A ROUTE-ORIENTED SIMULATION SYSTEM FOR ATC STUDIES

W.R. Sutherland, T.H. Myer, E. L. Thomas, D. A. Henderson, Jr.

Bolt Beranek & Newman, Inc., Cambridge, Mass.

A simulation system for modeling air traffic situations is described. Real- or fast-time control of many aircraft over easily defined routes is provided. Simulated aircraft can be vectored manually or instructed to follow routes predefined by the experimenter. Special language forms have been developed for describing both the geographical and procedural aspects of these routes.

The program runs in a PDP-10 timeshared computer and capitalizes on some of the standard system features. The influence of the interactive computer environment on the program's development and operation is discussed.

I. Introduction

This paper describes a simulation programming system for air traffic control (ATC) research called the Route-Oriented Simulation System (ROSS). The system is capable of both fully automated and interactive simulations in terminal area and enroute settings. It was developed on a PDP-10 timesharing system and operates under either the DEC 10/50 (1) or BBN TENEX (2) monitors. Among its basic facilities, the system provides for:

* definition of complicated ATC procedures and routes.

* real or fast time operation.

* realistic controller displays.

* inputs from an on-line terminal and/or a stored "scenario" file.

* instrumentation mechanisms to monitor the progress of a simulation, and record experimental data.

The ROSS system was created for the Transportation System Center (TSC) of the Department of Transportation to meet a need for a general purpose ATC simulator for use in experiments with manual and automated control techniques. Within the general framework of ATC research, an important goal for the ROSS system was flexibility. In addition to supporting experiments with both manual and automated ATC techniques, it had to model diverse ATC situations including terminal and enroute airspace structures, present and future aircraft types and traffic mixes, and various weather environments. Also, the system needed to include flexible mechanisms for generating display information during an experiment and for collecting and recording experimental data.

The goal of flexibility was met by providing a modular programming system with which each experimental user can build a separate simulation program, tailored to his own particular needs. In building a specific simulation program, the user selects from a collection of ROSS program modules that serve general functions, and creates other program modules that serve purposes specific to his experiment. These problem-specific modules govern the ATC techniques to be used, the structure of the airspace, the nature of the traffic and weather, and the formats for display presentation and data collection. In creating each of these problem-specific modules, the experimenter makes use of a special purpose language oriented toward the kind of information the module is to contain.

In the following section we provide some perspective for what follows by describing the functional organization of an operating ROSS simulator. We then return, in Section III, to the topic of system implementation. Section IV continues that discussion in more detail for the most important of the several user languages. Section V discusses the run time variability available in ROSS and Section VI briefly describe some ATC problems that have been simulated with ROSS.

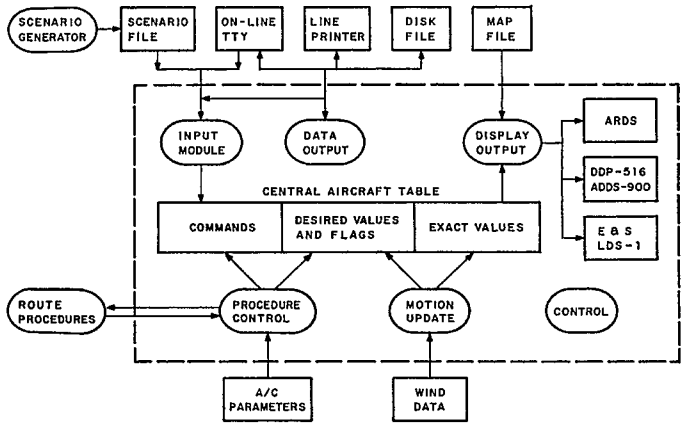

Figure 1
System Organization

II. System Organization

Figure 1 shows the relationships between the modules that make up an operating ROSS simulator. A small module called CONTROL has overall charge of the system's operation. It activates the other modules in an appropriate sequence and keeps track of timing. Timing is governed by an internal clocking mechanism which can be synchronized with the physical clock in the PDP-10 to yield a real-time simulation, or allowed to run free, yielding a fast-time simulation whose speed is limited only by the capacity of the timesharing computer.

At the heart of the simulator is a pair of processes, Motion Update and Procedure Control, which, taken together, model aircraft behavior. Motion Update "flies" the aircraft;

it is responsible for the simulation of aircraft kinematics including wind effects and aircraft performance characteristics. Procedure Control monitors and guides the aircraft at a higher level, using a series of programs which define the maneuvers and procedures to be followed, the physical form of the airspace, and the specific ATC control concepts used to control aircraft interactions. These programs are called route procedures. Taken together the route procedures for a given simulator form its most important problem-specific module.

The Central Aircraft Table contains all data known to the system about each aircraft including the aircraft's exact position, speed, heading, and altitude, the aircraft's desired conditions (for example, the altitude to which it is climbing), and the route procedure commands which are pending for that aircraft.

The input module interprets data from an on-line terminal, a scenario file of preplanned time-ordered inputs, or both. Except for an initial time field in scenario files, input lines from the two sources are identical; anything that can be typed in can be included in a pre-recorded scenario file. The allowable inputs from either source are calls on the route procedures and on the language primitives from which the route procedures are composed. Since the experimenter can define route procedures as he pleases, he has complete freedom in choosing what inputs his simulation will accept when it is running.

The system contains two kinds of output processes, either or both of which may be used in any given simulation. One output process generates a picture of the airspace, which includes a background map and a moving indicator for each aircraft, similar to the aircraft "tag" used in the ARTS-III display system (3). The background display map is encoded by the experimenter in a special language using the names of geography locations defined in the separate route procedure module. Different versions of the display output module are available for driving a Sanders ADDS-900 display, an E&S LDS-1 display, or an ARDS storage tube display.

The second output process records data collected during the course of a simulation. All output is kept as text, making it legible to the experimenter while still allowing for subsequent computer analysis. The basic output process can transmit simultaneously or separately to any of four output destinations:

* a disk file
* the controlling terminal
* the line printer
* the system input module

The output process can be "driven" by statements embedded in the route procedures or by a separate, user-prepared analysis program. Of these mechanism, the first generates output in response to simulation events, while the second generates regular, time-synchronized output.

III. System Implementation

The development of a specific simulation within the framework of ROSS is structured into four steps:

a) Definition of the experiment

b) Data translation

c) System loading

d) Simulation runs

Existing facilities of the PDP-10 timeshared environment are exploited at each step, resulting in a simulation programming system which was easy to implement.

For the first step, the experimenter uses the several special languages mentioned previously to prepare the problem-specific data for a particular experimental setup. This data consists of:

a) Geographical and procedural information describing the area and route structure of the experiment. (These are the Route Procedures mentioned above).

b) Performance parameters by type of the simulated aircraft.

c) Weather environment descriptions

d) Format descriptions for background display maps.

The experimenter prepares and stores this data in the computer system as a series of text files containing readable descriptive data, arranged in an orderly way. Any of the several text editing programs in the timesharing system can be employed in creating and modifying these files.

In the second step, these data files are translated into machine code by the standard PDP-10 assembler. Each of the special descriptive languages is actually a series of macro definitions that augment the assembler and determine how each kind of experiment data is to be translated. The experimenter's data files contain parameterized calls on these macros. On assembly, the macro calls are expanded, through the corresponding macro definitions, into a machine language form suitable for inclusion in the simulator. The result is a series of relocatable binary program modules which conform to ROSS system programming conventions, but which contain data specifically chosen for the simulation being prepared. It is important to note that the experimenter need not understand the macro mechanism, or even know that it exists. Exploiting the assembler's macro capabilities in this way provides a problem-specific language translator for each kind of experiment data. The ease with which these specialized language translators were constructed and altered as the languages were

designed and redesigned far offset the syntax restrictions imposed by the assembler's macro format conventions.

In step three, the relocatable binary modules from step two are link-loaded, using the PDP-10 loader program, along with other more permanent and experiment-independent modules which perform functions such as input/output, motion update, data storage, display generation, and timing control as described in the previous section. The result of the loading process is a runnable simulation program tailored to the specific ATC problem at hand and ready for experimental use (step four).

IV. The Route Procedure Mechanism

A key component in a ROSS simulation is the route procedure module. The route procedures it contains determine the individual shape and structure of any given experiment. They govern the geography and geometry of the airspace, the maneuvers to be followed by simulated aircraft, and the input language available at run time. They also have a central role in determining data output. In contrast, the other problem-specific modules such as wind or aircraft performance data play more peripheral roles, influencing the parameters of an experiment but not its structure.

Because of its role, the user language for creating route procedures has been called the route definition language (RDL). RDL provides constructs for expressing performance goals for an aircraft (i.e. desired speed, heading, altitude, or homing to a position) and for testing its progress toward these goals as a condition for further action. The passage of time can be handled explicitly (each simulated aircraft has a "stop watch" which can be cleared and checked) or implicity through tests which check a specific condition (i.e. higher than n thousand feet, farther than m miles from position p).

Primitive statements in RDL are either direct actions to be performed immediately or are conditional tests which proceed only when some predicate expression becomes true. Intermixing these actions and predicates provides a natural means of expressing ATC operational procedures.

Using the route definition language, it is possible to write procedural maneuver descriptions which do not imply an advance knowledge of an aircraft's trajectory. For example, a simple version of the DICE (DIrect Course Error) delaying technique can be defined as follows:

```
ATCDEF    DICE, $HDG, $POS, $ETA
ATCSTP    HDG, $HDG
ATCSTP    LATER, $POS, $ETA
ATCSTP    DIRECT, $POS
ATCSTP    NEARER, 1.0, $POS
ATCEND
```

The first line indicates that a new route procedure, DICE, is being defined with input parameters of initial heading ($HDG), final position ($POS), and desired arrival time at that position ($ETA). The first step in this procedure sets the aircraft desired heading to the parameter value supplied, causing the aircraft to turn to that heading. The next step, LATER, is a conditional test which remains false until the direct course arrival time at $POS is later than the time indicated ($ETA). As long as LATER is false, procedure execution is suspended and further steps are not considered. After the plane has moved sufficiently on its initial heading, LATER becomes true and execution will resume. The next step, DIRECT, causes the aircraft to home on the position, $POS. NEARER, another conditional test, blocks further action until the aircraft is within one mile of the position. When this occurs, the DICE procedure is complete and control returns to the point from which DICE was invoked.

In a programming sense, route procedures act like subroutines. They can be "called", and when complete, control returns to the calling location. Once defined, a route procedure can be used in the definition of a "higher-level" procedure. This cascading of procedure definitions can be carried through as many levels as desired.

Another useful property of route procedures is their ability to accept parameterized inputs. For example, the DICE maneuver above is defined independently of geography; it can be invoked at any physical location by supplying appropriate values for $HDG and $POS. In addition, the exact path the aircraft will follow and the geographic location where it will turn cannot be determined until the simulation is actually run.

Because route procedures can test conditions and wait for time to pass, the code they contain cannot be run to completion each time it is activated for an aircraft. Consequently, route procedures consist of re-entrant code, with variable data kept local to each aircraft. An operating simulator contains only a single set of route procedures, but many aircraft may be at different stages of route procedure execution at any one time.

As described previously, the route procedures defined by the experimenter together with the route definition primitives supply the input commands available at run time. Typically, inputs invoke procedures that create aircraft and set them in motion on preplanned maneuvers. However, commands can be issued to an aircraft already in the process of procedure execution. Inputs of this sort act as interrupts which must be completed before the ongoing procedures can be resumed. As an example, a human controller can "vector" a simulated aircraft off its flight plan and later return it to automatic control. Completely manual operation, involving direct vectoring of all aircraft, is possible simply by running a simulation containing no route procedures; the primitives of the route definition language are used for direct input of speed, heading, and altitude commands.

Action statements that output arbitrary text can be included in a route procedure. This text output can be directed back to the input module, making it possible to write procedures that generate and issue commands to simulated aircraft. Commands arriving from this source are indistinguishable from those originating at the console or scenario file.

Appendix I contains a simple but complete example of a route description for a standard published ATC procedure.

V. Run Time Variability

Although created with a specific set of route procedures, aircraft parameters, and environmental data, each ROSS simulator retains considerable run-to-run variability derived from the following sources.

a) The controlling inputs in a simulation are supplied either from a scenario file or the on-line terminal and can be varied from run to run, either randomly or in any other pattern selected by the user. In particular, the traffic to be injected into a simulated airspace from a scenario file can be generated by statistically controlled techniques.

b) In experiments involving interaction with controllers, the pattern and timing of on-line inputs will vary from run to run by virtue of the inherent variability in human style and performance.

c) Aircraft performance parameters are defined as gaussian distributions, with the mean and standard deviation of each distribution selected by the experimenter. During a run, actual performance values are drawn at random from these distributions. For example, in turning, an aircraft's turn rate will be selected from the distribution specified for that aircraft type.

d) RDL primitives are available for selecting "commanded" values of speed, altitude, and heading from random distributions. Again the experimenter can specify the means and standard deviations to be used. Another RDL primitive makes it possible to apply a gaussian function to any numerical value used in a simulation.

e) A simple, first order noise process is available for corrupting the position values "observed" by a simulated aircraft, or by ground control. As a result, all of the aircraft's maneuvers, and in particular its decision tests, can be affected by random errors.

VI. System Utilization

ROSS is currently in use in a number of ATC research projects, including preliminary testing of a flow-oriented incremental scheduling scheme and a series of display format evaluations for ATC automation.

A ROSS simulator was developed for the scheduling project to test the effectiveness of an incremental terminal area control scheme. The scheduling algorithm was incorporated into the simulator's route module using the standard RDL augmented with a few new primitive operations. Performance data was recorded, including landing intervals, schedule times assigned, and proximity conflicts, for simulation runs with different traffic loads and wind conditions.

The display evaluation project uses a ROSS simulator as a real-time target generator. It incorporates a simplified version of the metering and spacing system being developed by Computer Systems Engineering for the FAA. The main effort here has gone into more complex display function modules which provide a series of alternative display presentations for comparative evaluation by controllers.

While the basic ROSS system was developed for airborne simulations, it has been extended for ground-based research. We have defined the ground geometry of Boston's Logan International Airport, the performance parameters of aircraft moving on the ground, and a set of appropriate taxi, takeoff, and landing procedures. While further development in areas of dynamics and control are required, the resulting simulation operates reasonably and is an encouraging example of the system's versatility.

In the future we plan to improve the automated scheduling facilities already included in ROSS and to add capabilities for modeling and measuring the noise produced by aircraft. We also intend to use the system as a vehicle for exploring problems associated with multi-computer distributed computation systems utilizing the ARPA network. The basic ROSS system with its flexible and easily modified structure is an excellent foundation for these further studies and provides a base for the evolutionary development of more refined capabilities.

VII. Acknowledgements

The development of ROSS has been supported by the Transportation Systems Center of the Department of Transportation and by Bolt, Beranek, and Newman. The authors would like to thank John Sigona and Jean Roy of TSC and Tim Standish and Daniel Bobrow of BBN for their contributions in support and guidance of this effort.

References

1) "PDP-10 Reference Handbook", Digital Equipment Corporation, Maynard, Mass., 1971.

2) Bobrow, D. G., J. D. Burchfiel, D. L. Murphy, R. S. Tomlinson, "TENEX, A Paged Time Sharing System for the PDP-10", Third Symposium on Operating Systems Principles, October 18-20, 1971.

3) "System Design Handbook for the Automated Radar Traffic Control System (ARTS III)", Univac Corporation, St. Paul, Minn.

Appendix I

The exhibits that follow this appendix illustrate a simple but complete set of problem-specific modules that could be used to simulate the Weston One SID (Standard Instrument Departure) from Logan International Airport in Boston. A useful simulation would, in general, require more information including other approach and departure maneuvers, other aircraft types, more complete wind and map data, and data recording statements to record experimental results. Also, these exhibits illustrate only a small portion of the language forms and simulation capabilities available to a ROSS user.

Figure 2 shows Weston One as described and illustrated in a standard FAA publication. After takeoff from Logan, the aircraft is to fly out on the 266 radial from the Boston Vortac. In order to avoid intersecting traffic, the aircraft must hold an altitude of 4000 feet until crossing a point 17 miles from Boston. For aircraft not equipped with DME (distance measuring equipment), this point can also be defined as the intersection of Boston 266 with the Gardner 132 radial.

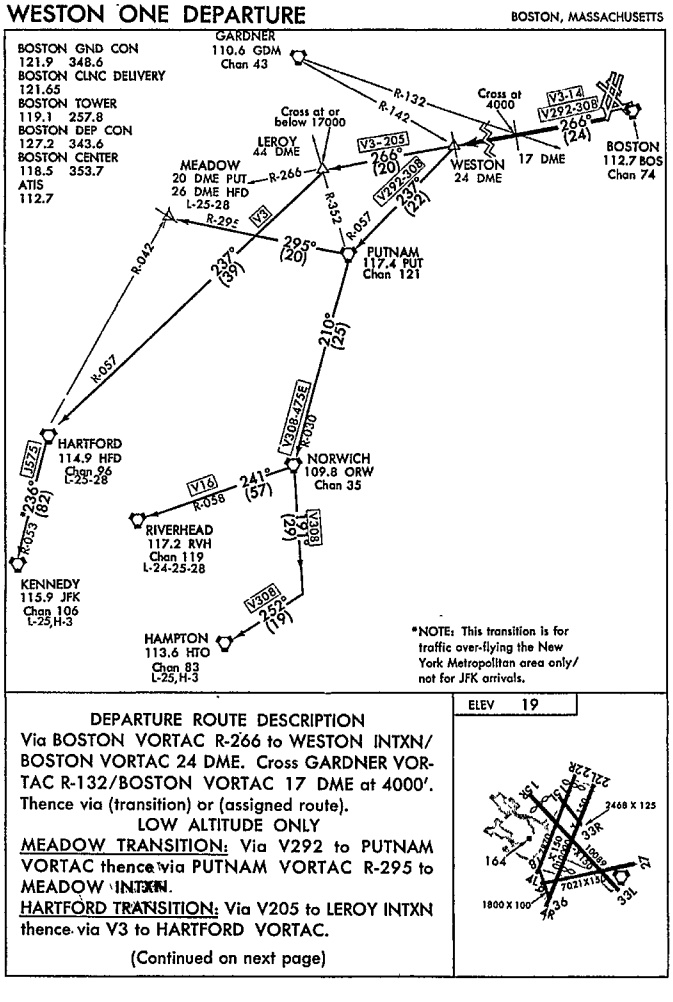

Figure 2

After this point, the aircraft climbs to its assigned altitude and continues on course until reaching Weston (defined as 24 miles from Boston or the intersection with the Gardner 142 radial). After this, the SID goes on to define alternative procedures for flying to more distant points, but for our purposes we will simply fly the aircraft to the Putnam Vortac, and then remove it from the simulation.

The Weston One SID assumes that the aircraft has already executed a takeoff and sufficient initial maneuvers to get onto the Boston 266 radial. For completeness, we shall include these as part of the simulation. Specifically, we shall cause the aircraft to take off on runway 33L and then fly southwest until it picks up the 266 radial.

Exhibit A defines the route procedures and geography information necessary to accomplish this simulation. As shown, each route procedure begins with an initial line (starting with the code ATCDEF) that names the procedure and the parameters it requires. Following this, a number of lines (each beginning with "ATCSTP") define the body of the procedure - the action that it contains. Each procedure step typically will contain a primitive action that establishes some desired value for the aircraft in question; a primitive predicate that blocks further action until a specified condition is true; or another route procedure, to be called, subroutine fashion. Each procedure terminates with a single line containing the code "ATCEND". As explained previously, each line of a route procedure is actually a macro call that will be "expanded" into appropriate machine language code when the procedure module is assembled. Also, as shown in Exhibit A, comments (beginning with semicolon) can be included freely for clarity.

The first two procedures serve general purpose functions. Procedures of this sort might be included in many different simulations. MAKEAC creates a new aircraft and gives it initial conditions of position, heading, speed, altitude, and aircraft type. The actual values to use will be supplied to MAKEAC at run time. The final line of MAKEAC (GO) sets the aircraft in motion. Procedure TO flies an aircraft from where it is to some named position. This procedure first causes the aircraft to home on the destination (DIRECT), and then calls into play a predicate (NEARER) that will block further action until the aircraft is close (within one mile in this case) to the destination.

The third route procedure, WESONE, defines the actual Weston One departure. WESONE takes parameters $ALT, $SPD, and $KIND, which allow final altitude, cruising speed, and aircraft type to be selected at run time. WESONE first calls MAKEAC to create an aircraft at Boston, with heading, altitude, and speed of 330 degrees, 100 feet, and 120 knots respectively. The aircraft type ($KIND) will be passed through WESONE into MAKEAC at run time. The net effect of this call on MAKEAC is to create an aircraft just after takeoff from runway 33L. After takeoff, the aircraft starts to accelerate to 240 knots and

begins a climb to 1000 feet. On reaching 1000 feet (ONALT means "on desired altitude") the aircraft turns left to a heading of 220 degrees and begins a further climb to 4000 feet. On reaching the Boston 266 radial (ONRAD, 266.0, BOSTON), the aircraft turns right and heads out on this radial. 17 miles out from Boston (DISTGT, 17.0, BOSTON), the aircraft is free to assume cruising altitude. 24 miles out, the aircraft has reached the Weston intersection. At this point it accelerates to cruising speed and flies to the Putnam Vortac. On reaching PUTNAM (as determined by the NEARER predicate embedded in TO) the aircraft is removed from the simulation by the action KILL.

In this particular procedure, distance checks were used to determine the 17- and 24-mile points from Boston. Referring to Figure 2, these checks could also have been based on the crossing of appropriate Gardner radials with ONRAD, 132.0, GARDNR and ONRAD, 142.0, GARDNR.

The remainder of Exhibit A defines geography points for use in the simulation and in establishing a background map. Each definition begins with the code "ATCPOS" followed by the name of the point and its x and y coordinates in nautical miles from an origin chosen by the experimenter. In this case, the origin was taken at Boston.

Exhibit B defines a map that could serve as background display for the Weston One simulation. The map consists of lines and labels. Lines are defined in terms of the geography points they connect. Labels are also defined in terms of geography points. In the case of labels, the geography point both establishes a nominal location on the display and supplies the label text.

The statement MAPSIZ establishes map scale as a distance in nautical miles from map center to perimeter. Both the map scale and origin can be changed at run time by commands entered at the terminal. Note that the size of the map need not correspond to the size of the airspace. In this case, for example, the Gardner Vortac is located well outside the map. Typically, the display can be viewed as a movable window into a much larger airspace.

Exhibit C shows part of a file that defines performance parameters for various types of aircraft. The name of each aircraft type is first defined by the statement KIND. Following this is a list of performance parameters, each stated in terms of a mean value and standard deviation (for error modeling purposes). As shown in the exhibit, the parameters govern rates of climb and descent, acceleration and deceleration, and turn; supply normal cruising and approach speeds; and determine the spacing rule for use in scheduling applications when aircraft are flying in trail.

Exhibit D defines wind data for a typical simulation. The simulator currently includes a "shear model" which divides the air into horizontal layers, each containing wind that differs somewhat in direction and intensity from adjoining layers. In the exhibit, the lines beginning with WLEVEL demarcate the boundaries between layers, while the lines beginning with WIND indicate the direction and intensity within each layer.

In order to simulate the Weston One SID, the modules defined in Exhibits A through D would be assembled, together with macro definition files to form relocatable binary program modules. These would then be link-loaded along with more permanent system modules to form a complete, runnable version of the simulator.

At run time, as described in the paper, the simulator accepts inputs from two sources: the computer terminal and/or a scenario file containing a preplanned series of input commands. Allowable inputs for either source consist of calls on the primitives of the route definition language (these are listed in Appendix II), or on the route procedures the experimenter has defined.

Exhibit E shows a fragment of a scenario file that could be used to generate a stream of departing aircraft along the Weston One SID. Each line begins with the simulated time at which the command is to be entered and executed, followed by an arbitrary flight label for the aircraft. The remaining information in this scenario invokes the WESONE route procedure and specifies the arguments to be used.

Exhibit F shows a series of commands that might be used to vector one of the aircraft temporarily off the WESONE departure. Assume that aircraft N7362 has turned onto the Boston 266 radial, but is still within 17 miles of Boston (as determined by observing the display). Then the first command line would take it off the 266 radial, head it south, and give it a speed of 150 knots. Later the second and third command lines could be used to turn the aircraft north, and put it back on the 266 radial. After this, the stored WESONE departure sequence would take over again.

Exhibit A

```
;ROUTE PROCEDURES FOR WESTON ONE DEPARTURE FROM BOSTON (LOGAN INTL)
;RUNWAY 33L ASSUMED IN USE

;CREATE A NEW AIRCRAFT
;VALUES FOR PARAMETERS ($POS,$HDG,ETC.) ARE SUPPLIED
;WHEN COMMAND USED AT RUN TIME

ATCDEF  MAKEAC,$POS,$HDG,$ALT,$SPD,$KIND        ;DEFINE NAME AND PARAMS
ATCSTP  NEWAC                   ;CREATE NEW A/C BLOCK IN CENTRAL TABLE
ATCSTP  JAMPOS,$POS             ;INSERT INITIAL VALUES FOR
ATCSTP  JAMHDG,$HDG             ;POSITION,SPEED,HEADING,ALTITUDE,KIND
ATCSTP  JAMSPD,$SPD
ATCSTP  JAMALT,$ALT
ATCSTP  JAMKND,$KIND
ATCSTP  GO                      ;START THE AIRCRAFT MOVING
ATCEND                          ;THIS TERMINATES EACH ROUTE PROCEDURE


;FLY TO A NAMED FIX

ATCDEF  TO,$POS
ATCSTP  DIRECT,$POS             ;HOME ON THE FIX
ATCSTP  NEARER,1.0,$POS         ;NO FURTHER ACTION UNTIL
ATCEND                          ;WITHIN ONE MILE OF FIX


;TAKEOFF AND FLY THE WESTON ONE DEPARTURE

ATCDEF  WESONE,$ALT,$SPD,$KIND
ATCSTP  MAKEAC,BOSTON,330.0,100.0,120.0,$KIND   ;TAKE OFF HEADING 330
ATCSTP  SPD,240.0               ;ACCELERATE TO 240 KNOTS
ATCSTP  ALT,1000.0              ;CLIMB TO 1000 FT
ATCSTP  ONALT                   ;WAIT UNTIL AT 1000 FT
ATCSTP  TLEFT,220.0             ;THEN TURN TO INTERCEPT RADIAL
ATCSTP  ALT,4000.0              ;AND START CLIMB TO 4000
ATCSTP  ONRAD,266.0,BOSTON      ;WAIT UNTIL ON RADIAL
ATCSTP  HDG,266.0               ;TURN ONTO RADIAL OUTBOUND
ATCSTP  DISGRT,17.0,BOSTON      ;HOLD 4000 FT UNTIL 17 MILES OUT
ATCSTP  ALT,$ALT                ;THEN CLIMB TO ASSIGNED ALTITUDE
ATCSTP  DISGRT,24.0,BOSTON      ;AT 24 MILES  OUT
ATCSTP  SPD,$SPD                ;ASSUME CRUISING SPEED
ATCSTP  TO,PUTNAM               ;& FLY TO PUTNAM VORTAC
ATCSTP  KILL                    ;AFTER PUTNAM,A/C NO LONGER OF INTEREST
ATCEND

;GEOGRAPHY POSITIONS FOR NAVIGATION AND DISPLAY OF MAP

ATCPOS  BOSTON,0.0,0.0          ;ORIGIN AT BOSTON FOR CONVENIENCE.
ATCPOS  WESTON,-22.8,4.6        ;VALUES ARE X & Y COORDINATES
ATCPOS  LEROY,-42.8,-8.0        ;IN NAUTICAL MILES FROM ORIGIN.
ATCPOS  GARDNR,-43.8,18.7
ATCPOS  PUTNAM,-40.7,-18.9

END
```

Exhibit B

```
;MAP FILE FOR WESTON ONE DEPARTURE

MAPSIZ  25.0                        ;DEFINE CENTER TO EDGE SCALE DISTANCE.

MAPLNS                              ;BEGINNING OF  LINE TABLE.
LINE    SOLID,BOSTON,WESTON         ;MAP LINES WILL BE DRAWN BETWEEN
LINE    SOLID,WESTON,LEROY          ;THE NAMED GEOGRAPHY POINTS.
LINE    SOLID,WESTON,PUTNAM         ;DOTTED LINES ARE ALSO ALLOWED.
ENTAB

MAPLBS                              ;BEGINNING OF LABEL TABLE.
LABEL   BOSTON,DR                   ;THE NAME ESTABLISHES BOTH LOCATION
LABEL   WESTON,D                    ;AND TEXT TO BE SHOWN.  THE CODE
LABEL   PUTNAM,R                    ;OFFSETS THE LABEL (UP, DOWN, RIGHT,
LABEL   LEROY,U                     ;DOWN AND RIGHT, ETC.) FROM ITS
ENTAB                               ;NOMINAL LOCATION.

END
```

Exhibit C

```
;**********SPECIFIC AIRCRAFT PARAMETERS**********

KIND    (PROPMT)          ;PROP MED TWIN

PARVAL  NORROD,-1000.0,110.0    ;NORMAL RATE OF DESCENT,FT/MIN
PARVAL  NORROC,1500.0,165.0     ;NORMAL RATE OF CLIMB,FT/MIN
PARVAL  NORACL,1.17,0.1         ;NORMAL ACCELERATION,KNOTS/SEC
PARVAL  NORDCL,-1.0,0.12        ;NORMAL DECELERATION,KNOTS/SEC
PARVAL  NORTRN,3.0,0.3          ;NORMAL RATE OF TURN;DEGREES/SEC
PARVAL  CRUSPD,180.0,15.0       ;CRUISE SPEED,KNOTS
PARVAL  APHSPD,125.0,3.7        ;APPROACH SPEED,KNOTS
PARVAL  SPRULE,3.0,0.0          ;SEPARATION SPACING DISTANCE, MILES

KIND    (CJETM)           ;MED COMM JET

PARVAL  NORROD,-1600.0,175.0
PARVAL  NORROC,2250.0,256.0
PARVAL  NORACL,1.0,0.1
PARVAL  NORDCL,-1.0,0.12
PARVAL  NORTRN,3.0,0.3
PARVAL  CRUSPD,330.0,27.0
PARVAL  APHSPD,140.0,4.2
PARVAL  SPRULE,3.0,0.0

KIND    (CJETH)           ;HVY COMM JET

PARVAL  NORROD,-1600.0,175.0
PARVAL  NORROC,1500.0,165.0
PARVAL  NORACL,1.08,0.1
PARVAL  NORDCL,-1.0,0.12
PARVAL  NORTRN,2.6,0.2
PARVAL  CRUSPD,330.0,27.0
PARVAL  APHSPD,150.0,4.5
PARVAL  SPRULE,5.0,0.0

END
```

Exhibit D

```
***** WIND DATA DEFINITION ***** WIND01 *****

GROUND                  ;UP TO 1500 FEET THE WIND IS
   WIND 243.0,14.7      ;FROM 243 DEG AT 14.7 KNOTS
WLEVEL 1500.0
   WIND 257.0,17.2      ;THIS WIND HOLDS FROM 1500 TO 3200 FEET
WLEVEL 3200.0
   WIND 268.4,19.3      ;AS ALTITUDE INCREASES, THE WIND STRENGTHENS
WLEVEL 4600.0           ;AND BECOMES MORE NORTHERLY
   WIND 281.0,23.6
WLEVEL 7200.0
   WIND 286.0,26.2
WLEVEL 11500.0
   WIND 293.6,31.8      ;THIS WIND FROM 11500 FT UP
SKY


END
```

Exhibit E

```
;SHORT SAMPLE OF A SCENARIO FILE FOR DRIVING THE WESTON ONE SIMULATION

00:00:53@  AA123:WESONE,17000,350,CJETM
00:01:49@  NE257:WESONE,19000,450,CJETH
00:03:01@  N7362:WESONE,12000,240,PROPMT
00:04:15@  NE356:WESONE,17000,390,CJETM
00:05:09@  N8943:WESONE,15000,270,PROPMT
```

Exhibit F

SAMPLE ON-LINE TYPED INPUT LINES

```
N7362:TLEFT,180;SPD,150
N7362:TRIGHT,0
N7362:ONRAD,266,BOSTON;TLEFT,266
```