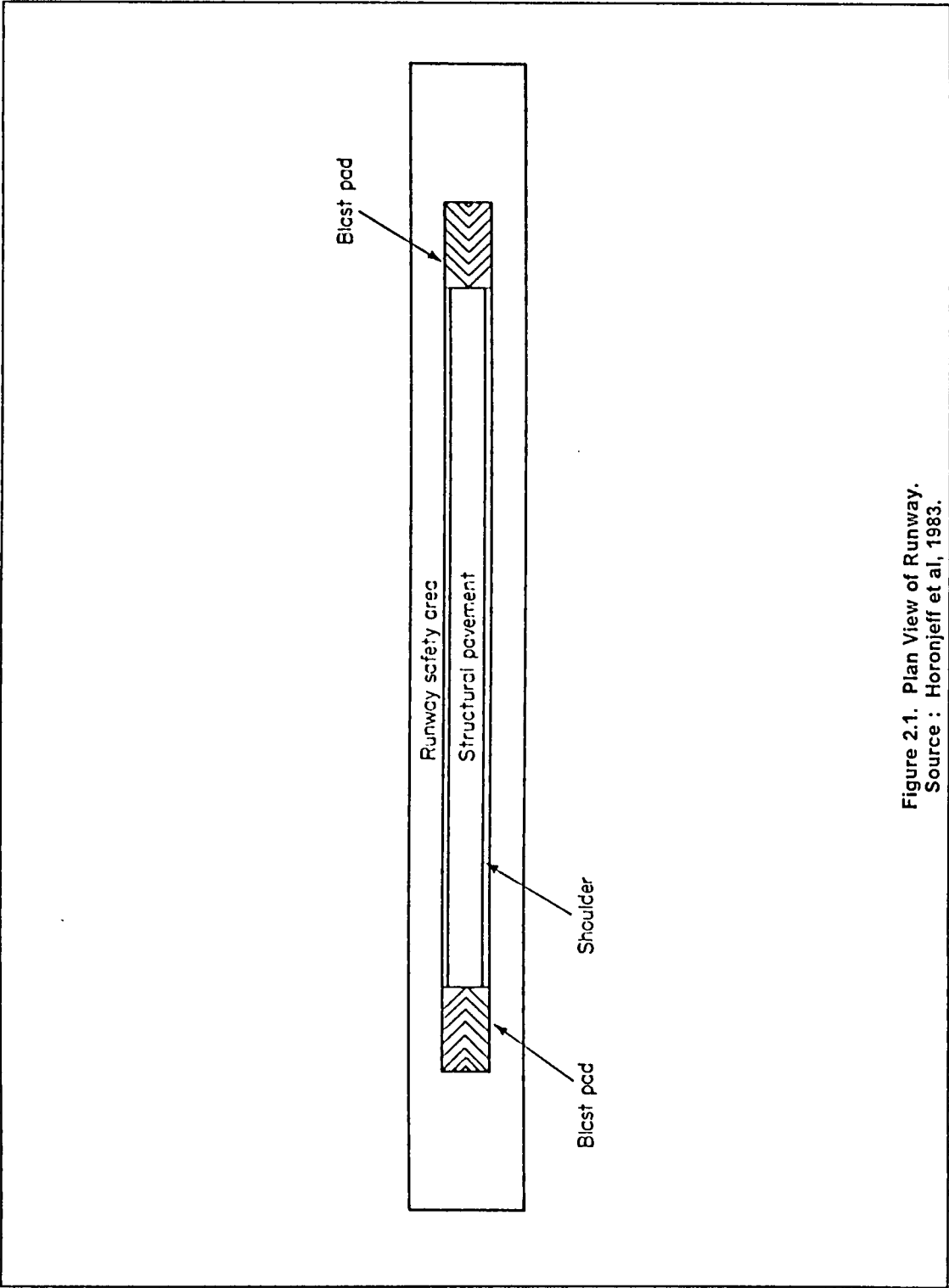


Figure 2.1. Plan View of Runway.
Source : Horonjeff et al, 1983.

future airport environment (i.e., 2.5 / 4.0 / 5.0 miles). An aircraft which has landed delays succeeding aircraft until it has cleared the runway; the runway component. A taxiway designed to permit higher turnoff speeds reduces the length of time a landing aircraft occupies the runway; the taxiway component. On the other hand, if the taxiways are at right angles to the runways, the aircraft must decelerate to very low speeds before they can negotiate a turn resulting in higher runway occupancy times (ROT). Right angle or 90 degree exit taxiways, although not desirable from the standpoint of minimizing runway occupancy, are often constructed for cost and simplicity reasons [Horonjeff et al, 1983].

When the exit taxiways are optimally located with appropriate turnoff geometries, thus reducing the runway occupancy time, then the airspace component will become the critical component. The capacity of an airport will then be dictated by the airspace component, assuming that the apron-gate component will not be critical.

2.3 Airport Classification System

The FAA has classified airports for geometric design purposes and is based upon the approach category of aircraft. The approach category is determined by the aircraft approach speed, which is defined as 1.3 times the stall speed in the landing configuration of that aircraft at maximum allowable landing weight. The FAA aircraft approach category classification is shown in Table 2.1. Aircraft with maximum certified takeoff weight in excess of 300,000 lb are classified as heavy aircraft, between 300,000 lb and 12,500 lb as large aircraft, and the rest are small aircraft. The FAA has also classified aircraft into six groups based upon wingspan, for airport geometric design standards as shown in Table 2.2.

**Table 2.1 FAA Aircraft Approach
Category Classification.**

Approach Category	Approach speed, kn
A	Less than 91
B	91 - 120
C	121 - 140
D	141 - 165
E	166 or greater

SOURCE: Federal Aviation Administration, 1976.

**Table 2.2 FAA Airplane Design Group Classification
for Geometric Design for Airports**

Airplane Design Group	Wingspan, ft	Typical aircraft
I	Less than 49	Learjet 24, Rockwell Sabre 75A
II	49 but less than 79	Gulfstream II, Rockwell Sabre 80
III	79 but less than 118	B-727, B-737, BAC1-11, Concorde, DC-9
IV	118 but less than 171	A-300, A-310, B-707, DC-8 B-757, B-767, L-1011
V	171 but less than 197	B-747
VI	197 but less than 262	Future

Source: Federal Aviation Administration, 1982.

2.4 Runway Configurations

There are four main configurations: (1) single runways, (2) parallel runways, (3) intersecting runways, and (4) open-V runways. Most runway systems are arranged according to some combination of these four basic configurations. The runway dimensional standards are shown in Table 2.3.

2.4.1 Single Runways

This is a very simple runway configuration and is shown in Fig. 2.2.(a). The hourly capacity of this runway in VFR (Visual Flight Rules) conditions is between 50 and 100 operations per hour, while in IFR (Instrument Flight Rules) conditions, the hourly capacity is between 50 and 70 operations. These depend on the composition of the aircraft mix and the navigational aids available.

2.4.2 Parallel Runways

Many parallel runway systems exist. Two and four parallel runways are common. The capacities of parallel runway systems depend a great deal on the number of runways and on the spacing between the runways. The spacing between the parallel runways may be classified as close, from a minimum 700 ft to less than 2500 ft; intermediate, from 2500 ft to less than 4300 ft; and far, for 4300 ft or more. Sometimes, it may be desirable to stagger the thresholds of parallel runways. The staggering may be necessary because of the shape of the land available for runway construction, or it may be desirable for reducing the taxiing distance of takeoff and landing aircraft, or even to reduce the distance between runways and still be

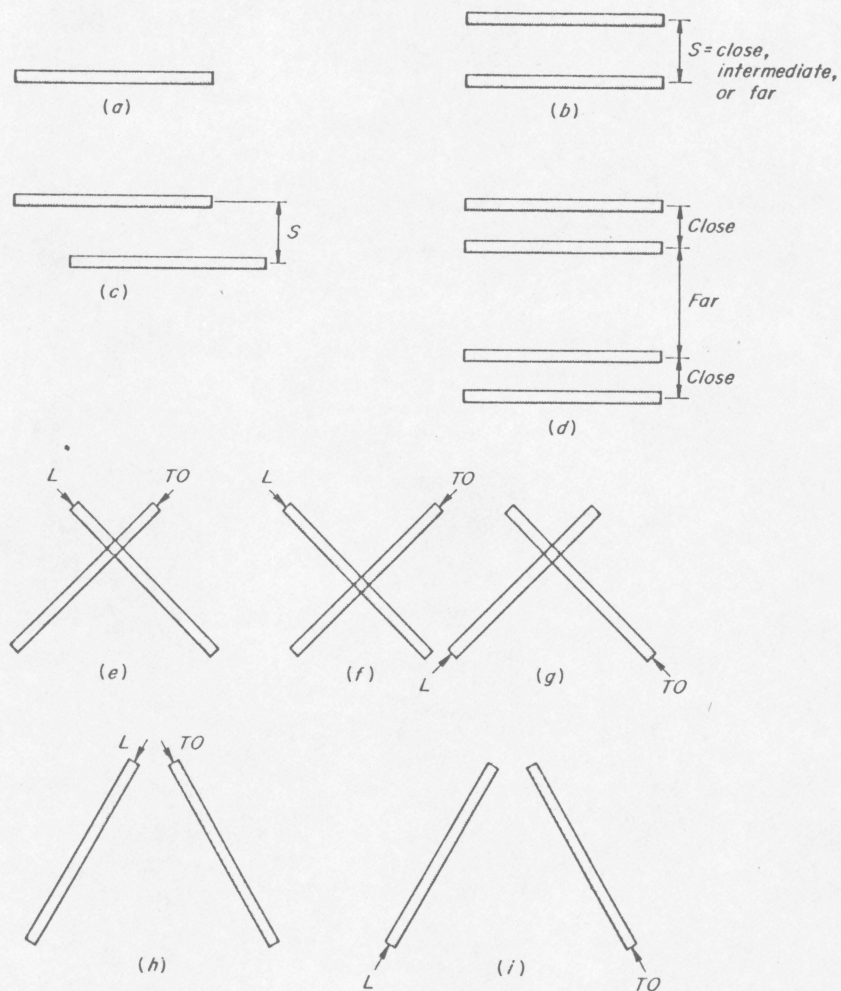
able to conduct independent parallel IFR approaches. Parallel runways are shown in Figs. 2.2(b), 2.2(c) and 2.2(d).

2.4.2.1 Dual-Lane Runway

A dual-lane runway consists of two closed-spaced (700 to 2499 ft) parallel runways with appropriate exit taxiways. Generally, it is desirable to dedicate the runway farthest from the terminal building (outer) for arrivals and the runway closest to the terminal building (inner) for departures. Estimates have shown that a dual-lane runway can handle at least 70 percent more traffic than a single runway in VFR conditions and about 60 percent more traffic than a single runway in IFR conditions [Horonjeff et al, 1983].

2.4.3 Intersecting Runways

Intersecting runways are two or more runways in different directions crossing each other. They are needed when relatively strong winds blow from more than one direction, resulting in excessive crosswinds when only one runway is provided. If the winds are relatively light, both runways can be used simultaneously with some operational dependence. But if the winds are strong, only one runway of a pair of intersecting runways can be used, thereby considerably reducing the capacity of the airfield. The location of the intersection (e.g., midway or near the ends) and on the way the runways are operated, known as strategy (takeoff or landing), play an important role in the capacity of the intersecting runway. Intersecting runways are shown in Figs. 2.2(e), 2.2(f) and 2.2(g). Higher capacities are achieved when the intersection is close to the takeoff end and the landing threshold.



(a) Single runway; (b) Two parallel runways--even threshold; (c) Two parallel runways--staggered threshold; (d) Four parallel runways; (e) Intersecting runways; (f) Intersecting runways; (g) Intersecting runways; (h) Open-V runways; (i) Open-V runways.

Figure 2.2 Typical Runway Configurations.
Source: Horonjeff et al, 1983

2.4.4 Open-V Runways

In open-v runways, the runways are placed in divergent directions and do not intersect. If the winds are relatively light, both runways can be used simultaneously with some operational dependence. But if the winds are strong, open-v runways revert to a single runway. The highest capacity is achieved when the operations are away from the V [Horonjeff et al, 1983].

2.5 Taxiway System

When the runway acceptance rate is increased, the capacity of the ground system will become the limiting factor. In order to attain the maximum capacity of an airport, a balance of the needs for runways, passenger and cargo facilities, and airplane storage and servicing areas should be realized. These separate and distinct airport components are tied together by a taxiway system. Taxiways, therefore, form the transitional media between the airports independent elements and are necessary for optimum airport utilization [FAA, 1970].

Fig. 2.3 depicts the taxiway and taxilane sections. Various components shown in the figure are defined in the following paragraphs.

(a) **Taxiway:** A defined path over which airplanes can taxi from one part of an airport to another. It includes the structural pavement, shoulder, taxiway safety area, and the obstacle free area. The taxiway drainage system, lighting, and marking are integral parts of the taxiway.

(b) **Shoulder:** An area abutting the edge of the taxiway structural pavement especially prepared to resist jet blast erosion and/or to accommodate maintenance equipment.

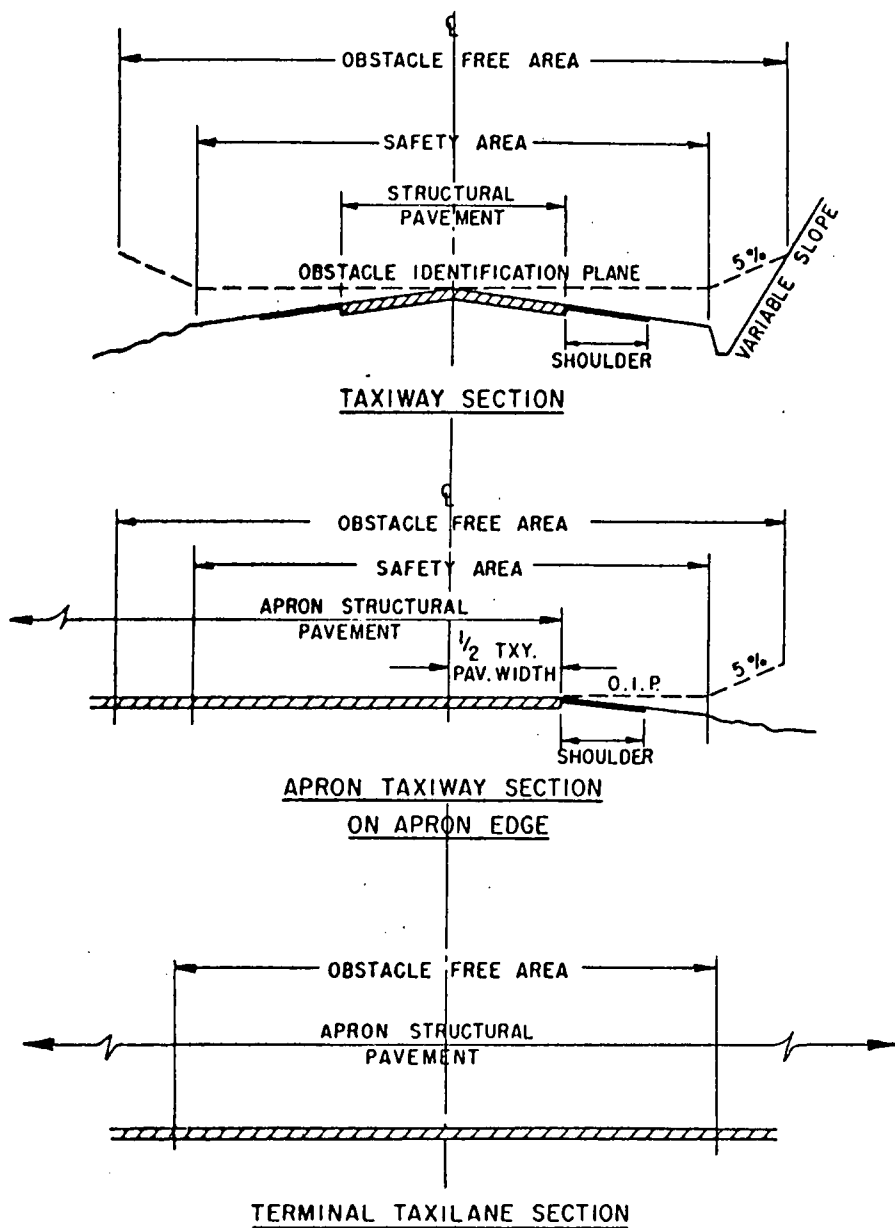


Figure 2.3 Taxiway and Taxilane sections.
Source: FAA, 1970.

(c) **Taxiway Safety Area:** An area symmetrical about the taxiway centerline which includes the taxiway structural pavement and shoulders. The portion abutting the edges of the shoulder are cleared, drained, graded, and usually turfed and should be capable of supporting fire, crash, and snow removal equipment under normal conditions.

(d) **Obstacle Free Area:** An area symmetrical about the taxiway centerline of specified width free of obstacles.

(e) **Obstacle Identification Plane:** An imaginary surface extending outward at zero slope from the centerline of the taxiway to the outer edge of the safety area and then continues outward and upward at a slope of 5 percent.

(f) **Obstacle:** Any fixed object natural or man-made, including earth and rock, excluding visual and navigational aids, that protrudes through the obstacle identification plane.

(g) **Terminal Taxilane:** A taxiway on an apron used for access to gate positions or other terminal maneuvers.

(h) **Holding Bay:** An area where airplanes can be held, or bypassed, to facilitate efficient ground traffic movement.

(i) **Airplane Parking Line Limit:** A line established by the airport authority beyond which no part of a parked airplane should protrude.

2.5.1 Requirements of the Taxiway System

A taxiway system should be planned such that it provides no restriction to individual movement to and from the runways. It should also maintain a smooth continuous flow of airplanes with a minimum of points requiring deceleration and acceleration. The taxiway flow pattern also places a major requirement on any new or modified airport design [FAA, 1970].

Taxiway design features necessary to achieve the maximum benefits from a taxiway system are as follows:

1. Avoid crossing runways where practical.
2. Be as straight and direct as possible.
3. Consider bypass capability and multiple access to runway ends.
4. Allow for visibility from air traffic control tower to taxiways and apron taxiways.
5. Basically consider a parallel taxiway serving each runway.
6. Contain sufficient appropriate exits to minimize runway occupancy time.
7. Always provide ample curve and fillet radii.
8. Build longevity into the system.
9. Create no bottlenecks.

The taxiway system comprises of through taxiways and intersections. This includes entrance and exit taxiways, bypass, crossover or transverse taxiways, and parallel and dual parallel taxiways. Each of these taxiways are discussed in ensuing paragraphs.

2.5.2 Dimensional Design Criteria

Since the speeds of aircraft are considerably less on taxiways, than on runways, criteria governing longitudinal slopes, vertical curves, and sight distance are not as stringent as for runways. Also the lower speeds permit the width of the taxiway to be less than that of the runway. The principal geometric design features of interest, the taxiway dimensional standards, are listed in Table 2.4 [Horonjeff et al 1983].

Table 2.3 Runway Dimensional Standards, FAA.

		Approach categories A, B, utility			Approach categories C, D, E, transport		
		Visual and Non-Precision		Precision			
		I	II	I	II	III	IV
Width, ft							
Pavement	60	75	100	100	100	150	200
	120	150	300	300	500	500	500
Safety					10	20	35
Shoulder							40
Gradient, %							
Pavement, longitudinal							
maximum	2.0	2.0	2.0	2.0	1.5	1.5	1.5
maximum effective	2.0	2.0	2.0	2.0	1.0	1.0	1.0
maximum change	2.0	2.0	2.0	2.0	1.5	1.5	1.5
Transition curve rate of							
slope change per 100 ft	0.33	0.33	0.33	0.1	0.1	0.1	0.1
Pavement transverse							
maximum	2.0	2.0	2.0	2.0	1.5	1.5	1.5
Safety area							
maximum longitudinal	2.0	2.0	2.0	2.0	1.5	1.5	1.5
maximum transverse	5.0	5.0	5.0	5.0	3.0	3.0	3.0

SOURCE: Federal Aviation Administration, 1982.

Table 2.4 Taxiway Dimensional Standards, FAA.

	Approach categories A, B, utility			Approach categories C, D, E, transport							
	Visual and Non-Precision			Precision							
	I	II	I	I	II	I	II	III	IV	V	VI
Width, ft											
Pavement	25	35	25	25	35	25	35	50*	75	75	100
Safety	50	80	50	50	80	49	79	118	171	197	262
Shoulder				10		10	10	20	25	35	40
Gradient, %											
Pavement, longitudinal											
maximum	2.0	2.0	2.0	2.0	2.0	1.5	1.5	1.5	1.5	1.5	1.5
maximum effective											
maximum change				3.0		3.0	3.0	3.0	3.0	3.0	3.0
Transition curve rate of											
slope change per 100 ft				1.0		1.0	1.0	1.0	1.0	1.0	1.0
Pavement transverse											
maximum	2.0	2.0	2.0	2.0	2.0	1.5	1.5	1.5	1.5	1.5	1.5
Safety area											
maximum longitudinal	2.0	2.0	2.0	2.0	2.0						
maximum transverse	5.0	5.0	5.0	5.0	5.0	3.0	3.0	3.0	3.0	3.0	3.0

* Use 60 ft if wheelbase is at least 60 ft.

SOURCE: Federal Aviation Administration, 1982.

2.5.3 Parallel Taxiway

A full parallel taxiway is considered a basic prerequisite for air carrier operations. Separation distance from parallel taxiway is connected to each end of the runway, and connecting transverse taxiways between the runway, parallel taxiway, and apron. Separation distance from the runway should satisfy the minimum specified by the FAA [FAA, 1970]. Excessive longitudinal grades should be avoided while designing the parallel taxiway centerline profile.

2.5.4 Taxiway Intersections

There are several ways of negotiating a turn by an airplane pilot. This may be done by tracking nosewheel on centerline, or by judgmental oversteering. The latter method, from a design standpoint, results in the least amount of pavement. The other method, from an operational standpoint, will reduce the possibility of excursion from the taxiway. Fig. 2.4 illustrates common taxiway intersections. The dimensional details are provided in Table 2.5 in accordance with the airplane design group [Horonjeff et al 1983].

2.5.5 Entrance Taxiways

Various configurations of runway entrance taxiways were proposed to increase the capacity of a runway. Entrance taxiways proposed by operational and traffic control groups permit acceleration while turning onto the runway. This would increase the capacity of the runway.

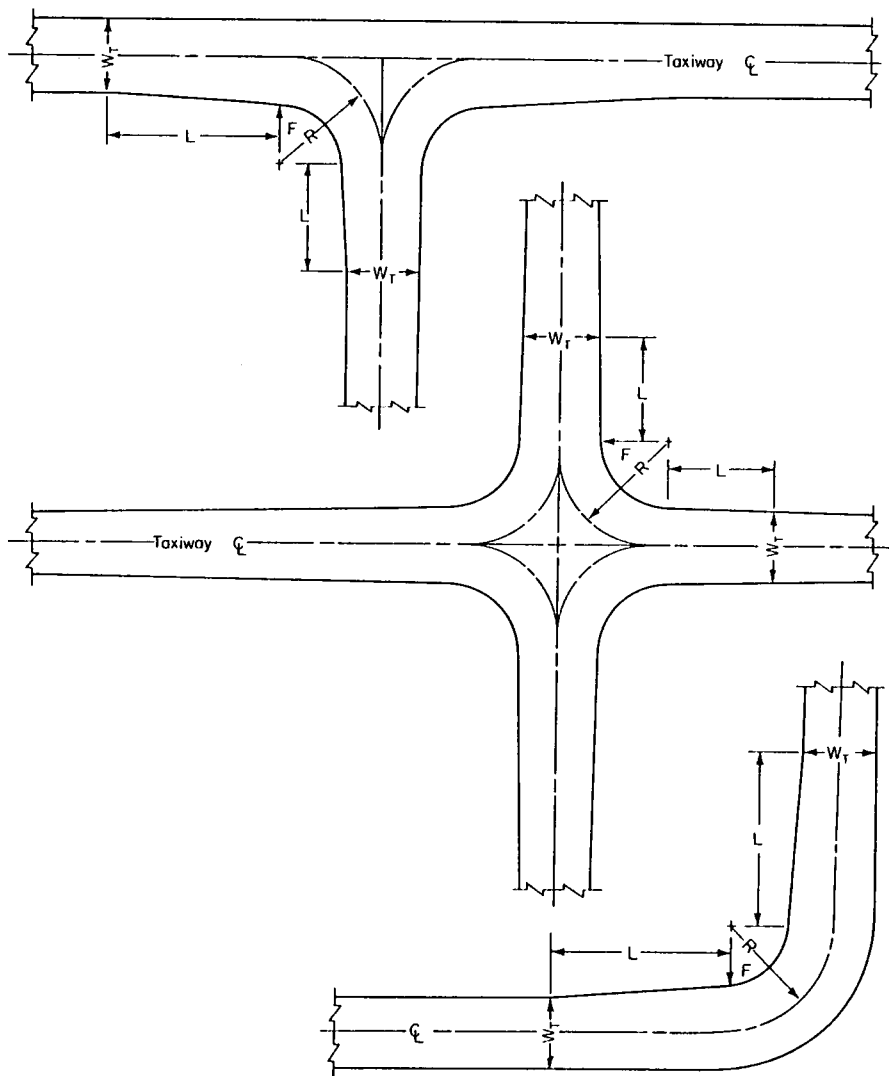


Figure 2.4 Common Taxiway Intersections.
Source: FAA, 1970.

An entrance taxiway which may also serve as the last exit taxiway on a bi-directional runway, is generally in the form of an "L" taxiway intersection with a right angle connection to the runway. In order to accommodate the higher speeds, the centerline radius of curve should be large [FAA, 1970]. The separation distance between runway and the parallel taxiway defines this centerline radius of curve.

2.5.6 Bypass Taxiways

Bypass taxiways are provided for flexibility in runway use by facilitating ground maneuvering for departing airplanes. They are normally located at or near the runway entrance. This taxiway feature is usually required at airports where activity is forecast to reach 20 total operations at peak hour or 50,000 air carrier operations annually [FAA, 1970].

2.5.7 Holding Bays

Holding bays or aprons are provided to enhance maneuverability for holding airplanes while at the same time permitting bypass. Holding bay may be provided at any suitable location, but it is more advantageous to locate it adjacent to the taxiway serving the runway end [FAA, 1970]. As in the case of bypass taxiway, this facility is warranted when air carrier operations are forecast to reach 50,000 or 20 total operations at peak hour.

2.5.8 Dual Parallel Taxiways

Multiple access to runways should be considered in airport planning for high density locations. Dual parallel taxiways provide for departure sequencing flexibility and increase in

handling efficiency. Along with dual parallel taxiways, crossover taxiways should also be provided to increase flexibility. In order to facilitate traffic flow, a central taxiway located between two closely spaced parallel runways should be built.

Partial dual taxiways may be required where air carrier operations reach 100,000 annually. A complete dual taxiway system is warranted when air carrier annual operations exceed or are forecast to exceed 150,000. The central taxiway is warranted when dual taxiways cannot be provided or when air carrier operations reach 200,000 [FAA, 1970].

2.5.9 Exit Taxiways

The efficiency of a runway, in terms of the volume of traffic that can be handled in a given time period, is directly related to the capability of the taxiway system to facilitate the movement of traffic to and from the runway. The exit taxiways should be so located that they provide for minimum runway occupancy time while serving each type of airplane under variable landing conditions. The economical optimum number and spacing, however, can be related to groups of airplanes and their average deceleration from touchdown to a point in the landing roll where turnoff can be negotiated. Effective exit taxiways are those that provide for free flow to the main parallel taxiway or at least to a point where the airplane is considered clear of the runway. Thus allowing another operation on the runway. The taxiway exit geometry, recommended by FAA is shown in Table 2.5 [Horonjeff et al, 1983].

A decision to design and construct an acute angled exit taxiway is based upon an analysis of the existing and contemplated traffic. The angled exits, commonly referred to as "high speed exits" are to enhance the capacity of the airport. The selection of the type of exit taxiway depends on the separation between the taxiway and runway. A separation distance of 600 feet or more permits the most efficient angled exit taxiway design. It provides sufficient room for a reverse curve in cases where the airplane is required to "double back" to its terminal destination. A 400-foot separation provides for an acceptable design only where the

Table 2.5 Taxiway Exit Geometry Recommended by FAA, ft.

	Airplane Design Group					
	I	II	III	IV	V	VI
Taxiway width WT	25	35	50*	75	75	100
Centerline radius R	75	75	100	150	150	170
Length of fillet lead-in L	50	50	150	250	250	250
Filletlet radius F	60	55	55*	80	85	85

If wheel base is equal to or greater than 60 ft, use a taxiway width of 60 ft and a fillet radius of 50 ft.

SOURCE: Federal Aviation Administration, 1982.

traffic flow is in one direction. If a dual taxiway is provided, however, the reverse curve can be accommodated. The 30 degree high speed exit taxiway is shown in Fig. 2.5.

When the design peak hour traffic is less than 30 operations, including landings and takeoffs, the normal perpendicular exit taxiway will suffice. This 90 degree exit taxiway can be constructed at less cost and, when properly located along the runway, achieves an efficient flow of traffic. But, from the standpoint of minimizing runway occupancy time, these exit taxiways are not desirable. The 90 degree exit taxiway is shown in Fig. 2.5.

2.5.9.1 Configuration

The establishment of a single standard for angled exits has many advantages. Pilots become familiar with the configuration and can expect the same results when landing at any airport with these facilities.

- In order to achieve the desired entrance speed (or runway turnoff speed) capability of up to 60 m.p.h., a minimum radius of curve of 1,800 feet is required. The angle for the 30 degree angled exit taxiway may be varied. But the curve radii should be maintained as standard.
- The entrance point at which turnoff speed of 60 m.p.h. may be realized is located on the runway centerline at the beginning of the curve (point of curvature, P.C.) as shown on the figures.
- The taxiway centerline marking starts 200 feet back of the P.C., and is offset three feet from the runway centerline.

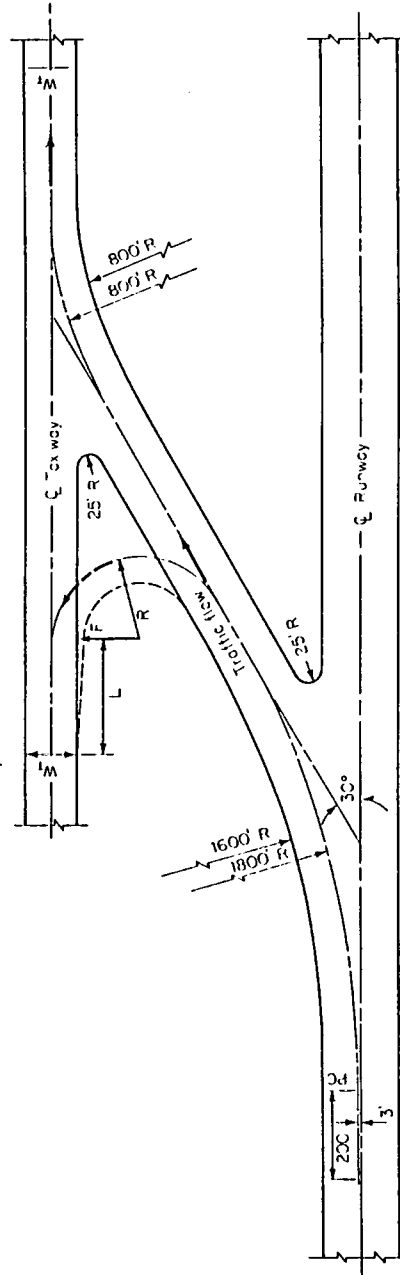


Figure 2.5. 30-Degree High Speed Exit Taxiway.
Source : FAA, 1970 & 1982.

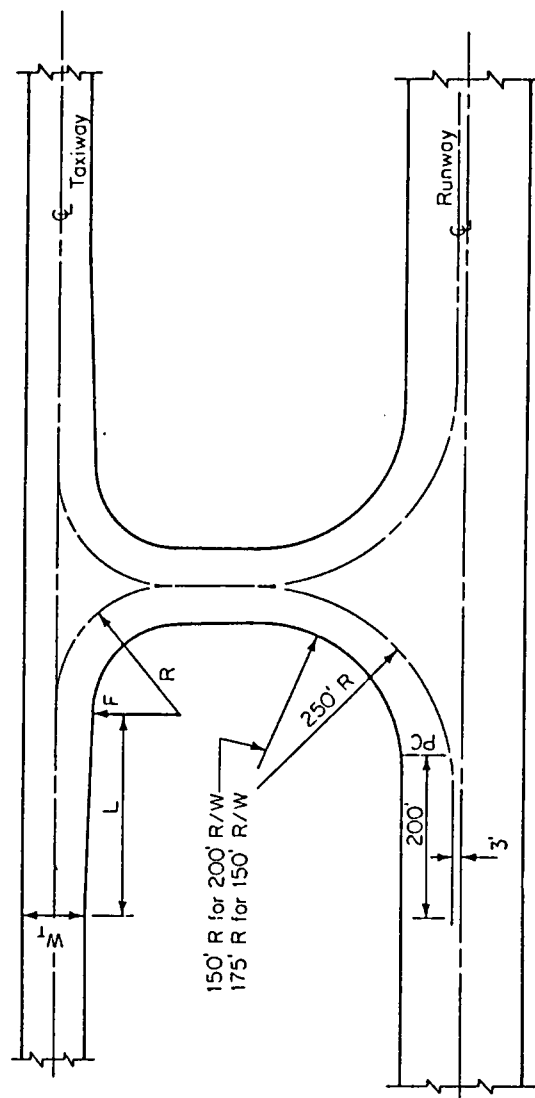


Figure 2.6. 90-Degree Exit Taxiway.
Source : FAA, 1970 & 1982.

2.5.10 Exits for Small Airplanes

An intersection angle of 45 degrees is used at air carrier airports where there are extensive operations of small airplanes (under 12,500 pounds maximum gross takeoff weight). This angled exit will have the capability of accepting a turnoff speed of 40 m.p.h. which dictates a minimum curve radius of 800 feet [FAA, 1970]. The 45 degree angled exit taxiway for small airplanes is shown in Fig. 2.7.

2.5.11 Apron Taxiways and Terminal Taxilanes

Requirements often exist to provide through taxi routes across an apron and to provide secondary access to gate positions and/or other terminal areas. Apron taxiway require the same clearances and separations as normal taxiways. When a ground vehicle service road or route is required adjacent to an apron taxiway, the clearance to be provided depends upon local conditions and amount of control maintained over the ground traffic.

Terminal taxilanes provide access from through apron taxiways to airplane parking positions and other terminal areas. The only design requirement is to provide the obstacle free area. Visibility from the air traffic control tower should be such that there is clear line-of-sight to the centerline of all apron taxiways. This is also desirable for terminal taxilanes, but the requirement can be satisfied where adequate control of the airplane exists through other means [FAA, 1970].

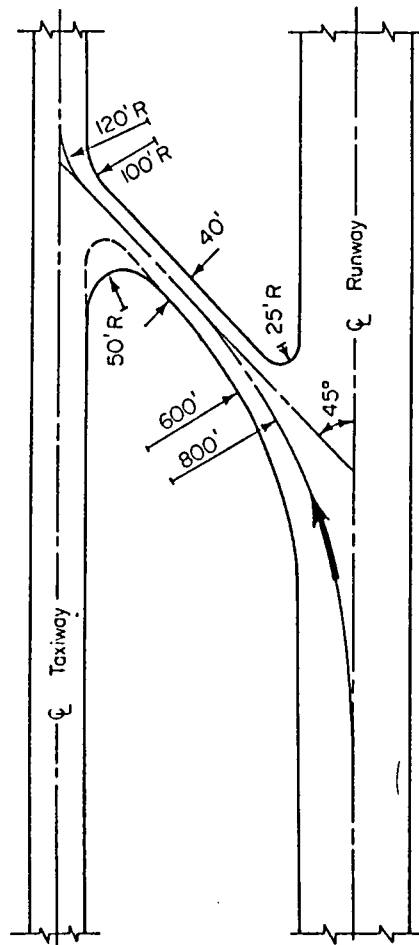


Figure 2.7. 45-Degree Exit Taxiway.
Source : FAA, 1970 & 1982.

2.5.12 Taxiway Safety Area and Shoulders

When the design and construction of taxiways are contemplated, the adjacent terrain or embankment along the taxiway would be graded to within specified tolerances, for a certain minimum distance. This taxiway safety area provides an obstacle free area intended only for minimizing the probability of serious damage to airplanes using these areas accidentally or in an emergency. Under normal conditions, it should be capable of supporting ready access by fire, crash, and snow removal equipment over its entire area [FAA, 1970].

On certain jet airplanes, the engines extend beyond or overhang the edges of the taxiway pavement which may create detrimental conditions of ingestion and blast erosion. Precaution must, therefore, be taken to reduce this possibility by providing shoulders with a stabilized or paved surface.

3.0 MODEL DESCRIPTION

The Runway Exit Design Interactive Model, (REDIM) incorporates several airfield variables affecting the landing performance of the aircraft as well as other important operational constraints (e.g., aircraft mix) having a direct impact on the selection of the turnoff location and their geometry. The model comprises of five models: (1) an input module, wherein the data is entered by the user, (2) a dynamic simulation module to estimate the runway occupancy time of the aircraft, (3) an optimization module to determine candidate turnoff locations to comply with a desired reliability threshold value, (4) an aircraft data module containing relevant aircraft performance and geometric parameters, and (5) an output module to show graphically the suggested runway turnoff configuration and display some measures of effectiveness desired by the analyst. The program contains a library of geometric and operational aircraft characteristics to allow the analyst to choose from a wide selection of operational airport scenarios. Each of these models are described in ensuing paragraphs.

3.1 *Input Module*

This module forms the main module of the program. It comprises of a series of interactive screens that allow the user to enter and edit data necessary for the analysis portion of the program (i.e., Dynamic and Optimization Modules). The program first offers a main menu which provides access to all the modules in the program. The options provided by the "Main

Menu” are: 1) “Start for a New Problem”, 2) “Edit Data”, 3) “Begin Analysis”, 4) “Edit Master File”, 5) “Go to Output Module”, 6) “Help” and 7) “Quit”.

Input data is classified into five broad categories. These five categories are essential for the analysis, and should be saved in a ‘working data file’ specified by the user with an arbitrary name. For the convenience of the user, the aircraft characteristics, which are saved in a ‘master data file’ are transferred to the ‘working data file’ automatically, when necessary. The following paragraphs define the categories in more detail.

(1) **Analysis type and related data:** The program provides the user with three choices of analysis type: (a) Evaluate an existing runway, (b) Improve an existing runway, and (c) Design a new runway. For each type of analysis, there are some accompanying data which vary depending on the user’s choice.

(2) **Aircraft mix and characteristics data:** In this category the percentages of the aircraft which consists of the mix of aircraft, the characteristics data of the aircraft, and the desirable exit speed are included.

(3) **Airport operational data:** This includes the free roll time between the touchdown and the beginning of braking, free roll time between the end of braking and the beginning of turnoff, taxiing speed, and their standard deviation.

(4) **Airport environmental data:** In this category, wind speed, wind direction, airport elevation, airport temperature, runway orientation, visual range, runway width, and distance to the nearest taxiway are included.

(5) **Runway gradient and friction data:** In this category, runway length, the effective gradient for every one tenth of runway, and pavement condition are included.

3.1.1 Data Input Method

Three different input methods are used in this module: 1) menu input, 2) line input, and 3) table input. Menu input is used when the user needs to select a particular choice among

a list displayed on the screen using the arrow keys and enter key. The flow in the module is controlled by the menu input method. Main menu, edit menu for working data file, edit menu for master data file, selection of an analysis type, etc., are the examples of the menu input method. Line input is that when the user inputs a numerical value like runway length or a string datum like the data file name at the position specified on screen. The user inputs file names (data and/or output file), the number of exits, the location of exits, etc., using this method. Table input is similar to line input. However, table input is used in order to get several numerical data on the same screen, while line input is used in order to get one numerical or string datum on a line. By the table input method the user inputs aircraft mix data, aircraft characteristics data, airport environmental data, etc.

3.1.2 Procedures in Input Module

The first screen presented to the user after the title screen is the main menu. As mentioned earlier, the main menu screen provides seven options: 1) Start for a New Problem, 2) Edit Data, 3) Begin Analysis, 4) Edit Master File, 5) Go to Output Menu, 6) Help, and 7) Quit. The function of each of the choices 3) and 5) is to transfer the flow control to the other modules. The purpose of choice 6) is basically to help the user by giving a brief explanation of the modules. The choices 1), 2) and 4) invoke the procedures belonging to the Input Module and finally choice 7) is provided to end the program.

3.1.2.1 *Start a New Problem*

In this mode, all the data necessary for the analysis should be provided by the user. Once the user enters this mode a complete set of values is expected before completing the entire input process. The categories wherein the user needs to input data are: (1) Analysis type and

related data, (2) Aircraft mix and characteristics data, (3) Airport operational data, (4) Airport environmental data, and (5) Runway gradient and friction data. All these categories were explained earlier at the start of this module.

3.1.2.2 *Edit Data*

This portion of the program allows the user to modify existing data files. All the five different categories, mentioned in the previous paragraph, can be edited selectively.

3.1.2.3 *Edit Master File*

While the function of 'Edit Data' mode is editing the working data file, the function of this mode is to edit the Master Data File which keeps the aircraft names and their characteristics. Two options are provided in this mode: (1) "Add a New Aircraft", and (2) "Change some specific Data". That is, the user can either enter a new aircraft along with its characteristics or change some specific data which already exists in the master file.

3.2 *Dynamic Module*

The Dynamic Module comprises several computational subroutines used to evaluate the aircraft performance during the landing flare maneuver, on the runway and during the turnoff maneuver. For modeling purposes, the aircraft flight and ground paths have been divided into five distinct segments as shown in Fig. 1.6.: 1) an air distance, 2) a free roll distance between the air and braking segments, 3) the braking segment, 4) a second free roll distance between the braking and turnoff segments, and 5) the turnoff distance until clearing the runway for the

next arrival. This segmentation eases the computational tasks in the model and also simplifies the inclusion in the model of operational policies arising at specific points in the landing phase. The following paragraphs describe the basic assumptions made in the dynamic analysis of the model, REDIM. Ultimately, the purpose of the Dynamic Model is the identification of potential candidate solutions to search for optimal turnoff locations.

3.2.1 Air Distance

The air distance is estimated indirectly from the basic geometric and performance characteristics. The Master File in the model contains geometric and basic aircraft performance characteristics that are used to estimate approach and touchdown speeds for each aircraft selected by the user. Once the approach speed (or reference speed) is known, an estimation of the air distance can be made by assuming a circular arc flare maneuver flown at constant load factor to transition from a constant rate of descent flight angle on final approach to a flat flight profile tangent to the runway.

The advantage in estimating air distance relying on information pertaining to each aircraft is two-fold: 1) frees the analyst from relying on field data for a particular aircraft population that in most cases is not available or which could be implemented at a later stage for calibration of the model, 2) introduces more realistic variations in the aircraft touchdown locations instead of assigning a fixed touchdown location to an entire aircraft category population.

3.2.2 Free Roll Distances

Free roll distances arise in the aircraft landing operation at two different times: 1) between the air distance and the braking stage, and 2) between the braking segment and the turnoff

maneuver. The first free roll distance tries to simulate an inherent human delay to apply braking mechanisms such as thrust reversers, spoilers, or normal wheel braking. A conservative average value of three seconds has been allocated for this transition stage with a typical standard deviation of one second. However the user can selectively modify these values through the Input Module.

The second transition segment tries to mimic a delay time arising from the proper suppression of braking action and recognition time of the turnoff geometry prior to exiting the runway. The last point could be critical under low runway visual range (RVR) and night conditions.

3.2.3 Braking Distance

Under normal landing conditions, the braking segment constitutes the largest component of the runway occupancy time. As such, it becomes necessary to estimate with some accuracy the braking distance if one is to have some confidence in the total distances covered by the aircraft on the ground. The previous requirement also stems from the incorporation of airport specific variables such as local runway slope and its effect on aircraft deceleration characteristics. The problem seems to be complicated by the fact that many aircraft parameters necessary to determine the forces and moments acting on the aircraft as its brakes are not only time dependent, (e.g., thrust reverse forces, braking forces, parasitic drag contributions, etc.,) but also aircraft specific in most instances (e.g., a small reciprocating aircraft have no thrust reverse capability while turbofan and turbopropeller-driven aircraft do).

The braking algorithm used in the model integrates the local deceleration rate of the aircraft as it travels along the runway. The local deceleration is estimated from the runway initial conditions specified by the user in the Input portion of the program. At the same time a deceleration multiplier is computed throughout the integration process in order to correct local variations of runway slope. This simplistic model then treats the aircraft as a second

order point mass model whose resultant drag force is integrated over time to obtain the velocity/distance states.

3.2.4 Turnoff Algorithm

The turnoff algorithm integrates the aircraft path throughout the exit maneuver. The exit maneuver is initiated when the aircraft reaches the user-defined exit speed and finalizes with the complete clearance of the runway by the landing aircraft. In this maneuver, the imposition of side loads on the landing gear is comprised of three fundamental forces: 1) the centripetal force, 2) the aircraft inertia, and 3) the tire scrubbing resistance to the turn. The approach used here is a modified version of that used by Schoen et al [Schoen et al, 1985] considering the aircraft parameters and time spent on the turn to be deterministic in nature.

3.2.5 Exit Assignment Algorithm

The exit assignment algorithm, as its name implies estimates the probable exit that an aircraft would take under a given set of operating conditions. The algorithm is particularly useful under fixed location turnoffs where the user has complete control of the turnoff locations and runway occupancy times are calculated for all exits available.

3.2.6 Airport Environmental Variables

Airport environmental variables play a significant role in the computation of the dynamic performance of the aircraft on the ground. The environmental characteristics of interest are:

1) wind speed, 2) wind direction, 3) airport elevation, 4) airport temperature, 5) runway orientation, 6) runway visual range, 7) runway width, and 8) distance to nearest taxiway.

The wind vector is used in conjunction with the runway orientation to estimate the longitudinal and lateral wind components affecting aircraft operations. The longitudinal wind component affects the landing speeds of the aircraft population and as such has direct impact in the runway occupancy time and turnoff locations. Temperature and airfield elevation have a direct impact in the performance of the aircraft in the air and on the ground. Changes to the aircraft equivalent airspeed (EAS) due to temperature and field elevation can have large impact in the ROT and the turnoff location parameters. The model estimates the aircraft stalling and approach speeds corrected through equivalent relative density conditions dictated by the airfield temperature and density. The runway width and runway distance to nearest taxiway are included in this set of parameters in order to estimate the time spent on the turnoff maneuver by each aircraft.

3.2.7 Aircraft Characteristics

The aircraft characteristics used in the model are mass, wheelbase, load on main landing gear, wheel-track, wing area, and the maximum lift coefficient. The aircraft mass, wing area, and the maximum landing lift coefficient dictate the approach speed and hence affect the ROT and exit location. The aircraft mass is also used to estimate the second moment of inertia of the aircraft around the vertical axis which influences the turning aircraft capabilities through a turnoff.

3.3 Optimization Module

The best turnoff locations for each aircraft, for both dry and wet runway surface conditions, are calculated through the simulation of aircraft landing movements. The best turnoff location is defined as the nearest location from the runway threshold where the aircraft decelerates to a pre-defined desirable exit speed with the specified reliability. If the aircraft reduces its speed to the pre-defined exit speed before reaching the assigned turnoff location, then the aircraft will be considered to exit the runway successfully. Reliability is defined as the probability that the aircraft exits the runway successfully. For example, if the reliability is specified as 95%, then 95 landing aircraft out of a total 100 landing attempts will exit the runway successfully. When an exit is constructed at the turnoff location for an aircraft, the runway occupancy time of the aircraft will be minimized without sacrificing the reliability.

When five aircraft are taken into consideration, the simulation of the aircraft landing movements, for two runway surface conditions, will provide ten different best turnoff locations for each aircraft. But the essence of the major project "Runway Exit Design Interactive Model" (REDIM), is to optimize these turnoff locations based on certain criteria. Therefore, optimization is done to find few locations (e.g., 2 or 3) at which all the aircraft in consideration can exit the runway with the minimum weighted sum ROT. Since each aircraft for different runway surface conditions can have different relative frequency, the weighted sum of ROT should be minimized instead of total ROT.

The optimization procedure in REDIM is performed with the following four steps:

1. Generate the complete set of candidate locations.
2. Calculate the ROT of every aircraft for each candidate location.
3. Find optimal location(s) out of the candidates.

4. Assign aircraft to the optimal locations.

3.4 Output Module

This module is provided to show graphically and in tabular form the suggested runway turnoff configuration and display certain measures of effectiveness desired by the analyst. Some of the screens that are displayed in the program pertaining to this module include: ROT / Reliability Table, Optimal turnoff locations along the runway downrange, Table of exit numbers and their location, Comparison of centerline turnoff geometries, Bar charts portraying the ROT's of each of the aircraft for both wet and dry runway surface conditions, Complete fixed turnoff geometries for 30, 45 and 90 degrees specified by the FAA and the variable turnoff geometry developed by REDIM etc.

The first screen that is presented in this module is the "Output Menu". The options that are presented in this module are: (1) View the Output, (2) Print the Report, (3) Help, and (4) Go to Main Menu. Choice (2) presents a hard-copy report which is a complete report of the "Runway Exit Design Interactive Model", choice (3) is provided to help the user with an on screen help menu, and choice (4) allows the user to return to the Main Menu.

The first option of the output menu, "View the Output", presents the appropriate view menu screen consistent with the runmode option selected in the Input Module. Runmode options are the three options provided in the Input Module within the mode 'Type of Analysis'. The three type of analysis available are: 1) Evaluate an existing runway, 2) Improve an existing runway, and 3) Design a new runway. These should to be selected at the start of the program. All the options that are provided in the "View Menu" are explained briefly below.

(i) ROT / Reliability Table:

This screen displays the number of turnoffs, their location, and the type of turnoffs. When runmode '1' (i.e., 'Evaluate an existing runway') is selected, then for each of the aircraft

selected earlier in the Input Module, the corresponding runway occupancy time and the reliability associated with each of the turnoffs is displayed. For runmodes '2' or '3' (i.e., either 'Improve an existing runway' or 'Design of a new runway') the corresponding runway occupancy time associated with the appropriate turnoffs is displayed for each aircraft. The reliability value for these runmodes, provided by the user in the Input Module, is also displayed on this screen. The ROT and the reliability value associated with each aircraft help the user in assigning appropriate turnoff(s) for that particular turnoff.

(ii) Turnoff Locations and their Geometries:

This screen displays the optimal turnoffs located along the runway downrange, a table with the exit numbers and their location, and a window providing choice to view either the centerline turnoff geometries or the complete turnoff geometry for the optimal turnoffs. An option to view the table with the coordinates of the centerline turnoff geometries is also provided. It should, however be understood that there are slight variations for each of the runmodes (i.e., the type of analysis) selected which are explained in Chapter 4 of this report.

(iii) Aircraft Statistics and their ROT's:

This screen displays the runway occupancy time for each of the aircraft in the form of a bar chart with each aircraft represented by two bars. The first one represents ROT for dry runway surface condition whereas the second one represents ROT for wet runway surface condition. On the bar chart, a straight line is drawn across the bars to portray the weighted average ROT.

The complete structure of the Output Module is explained in detail in the next Chapter.

4.0 OUTPUT STRUCTURE

The Output Structure plays a very significant role in the package. This is the Module wherein the user is exposed to the much awaited output in a discrete form. The program is so structured that at each stage the user is prompted with specific questions and guidelines that are to be followed in order to view the appropriate results. A significant feature in this module is that in each of the runmodes, all the screens clearly display information regarding the aircraft and the airport data. This information provided at the top of each of the screens, will be of great use to the user as (s)he will be presented with the general information pertaining to that type of analysis.

After defining the variables in the Input Module, the user may go back to the Main Menu. From this menu, the user has the option to go to the Output Menu among other options. When the user chooses to go to the Output Menu, (s)he is provided with four options. The options being: (1) View the Output (2) Print the Report (3) Help and (4) Go to Main Menu. Fig. 4.1 clearly depicts the Output Menu on the screen. In the following paragraphs, all the options that are provided to the user in the Output Menu will be discussed in detail.

4.1 *View the Output*

When the user selects this option, the program automatically goes into the runmode variable that was previously specified by the user in the Input Module and present the View

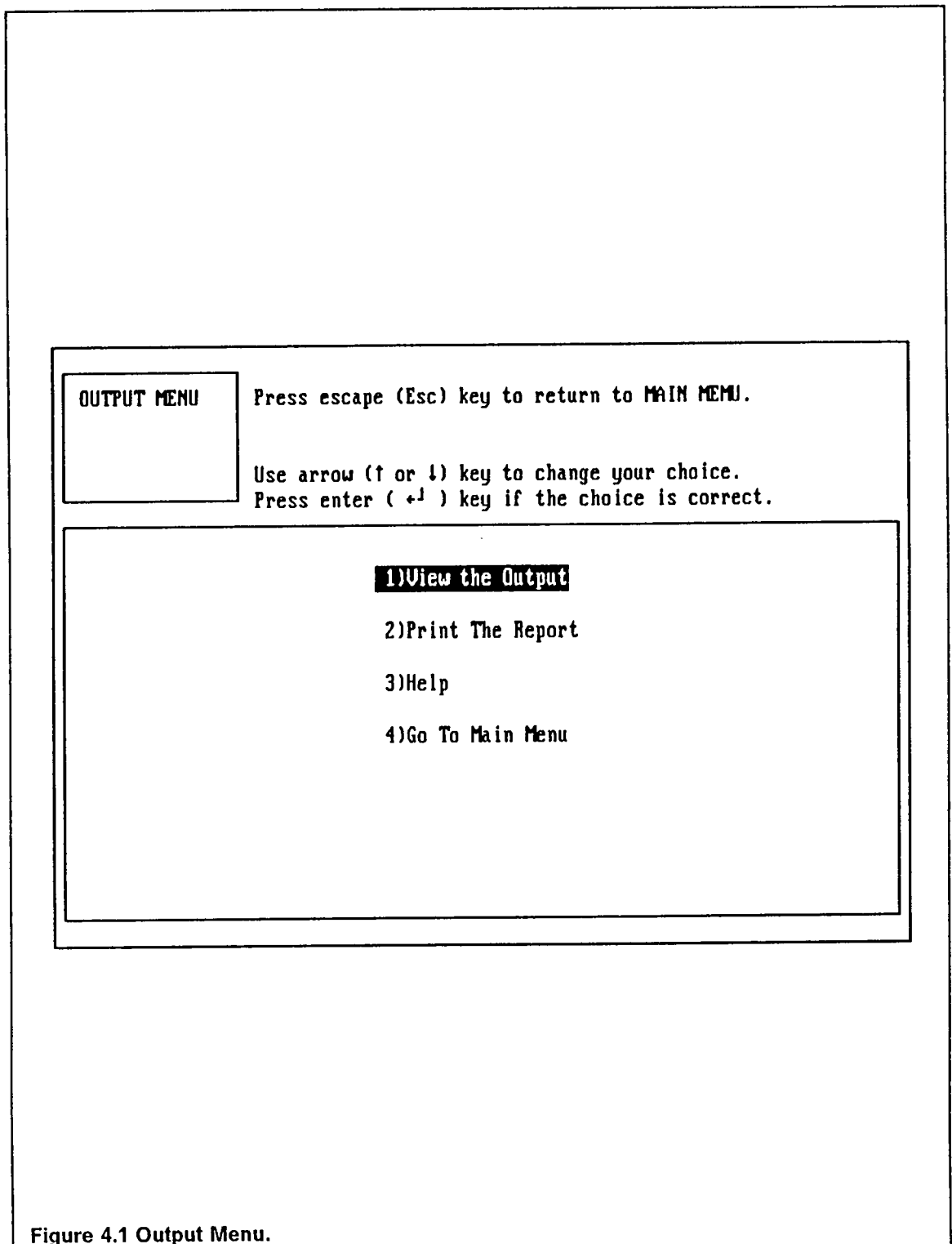


Figure 4.1 Output Menu.

Menu screen which corresponds to the aforesaid option. For each of the runmode variable options, the program presents different screens which prominently display the results in color, while at the same time guiding the user. Each screen also displays the general input variables that were initially provided by the user in the Input Module. The user at any stage in the Output Module is allowed to go back to the Main Menu, through which a wide range of options can be chosen. Now we will go into each of the runmode variable option and see explicitly as to what is presented, and also the extent of flexibility that is made available to the user.

4.1.1 Evaluate an Existing Runway

This runmode is specifically designed to evaluate an existing runway. The data required to evaluate an existing runway is entered by the user in the Input Module. As the user selects to view the Output from the Output Menu, (s)he is provided with a different screen which displays the View Menu. This View Menu corresponds to the runmode option which is to evaluate an existing runway. Three options are provided in this menu. The options being (1) ROT / REL Table of Results (2) Turnoff locations and their Geometries and (3) Go to Output Menu. The flow pattern for this runmode option is shown in Fig. 4.2 (attached at the end of the report) and the View Menu screen is illustrated in Fig.4.3. In the following paragraphs, we will discuss in detail these options.

4.1.1.1 ROT / Reliability Table of Results

This screen displays the number of turnoffs, their location, and the type of turnoffs. The FAA standard turnoffs, 30 degree, 45 degree and 90 degree are illustrated in Figs. 4.6, 4.7 and 4.8 respectively. For each of the aircraft selected earlier in the Input Module, the

VIEW MENU

Press escape (Esc) key to return to OUTPUT MENU.

Use arrow (↑ or ↓) key to change your choice.
Press enter (↵) key if the choice is correct.

1)View ROT Table

2)View Turnoff Locations And Their Geometries

3)Go To Output Menu

Figure 4.3 View Menu for 'Runmode 1'.

ROT / RELIABILITY TABLE
(This is for Evaluating an Existing Runway)

Exit # Location (m) Exit Type	1 1000 30-Deg	2 2000 45-Deg	3 3000 30-Deg
PA-38-112 DRY ROT (12.5%)REL	42.53 1.00	125.21 0.00	225.21 0.00
WET ROT (12.5%)REL	42.41 1.00	124.73 0.00	224.73 0.00
BE-300 DRY ROT (12.5%)REL	29.02 0.84	87.11 0.16	188.13 0.00
WET ROT (12.5%)REL	29.43 0.74	83.24 0.26	183.61 0.00
A-320-200 DRY ROT (12.5%)REL	30.93 0.00	51.52 0.99	128.17 0.01
WET ROT (12.5%)REL	32.11 0.00	47.58 0.93	113.72 0.07
MD-11 DRY ROT (12.5%)REL	34.78 0.00	48.44 0.22	83.77 0.78
WET ROT (12.5%)REL	36.54 0.00	50.30 0.02	73.64 0.98

ROT - Runway Occupancy Time in Secs
REL - Reliability
Average ROT = 63.97

Press any key to continue

Figure 4.4 ROT / Reliability Table of Results.

corresponding runway occupancy time and the reliability associated with each of the turnoffs is displayed. These help the user in judging the appropriate turnoff for that particular aircraft. One more significant feature in this table is the separation of the values for wet and dry airport conditions. The user is also in a position to view the change in runway occupancy time and the reliability associated with each of the turnoffs when the airport conditions (wet or dry) are changed. In addition to these, the relative frequency of occurrence of this aircraft, for a particular scenario (wet or dry conditions) is specified as a percentage of all the aircraft occurrences. This table is shown in Fig. 4.4.

In each screen as only four aircraft are shown, it is necessary to press 'F' key (for forward) to view additional aircraft and 'B' key (for backward) to view the previous screen. In each of the screens the viewer is presented with the average runway occupancy time which encompasses the whole population of aircraft selected by the user. These include dry and wet conditions at the airport. An option is provided to print the table by just pressing the 'P' key (for print). The user is also allowed to go back to the View Menu at any stage by entering 'V' key (for View Menu).

4.1.1.2 Turnoff Locations and their Geometries

In this screen, the user is presented with the display of turnoff locations along with other pertinent airport data. The turnoff locations along the runway downrange are shown in Fig. 4.5. An added feature in this screen is the presentation of exit numbers and their location in a tabular form. An option to view the standard FAA geometry for each of the turnoffs is also provided. The user is prompted to enter the exit number to view the standard FAA geometry. Each of the turnoffs may represent any of the standard FAA geometries viz., 30 degree or 45 degree or 90 degree. Only one exit number is to be entered to view the complete turnoff geometry. As the exit number is entered, the program determines the type of turnoff for the exit number from the Input Module. Earlier, in the Input Module, the user selected the type

of turnoff geometry for a particular exit number. This turnoff geometry may represent any of the three standard FAA turnoff geometries shown in Figs. 4.6, 4.7 and 4.8 respectively. After viewing the standard FAA turnoff geometry for the specified turnoff, the user is returned to the first screen where a choice for a different exit can be made. This screen would appear as shown in Fig. 4.4. The user may exit from this screen by entering '0' (zero).

When the user selects to view the standard turnoff geometry for a particular turnoff, the program automatically takes in the values of runway width, taxiway width, the distance between the runway and taxiway and the type of turnoff viz., 30 degree or 45 degree or 90 degree from the Input Model. This data was earlier supplied by the user in the Input Module. In the standard FAA turnoff geometry display, the specifications are prominently shown for the benefit of the user. Although, the Metric system is mainly used for computations in the program, the units for specifications are also displayed in the English system. This helps the user who might be still using the FAA standards in the English system.

4.1.1.3 Go to Output Menu

This option is provided to enable the user to go back to the Output Menu from the View Menu. The user may also exercise this option by pressing the escape 'Esc' key.

4.1.2 Improve an Existing Runway

This runmode option is made available to the user to improve or modify an existing runway. The required data of the airport facility that needs to be improved is entered in the model through the Input Module. The user after entering the data may go to the Output Menu through the Main Menu. The Output Menu screen is as shown in Fig. 4.1. and the flow pattern for this runmode option is shown in Fig. 4.9 (attached at the end of the report). As the user

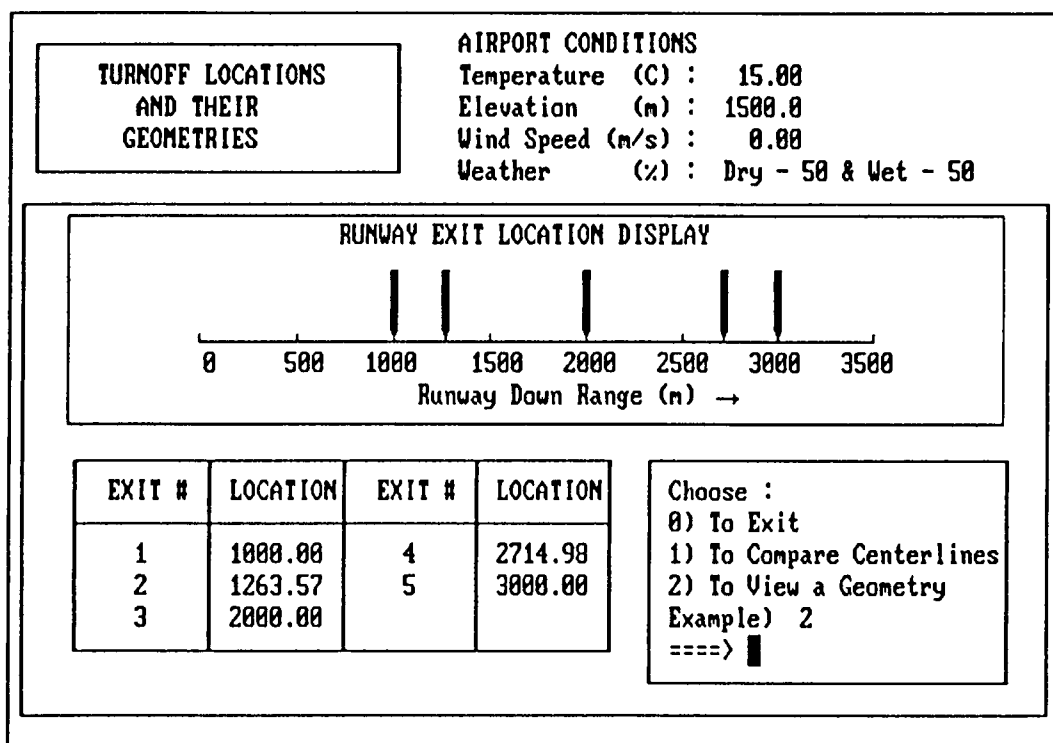
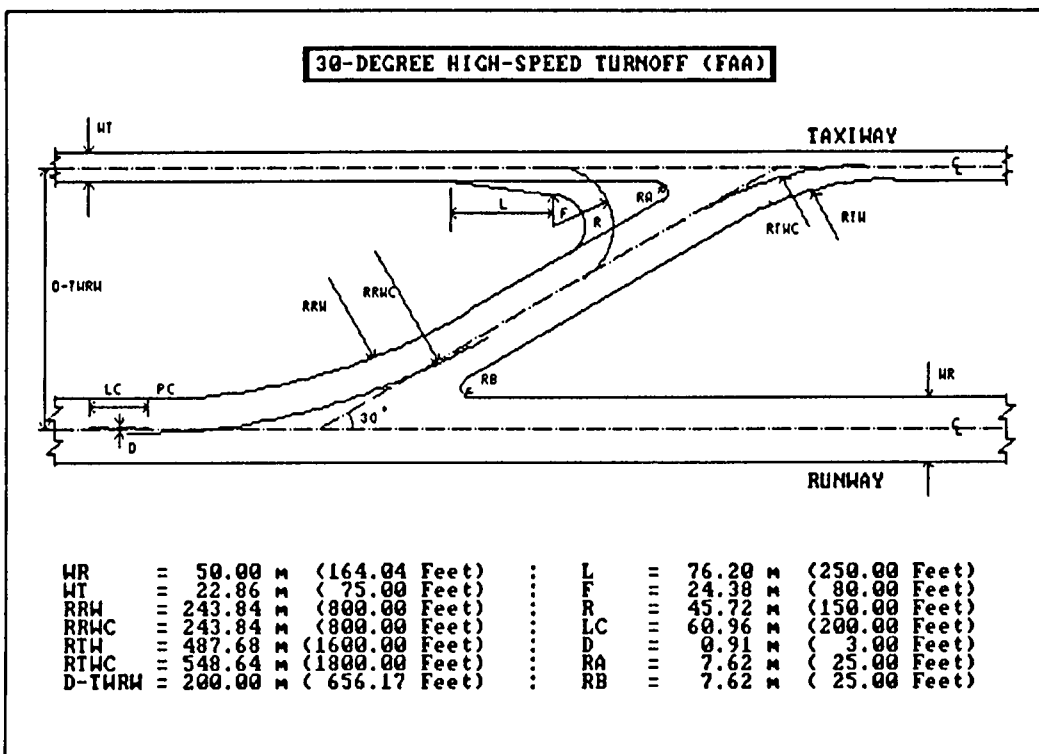
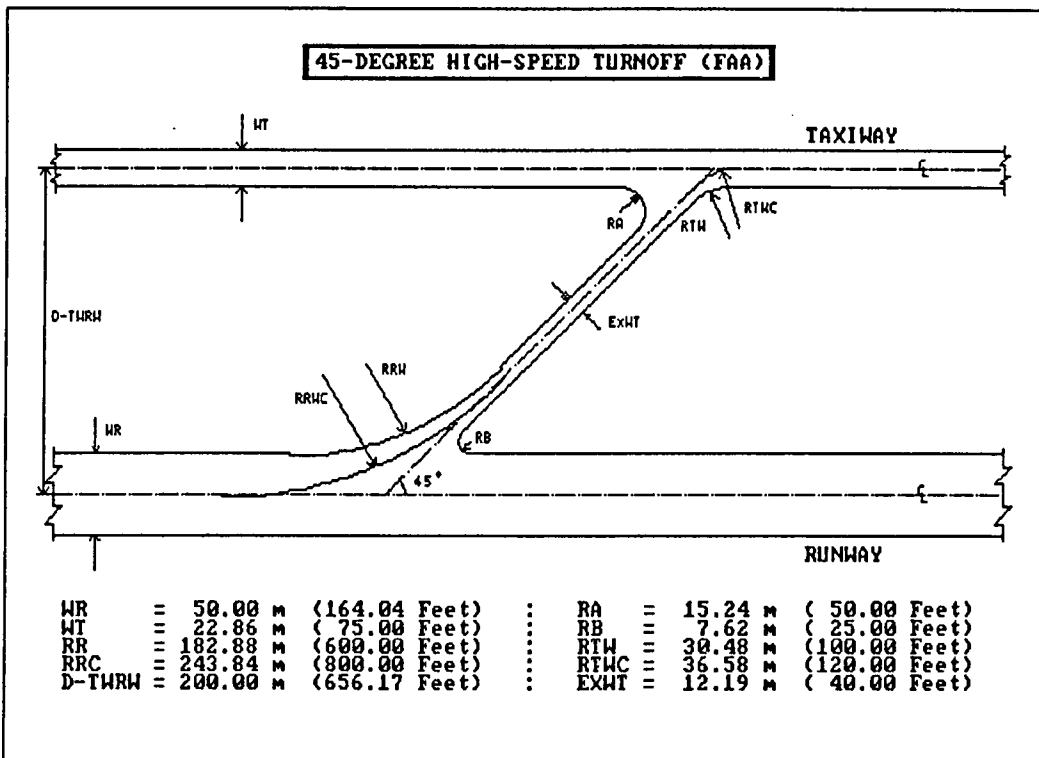


Figure 4.5 Turnoff Locations along the Runway Downrange.



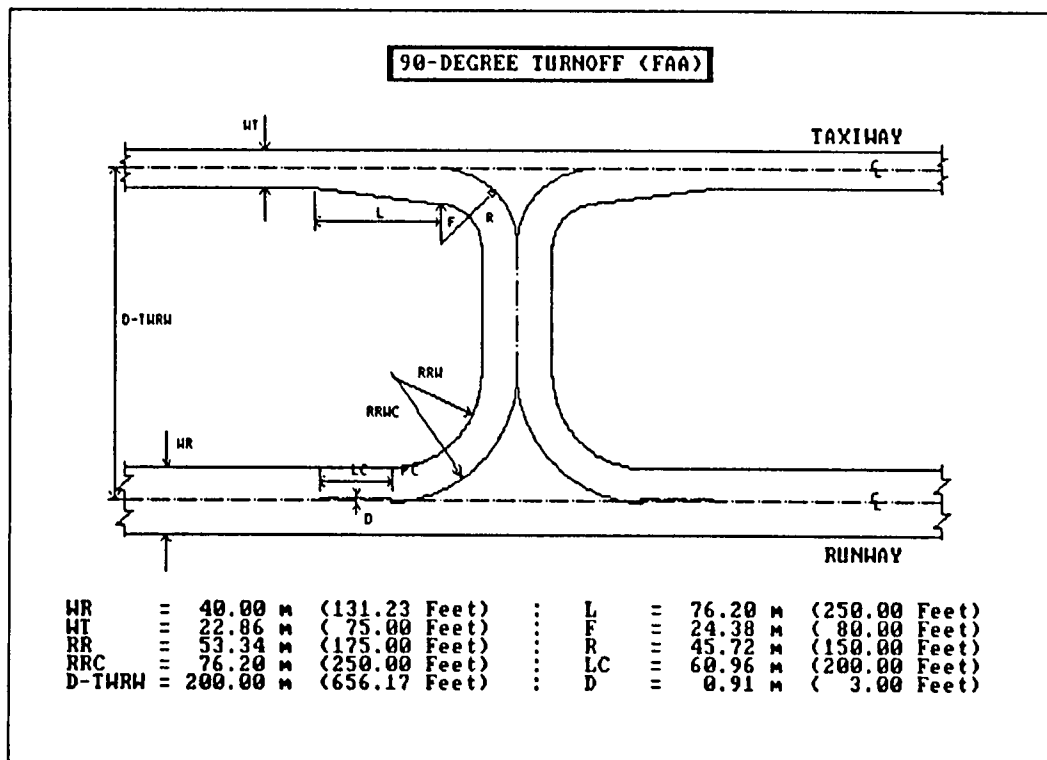
Press any key to continue

Figure 4.6 Standard FAA 30-Degree Turnoff Geometry.



Press any key to continue

Figure 4.7 Standard FAA 45-Degree Turnoff Geometry.



Press any key to continue

Figure 4.8 Standard FAA 90-Degree Turnoff Geometry.

selects to view the output from the Output Menu, he or she is presented with the View Menu screen. This View Menu screen corresponds to the runmode option which is to improve an existing facility. The View Menu screen is provided with four options: (1) ROT Table of Results, (2) Turnoff Locations and their Turnoff Geometries, (3) Aircraft Statistics and their ROT's, and (4) Go to Output Menu. The view menu screen is shown in Fig. 4.10. In the following paragraphs, we will discuss in detail each of the above said options.

4.1.2.1 ROT Table of Results

The table displays the number of turnoffs, the existing and the proposed ones, and their location. The type of turnoff geometry is also displayed in the third row of the table. The type of turnoff geometry either could be of standard FAA turnoff geometry viz., 30 degree or 45 degree or 90 degree, for the existing ones or a new turnoff geometry developed by the model for the proposed new ones. The new turnoffs are differentiated from the existing ones through color coding. This helps in knowing the location of new turnoffs at a simple glance. For each of the aircraft, selected earlier in the Input Module, the corresponding runway occupancy time associated with the appropriate turnoff is displayed. The relevant values for both the airport conditions, wet and dry are displayed. The user is also in a position to view the change in runway occupancy time associated with appropriate turnoff when the airport conditions (wet or dry) are changed. In addition to these, the relative frequency of occurrence of this aircraft, for a particular scenario (wet or dry conditions) is specified as a percentage of all the aircraft occurrences. The empty boxes in the table imply that the aircraft in question cannot negotiate that particular turnoff. The reliability associated with the aircraft and turnoff, which was earlier provided by the user in the Input Module, is displayed at the bottom of the table.

In each screen, as only four aircraft are shown in the table, it is necessary to press 'F' key (for forward) to view additional aircraft and 'B' key (for backward) to view the previous screen. In each of the screens the user is presented with the average runway occupancy time

which encompasses the whole population of aircraft selected by the user. These include dry and wet conditions at the airport. The user is provided with an option to print the table by just pressing the 'P' key (for print). The user is also allowed to go back to the View Menu at any stage by entering 'V' key (for View Menu). This table is illustrated in Fig. 4.11.

4.1.2.2 Turnoff Locations and their Geometries

In this screen, the user is presented with a graphical display of turnoff locations along the runway downrange. In the display, the existing turnoffs as well as the proposed turnoffs are shown. The new turnoffs are differentiated from the existing ones through color coding. This enables the user in noticing the location of new turnoffs along the runway downrange. An added feature in this screen is the presentation of exit numbers and their location in a tabular form.

On the same screen, depicted in Fig. 4.4, a small menu screen is provided with options to view the complete turnoff geometry or to compare the centerline geometries. For the latter option, the user needs to input the exit number(s). The user has the flexibility to select any combination of exit numbers for comparing the geometry of different turnoffs or may even select only one exit number to view the centerline geometry of a particular turnoff. Fig. 4.10 illustrates the output screen for the comparison of turnoff geometries. It should, however, be noted that all the requested turnoff geometries were started at a common point for comparison purposes. Also, as can be seen from Fig. 4.14, the user has the benefit to view the centerline turnoff geometry coordinates in a tabular form for each of the turnoffs. The user needs to press 'T' to view the table with coordinates of centerline turnoff geometry from the screen which displays the centerline turnoff geometry. If the user does not intend to view the coordinates, (s)he may press 'E' (for exit location) to return to the turnoff locations screen.

The user after returning to the previous screen, which depicts the small menu, may now view the centerline turnoff geometry or the complete turnoff geometry. The user may either

VIEW MENU

Press escape (Esc) key to return to OUTPUT MENU.

Use arrow (↑ or ↓) key to change your choice.

Press enter (↵) key if the choice is correct.

1)View ROT/REL Table

2)View Turnoff Locations And Their Geometries

3)View Aircraft Statistics

4)Go To Output Menu

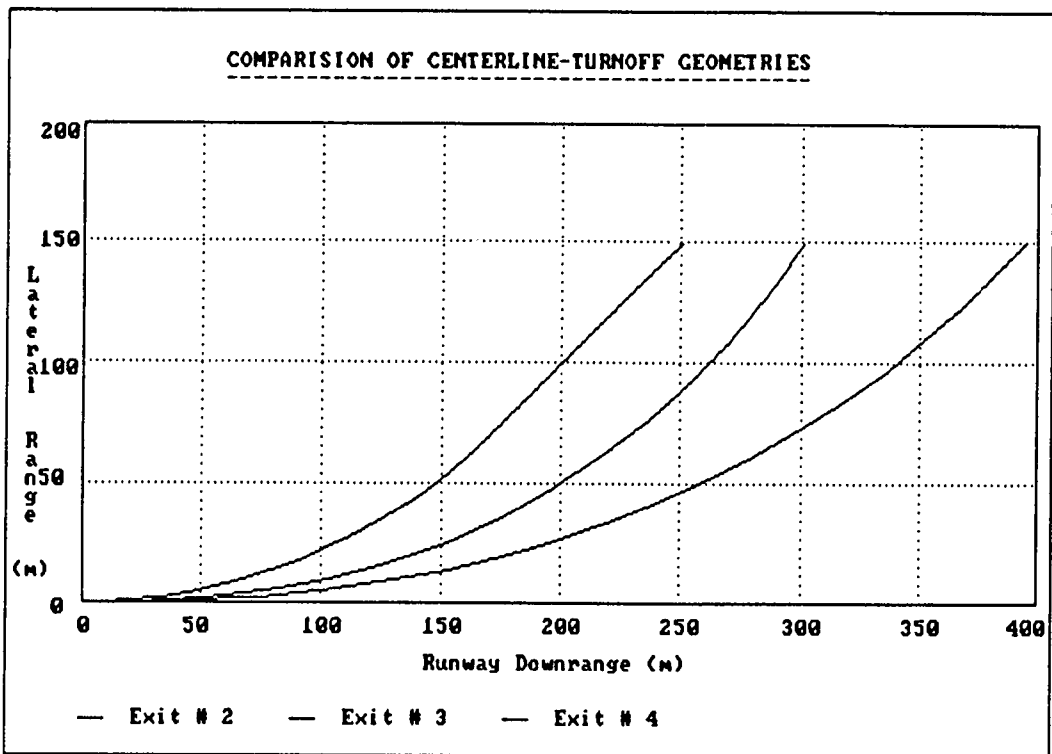
Figure 4.10 View Menu for Runmode '2' & '3'.

ROT / RELIABILITY TABLE
(This is for Designing a New Runway)

Exit # Location (m) Exit Type	1 1110.8 New	2 1964.2 New	3 2715.0 New
PA-38-112			
DRY ROT (12.5%)	46.88		
WET ROT (12.5%)	46.74		
BE-300			
DRY ROT (12.5%)	31.39		
WET ROT (12.5%)	30.46		
A-320-200			
DRY ROT (12.5%)		49.63	
WET ROT (12.5%)		45.98	
MD-11			
DRY ROT (12.5%)			67.36
WET ROT (12.5%)			60.87

ROT - Runway Occupancy Time in Secs
Reliability in % = 90
Average ROT = 47.41

Figure 4.11 ROT Table of Results.



Press any key to continue

Figure 4.12 Comparison of Centerline Turnoff Geometries.

select to view the complete turnoff geometry or may even again choose to view the centerline turnoff geometry for a different combination of turnoffs. When the user selects to view the complete turnoff geometry, the program requests the user to enter the exit number to be displayed. Here, the user needs to input only one exit number and not any combination of exit numbers. The program displays the complete turnoff geometry of that specific exit number entered by the user. For the exit number entered by the user, the program internally determines the type of turnoff geometry. The type of turnoff geometry can be either of the 30 degree or 45 degree or 90 degree standard FAA turnoff for the existing turnoffs or can be a new turnoff geometry developed by the model for the proposed new turnoff(s). The FAA standard 30 degree, 45 degree and 90 degree are best illustrated in Figs. 4.6, 4.7 and 4.8 respectively. The new turnoff geometry developed by the model for the proposed new turnoff is shown in Fig. 4.15. The user after viewing the complete turnoff geometry of the turnoff requested, may return back to the turnoff location and their geometries screen where (s)he may again enter a different turnoff to view its complete turnoff geometry.

When the user selects to view the standard turnoff geometry for a particular turnoff, the program automatically takes in the values of runway width, taxiway width, the distance between the runway and taxiway, the type of turnoff and other data supplied in the Input Module. In the standard FAA turnoff geometry display, the specifications are prominently shown, and the units are displayed in both English and Metric systems for the benefit of the user.

4.1.2.3 Aircraft Statistics and their ROT's

This screen displays the runway occupancy time of each of the aircraft in the form of a bar chart. At the top edge of each of the bars, the aircraft number is displayed. For each of the aircraft, selected by the user in the Input Module, the runway occupancy time is separately displayed for wet and dry runway surface conditions. For the benefit of user, the bars

representing the aircraft and its name are of the same color. On the bar chart, a straight line is drawn across the bars to portray the average ROT.

In each screen a maximum of six aircraft are shown, each with two bars, one for wet condition and the other for dry condition. It is necessary to press 'F' key (for forward) to view additional aircraft and 'B' key (for backward) to view the previous screen. In each of the screens, the average runway occupancy time which encompasses the whole population of aircraft selected by the user is shown. This includes dry and wet conditions at the airport. An option is provided to print the table by just pressing the 'P' key (for print). The user is also allowed to go back to the View Menu at any stage by entering 'V' key (for View Menu). The bar chart is best illustrated in Fig. 4.16.

4.1.2.4 Go to Output Menu

This option is provided to enable the user to go back to the Output Menu from the View Menu. The user may also exercise this option by pressing the escape 'Esc' key.

4.1.3 Design of a New Runway

This runmode deals with the design of a new runway facility. The user is requested to enter the relevant data in the Input Module for the design of a new runway facility. The user after entering the data in the Input Module is returned to the Main Module. The user at this point may go to the Output Menu to view the results. The Output Menu is as shown in Fig. 4.1. and the flow pattern corresponding to this runmode option is shown in Fig. 4.13 (attached at the end of the report). As the user selects to view the output from the Output Menu, he or she is presented with a new screen which displays the View Menu screen. This View Menu screen corresponds to the runmode option which is to design a new runway facility. Four

AIRPORT CONDITIONS

Temperature (C) : 15.00
Elevation (m) : 1500.0
Wind Speed (m/s) : 0.00

AIRCRAFT POPULATION

TERP A : 25.00
TERP B : 30.00
TERP C : 30.00
TERP D : 15.00

RUNWAY TURNOFF GEOMETRY COORDINATES FOR EXIT # 2

X	Y	X	Y	X	Y
27.500	0.009	97.306	4.723	165.897	18.416
44.994	0.508	114.620	7.259	182.696	23.316
52.472	1.377	131.838	10.385	199.313	28.801
79.916	2.769	148.937	14.104		

Press any key to continue

Figure 4.14 Table of Turnoff Geometry Coordinates.

options are provided in the View Menu. The options are (1) ROT Table of Results (2) Turnoff Location and their Geometries (3) Aircraft Statistics and their ROT's, and (4) Go to Output Menu. The View Menu screen is best illustrated in Fig. 4.10. In the following paragraphs, we will discuss in detail each of these options.

4.1.3.1 ROT Table of Results

This table displays the number of new turnoffs for the proposed new runway facility and their location. As none of the standard FAA turnoffs are proposed in the new runway, the type of turnoff geometry is described as a new turnoff geometry developed by the model. For each of the aircraft selected earlier in the Input Module by the user, the corresponding runway occupancy time associated with the appropriate turnoff is displayed. These values, for both wet and dry airport conditions, are displayed separately in the table. In addition to these, the relative frequency of occurrence of this aircraft, for a particular scenario (wet or dry conditions) is specified as a percentage of all the aircraft occurrences. The empty boxes in the table imply that the aircraft in question cannot negotiate that particular turnoff. The reliability associated with each aircraft and turnoff(s), which was earlier provided by the user in the Input Module, is displayed at the bottom of the table.

In each screen, as only four aircraft are shown in the table, it is necessary to press 'F' key (for forward) to view additional aircraft and 'B' key (for backward) to view the previous screen. In each of the screens the average ROT which encompasses the whole population of aircraft selected by the user is shown. This includes dry and wet conditions at the airport. An option to print the table by just pressing the 'P' key (for print) is also provided. The user is also allowed to go back to the View Menu at any stage by entering 'V' key (for View Menu). This table is illustrated in Fig. 4.11.

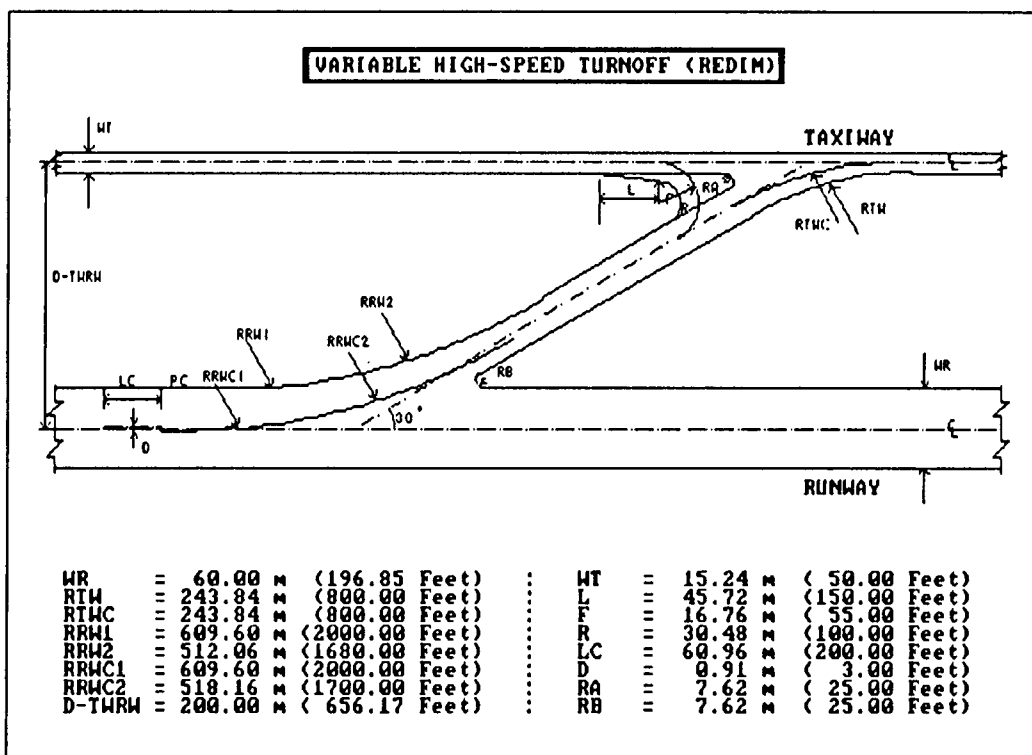


Figure 4.15 Variable Turnoff Geometry (Developed by REDIM).

**STATISTICS ON ROT
PER AIRCRAFT FOR DRY
AND WET CONDITIONS**

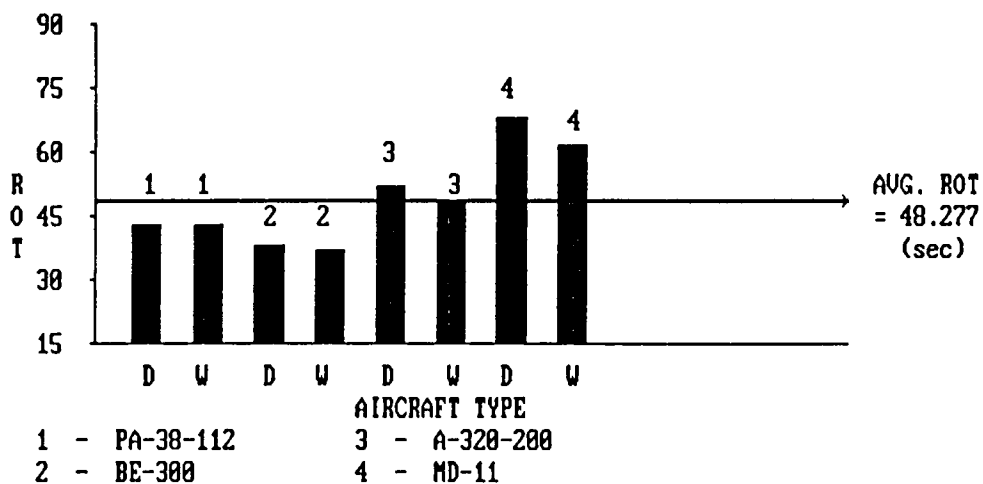
AIRPORT CONDITIONS

Temperature (C) : 15.00

Elevation (m) : 1500.0

Wind Speed (m/s) : 0.00

Weather (%) : Dry - 50 & Wet - 50



Press any key to continue.

Figure 4.16 Aircraft Statistics and their ROT's.

4.1.3.2 Turnoff Locations and their Geometries

This screen presents the user with a graphical display of optimal turnoff locations along the runway downrange. These optimally located turnoffs are all proposed by the model for the new runway facility. A table with the exit numbers and their location is also presented on the same screen. The output screen is shown in Fig. 4.4.

On the same screen, which is very well depicted in Fig. 4.4, a small menu screen is provided with options to view the complete turnoff geometry or to compare the centerline geometries. For the comparison of centerline geometries, the user needs to input the exit number(s). The user has the flexibility to select any combination of exit numbers for comparing the geometry of different turnoffs or may even select only one exit number to view the centerline geometry of a particular turnoff. Fig.4.10 illustrates the output screen for the comparison of turnoff geometries. It should, however, be noted that all the requested turnoff geometries were started at a common point for comparison purposes. Also, as can be seen from Fig. 4.14, the user has the benefit to view the centerline turnoff geometry coordinates in a tabular form for each of the turnoffs. The user needs to press 'T' to view the table with coordinates of centerline turnoff geometry from the screen which displays the centerline turnoff geometry. If the user does not intend to view the coordinates, (s)he may press 'E' (for exit location) to return to the turnoff locations screen.

The user after returning to the previous screen, which depicts the small menu, may now view the centerline turnoff geometry or the complete turnoff geometry. The user may either select to view the complete turnoff geometry or may even again choose to view the centerline turnoff geometry for a different combination of turnoffs. When the user selects to view the complete turnoff geometry, the program requests the user to enter the exit number to be displayed. Here, the user needs to input only one exit number and not any combination of exit numbers. The program displays the complete turnoff geometry of that specific exit number entered by the user. This high speed turnoff geometry is developed by the model for the proposed new turnoff(s) and is shown in Fig. 4.15. The user after viewing the complete

turnoff geometry of the turnoff requested, may return back to the turnoff location and their geometries screen where (s)he may again enter a different turnoff to view its complete turnoff geometry.

When the user selects to view the standard turnoff geometry for a particular turnoff, the program automatically takes in the values of runway width, taxiway width, the distance between the runway and taxiway, the type of turnoff and other data supplied in the Input Module.

4.1.3.3 Aircraft Statistics and their ROT's

This screen displays the runway occupancy time for each of the aircraft in the form of a bar chart. At the top edge of each of the bars, the aircraft number is displayed. Each aircraft is represented by two bars. The first one represents ROT for dry condition whereas the second one represents ROT for wet runway surface condition. For the benefit of user, the bars representing the aircraft and its name are of the same color. On the bar chart, a straight line is drawn across the bars to portray the weighted average ROT.

Each screen accommodates six aircraft, each with two bars, one for wet condition and the other for dry condition. The user needs to press 'F' key (for forward) to view additional aircraft and 'B' key (for backward) to view the previous screen. In each of the screens the user is presented with the average ROT, which encompasses the whole population of aircraft selected by the user. This includes dry and wet conditions at the airport. The user is provided with an option to print the table by just pressing the 'P' key (for print). An option is provided to go back to the View Menu at any stage by entering 'V' key (for View Menu). The bar chart is illustrated in Fig. 4.16.

4.1.3.4 Go to Output Menu

This option is provided to enable the user to go back to the Output Menu from the View Menu. The user may also exercise this option by pressing the escape 'Esc' key.

4.2 Print the Report

This option provides a hard-copy report which is a complete report of the "Runway Exit Design Interactive Model". The report is divided into two sections. The first one deals with the summary of input data where as the second one concentrates on the results of the analysis.

The input data summary is subdivided into five categories, as was done earlier in the Input Module. Category (1) 'Analysis Type and Existing Exits' gives the type of analysis selected and the number of exits, their type and their location. Category (2) 'Aircraft Mix and Characteristics' provides a table with the names of the aircraft selected and their characteristics. The characteristics of the aircraft include: wheelbase, wheeltrack, landing mass, wing area, landing run distance, load on main gear, distance of nose gear to wingtip, and the maximum clearance distance. Category (3) 'Operational Data' provides free roll times, taxiing speed and their standard deviations, and the safety factor for skid. Category (4) 'Environmental Data' provides the wind speed, wind direction, airport elevation, temperature, runway visual range, runway orientation, runway width and the distance to the runway. Category (5) 'Runway Gradients' provides the runway length and the gradients for every one-tenth of the runway length specified. Category (6) 'Weather and Exit Speeds' provides the weather conditions (probability of dry and wet condition) in percentage and the speeds for each of the TERP categories, both for dry and wet conditions.

In the results of the analysis section, the weighted average runway occupancy time (ROT) and a table with the number of exits, their location and their type is provided. For runmode '1', ie., for the analysis of an existing runway, the reliability associated with each of the aircraft and exit, for both dry and wet runway surface conditions is provided along with the corresponding ROT's. For runmodes '2' and '3', ie., for the improvement of an existing runway or for designing a new runway, the reliability associated with all the turnoffs is given separately. The table also provides the turnoff assignment to each of the aircraft. Another portion of the results pertaining to runmodes '2' and '3', provides the coordinates of the centerline for each of the turnoffs.

4.3 Help

The main prupose of this option is to help the user by explaining each of the options that are made available in the Output Module. The help screen explains briefly the first option, 'View the Output' which displays the output on the screen and the second option, 'Print the Report', which gives the hard-copy of the complete report. The user may press any key to exit from this screen and return to the 'Output Menu' screen.

'Go to Main Menu' is the last option that is provided in the Output Menu. This allows the user to go back to the Main Menu from the Output Menu. The user may also exercise this option by pressing the escape 'Esc' key.

5.0 RESULTS AND FURTHER RESEARCH

High-speed turnoffs are gaining more attention due to the fact that they could significantly reduce the runway occupancy time, thereby boosting the airport capacity. It is preferable to maintain the exit angle (of the high speed exit taxiway) below 30-degree especially if speeds of 60 mph or more are to be involved in the design. This reduces the length of travel on a curve which in turn requires less concentration on the part of the pilot. As the exit angle is less, the sight distance is also improved [Horonjeff et al, 1958].

Test were conducted by Horonjeff et al [Horonjeff et al, 1958] to develop taxiways which would meet the following requirements for speeds upward of 30 mph: (1) be safe under conditions of intended use; (2) be comfortable for passengers; (3) be relatively easy to maneuver by pilots; and (4) not result in excessive tire wear or stresses in the aircraft structure.

In order to find a proper relationship between speed, radius of curvature, and the general configuration of the taxiway, tests were conducted on wet and dry concrete and asphalt pavement with various types of civil and military aircraft. As the plots of typical test runs were evaluated, a conclusion was reached that the path of the aircraft were best approximated by two- or three-centered compound curves, though the actual curves were probably circular arcs with spirals at each end. In fact, as the plots were studied further, it indicated that it would be sufficiently accurate to approximate the path by two circular arcs. Moreover, it was found that at high speeds a compound curve is necessary to minimize tire wear on the nose gear. The 'entrance curve' is the circular arc developed by the aircraft in the early portion

of the turn with radius as R_1 and the 'central curve' is the tighter circular arc developed in the remainder of the turn with radius as R_2 .

Horonjeff et al have suggested that a compound curve be used as it is easy to establish in the field and begins to approach the shape of a spiral. The most significant factor affecting the turning radius was specified as speed, not the total angle of turn nor passenger comfort.

A relation between the turning radius (i.e., the central or main curve) and the speed was given by the formula, $R_2 = V^2 / 15 F$ where V is the velocity in miles per hour, and F is made equal to 0.13 for a smooth, comfortable turn on a wet or dry pavement.

The entrance curve with large-radius curve R_1 is necessary to provide a gradual transition from a straight ahead or tangent direction to a curved path. The length of the transition curve can be roughly approximated by the relation, $L_1 = V^3 / C R_2$ where V is in feet per second and R_2 is in feet. C was found experimentally to be on the order of 1.3 [Horonjeff et al, 1958]. Fig. 5.1 depicts a chart showing the relationship of exit speed to radius R_1 and R_2 and the length of the transition curve L_1 .

The Runway Exit Design Interactive Model REDIM, uses two radii to approximate the radius of curvature of the high speed exit turnoff. The length of transition for the entrance curve is obtained by using the same relation suggested by Horonjeff et al [Horonjeff et al, 1958]. As the critical aircraft for each exit taxiway is known, the aircraft design group, shown in Table 2.2, can be determined. Since the aircraft design group is obtained, the taxiway exit geometry recommended by FAA can also be determined. The high-speed exit taxiway, developed by REDIM is shown in Fig. 4.16. This variable high-speed exit taxiway is very similar to the standard 30-degree high-speed exit taxiway developed by FAA. The most significant difference between these high speed exit taxiways is that the former uses two radii for the radius of curvature whereas the latter uses only one. The exit angle for the variable high-speed turnoff developed by REDIM can be varied to allow specific airport constraints to be incorporated in the design. This gives room for sensitivity analysis on the geometry of the exit taxiway. The FAA has recommended that for speeds of the order of 60 mph, the radius of exit curve should be 1800 ft. However, it was found that pilots were reluctant to exit turnoffs

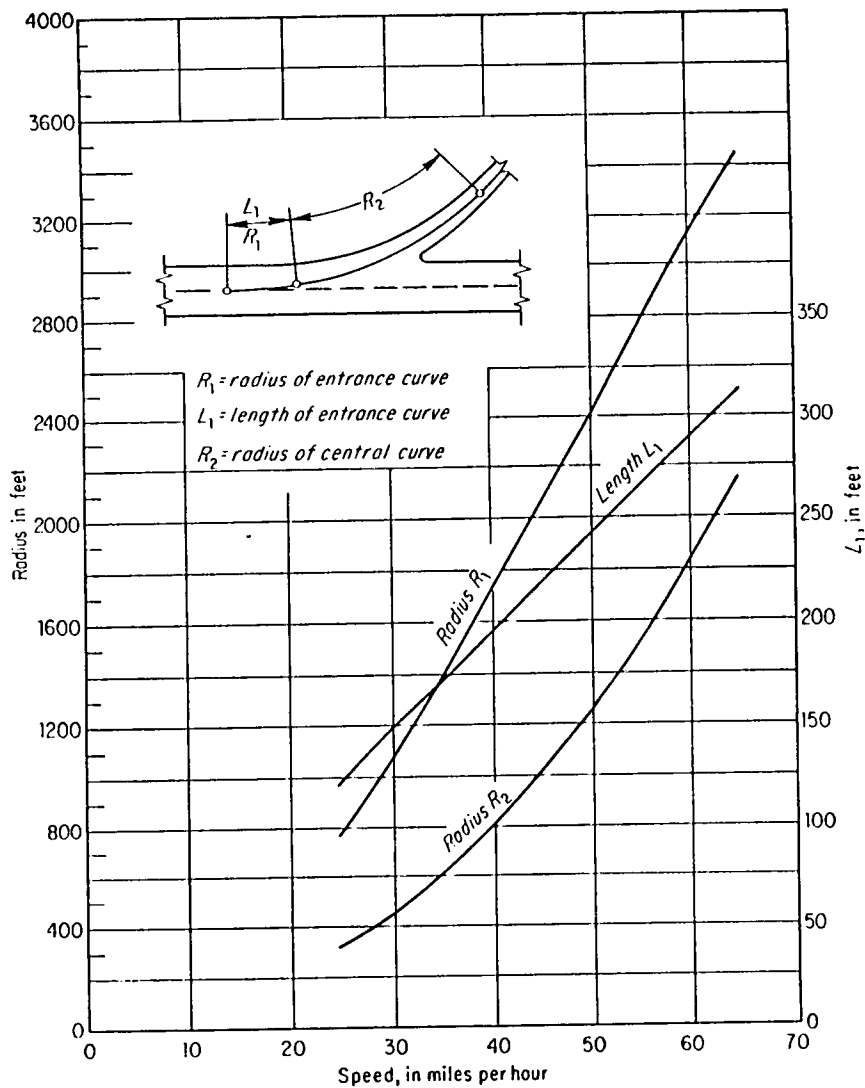


Figure 5.1 Radius of curvature and entrance curves for taxiways.
 Source : Horonjeff et al, 1958.

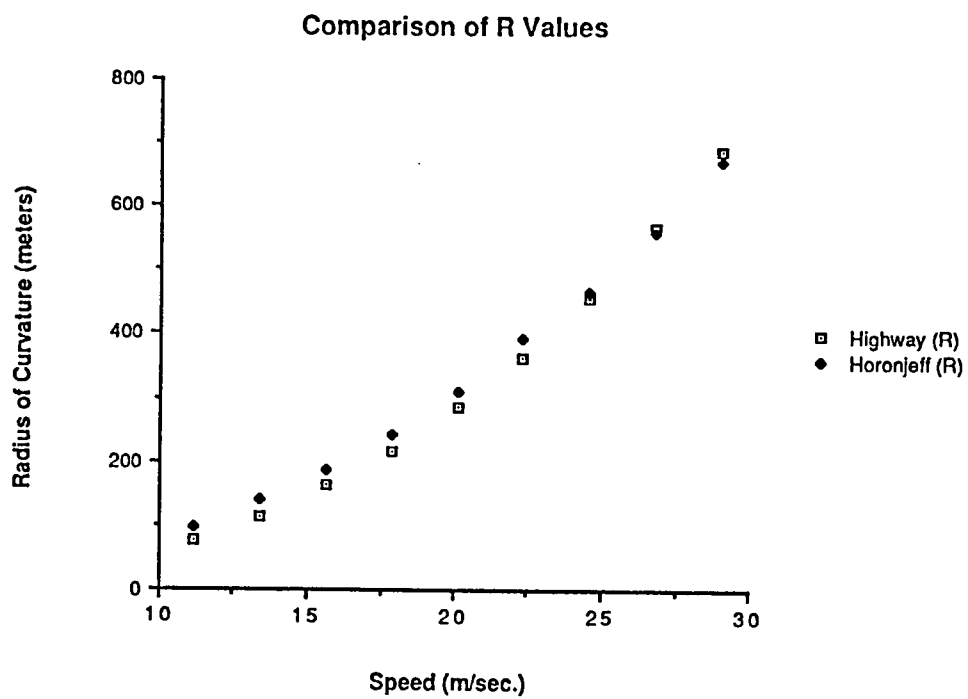


Figure 5.2 Comparison of Radius of Curvature.

at those speeds. Moreover, speeds used by FAA for that radius of curvature were about 45 mph.

A comparison of radius of curvatures given by Horonjeff et al and the Highway formula, for various speeds is shown in Fig. 5.2. It can be observed that the radius of curvature proposed by Horonjeff approximates very closely with the highway radius of curvature. The radius of curvature shown here on Fig. 5.2 forms the central or main curve of the exit taxiway.

A sensitivity analysis was performed to find out the behavior of turnoff time with the change in exit angle. The analysis was done for five different speeds. Fig. 5.3 shows the analysis of Boeing 747-400 aircraft for a fifty percent safety factor under wet pavement conditions. A careful study of these plots indicate that for exit angles greater than 30-degrees there is no significant change in turnoff times. The result was observed to be the same for all exit speeds. However, at very low speed (15 m/s) and exit angle (10-degree), the turnoff time was observed to be close to 50 seconds. This explains the fact that lower angles require more distance for the aircraft wing tip to cross the clearance point. A closer look at the plots indicate that for higher exit speeds and for exit angles greater than 20, the change in turnoff times is very marginal.

Similar results were observed when sensitivity analysis was performed for a Boeing 727-200 aircraft for a fifty percent skid safety factor under wet pavement conditions. This is shown in Fig. 5.4. As expected, the turnoff times were low when compared with the Boeing 747-400 aircraft.

A notable feature in the Runway Exit Design Interactive Model REDIM, is that it accounts for the behavior of a large aircraft overtime with respect to the entrance curve $R_{sub 1}$ and the central curve $R_{sub 2}$. The model also takes care of the fact that small aircraft whose stalling speed is approximately equal to 25 m/s, use exit speeds below 20 m/s.

The runway occupancy times for various aircraft using the REDIM model were compared with the ones predicted by Barrier & Diehl [Barrier & Diehl, 1988] shown in Table 1.1. They were found to be satisfactory. However, much research is to be done to find out the behavior of the exit taxiway at the latter half of the exit taxiway. Evaluations are to be done whether a

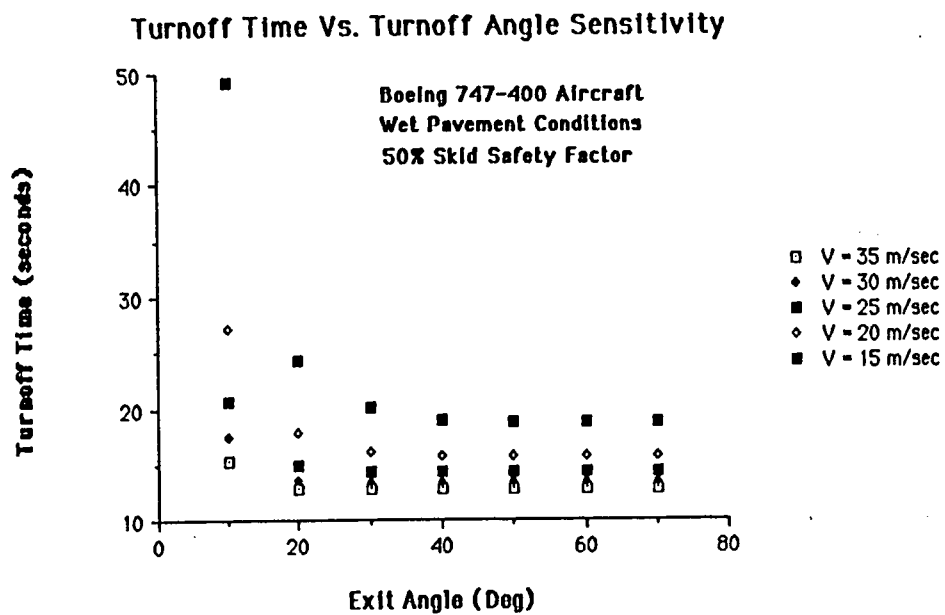


Figure 5.3 Turnoff Time Vs. Turnoff Angle Sensitivity (Boeing 747-400 Aircraft).

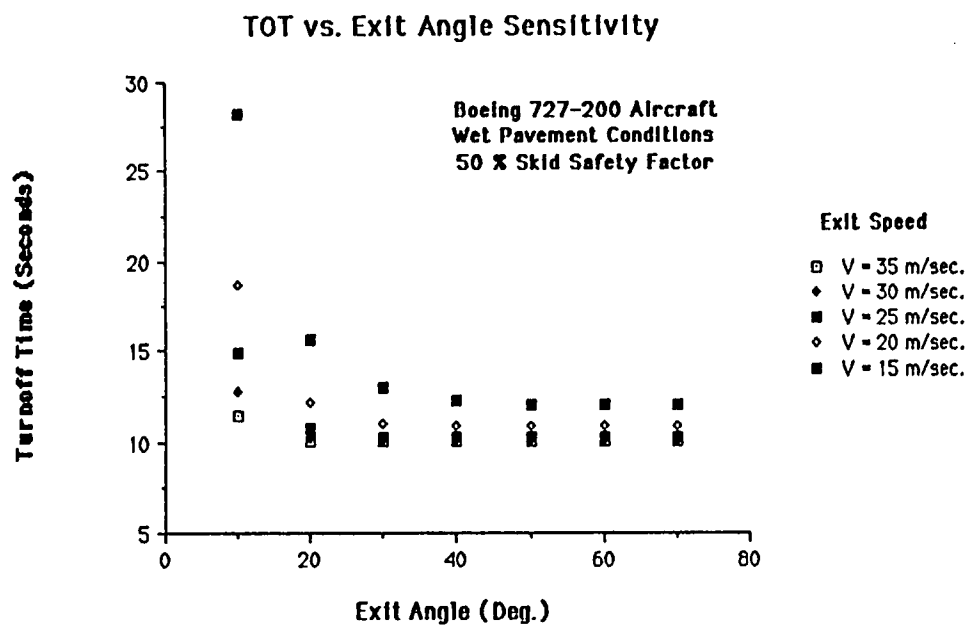


Figure 5.4 Turnoff Time Vs. Turnoff Angle Sensitivity (Boeing 727-200 Aircraft).

reverse curve would be appropriate or any other form at the intersection of the exit taxiway with the taxiway.

Bibliography

1. Air Transportation Association of America, *Air Transport Facts and Figures*, Washington, D.C., annual.
2. Ashford, N. and Wright, P.H., *Airport Engineering*, John Wiley & Sons, Inc., New York, 1979.
3. *Aviation Week and Space Technology*, McGraw-Hill Publishing Company, various issues 1986-1989.
4. Barrer, J.N. and Deilh, J.M., *Toward a Goal-Oriented Plan for Identifying Technology to Increase Airfield Capacity*, The MITRE Corporation, Mclean, Virginia, 1988.
5. Branstetter, J.R., Houck, J.A., and Guenther, A.D., *Flight Simulation of a Wide-Body Transport Aircraft to Evaluate System Plan*, Journal of Aircraft, Volume 25, No. 6, June 1988.
6. Civil Aeronautics Board, *Report on Airline Service, Fares, Traffic, Load Factors, and Market Share*, a staff study, 14th in a series, Washington, D.C., 1981.
7. Department of Commerce, *National Travel Survey, 1977 Census of Transportation*, Washington, D.C., October, 1979.
8. Federal Aviation Administration, *Airport Capacity Enhancement Plan*, DOT / FAA / CP-87-3, Washington, D.C., 1987.
9. Federal Aviation Administration, *Airport Design*, DOT / FAA, AC No: 150 / 5300-00, Washington, D.C., October, 1988.
10. Federal Aviation Administration, *Airport Design Standards - Airports served by Air Carriers - Taxiways*, DOT / FAA AC No: 150 / 5335-1A, Washington, D.C., May 1970.
11. Federal Aviation Administration, *Airport Design Standards - Transport Airports*, AC No: 150 / 5300-12, U.S. DOT, Washington, D.C., August, 1982.
12. Federal Aviation Administration, *FAA Aviation Forecasts, Fiscal Years 1987-1988*. Report FAA-APO-87-1. U.S. DOT, February, 1987.
13. Federal Aviation Administration, *Utility Airports - Air Access to National Transportation*, AC No: 150 / 5300-4B, U.S. DOT, Washington, D.C., January, 1981. Also Draft change 6, 1982.
14. Federal Aviation Administration, *United States Standard for Terminal Instrument Operations (TERPS)*, 3rd Edition, U.S. DOT, Washington, D.C., July, 1976.
15. Hauer E., *Determination of Wheel Trajectories*, Transportation Engineering Journal, November, 1970.
16. Haury, R.L., *Fast Exit*, Civil Engineering, October, 1987.
17. Hobioka, A.G., *Optimal Location of Runway Turnoffs in an Airport Environment*, Research Proposal submitted to NASA / Langley, June, 1988.
18. Horonjeff, R., Finch, D.M., Belmont, D.M., and Ahlborn, G., *Exit Taxiway Location and Design*, Institute of Transportation and Traffic Engineering, University of California, Berkeley, August 1958.

19. Horonjeff, R., Grassi, R.C., Read, R.R., and Ahlborn, G., *A Mathematical Model for Locating Exit Taxiways*, Institute of Transportation and Traffic Engineering, University of California, Berkeley, December, 1959.
20. Horonjeff, R., Read, R.R., and Ahlborn, G., *Exit Taxiway Locations*, Institute of Transportation and Traffic Engineering, University of California, Berkeley, September, 1960.
21. Horonjeff, R. and Mckelvey, F.X., *Planning and Design of Airports*, 3rd Edition, McGraw-Hill Book Company, New York, 1983.
22. Joline, E.S., *Optimization of Runway Exit Configurations*, Transportation Engineering Journal, February, 1974.
23. Lebron, J.E., *Estimates of Potential Increases in Airport Capacity Through ATC System Improvements in the Airport and Terminal Areas*, The MITRE Corporation, McLean, Virginia, FAA-DL5-87-1 (MTR-87W203), October 1987.
24. Pracht, W.E., and Courtney, J.F., *The Effects of an Interactive Graphics-Based DSS to Support Problem Structuring*, Decision Sciences, Vol. 19., 1988.
25. Rossow, V.J, and Tinlin, B.E., *Research on Aircraft/Vortex-Wake Interactions to Determine Acceptable Level of Wake Intensity*, Journal of Aircraft, Volume 25, No. 6, June 1988.
26. Schoen, M.L. et al, *Probabilistic Computer Model of Optimal Runway Turnoffs*, NASA Contractor Report 172549, April 1985.
27. Simpson, R.W., Odoni, A.R. and Salas-Roche, F., *Potential impacts of Advanced Technologies on the ATC Capacity of High-Density Terminal Areas*, NASA Contractor Report 4024, October 1986.
28. Transportation Research Board, *Future Development of the U.S. Airport Network*, National Research Council, Washington, D.C., 1988.
29. Van Aswegen, J.W.L., *Calculation of Aircraft Wheel Paths and Taxiway Fillets*, The Institute of Transportation and Traffic Engineering, University of California, July 1973.

Appendix A. PROGRAM CODE

```
DECLARE SUB geo.var ()
DECLARE SUB air.design.grp (ws!)
DECLARE SUB prt ()
DECLARE SUB fc.inkey (wd$)
DECLARE SUB geo.new.30 ()
DECLARE SUB tax.exit.geo (des.grp!)
DECLARE SUB font (x!, y!, z%, w$)
DECLARE SUB box.43 ()
DECLARE SUB box ()
DECLARE SUB guide.mm ()
DECLARE SUB fc.inkey (wd$)
DECLARE SUB cvex (st$, nn!, ann!())
DECLARE SUB box.small2 ()
DECLARE SUB clr.dec (i!, jk!)
DECLARE SUB cal.track (kex!, ncrit!, trackmax!)
DECLARE SUB track (sai!, vpath!, xpath!, ypath!, dtl, trackmax!, xmrl!, ymrl!, xml!, yml!, xmpath!(), ympath!(), xnpath!(), ynpath!(), dx, dy)
DECLARE SUB skid (ic!, iw!, vpath!, ymumax!)
DECLARE SUB geo.new.30 ()
DECLARE SUB geo.new.45 ()
DECLARE SUB geo.new.90 ()
DECLARE SUB cal.cline (kex!, npnt!(), xtra!(), ytra!())
DECLARE SUB prt.out (rotrun!(), relrun!())
DECLARE SUB ccline.run2 (ex.com!, ex.cen!(), xtra!(), ytra!())
DECLARE SUB std.cline (kex!, extp!, npnt!(), xtra!(), ytra!())
DECLARE SUB pexl.run2 ()
DECLARE SUB exloc.ttl ()
DECLARE SUB stat.bar2 (rotrun!(), relrun!())
DECLARE SUB stat.bar3 (rotrun!(), relrun!())
DECLARE SUB table.ab.2 (rotrun!(), relrun!())
DECLARE SUB view.menu2 (rotrun!(), relrun!())
DECLARE SUB plot.cen (ex.com!, npnt!(), xtra!(), ytra!())
DECLARE SUB ccline.run3 (ex.com!, ex.cen!(), xtra!(), ytra!())
DECLARE SUB table.ab.3 (rotrun!(), relrun!())
DECLARE SUB table.ab.1 (rotrun!(), relrun!())
DECLARE SUB view.menu3 (rotrun!(), relrun!())
DECLARE SUB view.menu1 (rotrun!(), relrun!())
DECLARE SUB menu.out (rotrun!(), relrun!())
DECLARE SUB sort.exl ()
DECLARE SUB cl.outop ()
DECLARE SUB scrub (rapath!, acfmass!(), ymuscl!, nl)
DECLARE SUB EXSTEER (radius!, vel!, heading!, headdot!, ypath!, dist!!)
DECLARE SUB pexl.run3 ()
DECLARE SUB cl.bot.sbox ()
DECLARE SUB ask.cen (icen!)
DECLARE SUB pexl.run1 ()
DECLARE SUB box.s43 ()
DECLARE SUB ask.exnum.run1 (nex.fix!)
DECLARE SUB trans.ind (w.line$())
DECLARE SUB table.ab.23 ()
DECLARE SUB guide.ed.out.opt ()
DECLARE SUB menu.out.opt ()
DECLARE SUB table.ab ()
DECLARE SUB table.a ()
DECLARE SUB rundat (nmix!, lwetrun!, iexrun!)
DECLARE SUB box.43 ()
DECLARE SUB stat.bar.ttl ()
DECLARE SUB cl.outoprpt ()
DECLARE SUB clr.outoprpt ()
DECLARE SUB omsg.airpomix ()
DECLARE SUB omsg.airchar ()
DECLARE SUB omsg.aircon ()
DECLARE SUB omsg.airpop ()
DECLARE SUB clr.dec (i!, jk!)
DECLARE SUB ask.extran (exnum!, exarr!())
DECLARE SUB outcase2 ()
DECLARE SUB locexit (nexit!, xea!())
DECLARE SUB tbl.exloc (nexit!, xea!())
DECLARE SUB ask.tbl (q$)
DECLARE SUB cl.sbox2 ()
```

```

DECLARE SUB frame.rr (nexit, xnam$, ynam$)
DECLARE SUB cl.rhs ()
DECLARE SUB outcase1 ()
DECLARE SUB ask.exn (exnum!, exarr!())
DECLARE SUB cvex (st$, nn!, ann!())
DECLARE SUB tbl.turnoff (ntr!, xtra(), ytra(), exnum, exarr())
DECLARE SUB turnoff (xturnmax)
DECLARE SUB tbl.rore (ndata!, RA!(), RB!(), exnum, exarr())
DECLARE SUB turn (xtra1!(), ytra1!(), xturnmax, jk)
DECLARE SUB rrplot (ndata!, xa!(), ya!(), i)
DECLARE SUB tbl.1 (nexit!, aa!(), bb!())
DECLARE SUB xyplot (ndata!, xa!(), ya!(), xnam$, ynam$)
DECLARE SUB box.small2 ()
DECLARE SUB box ()
DECLARE SUB stitle (W$)
DECLARE SUB box.small ()
DECLARE SUB guide.mm ()
DECLARE SUB print.str (W$)
DECLARE SUB get.ival (iab!)
DECLARE SUB fc.inkey (wd$)
DECLARE SUB guide.ed ()
DECLARE SUB guide.mm ()
DECLARE SUB friction (ymumax!, gcf!, fricon!)
DECLARE SUB scrub (rapath!, acfmass!(), ymuscl, n!)
DECLARE SUB EXSTEER (radius!, vel!, heading!, headdot!, ypath!, distt!)
DECLARE SUB plot.cen (ex.com!, npnt!(), xtra!(), ytra!())
DECLARE SUB std.cline (kex!, extp!, npnt!(), xtra!(), ytra!())
DECLARE SUB cvex (st$, nn!, ann!())

'-----
' INPUT DATA
'-----

DIM SHARED des.grp
DIM SHARED tw, R, L, F
DIM nexit(2), exloc(9), exctype(9), rungrad(10)
DIM ncat(5), acfmix(5, 20)
DIM p.wth(2), vexit(5, 2), v.std(3)
DIM acflist$(25), mixlist(25), catlist(25)
DIM acfnwt(25), acfmass(25), acfclmax(25), acfwa(25), acfwb(25)
DIM acflm(25), acflrun(25), acfws(25), acfwt(25), yclear(25)
DIM SHARED catprop(5)
DIM SHARED avg.rot, crit.acf(10), exloc.new(10), lex.all(25, 2)
DIM SHARED exloc.old(10), exctype.old(10), msum
DIM SHARED ex.cen(10), ex.spd(10), radius(10, 2)
DIM SHARED ia.s(10), ia.e(10)
DIM nexit(2), exloc(9), exctype(9), rungrad(10)
DIM ncat(5), acfmix(5, 20)
DIM p.wth(2), vexit(5, 2), v.std(3)
DIM acflist$(25), mixlist(25), catlist(25)
DIM acfnwt(25), acfmass(25), acfclmax(25), acfwa(25), acfwb(25)
DIM acflm(25), acflrun(25), acfws(25), acfwt(25), yclear(25)
DIM SHARED tw, R, F, L
DIM nexit(2), exloc(9), exctype(9), rungrad(10)
DIM ncat(5), acfmix(5, 20)
DIM p.wth(2), vexit(5, 2), v.std(3)
DIM acflist$(25), mixlist(25), catlist(25)
DIM acfmaxa(25), acfmass(25), acfclmax(25), acfwa(25), acfwb(25)
DIM acflm(25), acflrun(25), acfws(25)

COMMON SHARED /id1/ runmode, rel, nexit(), exloc(), exctype()
COMMON SHARED /id2/ tfr1, tfr1sd, tfr2, tfr2sd, vtaxi, vtaxisd, sfskid
COMMON SHARED /id3/ awsp, awdir, airelv, airtem, rvr, runor, runwid, distt
COMMON SHARED /id4/ runlen, fricon, rungrad()
COMMON SHARED /id5/ nmix, acfmix(), ncat()
COMMON SHARED /id6/ acflist$(), mixlist(), catlist()
COMMON SHARED /id7/ p.wth(), vexit(), v.std()
COMMON SHARED /id8/ acfnwt(), acfmass(), acfclmax(), acfwa()
COMMON SHARED /id9/ acfwb(), acflm(), acflrun(), acfws(), acfwt()
COMMON SHARED /id10/ yclear()
COMMON SHARED /fname/ dfile$, ofile$
COMMON SHARED /clr/ bcl, ccl, lcl, tcl, cscl
COMMON SHARED /iil/ il, ih, ichoice, mode, nrec, skip, a.flag
COMMON SHARED /id1/ runmode, rel, nexit(), exloc(), exctype()
COMMON SHARED /id2/ tfr1, tfr1sd, tfr2, tfr2sd, vtaxi, vtaxisd, sfskid
COMMON SHARED /id3/ awsp, awdir, airelv, airtem, rvr, runor, runwid, disttw
COMMON SHARED /id4/ runlen, fricon, rungrad()
COMMON SHARED /id5/ nmix, acfmix(), ncat()
COMMON SHARED /id6/ acflist$(), mixlist(), catlist()
COMMON SHARED /id7/ p.wth(), vexit(), v.std()

```

```

COMMON SHARED /id8/ acfnwt(), acfmass(), acfclmax(), acfwa()
COMMON SHARED /id9/ acfwb(), acflm(), acflrun(), acfws(), acfwt()
COMMON SHARED /id10/ yclear()
COMMON SHARED /id1/ runmode, rel, nexit(), exloc(), extype()
COMMON SHARED /id2/ tfr1, tfr1sd, tfr2, tfr2sd, vtaxi, vtaxisd, hthres
COMMON SHARED /id3/ awsp, awdir, alrelv, alrtem, rvr, runor, runwid, disttw
COMMON SHARED /id4/ runlen, fricon, rungrad()
COMMON SHARED /id5/ nnix, acfmix(), ncat()
COMMON SHARED /id6/ acflist$(), mixlist(), catlist()
COMMON SHARED /id7/ p.wth(), vexit(), v.std()
COMMON SHARED /id8/ acfmaxa(), acfmass(), acfclmax(), acfwa()
COMMON SHARED /id9/ acfwb(), acflm(), acflrun(), acfws()

```

END

EACH SUBROUTINE IS EXPLAINED BRIEFLY AFTER THE CODE. EXPLANATIONS WITHIN THE SUBROUTINE ARE ALSO PROVIDED.

SUB air.design.grp (ws)

‘ this is for assigning an aircraft for a design group depending
‘ on its wingspan

mts = .3048 ‘ for conversion from feet to meters

```

IF ws < 48.999 * mts THEN
  des.grp = 1
ELSEIF ws > 49 * mts AND ws < 78.999 * mts THEN
  des.grp = 2
ELSEIF ws > 79 * mts AND ws < 117.999 * mts THEN
  des.grp = 3
ELSEIF ws > 118 * mts AND ws < 170.999 * mts THEN
  des.grp = 4
ELSEIF ws > 171 * mts AND ws < 196.999 * mts THEN
  des.grp = 5
ELSEIF ws > 197 * mts AND ws < 261.999 * mts THEN
  des.grp = 6
END IF

```

END SUB

The main purpose of this subroutine is to segregate the aircraft based on the wing span. The aircraft are segregated into six design groups. The input to this subroutine is ws, the wingspan which is given in meters.

SUB ask.cen (icen)

‘ This draws a box at the lower right hand side of the screen

```

LOCATE 15, 49
PRINT CHR$(218); STRING$(26, 196); CHR$(191);
FOR i = 16 TO 21
  LOCATE i, 49
  PRINT CHR$(179);
  LOCATE i, 76
  PRINT CHR$(179);
NEXT i
LOCATE 22, 49
PRINT CHR$(192); STRING$(26, 196); CHR$(217);

```

‘ This prints the message in the box

```

COLOR 14
LOCATE 16, 51: PRINT "Choose : "
COLOR 15
LOCATE 17, 51: PRINT "0) To Exit";
LOCATE 18, 51: PRINT "1) To Compare Centerlines"
LOCATE 19, 51: PRINT "2) To View a Geometry"
LOCATE 20, 51: PRINT "Example) 2"
LOCATE 21, 51, 1: LINE INPUT " = = = > ", exstr$

```

icen = VAL(LEFT\$(exstr\$, 1))

END SUB

This subroutine draws a small box at the lower right hand side of the screen and provides three options. The input to this subroutine is obtained from the user in the program. The output of this subroutine is 'icen', a string which takes in the left most value (one digit only).

SUB ask.exnum.run1 (nex.fix)

' This draws a box at the lower right hand side of the screen

```
LOCATE 15, 49
PRINT CHR$(218); STRING$(26, 196); CHR$(191);
FOR I = 16 TO 21
LOCATE I, 49
PRINT CHR$(179);
LOCATE I, 76
PRINT CHR$(179);
NEXT I
LOCATE 22, 49
PRINT CHR$(192); STRING$(26, 196); CHR$(217);
```

' This prints the message in the box

```
LOCATE 16, 51: PRINT "Enter the Exit Number"
LOCATE 17, 51: PRINT "to view the complete"
LOCATE 18, 51: PRINT "Turnoff Geometry."
LOCATE 19, 51: PRINT "(Enter '0' to exit)";
LOCATE 20, 51: PRINT "Example) 2"
LOCATE 21, 51: LINE INPUT " = = = > ", exstr$
```

nex.fix = VAL(LEFT\$(exstr\$, 1))

END SUB

This subroutine displays a box with a request to enter the exit number at the prompt. The input to this subroutine is obtained from the user and the output of this subroutine 'nex.fix' is a string which takes in the left most value.

SUB ask.extran (exnum, exarr())

' This draws a box at the lower right hand side of the screen

```
LOCATE 15, 49
PRINT CHR$(218); STRING$(26, 196); CHR$(191);
FOR I = 16 TO 21
LOCATE I, 49
PRINT CHR$(179);
LOCATE I, 76
PRINT CHR$(179);
NEXT I
LOCATE 22, 49
PRINT CHR$(192); STRING$(26, 196); CHR$(217);
```

' This prints the message in the box

```
LOCATE 16, 51: PRINT "Enter the Exit Number(s)"
LOCATE 17, 51: PRINT "to view the"
LOCATE 18, 51: PRINT "Centerline Geometry."
LOCATE 19, 51: PRINT "Example) 1,3"
LOCATE 21, 51: LINE INPUT " = = = > ", exstr$
```

CALL cvex(exstr\$, exnum, exarr())

END SUB

The main purpose of this subroutine is to draw a box with a request to enter the exit numbers to view the centerline turnoff geometries. The arguments which are sent as output are passed to the subroutine cvex.

SUB ask.tbl (q\$)

' This print the message in the last row for the user
' to either view the Table or go back to the Output Menu

```
COLOR 10, bcl
LOCATE 25, 2
PRINT "Press ' ' to view the values of the curve(s) or ' ' to return to VIEW MENU";
COLOR 12, bcl
LOCATE 25, 9: PRINT "T";
LOCATE 25, 53: PRINT "V";
COLOR lcl, bcl
```

1 CALL fc.inkey(q\$)

```

IF (q$ = "t") OR (q$ = "T") OR (q$ = "v") OR (q$ = "V") THEN
  LOCATE 25, 2: PRINT STRING$(79, 32);
  EXIT SUB
ELSE
  GOTO 1
END IF
END SUB

```

This subroutine prints a message at the last row of the screen giving the user an option to either view the table with the coordinates for the corresponding curve or to exit that particular screen and return to the output menu screen. This subroutine makes certain that it responds only to either 'T' or 'O', for both upper and lower cases.

```

DEFSTR W
'-----
SUB box.small2 STATIC

' This creates a small box used in output module
' (which is slightly bigger than the one used in the input module)

```

```

LOCATE 7, 2
PRINT CHR$(218); STRING$(76, 196); CHR$(191);
FOR I = 8 TO 22
  LOCATE I, 2
  PRINT CHR$(179);
  LOCATE I, 79
  PRINT CHR$(179);
NEXT I
LOCATE 23, 2
PRINT CHR$(192); STRING$(76, 196); CHR$(217);

```

END SUB

The main purpose of this subroutine is to make a box large enough to accommodate most of the output screens, but small enough to fit in the main box. This subroutine also makes certain that enough space is allotted outside and above it to display any relevant data corresponding to that particular screen.

```

DEFSNG W
SUB cal.cline (kex, npnt(), xtra(), ytra())

' This subroutine calculates the centerline coordinates
' of each of the turnoffs

gmax = 10000!      ' Maximum lateral Acceleration (g's)
jmax = 10000!      ' Maximum allowable jerk (g/sec.)
g = 9.81

```

```

IF (ex.type(ex.cen(kex)) = 0) OR (runmode = 3) THEN

  npnt(kex) = 0
  '-----
  n = INT((crit.acf(ex.cen(kex)) - .1) / 2) + 1      'I
  twet1 = INT(crit.acf(ex.cen(kex)) / 2)
  twet2 = crit.acf(ex.cen(kex)) / 2
  IF (twet1 = twet2) THEN
    iwet = 2
  ELSE
    iwet = 1
  END IF
  acfyaw = 10 * (1.7215 * LOG(acfmass(n)) / LOG(10!) - 1.673)
  acfymu = 0!

  REM exit path deterministic computations      'I
  h = .01
  count = 0
  prt = 100

  REM initialize state variables
  xpath = 0!
  ypath = 0!
  vpath = ex.spd(ex.cen(kex))
  sal = 0!
  tpath1 = 0!
  rapath = 200 + (4000 / 40) * vpath

  CALL skid(cattlist(n), iwet, vpath, ymumax)

  REM estimate rate variables
  rdotp1 = acfmass(n) * acfwb(n)

```

```

rdotp0 = (acflm(n) / 100!) * (1! - acflm(n) / 100!)
froll = .03
saimax = 1!

235
IF (vpath > vtaxi) THEN
  acpath = (-1) * g * froll      'free roll deceleration
ELSE
  acpath = 0
END IF

CALL EXSTEER(rapath, vpath, sai, saidot, ypath, disttw)

REM EXSTEER chooses among several exit steering laws to execute a high-speed turn

CALL scrub(rapath, acfmass(), ymusc, n)

ymuc = vpath ^ 2 / (rapath * g)
ymui = ymumax - ymuc - ymusc
rdotp2 = -(ymui * g * rapath ^ 2 / (acfyaw * vpath))
rdot = rdotp1 * rdotp2 * rdotp0
rapath = rapath + h * rdot

REM integrate the exith path parameters
sai = sai + h * saidot
xpath = xpath + h * vpath * COS(sai)
ypath = ypath + h * vpath * SIN(sai)
vpath = vpath + h * acpath
tpath1 = tpath1 + h
count = count + 1

'CALL track(sai, vpath, xpath, ypath, h, trackmax, xmr, ymr, xml, yml, xmpath(), ympath(), xnpath(), ynpath(), dx, dy)

' printing x and y coordinates into arrays xtra and ytra

IF (count >= prt) THEN
  npnt(kex) = npnt(kex) + 1
  xtra(kex, npnt(kex)) = xpath
  ytra(kex, npnt(kex)) = ypath
  count = 0
END IF

IF ypath < (disttw / 2) THEN GOTO 235
  npnt(kex) = npnt(kex) + 1
  xtra(kex, npnt(kex)) = xpath
  ytra(kex, npnt(kex)) = ypath

'LPRINT kex, xpath, ypath
'-----!
ELSE
  CALL std.cline(kex, exctype(ex.cen(kex)), npnt(), xtra(), ytra())
END IF

END SUB

This subroutine calculates the centerline coordinates of the turnoff. Subroutines skid, exsteer, scrub and std.cline are called in this subroutine. The 'x' and 'y' coordinates are put in the arrays 'xtra' and 'ytra'.

SUB cal.track (kex, ncrit, trackmax)

'sai, vpath, xpath, ypath, dt, trackmax, xmr, ymr, xml, yml, xmpath(), ympath(), xnpath(), ynpath(), dx, dy)
REM $DYNAMIC
DIM trackin(100), xnose(100), ynose(100), xmain(100), ymain(100), saih(100)
CLS 0
gmax = 10000!      ' Maximum lateral Acceleration (g's)
jmax = 10000!      ' Maximum allowable jerk (g/sec.)
g = 9.81

'-----!
n = INT((ncrit - .1) / 2!) + 1
twet1 = INT(ncrit / 2)
twet2 = ncrit / 2
IF (twet1 = twet2) THEN
  lwet = 2

```

```

ELSE
lwet = 1
END IF

acfyaw = 10 * (1.7215 * LOG(acfmass(n)) / LOG(10!) - 1.673)
acfyaw = 0!

REM exit path deterministic computations
h = .01
dx = 5
dy = 5
count = 0
prt = 100

REM initialize state variables
xpath = 0!
ypath = 0!
xm = xpath - acfwb(n)
ym = ypath
vpath = ex.spd(kex)
sai = 0!
saimax = 1!
tpath1 = 0!
rapath = 200 + (4000 / 40) * vpath

CALL skid(catlist(n), lwet, vpath, ymumax)

REM estimate rate variables
rdotp1 = acfmass(n) * acfwb(n)
rdotp0 = (acflm(n) / 100!) * (1! - acflm(n) / 100!)
froll = .03
saimax = 1!

236 acpath = (-1) * g * froll      'free roll deceleration

CALL EXSTEER(rapath, vpath, sai, saidot, ypath, distlw)

REM EXSTEER chooses among several exit steering laws to execute a high-speed turn

CALL scrub(rapath, acfmass(), ymusc, n)

ymuc = vpath * 2 / (rapath * g)
ymul = ymumax - ymuc - ymusc
rdotp2 = -(ymul * g * rapath * 2 / (acfyaw * vpath))
rdot = rdotp1 * rdotp2 * rdotp0
rapath = rapath + h * rdot

REM integrate the exith path parameters
sai = sai + h * saidot
xpath = xpath + h * vpath * COS(sai)
ypath = ypath + h * vpath * SIN(sai)
vpath = vpath + h * acpath
tpath1 = tpath1 + h
count = count + 1

' = = = = Estimate the main landing gear centerline position

saifus = (ypath - ym) / (xpath - xm)      ' Fuselage heading (rad)

xm = xm + h * vpath * COS(saifus)      ' Main gear coordinates
ym = ym + h * vpath * SIN(saifus)

xmr = xm + (acfwt(n) / 2!) * SIN(saifus)      ' Right-hand side parameters
ymr = ym - (acfwt(n) / 2!) * COS(saifus)

xml = xm - (acfwt(n) / 2!) * SIN(saifus)      ' Left-hand side parameters
yml = ym + (acfwt(n) / 2!) * COS(saifus)

' Estimation of castor angle (castor)
' Constrain castor if necessary for nose gear limitations

IF (sai - saifus) >= saimax THEN
castor = saimax
ELSE
castor = sai - saifus
END IF

' Extraction of x-y values of main and nose gear from computational data

```

```

' Arrays xnose (hi), ynose (hi), xmain (ii), and ymain (ii) are filled
' with data from actual simulation

```

```

IF (xpath / dx) > 1 THEN
    hi = dx / 5
    xnose(hi) = xpath
    ynose(hi) = ypath
    dx = dx + 5
'PRINT hi, xnose(hi), ynose(hi)
END IF

IF (xm / dy) > 1 THEN
    ii = dy / 5
    xmain(ii) = xm
    ymain(ii) = ym
    saih(ii) = saifus
    dy = dy + 5
'PRINT ii, xmain(ii), ymain(ii): INPUT a
END IF

```

```

IF ypath < disttw / 2! THEN GOTO 236
'-----|

```

```

' Estimation of track-in distance from discrete arrays ymain(i) and ynose(i)
trackmax = 0
FOR i = 1 TO ii
    trackin(i) = ABS(ymain(i) - ynose(i)) * COS(saih(i))
    IF (trackin(i) > trackmax) THEN trackmax = trackin(i)
NEXT i
ERASE trackin, xnose, ynose, xmain, ymain, saih
END SUB

```

The main purpose of this subroutine is to calculate the maximum track-in. Again subroutines skid, exsteer and scrub are called in this subroutine.

```

REM $STATIC
SUB ccline.run2 (ex.com, ex.cen(), xtra(), ytra())

```

```

REM $DYNAMIC
DIM npnt(10)

```

```

FOR kex = 1 TO ex.com
    CALL cal.cline(kex, npnt(), xtra(), ytra())
NEXT kex

CLS 0
CALL plot.cen(ex.com, npnt(), xtra(), ytra())
ERASE npnt

END SUB

```

This subroutine is used to compare the centerline turnoff geometries of the exits requested in the runmode '2'. First the coordinates of each of the exits requested is calculated, (by calling subroutine cal.cline) and all the exits are plotted on a common plot with the aid of subroutine plot.cen.

```

REM $STATIC
SUB ccline.run3 (ex.com, ex.cen(), xtra(), ytra())

```

```

REM $DYNAMIC
DIM npnt(10)

FOR kex = 1 TO ex.com
    CALL cal.cline(kex, npnt(), xtra(), ytra())
NEXT kex

```

```

CLS 0
CALL plot.cen(ex.com, npnt(), xtra(), ytra())

ERASE npnt

END SUB

```

The main purpose of this subroutine is to compare the centerline turnoff geometries of the exits requested in the runmode '3'. First the coordinates of each of the exits requested is calculated, (by calling subroutine cal.cline) and all the exits are plotted on a common plot with the aid of subroutine plot.cen.

```

REM $STATIC
SUB cl.bot.sbox

' This clears the box and the message created by the subroutine ask.exn

FOR I = 16 TO 21
LOCATE i, 51: PRINT STRING$(25, 32)
NEXT
END SUB

```

The main purpose of this subroutine is to create a small box at the lower right hand side of the screen. The box is used to display the small menu screen which gives options to view the complete turnoff geometries or the centerline turnoff geometries.

```

SUB cl.outoplef

' This clears the top left side of the screen

a.line$ = STRING$(37, 32)
FOR I = 2 TO 6
LOCATE i, 3: PRINT a.line$
NEXT I

END SUB

```

This subroutine clears any text or graphics that is printed on the top left side of the screen. This is especially useful when a need for a change

```

SUB cl.outoprht

' This clears the top right hand side of the screen

a.line$ = STRING$(37, 32)
FOR I = 2 TO 6
LOCATE i, 40: PRINT a.line$
NEXT i

END SUB

```

```

SUB cl.sbox2

' This clears the small box

a.line$ = STRING$(76, 32)
FOR I = 8 TO 22
LOCATE i, 3: PRINT a.line$
NEXT I

END SUB

```

This subroutine clears all the text and the graphics which are within the small box, specified in subroutine box.small2. This is especially useful when it is felt that the text and graphics inside the box are to be removed while retaining the ones outside the box.

```

SUB clr.dec (i, jk)

' This is for the purpose of getting good colors

IF I <= 8 THEN

SELECT CASE I
CASE 1
jk = 14
CASE 2
jk = 12
CASE 3
jk = 11
CASE 4

```

```

jk = 13
CASE 5
jk = 7
CASE 6
jk = 10
CASE 7
jk = 5
CASE 8
jk = 3
END SELECT
ELSE
SELECT CASE (I - 8)
CASE 1
jk = 6
CASE 2
jk = 15
CASE 3
jk = 2
CASE 4
jk = 4
CASE 5
jk = 12
CASE ELSE
jk = 15
END SELECT
END IF
END SUB

```

This subroutine is used to define the numbers of good colors in an order. This helps in portraying good colors on a particular screen. This subroutine can be repeatedly called in various subroutines to display the best colors available.

```
SUB cvex (st$, nn, ann())
```

```

REM $DYNAMIC
DIM comma(10)

nn = 0
IF VAL(st$) = 0 THEN EXIT SUB

FOR jj = 1 TO LEN(st$)
  IF MID$(st$, jj, 1) = "," THEN
    nn = nn + 1
    comma(nn) = jj
  END IF
NEXT jj

FOR kk = 2 TO nn
  ann(kk) = VAL(MID$(st$, comma(kk - 1) + 1, comma(kk) - comma(kk - 1)))
NEXT kk
ann(1) = VAL(MID$(st$, 1, comma(1)))
IF VAL(RIGHT$(st$, LEN(st$) - comma(nn))) < > 0 THEN
  ann(nn + 1) = VAL(RIGHT$(st$, LEN(st$) - comma(nn)))
  nn = nn + 1
END IF
ERASE comma
END SUB

```

```

REM $STATIC
SUB exloc.ttl

```

```

' This draws a box and prints the title for the exit location display

```

```

COLOR 11
LOCATE 2, 3
PRINT CHR$(218); STRING$(26, 196); CHR$(191);
FOR i = 3 TO 5
  LOCATE i, 3
  PRINT CHR$(179);
LOCATE i, 30
  PRINT CHR$(179);
NEXT i
LOCATE 6, 3
PRINT CHR$(192); STRING$(26, 196); CHR$(217);
COLOR 10
LOCATE 3, 7: PRINT " TURNOFF LOCATIONS"

```

```
LOCATE 4, 7: PRINT " AND THEIR "
```

```
LOCATE 5, 7: PRINT " GEOMETRIES "
```

```
COLOR lcl, bcl
```

```
END SUB
```

```
SUB fc.inkey (wd$) STATIC
```

```
    wd$ = ""
    WHILE wd$ = ""
        wd$ = INKEY$
    WEND
```

```
END SUB
```

This subroutine is used to pause a screen, when needed. Every time this subroutine is called, the argument that should be provided should be a string variable.

```
SUB geo.new.30
```

```
SCREEN 9
WIDTH 80, 43
COLOR 15, 9
mts = .3048 ' for conversion from feet to meters
```

```
LINE (5, 4)-(635, 334), , B
LINE (184, 20)-(468, 35), 12, BF
LINE (188, 22)-(464, 33), 0, BF
COLOR 15
LOCATE 4, 25: PRINT "30-DEGREE HIGH-SPEED TURNOFF (FAA)"
COLOR 15, 9
```

```
VIEW (24, 24)-(616, 336)
WINDOW (0, 0)-(590, 312)
```

```
rw = 50: disttw = 200: LC = 200 * mts
RR = 1600 * mts: RRC = 1800 * mts
TR = 800 * mts: TRC = 800 * mts
RA = 25 * mts: RB = 25 * mts
```

```
ws = 45
CALL air.design.grp(ws)
CALL tax.exit.geo(des.grp)
```

```
pi = 3.141592
brdth = (tw / 2 + rw / 2 + disttw) / (SIN(pi / 180 * 30) / COS(pi / 180 * 30))
```

```
xscale = 325 / brdth
yscale = xscale / 1.37
```

```
' to display the tw & the centerline
```

```
LINE (10, 270)-(582, 270)
    twc = 270 - tw / 2 * yscale
LINE (10, twc)-(582, twc), 7, , &HFFF2
```

```
' 582 / 2 = 296
```

```
x3 = 375 '296 + disttw / 2 * xscale ' NEEDS TO BE CORRECTED
xs4 = x3 - RA * xscale * COS(pi / 180 * 60)
x4 = xs4
y4 = twc - tw / 2 * yscale
```

```
' for the bottom line of the tw
```

```
LINE (10, y4)-(x4, y4)

ys4 = y4 - RA * yscale
y3 = ys4 - RA * yscale * SIN(pi / 180 * 60)
```

```
' curve joining the bottom left side of the tw
```

```
CIRCLE (xs4, ys4), RA * xscale, 15, pi / 180 * 300, pi / 180 * 90
```

```
' these are to determine the straight line segment joining tw & rw
```

```
top = RA * (1 + SIN(pi / 180 * 60))
btm = RR * (1 - SIN(pi / 180 * 60))
```

```

Distleft = (disttw - (tw + rw) / 2 - (top + btm)) * yscale

y2 = y3 - Distleft
x2 = x3 - (y3 - y2) * 1.37 / TAN(pi / 180 * 30)

' to draw the straight line segment

LINE (x3, y3)-(x2, y2)

' to determine the center of the top left curve of the rw & to draw it

ycrr = y2 + RR * yscale * SIN(pi / 180 * 60)
xcrr = x2 - RR * xscale * COS(pi / 180 * 60)

CIRCLE (xcrr, ycrr), RR * xscale, 15, pi / 180 * 270, pi / 180 * 300

' for the point of curvature on the top line of the runway

x1 = xcrr
y1 = ycrr - RR * yscale

' to display the rw & the centerline

rwc = y1 - rw / 2 * yscale

LINE (10, rwc)-(582, rwc), 7, , &HFFF2
LINE (10, rwc - rw / 2 * yscale)-(582, rwc - rw / 2 * yscale)

' line to join the start of runway to point of curvature

LINE (10, y1)-(x1, y1)

' to draw the centerline of the exit taxiway

xexitlw = x3 + ((R - F) * xscale / COS(pi / 180 * 60))
xtwsc = xexitlw + (twc - y3) * 1.37 / TAN(pi / 180 * 30)
xrwsc = xexitlw - (y3 - rwc) * 1.37 / TAN(pi / 180 * 30)

LINE (xexitlw, y3)-(xtwsc, twc), 7, , &HFFF2      ' till the cl of tw
LINE (xexitlw, y3)-(xrwsc, rwc), 7, , &HFFF2      ' till the cl of rw

' to print "30"

CIRCLE (xrwsc, rwc), rw * xscale / 2, 13, pi / 180 * 0, pi / 180 * 30
CIRCLE (xrwsc + rw * xscale / 2 + 19, rwc + 9), 1, 10
CALL font(xrwsc + rw * xscale / 2 + 5, rwc + 3, 10, "3")
CALL font(xrwsc + rw * xscale / 2 + 11, rwc + 3, 10, "0")

' to draw the line that joins the centerlines of rw & exit rw

xscrrc = xrwsc - RRC * xscale * TAN(pi / 180 * 15)
yscrrc = rwc + RRC * yscale

CIRCLE (xscrrc, yscrrc), RRC * xscale, 7, pi / 180 * 270, pi / 180 * 300

' to draw the line that joins the centerlines of tw & exit tw

xas4 = xtwsc + TRC * xscale * TAN(pi / 180 * 15)
yas4 = twc - TRC * yscale

CIRCLE (xas4, yas4), TRC * xscale, 7, pi / 180 * 90, pi / 180 * 120

' to draw the right side of the exit taxiway

ya1 = y1
ya2 = ya1 + RB * (1 + SIN(pi / 180 * 60)) * yscale

xextwrgt = xexitlw + ((R - F) * xscale / COS(pi / 180 * 60))
yyextw = (y3 - ya2) / TAN(pi / 180 * 30) * 1.37
xa2 = xextwrgt - yyextw

LINE (xa2, ya2)-(xextwrgt, y3)

' to draw the top right curve of the rw

xas1 = xa2 + RB * COS(pi / 180 * 60) * xscale
yas1 = ya1 + RB * yscale

CIRCLE (xas1, yas1), RB * xscale, 15, pi / 180 * 120, pi / 180 * 270

```

```

' to draw the right top line of the rw
xa1 = xas1
LINE (xa1, y1)-(582, y1)
' to draw the top right curve of the tw
ya4 = y4
yas3 = ya4 - TR * yscale
ya3 = yas3 + COS(pi / 180 * 30) * TR * yscale
xa3 = xextwrgt + (ya3 - y3) / (TAN(pi / 180 * 30)) * 1.37 '=====
' continuation of right side line of the exit taxiway
LINE (xa3, ya3)-(xextwrgt, y3), 0 '=====
' to draw the top right curve of the tw
xas3 = xa3 + SIN(pi / 180 * 30) * TR * xscale
CIRCLE (xas3, yas3), TR * xscale, 15, pi / 180 * 90, pi / 180 * 120
' to draw the bottom right side of the tw
xa4 = xas3
LINE (xa4, ya4)-(582, ya4)
' to draw the curves with R and F for the tw
Ty2 = twc - R * yscale
Ty1 = Ty2 - COS(pi / 180 * 30) * R * yscale
Tx1 = xrwsc + (Ty1 - rwc) * 1.37 / (SIN(pi / 180 * 30) / COS(pi / 180 * 30))
Tx2 = Tx1 - R * xscale * SIN(pi / 180 * 30)
CIRCLE (Tx2, Ty2), R * xscale, 7, pi / 180 * 300, pi / 180 * 90
CIRCLE (Tx2, Ty2), F * xscale, 7, pi / 180 * 300, pi / 180 * 90
' for the fillets
xf5 = Tx2
yf5 = Ty2 + F * yscale
xl6 = xf5 - L * xscale
yl6 = y4
LINE (xf5, yf5)-(xl6, yl6), 7
COLOR 13 ' color used for arrows
' for displaying the radius of outer curve (F) for the taxiway
LINE (Tx2, yf5)-(Tx2, Ty2)
LINE (Tx2, yf5)-(Tx2 - 3, yf5 - 3)
LINE (Tx2, yf5)-(Tx2 + 3, yf5 - 3)
CALL font(Tx2 + 5, yf5 - 10, 10, "F")
' for displaying the length 'L' of the fillet
LINE (xf5, yf5 - 8)-(xl6, yf5 - 8)
LINE (xf5, yf5 - 8)-(xl6 - 3, yf5 - 10)
LINE (xf5, yf5 - 8)-(xl6 - 3, yf5 - 6)
LINE (xl6, yf5 - 8)-(xl6 + 3, yf5 - 10)
LINE (xl6, yf5 - 8)-(xl6 + 3, yf5 - 6)
LINE (xl6, yl6 - 3)-(xl6, yl6 - 22)
CALL font(xf5 + (xl6 - xf5) / 2, yf5 - 6, 10, "L")
' for displaying the radius of centerline curve (R) for the taxiway
yyR = R * yscale * SIN(25 * pi / 180)
xxR = R * xscale * COS(25 * pi / 180)
LINE (Tx2, Ty2)-(Tx2 + xxR, Ty2 + yyR)
LINE (Tx2 + xxR, Ty2 + yyR)-(Tx2 + xxR - 3, Ty2 + yyR - 3)
LINE (Tx2 + xxR, Ty2 + yyR)-(Tx2 + xxR - 5, Ty2 + yyR + 1)
CALL font(Tx2 + xxR - 8, Ty2 + yyR - 11, 10, "R")

```

```

' for displaying the width of taxiway

LINE (60 - 30, y4)-(60 - 30, y4 - 15)
LINE (60 - 30, y4)-(60 - 27, y4 - 3)
LINE (60 - 30, y4)-(60 - 33, y4 - 3)

LINE (60 - 30, 270)-(60 - 30, 285)
LINE (60 - 30, 270)-(60 - 27, 270 + 3)
LINE (60 - 30, 270)-(60 - 33, 270 + 3)

CALL font(78 - 42, 270 + 9, 10, "W")
CALL font(78 - 37, 270 + 9, 10, "T")

' for the arrows to display the width of runway

LINE (535, y1)-(535, y1 + 15)
LINE (535, y1)-(537, y1 + 3)
LINE (535, y1)-(533, y1 + 3)
LINE (535, y1 - rw * yscale)-(535, y1 - rw * yscale - 15)
LINE (535, y1 - rw * yscale)-(538, y1 - rw * yscale - 2)
LINE (535, y1 - rw * yscale)-(532, y1 - rw * yscale - 2)

CALL font(542, y1 + 8, 10, "W")
CALL font(547, y1 + 8, 10, "R")

' to display the distance between the rw & tw

LINE (0, twc)-(9, twc)
LINE (0, rwc)-(9, rwc)
LINE (4, twc)-(4, rwc)
LINE (4, twc)-(1, twc - 3)
LINE (4, twc)-(7, twc - 3)
LINE (4, rwc)-(1, rwc + 3)
LINE (4, rwc)-(7, rwc + 3)

FOR x = 1 TO LEN("D-TWRW")
    t$ = MID$("D-TWRW", x, 1)
    CALL font(4 + 5 * x, y1 + (disttw - (tw + rw) / 2) * yscale / 2, 10, t$)
NEXT x

' to display the radius of curve joining bottom line (left) of tw

yiR = RA * yscale * SIN(40 * pi / 180)
xiR = RA * xscale * COS(40 * pi / 180)
LINE (xs4, ys4)-(xs4 + xiR, ys4 + yiR)
LINE (xs4 + xiR, ys4 + yiR)-(xs4 + xiR - 2, ys4 + yiR - 3)
LINE (xs4 + xiR, ys4 + yiR)-(xs4 + xiR - 5, ys4 + yiR - 1)

CALL font(xs4 - 12, ys4 - 4, 10, "R")
CALL font(xs4 - 7, ys4 - 4, 10, "A")

' to display the radius of curve joining top line (right) of rw

xiR = RB * xscale * SIN(45 * pi / 180)
yiR = RB * yscale * COS(45 * pi / 180)
LINE (xas1, yas1)-(xas1 - xiR, yas1 - yiR)
LINE (xas1 - xiR, yas1 - yiR)-(xas1 - xiR + 2, yas1 - yiR + 3)
LINE (xas1 - xiR, yas1 - yiR)-(xas1 - xiR + 5, yas1 - yiR + 1)

CALL font(xas1 + xiR + 3, yas1 + yiR - 2, 10, "R")
CALL font(xas1 + xiR + 8, yas1 + yiR - 2, 10, "B")

' to display 'cl' on the centerlines of tw & rw

CALL font(550, twc, 10, "c")
CALL font(552, twc - 3, 10, "l")

CALL font(550, rwc, 10, "c")
CALL font(552, rwc - 3, 10, "l")

' for displaying the extended centerline of the runway for 'LC' (200 feet)

LINE (x1, y1 - rw * yscale / 8)-(x1 - LC * yscale, y1 - rw * yscale / 8)
LINE (x1, y1 - rw * yscale / 8)-(x1 - 3, y1 + 2 - rw * yscale / 8)
LINE (x1, y1 - rw * yscale / 8)-(x1 - 3, y1 - 2 - rw * yscale / 8)
LINE (x1 - LC * yscale, y1 - rw * yscale / 8)-(x1 - LC * yscale + 3, y1 + 2 - rw * yscale / 8)
LINE (x1 - LC * yscale, y1 - rw * yscale / 8)-(x1 - LC * yscale + 3, y1 - 2 - rw * yscale / 8)

```

```

LINE (x1, y1 - 1)-(x1, y1 - 4 - rw * yscale / 8)
LINE (x1 - LC * yscale, y1 - 1)-(x1 - LC * yscale, y1 - 4 - rw * yscale / 8)

  FOR x = 1 TO LEN("LC")
    q$ = MID$("LC", x, 1)
    CALL font(x1 - 170 * mts * yscale + 5 * x, y1 - rw * yscale / 8 + 6, 10, q$)
  NEXT x

CALL font(x1 + 6, y1 - rw * yscale / 8 + 6, 10, "P")
CALL font(x1 + 12, y1 - rw * yscale / 8 + 6, 10, "C")

' for the seperation between the above line and the centerline (3 feet)

LINE (x1, rwc + 1)-(x1 - LC * yscale, rwc + 1), 7, , &HFFF2
  LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale, rwc + rw * yscale / 4)
  LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale, rwc - rw * yscale / 4)
  LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale - 2, rwc - 3 - 1)
  LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale + 2, rwc - 3 - 1)
  LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale - 2, rwc + 2 + 3)
  LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale + 2, rwc + 2 + 3)

CALL font(x1 - LC / 2 * yscale + 6, rwc - 10, 10, "D")

' to display the radius of curve joining centerline of tw

xjr = TRC * xscale * SIN(15 * pi / 180)
yjr = TRC * yscale * COS(15 * pi / 180)

LINE (xas4 - xjr, yas4 + yjr)-(xas4 - xjr + (xjr / 3), yas4 + yjr - xjr / 3 / 1.37 * TAN(pi / 180 * 60))
LINE (xas4 - xjr, yas4 + yjr)-(xas4 - xjr - 1, yas4 + yjr - 3)
LINE (xas4 - xjr, yas4 + yjr)-(xas4 - xjr + 6, yas4 + yjr - 1)

CALL font(xas4 - xjr * 1.5 + (xjr / 3), yas4 + yjr - xjr / 1.5 / 1.37, 10, "R")
CALL font(xas4 - xjr * 1.5 + 5 + (xjr / 3), yas4 + yjr - xjr / 1.5 / 1.37, 10, "T")
CALL font(xas4 - xjr * 1.5 + 10 + (xjr / 3), yas4 + yjr - xjr / 1.5 / 1.37, 10, "W")
CALL font(xas4 - xjr * 1.5 + 15 + (xjr / 3), yas4 + yjr - xjr / 1.5 / 1.37, 10, "C")

' to display the radius of curve joining top line (right) of tw

xkR = TR * xscale * SIN(15 * pi / 180)
ykR = TR * yscale * COS(15 * pi / 180)

LINE (xas3 - xkR, yas3 + ykR)-(xas3 - xkR + (xkR / 3), yas3 + ykR - xkR / 3 / 1.37 * TAN(pi / 180 * 60))
LINE (xas3 - xkR, yas3 + ykR)-(xas3 - xkR - 1, yas3 + ykR - 3)
LINE (xas3 - xkR, yas3 + ykR)-(xas3 - xkR + 6, yas3 + ykR - 1)

CALL font(xas3 - xkR * .94 + (xkR / 3), yas3 + ykR * 1.04 - xkR / 2 / 1.37, 10, "R")
CALL font(xas3 - xkR * .94 + 5 + (xkR / 3), yas3 + ykR * 1.04 - xkR / 2 / 1.37, 10, "T")
CALL font(xas3 - xkR * .94 + 10 + (xkR / 3), yas3 + ykR * 1.04 - xkR / 2 / 1.37, 10, "W")

' to display the radius of curve joining top line (left) of rw

xmr = RR * xscale * SIN(20.5 * pi / 180)
ymr = RR * yscale * COS(20.5 * pi / 180)

LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr - xmr / 5, ycrr - ymr + xmr / 5 / 1.37 * TAN(pi / 180 * 60))
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr + 1, ycrr - ymr + 3)
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr - 4, ycrr - ymr + 1)

CALL font(xcrr + 3 + xmr - xmr / 3, ycrr - ymr + xmr / 5 / 1.37, 10, "R")
CALL font(xcrr + 8 + xmr - xmr / 3, ycrr - ymr + xmr / 5 / 1.37, 10, "R")
CALL font(xcrr + 13 + xmr - xmr / 3, ycrr - ymr + xmr / 5 / 1.37, 10, "W")

' to display the radius of curve joining centerline (left) of rw

xnr = RRC * xscale * SIN(25.5 * pi / 180)
ynr = RRC * yscale * COS(25.5 * pi / 180)

LINE (xscrrc + xnr, yscrrc - ynr)-(xscrrc + xnr - xnr / 5, yscrrc - ynr + xnr / 5 / 1.37 * TAN(pi / 180 * 60))
LINE (xscrrc + xnr, yscrrc - ynr)-(xscrrc + xnr + 1, yscrrc - ynr + 3)
LINE (xscrrc + xnr, yscrrc - ynr)-(xscrrc + xnr - 4, yscrrc - ynr + 1)

CALL font(xscrrc + 3 + xnr - xnr / 4, yscrrc - ynr + xnr / 5 / 1.37, 10, "R")
CALL font(xscrrc + 8 + xnr - xnr / 4, yscrrc - ynr + xnr / 5 / 1.37, 10, "R")
CALL font(xscrrc + 13 + xnr - xnr / 4, yscrrc - ynr + xnr / 5 / 1.37, 10, "W")
CALL font(xscrrc + 18 + xnr - xnr / 4, yscrrc - ynr + xnr / 5 / 1.37, 10, "C")

```

```

COLOR 10
  LOCATE 8, 62: PRINT "TAXIWAY";
  ROW = (43 - (y1 - rw * yscale) / 8.14)
  LOCATE ROW, 62: PRINT "RUNWAY";

COLOR 15
a = rw / 2 * yscale
b = tw / 2 * yscale

LINE (10, y1 - a + a + .1 * a)-(10, y1 - a + a - .3 * a)
LINE (10, y1 - a + a - .3 * a)-(15, y1 - a + a - .3 * a)
LINE (15, y1 - a + a - .3 * a)-(5, y1 - a + a - .7 * a)
LINE (5, y1 - a + a - .7 * a)-(10, y1 - a + a - .7 * a)
LINE (10, y1 - a + a - .7 * a)-(10, y1 - a - .3 * a)
LINE (10, y1 - a - .3 * a)-(15, y1 - a - .3 * a)
LINE (15, y1 - a - .3 * a)-(5, y1 - a - .7 * a)
LINE (5, y1 - a - .7 * a)-(10, y1 - a - .7 * a)
LINE (10, y1 - a - .7 * a)-(10, y1 - a - .1 * a)

LINE (582, y1 - a + a + .1 * a)-(582, y1 - a + a - .3 * a)
LINE (582, y1 - a + a - .3 * a)-(587, y1 - a + a - .3 * a)
LINE (587, y1 - a + a - .3 * a)-(577, y1 - a + a - .7 * a)
LINE (577, y1 - a + a - .7 * a)-(582, y1 - a + a - .7 * a)
LINE (582, y1 - a + a - .7 * a)-(582, y1 - a - .3 * a)
LINE (582, y1 - a - .3 * a)-(587, y1 - a - .3 * a)
LINE (587, y1 - a - .3 * a)-(577, y1 - a - .7 * a)
LINE (577, y1 - a - .7 * a)-(582, y1 - a - .7 * a)
LINE (582, y1 - a - .7 * a)-(582, y1 - a - .1 * a)

LINE (10, y4 + b + b + .3 * b)-(10, y4 + b + b - .3 * b)
LINE (10, y4 + b + b - .3 * b)-(13, y4 + b + b - .3 * b)
LINE (13, y4 + b + b - .3 * b)-(7, y4 + b + b - .7 * b)
LINE (7, y4 + b + b - .7 * b)-(10, y4 + b + b - .7 * b)
LINE (10, y4 + b + b - .7 * b)-(10, y4 + b - .3 * b)
LINE (10, y4 + b - .3 * b)-(13, y4 + b - .3 * b)
LINE (13, y4 + b - .3 * b)-(7, y4 + b - .7 * b)
LINE (7, y4 + b - .7 * b)-(10, y4 + b - .7 * b)
LINE (10, y4 + b - .7 * b)-(10, y4 + b - .3 * b)

LINE (582, y4 + b + b + .3 * b)-(582, y4 + b + b - .3 * b)
LINE (582, y4 + b + b - .3 * b)-(585, y4 + b + b - .3 * b)
LINE (585, y4 + b + b - .3 * b)-(579, y4 + b + b - .7 * b)
LINE (579, y4 + b + b - .7 * b)-(582, y4 + b + b - .7 * b)
LINE (582, y4 + b + b - .7 * b)-(582, y4 + b - .3 * b)
LINE (582, y4 + b - .3 * b)-(585, y4 + b - .3 * b)
LINE (585, y4 + b - .3 * b)-(579, y4 + b - .7 * b)
LINE (579, y4 + b - .7 * b)-(582, y4 + b - .7 * b)
LINE (582, y4 + b - .7 * b)-(582, y4 + b - .3 * b)

COLOR 11

RO = ROW + 5

LOCATE RO, 6: PRINT "WR" =";
LOCATE RO, 15: PRINT USING "### #"; rw: LOCATE RO, 22: PRINT "m"
LOCATE RO, 26: PRINT USING "### #"; rw / mts
LOCATE RO, 25: PRINT "(" LOCATE RO, 33: PRINT "Feet) :";

RO = ROW + 6

LOCATE RO, 6: PRINT "WT" =";
LOCATE RO, 15: PRINT USING "### #"; tw: LOCATE RO, 22: PRINT "m"
LOCATE RO, 26: PRINT USING "### #"; tw / mts
LOCATE RO, 25: PRINT "(" LOCATE RO, 33: PRINT "Feet) :";

RO = ROW + 7

LOCATE RO, 6: PRINT "RTW" =";
LOCATE RO, 15: PRINT USING "### #"; TR: LOCATE RO, 22: PRINT "m"
LOCATE RO, 26: PRINT USING "### #"; TR / mts
LOCATE RO, 25: PRINT "(" LOCATE RO, 33: PRINT "Feet) :";

RO = ROW + 8

LOCATE RO, 6: PRINT "RTWC" =";
LOCATE RO, 15: PRINT USING "### #"; TRC: LOCATE RO, 22: PRINT "m"

```

```

LOCATE RO, 26: PRINT USING "### ##"; TRC / mts
LOCATE RO, 25: PRINT "(" LOCATE RO, 33: PRINT "Feet) :"

```

END SUB

This subroutine displays the standard FAA turnoff geometry (30 degrees). The inputs to this program are obtained by calling in the subroutines air.design.grp and tax.exit.geo. In addition to the values that are obtained through these subroutines, this needs values of runway width, distance between the runway and the taxiway and the wingspan. All other values that are needed are specified at the start of the subroutine. In this subroutine the scale depends on the distance between runway and taxiway, the runway width and the taxiway width. All the dimensions are displayed at the bottom of the screen in both English and Metric systems.

SUB geo.new.45

```
SCREEN 9
WIDTH 80, 43
COLOR 15, 9
mts = .3048 ' for conversion from feet to meters

LINE (5, 4)-(635, 334), , B
LINE (184, 20)-(468, 35), 12, BF
LINE (188, 22)-(464, 33), 0, BF
COLOR 15
LOCATE 4, 25: PRINT "45-DEGREE HIGH-SPEED TURNOFF (FAA)"
COLOR 15, 9
```

```
VIEW (24, 24)-(616, 336)
WINDOW (0, 0)-(590, 312)
```

```
rw = 50: disttw = 200
RR = 600 * mts: RRC = 800 * mts
TR = 100 * mts: TRC = 120 * mts
RA = 50 * mts: RB = 25 * mts: ETW = 40 * mts
```

```
ws = 45
CALL air.design.grp(ws)
CALL tax.exit.geo(des.grp)
```

```
yscale = 169 / (tw / 2 + rw / 2 + disttw)
xscale = yscale * 1.37
```

```
pi = 3.141592
```

```
' to display the tw & the centerline
```

```
LINE (10, 270)-(582, 270)
twc = 270 - tw / 2 * yscale
LINE (10, twc)-(582, twc), 7, , &HFFF2
```

```
CALL font(530, twc, 10, "c")
CALL font(532, twc - 3, 10, "l")
```

```
' to display the rw & the centerline
```

```
LINE (10, 101)-(582, 101)
rwc = 101 + rw / 2 * yscale
LINE (10, rwc)-(582, rwc), 7, , &HFFF2
```

```
CALL font(530, rwc, 10, "c")
CALL font(532, rwc - 3, 10, "l")
```

```
' 582 / 2 = 296
```

```
x3 = 296 + disttw * xscale / 3
xs4 = x3 - RA * xscale * COS(pi / 180 * 45)
x4 = xs4
y4 = twc - tw / 2 * yscale
```

```
' for the bottom line of the tw
```

```
LINE (10, y4)-(x4, y4)
```

```
ys4 = y4 - RA * yscale
y3 = ys4 - RA * yscale * SIN(pi / 180 * 45)
```

```
' curve joining the bottom left side of the tw
```

```
CIRCLE (xs4, ys4), RA * xscale, 15, pi / 180 * 315, pi / 180 * 90
```

```
' these are to determine the straight line segment joining tw & rw
```

```

top = RA * (1 + SIN(pi / 180 * 45))
btm = RR * (1 - SIN(pi / 180 * 45))
Distleft = (disttw - (tw + rw) / 2 - (top + btm)) * yscale

y2 = y3 - Distleft
x2 = x3 - (y3 - y2) * 1.37

' to draw the straight line segment
LINE (x3, y3)-(x2, y2)

' to determine the center of the top left curve of the rw & to draw it
ycrr = y2 + RR * yscale * SIN(pi / 180 * 45)
xcrr = x2 - RR * xscale * COS(pi / 180 * 45)
CIRCLE (xcrr, ycrr), RR * xscale, 15, pi / 180 * 270, pi / 180 * 315

' for the point of curvature on the top line of the runway
x1 = xcrr
y1 = rwc + rw / 2 * yscale

' line to join the start of runway to point of curvature
LINE (10, y1)-(x1, y1)

' to draw the centerline of the exit taxiway
xexitlw = x3 + (ETW / 2 * xscale / COS(pi / 180 * 45))
xtwsc = xexitlw + (twc - y3) * 1.37
xrwsc = xexitlw - (y3 - rwc) * 1.37
LINE (xexitlw, y3)-(xtwsc, twc), 7, , &HFFF2 ' till the cl of tw
LINE (xexitlw, y3)-(xrwsc, rwc), 7, , &HFFF2 ' till the cl of rw

' to print "45"
CIRCLE (xrwsc, rwc), rw * xscale / 4, 13, pi / 180 * 0, pi / 180 * 45
CIRCLE (xrwsc + rw / 2 * xscale / 2 + 19, rwc + 9), 1, 10
CALL font(xrwsc + rw * xscale / 4 + 5, rwc + 4, 10, "4")
CALL font(xrwsc + rw * xscale / 4 + 11, rwc + 4, 10, "5")

' to draw the line that joins the centerlines of rw & exit tw
xscrrc = xrwsc - RRC * xscale * (SIN(pi / 180 * 22.5) / COS(pi / 180 * 22.5))
yscrrc = rwc + RRC * yscale
CIRCLE (xscrrc, yscrrc), RRC * xscale, 7, pi / 180 * 270, pi / 180 * 315

' to draw the line that joins the centerlines of tw & exit tw
xas4 = xtwsc + TRC * xscale * (SIN(pi / 180 * 22.5) / COS(pi / 180 * 22.5))
yas4 = twc - TRC * yscale
CIRCLE (xas4, yas4), TRC * xscale, 7, pi / 180 * 90, pi / 180 * 135

' to draw the right side of the exit taxiway
ya1 = y1
ya2 = ya1 + RB * (1 + SIN(pi / 180 * 45)) * yscale

xextwrgt = xexitlw + (ETW / 2 * xscale / COS(pi / 180 * 45))
yyextw = (y3 - ya2) * 1.37
xa2 = xextwrgt - yyextw
LINE (xa2, ya2)-(xextwrgt, y3)

' to draw the top right curve of the rw
xas1 = xa2 + RB * COS(pi / 180 * 45) * xscale
yas1 = ya1 + RB * yscale
CIRCLE (xas1, yas1), RB * xscale, 15, pi / 180 * 135, pi / 180 * 270

' to draw the right top line of the rw
xa1 = xas1

```

```

LINE (xa1, y1)-(582, y1)

' to draw the top right curve of the tw

ya4 = y4
yas3 = ya4 - TR * yscale
ya3 = yas3 + COS(pi / 180 * 45) * TR * yscale
xa3 = xextwrgt + (ya3 - y3) * 1.37

' continuation of right side line of the exit taxiway
LINE (xa3, ya3)-(xextwrgt, y3)

' to draw the top right curve of the tw

xas3 = xa3 + SIN(pi / 180 * 45) * TR * xscale

CIRCLE (xas3, yas3), TR * xscale, 15, pi / 180 * 90, pi / 180 * 135

' to draw the bottom right side of the tw

xa4 = xas3

LINE (xa4, ya4)-(582, ya4)

COLOR 13          ' color used for arrows

' for displaying the width of taxiway

LINE (x1 - 30, y4)-(x1 - 30, y4 - 15)
LINE (x1 - 30, y4)-(x1 - 27, y4 - 3)
LINE (x1 - 30, y4)-(x1 - 33, y4 - 3)

LINE (x1 - 30, 270)-(x1 - 30, 285)
LINE (x1 - 30, 270)-(x1 - 27, 270 + 3)
LINE (x1 - 30, 270)-(x1 - 33, 270 + 3)

CALL font(x1 - 22, 270 + 9, 10, "W")
CALL font(x1 - 17, 270 + 9, 10, "T")

' for the arrows to display the width of runway

LINE (35, y1)-(35, y1 + 15)
LINE (35, y1)-(37, y1 + 3)
LINE (35, y1)-(33, y1 + 3)
LINE (35, 86)-(35, 101)
LINE (35, 101)-(38, 99)
LINE (35, 101)-(32, 99)

CALL font(42, y1 + 8, 10, "W")
CALL font(47, y1 + 8, 10, "R")

' to display the distance between the rw & tw

LINE (0, twc)-(9, twc)
LINE (0, rwc)-(9, rwc)
LINE (4, twc)-(4, rwc)
LINE (4, twc)-(1, twc - 3)
LINE (4, twc)-(7, twc - 3)
LINE (4, rwc)-(1, rwc + 3)
LINE (4, rwc)-(7, rwc + 3)

FOR x = 1 TO LEN("D-TWRW")
    t$ = MID$("D-TWRW", x, 1)
    CALL font(4 + 5 * x, y1 + (disttw - (tw + rw) / 2) * yscale / 2, 10, t$)
NEXT x

' to display the radius of curve joining bottom line (left) of tw

yiR = RA * yscale * SIN(40 * pi / 180)
xiR = RA * xscale * COS(40 * pi / 180)
LINE (xs4, ys4)-(xs4 + xiR, ys4 + yiR)
LINE (xs4 + xiR, ys4 + yiR)-(xs4 + xiR - 2, ys4 + yiR - 3)
LINE (xs4 + xiR, ys4 + yiR)-(xs4 + xiR - 5, ys4 + yiR - 1)

CALL font(xs4 + xiR - 18, ys4 + yiR - 14, 10, "R")
CALL font(xs4 + xiR - 13, ys4 + yiR - 14, 10, "A")

```

' to display the radius of curve joining top line (right) of rw

```
xIR = RB * xscale * SIN(45 * pi / 180)
yIR = RB * xscale * COS(45 * pi / 180)
LINE (xas1, yas1)-(xas1 - xIR, yas1 - yIR)
LINE (xas1 - xIR, yas1 - yIR)-(xas1 - xIR + 2, yas1 - yIR + 3)
LINE (xas1 - xIR, yas1 - yIR)-(xas1 - xIR + 5, yas1 - yIR + 1)
```

```
CALL font(xas1 + xIR - 2, yas1 + yIR - 4, 10, "R")
CALL font(xas1 + xIR + 3, yas1 + yIR - 4, 10, "B")
```

' to display the radius of curve joining centerline of tw

```
xjr = TRC * xscale * SIN(15 * pi / 180)
yjr = TRC * xscale * COS(15 * pi / 180)
LINE (xas4, yas4)-(xas4 - xjr, yas4 + yjr)
LINE (xas4 - xjr, yas4 + yjr)-(xas4 - xjr - 2, yas4 + yjr - 3)
LINE (xas4 - xjr, yas4 + yjr)-(xas4 - xjr + 5, yas4 + yjr - 3)
```

```
CALL font(xas4 + 4, yas4 + 7, 10, "R")
CALL font(xas4 + 9, yas4 + 7, 10, "T")
CALL font(xas4 + 14, yas4 + 7, 10, "W")
CALL font(xas4 + 19, yas4 + 7, 10, "C")
```

' to display the radius of curve joining top line (right) of tw

```
xkR = TR * xscale * SIN(25 * pi / 180)
ykR = TR * xscale * COS(25 * pi / 180)
LINE (xas3, yas3)-(xas3 - xkR, yas3 + ykR)
LINE (xas3 - xkR, yas3 + ykR)-(xas3 - xkR + 1, yas3 + ykR - 4)
LINE (xas3 - xkR, yas3 + ykR)-(xas3 - xkR + 6, yas3 + ykR - 3)
```

```
CALL font(xas3 - 32 + 4, yas3 + 3, 10, "R")
CALL font(xas3 - 32 + 9, yas3 + 3, 10, "T")
CALL font(xas3 - 32 + 14, yas3 + 3, 10, "W")
```

' to draw the arrow for the width of exit tw

```
tay = y3 - (y3 - y2) / 2
tax = x3 - (y3 - y2) / 2 * 1.37
```

```
LINE (tax, tay)-(tax - 10, tay + 10 / 1.37)
LINE (tax, tay)-(tax - 2, tay + 3)
LINE (tax, tay)-(tax - 4, tay + 1)
```

```
tby = tay - ETW * xscale * SIN(pi / 180 * 45)
tbx = tax + ETW * xscale * COS(pi / 180 * 45)
```

```
LINE (tbx, tby)-(tbx + 10, tby - 10 / 1.37)
LINE (tbx, tby)-(tbx + 2, tby - 3)
LINE (tbx, tby)-(tbx + 4, tby - 1)
```

```
CALL font(tbx + 16, tby - 6, 10, "E")
CALL font(tbx + 21, tby - 6, 10, "X")
CALL font(tbx + 26, tby - 6, 10, "W")
CALL font(tbx + 31, tby - 6, 10, "T")
```

' to display the radius of curve joining top line (left) of rw

```
xmr = RR * xscale * SIN(22.5 * pi / 180)
ymr = RR * xscale * COS(22.5 * pi / 180)
```

```
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr - xmr / 3, ycrr - ymr + xmr / 3 / 1.37 * TAN(pi / 180 * 60))
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr + 1, ycrr - ymr + 3)
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr - 4, ycrr - ymr + 1)
```

```
CALL font(xcrr + 8 + xmr * 1.2 - xmr / 2, ycrr - ymr + xmr / 2 / 1.37, 10, "R")
CALL font(xcrr + 13 + xmr * 1.2 - xmr / 2, ycrr - ymr + xmr / 2 / 1.37, 10, "R")
CALL font(xcrr + 18 + xmr * 1.2 - xmr / 2, ycrr - ymr + xmr / 2 / 1.37, 10, "W")
```

' to display the radius of curve joining centerline (left) of rw

```
xnr = RRC * xscale * SIN(22.5 * pi / 180)
ynr = RRC * xscale * COS(22.5 * pi / 180)
```

```
LINE (xscrrc + xnr, yscrrc - ynr)-(xscrrc + xnr - xnr / 3, yscrrc - ynr + xnr / 3 / 1.37 * TAN(pi / 180 * 60))
LINE (xscrrc + xnr, yscrrc - ynr)-(xscrrc + xnr + 1, yscrrc - ynr + 3)
LINE (xscrrc + xnr, yscrrc - ynr)-(xscrrc + xnr - 4, yscrrc - ynr + 1)
```

```
CALL font(xscrrc + 8 + xnr * .9 - xnr / 2, yscrrc - ynr * 1.04 + xnr / 2 / 1.37, 10, "R")
CALL font(xscrrc + 13 + xnr * .9 - xnr / 2, yscrrc - ynr * 1.04 + xnr / 2 / 1.37, 10, "R")
CALL font(xscrrc + 18 + xnr * .9 - xnr / 2, yscrrc - ynr * 1.04 + xnr / 2 / 1.37, 10, "W")
CALL font(xscrrc + 23 + xnr * .9 - xnr / 2, yscrrc - ynr * 1.04 + xnr / 2 / 1.37, 10, "C")
```

```
COLOR 10
LOCATE 31, 62: PRINT "RUNWAY";
LOCATE 8, 62: PRINT "TAXIWAY";
```

```
COLOR 15
a = rw / 2 * yscale
b = tw / 2 * yscale
```

```
LINE (10, y1 - a + a + .1 * a)-(10, y1 - a + a - .3 * a)
LINE (10, y1 - a + a - .3 * a)-(15, y1 - a + a - .3 * a)
LINE (15, y1 - a + a - .3 * a)-(5, y1 - a + a - .7 * a)
LINE (5, y1 - a + a - .7 * a)-(10, y1 - a + a - .7 * a)
LINE (10, y1 - a + a - .7 * a)-(10, y1 - a - .3 * a)
LINE (10, y1 - a - .3 * a)-(15, y1 - a - .3 * a)
LINE (15, y1 - a - .3 * a)-(5, y1 - a - .7 * a)
LINE (5, y1 - a - .7 * a)-(10, y1 - a - .7 * a)
LINE (10, y1 - a - .7 * a)-(10, y1 - a - .1 * a)
```

```
LINE (582, y1 - a + a + .1 * a)-(582, y1 - a + a - .3 * a)
LINE (582, y1 - a + a - .3 * a)-(587, y1 - a + a - .3 * a)
LINE (587, y1 - a + a - .3 * a)-(577, y1 - a + a - .7 * a)
LINE (577, y1 - a + a - .7 * a)-(582, y1 - a + a - .7 * a)
LINE (582, y1 - a + a - .7 * a)-(582, y1 - a - .3 * a)
LINE (582, y1 - a - .3 * a)-(587, y1 - a - .3 * a)
LINE (587, y1 - a - .3 * a)-(577, y1 - a - .7 * a)
LINE (577, y1 - a - .7 * a)-(582, y1 - a - .7 * a)
LINE (582, y1 - a - .7 * a)-(582, y1 - a - .1 * a)
```

```
LINE (10, y4 + b + b + .3 * b)-(10, y4 + b + b - .3 * b)
LINE (10, y4 + b + b - .3 * b)-(13, y4 + b + b - .3 * b)
LINE (13, y4 + b + b - .3 * b)-(7, y4 + b + b - .7 * b)
LINE (7, y4 + b + b - .7 * b)-(10, y4 + b + b - .7 * b)
LINE (10, y4 + b + b - .7 * b)-(10, y4 + b - .3 * b)
LINE (10, y4 + b - .3 * b)-(13, y4 + b - .3 * b)
LINE (13, y4 + b - .3 * b)-(7, y4 + b - .7 * b)
LINE (7, y4 + b - .7 * b)-(10, y4 + b - .7 * b)
LINE (10, y4 + b - .7 * b)-(10, y4 + b - .3 * b)
```

```
LINE (582, y4 + b + b + .3 * b)-(582, y4 + b + b - .3 * b)
LINE (582, y4 + b + b - .3 * b)-(585, y4 + b + b - .3 * b)
LINE (585, y4 + b + b - .3 * b)-(579, y4 + b + b - .7 * b)
LINE (579, y4 + b + b - .7 * b)-(582, y4 + b + b - .7 * b)
LINE (582, y4 + b + b - .7 * b)-(582, y4 + b - .3 * b)
LINE (582, y4 + b - .3 * b)-(585, y4 + b - .3 * b)
LINE (585, y4 + b - .3 * b)-(579, y4 + b - .7 * b)
LINE (579, y4 + b - .7 * b)-(582, y4 + b - .7 * b)
LINE (582, y4 + b - .7 * b)-(582, y4 + b - .3 * b)
```

```
COLOR 11
```

```
LOCATE 34, 6: PRINT "WR" =";
LOCATE 34, 15: PRINT USING "#### #"; rw: LOCATE 34, 22: PRINT "m"
LOCATE 34, 26: PRINT USING "#### #"; rw / mts
LOCATE 34, 25: PRINT "(" LOCATE 34, 33: PRINT "Feet) :"
```

```
LOCATE 35, 6: PRINT "WT" =";
LOCATE 35, 15: PRINT USING "#### #"; tw: LOCATE 35, 22: PRINT "m"
LOCATE 35, 26: PRINT USING "#### #"; tw / mts
LOCATE 35, 25: PRINT "(" LOCATE 35, 33: PRINT "Feet) :"
```

```
LOCATE 36, 6: PRINT "RRW" =";
LOCATE 36, 15: PRINT USING "#### #"; RR: LOCATE 36, 22: PRINT "m"
LOCATE 36, 26: PRINT USING "#### #"; RR / mts
LOCATE 36, 25: PRINT "(" LOCATE 36, 33: PRINT "Feet) :"
```

```
LOCATE 37, 6: PRINT "RRWC" =";
LOCATE 37, 15: PRINT USING "#### #"; RRC: LOCATE 37, 22: PRINT "m"
LOCATE 37, 26: PRINT USING "#### #"; RRC / mts
LOCATE 37, 25: PRINT "(" LOCATE 37, 33: PRINT "Feet) :"
```

```
LOCATE 38, 6: PRINT "D-TWRW" =";
LOCATE 38, 15: PRINT USING "#### #"; disttw: LOCATE 38, 22: PRINT "m"
IF disttw / mts > 999 THEN
LOCATE 38, 25: PRINT USING "#####"; disttw / mts
```

```

LOCATE 38, 24: PRINT "(" LOCATE 38, 33: PRINT "Feet) ."
ELSE
LOCATE 38, 26: PRINT USING "### ##"; disttw / mts
LOCATE 38, 25: PRINT "(" LOCATE 38, 33: PRINT "Feet) ."
END IF

' Right side

LOCATE 34, 45: PRINT "RA =";
LOCATE 34, 52: PRINT USING "### ##"; RA: LOCATE 34, 59: PRINT "m"
LOCATE 34, 63: PRINT USING "### ##"; RA / mts
LOCATE 34, 62: PRINT "(" LOCATE 34, 70: PRINT "Feet)"

LOCATE 35, 45: PRINT "RB =";
LOCATE 35, 52: PRINT USING "### ##"; RB: LOCATE 35, 59: PRINT "m"
LOCATE 35, 63: PRINT USING "### ##"; RB / mts
LOCATE 35, 62: PRINT "(" LOCATE 35, 70: PRINT "Feet)"

LOCATE 36, 45: PRINT "RTW =";
LOCATE 36, 52: PRINT USING "### ##"; TR: LOCATE 36, 59: PRINT "m"
LOCATE 36, 63: PRINT USING "### ##"; TR / mts
LOCATE 36, 62: PRINT "(" LOCATE 36, 70: PRINT "Feet)"

LOCATE 37, 45: PRINT "RTWC =";
LOCATE 37, 52: PRINT USING "### ##"; TRC: LOCATE 37, 59: PRINT "m"
LOCATE 37, 63: PRINT USING "### ##"; TRC / mts
LOCATE 37, 62: PRINT "(" LOCATE 37, 70: PRINT "Feet)"

d = 3 * mts

LOCATE 38, 45: PRINT "EXWT =";
LOCATE 38, 52: PRINT USING "### ##"; ETW: LOCATE 38, 59: PRINT "m"
LOCATE 38, 63: PRINT USING "### ##"; ETW / mts
LOCATE 38, 62: PRINT "(" LOCATE 38, 70: PRINT "Feet)"

```

```

CALL fc.inkey(W$)
WIDTH 80, 25
COLOR 15

```

```
END SUB
```

The main purpose of this subroutine is to display the standard 45-degree FAA turnoff. Some of the inputs to this program are obtained by calling in the subroutine air.design.grp and tax.exit.geo while others are specified in the subroutine itself. The subroutine requires the runway width, the distance between the runway and the taxiway and the wing span. The runway width, the taxiway width and the distance between the runway and the taxiway define the scale of this subroutine.

```
SUB geo.new.90
```

```

SCREEN 9
WIDTH 80, 43
COLOR 15, 9
mts = .3048 ' for conversion from feet to meters

```

```

LINE (5, 4)-(635, 334), , B
LINE (232, 20)-(428, 35), 12, BF
LINE (236, 22)-(424, 33), 0, BF
COLOR 15
LOCATE 4, 31: PRINT "90-DEGREE TURNOFF (FAA)"
COLOR 15, 9

```

```

VIEW (64, 24)-(576, 336)
WINDOW (0, 0)-(512, 312)

```

```
' *** ALL DIMENSIONS ARE IN METERS ***
```

```

rw = 40: disttw = 200
RRC = 250 * mts: LC = 200 * mts

```

```

IF rw > (199 * mts) THEN
  RR = 150 * mts
ELSE
  RR = 175 * mts
END IF

```

```

ws = 45
CALL air.design.grp(ws)
CALL tax.exit.geo(des.grp)

```

```

yscale = 169 / ((rw / 2 + tw / 2 + disttw))
xscale = yscale * 1.37

pi = 3.141592

LINE (10, 270)-(502, 270) ' top line of tw
twc = 270 - (tw * yscale / 2)
LINE (10, twc)-(502, twc), 7, , &HFFF2
CALL font(460, twc, 10, "c")
CALL font(462, twc - 3, 10, "l")

' 492 / 2 = 246

x3 = 246 - (R - F) * xscale
x4 = x3 - F * xscale
x5 = x4 - L * xscale

y6 = 270 - tw * yscale
x6 = 10

y5 = y6
LINE (x6, y6)-(x5, y5)

y4 = y5 + (tw / 2 - (R - F)) * yscale
LINE (x5, y5)-(x4, y4)

x4c = x4
y4c = y4 - F * yscale
CIRCLE (x4c, y4c), F * xscale, 15, pi / 180 * 0!, pi / 180 * 90

LINE (10, 101)-(502, 101) ' bottom line of rw
rwc = 101 + (rw / 2 * yscale)
LINE (10, rwc)-(502, rwc), 7, , &HFFF2
CALL font(460, rwc, 10, "c")
CALL font(462, rwc - 3, 10, "l")

x2 = x3
xrrc = x2 - RR * xscale
x1 = xrrc
y1 = 101 + (rw * yscale)
yrrc = y1 + (RR * yscale)
CIRCLE (xrrc, yrrc), RR * xscale, 15, pi / 180 * 270, pi / 180 * 360
LINE (10, y1)-(x1, y1)

y2 = y1 + RR * yscale
y3 = y4 - F * yscale
LINE (x2, y2)-(x3, y3)

' Right side of the figure

xa3 = 246 + (R - F) * xscale
xa4 = xa3 + F * xscale
xa5 = xa4 + L * xscale

ya6 = 270 - tw * yscale
xa6 = 502

ya5 = ya6
LINE (xa6, ya6)-(xa5, ya5)

ya4 = ya5 + (tw / 2 - (R - F)) * yscale
LINE (xa5, ya5)-(xa4, ya4)

xa4c = xa4
ya4c = ya4 - F * yscale
CIRCLE (xa4c, ya4c), F * xscale, 15, pi / 180 * 90, pi / 180 * 180

xa2 = xa3
xarrc = xa2 + RR * xscale
xa1 = xarrc
ya1 = 101 + rw * yscale
yarrc = y1 + RR * yscale
CIRCLE (xarrc, yarrc), RR * xscale, 15, pi / 180 * 180, pi / 180 * 270
LINE (502, ya1)-(xa1, ya1)

ya2 = ya1 + RR * yscale
ya3 = ya4 - F * yscale
LINE (xa2, ya2)-(xa3, ya3)

```

```

xc3 = x3 + (R - F) * xscale
LINE (xc3, y2)-(xc3, y3), 7, , &HFFF2

CIRCLE (x1c, y1c), R * xscale, 7, pi / 180 * 0!, pi / 180 * 90
CIRCLE (xa1c, ya1c), R * xscale, 7, pi / 180 * 90, pi / 180 * 180

xscrrc = xc3 - RRC * xscale
yscrrc = rwc + RRC * yscale

CIRCLE (xscrrc, yscrrc), RRC * xscale, 7, pi / 180 * 270, pi / 180 * 360

xascrrc = xc3 + RRC * xscale
yascrrc = rwc + RRC * yscale

CIRCLE (xascrrc, yascrrc), RRC * xscale, 7, pi / 180 * 180, pi / 180 * 270

' for displaying the radius of outer curve (F) for the taxiway

COLOR 13
LINE (x1c, y4)-(x1c, y1c)
LINE (x1c, y4)-(x1c - 3, y4 - 3)
LINE (x1c, y4)-(x1c + 3, y4 - 3)

CALL font(x4 + 5, y4 - 10, 10, "F")

LINE (x5, y4 - 8)-(x4, y4 - 8)
LINE (x5, y4 - 8)-(x5 + 3, y4 - 10)
LINE (x5, y4 - 8)-(x5 + 3, y4 - 6)
LINE (x4, y4 - 8)-(x4 - 3, y4 - 10)
LINE (x4, y4 - 8)-(x4 - 3, y4 - 6)

LINE (x5, y5 - 2)-(x5, y4 - 12)
CALL font(x5 + (x4 - x5) / 2, y4 - 6, 10, "L")

' for displaying the radius of centerline curve (R) for the taxiway

yyR = R * yscale * SIN(45 * pi / 180)
xxR = R * xscale * COS(45 * pi / 180)
LINE (x1c, y1c)-(x1c + xxR, y1c + yyR)
LINE (x1c + xxR, y1c + yyR)-(x1c + xxR - 2, y1c + yyR - 3)
LINE (x1c + xxR, y1c + yyR)-(x1c + xxR - 5, y1c + yyR - 1)

CALL font(x1c + xxR - 5, y1c + yyR - 14, 10, "R")

' for displaying the radius of outer curve for the runway

yyRR = RR * yscale * SIN(25 * pi / 180)
xxRR = RR * xscale * COS(25 * pi / 180)
LINE (xrrc, yrrc)-(xrrc + xxRR, yrrc - yyRR)
LINE (xrrc + xxRR, yrrc - yyRR)-(xrrc + xxRR - 4, yrrc - yyRR + 3)
LINE (xrrc + xxRR, yrrc - yyRR)-(xrrc + xxRR - 6, yrrc - yyRR - 1)

yy2RR = RR / 2 * yscale * SIN(55 * pi / 180)
xx2RR = RR / 2 * xscale * COS(55 * pi / 180)

CALL font(xrrc + 40 - 24, yrrc - yy2RR + 16, 10, "R")
CALL font(xrrc + 40 - 19, yrrc - yy2RR + 16, 10, "R")
CALL font(xrrc + 40 - 14, yrrc - yy2RR + 16, 10, "W")

' for displaying the radius of centerline curve for the runway

yyRRC = RRC * yscale * SIN(55 * pi / 180)
xxRRC = RRC * xscale * COS(55 * pi / 180)
LINE (xscrrc, yscrrc)-(xscrrc + xxRRC, yscrrc - yyRRC)
LINE (xscrrc + xxRRC, yscrrc - yyRRC)-(xscrrc + xxRRC - 2, yscrrc - yyRRC + 5)
LINE (xscrrc + xxRRC, yscrrc - yyRRC)-(xscrrc + xxRRC - 6, yscrrc - yyRRC + 3)

yy2RRC = RRC / 2 * yscale * SIN(55 * pi / 180)
xx2RRC = RRC / 2 * xscale * COS(55 * pi / 180)
CALL font(xscrrc + 10 - 24, yscrrc - yy2RRC + 4, 10, "R")
CALL font(xscrrc + 10 - 19, yscrrc - yy2RRC + 4, 10, "R")
CALL font(xscrrc + 10 - 14, yscrrc - yy2RRC + 4, 10, "W")
CALL font(xscrrc + 10 - 9, yscrrc - yy2RRC + 4, 10, "C")

' for displaying the width of taxiway

LINE (x5 - 30, y5)-(x5 - 30, y5 - 15)

```

```

LINE (x5 - 30, y5)-(x5 - 27, y5 - 3)
LINE (x5 - 30, y5)-(x5 - 33, y5 - 3)

LINE (x5 - 30, 270)-(x5 - 30, 285)
LINE (x5 - 30, 270)-(x5 - 27, 270 + 3)
LINE (x5 - 30, 270)-(x5 - 33, 270 + 3)

CALL font(x5 - 42, 270 + 9, 10, "w")
CALL font(x5 - 37, 270 + 9, 10, "t")

' for displaying the extended centerline of the runway for 'LC' (200 feet)

LINE (x1, y1 - rw * yscale / 5)-(x1 - LC * yscale, y1 - rw * yscale / 5)
LINE (x1, y1 - rw * yscale / 5)-(x1 - 3, y1 + 2 - rw * yscale / 5)
LINE (x1, y1 - rw * yscale / 5)-(x1 - 3, y1 - 2 - rw * yscale / 5)
LINE (x1 - LC * yscale, y1 - rw * yscale / 5)-(x1 - LC * yscale + 3, y1 + 2 - rw * yscale / 5)
LINE (x1 - LC * yscale, y1 - rw * yscale / 5)-(x1 - LC * yscale + 3, y1 - 2 - rw * yscale / 5)

LINE (x1, y1 - 1)-(x1, y1 - 4 - rw * yscale / 5)
LINE (x1 - LC * yscale, y1 - 1)-(x1 - LC * yscale, y1 - 4 - rw * yscale / 5)

FOR x = 1 TO LEN("LC")
  q$ = MID$("LC", x, 1)
  CALL font(x1 - 130 * mts * yscale + 5 * x, y1 - rw * yscale / 5 + 3, 10, q$)
NEXT x

CALL font(x1 + 6, y1 - rw * yscale / 5 + 3, 10, "p")
CALL font(x1 + 12, y1 - rw * yscale / 5 + 3, 10, "c")

' for the separation between the above line and the centerline (3 feet)

LINE (x1, rwc + 1)-(x1 - LC * yscale, rwc + 1), 7, , &HFFF2
LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale, rwc + rw * yscale / 5)
LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale, rwc - rw * yscale / 5)
LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale - 2, rwc - 3 - 1)
LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale + 2, rwc - 3 - 1)
LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale - 2, rwc + 2 + 3)
LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale + 2, rwc + 2 + 3)

CALL font(x1 - LC / 2 * yscale + 6, rwc - 10, 10, "D")
LINE (xa1, rwc + 1)-(xa1 + LC * yscale, rwc + 1), 7, , &HFFF2

' for the arrows to display the width of runway

LINE (35, y1)-(35, y1 + 15)
LINE (35, y1)-(37, y1 + 3)
LINE (35, y1)-(33, y1 + 3)
LINE (35, 85)-(35, 100)
LINE (35, 100)-(38, 98)
LINE (35, 100)-(32, 98)

CALL font(42, y1 + 8, 10, "W")
CALL font(47, y1 + 8, 10, "R")

LINE (0, twc)-(9, twc)
LINE (0, rwc)-(9, rwc)
LINE (4, twc)-(4, rwc)
LINE (4, twc)-(1, twc - 3)
LINE (4, twc)-(7, twc - 3)
LINE (4, rwc)-(1, rwc + 3)
LINE (4, rwc)-(7, rwc + 3)

FOR x = 1 TO LEN("D-TWRW")
  t$ = MID$("D-TWRW", x, 1)
  CALL font(4 + 5 * x, y1 + (distlw - (tw + rw) / 2) * yscale / 2, 10, t$)
NEXT x

COLOR 10
LOCATE 31, 63: PRINT "RUNWAY";
LOCATE 8, 62: PRINT "TAXIWAY"

COLOR 15
a = rw / 2 * yscale
b = tw / 2 * yscale

LINE (10, y1 - a + a * .1 * a)-(10, y1 - a + a * .3 * a)
LINE (10, y1 - a + a * .3 * a)-(15, y1 - a + a * .3 * a)

```

```

LINE (15, y1 - a + a - .3 * a)-(5, y1 - a + a - .7 * a)
LINE (5, y1 - a + a - .7 * a)-(10, y1 - a + a - .3 * a)
LINE (10, y1 - a + a - .7 * a)-(10, y1 - a - .3 * a)
LINE (10, y1 - a - .3 * a)-(15, y1 - a - .3 * a)
LINE (15, y1 - a - .3 * a)-(5, y1 - a - .7 * a)
LINE (5, y1 - a - .7 * a)-(10, y1 - a - .7 * a)
LINE (10, y1 - a - .7 * a)-(10, y1 - a - a - .1 * a)

LINE (502, y1 - a + a + .1 * a)-(502, y1 - a + a - .3 * a)
LINE (502, y1 - a + a - .3 * a)-(507, y1 - a + a - .3 * a)
LINE (507, y1 - a + a - .3 * a)-(497, y1 - a + a - .7 * a)
LINE (497, y1 - a + a - .7 * a)-(502, y1 - a + a - .7 * a)
LINE (502, y1 - a + a - .7 * a)-(502, y1 - a - .3 * a)
LINE (502, y1 - a - .3 * a)-(507, y1 - a - .3 * a)
LINE (507, y1 - a - .3 * a)-(497, y1 - a - .7 * a)
LINE (497, y1 - a - .7 * a)-(502, y1 - a - .7 * a)
LINE (502, y1 - a - .7 * a)-(502, y1 - a - a - .1 * a)

LINE (10, y5 + b + b + .3 * b)-(10, y5 + b + b - .3 * b)
LINE (10, y5 + b + b - .3 * b)-(13, y5 + b + b - .3 * b)
LINE (13, y5 + b + b - .3 * b)-(7, y5 + b + b - .7 * b)
LINE (7, y5 + b + b - .7 * b)-(10, y5 + b + b - .7 * b)
LINE (10, y5 + b + b - .7 * b)-(10, y5 + b - .3 * b)
LINE (10, y5 + b - .3 * b)-(13, y5 + b - .3 * b)
LINE (13, y5 + b - .3 * b)-(7, y5 + b - .7 * b)
LINE (7, y5 + b - .7 * b)-(10, y5 + b - .7 * b)
LINE (10, y5 + b - .7 * b)-(10, y5 + b - b - .3 * b)

LINE (502, y5 + b + b + .3 * b)-(502, y5 + b + b - .3 * b)
LINE (502, y5 + b + b - .3 * b)-(505, y5 + b + b - .3 * b)
LINE (505, y5 + b + b - .3 * b)-(499, y5 + b + b - .7 * b)
LINE (499, y5 + b + b - .7 * b)-(502, y5 + b + b - .7 * b)
LINE (502, y5 + b + b - .7 * b)-(502, y5 + b - .3 * b)
LINE (502, y5 + b - .3 * b)-(505, y5 + b - .3 * b)
LINE (505, y5 + b - .3 * b)-(499, y5 + b - .7 * b)
LINE (499, y5 + b - .7 * b)-(502, y5 + b - .7 * b)
LINE (502, y5 + b - .7 * b)-(502, y5 + b - b - .3 * b)

COLOR 11

LOCATE 34, 6: PRINT "WR   =";
LOCATE 34, 15: PRINT USING "### ##"; rw: LOCATE 34, 22: PRINT "m"
LOCATE 34, 26: PRINT USING "### ##"; rw / mts
LOCATE 34, 25: PRINT "(" LOCATE 34, 33: PRINT "Feet)  :\"

LOCATE 35, 6: PRINT "WT   =";
LOCATE 35, 15: PRINT USING "### ##"; tw: LOCATE 35, 22: PRINT "m"
LOCATE 35, 26: PRINT USING "### ##"; tw / mts
LOCATE 35, 25: PRINT "(" LOCATE 35, 33: PRINT "Feet)  :\"

LOCATE 36, 6: PRINT "RRW  =";
LOCATE 36, 15: PRINT USING "### ##"; RR: LOCATE 36, 22: PRINT "m"
LOCATE 36, 26: PRINT USING "### ##"; RR / mts
LOCATE 36, 25: PRINT "(" LOCATE 36, 33: PRINT "Feet)  :\"

LOCATE 37, 6: PRINT "RRWC =";
LOCATE 37, 15: PRINT USING "### ##"; RRC: LOCATE 37, 22: PRINT "m"
LOCATE 37, 26: PRINT USING "### ##"; RRC / mts
LOCATE 37, 25: PRINT "(" LOCATE 37, 33: PRINT "Feet)  :\"

LOCATE 38, 6: PRINT "D-TWRW =";
LOCATE 38, 15: PRINT USING "### ##"; disttw: LOCATE 38, 22: PRINT "m"
IF disttw / mts > 999 THEN
LOCATE 38, 25: PRINT USING "#### ##"; disttw / mts
LOCATE 38, 24: PRINT "(" LOCATE 38, 33: PRINT "Feet)  :\"
ELSE
LOCATE 38, 26: PRINT USING "### ##"; disttw / mts
LOCATE 38, 25: PRINT "(" LOCATE 38, 33: PRINT "Feet)  :\"
END IF

' Right side

LOCATE 34, 45: PRINT "L   =";
LOCATE 34, 52: PRINT USING "### ##"; L: LOCATE 34, 59: PRINT "m"
LOCATE 34, 63: PRINT USING "### ##"; L / mts
LOCATE 34, 62: PRINT "(" LOCATE 34, 70: PRINT "Feet)\"

LOCATE 35, 45: PRINT "F   =";
LOCATE 35, 52: PRINT USING "### ##"; F: LOCATE 35, 59: PRINT "m"

```

```

LOCATE 35, 63: PRINT USING "### ##"; F / mts
LOCATE 35, 62: PRINT "(" LOCATE 35, 70: PRINT "Feet)"

LOCATE 36, 45: PRINT "R  =";
LOCATE 36, 52: PRINT USING "### ##"; R: LOCATE 36, 59: PRINT "m"
LOCATE 36, 63: PRINT USING "### ##"; R / mts
LOCATE 36, 62: PRINT "(" LOCATE 36, 70: PRINT "Feet)"

LOCATE 37, 45: PRINT "LC  =";
LOCATE 37, 52: PRINT USING "### ##"; LC: LOCATE 37, 59: PRINT "m"
LOCATE 37, 63: PRINT USING "### ##"; LC / mts
LOCATE 37, 62: PRINT "(" LOCATE 37, 70: PRINT "Feet)"

d = 3 ^ mts

LOCATE 38, 45: PRINT "D  =";
LOCATE 38, 52: PRINT USING "### ##"; d: LOCATE 38, 59: PRINT "m"
LOCATE 38, 63: PRINT USING "### ##"; d / mts
LOCATE 38, 62: PRINT "(" LOCATE 38, 70: PRINT "Feet)"

```

```

CALL fc.inkey(W$)
WIDTH 80, 25
COLOR 15

```

```
END SUB
```

This subroutine is similar to the subroutines geo.new.30 and geo.new.45. Here too, the inputs required are obtained by calling in the subroutines air.design.grp and tax.exit.geo and the others are specified in the subroutine itself. But the wing span, the width of runway, and the distance between the taxiway and the runway are to be supplied at the start of this subroutine.

```
SUB geo.var
```

```

' this subroutine displays the variable turnoff geometry
' developed by REDIM

```

```

SCREEN 9
WIDTH 80, 43
COLOR 15, 9
mts = .3048 ' for conversion from feet to meters

```

```

LINE (5, 4)-(635, 332), , B
LINE (184, 20)-(475, 35), 12, BF
LINE (188, 22)-(471, 33), 0, BF
COLOR 15
LOCATE 4, 25: PRINT "VARIABLE HIGH-SPEED TURNOFF (REDIM)"
COLOR 15, 9

```

```

VIEW (24, 24)-(616, 336)
WINDOW (0, 0)-(590, 312)

```

```

rw = 60: distlw = 200:
l1 = 100 'in meters

```

```

LC = 200 ^ mts
rrc1 = 2000 ^ mts: rrc2 = 1700 ^ mts
TR = 800 ^ mts: TRC = 800 ^ mts
RA = 25 ^ mts: RB = 25 ^ mts

```

```
theta = 55 'the angle of the turnoff
```

```

IF theta > 45 THEN
    theta = 45
ELSEIF theta < 10 THEN
    theta = 10
END IF

```

```
startpt = 625 - 10 ^ theta
```

```

ws = 35
CALL air.design.grp(ws)
CALL tax.exit.geo(des.grp)

```

```

trackmax = 20 ' (R - F)
RR1 = rrc1: RR2 = rrc2 - trackmax ^ mts

```

```

pi = 3.141593
brdth = (tw / 2 + rw / 2 + disttw) / (TAN(pi / 180 * theta))

xscale = startpt / brdth
yscale = xscale / 1.37

' to display the tw & the centerline

LINE (10, 270)-(582, 270)
twc = 270 - tw / 2 * yscale
LINE (10, twc)-(582, twc), 7, , &HFFF2

rwc = twc - disttw * yscale
LINE (10, rwc - rw / 2 * yscale)-(582, rwc - rw / 2 * yscale)
LINE (10, rwc)-(582, rwc), 7, , &HFFF2
y1 = rwc + rw / 2 * yscale

' to find theta1 (angle of circle for r1)

theta1 = (l1 / rrc1) * yscale * 180 / pi

' to draw the centerline with r1
' -----
xscrrc1 = 75 ' an arbitrary number for the start of turn
yscrrc1 = rwc + rrc1 * yscale

CIRCLE (xscrrc1, yscrrc1), rrc1 * xscale, 7, pi / 180 * 270, pi / 180 * (270 + theta1)

' to locate the end of centerline r1

xrrc1end = xscrrc1 + SIN(pi / 180 * theta1) * rrc1 * xscale
yrrc1end = yscrrc1 - COS(pi / 180 * theta1) * rrc1 * yscale

' to locate the center of circle for r2 and to draw it

xscrrc2 = xrrc1end - SIN(pi / 180 * theta1) * rrc2 * xscale
yscrrc2 = yrrc1end + COS(pi / 180 * theta1) * rrc2 * yscale

CIRCLE (xscrrc2, yscrrc2), rrc2 * xscale, 7, pi / 180 * (270 + theta1), pi / 180 * (270 + theta)

' to locate the end of centerline r2

theta2 = theta - theta1

xrrc2end = xscrrc2 + SIN(pi / 180 * theta) * rrc2 * xscale
yrrc2end = yscrrc2 - COS(pi / 180 * theta) * rrc2 * yscale

' to draw the centerline of the exit taxiway from the end of r2
' till the centerline of taxiway

xtwsc = xrrc2end + (twc - yrrc2end) * 1.37 / TAN(pi / 180 * theta)
xrwsc = xrrc2end - (yrrc2end - rwc) * 1.37 / TAN(pi / 180 * theta)

LINE (xrrc2end, yrrc2end)-(xtwsc, twc), 7, , &HFFF2
LINE (xrrc2end, yrrc2end)-(xrwsc, rwc), 7, , &HFFF2

' to draw the line that joins the centerlines of tw & exit tw

xasc4 = xtwsc + TRC * xscale * TAN(pi / 180 * (theta / 2))
yasc4 = twc - TRC * yscale

CIRCLE (xasc4, yasc4), TRC * xscale, 7, pi / 180 * 90, pi / 180 * (90 + theta)

' for locating a point on the centerline of the exit tw
' (this helps in building the width of exit tw)

yyy = twc - disttw / 3 * yscale
xxx = xtwsc - (disttw / 3 * yscale) / TAN(pi / 180 * theta) * 1.37

'right side of xxx at width of exit tw equal to (R-F)

xextwrgt = xxx + (R - F) * xscale / COS(pi / 180 * (90 - theta))

' to draw the top right curve joining rw and the exit tw

ya1 = y1
yas1 = ya1 + RB * yscale
ya2 = yas1 + COS(pi / 180 * (90 - theta))

```

```

xa2 = xextwrgt - (yyy - ya2) * 1.37 / TAN(pi / 180 * theta)
LINE (xextwrgt, yyy)-(xa2, ya2), 15

xas1 = xa2 + RB * xscale * SIN(pi / 180 * (90 - theta))
CIRCLE (xas1, yas1), RB * xscale, 15, pi / 180 * (90 + (90 - theta)), pi / 180 * 270

' to join the top (right side) line of rw

xa1 = xas1
LINE (xa1, ya1)-(582, ya1)

' to draw the top right curve joining the tw

y4 = twc - tw / 2 * yscale
ya4 = y4
yasc3 = ya4 - TR * yscale
ya3 = yasc3 + COS(pi / 180 * theta) * TR * yscale
xa3 = xextwrgt + (ya3 - yyy) / (TAN(pi / 180 * theta)) * 1.37

' continuation of right side of exit tw
LINE (xa3, ya3)-(xextwrgt, yyy)

xasc3 = xa3 + SIN(pi / 180 * theta) * TR * xscale
CIRCLE (xasc3, yasc3), TR * xscale, 15, pi / 180 * 90, pi / 180 * (90 + theta)

xa4 = xasc3

' bottom (right side) line of tw
LINE (xa4, ya4)-(582, ya4)

' to draw the curves with R and F for the exit tw

Ty2 = twc - R * yscale
Ty1 = Ty2 - COS(pi / 180 * theta) * R * yscale
Tx1 = xrwsc + (Ty1 - rwc) * 1.37 / (TAN(pi / 180 * theta))
Tx2 = Tx1 - R * xscale * SIN(pi / 180 * theta)

CIRCLE (Tx2, Ty2), R * xscale, 7, pi / 180 * (270 + theta), pi / 180 * 90
CIRCLE (Tx2, Ty2), F * xscale, 7, pi / 180 * (270 + theta), pi / 180 * 90

' for the fillets

xf5 = Tx2
yf5 = Ty2 + F * yscale
xl6 = xf5 - L * xscale
yl6 = y4

LINE (xf5, yf5)-(xl6, yl6), 7

' to locate the point on the left side of the exit tw
' from the center of fillet curve

xx0 = Tx2 + F * COS(pi / 180 * (90 - theta)) * xscale
yy0 = Ty2 - F * SIN(pi / 180 * (90 - theta)) * yscale

' to locate the point of intersection of the left side of exit tw
' and the bottom line of tw

yy1 = y4
xx1 = xx0 + (yy1 - yy0 + TAN(pi / 180 * theta) * xx1) / TAN(pi / 180 * theta)

' curve to join the left side of exit tw with the taxiway

xs4 = xx1 - RA * xscale * COS(pi / 180 * theta)
ys4 = y4 - RA * yscale

CIRCLE (xs4, ys4), RA * xscale, 15, pi / 180 * (270 + (90 - theta)), pi / 180 * 90

' to join the bottom (left side) of the tw

x4 = xs4

```

```

LINE (10, y4)-(x4, y4)

' to join the top left part of the exit tw

x3 = xs4 + RA * xscale * COS(pi / 180 * theta)
y3 = ys4 - RA * yscale * SIN(pi / 180 * theta)

LINE (x3, y3)-(xx0, yy0)

'=====

' to draw the outline of rw with R1

xrr1 = xscrrc1          ' an arbitrary number for the start of turn
yrr1 = y1 + RR1 * yscale

' to draw the left portion of the top line of rw

LINE (10, y1)-(xrr1, y1)

' to locate the end of outer curve with R1

xrr1end = xrr1 + SIN(pi / 180 * theta1) * RR1 * xscale
yrr1end = yrr1 - COS(pi / 180 * theta1) * RR1 * yscale

' to locate the center of circle for R2 and to draw it

xrr2 = xrr1end - SIN(pi / 180 * theta1) * RR2 * xscale
yrr2 = yrr1end + COS(pi / 180 * theta1) * RR2 * yscale

' to locate the end of centerline R2

'theta2 = theta - theta1

xrr2end = xrr2 + SIN(pi / 180 * theta) * RR2 * xscale
yrr2end = yrr2 - COS(pi / 180 * theta) * RR2 * yscale

' to join the end of outer (R2) curve with the fillet curve

yy2 = yrr2end
xx2 = xx0 - (yy0 - yrr2end) * 1.37 / TAN(pi / 180 * theta)

LINE (xx2, yy2)-(xx0, yy0)

xr2new = xx2 - RR2 * xscale * SIN(pi / 180 * theta)
yr2new = yy2 + RR2 * yscale * COS(pi / 180 * theta)

' these complete the R1 and R2 (for the outer part)
' joining rw with the exit tw

CIRCLE (xr2new, yr2new), RR2 * xscale, 15, pi / 180 * (270 + theta1), pi / 180 * (270 + theta)

xx3 = xr2new + RR2 * xscale * SIN(pi / 180 * (theta1))
yy3 = yr2new - RR2 * yscale * COS(pi / 180 * (theta1))

xr1new = xx3 - SIN(pi / 180 * theta1) * RR1 * xscale
yr1new = yy3 + COS(pi / 180 * theta1) * RR1 * yscale

'CIRCLE (xr1new, yr1new), RR1 * xscale, 15, pi / 180 * 267, pi / 180 * (270 + theta1)

' ** ** circle statement could be used ** **

LINE (xrr1, y1)-(xx3, yy3 - 1) '-----

'=====

COLOR 13          ' color used for arrows

LINE (Tx2, yf5)-(Tx2, Ty2)
LINE (Tx2, yf5)-(Tx2 - 3, yf5 - 3)
LINE (Tx2, yf5)-(Tx2 + 3, yf5 - 3)

' for displaying the length 'L' of the fillet

```

```

LINE (xf5, yf5 - 8)-(xl6, yf5 - 8)
LINE (xf5, yf5 - 8)-(xf5 - 3, yf5 - 10)
LINE (xf5, yf5 - 8)-(xf5 - 3, yf5 - 6)
LINE (xl6, yf5 - 8)-(xl6 + 3, yf5 - 10)
LINE (xl6, yf5 - 8)-(xl6 + 3, yf5 - 6)

LINE (xl6, yf6 - 3)-(xl6, yf6 - 22)
CALL font(Tx2 + 5, yf5 - 10, 10, "F")
CALL font(xf5 + (xl6 - xf5) / 2, yf5 - 6, 10, "L")

' for displaying the radius of centerline curve {R} for the taxiway

yyR = R * yscale * SIN(25 * pi / 180)
xxR = R * xscale * COS(25 * pi / 180)
LINE (Tx2, Ty2)-(Tx2 + xxR, Ty2 + yyR)
LINE (Tx2 + xxR, Ty2 + yyR)-(Tx2 + xxR - 3, Ty2 + yyR - 3)
LINE (Tx2 + xxR, Ty2 + yyR)-(Tx2 + xxR - 5, Ty2 + yyR + 1)

CALL font(Tx2 + xxR - 8, Ty2 + yyR - 11, 10, "R")

' for displaying the width of taxiway

LINE (60 - 30, y4)-(60 - 30, y4 - 15)
LINE (60 - 30, y4)-(60 - 27, y4 - 3)
LINE (60 - 30, y4)-(60 - 33, y4 - 3)

LINE (60 - 30, 270)-(60 - 30, 285)
LINE (60 - 30, 270)-(60 - 27, 270 + 3)
LINE (60 - 30, 270)-(60 - 33, 270 + 3)

CALL font(78 - 42, 270 + 9, 10, "W")
CALL font(78 - 37, 270 + 9, 10, "T")

' for the arrows to display the width of runway

LINE (535, y1)-(535, y1 + 15)
LINE (535, y1)-(537, y1 + 3)
LINE (535, y1)-(533, y1 + 3)
LINE (535, y1 - rw * yscale)-(535, y1 - rw * yscale - 15)
LINE (535, y1 - rw * yscale)-(538, y1 - rw * yscale - 2)
LINE (535, y1 - rw * yscale)-(532, y1 - rw * yscale - 2)

CALL font(542, y1 + 8, 10, "W")
CALL font(547, y1 + 8, 10, "R")

' to display the distance between the rw & tw

LINE (0, twc)-(9, twc)
LINE (0, rwc)-(9, rwc)
LINE (4, twc)-(4, rwc)
LINE (4, twc)-(1, twc - 3)
LINE (4, twc)-(7, twc - 3)
LINE (4, rwc)-(1, rwc + 3)
LINE (4, rwc)-(7, rwc + 3)

FOR x = 1 TO LEN("D-TWRW")
    t$ = MID$("D-TWRW", x, 1)
    CALL font(4 + 5 * x, y1 + (distfw - (tw + rw) / 2) * yscale / 2, 10, t$)
NEXT x

' to display the radius of curve joining bottom line (left) of tw

yiR = RA * yscale * SIN(40 * pi / 180)
xiR = RA * xscale * COS(40 * pi / 180)
LINE (xs4, ys4)-(xs4 + xiR, ys4 + yiR)
LINE (xs4 + xiR, ys4 + yiR)-(xs4 + xiR - 2, ys4 + yiR - 3)
LINE (xs4 + xiR, ys4 + yiR)-(xs4 + xiR - 5, ys4 + yiR - 1)

CALL font(xs4 - 12, ys4 - 4, 10, "R")
CALL font(xs4 - 7, ys4 - 4, 10, "A")

' to display the radius of curve joining top line (right) of rw

xiR = RB * xscale * SIN(45 * pi / 180)
yiR = RB * yscale * COS(45 * pi / 180)
LINE (xas1, yas1)-(xas1 - xiR, yas1 - yiR)
LINE (xas1 - xiR, yas1 - yiR)-(xas1 - xiR + 2, yas1 - yiR + 3)
LINE (xas1 - xiR, yas1 - yiR)-(xas1 - xiR + 5, yas1 - yiR + 1)

```

```

CALL font(xas1 + xlr + 3, yas1 + ylr - 2, 10, "R")
CALL font(xas1 + xlr + 8, yas1 + ylr - 2, 10, "B")

' to display 'cl' on the centerlines of tw & rw

CALL font(550, twc, 10, "c")
CALL font(552, twc - 3, 10, "l")

CALL font(550, rwc, 10, "c")
CALL font(552, rwc - 3, 10, "l")

' for displaying the extended centerline of the runway for 'LC' (200 feet)

x1 = xscrrc1 '-----

LINE (x1, y1 - rw * yscale / 8)-(x1 - LC * yscale, y1 - rw * yscale / 8)
LINE (x1, y1 - rw * yscale / 8)-(x1 - 3, y1 + 2 - rw * yscale / 8)
LINE (x1, y1 - rw * yscale / 8)-(x1 - 3, y1 - 2 - rw * yscale / 8)
LINE (x1 - LC * yscale, y1 - rw * yscale / 8)-(x1 - LC * yscale + 3, y1 + 2 - rw * yscale / 8)
LINE (x1 - LC * yscale, y1 - rw * yscale / 8)-(x1 - LC * yscale + 3, y1 - 2 - rw * yscale / 8)

LINE (x1, y1 - 1)-(x1, y1 - 4 - rw * yscale / 8)
LINE (x1 - LC * yscale, y1 - 1)-(x1 - LC * yscale, y1 - 4 - rw * yscale / 8)

FOR x = 1 TO LEN("LC")
q$ = MID$("LC", x, 1)
CALL font(x1 - 170 * mts * yscale + 5 * x, y1 - rw * yscale / 8 + 6, 10, q$)
NEXT x

CALL font(x1 + 6, y1 - rw * yscale / 8 + 6, 10, "P")
CALL font(x1 + 12, y1 - rw * yscale / 8 + 6, 10, "C")

' for the separation between the above line and the centerline (3 feet)

LINE (x1, rwc + 1)-(x1 - LC * yscale, rwc + 1), 7, , &HFFF2
LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale, rwc + rw * yscale / 4)
LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale, rwc - rw * yscale / 4)
LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale - 2, rwc - 3 - 1)
LINE (x1 - LC / 2 * yscale, rwc - 1)-(x1 - LC / 2 * yscale + 2, rwc - 3 - 1)
LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale - 2, rwc + 2 + 3)
LINE (x1 - LC / 2 * yscale, rwc + 2)-(x1 - LC / 2 * yscale + 2, rwc + 2 + 3)

CALL font(x1 - LC / 2 * yscale + 6, rwc - 10, 10, "D")

' to display the radius of curve joining centerline of tw

xas4 = xasc4: yas4 = yasc4

xjr = TRC * xscale * SIN(15 * pi / 180)
yjr = TRC * yscale * COS(15 * pi / 180)

LINE (xas4 - xjr, yas4 + yjr)-(xas4 - xjr + (xjr / 3), yas4 + yjr - xjr / 3 / 1.37 * TAN(pi / 180 * 60))
LINE (xas4 - xjr, yas4 + yjr)-(xas4 - xjr - 1, yas4 + yjr - 3)
LINE (xas4 - xjr, yas4 + yjr)-(xas4 - xjr + 6, yas4 + yjr - 1)

CALL font(xas4 - xjr * 1.5 + (xjr / 3), yas4 + yjr - xjr / 1.5 / 1.37, 10, "R")
CALL font(xas4 - xjr * 1.5 + 5 + (xjr / 3), yas4 + yjr - xjr / 1.5 / 1.37, 10, "T")
CALL font(xas4 - xjr * 1.5 + 10 + (xjr / 3), yas4 + yjr - xjr / 1.5 / 1.37, 10, "W")
CALL font(xas4 - xjr * 1.5 + 15 + (xjr / 3), yas4 + yjr - xjr / 1.5 / 1.37, 10, "C")

' to display the radius of curve joining top line (right) of tw

xas3 = xasc3: yas3 = yasc3

xkR = TR * xscale * SIN(15 * pi / 180)
ykR = TR * yscale * COS(15 * pi / 180)

LINE (xas3 - xkR, yas3 + ykR)-(xas3 - xkR + (xkR / 3), yas3 + ykR - xkR / 3 / 1.37 * TAN(pi / 180 * 60))
LINE (xas3 - xkR, yas3 + ykR)-(xas3 - xkR - 1, yas3 + ykR - 3)
LINE (xas3 - xkR, yas3 + ykR)-(xas3 - xkR + 6, yas3 + ykR - 1)

CALL font(xas3 - xkR * .94 + (xkR / 3), yas3 + ykR * 1.04 - xkR / 2 / 1.37, 10, "R")
CALL font(xas3 - xkR * .94 + 5 + (xkR / 3), yas3 + ykR * 1.04 - xkR / 2 / 1.37, 10, "T")
CALL font(xas3 - xkR * .94 + 10 + (xkR / 3), yas3 + ykR * 1.04 - xkR / 2 / 1.37, 10, "W")

' to print the value of theta (30) -----

```

```

CIRCLE (xrwsc, rwc), rw * xscale / 2, 13, pi / 180 * 0, pi / 180 * theta
CIRCLE (xrwsc + rw * xscale / 2 + 16, rwc + 9), 1, 10

CALL font(xrwsc + rw * xscale / 2 + 2, rwc + 3, 10, "3")
CALL font(xrwsc + rw * xscale / 2 + 8, rwc + 3, 10, "0")

' to display the radius of curve joining the centerline of rw (r1)

xmr = rrc1 * xscale * SIN(theta1 * pi / 180)
ymr = rrc1 * yscale * COS(theta1 * pi / 180)

xcrr = xscrrc1: ycrr = yscrrc1

LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr - xmr / 3, ycrr - ymr + xmr / 3 / 1.37 * TAN(pi / 180 * 60))
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr + 1, ycrr - ymr + 3)
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr - 4, ycrr - ymr + 1)

CALL font(xcrr + 3 + xmr - xmr / 2, ycrr - ymr + xmr / 2 / 1.37 + 3, 10, "R")
CALL font(xcrr + 8 + xmr - xmr / 2, ycrr - ymr + xmr / 2 / 1.37 + 3, 10, "R")
CALL font(xcrr + 13 + xmr - xmr / 2, ycrr - ymr + xmr / 2 / 1.37 + 3, 10, "W")
CALL font(xcrr + 18 + xmr - xmr / 2, ycrr - ymr + xmr / 2 / 1.37 + 3, 10, "C")
CALL font(xcrr + 23 + xmr - xmr / 2, ycrr - ymr + xmr / 2 / 1.37 + 3, 10, "1")

' to display the radius of curve joining the centerline of rw (r2)

xmr = rrc2 * xscale * SIN(.7 * theta2 * pi / 180)
ymr = rrc2 * yscale * COS(.7 * theta2 * pi / 180)

xcrr = xscrrc2: ycrr = yscrrc2

LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr - xmr / 8, ycrr - ymr + xmr / 8 / 1.37 * TAN(pi / 180 * 60))
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr + 1, ycrr - ymr + 3)
LINE (xcrr + xmr, ycrr - ymr)-(xcrr + xmr - 4, ycrr - ymr + 1)

CALL font(xcrr + 3 + xmr - xmr / 4, ycrr - ymr + xmr / 4 / 1.37, 10, "R")
CALL font(xcrr + 8 + xmr - xmr / 4, ycrr - ymr + xmr / 4 / 1.37, 10, "R")
CALL font(xcrr + 13 + xmr - xmr / 4, ycrr - ymr + xmr / 4 / 1.37, 10, "W")
CALL font(xcrr + 18 + xmr - xmr / 4, ycrr - ymr + xmr / 4 / 1.37, 10, "C")
CALL font(xcrr + 23 + xmr - xmr / 4, ycrr - ymr + xmr / 4 / 1.37, 10, "2")

' to display the radius of curve joining the outer line of rw (r1)

xur = RR1 * xscale * SIN(theta1 * pi / 180)
yur = RR1 * yscale * COS(theta1 * pi / 180)

xcrr = xr1new: ycrr = yr1new

LINE (xcrr + xur, ycrr - yur)-(xcrr + xur - xur / 3, ycrr - yur + xur / 3 / 1.37 * TAN(pi / 180 * 60))
LINE (xcrr + xur, ycrr - yur)-(xcrr + xur + 1, ycrr - yur + 3)
LINE (xcrr + xur, ycrr - yur)-(xcrr + xur - 4, ycrr - yur + 1)

CALL font(xcrr + 3 + xur - xur / 2, ycrr - yur + xur / 2 / 1.37 + 3, 10, "R")
CALL font(xcrr + 8 + xur - xur / 2, ycrr - yur + xur / 2 / 1.37 + 3, 10, "R")
CALL font(xcrr + 13 + xur - xur / 2, ycrr - yur + xur / 2 / 1.37 + 3, 10, "W")
CALL font(xcrr + 18 + xur - xur / 2, ycrr - yur + xur / 2 / 1.37 + 3, 10, "1")

' to display the radius of curve joining the outer line of rw (r2)

xvr = RR2 * xscale * SIN(.7 * theta2 * pi / 180)
yvr = RR2 * yscale * COS(.7 * theta2 * pi / 180)

xcrr = xr2new: ycrr = yr2new

LINE (xcrr + xvr, ycrr - yvr)-(xcrr + xvr - xvr / 8, ycrr - yvr + xvr / 8 / 1.37 * TAN(pi / 180 * 60))
LINE (xcrr + xvr, ycrr - yvr)-(xcrr + xvr + 1, ycrr - yvr + 3)
LINE (xcrr + xvr, ycrr - yvr)-(xcrr + xvr - 4, ycrr - yvr + 1)

CALL font(xcrr + 3 + xvr - xvr / 4, ycrr - yvr + xvr / 4 / 1.37, 10, "R")
CALL font(xcrr + 8 + xvr - xvr / 4, ycrr - yvr + xvr / 4 / 1.37, 10, "R")
CALL font(xcrr + 13 + xvr - xvr / 4, ycrr - yvr + xvr / 4 / 1.37, 10, "W")
CALL font(xcrr + 18 + xvr - xvr / 4, ycrr - yvr + xvr / 4 / 1.37, 10, "2")

COLOR 10
LOCATE 8, 62: PRINT "TAXIWAY";

```

```

    ROW = (43 - (y1 - rw * yscale) / 8.14)
    LOCATE ROW, 62: PRINT "RUNWAY";

COLOR 15
a = rw / 2 * yscale
b = tw / 2 * yscale

LINE (10, y1 - a + a + .1 * a)-(10, y1 - a + a - .3 * a)
LINE (10, y1 - a + a - .3 * a)-(15, y1 - a + a - .3 * a)
LINE (15, y1 - a + a - .3 * a)-(5, y1 - a + a - .7 * a)
LINE (5, y1 - a + a - .7 * a)-(10, y1 - a + a - .7 * a)
LINE (10, y1 - a + a - .7 * a)-(10, y1 - a - .3 * a)
LINE (10, y1 - a - .3 * a)-(15, y1 - a - .3 * a)
LINE (15, y1 - a - .3 * a)-(5, y1 - a - .7 * a)
LINE (5, y1 - a - .7 * a)-(10, y1 - a - .7 * a)
LINE (10, y1 - a - .7 * a)-(10, y1 - a - a - .1 * a)

LINE (582, y1 - a + a + .1 * a)-(582, y1 - a + a - .3 * a)
LINE (582, y1 - a + a - .3 * a)-(587, y1 - a + a - .3 * a)
LINE (587, y1 - a + a - .3 * a)-(577, y1 - a + a - .7 * a)
LINE (577, y1 - a + a - .7 * a)-(582, y1 - a + a - .7 * a)
LINE (582, y1 - a + a - .7 * a)-(582, y1 - a - .3 * a)
LINE (582, y1 - a - .3 * a)-(587, y1 - a - .3 * a)
LINE (587, y1 - a - .3 * a)-(577, y1 - a - .7 * a)
LINE (577, y1 - a - .7 * a)-(582, y1 - a - .7 * a)
LINE (582, y1 - a - .7 * a)-(582, y1 - a - a - .1 * a)

LINE (10, y4 + b + b + .3 * b)-(10, y4 + b + b - .3 * b)
LINE (10, y4 + b + b - .3 * b)-(13, y4 + b + b - .3 * b)
LINE (13, y4 + b + b - .3 * b)-(7, y4 + b + b - .7 * b)
LINE (7, y4 + b + b - .7 * b)-(10, y4 + b + b - .7 * b)
LINE (10, y4 + b + b - .7 * b)-(10, y4 + b - .3 * b)
LINE (10, y4 + b - .3 * b)-(13, y4 + b - .3 * b)
LINE (13, y4 + b - .3 * b)-(7, y4 + b - .7 * b)
LINE (7, y4 + b - .7 * b)-(10, y4 + b - .7 * b)
LINE (10, y4 + b - .7 * b)-(10, y4 + b - b - .3 * b)

LINE (582, y4 + b + b + .3 * b)-(582, y4 + b + b - .3 * b)
LINE (582, y4 + b + b - .3 * b)-(585, y4 + b + b - .3 * b)
LINE (585, y4 + b + b - .3 * b)-(579, y4 + b + b - .7 * b)
LINE (579, y4 + b + b - .7 * b)-(582, y4 + b + b - .7 * b)
LINE (582, y4 + b + b - .7 * b)-(582, y4 + b - .3 * b)
LINE (582, y4 + b - .3 * b)-(585, y4 + b - .3 * b)
LINE (585, y4 + b - .3 * b)-(579, y4 + b - .7 * b)
LINE (579, y4 + b - .7 * b)-(582, y4 + b - .7 * b)
LINE (582, y4 + b - .7 * b)-(582, y4 + b - b - .3 * b)

ROW = ROW + 1
COLOR 11

RO = ROW + 4

LOCATE RO, 6: PRINT "WR      =";
LOCATE RO, 15: PRINT USING "### ##"; rw: LOCATE RO, 22: PRINT "m"
LOCATE RO, 26: PRINT USING "### ##"; rw / mts
LOCATE RO, 25: PRINT "(": LOCATE RO, 33: PRINT "Feet)  :\"

RO = ROW + 5

LOCATE RO, 6: PRINT "RTW      =";
LOCATE RO, 15: PRINT USING "### ##"; TR: LOCATE RO, 22: PRINT "m"
LOCATE RO, 26: PRINT USING "### ##"; TR / mts
LOCATE RO, 25: PRINT "(": LOCATE RO, 33: PRINT "Feet)  :\"

RO = ROW + 6

LOCATE RO, 6: PRINT "RTWC     =";
LOCATE RO, 15: PRINT USING "### ##"; TRC: LOCATE RO, 22: PRINT "m"
LOCATE RO, 26: PRINT USING "### ##"; TRC / mts
LOCATE RO, 25: PRINT "(": LOCATE RO, 33: PRINT "Feet)  :\"

RO = ROW + 7

LOCATE RO, 6: PRINT "RRW1     =";
LOCATE RO, 15: PRINT USING "### ##"; RR1: LOCATE RO, 22: PRINT "m"
LOCATE RO, 25: PRINT USING "### ##"; RR1 / mts
LOCATE RO, 24: PRINT "(": LOCATE RO, 33: PRINT "Feet)  :\"

```

```

RO = ROW + 8

LOCATE RO, 6: PRINT "RRW2  =";
LOCATE RO, 15: PRINT USING "### ##"; RR2: LOCATE RO, 22: PRINT "m"
LOCATE RO, 25: PRINT USING "#####"; RR2 / mts
LOCATE RO, 24: PRINT "(" LOCATE RO, 33: PRINT "Feet)  :"

```

```
LOCATE RO, 63: PRINT USING "### ##"; RA / mts
LOCATE RO, 62: PRINT "(" LOCATE RO, 70: PRINT "Feet)"
```

```
RO = ROW + 11
```

```
LOCATE RO, 45: PRINT "RB = ";
LOCATE RO, 52: PRINT USING "### ##"; RB: LOCATE RO, 59: PRINT "m"
LOCATE RO, 63: PRINT USING "### ##"; RB / mts
LOCATE RO, 62: PRINT "(" LOCATE RO, 70: PRINT "Feet)"
```

```
CALL fc.inkey(W$)
WIDTH 80, 25
COLOR 15
```

```
END SUB
```

The main purpose of this subroutine is to display a variable high-speed turnoff geometry, which is similar to 30-degree FAA standard high speed turnoff geometry. In this subroutine too, the inputs required are obtained by calling in the subroutines air.design.grp and tax.exit.geo and the others are specified in the subroutine itself. The wing span, the runway width, the distance between the runway and taxiway are to be supplied to this subroutine. The main difference of this subroutine from the others which display turnoff geometries is that here the turnoff geometry is portrayed by two radii. All other dimensions are similar to the 30-degree FAA turnoff. One more significant feature of this subroutine is that the angle of the turnoff can be varied by changing the value of theta in the subroutine.

```
SUB guide.ed.out.opt
```

```
' This subroutine displays the message necessary at the
' top of each of the view menus
```

```
CALL guide.mm
COLOR tcl, bcl
LOCATE 3, 20
PRINT "Press escape (Esc) key to return to OUTPUT MENU."
COLOR ccl, bcl
```

```
END SUB
```

The main purpose of this subroutine is to display a message at the top of the screen. This is used in all the view menus.

```
SUB locexit (nextt, xea())
```

```
' this is for the exit location display with variable scale
```

```
SCREEN 9
COLOR lcl, bcl

CALL box
CALL box.small2
CALL cl.outop
CALL ommsg.aircon
CALL exloc.ttl
```

```
VIEW (40, 97)-(600, 186), 15
WINDOW (0, 0)-(580, 86)
```

```
COLOR 10, 9
LINE (80, 34)-(500, 34), 10
```

```
' This is to determine the scale for the runway downrange
' (which depends on the farthest turnoff)
```

```
xeamax = xea(nextt)
```

```
IF xeamax <= 1500 THEN
    ximax = 1500
ELSEIF xeamax > 2500 THEN
    ximax = 3500
ELSE ximax = 2500
END IF
```

```
' This gives the number of divisions according to the
' maximum value on the scale
```

```
IF ximax = 1500 THEN
    nea = 6
ELSEIF ximax = 3500 THEN
    nea = 7
ELSE nea = 5
```

```

END IF

FOR i = 0 TO nea
  LINE (80 + 420 * i / nea, 34)-(80 + 420 * i / nea, 37), 10
NEXT i

' This draws the arrow using the color from the subroutine clr.dec

FOR i = 1 TO nexit
  CALL clr.dec(i, jk)
  LINE (78 + xea(i) / ximax * 420, 39)-(78 + xea(i) / ximax * 420, 64), jk
  LINE (79 + xea(i) / ximax * 420, 37)-(79 + xea(i) / ximax * 420, 64), jk
  LINE (80 + xea(i) / ximax * 420, 34)-(80 + xea(i) / ximax * 420, 64), jk
  LINE (81 + xea(i) / ximax * 420, 37)-(81 + xea(i) / ximax * 420, 64), jk
  LINE (82 + xea(i) / ximax * 420, 39)-(82 + xea(i) / ximax * 420, 64), jk
NEXT i

COLOR 12, bcl
LOCATE 13, 32: PRINT "Runway Down Range (m)"
  LINE (415, 10)-(402, 10), 12      ' prints the
  LINE (415, 10)-(412, 8), 12      ' small arrow
  LINE (415, 10)-(412, 12), 12
COLOR 1cl, bcl
LOCATE 8, 26: PRINT "RUNWAY EXIT LOCATION DISPLAY"

LOCATE 12, 16: PRINT "0":

' These print the values at each division for the corresponding scale

IF nea = 6 THEN
  FOR i = 0 TO nea
    LINE (80 + 420 * i / nea, 34)-(80 + 420 * i / nea, 37), 10
  NEXT i
  LOCATE 12, 23: PRINT "250": LOCATE 12, 31: PRINT "500":
  LOCATE 12, 40: PRINT "750": LOCATE 12, 48: PRINT "1000":
  LOCATE 12, 56: PRINT "1250": LOCATE 12, 64: PRINT ximax
END IF

IF nea = 5 THEN
  FOR i = 0 TO nea
    LINE (80 + 420 * i / nea, 34)-(80 + 420 * i / nea, 37), 10
  NEXT i
  LOCATE 12, 24: PRINT "500": LOCATE 12, 34: PRINT "1000":
  LOCATE 12, 44: PRINT "1500": LOCATE 12, 54: PRINT "2000":
  LOCATE 12, 64: PRINT ximax
END IF

IF nea = 7 THEN
  FOR i = 0 TO nea
    LINE (80 + 420 * i / nea, 34)-(80 + 420 * i / nea, 37), 10
  NEXT i
  LOCATE 12, 22: PRINT "500": LOCATE 12, 28: PRINT "1000":
  LOCATE 12, 36: PRINT "1500": LOCATE 12, 43: PRINT "2000":
  LOCATE 12, 50: PRINT "2500": LOCATE 12, 57: PRINT "3000":
  LOCATE 12, 64: PRINT "3500"
END IF
  COLOR 1cl, bcl

IF nexit = 1 THEN
  COLOR 14, bcl
  LOCATE 19, 6: PRINT "EXIT LOCATION = (m)"
  LOCATE 19, 22: PRINT USING "#####": xea(1)
  ELSE CALL tbl.exloc(nexit, xea())
END IF

COLOR 1cl, bcl

END SUB

```

This subroutine displays the location of the exits on the runway downrange. The runway downrange comprises of three different scales, viz., 0-1500, 0-2500, 0-3500 meters. The specific scale used here depends on the location of the farthest turnoff from the start of the runway. In addition to general subroutines, the subroutine Tbl exloc is called in if the number of exits is greater than one, otherwise just the exit number and its location are displayed on the screen. The inputs to this subroutine are nexit, the number of exits and xea's the location of each exit.

```

DEFSTR W
'= =====
SUB menu.out (rotrun(), relrun())

```

' This subroutine displays the Output menu screen

```

SCREEN 0
tcl = 14
COLOR lcl, bcl
CALL box
CALL stitle("OUTPUT MENU")
CALL box.small
CALL guide.ed
COLOR ccl, bcl
W$ = " 1)View The Output"
ih = 10: ichoice = 31: CALL print.str(W$): is1 = POS(n)

W$ = " 2)Print The Report"
ih = 12: ichoice = 31: CALL print.str(W$): is2 = POS(n)

W$ = " 3)Help"
ih = 14: ichoice = 31: CALL print.str(W$): is3 = POS(n)

W$ = " 4)Go To Main Menu"
ih = 16: ichoice = 31: CALL print.str(W$): is4 = POS(n)

Ik = 1
DO
  COLOR bcl, csc
  SELECT CASE Ik
    CASE 1
      W$ = " 1)View the Output"
      ih = 10: ichoice = 31: CALL print.str(W$)
      LOCATE 8, (is1 - LEN(W$))
      CALL get.ival(nj)
      COLOR ccl, bcl: CALL print.str(W$)
      ednum = 1
    CASE 2
      W$ = " 2)Print The Report"
      ih = 12: ichoice = 31: CALL print.str(W$): LOCATE 10, is2 - LEN(W$)
      CALL get.ival(nj)
      COLOR ccl, bcl: CALL print.str(W$)
      ednum = 2
    CASE 3
      W$ = " 3)Help"
      ih = 14: ichoice = 31: CALL print.str(W$): LOCATE 12, is3 - LEN(W$)
      CALL get.ival(nj)
      COLOR ccl, bcl: CALL print.str(W$)
      ednum = 3
    CASE 4
      W$ = " 4)Go To Main Menu"
      ih = 16: ichoice = 31: CALL print.str(W$): LOCATE 14, is4 - LEN(W$)
      CALL get.ival(nj)
      COLOR ccl, bcl: CALL print.str(W$)
      ednum = 4

    CASE ELSE
      nj = 1
      END SELECT
      Ik = Ik + nj
      IF Ik = 0 THEN Ik = 4
      IF Ik = 5 THEN Ik = 1
  LOOP UNTIL nj >= 2

  IF nj = 3 THEN
    EXIT SUB
  ELSEIF ednum = 3 THEN
    CALL menu.out(rotrun(), relrun())
  ELSEIF ednum = 2 THEN
    CALL prt.out(rotrun(), relrun())
    CALL menu.out(rotrun(), relrun())
  ELSEIF ednum = 1 THEN
    IF (runmode = 1) THEN
      CALL view.menu1(rotrun(), relrun())

    ELSEIF (runmode = 3) THEN
      CALL view.menu3(rotrun(), relrun())

    ELSE
      CALL view.menu2(rotrun(), relrun())
    END IF

    CALL menu.out(rotrun(), relrun())
  ELSE

```

```

EXIT SUB
END IF

```

```

CLS

```

```

END SUB

```

The main purpose of this subroutine is to create an output menu screen. Here, screen mode '0' is used, whereas in most of the subroutines, screen mode '9' is used. This subroutine calls in several other subroutines depending on the option selected.

```

DEFSNG W
SUB msg.airchar

```

```

' This is to display the aircraft characteristics on the
' top right side of the screen

```

```

COLOR 14
LOCATE 2, 40: PRINT "AIRCRAFT CHARACTERISTICS"
COLOR lcl, bcl
LOCATE 3, 40: PRINT "Aircraft": LOCATE 3, 60: PRINT ":"
LOCATE 4, 40: PRINT "Landing Mass": LOCATE 4, 60: PRINT ":"
LOCATE 5, 40: PRINT "Approach Speed": LOCATE 5, 60: PRINT ":"
LOCATE 6, 40: PRINT "Touchdown Point": LOCATE 6, 60: PRINT ":"

```

```

END SUB

```

This subroutine prints the aircraft characteristics on the top right hand side of the screen. This subroutine utilizes a different color for the heading of the message for easy reference.

```

SUB msg.aircon

```

```

' This displays the airport conditions on the
' top left side of the screen

```

```

COLOR 14
LOCATE 2, 35: PRINT "AIRPORT CONDITIONS"
COLOR lcl, bcl
LOCATE 3, 35: PRINT "Temperature (C) :": LOCATE 3, 55: PRINT USING "### ##"; airtem
LOCATE 4, 35: PRINT "Elevation (m) :": LOCATE 4, 55: PRINT USING "#### #"; airelv
LOCATE 5, 35: PRINT "Wind Speed (m/s) :": LOCATE 5, 55: PRINT USING "### ##"; awsp
LOCATE 6, 35: PRINT "Weather (%) :":
LOCATE 6, 55: PRINT "Dry - ": LOCATE 6, 61: PRINT USING "##"; p.wth(1);
LOCATE 6, 63: PRINT " & Wet - ": LOCATE 6, 72: PRINT USING "##"; p.wth(2);

```

```

END SUB

```

This subroutine displays the airport conditions on the top of the screen. These values were given by the user at the start of the program.

```

SUB msg.airpomix

```

```

' This is to display the aircraft population mixes on the
' top right side of the screen

```

```

COLOR 14
LOCATE 2, 48: PRINT "AIRCRAFT POPULATION MIXES"
COLOR lcl, bcl
LOCATE 3, 48: PRINT "TERP A (%) :"; catprop(1) * 100
LOCATE 4, 48: PRINT "TERP B (%) :"; catprop(2) * 100
LOCATE 5, 48: PRINT "TERP C (%) :"; catprop(3) * 100
LOCATE 6, 48: PRINT "TERP D (%) :"; catprop(4) * 100

```

```

END SUB

```

The main purpose of this subroutine is to display the aircraft population mixes at the top right side of the screen. These mixes are determined internally in the program.

```

SUB msg.airpop

```

```

' This prints the aircraft population on the top right

```

```

COLOR 14
LOCATE 2, 40: PRINT "AIRCRAFT POPULATION"
COLOR lcl, bcl
LOCATE 3, 40: PRINT "TERP A": LOCATE 3, 52: PRINT ":"
LOCATE 4, 40: PRINT "TERP B": LOCATE 4, 52: PRINT ":"
LOCATE 5, 40: PRINT "TERP C": LOCATE 5, 52: PRINT ":"
LOCATE 6, 40: PRINT "TERP D": LOCATE 6, 52: PRINT ":"

```

END SUB

This subroutine displays the aircraft population on the top right hand side of the screen. Here too, the heading is displayed in a different color for easy reference.

SUB out.mod

REM \$DYNAMIC

DIM reirun(25, 2, 10), rotrun(25, 2, 10)

.....

SCREEN 0, 0

bcl = 9

ccl = 15

tcl = 11

lcl = 15

csccl = 7

msum = 0

FOR iacf = 1 TO nmix

 msum = msum + mixlist(iacf)

 catprop(catlist(iacf)) = catprop(catlist(iacf)) + mixlist(iacf)

NEXT iacf

FOR icat = 1 TO 5

 catprop(icat) = catprop(icat) / msum

NEXT icat

.....

IF (runmode = 1) THEN

.....

OPEN ofile\$ FOR INPUT AS #99

INPUT #99, avg.rot

FOR iacf = 1 TO nmix

 INPUT #99, acflist\$(iacf)

NEXT iacf

FOR iacf = 1 TO nmix

FOR iwet = 1 TO 2

FOR kex = 1 TO nextit(1)

 INPUT #99, rotrun(iacf, iwet, kex), reirun(iacf, iwet, kex)

NEXT kex, iwet, iacf

CLOSE #99

.....

ELSEIF (runmode = 3) THEN

.....

OPEN ofile\$ FOR INPUT AS #99

INPUT #99, nextit(2), avg.rot

FOR iacf = 1 TO nmix

 INPUT #99, acflist\$(iacf)

NEXT iacf

ERASE extype

FOR kex = 1 TO nextit(2)

INPUT #99, exloc.new(kex), ex.spd(kex), crit.acf(kex), radius(kex, 1), radius(kex, 2)

NEXT kex

FOR iacf = 1 TO nmix

FOR iwet = 1 TO 2

 INPUT #99, iex.all(iacf, iwet), rotrun(iacf, iwet, iex.all(iacf, iwet))

NEXT iwet, iacf

CLOSE #99

.....

ELSE

.....

OPEN ofile\$ FOR INPUT AS #99

INPUT #99, nextit(2), avg.rot

FOR iacf = 1 TO nmix

 INPUT #99, acflist\$(iacf)

NEXT iacf

FOR kex = 1 TO nextit(2)

INPUT #99, exloc.new(kex), ex.spd(kex), crit.acf(kex), radius(kex, 1), radius(kex, 2)

NEXT kex

FOR kex = 1 TO nextit(1)

exloc.old(kex) = exloc(kex)

```

extype.old(kex) = extype(kex)
NEXT

CALL sort.ex1
FOR iacf = 1 TO nmix
FOR iwet = 1 TO 2
    INPUT #99, wkloc, wkrot
    FOR kex = 1 TO (nexit(1) + nextit(2))
        IF (wkloc = exloc(kex)) THEN
            rotrun(iacf, iwet, kex) = wkrot
        END IF
    NEXT kex
NEXT iwet, iacf

CLOSE #99

```

```

'-----
END IF

CALL menu.out(rotrun(), relrun())
ERASE relrun, rotrun

END SUB

```

This subroutine forms the main portion of the program. All the subroutines are connected through this subroutine. Here, depending on the runmode option selected, files are opened appropriately to save the values from the input. Subroutines sort and menu.out are called here to assist the main subroutine.

```

REM $STATIC
SUB pexl.run1

' This subroutine calls in the requested FAA Turnoff
' pertaining to runmode '1'

```

```

SCREEN 9
CALL locexit(nexit(1), exloc())
CALL ask.exnum.run1(nex.fix)
' CALL cl.sbox2
' CALL cl.bot.sbox
IF (nex.fix = 0) THEN
    EXIT SUB
ELSEIF (extype(nex.fix) = 1) THEN
    'CALL geo.new.30
ELSEIF (extype(nex.fix) = 2) THEN
    'CALL geo.new.45
ELSEIF (extype(nex.fix) = 3) THEN
    'CALL geo.new.90
END IF
CALL pexl.run1
END SUB

```

This subroutine calls in the appropriate turnoff geometry subroutines in accordance with the type of turnoff selected for each of the exits. This subroutine is exclusively used in runmode '1'. Subroutines locexit, ask.exnum.run1, cl.sbox2 and cl.bot.sbox are also called in this subroutine.

```

SUB pexl.run2

' This subroutine calls in the requested FAA Turnoff and the
' variable Turnoff pertaining to runmode '2'

```

```

REM $DYNAMIC
DIM xtra(10, 50), ytra(10, 50)

SCREEN 9
CALL locexit(nexit(1) + nextit(2), exloc())
181
CALL ask.cen(licen)
IF licen > 2 THEN GOTO 181

IF (licen = 0) THEN

    ERASE xtra, ytra
    EXIT SUB

ELSEIF (licen = 1) THEN
    CALL cl.bot.sbox
    CALL ask.extran(ex.com, ex.cen())
    LOCATE 25, 2: PRINT "Calculating the centerline coordinates. Please wait !";
    CALL ccline.run2(ex.com, ex.cen(), xtra(), ytra())
    CALL pexl.run2

```

```

ELSE
  CALL cl.bot.sbox
  CALL ask.exnum.run1(nex.fix)

  IF (nex.fix = 0) THEN
  EXIT SUB
  ELSEIF (extype(nex.fix)) = 1 THEN
    'CALL geo.new.30
  ELSEIF (extype(nex.fix) = 2) THEN
    'CALL geo.new.45
  ELSEIF (extype(nex.fix) = 3) THEN
    'CALL geo.new.90
  ELSEIF (extype(nex.fix) = 0) THEN
    CALL cal.track(nex.fix, crit.acf(nex.fix), trackmax)
    PRINT " = = = = > "; trackmax
    'call geo.new.var
  END IF

  CALL pexl.run2
END IF

END SUB

```

The main purpose of the subroutine is to call appropriate turnoff geometry subroutines in accordance with the type of turnoff selected for each of the exits. Since this subroutine is used for runmode '2', the FAA turnoff subroutines and the variable turnoff subroutine, developed by REDIM are called depending on the type of turnoff. Subroutines locexit, ask.exnum.run1, cl.sbox2 and cl.bot.sbox are also called in this subroutine. The variable turnoff geometry subroutine is called after calling the subroutine ca.track, which determines the maximum track-in.

```

REM $STATIC
SUB pexl.run3

```

```

' This subroutine calls in the requested FAA Turnoff and the
' the variable Turnoff pertaining to runmode '3'

```

```

REM $DYNAMIC
DIM xtra(10, 50), ytra(10, 50)

```

```

SCREEN 9
CALL locexit(nexit(2), exloc.new())
81
CALL ask.cen(icen)
IF icen > 2 THEN GOTO 81

```

```

IF (icen = 0) THEN
  ERASE xtra, ytra
  EXIT SUB
ELSEIF (icen = 1) THEN
  CALL cl.bot.sbox
  CALL ask.extran(ex.com, ex.cen())
  LOCATE 25, 2: PRINT "Calculating the centerline coordinates. Please wait !";
  CALL ccline.run3(ex.com, ex.cen(), xtra(), ytra())
  CALL pexl.run3
ELSE
  CALL cl.bot.sbox
  CALL ask.exnum.run1(nex.fix)

```

```

  IF (nex.fix = 0) THEN
  EXIT SUB
  ELSEIF (extype(nex.fix)) = 1 THEN
    'CALL geo.new.30
  ELSEIF (extype(nex.fix) = 2) THEN
    'CALL geo.new.45
  ELSEIF (extype(nex.fix) = 3) THEN
    'CALL geo.new.90
  ELSEIF (extype(nex.fix) = 0) THEN
    'call geo.new.var
  END IF

```

```

  CALL pexl.run3
END IF
END SUB

```

This subroutine is similar to the subroutine pexl.run2. But this is used for runmode '3' only. Since all the exits proposed in runmode are new, the subroutine called for turnoff geometry corresponds to only the variable turnoff geometry subroutine.

```

REM $STATIC
SUB plot.cen (ex.com, npnt(), xtra(), ytra())

```

```

' This subroutine draws the centerline turnoff(s)

```

```

' along with the grid and the dimensions

SCREEN 9
WIDTH 80, 43
COLOR 15, 9

' To determine the scale

' This determines the maximum value of the turnoff
' geometry coordinates

xtramax = xtra(1, npnt(1))
FOR i = 2 TO ex.com
IF xtra(i, npnt(i)) > xtramax THEN
xtramax = xtra(i, npnt(i))
END IF
NEXT i
IF (xtramax < distlw) THEN
xtramax = distlw * 1.1
END IF

IF xtramax > 40 AND xtramax <= 80 THEN
xpnmax = 80
ELSEIF xtramax > 80 AND xtramax <= 120 THEN
xpnmax = 120
ELSEIF xtramax > 120 AND xtramax <= 160 THEN
xpnmax = 160
ELSEIF xtramax > 160 AND xtramax <= 200 THEN
xpnmax = 200
ELSEIF xtramax > 200 AND xtramax <= 240 THEN
xpnmax = 240
ELSEIF xtramax > 240 AND xtramax <= 280 THEN
xpnmax = 280
ELSEIF xtramax > 280 AND xtramax <= 320 THEN
xpnmax = 320
ELSEIF xtramax > 320 AND xtramax <= 360 THEN
xpnmax = 360
ELSEIF xtramax > 360 AND xtramax <= 400 THEN
xpnmax = 400
ELSEIF xtramax > 400 AND xtramax <= 440 THEN
xpnmax = 440
ELSEIF xtramax > 440 AND xtramax <= 480 THEN
xpnmax = 480
ELSEIF xtramax > 480 AND xtramax <= 520 THEN
xpnmax = 520
ELSEIF xtramax > 520 AND xtramax <= 560 THEN
xpnmax = 560
ELSEIF xtramax > 560 AND xtramax <= 600 THEN
xpnmax = 600
ELSEIF xtramax > 600 AND xtramax <= 640 THEN
xpnmax = 640
ELSEIF xtramax > 640 AND xtramax <= 680 THEN
xpnmax = 680
ELSEIF xtramax > 680 AND xtramax <= 720 THEN
xpnmax = 720
ELSEIF xtramax > 720 AND xtramax <= 760 THEN
xpnmax = 760
ELSEIF xtramax > 760 AND xtramax <= 800 THEN
xpnmax = 800
ELSEIF xtramax > 800 AND xtramax <= 840 THEN
xpnmax = 840
ELSEIF xtramax > 840 AND xtramax <= 880 THEN
xpnmax = 880
ELSE xpnmax = 1000
END IF

xscale = 576 / xpnmax
yscale = xscale

' This is for displaying the value at each division

xypn = 210 / 576 * 1.37 * xpnmax

COLOR 15 ' horizontal
LOCATE 35, 6: PRINT 0
FOR i = 1 TO 8
LOCATE 35, 5 + 72 * i / 8:
IF i < 8 THEN

```

```

        PRINT xpnmax / 8 * I
    ELSE
        IF xpnmax > 999 THEN
            LOCATE 35, 3 + 72 * I / 8: PRINT xpnmax / 8 * I
        ELSE
            LOCATE 35, 4 + 72 * I / 8: PRINT xpnmax / 8 * I
        END IF
    END IF
NEXT I

' Does not the give correct spacing as the
' division of 15 rows by 4 is not an integer

    LOCATE 34, 4: PRINT 0
    FOR I = 1 TO 4 ' vertical
        LOCATE 34 - 52.5 * I / 8, 3: PRINT CINT(xypn / 4 * I)
    NEXT I

VIEW (4, 5)-(636, 345)
WINDOW (0, 0)-(632, 340)

LINE (0, 13)-(630, 340), , B

' These lines draw the frame

LINE (621, 80)-(621, 290), 10
LINE (46, 80)-(621, 80), 10
LINE (46, 290)-(621, 290), 10
LINE (46, 80)-(46, 290), 10

' To draw the grid

FOR I = 1 TO 7
    LINE (46 + (576 * I / 8), 56 + 24)-(46 + (576 * I / 8), 266 + 24), 11, , &H8888
NEXT I

FOR I = 1 TO 3
    LINE (46, 56 + 24 + (210 * I / 4))-(621, 56 + 24 + (210 * I / 4)), 11, , &H8888
NEXT I

' This is for plotting the centerline of the turnoff(s)

nrow = INT(ex.com / 4 + .99)

FOR kex = 1 TO ex.com 'ex.cen(kex)
    CALL clr.dec(kex, jk)
    FOR k = 1 TO npnt(kex) - 1
        p1x = 46 + (xtra(kex, k)) * xscale
        p1y = 80 + (ytra(kex, k)) * yscale / 1.37
        p2x = 46 + (xtra(kex, k + 1)) * xscale
        p2y = 80 + (ytra(kex, k + 1)) * yscale / 1.37
        LINE (INT(p1x), INT(p1y))-(INT(p2x), INT(p2y)), jk

        lr = kex - INT(kex / nrow - .01) * nrow
        LC = INT(kex / nrow + .99)
        lcx = (LC - 1) * 16 + 7
        lrx = 39 + lr
        COLOR jk
        LOCATE lrx, lcx: PRINT STRING$(2, 196);
        LOCATE lrx, lcx + 4: PRINT "Exit #"; ex.cen(kex);
    NEXT k, kex

COLOR 10
LOCATE 37, 33: PRINT "Runway Downrange (m)";
LOCATE 16, 3: PRINT "L": LOCATE 17, 3: PRINT "a"
LOCATE 18, 3: PRINT "l": LOCATE 19, 3: PRINT "e"
LOCATE 20, 3: PRINT "r": LOCATE 21, 3: PRINT "a"
LOCATE 22, 3: PRINT "l": LOCATE 25, 3: PRINT "R"
LOCATE 26, 3: PRINT "a": LOCATE 27, 3: PRINT "n"
LOCATE 28, 3: PRINT "g": LOCATE 29, 3: PRINT "e"
LOCATE 32, 2: PRINT "(m)"

COLOR 14
LOCATE 4, 18: PRINT "COMPARISION OF CENTERLINE-TURNOFF GEOMETRIES"
LOCATE 5, 18: PRINT "-----"
COLOR 15

```



```

LPRINT " -----"
IF (runmode < > 1) THEN

LPRINT "      TERPS A      I";
LPRINT USING "      ### # & ### #"; vexit(1, 1), "I", vexit(1, 2)
LPRINT "      TERPS B      I";
LPRINT USING "      ### # & ### #"; vexit(2, 1), "I", vexit(2, 2)
LPRINT "      TERPS C      I";
LPRINT USING "      ### # & ### #"; vexit(3, 1), "I", vexit(3, 2)
LPRINT "      TERPS D      I";
LPRINT USING "      ### # & ### #"; vexit(4, 1), "I", vexit(4, 2)
LPRINT "      TERPS E      I";
LPRINT USING "      ### # & ### #"; vexit(5, 1), "I", vexit(5, 2)
LPRINT " -----"

END IF

LPRINT ""
LPRINT ""
LPRINT ""
LPRINT ""
LPRINT "II. ANALYSIS RESULTS"
LPRINT ""
LPRINT ""
LPRINT "II-1. Average ROT"
LPRINT ""
LPRINT USING " & ### # &"; "average ROT =", avg.rot, "(sec)"
LPRINT ""
LPRINT ""
LPRINT "II-2. Exit Locations, Types, and Turn-off Assignment."
LPRINT ""

IF runmode = 1 THEN
nex = nexit(1)
ELSEIF runmode = 2 THEN
nex = nexit(1) + nexit(2)
ELSE
nex = nexit(2)
END IF

ww$ = " -----"
ww$ = ww$ + STRING$(7 * nex, 45)
LPRINT ww$

LPRINT " Exit #      I";
FOR kex = 1 TO nex
LPRINT USING " ## "; kex;
NEXT kex
LPRINT ""
LPRINT ww$

LPRINT " Location (m) I";
'-----
IF runmode = 3 THEN
'-----
FOR kex = 1 TO nex
LPRINT USING " #### "; exloc.new(kex);
NEXT kex
LPRINT ""
LPRINT ww$
LPRINT " Type      I";
FOR kex = 1 TO nex
LPRINT USING " & "; " new";
NEXT kex
LPRINT ""

'-----
ELSE
'-----
FOR kex = 1 TO nex
LPRINT USING " #### "; exloc(kex);
NEXT kex
LPRINT ""
LPRINT ww$
LPRINT " Type      I";
FOR kex = 1 TO nex
IF extype(kex) = 0 THEN
tt$ = "new "
ELSEIF extype(kex) = 1 THEN
tt$ = "30-d"

```

```

ELSEIF extype(kex) = 2 THEN
    tt$ = "45-d"
ELSE
    tt$ = "90-d"
END IF
LPRINT USING " & "; tt$;
NEXT kex
LPRINT ""
'-----
END IF
'-----

LPRINT ww$

FOR iacf = 1 TO nmix
LPRINT USING " \ \ \ &"; acfllst$(iacf), "I";

'-----
IF runmode < > 1 THEN
'-----
    LPRINT ""
    FOR lwet = 1 TO 2
    IF lwet = 1 THEN
        iw$ = "dry"
    ELSE
        iw$ = "wet"
    END IF
    LPRINT USING " \ \ \ &"; lw$, "I";
    FOR kex = 1 TO nex
    IF rotrun(iacf, lwet, kex) = 0 THEN
        LPRINT " ";
    ELSE
        LPRINT USING " ### # "; rotrun(iacf, lwet, kex);
    END IF
    NEXT kex
    LPRINT ""
    NEXT lwet
'-----
ELSE
'-----
    FOR lwet = 1 TO 2
    LPRINT ""
    IF lwet = 1 THEN
        iw$ = "dry"
    ELSE
        iw$ = "wet"
    END IF
    LPRINT USING " \ \ \ \ &"; lw$, "ROT", "I";
    FOR kex = 1 TO nex
    IF relrun(iacf, lwet, kex) = 0 THEN
        LPRINT " ";
    ELSE
        LPRINT USING " ### # "; relrun(iacf, lwet, kex);
    END IF
    NEXT kex
    LPRINT ""
    LPRINT USING " \ \ \ &"; "REL(%)", "I";
    FOR kex = 1 TO nex
    IF relrun(iacf, lwet, kex) = 0 THEN
        LPRINT " ";
    ELSE
        LPRINT USING " ### # "; relrun(iacf, lwet, kex) * 100;
    END IF
    NEXT kex
    LPRINT ""
    NEXT lwet
'-----
END IF
'-----

LPRINT ww$
NEXT iacf

IF (runmode < > 1) THEN
LPRINT ""
LPRINT ""
LPRINT "II-3. Turn-off Centerline Geometries."
LPRINT ""

```

```

FOR kex = 1 TO nex
ex.cen(kex) = kex
CALL cal.cline(kex, npnt(), xtra(), ytra())
NEXT kex

npg = INT(nex / 4.1) + 1
ex.s(1) = 1
ex.s(2) = 5
IF npg = 1 THEN
ex.e(1) = nex
ELSE
ex.e(1) = 4
ex.e(2) = nex
END IF

FOR ipg = 1 TO npg
nmax = 0
pex = ex.e(ipg) - ex.s(ipg) + 1
ww$ = STRING$(16 * pex + 1, 45)
LPRINT " "; ww$
LPRINT " I";
FOR kex = ex.s(ipg) TO ex.e(ipg)
LPRINT USING " \ \ # &"; "Exit", kex, "I";
IF npnt(kex) > nmax THEN nmax = npnt(kex)
NEXT kex
LPRINT ""
LPRINT " "; ww$

LPRINT " I";
FOR kex = ex.s(ipg) TO ex.e(ipg)
LPRINT USING " & & & &"; "X", "I", "Y", "I";
NEXT kex
LPRINT ""
LPRINT " "; ww$

FOR ipnt = 1 TO nmax
LPRINT " I";
FOR kex = ex.s(ipg) TO ex.e(ipg)
IF xtra(kex, ipnt) = 0 THEN
LPRINT USING " &"; "I";
ELSE
LPRINT USING " #### # #### # &"; xtra(kex, ipnt), ytra(kex, ipnt), "I";
END IF
NEXT kex
LPRINT ""
NEXT ipnt
LPRINT " "; ww$
LPRINT ""
LPRINT ""
NEXT ipg
END IF
LOCATE 25, 2: PRINT STRING$(25, 32);
ERASE npnt, xtra, ytra, ex.s, ex.e
END SUB

```

Sub Prt.out (rotrun(),retrun())

The main purpose of this subroutine is to give a hard-copy of complete report of the REDIM. The ROT and the reliability are the inputs in this subroutine. The subroutine internally determines the type of runmode selected and gives the report accordingly. The report includes the values supplied in the Input Module as well as the results of the analysis.

```

REM $STATIC
SUB sort.ex1

```

```

DIM wk.ex1(nexit(1)), wk.type(nexit(1)), wk.acf(nexit(2)), wk.spd(nexit(2))
FOR jex = 1 TO nexit(1)
wk.ex1(jex) = exloc(jex)
wk.type(jex) = extype(jex)
NEXT jex
FOR iex = 1 TO nexit(2)
wk.acf(iex) = crit.acf(iex)
wk.spd(iex) = ex.spd(iex)
NEXT iex

```

```

lex = 1: jex = 1
FOR kex = 1 TO nexit(1) + nexit(2)

```

```

exloc(kex) = exloc.new(iex)
extype(kex) = 0
IF (exloc(kex) > wk.exl(jex)) THEN
exloc(kex) = wk.exl(jex)
extype(kex) = wk.type(jex)
jex = jex + 1
ELSE
crit.acf(kex) = wk.acf(iex)
ex.spd(kex) = wk.spd(iex)
iex = iex + 1
END IF
IF (jex = nexit(2) + 1) THEN
FOR kkex = kex + 1 TO nexit(1) + nexit(2)
exloc(kkex) = wk.exl(jex)
extype(kkex) = wk.type(jex)
jex = jex + 1
NEXT kkex
'FOR kex = 1 TO nexit(1) + nexit(2)
'PRINT exloc(kex), extype(kex), crit.acf(kex), ex.spd(kex)
'NEXT kex
'INPUT r

EXIT SUB
END IF
IF (jex = nexit(1) + 1) THEN
FOR kkex = kex + 1 TO nexit(1) + nexit(2)
exloc(kkex) = exloc.new(iex)
crit.acf(kkex) = wk.acf(iex)
ex.spd(kkex) = wk.spd(iex)
NEXT kkex

EXIT SUB
END IF
NEXT kex
END SUB

```

SUB stat.bar.ttl

' This draws a box and prints the title for the bar chart

```

COLOR 11
LOCATE 2, 3
PRINT CHR$(218); STRING$(26, 196); CHR$(191);
FOR i = 3 TO 5
LOCATE i, 3
PRINT CHR$(179);
LOCATE i, 30
PRINT CHR$(179);
NEXT i
LOCATE 6, 3
PRINT CHR$(192); STRING$(26, 196); CHR$(217);
COLOR 10
LOCATE 3, 7: PRINT "STATISTICS ON ROT"
LOCATE 4, 7: PRINT "PER AIRCRAFT FOR DRY"
LOCATE 5, 7: PRINT "AND WET CONDITIONS"
COLOR lcl, bcl

```

END SUB

This subroutine displays the title for the "Aircraft Statistics and their ROT's" screen. This is displayed on the top left side of the screen.

SUB stat.bar2 (rotrun(), relrun())

' This subroutine displays the bar chart (for ROT's) along with the names
' of each of the aircraft. The weighted average ROT is shown as a line
' across the bars. This pertains to runmode '2' only

```

REM $DYNAMIC
DIM rrot(25, 2)

```

```

FOR iacf = 1 TO nmix
FOR iwet = 1 TO 2
iii = 1
IF runmode = 2 THEN
nex = nexit(1) + nexit(2)
ELSE

```

```

    nex = next(2)
END IF
FOR kex = 1 TO nex
    IF (rotun(iacf, iwet, kex) > .001) AND (ili = 1) THEN
        rrot(iacf, iwet) = rotun(iacf, iwet, kex)
        iii = 0
    END IF
NEXT kex
NEXT iwet, iacf

nscr = INT((nmix - .5) / 6) + 1

ia.s(1) = 1: ia.s(2) = 7: ia.s(3) = 13: ia.s(4) = 19: ia.s(5) = 25

FOR iss = 1 TO nscr
    IF (nmix >= iss * 6) THEN
        ia.e(iss) = iss * 6
    ELSE
        ia.e(iss) = nmix
    END IF
NEXT iss

' nmix - number of aircrafts

FOR iscr = 1 TO nscr

    SCREEN 9
    COLOR icl, bcl
    VIEW (40, 98)-(601, 302)
    WINDOW (0, 0)-(520, 204)

    CALL box
    CALL box.small2
    'CALL cl.outop
    CALL omeg.aircon
    CALL stat.bar.ttl

    LINE (40, 58)-(460, 58)          ' To draw the horizontal
    LINE (40, 58)-(40, 198)          ' and vertical axis

    FOR i = 0 TO 5                    ' To make divisions
        LINE (40, 58 + 140 * i / 5)-(37, 58 + 140 * i / 5) ' on vertical axis
    NEXT i

    LOCATE 18, 7: PRINT "15": LOCATE 16, 7: PRINT "30"
    LOCATE 14, 7: PRINT "45": LOCATE 12, 7: PRINT "60"
    LOCATE 10, 7: PRINT "75": LOCATE 8, 7: PRINT "90"
    COLOR 10
    LOCATE 13, 5: PRINT "R": LOCATE 14, 5: PRINT "O": LOCATE 15, 5: PRINT "T"
    LOCATE 20, 31: PRINT "AIRCRAFT TYPE"

    ' LINE (334, 30)-(345, 30), 15    ' To draw an arrow
    ' LINE (342, 32)-(345, 30), 15
    ' LINE (342, 28)-(345, 30), 15

    COLOR 15
    ' This is to draw the bars corresponding to the ROT for each
    ' aircraft type

    FOR air = ia.s(iscr) TO ia.e(iscr)
        FOR j = 1 TO 2
            ibar = (air - 1) * 2 + j
            iclr = ibar - INT(ibar / 2)
            CALL clr.dec(iclr, jk)
            LINE (460 * 2 * (ibar) / 27 + 26, 59)-(460 * 2 * (ibar) / 27 + 26 + 16, 59 +
                ((rrot(air, j) - 15) / 75) * 140), jk, BF
            COLOR 14
            ' To print the corresponding turnoff number taken by that
            ' particular aircraft just above the bar
            LOCATE 17 - ((rrot(air, j) - 15) / (75 * 14) * 140), 9 + ibar * 4.6: PRINT iclr
        NEXT j
    NEXT air

    COLOR 15
    ' To print the airport conditions (wet or dry) to their bars
    FOR iar = 1 TO ibar STEP 2
        LOCATE 19, 10 + iar * 4.6: PRINT "D"
    NEXT iar

```

```

FOR ibr = 2 TO ibar STEP 2
  LOCATE 19, 10 + ibr * 4.6: PRINT "W"
NEXT ibr
COLOR 15

' nrow - number of rows in which the aircrafts names are displayed
nrow = INT(nmix / 3 + .99)

FOR ii = 1 TO (air - 1)
  CALL clr.dec(ii, jk)
  COLOR jk, bcl
  lr = ii - INT(ii / nrow * .01) * nrow
  LC = INT(ii / nrow + .99)
  lcx = (LC - 1) * 24 + 6
  lrx = 20 + lr
  LOCATE lrx, lcx: PRINT ii
  LOCATE lrx, lcx + 4: PRINT "~"
  LOCATE lrx, lcx + 7: PRINT acflist$(ii)
NEXT ii
COLOR lcl, bcl

COLOR 15
' This draws the average ROT arrow

FOR L = 1 TO 420
  LINE (40, 58 + (avg.rot - 15) / 75 * 140)-(40 + L, 58 + (avg.rot - 15) / 75 * 140), 14
  LINE (40, 59 + (avg.rot - 15) / 75 * 140)-(40 + L, 59 + (avg.rot - 15) / 75 * 140), 12
NEXT L
LINE (457, 56 + (avg.rot - 15) / 75 * 140)-(460, 58 + (avg.rot - 15) / 75 * 140), 12
LINE (457, 61 + (avg.rot - 15) / 75 * 140)-(460, 58 + (avg.rot - 15) / 75 * 140), 12

' This prints the average ROT and makes it blink

FOR s = 1 TO 5
  FOR c = 0 TO 15
    COLOR c
    LOCATE 17 - (avg.rot - 15) * 140 / (75 * 14), 70: PRINT "AVG. ROT"
    LOCATE 18 - (avg.rot - 15) * 140 / (75 * 14), 70: PRINT "="
    LOCATE 18 - (avg.rot - 15) * 140 / (75 * 14), 72: PRINT USING "## ###"; avg.rot
    LOCATE 19 - (avg.rot - 15) * 140 / (75 * 14), 72: PRINT "(sec)"
  NEXT c
NEXT s

COLOR 10
' This leaves color 10 on the average ROT after blinking

LOCATE 17 - (avg.rot - 15) * 140 / (75 * 14), 70: PRINT "AVG. ROT"
LOCATE 18 - (avg.rot - 15) * 140 / (75 * 14), 70: PRINT "="
LOCATE 18 - (avg.rot - 15) * 140 / (75 * 14), 72: PRINT USING "## ###"; avg.rot
LOCATE 19 - (avg.rot - 15) * 140 / (75 * 14), 72: PRINT "(sec)"

COLOR 15
LOCATE 25, 2: PRINT "Press any key to continue.";
CALL fc.inkey(wd$)

NEXT lscr
ERASE rrot
END SUB

The main purpose of this subroutine is to display the ROT's of each of the aircraft, for both wet and dry conditions, in a bar chart form. The inputs to this subroutine are the ROT of each aircraft and reliability associated with each aircraft and turnoff. The range of ROT that can be displayed on this chart is between fifteen seconds and ninety seconds. The aircraft name, the bars representing the aircraft are of the same color. The average ROT is shown as a horizontal line across the bars. This subroutine corresponds to only runmode '2'.

REM $STATIC
SUB stat.bar3 (rotrun(), relrun())

' This subroutine displays the bar chart (for ROT's) along with the names
' of each of the aircraft. The weighted average ROT is shown as a line
' across the bars. This pertains to runmode '3' only

REM $DYNAMIC
DIM rrot(25, 2)

```

```

FOR iacf = 1 TO nmix
FOR iwet = 1 TO 2
    rrot(iacf, iwet) = rotrun(iacf, iwet, lex.all(iacf, iwet))
NEXT iwet, iacf

nscr = INT((nmix - .5) / 6) + 1

ia.s(1) = 1: ia.s(2) = 7: ia.s(3) = 13: ia.s(4) = 19: ia.s(5) = 25

FOR iss = 1 TO nscr
    IF (nmix >= iss * 6) THEN
        ia.e(iss) = iss * 6
    ELSE
        ia.e(iss) = nmix
    END IF
NEXT iss

' nmix - number of aircrafts

FOR iscr = 1 TO nscr

    SCREEN 9
    COLOR lcl, bcl
    VIEW (40, 98)-(601, 302)
    WINDOW (0, 0)-(520, 204)

    CALL box
    CALL box.small2
    'CALL cl.outop
    CALL ommsg.aircon
    CALL stat.bar.ttl

    LINE (40, 58)-(460, 58)          ' To draw the horizontal
    LINE (40, 58)-(40, 198)          ' and vertical axis

    FOR i = 0 TO 5                    ' To make divisions
        LINE (40, 58 + 140 * i / 5)-(37, 58 + 140 * i / 5) ' on vertical axis
    NEXT i

    LOCATE 18, 7: PRINT "15": LOCATE 16, 7: PRINT "30"
    LOCATE 14, 7: PRINT "45": LOCATE 12, 7: PRINT "60"
    LOCATE 10, 7: PRINT "75": LOCATE 8, 7: PRINT "90"
    COLOR 15
    LOCATE 13, 5: PRINT "R": LOCATE 14, 5: PRINT "O": LOCATE 15, 5: PRINT "T"
    ' LOCATE 11, 5: PRINT CHR$(24)
    LOCATE 20, 31: PRINT "AIRCRAFT TYPE"

    ' LINE (334, 30)-(345, 30), 15      ' To draw an arrow
    ' LINE (342, 32)-(345, 30), 15
    ' LINE (342, 28)-(345, 30), 15

    COLOR 15
    ' This is to draw the bars corresponding to the ROT for each
    ' aircraft type

    FOR air = ia.s(iscr) TO ia.e(iscr)
        FOR j = 1 TO 2
            ibar = (air - 1) * 2 + j
            iclr = ibar - INT(ibar / 2)
            CALL clr.dec(iclr, jk)
            LINE (460 * 2 * (ibar) / 27 + 26, 59)-(460 * 2 * (ibar) / 27 + 26 + 16, 59 + ((rrot(air, j) - 15) / 75) * 140), jk, BF
            COLOR 14

            ' To print the corresponding turnoff number taken by that
            ' particular aircraft just above the bar
            LOCATE 17 - ((rrot(air, j) - 15) / (75 * 14) * 140), 9 + ibar * 4.6: PRINT iclr 'lex.all(air, j);
        NEXT j
    NEXT air

    COLOR 15
    ' To print the aircraft numbers corresponding to their bars
    FOR iar = 1 TO ibar STEP 2
        LOCATE 19, 10 + iar * 4.6: PRINT "D"
    NEXT iar

    FOR ibr = 2 TO ibar STEP 2
        LOCATE 19, 10 + ibr * 4.6: PRINT "W"
    NEXT ibr

```

```

NEXT ibr
  COLOR 15

  ' nrow - number of rows in which the aircrafts names are displayed
  nrow = INT(nmix / 3 + .99)

  FOR ii = 1 TO (air - 1)
    CALL clr.dec(ii, jk)
    COLOR jk, bcl
    lr = ii - INT(ii / nrow - .01) * nrow
    LC = INT(ii / nrow + .99)
    lcx = (LC - 1) * 24 + 6
    lrx = 20 + lr
    LOCATE lrx, lcx: PRINT ii
    LOCATE lrx, lcx + 4: PRINT "-"
    LOCATE lrx, lcx + 7: PRINT acflst$(ii)
  NEXT ii
  COLOR lcl, bcl

  COLOR 15
  ' This draws the average ROT arrow

  FOR L = 1 TO 420
    LINE (40, 58 + (avg.rot - 15) / 75 * 140)-(40 + L, 58 + (avg.rot - 15) / 75 * 140), 14
    LINE (40, 59 + (avg.rot - 15) / 75 * 140)-(40 + L, 59 + (avg.rot - 15) / 75 * 140), 12
  NEXT L
  LINE (457, 56 + (avg.rot - 15) / 75 * 140)-(460, 58 + (avg.rot - 15) / 75 * 140), 12
  LINE (457, 61 + (avg.rot - 15) / 75 * 140)-(460, 58 + (avg.rot - 15) / 75 * 140), 12

  ' This prints the average ROT and makes it blink

  FOR s = 1 TO 5
    FOR c = 0 TO 15
      COLOR c
      LOCATE 17 - (avg.rot - 15) * 140 / (75 * 14), 70: PRINT "AVG. ROT"
      LOCATE 18 - (avg.rot - 15) * 140 / (75 * 14), 70: PRINT "="
      LOCATE 18 - (avg.rot - 15) * 140 / (75 * 14), 72: PRINT USING "### ###"; avg.rot
      LOCATE 19 - (avg.rot - 15) * 140 / (75 * 14), 72: PRINT "{sec}"
    NEXT c
  NEXT s

  COLOR 10
  ' This leaves color 10 on the average ROT after blinking

  LOCATE 17 - (avg.rot - 15) * 140 / (75 * 14), 70: PRINT "AVG. ROT"
  LOCATE 18 - (avg.rot - 15) * 140 / (75 * 14), 70: PRINT "="
  LOCATE 18 - (avg.rot - 15) * 140 / (75 * 14), 72: PRINT USING "### ###"; avg.rot
  LOCATE 19 - (avg.rot - 15) * 140 / (75 * 14), 72: PRINT "{sec}"

  COLOR 15
  LOCATE 25, 2: PRINT "Press any key to continue.";
  CALL fc.inkey(wd$)

  NEXT lscr
  ERASE rrot
  END SUB

```

This subroutine is very similar to the subroutine Stat.bar2 (rotrun(), relrun()). But, this subroutine corresponds to runmode '3'.

```

REM $STATIC
SUB std.cline (kex, extp, npnt(), xtra(), ytra())

  ' This subroutine calculates the centerline geometries for
  ' all the three (30, 45 & 90) FAA Turnoff geometries

  npnt(kex) = 15
  pi = 3.141593

  IF (extp = 1) THEN          '30-degree angle
    r1 = 548.64
    theta = 30
    'y.crit = (runwid + acfws(ja)) / 2
  ELSEIF (extp = 2) THEN      '45-degree angle
    r1 = 243.84
    theta = 45
    'y.crit = (runwid + acfws(ja)) / 2
  ELSEIF (extp = 3) THEN      '90-degree angle

```

```

r1 = 76.05
theta = 90
'y.crit = acfws(ja) + runwid / 2
END IF

IF (extp = 1) OR (extp = 2) THEN
  xt = r1 * SIN(theta / 360! * 2! * pi)
  yt = r1 - r1 * COS(theta / 360! * 2 * pi)
  FOR ipnt = 1 TO npnt(kex)
    xtra(kex, ipnt) = ipnt * xt / 15!
    ytra(kex, ipnt) = r1 - SQR(r1  $\neq$  2 - xtra(kex, ipnt)  $\neq$  2)
  NEXT ipnt
'PRINT (360 * theta1 / 2! / pi), SIN(theta1)
'PRINT "tr"; xt; yt; "cr", x.crit; y.crit; theta1; lc
ELSE
  xt = r1
  yt = r1
  FOR ipnt = 1 TO npnt(kex)
    xtra(kex, ipnt) = ipnt * xt / 15!
    ytra(kex, ipnt) = r1 - SQR(r1  $\neq$  2 - xtra(kex, ipnt)  $\neq$  2)
  NEXT ipnt
END IF
npnt(kex) = 16
ytra(kex, 16) = distlw / 2!
ydiff = ytra(kex, 16) - ytra(kex, 15)
IF extp < > 3 THEN
  xtra(kex, 16) = xtra(kex, 15) + ydiff / TAN(theta / 360! * 2 * pi)
ELSE
  xtra(kex, 16) = xtra(kex, 15)
END IF
END SUB

```

This subroutine calculates the centerline coordinates for the standard FAA turnoffs. This is especially useful when the comparison of centerline turnoff geometry coordinates of the new ones are done against the existing ones.

SUB table.AB.1 (rotrun(), relrun())

' This subroutine displays the table with the ROT & Reliability
' for each of the aircraft (upto four on each screen)
' for both wet and dry conditions. This pertains to runmode '1' only

SCREEN 9
WIDTH 80, 43

' to get 1 to 4 on first screen, 5 to 8 on second screen etc.,

nscr = INT((nmix - .5) / 4) + 1
ia.s(1) = 1: ia.s(2) = 5: ia.s(3) = 9: ia.s(4) = 13: ia.s(5) = 17
ia.s(6) = 21: ia.s(7) = 25

```

FOR iss = 1 TO nscr
  IF (nmix >= iss * 4) THEN
    ia.e(iss) = iss * 4
  ELSE
    ia.e(iss) = nmix
  END IF
NEXT iss

```

' to fix the location of the table depending on the number of exits
' i.e., for centering the table on the screen

```

SELECT CASE nextit(1)
CASE 1
  a = 29
CASE 2
  a = 25
CASE 3
  a = 22
CASE 4
  a = 19
CASE 5
  a = 15
CASE 6
  a = 12
CASE 7
  a = 8
CASE 8
  a = 5
CASE ELSE

```

```

a = 2
END SELECT

aa = a
ab = a + 14: ac = a + 21: ad = a + 28: ae = a + 35: af = a + 42
ag = a + 49: ah = a + 56: ai = a + 63: aj = a + 70: ak = a + 77

' this is for each screen
FOR iscr = 1 TO nscr

    COLOR lcl, bcl
    CALL box,43
    COLOR 13
    LOCATE 3, 28: PRINT "ROT / RELIABILITY TABLE"
    COLOR 10
    LOCATE 4, 19: PRINT "(This Is for Evaluating an Existing Runway)"

' this is to include all exits
FOR kex = 1 TO nextit(1)
    FOR air = ia.s(iscr) TO la.e(iscr)

        icol = (kex - 1) * 7 + aa + 15
        irow = (air - ((iscr - 1) * 4) - 1) * 7 + 11

        COLOR 14
        LOCATE irow - 1, aa + 2: PRINT acflist$(air)
        COLOR 15

' to print the reliability & ROT
        IF (relrun(air, 1, kex) > .049) THEN
            LOCATE irow, icol:
            PRINT USING "### ##"; rotrun(air, 1, kex)
            LOCATE irow + 1, icol:
            PRINT USING "### ##"; relrun(air, 1, kex)
        END IF

        IF (relrun(air, 2, kex) > .049) THEN
            LOCATE irow + 3, icol:
            PRINT USING "### ##"; rotrun(air, 2, kex)
            LOCATE irow + 4, icol:
            PRINT USING "### ##"; relrun(air, 2, kex)
        END IF
    NEXT air
NEXT kex

    COLOR 7

' This draws the table depending the number of exits
' and the number of aircrafts on that screen

    LOCATE 5, aa: PRINT CHR$(218); STRING$(13, 196);
    t1$ = ""
    FOR kkex = 1 TO nextit(1)
        t1$ = t1$ + CHR$(194) + STRING$(6, 196)
    NEXT kkex
    LOCATE 5, aa + 14: PRINT t1$, CHR$(191)

    LOCATE 6, aa: PRINT CHR$(179)
    LOCATE 7, aa: PRINT CHR$(179)
    LOCATE 8, aa: PRINT CHR$(179)

    FOR i = 0 TO nextit(1)
        LOCATE 6, ab + i * 7: PRINT CHR$(179)
        LOCATE 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 8, ab + i * 7: PRINT CHR$(179)
    NEXT i

    LOCATE 9, aa: PRINT CHR$(195); STRING$(13, 196);
    t1$ = ""
    FOR kkex = 1 TO nextit(1)
        t1$ = t1$ + CHR$(197) + STRING$(6, 196)
    NEXT kkex
    LOCATE 9, aa + 14: PRINT t1$, CHR$(180)

' for naming the sections and numbering accordingly

```

```

COLOR 14
LOCATE 6, aa + 2: PRINT "Exit #"
COLOR 15
LOCATE 7, aa + 2: PRINT "Location (m)"
LOCATE 8, aa + 2: PRINT "Exit Type"

FOR i = 1 TO nextit(1)
    COLOR 14
    LOCATE 6, aa + 10 + i * 7: PRINT i
    COLOR 15
    LOCATE 7, aa + 8 + i * 7: PRINT exloc(i)
    IF extype(i) = 1 THEN
        LOCATE 8, aa + 8 + i * 7: PRINT "30-Deg"
    ELSEIF extype(i) = 2 THEN
        LOCATE 8, aa + 8 + i * 7: PRINT "45-Deg"
    ELSEIF extype(i) = 3 THEN
        LOCATE 8, aa + 8 + i * 7: PRINT "90-Deg"
    ELSE
        LOCATE 8, aa + 8 + i * 7: PRINT " New "
    END IF
NEXT i

COLOR 7
FOR k = 0 TO ia.e(lscr) - ((lscr - 1) ^ 4) - 1

    LOCATE 9 + 1 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 2 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 3 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 4 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 5 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 6 + k * 7, aa: PRINT CHR$(179)

    FOR i = 0 TO nextit(1)
        LOCATE 9 + 1 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 2 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 3 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 4 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 5 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 6 + k * 7, ab + i * 7: PRINT CHR$(179)
    NEXT i

    hh = 9 + 7 * (k + 1)

    IF k < ia.e(lscr) - ((lscr - 1) ^ 4) - 1 THEN
        ' if this is the not the last line

        LOCATE hh, aa: PRINT CHR$(195); STRING$(13, 196);
        tl$ = ""
        FOR kkex = 1 TO nextit(1)
            tl$ = tl$ + CHR$(197) + STRING$(6, 196)
        NEXT kkex
        LOCATE hh, aa + 14: PRINT tl$; CHR$(180)

    ELSE
        ' if this is the last line (note change in ASCII code)

        LOCATE hh, aa: PRINT CHR$(192); STRING$(13, 196);
        tl$ = ""
        FOR kkex = 1 TO nextit(1)
            tl$ = tl$ + CHR$(193) + STRING$(6, 196)
        NEXT kkex
        LOCATE hh, aa + 14: PRINT tl$; CHR$(217)

    END IF

NEXT k

FOR k = 0 TO ia.e(lscr) - ((lscr - 1) ^ 4) - 1

    COLOR 15
    LOCATE 9 + 2 + k * 7, aa + 10: PRINT "ROT"
    LOCATE 9 + 3 + k * 7, aa + 10: PRINT "REL"

```

```

LOCATE 9 + 5 + k * 7, aa + 10: PRINT "ROT"
LOCATE 9 + 6 + k * 7, aa + 10: PRINT "REL"
COLOR 10
LOCATE 9 + 2 + k * 7, aa + 2: PRINT "DRY"
LOCATE 9 + 5 + k * 7, aa + 2: PRINT "WET"

' to print the frequencies

FOR air = ia.s(iscr) TO la.e(iscr)

    freq = mixlist(air) * p.wth(1) / msum
    LOCATE 9 + 3 + k * 7, aa + 2: PRINT "("
    LOCATE 9 + 3 + k * 7, aa + 7: PRINT " %)"
    LOCATE 9 + 3 + k * 7, aa + 3: PRINT USING "###.##"; freq

    freq = mixlist(air) * p.wth(2) / msum
    LOCATE 9 + 6 + k * 7, aa + 2: PRINT "("
    LOCATE 9 + 6 + k * 7, aa + 7: PRINT " %)"
    LOCATE 9 + 6 + k * 7, aa + 3: PRINT USING "###.##"; freq

    COLOR 7

NEXT air
NEXT k

LOCATE hh + 1, aa + 1: PRINT "ROT - Runway Occupancy Time in Secs"
LOCATE hh + 2, aa + 1: PRINT "REL - Reliability "
COLOR 15
LOCATE hh + 3, aa + 1: PRINT "Average ROT = ";
LOCATE hh + 3, aa + 15: PRINT USING "###.##"; avg.rot
LOCATE 43, 2: PRINT "Press any key to continue ";
COLOR 7

CALL fc.inkey(W$)

NEXT iscr

20
COLOR lcl, bcl
WIDTH 80, 25

END SUB

This subroutine provides a table with the exit numbers, exit location, exit types and the ROT and reliability for each of the aircraft
selected. The maximum number of exits that can be shown in this table are nine. Moreover, as far as four aircraft can be shown in
each screen. The subroutine centers the table depending on the number of exits. The average ROT is shown at the bottom of the table.
The inputs to this subroutine are the ROT and reliability. This subroutine is applicable for only runmode '1'.

SUB table.AB.2 (rotrun(), relrun())

' This subroutine displays the table with the ROT & Reliability
' for each of the aircraft (upto four on each screen)
' for both wet and dry conditions. This pertains to runmode '2' & '3'

SCREEN 9
WIDTH 80, 43

' this is to get 1 - 4 aircraft on first screen, 5 - 8 on second screen, etc.
nscr = INT((nmix - .5) / 4) + 1
ia.s(1) = 1: ia.s(2) = 5: ia.s(3) = 9: ia.s(4) = 13: ia.s(5) = 17
ia.s(6) = 21: ia.s(7) = 25

FOR iss = 1 TO nscr
    IF (nmix >= iss * 4) THEN
        la.e(iss) = iss * 4
    ELSE
        la.e(iss) = nmix
    END IF
NEXT iss
IF runmode = 2 THEN
    nex = nextit(1) + nextit(2)
ELSE
    nex = nextit(2)
END IF

' to fix the location of the table depending on the number of exits
' i.e., for centering the table on the screen

```

```

SELECT CASE (nex)
CASE 1
a = 29
CASE 2
a = 25
CASE 3
a = 22
CASE 4
a = 19
CASE 5
a = 15
CASE 6
a = 12
CASE 7
a = 8
CASE 8
a = 5
CASE ELSE
a = 2
END SELECT

aa = a
ab = a + 14: ac = a + 21: ad = a + 28: ae = a + 35: af = a + 42
ag = a + 49: ah = a + 56: ai = a + 63: aj = a + 70: ak = a + 77

' for every screen ( four aircraft per screen)
FOR iscr = 1 TO nscr

COLOR lcl, bcl
CALL box.43

COLOR 13
LOCATE 3, 28: PRINT "ROT / RELIABILITY TABLE"
COLOR 10

IF runmode = 2 THEN
LOCATE 4, 19: PRINT "(This Is for Improving an Existing Runway)"
ELSE
LOCATE 4, 19: PRINT " (This Is for Designing a New Runway)"
END IF

' to print the aircraft names, ROT etc.,

FOR kex = 1 TO (nex)
FOR air = la.s(iscr) TO ia.e(iscr)

icol = (kex - 1) * 7 + aa + 15
irow = (air - ((iscr - 1) * 4) - 1) * 7 + 11

COLOR 14
LOCATE irow - 1, aa + 2: PRINT acflist$(air)
COLOR 15

LOCATE irow + 1, icol:
IF rotrun(air, 1, kex) = 0 THEN
PRINT CHR$(32)
ELSE
PRINT USING "###.##"; rotrun(air, 1, kex)
END IF
LOCATE irow + 3, icol:
IF rotrun(air, 2, kex) = 0 THEN
PRINT CHR$(32)
ELSE
PRINT USING "###.##"; rotrun(air, 2, kex)
END IF

NEXT air
NEXT kex

COLOR 7

' this draws the table consistent with the number of
' exits and the number of aircraft to be shown

LOCATE 5, aa: PRINT CHR$(218); STRING$(13, 196);
tl$ = ""
FOR kkex = 1 TO nex
tl$ = tl$ + CHR$(194) + STRING$(6, 196)

```

```

NEXT kkex
LOCATE 5, aa + 14: PRINT tl$; CHR$(191)

LOCATE 6, aa: PRINT CHR$(179)
LOCATE 7, aa: PRINT CHR$(179)
LOCATE 8, aa: PRINT CHR$(179)

FOR i = 0 TO (nex)
    LOCATE 6, ab + i * 7: PRINT CHR$(179)
    LOCATE 7, ab + i * 7: PRINT CHR$(179)
    LOCATE 8, ab + i * 7: PRINT CHR$(179)
NEXT i

LOCATE 9, aa: PRINT CHR$(195); STRING$(13, 196);
tl$ = ""
FOR kkex = 1 TO nex
    tl$ = tl$ + CHR$(197) + STRING$(6, 196)
NEXT kkex
LOCATE 9, aa + 14: PRINT tl$; CHR$(180)

' for naming the sections in the table and numbering accordingly

COLOR 14
LOCATE 6, aa + 2: PRINT "Exit #"
COLOR 15
LOCATE 7, aa + 2: PRINT "Location (m)"
LOCATE 8, aa + 2: PRINT "Exit Type"

FOR i = 1 TO (nex)
    COLOR 14
    LOCATE 6, aa + 10 + i * 7: PRINT i
    COLOR 15
    LOCATE 7, aa + 8 + i * 7
    IF runmode = 2 THEN
        PRINT USING "#### #"; exloc(i)
        IF extype(i) = 0 THEN
            LOCATE 8, aa + 8 + i * 7: PRINT "New "
        ELSEIF extype(i) = 1 THEN
            LOCATE 8, aa + 8 + i * 7: PRINT "30-Deg"
        ELSEIF extype(i) = 2 THEN
            LOCATE 8, aa + 8 + i * 7: PRINT "45-Deg"
        ELSEIF extype(i) = 3 THEN
            LOCATE 8, aa + 8 + i * 7: PRINT "90-Deg"
        END IF
    ELSE
        PRINT USING "#### #"; exloc.new(i)
        LOCATE 8, aa + 8 + i * 7: PRINT "New "
    END IF
NEXT i

COLOR 7

FOR k = 0 TO ia.e(iscr) - ((iscr - 1) * 4) - 1

    LOCATE 9 + 1 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 2 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 3 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 4 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 5 + k * 7, aa: PRINT CHR$(179)
    LOCATE 9 + 6 + k * 7, aa: PRINT CHR$(179)

    FOR i = 0 TO (nex)
        LOCATE 9 + 1 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 2 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 3 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 4 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 5 + k * 7, ab + i * 7: PRINT CHR$(179)
        LOCATE 9 + 6 + k * 7, ab + i * 7: PRINT CHR$(179)
    NEXT i

    hh = 9 + 7 * (k + 1)

' If this is not the last line, then

IF k < ia.e(iscr) - ((iscr - 1) * 4) - 1 THEN

```

```

LOCATE hh, aa: PRINT CHR$(195); STRING$(13, 196);
tl$ = ""
FOR kkex = 1 TO nex
tl$ = tl$ + CHR$(197) + STRING$(6, 196)
NEXT kkex
LOCATE hh, aa + 14: PRINT tl$; CHR$(180)

ELSE

' if this is the last line ( note change in ASCII code)

LOCATE hh, aa: PRINT CHR$(192); STRING$(13, 196);
tl$ = ""
FOR kkex = 1 TO nex
tl$ = tl$ + CHR$(193) + STRING$(6, 196)
NEXT kkex
LOCATE hh, aa + 14: PRINT tl$; CHR$(217)

END IF

NEXT k

' to print the frequencies

FOR k = 0 TO la.e(iscr) - ((iscr - 1) * 4) - 1

    COLOR 10
    LOCATE 9 + 5 + k * 7, aa + 2: PRINT "WET"
    LOCATE 9 + 3 + k * 7, aa + 2: PRINT "DRY "
    COLOR 15
    LOCATE 9 + 3 + k * 7, aa + 10: PRINT "ROT"
    LOCATE 9 + 5 + k * 7, aa + 10: PRINT "ROT"
    COLOR 7

    FOR air = la.s(iscr) TO la.e(iscr)

        freq = mixlist(air) * p.wth(1) / msum
        LOCATE 9 + 4 + k * 7, aa + 2: PRINT "("
        LOCATE 9 + 4 + k * 7, aa + 7: PRINT "%)"
        LOCATE 9 + 4 + k * 7, aa + 3: PRINT USING "###.#"; freq

        freq = mixlist(air) * p.wth(2) / msum
        LOCATE 9 + 6 + k * 7, aa + 2: PRINT "("
        LOCATE 9 + 6 + k * 7, aa + 7: PRINT "%)"
        LOCATE 9 + 6 + k * 7, aa + 3: PRINT USING "###.#"; freq

    NEXT air
NEXT k

LOCATE hh + 1, aa + 1: PRINT "ROT - Runway Occupancy Time in Secs"
LOCATE hh + 2, aa + 1: PRINT "Reliability in % = "; rel
COLOR 15
LOCATE hh + 3, aa + 1: PRINT "Average ROT = ";
LOCATE hh + 3, aa + 14: PRINT USING "###.###"; avg.rot

COLOR 7
CALL fc.inkey(W$)

NEXT iscr

40
COLOR lcl, bcl
WIDTH 80, 25

COLOR 15

END SUB

```

This subroutine is very similar to the subroutine Table.AB.1. But, this subroutine can be used for both runmodes '2' and '3'. The inputs to this subroutine are the ROT, and the reliability. Unlike in subroutine Table.AB.1, the reliability is not provided in the table. It is displayed outside the table along with the average ROT.

SUB tax.exit.geo (des.grp)

' this subroutine gives the values of taxiway width, centerline radius,
' length of fillet lead-in and fillet radius depending on design group.

```
mts = .3048 ' for conversion from feet to meters
```

```
SELECT CASE des.grp
```

```
CASE 1
```

```
tw = mts * 25  
R = mts * 75  
L = mts * 50  
F = mts * 60
```

```
CASE 2
```

```
tw = mts * 35  
R = mts * 75  
L = mts * 50  
F = mts * 55
```

```
CASE 3
```

```
tw = mts * 50  
R = mts * 100  
L = mts * 150  
F = mts * 55
```

```
CASE 4
```

```
tw = mts * 75  
R = mts * 150  
L = mts * 250  
F = mts * 80
```

```
CASE 5
```

```
tw = mts * 75  
R = mts * 150  
L = mts * 250  
F = mts * 85
```

```
CASE ELSE
```

```
tw = mts * 100  
R = mts * 170  
L = mts * 250  
F = mts * 85
```

```
END SELECT
```

```
END SUB
```

This subroutine specifies the values of the width of taxiway, centerline radius, fillet radius and the length of the fillet based on the design group. These values are of great use while displaying the turnoff geometries.

```
DEFSTR W
```

```
SUB tbl.exloc (nexit, xea())
```

```
' This is for the table of exit number and their location(s)
```

```
' This draws the table with the number of lines being  
' consistent with half the number of exits
```

```
LOCATE 15, 6: PRINT CHR$(218); STRING$(9, 196); CHR$(194)  
LOCATE 15, 17: PRINT STRING$(9, 196); CHR$(194);  
LOCATE 15, 27: PRINT STRING$(9, 196); CHR$(194)  
LOCATE 15, 37: PRINT STRING$(9, 196); CHR$(191)  
LOCATE 16, 6: PRINT CHR$(179); LOCATE 16, 16: PRINT CHR$(179);  
LOCATE 16, 26: PRINT CHR$(179); LOCATE 16, 36: PRINT CHR$(179)  
LOCATE 16, 46: PRINT CHR$(179);  
LOCATE 16, 9: PRINT "EXIT #": LOCATE 16, 18: PRINT "LOCATION"  
LOCATE 16, 29: PRINT "EXIT #": LOCATE 16, 38: PRINT "LOCATION"  
LOCATE 17, 6: PRINT CHR$(195); STRING$(9, 196); CHR$(197)  
LOCATE 17, 17: PRINT STRING$(9, 196); CHR$(197);  
LOCATE 17, 27: PRINT STRING$(9, 196); CHR$(197)  
LOCATE 17, 37: PRINT STRING$(9, 196); CHR$(180)
```

```
nrow = INT(nexit / 2 + .99)
```

```
FOR ii = 1 TO nrow
```

```
LOCATE ii + 17, 6: PRINT CHR$(179); LOCATE ii + 17, 26: PRINT CHR$(179)  
LOCATE ii + 17, 16: PRINT CHR$(179); LOCATE ii + 17, 36: PRINT CHR$(179)  
LOCATE ii + 17, 46: PRINT CHR$(179);
```

```
NEXT ii
```

```
LOCATE nrow + 17, 6: PRINT CHR$(179); LOCATE nrow + 17, 26: PRINT CHR$(179)  
LOCATE nrow + 17, 36: PRINT CHR$(179); LOCATE nrow + 17, 46: PRINT CHR$(179)  
LOCATE nrow + 17, 16: PRINT CHR$(179);  
LOCATE nrow + 18, 6: PRINT CHR$(192); STRING$(9, 196); CHR$(193)  
LOCATE nrow + 18, 17: PRINT STRING$(9, 196); CHR$(193)  
LOCATE nrow + 18, 27: PRINT STRING$(9, 196); CHR$(193)  
LOCATE nrow + 18, 37: PRINT STRING$(9, 196); CHR$(217)
```

```
' To print the exit(s) and their location(s) consistent
```

```

' with the colors
FOR ii = 1 TO nextit
  lr = ii - INT(ii / nrow - .01) * nrow
  LC = INT(ii / nrow + .99)
  lcx = (LC - 1) * 20 + 10
  lrx = 17 + lr
  CALL clr.dec(ii, jk)
  COLOR jk, bcl
  LOCATE lrx, lcx: PRINT USING "##"; (ii)
  LOCATE lrx, lcx + 8: PRINT USING "####.##"; xea(ii)
NEXT ii

END SUB

```

This subroutine creates a table and displays the exit number(s) and their location(s). The input for this subroutine is nextit, the number of exits. The subroutine makes the table in such a manner that half the exit numbers and their locations are displayed on the first two columns, and the rest in the other two columns. Subroutine clr.dec is called here to assist the main subroutine in assigning appropriate colors in consistence with the ones in the subroutine Locexit which displays the exit locations.

```

DEFSNG W
SUB tbl.turnoff (ntr, xtra(), ytra(), exnum, exarr())

' This is for the table of Runway Turnoff Coordinates

```

```

SCREEN 9
COLOR lcl, bcl
CALL box
CALL box.small2

```

```

' This draws the table for the turnoff coordinates
' (in three pairs of columns)

```

```

LOCATE 8, 4: PRINT CHR$(218); STRING$(23, 196); CHR$(196)
LOCATE 8, 29: PRINT STRING$(23, 196); CHR$(196);
LOCATE 8, 53: PRINT STRING$(23, 196); CHR$(191)
LOCATE 10, 4: PRINT CHR$(195); STRING$(11, 196); CHR$(194)
LOCATE 10, 17: PRINT STRING$(11, 196); CHR$(194)
LOCATE 10, 29: PRINT STRING$(11, 196); CHR$(194)
LOCATE 10, 41: PRINT STRING$(11, 196); CHR$(194)
LOCATE 10, 53: PRINT STRING$(11, 196); CHR$(194)
LOCATE 10, 65: PRINT STRING$(11, 196); CHR$(180)
LOCATE 9, 4: PRINT CHR$(179);
LOCATE 9, 76: PRINT CHR$(179);
LOCATE 11, 76: PRINT CHR$(179); LOCATE 11, 64: PRINT CHR$(179)
LOCATE 11, 16: PRINT CHR$(179); LOCATE 11, 40: PRINT CHR$(179)
LOCATE 9, 16: PRINT "RUNWAY TURNOFF GEOMETRY COORDINATES"
LOCATE 11, 4: PRINT CHR$(179); LOCATE 11, 28: PRINT CHR$(179)
LOCATE 11, 52: PRINT CHR$(179)
LOCATE 11, 10: PRINT "X"; LOCATE 11, 22: PRINT "Y"
LOCATE 11, 34: PRINT "X"; LOCATE 11, 46: PRINT "Y"
LOCATE 11, 58: PRINT "X"; LOCATE 11, 70: PRINT "Y"
LOCATE 12, 4: PRINT CHR$(195); STRING$(11, 196); CHR$(197)
LOCATE 12, 17: PRINT STRING$(11, 196); CHR$(197)
LOCATE 12, 29: PRINT STRING$(11, 196); CHR$(197)
LOCATE 12, 41: PRINT STRING$(11, 196); CHR$(197)
LOCATE 12, 53: PRINT STRING$(11, 196); CHR$(197)
LOCATE 12, 65: PRINT STRING$(11, 196); CHR$(180)

```

```

nrow = INT(ntr / 3 + .99)

```

```

FOR ii = 1 TO nrow
  LOCATE ii + 12, 4: PRINT CHR$(179); LOCATE ii + 12, 28: PRINT CHR$(179)
  LOCATE ii + 12, 16: PRINT CHR$(179); LOCATE ii + 12, 76: PRINT CHR$(179)
  LOCATE ii + 12, 52: PRINT CHR$(179); LOCATE ii + 12, 40: PRINT CHR$(179)
  LOCATE ii + 12, 64: PRINT CHR$(179);
NEXT ii
LOCATE nrow + 12, 4: PRINT CHR$(179); LOCATE nrow + 12, 28: PRINT CHR$(179)
LOCATE nrow + 12, 16: PRINT CHR$(179); LOCATE nrow + 12, 76: PRINT CHR$(179)
LOCATE nrow + 12, 40: PRINT CHR$(179); LOCATE nrow + 12, 52: PRINT CHR$(179)
LOCATE nrow + 12, 64: PRINT CHR$(179);
LOCATE nrow + 13, 4: PRINT CHR$(192); STRING$(11, 196)
LOCATE nrow + 13, 16: PRINT CHR$(193); STRING$(11, 196); CHR$(193)
LOCATE nrow + 13, 29: PRINT STRING$(11, 196); CHR$(193)
LOCATE nrow + 13, 41: PRINT STRING$(11, 196); CHR$(193)
LOCATE nrow + 13, 53: PRINT STRING$(11, 196); CHR$(193)
LOCATE nrow + 13, 65: PRINT STRING$(11, 196); CHR$(217)

```

```

' This is to print the coordinates for each of the turnoffs
' with the corresponding color using subroutine clr.dec

```

```

FOR jj = 1 TO exnum
  CALL clr.dec(exarr(jj), jk)
  COLOR jk, bcl
  FOR ii = 1 TO ntr
    xturn(ii) = xtra(exarr(jj), ii)
    yturn(ii) = ytra(exarr(jj), ii)
    lr = ii - INT(ii / nrow - .01) * nrow
    LC = INT(ii / nrow + .99)
    lcx = (LC - 1) * 24 + 6
    lrx = 12 + lr
    LOCATE lrx, lcx: PRINT USING "#### ##"; xturn(ii)
    LOCATE lrx, lcx + 11: PRINT USING "#### ##"; yturn(ii)
  NEXT ii
  LOCATE 9, 52: PRINT "FOR EXIT # "
  LOCATE 9, 62: PRINT exarr(jj)
  COLOR 10, bcl
  LOCATE 25, 2: PRINT "Press any key to continue";
  COLOR lcl, bcl
  CALL fc.inkey(wd$)
NEXT jj

END SUB

```

The main purpose of this subroutine is to make a table with three pairs of columns for each, x, y coordinates of the turnoff geometry. Depending on the number of turnoff geometries the user wishes to view, the subroutine runs iteratively for each turnoff to display the coordinates of the particular turnoff. The inputs for this subroutine are ntr, the number of turnoff geometry coordinates; the x, y coordinates; exnum, the exit number, and exarr(), the exit array. The coordinates of the turnoff (xtra, ytra) which are in two dimensional form are converted into one dimensional arrays, as this enables the program to print these coordinates in the table.

```
SUB view.menu1 (rotrun(), relrun())
```

```
' this displays the view menu screen for runmode '1'
```

```

SCREEN 0
tcl = 14
COLOR lcl, bcl
CALL box
CALL stitle("VIEW MENU")
CALL box.small
CALL guide.ed.out.opt
COLOR ccl, bcl
W$ = " 1)View ROT Table"
ih = 10: ichoice = 24: CALL print.str(W$): is1 = POS(n)

W$ = " 2)View Turnoff Locations And Their Geometries"
ih = 12: ichoice = 24: CALL print.str(W$): is2 = POS(n)

W$ = " 3)Go To Output Menu"
ih = 14: ichoice = 24: CALL print.str(W$): is3 = POS(n)

lk = 1
DO
  COLOR bcl, ccl
  SELECT CASE lk
    CASE 1
      W$ = " 1)View ROT Table"
      ih = 10: ichoice = 24: CALL print.str(W$)
      LOCATE 8, (is1 - LEN(W$))
      CALL get.ival(nj)
      COLOR ccl, bcl: CALL print.str(W$)
      ednum = 1
    CASE 2
      W$ = " 2)View Turnoff Locations And Their Geometries"
      ih = 12: ichoice = 24: CALL print.str(W$): LOCATE 10, is2 - LEN(W$)
      CALL get.ival(nj)
      COLOR ccl, bcl: CALL print.str(W$)
      ednum = 2
    CASE 3
      W$ = " 3)Go To Output Menu"
      ih = 14: ichoice = 24: CALL print.str(W$): LOCATE 12, is3 - LEN(W$)
      CALL get.ival(nj)
      COLOR ccl, bcl: CALL print.str(W$)
      ednum = 3

    CASE ELSE
      nj = 1
  END SELECT

```

```

        Ik = Ik + nj
        IF Ik = 0 THEN Ik = 3
        IF Ik = 4 THEN Ik = 1
    LOOP UNTIL nj >= 2

    IF nj = 3 THEN
        EXIT SUB
    ELSEIF ednum = 2 THEN
        ' CALL turnoff geometries
        CALL pexl.run1:
        CALL view.menu1(rotrun(), relrun())
    ELSEIF ednum = 1 THEN
        CALL table AB.1(rotrun(), relrun())
        CALL view.menu1(rotrun(), relrun())
    ELSE
        EXIT SUB
    END IF

    CLS

END SUB

```

This subroutine displays the view menu screen. The view menu screen corresponds to runmode '1'. The subroutines are called according to the option selected by the user. Various screens are connected through this subroutine.

```

SUB view.menu2 (rotrun(), relrun())

' this displays the view menu screen for runmode '2'

SCREEN 0
tcl = 14
COLOR tcl, bcl
CALL box
CALL stitle("VIEW MENU")
CALL box.small
CALL guide.ed.out.opt
COLOR ccl, bcl
W$ = " 1)View ROT/REL Table"
ih = 10: ichoice = 24: CALL print.str(W$): is1 = POS(n)

W$ = " 2)View Turnoff Locations And Their Geometries"
ih = 12: ichoice = 24: CALL print.str(W$): is2 = POS(n)

W$ = " 3)View Aircraft Statistics"
ih = 14: ichoice = 24: CALL print.str(W$): is3 = POS(n)

W$ = " 4)Go To Output Menu"
ih = 16: ichoice = 24: CALL print.str(W$): is4 = POS(n)

Ik = 1
DO
    COLOR bcl, ccl
    SELECT CASE Ik
        CASE 1
            W$ = " 1)View ROT/REL Table"
            ih = 10: ichoice = 24: CALL print.str(W$)
            LOCATE 8, (is1 - LEN(W$))
            CALL get.ival(nj)
            COLOR ccl, bcl: CALL print.str(W$)
            ednum = 1
        CASE 2
            W$ = " 2)View Turnoff Locations And Their Geometries"
            ih = 12: ichoice = 24: CALL print.str(W$): LOCATE 10, is2 - LEN(W$)
            CALL get.ival(nj)
            COLOR ccl, bcl: CALL print.str(W$)
            ednum = 2
        CASE 3
            W$ = " 3)View Aircraft Statistics"
            ih = 14: ichoice = 24: CALL print.str(W$): LOCATE 12, is3 - LEN(W$)
            CALL get.ival(nj)
            COLOR ccl, bcl: CALL print.str(W$)
            ednum = 3
        CASE 4
            W$ = " 4)Go To Output Menu"
            ih = 16: ichoice = 24: CALL print.str(W$): LOCATE 14, is4 - LEN(W$)
            CALL get.ival(nj)
            COLOR ccl, bcl: CALL print.str(W$)
            ednum = 4
    END SELECT
    IF ednum = 0 THEN
        EXIT DO
    END IF
    IF ednum = 1 THEN
        CALL view.menu1(rotrun(), relrun())
    ELSEIF ednum = 2 THEN
        CALL turnoff.geometries(rotrun(), relrun())
    ELSEIF ednum = 3 THEN
        CALL aircraft.statistics(rotrun(), relrun())
    ELSEIF ednum = 4 THEN
        CALL output.menu(rotrun(), relrun())
    END IF
    Ik = Ik + 1
LOOP UNTIL Ik = 4

```

```

CASE ELSE
nj = 1
END SELECT
Ik = Ik + nj
IF Ik = 0 THEN Ik = 4
IF Ik = 5 THEN Ik = 1
LOOP UNTIL nj >= 2

IF nj = 3 THEN
EXIT SUB
ELSEIF ednum = 3 THEN
CALL stat.bar2(rotrun(), relrun())
CALL view.menu2(rotrun(), relrun())
ELSEIF ednum = 2 THEN
CALL pexl.run2
CALL view.menu2(rotrun(), relrun())
ELSEIF ednum = 1 THEN
CALL table.AB.2(rotrun(), relrun())
CALL view.menu2(rotrun(), relrun())
ELSE
EXIT SUB
END IF

CLS

END SUB

```

The main purpose of this subroutine is to display the view menu screen corresponding to the runmode option '2'. The inputs to this subroutine are the ROT and the reliability. Most of the screens that pertain to this runmode are connected through this subroutine.

SUB view.menu3 (rotrun(), relrun())

' this displays the veiw menu screen for runmode '3'

```

SCREEN 0
tcl = 14
COLOR tcl, bcl
CALL box
CALL stitle("VIEW MENU")
CALL box.small
CALL guide.ed.out.opt
COLOR ccl, bcl
W$ = " 1)View ROT/REL Table"
ih = 10: ichoice = 24: CALL print.str(W$): Is1 = POS(n)

W$ = " 2)View Turnoff Locations And Their Geometries"
ih = 12: ichoice = 24: CALL print.str(W$): Is2 = POS(n)

W$ = " 3)View Aircraft Statistics"
ih = 14: ichoice = 24: CALL print.str(W$): Is3 = POS(n)

W$ = " 4)Go To Output Menu"
ih = 16: ichoice = 24: CALL print.str(W$): Is4 = POS(n)

```

```

Ik = 1
DO
COLOR bcl, cscl
SELECT CASE Ik
CASE 1
W$ = " 1)View ROT/REL Table"
ih = 10: ichoice = 24: CALL print.str(W$)
LOCATE 8, (Is1 - LEN(W$))
CALL get.ival(nj)
COLOR ccl, bcl: CALL print.str(W$)
ednum = 1
CASE 2
W$ = " 2)View Turnoff Locations And Their Geometries"
ih = 12: ichoice = 24: CALL print.str(W$): LOCATE 10, Is2 - LEN(W$)
CALL get.ival(nj)
COLOR ccl, bcl: CALL print.str(W$)
ednum = 2
CASE 3
W$ = " 3)View Aircraft Statistics"
ih = 14: ichoice = 24: CALL print.str(W$): LOCATE 12, Is3 - LEN(W$)
CALL get.ival(nj)
COLOR ccl, bcl: CALL print.str(W$)
ednum = 3
CASE 4
W$ = " 4)Go To Output Menu"

```

```

ih = 16: lchoice = 24: CALL print.str(W$): LOCATE 14, ls4 - LEN(W$)
CALL get.ival(nj)
COLOR ccl, bcl: CALL print.str(W$)
ednum = 4

CASE ELSE
nj = 1
END SELECT
lk = lk + nj
IF lk = 0 THEN lk = 4
IF lk = 5 THEN lk = 1
LOOP UNTIL nj >= 2

IF nj = 3 THEN
EXIT SUB
ELSEIF ednum = 3 THEN
CALL stat.bar3(rotrun(), relrun())
CALL view.menu3(rotrun(), relrun())
ELSEIF ednum = 2 THEN
CALL pexl.run3
CALL view.menu3(rotrun(), relrun())
ELSEIF ednum = 1 THEN
CALL table.AB.2(rotrun(), relrun())
CALL view.menu3(rotrun(), relrun())
ELSE
EXIT SUB
END IF

CLS

END SUB

```

This subroutine is very similar to the subroutine View.menu2. This view menu screen corresponds to the runmode option '3'. The inputs to this subroutine are the ROT and the reliability.

**The vita has been removed from
the scanned document**

Run Mode # 1
Analysis of an Existing Runway

OUTPUT MENU

VADM MENU Press escape (Esc) key to return to OUTPUT MENU.
 Use arrow (↑ or ↓) key to change your choice.
 Press enter (↵) key if the choice is correct.

1)View Runway Data
 2)View Turnoff Locations and Their Geometries
 3)Go To Output Menu

RUN RELIABILITY DATA
 (This is for Evaluating an Existing Runway)

Exit #	Runway	1000'	1500'	2000'	2500'	3000'
1	1000'	1.11	1.11	1.11	1.11	1.11
2	1000'	1.11	1.11	1.11	1.11	1.11
3	1000'	1.11	1.11	1.11	1.11	1.11
4	1000'	1.11	1.11	1.11	1.11	1.11
5	1000'	1.11	1.11	1.11	1.11	1.11
6	1000'	1.11	1.11	1.11	1.11	1.11
7	1000'	1.11	1.11	1.11	1.11	1.11
8	1000'	1.11	1.11	1.11	1.11	1.11
9	1000'	1.11	1.11	1.11	1.11	1.11
10	1000'	1.11	1.11	1.11	1.11	1.11
11	1000'	1.11	1.11	1.11	1.11	1.11
12	1000'	1.11	1.11	1.11	1.11	1.11
13	1000'	1.11	1.11	1.11	1.11	1.11
14	1000'	1.11	1.11	1.11	1.11	1.11
15	1000'	1.11	1.11	1.11	1.11	1.11
16	1000'	1.11	1.11	1.11	1.11	1.11
17	1000'	1.11	1.11	1.11	1.11	1.11
18	1000'	1.11	1.11	1.11	1.11	1.11
19	1000'	1.11	1.11	1.11	1.11	1.11
20	1000'	1.11	1.11	1.11	1.11	1.11
21	1000'	1.11	1.11	1.11	1.11	1.11
22	1000'	1.11	1.11	1.11	1.11	1.11
23	1000'	1.11	1.11	1.11	1.11	1.11
24	1000'	1.11	1.11	1.11	1.11	1.11
25	1000'	1.11	1.11	1.11	1.11	1.11
26	1000'	1.11	1.11	1.11	1.11	1.11
27	1000'	1.11	1.11	1.11	1.11	1.11
28	1000'	1.11	1.11	1.11	1.11	1.11
29	1000'	1.11	1.11	1.11	1.11	1.11
30	1000'	1.11	1.11	1.11	1.11	1.11
31	1000'	1.11	1.11	1.11	1.11	1.11
32	1000'	1.11	1.11	1.11	1.11	1.11
33	1000'	1.11	1.11	1.11	1.11	1.11
34	1000'	1.11	1.11	1.11	1.11	1.11
35	1000'	1.11	1.11	1.11	1.11	1.11
36	1000'	1.11	1.11	1.11	1.11	1.11
37	1000'	1.11	1.11	1.11	1.11	1.11
38	1000'	1.11	1.11	1.11	1.11	1.11
39	1000'	1.11	1.11	1.11	1.11	1.11
40	1000'	1.11	1.11	1.11	1.11	1.11
41	1000'	1.11	1.11	1.11	1.11	1.11
42	1000'	1.11	1.11	1.11	1.11	1.11
43	1000'	1.11	1.11	1.11	1.11	1.11
44	1000'	1.11	1.11	1.11	1.11	1.11
45	1000'	1.11	1.11	1.11	1.11	1.11
46	1000'	1.11	1.11	1.11	1.11	1.11
47	1000'	1.11	1.11	1.11	1.11	1.11
48	1000'	1.11	1.11	1.11	1.11	1.11
49	1000'	1.11	1.11	1.11	1.11	1.11
50	1000'	1.11	1.11	1.11	1.11	1.11
51	1000'	1.11	1.11	1.11	1.11	1.11
52	1000'	1.11	1.11	1.11	1.11	1.11
53	1000'	1.11	1.11	1.11	1.11	1.11
54	1000'	1.11	1.11	1.11	1.11	1.11
55	1000'	1.11	1.11	1.11	1.11	1.11
56	1000'	1.11	1.11	1.11	1.11	1.11
57	1000'	1.11	1.11	1.11	1.11	1.11
58	1000'	1.11	1.11	1.11	1.11	1.11
59	1000'	1.11	1.11	1.11	1.11	1.11
60	1000'	1.11	1.11	1.11	1.11	1.11
61	1000'	1.11	1.11	1.11	1.11	1.11
62	1000'	1.11	1.11	1.11	1.11	1.11
63	1000'	1.11	1.11	1.11	1.11	1.11
64	1000'	1.11	1.11	1.11	1.11	1.11
65	1000'	1.11	1.11	1.11	1.11	1.11
66	1000'	1.11	1.11	1.11	1.11	1.11
67	1000'	1.11	1.11	1.11	1.11	1.11
68	1000'	1.11	1.11	1.11	1.11	1.11
69	1000'	1.11	1.11	1.11	1.11	1.11
70	1000'	1.11	1.11	1.11	1.11	1.11
71	1000'	1.11	1.11	1.11	1.11	1.11
72	1000'	1.11	1.11	1.11	1.11	1.11
73	1000'	1.11	1.11	1.11	1.11	1.11
74	1000'	1.11	1.11	1.11	1.11	1.11
75	1000'	1.11	1.11	1.11	1.11	1.11
76	1000'	1.11	1.11	1.11	1.11	1.11
77	1000'	1.11	1.11	1.11	1.11	1.11
78	1000'	1.11	1.11	1.11	1.11	1.11
79	1000'	1.11	1.11	1.11	1.11	1.11
80	1000'	1.11	1.11	1.11	1.11	1.11
81	1000'	1.11	1.11	1.11	1.11	1.11
82	1000'	1.11	1.11	1.11	1.11	1.11
83	1000'	1.11	1.11	1.11	1.11	1.11
84	1000'	1.11	1.11	1.11	1.11	1.11
85	1000'	1.11	1.11	1.11	1.11	1.11
86	1000'	1.11	1.11	1.11	1.11	1.11
87	1000'	1.11	1.11	1.11	1.11	1.11
88	1000'	1.11	1.11	1.11	1.11	1.11
89	1000'	1.11	1.11	1.11	1.11	1.11
90	1000'	1.11	1.11	1.11	1.11	1.11
91	1000'	1.11	1.11	1.11	1.11	1.11
92	1000'	1.11	1.11	1.11	1.11	1.11
93	1000'	1.11	1.11	1.11	1.11	1.11
94	1000'	1.11	1.11	1.11	1.11	1.11
95	1000'	1.11	1.11	1.11	1.11	1.11
96	1000'	1.11	1.11	1.11	1.11	1.11
97	1000'	1.11	1.11	1.11	1.11	1.11
98	1000'	1.11	1.11	1.11	1.11	1.11
99	1000'	1.11	1.11	1.11	1.11	1.11
100	1000'	1.11	1.11	1.11	1.11	1.11

Press any key to continue

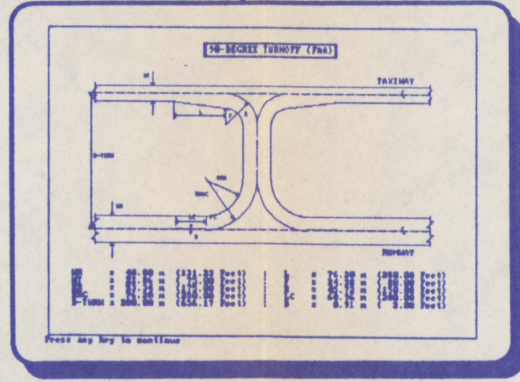
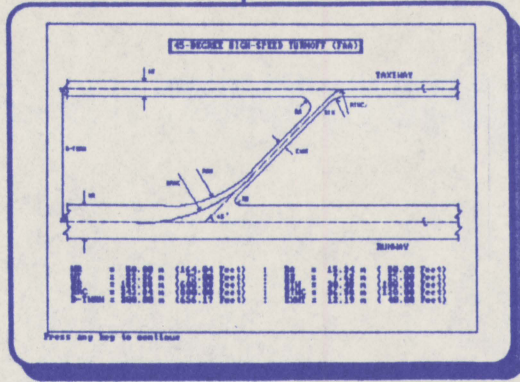
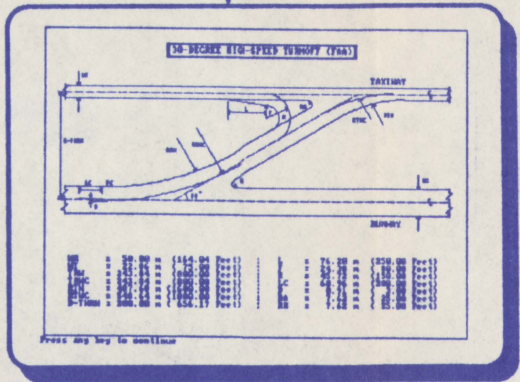
TURNOFF LOCATIONS AND THEIR GEOMETRIES
 AIRPORT CONDITIONS
 Temperature (C): 15.00
 Elevation (m): 1500.0
 Wind Speed (m/s): 0.00
 Weather (C): Dry - 50 & Wet - 50

REPEAT EXIT LOCATION DISPLAY
 0 500 1000 1500 2000 2500 3000 3500
 Runway Run Range (m)

EXIT #	LOCATION	EXIT #	LOCATION
1	1000.00	3	3000.00
2	2000.00		

Enter the Ball Number to view the complete Turnoff Geometry. (Enter '0' to exit)
 Example) 2
 ans) 2

GO TO OUTPUT MENU



Run Mode # 3
Design of a New Runway Facility

OUTPUT MENU

Press escape (Esc) key to return to MENU MENU

This screen (F) or (F) key to change your choice.
Press enter (Enter) key if the choice is correct.

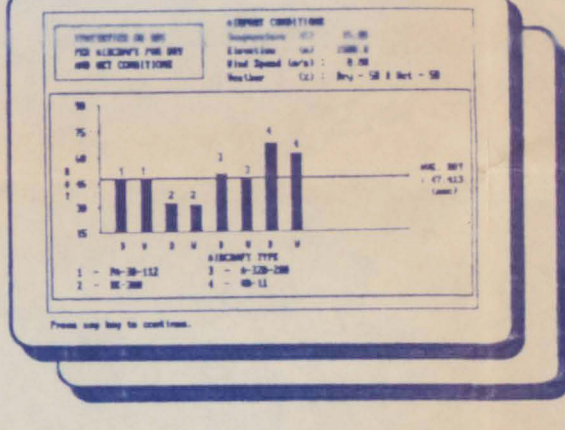
1) View MY-REQ. Table
2) View Turnoff Locations and Their Coordinates
3) View Runway Data
4) Go To Output Menu

TABLE 1: AIRCRAFT CHARACTERISTICS

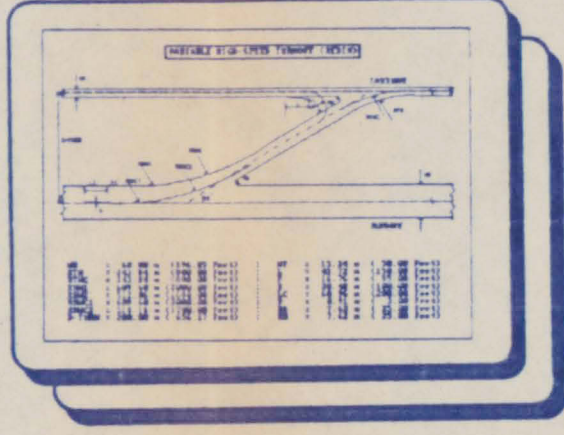
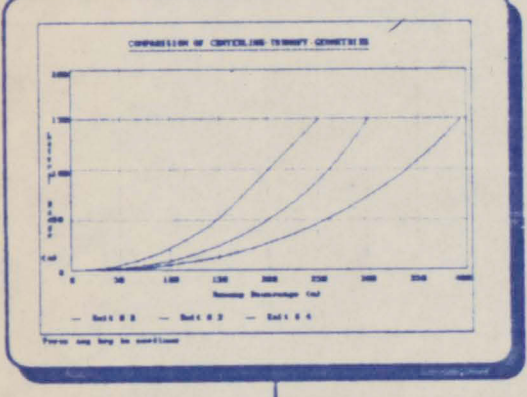
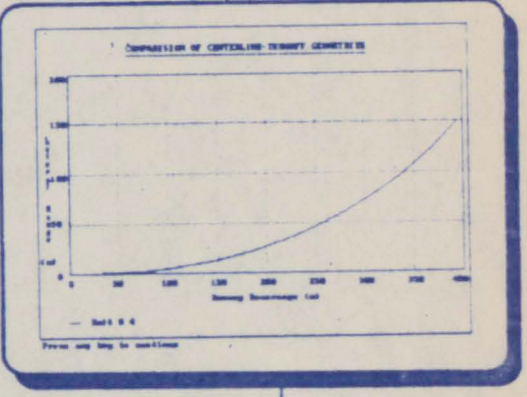
Category	Model	Wing Span (ft)	Wing Area (sq ft)	Wing Loading (lb/sq ft)	Wing Area (sq ft)	Wing Loading (lb/sq ft)
General	101	100	100	100	100	100
	102	100	100	100	100	100
	103	100	100	100	100	100
	104	100	100	100	100	100
Special	201	100	100	100	100	100
	202	100	100	100	100	100
	203	100	100	100	100	100
	204	100	100	100	100	100

TURNOFF LOCATION DISPLAY

Runway	Location	Distance (ft)	Direction
1	1100.75	2714.16	Runway
2	2000.00	2714.16	Runway



GO TO
OUTPUT MENU



TURNOFF GEOMETRIES COMPARISON FOR DIFFERENT

Category	Model	Wing Span (ft)	Wing Area (sq ft)	Wing Loading (lb/sq ft)	Wing Area (sq ft)	Wing Loading (lb/sq ft)
General	101	100	100	100	100	100
	102	100	100	100	100	100
	103	100	100	100	100	100
	104	100	100	100	100	100
Special	201	100	100	100	100	100
	202	100	100	100	100	100
	203	100	100	100	100	100
	204	100	100	100	100	100

Run Mode # 2
Redesign of an Existing Runway

OUTPUT MENU

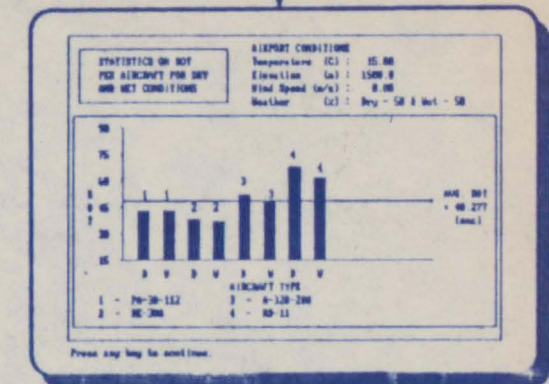
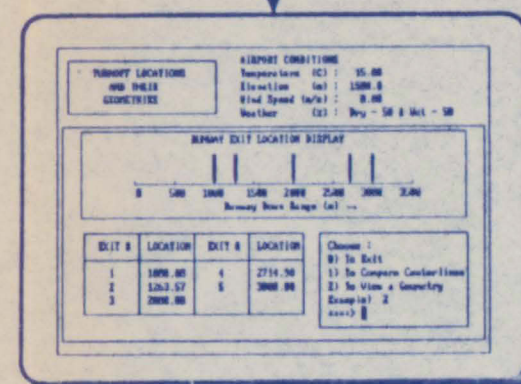
BACK MENU: Press escape (Esc) key to return to INPUT MENU.

Use arrow (↑ or ↓) key to change your choice.
Press enter (↵) key if the choice is correct.

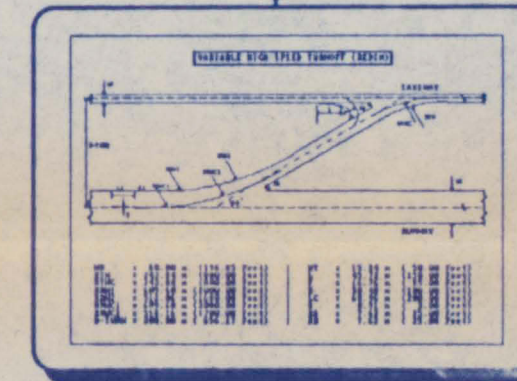
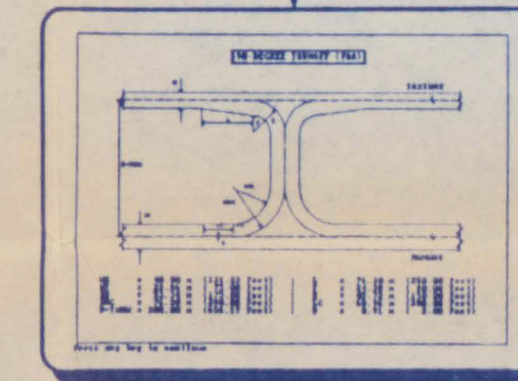
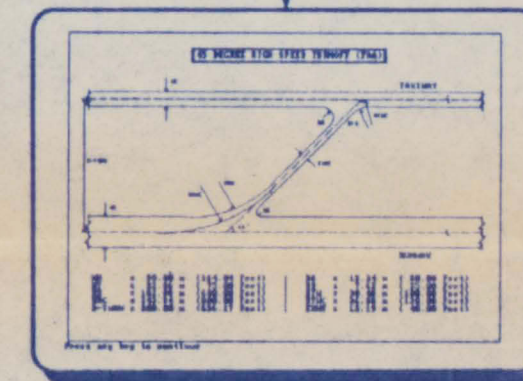
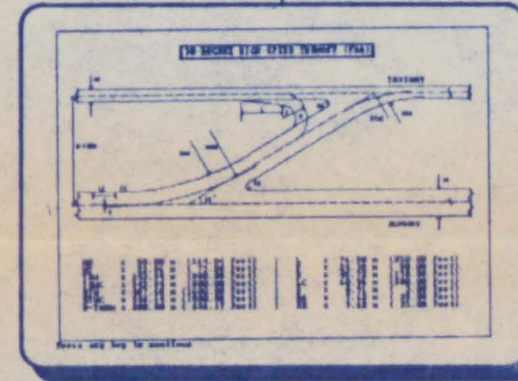
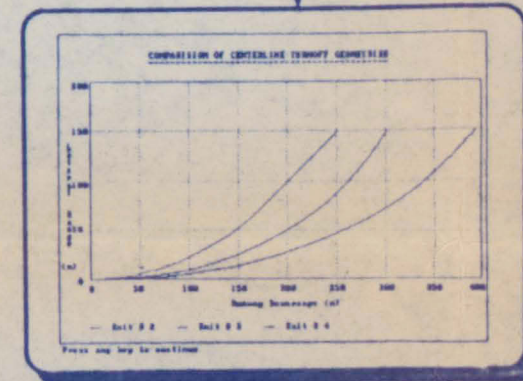
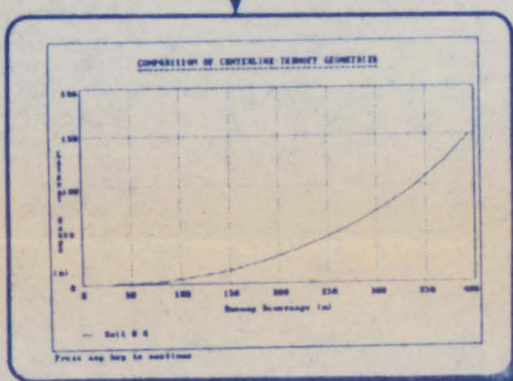
VIEW RUNWAY DATA
VIEW AIRPORT CONDITIONS AND WEATHER
VIEW AIRCRAFT POPULATION
VIEW AIRCRAFT SCHEDULE
VIEW TO OUTPUT MENU

TABLE 1: RUNWAY GEOMETRY DATA

Runway	Length (ft)	Width (ft)	Area (sq ft)	Volume (cu ft)
1	1000	100	100,000	1,000,000
2	1000	100	100,000	1,000,000
3	1000	100	100,000	1,000,000
4	1000	100	100,000	1,000,000
5	1000	100	100,000	1,000,000
6	1000	100	100,000	1,000,000
7	1000	100	100,000	1,000,000
8	1000	100	100,000	1,000,000
9	1000	100	100,000	1,000,000
10	1000	100	100,000	1,000,000



GO TO
OUTPUT MENU



Runway Geometry Data

Runway	Length (ft)	Width (ft)	Area (sq ft)	Volume (cu ft)
1	1000	100	100,000	1,000,000
2	1000	100	100,000	1,000,000
3	1000	100	100,000	1,000,000
4	1000	100	100,000	1,000,000
5	1000	100	100,000	1,000,000
6	1000	100	100,000	1,000,000
7	1000	100	100,000	1,000,000
8	1000	100	100,000	1,000,000
9	1000	100	100,000	1,000,000
10	1000	100	100,000	1,000,000