DOT-TSC-FAA-71-1

EN ROUTE AIR TRAFFIC FLOW SIMULATION

Manuel F. Medeiros, Jr.
Computer Technology Division

January 1971

TECHNICAL REPORT

Prepared for

FEDERAL AVIATION ADMINISTRATION

TRANSPORTATION SYSTEMS CENTER
55 Broadway
Cambridge, Mass. 02142

En Route Air Traffic Flow Simulation

Prepared by

Manuel F. Medeiros, Jr.
Computer Technology Division
Transportation Systems Center
Department of Transportation
Cambridge, Mass. 02142

January 1971

TABLE OF CONTENTS

EN ROUTE AIR TRAFFIC FLOW SIMULATION

By MANUEL MEDEIROS

Transportation Systems Center

I. INTRODUCTION

This report covers the conception, design, development, and initial implementation of an advanced simulation technique applied to a study of national air traffic flow and its control by En Route Air Route Traffic Control Centers (ARTCC). The program was constructed at TSC under PPA FA17 (FY 70), and has also been submitted as a term project* by the author. It is intended to be the first step in gaining an insight into the nature of the national flow control problem and into the utility and limitations of digital simulation for that end.

A flexible digital computer implemented simulation has been developed which provides a family of model configurations and simulated environments for the U.S. air traffic system, restricted to positive controlled high altitude airspace. Exploitation, validation, and verification of this simulation model are just beginning. This report describes the purpose, design, development, and initial implementation of the simulation and presents future plans. Detailed information on the design and program structure is presented in the Appendices.

II. DESCRIPTION OF GENERAL PROBLEM

It has been well documented and publicized that the air traffic has been increasing at greater than expected rates. In 1968, peak airborne aircraft at one instance of time reached 12,800. By 1980, 22,200 peak airborne are projected and by 1995 54,400 are possible to be in the air at peak times.[1] Benefits

---

*Term Project for MIT Graduate Course on Systems Simulation.

from centralized air traffic control monitoring during peak traffic periods and abnormal airspace or terminal restrictions have been demonstrated by the new Central Flow Control Facility at Washington, D.C. This facility manually poles the nation's 20 En Route ARTCC's hourly. When abnormal flight delays or equipment outages are reported, the facility initiates remedial action. As the future air traffic increases, one sees a greater need for centralized monitoring and flow control, coordinating the smooth operation of the 20 En Route ARTCC's, minimizing delays, and providing desired service and safety. To determine specific future facility automation and procedural requirements for centralized flow management, a deeper understanding and insight into the total air traffic flow system is needed. An in-depth understanding of the total system is extremely difficult, if not impossible, for human inductive reasoning alone to achieve, because of the autonomous nature of each ARTCC's operation, the localized weather effects, the terminal acceptance and departure rates, the localized controller work loads, and the inter and intra Center (ARTCC) route, airspace, and flight restrictions.

The following questions may be asked: Can digital computer simulation techniques aid in understanding this flow system? Can simulation resolve those factors which significantly affect the air traffic flow from those factors which just add "noise" to the system? Can the air traffic system be successfully modeled? Can the model be verified to a useful level of confidence? What part can simulation play in determining the requirements for a future automated Central Flow Control Facility?

This program was constructed to explore the possibility that a digital computer simulation could provide considerable insight into a future national air traffic system having a centralized flow management operation, coordinating the operations of the ARTCC's.

III. PREVIOUS RELATED WORK

There is only a limited amount of related literature in the area of air traffic flow modeling. Back in December 1963, Arad,[4] then working for the FAA, reported on the complexity of total air traffic in controlled airspace as it relates to the controller work loading. Arad developed an analytical model, employing a weighted series of terms to describe the control position workload. Model validation was planned but not reported. This approach to control position workload determination could be useful in computing ARTCC capacity levels in an en route flow model. In fact, it has been recently learned that work at FAA has been initiated which uses Arad's model along with actual flight schedules in modeling a multi-ARTCC traffic flow situation. Additional information concerning this new work is being sought.

The Mitre Corp. has developed a terminal reservation system for 5 high density airports. The resulting reservation lists are used by a flow control model to meter the traffic into the N.Y. metroplex terminal area, based on hourly acceptance rates and current numbers of aircraft in hold patterns, waiting to land. This flow control model is not designed to consider en route loads but advises aircraft which hold reservations for a N.Y. landing (up to six hours later), of expected delays at the destination or of cancellation. This system, in effect, meters the traffic nation wide with regard to their landing times at the N.Y. terminal. This work and Arad's are the two closest efforts (documented) in the area.

IV. SIMULATION APPROACH RATIONAL

The starting point for a simulation is to model today's air traffic system. It is most important, if high confidence in a simulated model is to be developed, that the model be

verified with the actual system. This is only possible if one is modeling the current or past system, (assuming that adequate comparative data can be obtained on the actual air traffic operations). If a successful model of today's air traffic system is developed, it is reasonable to expect that this model can be upgraded to future increased traffic loads, making provisions for automated communication functions, controller workload changes due to automated supporting function, and anticipated procedural changes. While upgrading the model one can develop a better understanding of the future system performance and through experimental design iterations (of the simulation model), it is possible to evaluate the system sensitive factors and the requirements for automating centralized flow management operations.

There are two principle approaches to simulating the air traffic flow system. One is Arad's approach, analytical modeling, the other is discrete event modeling, (a third approach is possible by combining the two). One could write a book advocating either approach for numerous different reasons. The discrete event simulation was chosen because this approach provides the most flexibility; it is easier to visualize, and it can be modularly designed. The work was to be performed in conjunction with a graduate course term project with access to a new interactive simulation language currently being developed at MIT. This language, known as SIMPL, is a superset of PL/1 (i.e., it will also accept straight PL/1 program code). The language currently is implemented only on the MULTICS time sharing computer system (Project MAC) at MIT.

Because of the on-line real time computer (MULTICS), whole families of air traffic flow models could be simulated if a master model (basic framework) were designed with interactive dynamic model structuring. In other words, it should be possible

to model sectors, Centers, national flow system, or any combinations thereof through an interactive, man/computer, structuring of the models.

There are a number of other features considered and designed into the model and simulation approach. The model handles data internally in a manner which is conducive to graphical display of maps (weather, centers, and terminals) and the traffic flow. A micro level of simulation was selected with individual aircraft being separately identifiable, as well as the controlling centers and sectors. Some effects of weather would be included in the simulated environment. A final consideration is that no role playing would be included in the simulation (i.e., a human being acting for a modeled element). This is necessary in order to permit a fast time simulation within realistic run times and available facilities.

V. DESCRIPTION OF SYSTEM TO BE MODELED

The nation wide air traffic system to be modeled is composed of the following:

(1) High altitude positive control airspace - this is the zone where 100% of the aircraft are IFR and must be cleared and controlled by the ARTCC's.

(2) The 20 ARTCC's across the nation - these centers are further subdivided into sectors.

(3) Aircraft - terminals (airports - airspace - and communication facilities.

(4) Traffic restrictions due to: weather, overloads, center to center rate specifications, terminal acceptance/departure rates, control procedures, standards and operating practices.

(5) Principle objectives in handling traffic flow: safety, minimize delays and convenience.

(6) Flight loads dependent on flight schedules, route selections, and non-controllable factors (weather, terminal rates saturated, route and center saturation).

VI. DESCRIPTION OF DEVELOPED SIMULATION MODEL

Because of the complexity of the final model's structure and the many options for initializing and altering the model during run time it is best to describe a conceptualized picture of the modeled elements. The model has an internal picture of the airspace being modeled. It also maintains the pictures of the weather (at any point in time) and the terminals and has provisions for pictures of route structure and winds aloft by altitude. The pictures are resolved into grids of 100 lines along the east-west direction and 50 lines along the north-south direction. In the picture of the airspace, the centers and/or sectors have been identified by number codes. When these pictures are superimposed, a total view of the current ground and airborne environment is presented - minus the aircraft. Each aircraft can be considered a little model airplane, originating at a terminal, flying through the grid and landing at a destination terminal. The aircraft's movement through the airspace is controlled by clearance from the local Center or sector. The Center considers weather en route, other Center restrictions, and weather at terminals, in clearing aircraft. The Centers command a flight speed, vector, and next fix point for each aircraft with each clearance. The aircraft then fly the vector to the next fix and call the Center (which jurisdiction includes the new aircraft position) for the next leg clearance. The controllers can hold aircraft when necessary, can reroute around thunderstorms, can send aircraft to nearest alternate terminals when original destination is weathered in beyond its capability and the fuel reserves are less than

expected landing delays. Communication times between aircraft and Centers are currently used in determining loads on the Centers and numbers of communication channels available at each Center are used to modulate these communication loads. This selection of communications as a first measure of Center loading is based on the belief that communications take the major percentage of the current controllers time. In a later version of the model it is planned to adjust the communication time durations as a function of the controller's work load, complexity of his traffic picture, and weather in his zone, thus feeding this variability into the total flow picture. One can now visualize the aircraft at various stages of their flight, dotting across the picture of airspace, center/sector boundaries, weather, and terminals. Based on run time reports typed on the computer console, one sees a peak loading condition develop at Center #2 (hypothetical example). The simulation operator can interrupt the simulation run, set the pictures, aircraft, and all necessary details back to a previously noted time. He then can adjust the controller procedures, operating ranges or even geographical boundaries of the Centers/sectors and can restart the simulation. At the same point in time as the observed peak loading occurred, he can interrupt the simulation and compare the new report. If the alteration did not produce the desired results he can repeat the setback, adjust and restart operations. This in effect is the basic functional picture of the developed simulation. This iterative capability should be most useful in tuning up the model to best represent the actual traffic flow and in developing an understanding of the important-most sensitive aspects of the total air traffic flow and control system. One more feature of the simulation should be noted in this descriptive section. Because their geographical shapes probably play a significant role in sector and Center loading, the model has been designed with

complete geographical freedom. To illustrate this point, a twenty-center national system, a three-center zonal flow system, a single Center-sectorized system or any such collection of controlling region divisions can all be modeled with no reprogramming. These geographical patterns and their related procedural pattern can be created readily at the start of a new simulation run. Once a desired configuration is created it can be saved and used as a starting point as often as desired.

The detailed program description is too long and complex to contain in this text, so it is assembled in the Appendices for those interested in the programming aspects. In addition, since the operation of this program involves considerable user interface, (with question and answer sessions dependent on previous answers), a "Users Guide" is required. This guide is currently being prepared; but sample inputs, of the question/answer type, and sample outputs are annotated in the Appendices. The outputs are of several types, the aircraft tracing by the teletype, the Teletype ARTCC PROGRESS REPORT, the interrupted "frozen" status of the airborne aircraft, and a sample piece of the Centers Map, (the airspace picture).

VII. SIMULATION MODEL DESIGN TRADEOFFS

A principle concern in digital simulation is the tradeoff between simulated run time and actual run time. If the modeler gets sidetracked in minute model preciseness, programming numerous subelements and numerous computational steps into the model, the actual computer run time for an air traffic simulation could take longer than the real world flight times. For the purposes which this model is intended, it is unrealistic to expect the users to take advantage of the run time editing and restart features if turn around times are greater than the real world flight times. As a consequence, a scale factor has been built into the simulation model. This scale factor

is applied to the speed values of the aircraft, in effect shrinking the size of the simulated airspace when speed is increased. Through a combination of scale factor, adjustments in frequency of calls to the Centers and utilization of smaller portions of the internal airspace picture (i.e., a reduced picture size, not a truncation of the airspace model), it should be possible to tailor the simulation run time to the users desire, (decreasing the computation resolution while speeding up the computer run time* for equivalent simulated time increments). In effect, the user is given a broad range of micro levels to model within. Questions like "Is it necessary to divide a Center into 100 fix points (grid squares) to get representative behavior or can a 10 points model give reasonable results?", can also be explored using the above mentioned adjustments.

A second major concern in the design tradeoffs is the balance between straight forward programming and programming modularity. Programming is like giving instructions. It is easiest to instruct for the specific job to be done. But, if other jobs of a similar scope are anticipated, it is well worth the additional time to instruct for performing the family of jobs. This extra scope of programming has been employed by dividing the simulation model into subtasks and then chaining together the subtasks to perform the computation/simulated task. This is like teaching a person several separate duties and then naming those in the order desired to be performed. When one has different jobs to be done (within the framework of the taught duties) a simple reordered list can produce the job results. One manifestation of this in the model is the operational

---

*It must be noted that computer run times for this model are dependent on the computer size, speed, and core; whether time shared, what percentage of the cpu time is available if time shared, and how much tracing (output) is activated.

attributes of the Centers (or Sectors). By naming the procedures for takeoff, landing and en route clearances and assigning them to a given Center, that Center will exhibit those functional results. If one wants to add a new procedure to the model, it is not necessary to reprogram the entire simulation. Just program the procedure, give it a name (actually a number) and add it to the simulation, thereafter references to the new procedure implements it in the Centers operational attributes. This modularity feature is expected to facilitate tuning up the model's resemblence to the real world, by permitting quick and easy changes to the heart of the model - the Center/Sector procedures.

VIII. CURRENT SIMULATION STATUS

The En Route Air Traffic Flow Simulation Model is completely programmed, successfully compiled, and fully implemented on the MIT MULTICS Time Sharing Computer system. Most of the features designed into the model have been tested for run time "bugs". Current effort is being focused on some minor errors in the console report generation routines. The evaluation of aircraft flow has begun using a multi-center control network and observing the models start-up characteristics. The built-in aircraft trace feature is being used to track the aircraft through the system of Centers, along with the information of the total system status, written on the disc files when the simulation is interrupted. The console-progress report is providing some data even though it is faulty. Samples of the simulations inputs and outputs are shown in the Appendices.

IX. PLANNED WORK

During the month of January the simulation should be substantially ridden of bugs and run through a broad spectrum

of air traffic flow problems. At the end of January an evaluation will be made of the simulation's performance and the intuitive flow patterns of air traffic under similar conditions. A decision must then be made whether to convert the model to another machine because the MULTICS System will not be available beyond January 31. If the simulation proves useful as a tool to gain understanding of the national air traffic flow phenomena, the recommended decision is that the model and simulation technique be converted and implemented on the Graphics Time Sharing Computer System of the Computer Technology Division of the Transportation Systems Center.

The simulation has been designed to be readily compatible with graphical techniques to display the airspace, weather, center and terminal maps, and the aircraft transiting this airspace. It is felt that graphical input and output techniques can greatly enhance the information gained from viewing the traffic patterns, overloads, and aircraft route variations as they are developing. The graphical input devices can facilitate inputs such as center boundary changes, route changes, and weather changes.*

In order to develop a satisfactory model of today's air traffic system, the program should be expanded to include the following:

- The incorporation of all high altitude en route way-points on the map provided for them.
- Aircraft departures according to prestored airline schedules, plus random delays.
- Storage of planned flight routes (as a series of way-points and ETA's at the way-points).

---

*The current weather simulation takes the weather conditions present on the weather map and uniformly moves the weather over the country (airspace-map) at a preset rate. There is no weather predictive capability.

- High altitude wind patterns.
- Improved movement, size and intensity of weather patterns.
- Input of historical weather data (movement, patterns and type).
- Interactive input of aircraft emergencies.
- Random variation in aircraft speed.
- Interactive input of changes in terminal operations capacity.

As discussed earlier, once a satisfactory model of today's air traffic system has been developed, it should be possible to increase traffic loads, simulate automatic communication of flight data, simulate the controller work loads and his increased capacity (an assumption at this time) due to automation of some of his functions and tasks and then study the resultant flow patterns for problems, sensitivity and saturation conditions. It then should be possible, assuming existence of adequate communications between ARTCC computers and a Central Flow Control Facility, to test out control algorithms and procedures representative of an automated Central Flow Management System. The simulated Flow Management System can be fed different quantities, qualities and timeliness of aircraft positional or planned positional data, and various measures of center and sector loads which are to be used in the Flow Control decisions. This approach may shed considerable light on the automation and other requirements for a Central Flow Management Facility under future traffic environments.

X. REFERENCES

(1) Report of D.O.T. Air Traffic Control Advisory Committee, Vol. 1, Dec. 1969, U.S. Government Printing Office.

(2) Ten Year Plan, The National Aviation System Plan 1971-1980, March 1970, D.O.T. Federal Aviation Admin.

(3) En Route Air Traffic Control, Manual, January 1, 1970, D.O.T. Federal Aviation Admin., 7110.9A.

(4) Control Load, Control Capacity and Optical Sector Design, Bar-Atid Arad, et al, Dec. 1963, D.O.T. FAA, AD429 906.

(5) R&D Plan to Increase Airport and Airways System Capacity, May 1970, D.O.T., FAA, AD707 186.

(6) Verification of Computer Simulation Models, Oct. 1967, (Management Science) by Thomas H. Naylor and J. M. Finger.

(7) Systems Simulation Methodology and Digital Simulation Languages, Malcolm M. Jones, Prof. MIT.

(8) The Process Concept as a basis for Simulation Modeling, G. Blunden and H. Krasnow, 28th National Meeting ORS, Nov. 1965.

(9) A New Methodology for Computer Simulation, M. Greenberger, Oct. 1964, Project MAC, MIT.

Appendix A

Sample Inputs and Outputs

APPENDIX A

SAMPLE INPUTS AND OUTPUTS

I. MODEL CONFIGURATION RESTRUCTURING

When execution of the simulation begins, a series of questions is asked and, depending on the answers, other groups of questions will be asked or skipped over. The following illustrates one sequence of questions and the responses. The answers have been annotated to clerify their purpose. "I" indicates operator.

A. Center Mapping.- In this model, the Center can be sectorized by assigning unique numbers to the sectors and then considering the sum of the sectors as the Center.

'n'ew or 'o'ld data base?
n ←——— *I have chosen to start a new model configuration.*
Map input-Center number?
1 ←——— *#1 will be my first ARTCC Center.*
Give blx,brx,bly,tly of a rectangular portion
50 100 1 50 ←——— *I specify the bottom left x-coordinate, the bottom right x, etc. - thus claiming an area under Center #1 (Picture a blank grid 100 by 50 which is being filled with the Center I.D. numbers in the claimed areas).*
Map input-Center number?
2 ←——— *#2 Center*
Give blx,brx,bly,tly of a rectangular portion
1 49 1 50 ←——— *Its area of jurisdiction*
Map input-Center number?
3 ←——— *#3 Center*

Give blx,brx,bly,tly of a rectangular portion
40 60 20 30
Map input-Center number?
4 ← #4 Center
Give blx,brx,bly,tly of a rectangular portion
80 95 35 48
Map input-Center number?
4 ← #4 Center
Give blx,brx,bly,tly of a rectangular portion
75 87 30 40

Note! The composite area becomes Center #4. It is thus possible to shape any Center configuration down to the single sq. area unit.

Map input-Center number?
5 ← *My last area for this Center Map*
Give blx,brx,bly,tly of a rectangular portion
60 75 25 40
Map input-Center number?
0 ← *A zero answer to any of the questions terminates that block of questions*

B. Weather Mapping.

Note: Severity weather codes indicate:

1 = Rain (usually delays arrivals and departures to a small degree, 10 minutes)

2 = Fog (delays arrivals/departures at terminals, 30 minutes)

3 = Thunder Storms (delays arrivals/departures at terminals, 1 1/2 hours, and all airborne flights will be routed around them)

4 = Snow and Ice (essentially closes the terminals for 5 hours)

Weather input-Give severity(1,2,3 or 4)
3 ← *I chose to develop thunderstorms*
Give blx,brx,bly,tly portion of rectangular shape
65 65 27 32 ← *In a line running north/south 6 squares tall and 1 wide*
Weather input-Give severity(1,2,3 or 4)
2 ← *I want fog at a*

Give blx,brx,bly,tly portion of rectangular shape
45 45 23 23 ← *Terminal located at coordinates 45, 23*
The current model will move this weather one sg in a n.e. direction every simulated hour. (For normal scale this is 26 mph frontal speed)

Weather input-Give severity(1,2,3 or 4)
0 ← *Zero terminates block*

C. Terminal Mapping.

Give new terminal location x y
22 42 ← *Coordinates of a terminal*
Give new terminal location x y
85 37 ← *Another terminal*
Give new terminal location x y
73 36 ← *Etc.*
Give new terminal location x y
55 40
Give new terminal location x y
50 24
Give new terminal location x y
68 30
Give new terminal location x y
45 23
Give new terminal location x y
0 0 ← *Zeros terminate this block*

D. Center-to-Center Rate Restrictions.

Handover restriction input-Give center imposing restriction
4 ← *Center #4 will impose a*
Give center receiving restriction and the min time (sec) between aircraft(mit)
5 200 ← *200 Second (3.3 minute - or 33 miles in trail) restriction on Center #5*
Handover restriction input-Give center imposing restriction
0 ← *Terminate block*

E. Center (or Sector) Procedural Specifications.- The choice in "operation number" refers to a program module currently in the simulation. These modules can be expanded by adding new modules which function to control the aircraft in the modules own way.

(See paragraph VII of text.) With each modular operation it may be desired to set controlling parameters and, therefore, a second question requesting these parameters is possible.

There is an automatic (default) procedural specification list given any Center or Sector not specifically defined in this question block.

It is noted that the user must know what each numbered procedure does functionally.

```
Procedure Input-Give center number for new operating structure
3<------Set procedures for Center #3
Give takeoff operation number(1 or 2)
2<------I chose the terminal operation which is weather
        dependent
Give ils capability(0,1,2,3 or5),call in time and delay time to
next call(sec)
3 21 21<--I have selected the full ils facilities, call-in time
        is comm time, etc.
Give landing operation number(only 3 currently)
3<------Only one landing procedure is available (it is very
        sophisticated, rerouting to alternate airports within
        fuel levels)
Give ils capability (0,1,2,3 or 5) and call in time(sec)
3 23
Give enroute operation number(only 4 currently)
4<------Again - one selection
Give call in time and delay to next call(sec)
24 90<----Comm time and next clearance request time
Procedure Input-Give center number for new operating structure
0<------Terminate block
```

A current weakness is that the terminal operations are applied uniformly through the Centers Area and are not assignable to specific airports - (A model change is not difficult)

F. Major Model Parameters.- Nominal values are assigned these parameters unless they are edited as shown below:

```
Do you want to change remaining parameters?'y' or 'n'
y<------- Yes let's edit
Edit which?'s'cale,'a'vailable center communication channels
or 't'ime base
s<-------The scale factor
Give new scale factor.
1.0e-4<---and set it to 0.0001
Do you want to change remaining parameters?'y' or 'n'
y<-------I would like to edit something else
Edit which?'s'cale,'a'vailable center communication channels
or 't'ime base
t<-------Specifically the time base (time starts with zero hours
     unless otherwise set)
Give time base in hours
6.0001<---Starting time will be named 6 o'clock
Do you want to change remaining parameters?'y' or 'n'
y<-------More editing wanted
Edit which?'s'cale,'a'vailable center communication channels
or 't'ime base
a<-------Lets set the number of communication channels
Give center number and number of communication channels
5 5<------For Center #5 to five available at one time
Give center number and number of communication channels
0 0<------No more channel editing
Do you want to change remaining parameters?'y' or 'n'
n<-------No more parameter changes
```

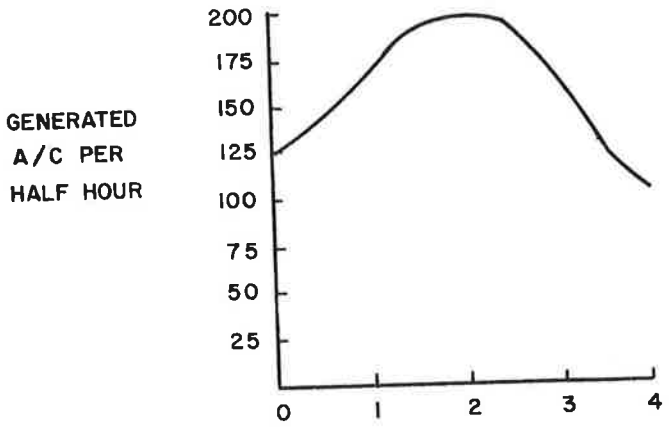
G. Another chance.- To correct a previous typing error or input a change of mind about either the maps or parameters.

```
Do you want to change maps or parameters?'y' or 'n'
n<------ I'm through editing
```

H. Traffic Generator.

```
Do you want to change the traffic generator?'y' 'n'
n<-------No change
```

The built-in generator has a four-hour cycle, and produces a nationwide traffic load with a distribution like:



Traffic loads can be set for 4, 8, 12, 16, 20 and 24-hour cycles and numbers of aircraft per half hour are set for the select cycle time.

Aircraft are then randomly spread within each half hour to total the desired loading.

An alternate means can be used such as following exact flight schedule read from a data file, (not currently implemented).

I. Set Run Time Interrupt.

Set interrupt in '?' sec. '0' means no interuptions
.001 ← *I want an immediate interrupt to save my new data base so that I can restart with this model configuration another time. (The interrupt writes out the entire data base and controlling information on the disc files).*

Note! When the run is restarted you are asked again to set the next interrupt

J. Set Time For Next Console ARTCC PROGRESS REPORT.

Console report in '?' sec. '0' means none.
750 ← *I want the report in 12.5 minutes of simulated time.*

K. Aircraft Data Gathering.- A sampling or all of the aircraft's (each) movements, clearances, delays and time of takeoff, etc. can be either recorded on the disc files or printed on the console. (The answer 100 means every 100th aircraft.)

Record every '?' and console trace every '?' aircraft?
0 100 ← *No disc files required but I want to follow the step by step progress of every 100th aircraft starting with #1.*

The run now begins - ending the editing sessions.

II. RESTARTING THE SIMULATION

When one restarts the simulation he is asked the same question as I.a:

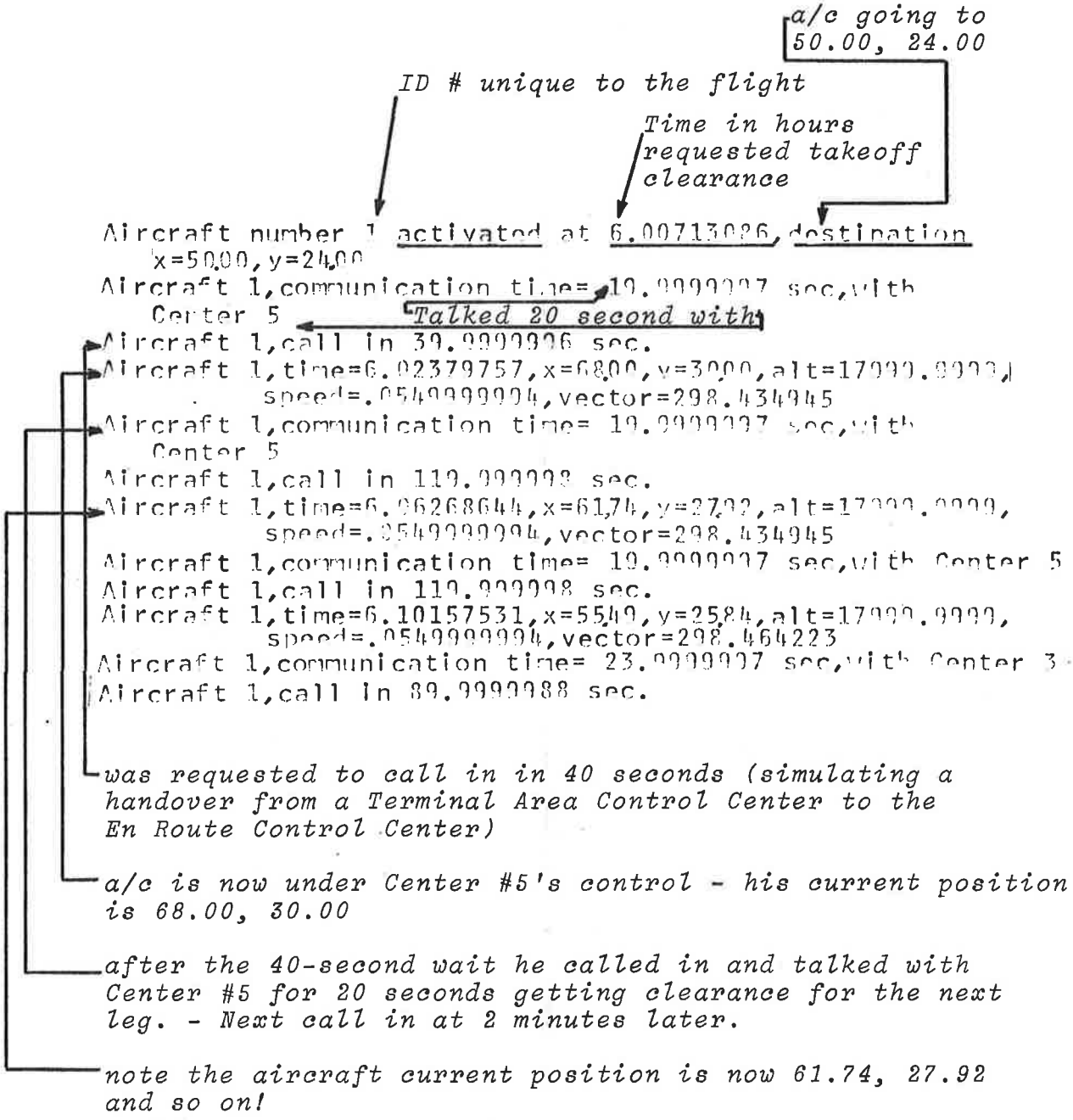
'n'ew or 'o'ld data base?
o ← *Old data base to be used*

The appropriate files are then read in and the questions from I.G. on are asked.

III. OUTPUTS OF SIMULATION

A. On Console.

(1) Aircraft Trace.



There are other trace messages when restrictions and delays are encountered - explaining them.

The file record output trace is exactly the same as the console but printing is done off line.

(2) ARTCC PROGRESS REPORTS.- (Note: Not fully debugged.) (This report printed early in run.)

ARTCC PROGRESS REPORT
Time= 756.000089 ← *Time in sec. - should be in hours*
Delays: ← *Highlight total sum delays*

(round off) should read 6.0

Enroute: 5.99999994 aircraft delayed 50.4390996 sec. avg.
Takeoff: 0. aircraft delayed 0. sec. avg.
Landing: 0. aircraft delayed 0. sec. avg.
Avg number of aircraft originating in centers= 09.9999998,
terminating at centers= 0 ← *Average for 4 Centers*
entering center= 7.74999994, and leaving center= 0.
Many aircraft left Centers
Avg no. of aircraf3 vectored to different destinations= 0.
No rerouting was required
Avg no. of aircraft rerouted because of weather= .750000000
3 aircraft rerouted (avg. for 4 Centers)

Peak values ← *Aids in picking Center reports to view.*

descripti	center no.	peak value	
com sesions	2.99999996	83.9999985	← *communication sessions*
a/c origins	5.00000000	14.9999998	← *aircraft originating in Center's area*
a/c termins	0.	0.	← *a/c landing in Center's area*
a/c exits	0.99999998	13.9999997	← *a/c exiting a Center's control*
a/c entered	0.99999998	13.9999997	← *a/c entering a Center's control*
a/c del/TOFF	0.	0.	← *takeoff delays*
a/c del landg	0.	0.	← *landing delays*
a/c destin chg	0.	0.	← *destination changes*
a/c weath-rerut	5.00000000	2.99999996	← *weather rerouting*
a/c ctr restr	3.99999996	5.99999994	← *a/c held because of Center-to-Center restrictions*

```
avg Toff dely  0.      0.  <-- average takeoff delays
avg landg del  0.      0.  <-- average landing delays
avg enr delay  3.99999996    201.756399 <-- average en route delays
Give the center number you wish to see progress(?>21 gives all,
   =0 concludes list)
23 <-- Give me the reports for all Centers.

Center  1  Progress report
 description        value
com sesions      34.9999997
a/c origins      13.9999997
a/c termins      0.
a/c exits     13.9999997
a/c entered      13.9999997
a/c del/TOFF      0.
a/c del landg      0.
a/c destin chg      0.
a/c weath-rerut      0.
a/c ctr restr      0.
avg Toff dely      0.
avg landg del      0.
avg enr delay      0.
```

Report for Center #1

```
Center  2  Progress report
 description        value
com sesions      0.
a/c origins      0.
a/c termins      0.
a/c exits     0.
a/c entered      0.
a/c del/TOFF      0.
a/c del landg      0.
a/c destin chg      0.
a/c weath-rerut      0.
a/c ctr restr      0.
avg Toff dely      0.
avg landg del      0.
avg enr delay      0.
```

Report for Center #2 (This Center had no terminals and the transcient aircraft hadn't arrived in this early start up report.)

etc., for all Centers

```
Do you want to 'e'dit model,set an earlier 'i'nterupt,
   or 'c'ontinue?
```

You are given the opportunity to restructure the system

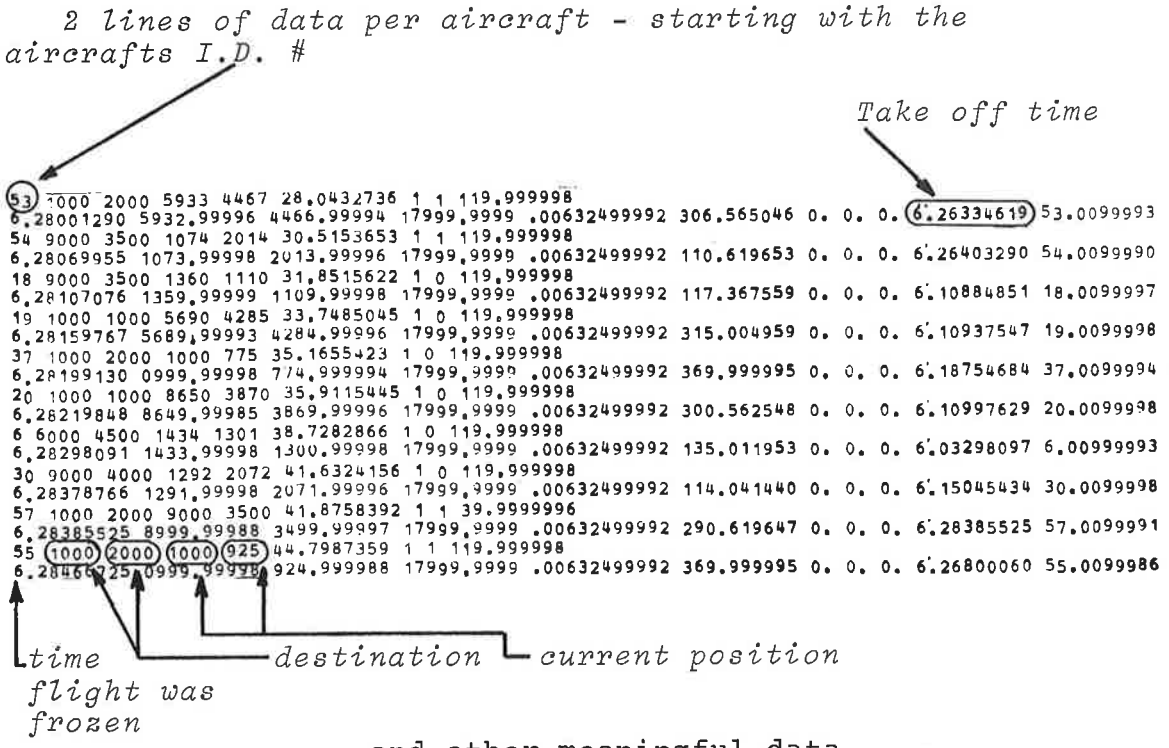
B. On the Disc Files.

(1) Aircraft Trace.- Same output information as III.A.1.

(2) Aircraft Progress Report.- When the aircraft finishes its flight a summary of it is recorded showing the aircrafts I.D., the various delays encountered the times of takeoff and landing and the origin and destination points.

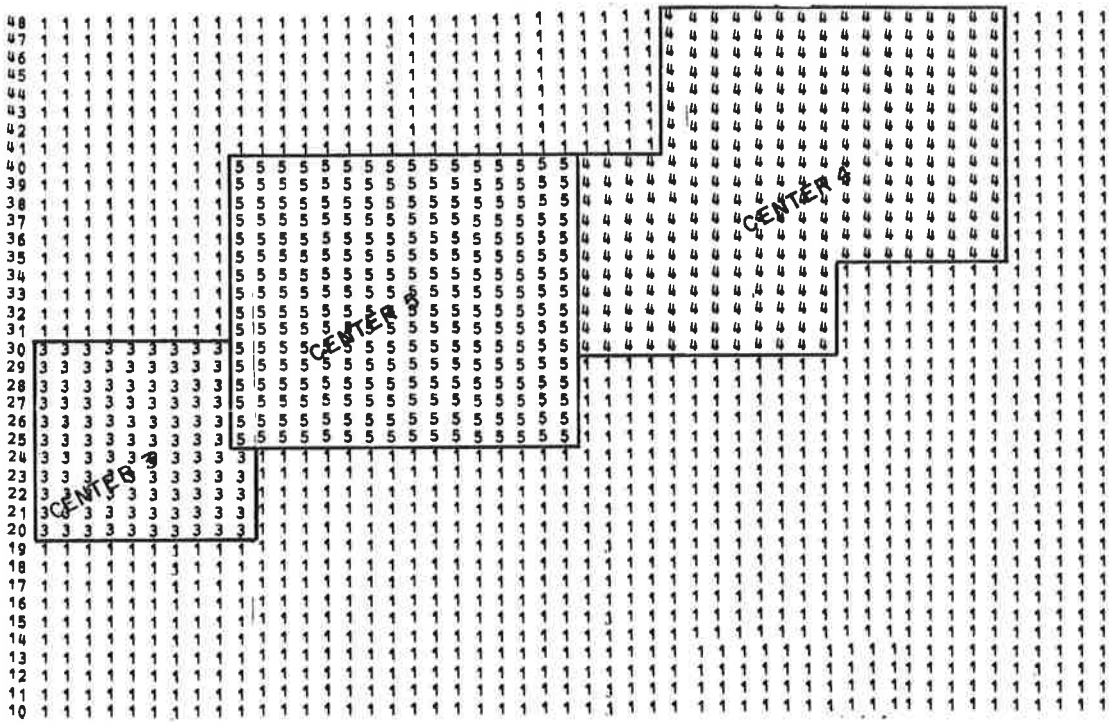
(This output file was not activated as yet.)

(3) Interrupted Aircraft Airborne Status File.- When the simulation is interrupted and all aircraft motion is frozen this file is written with all the information required to reactivate the aircraft. A sample is shown below: (no descriptive information is stored with this data.)



.... and other meaningful data ....

(4) Interrupted Model Structure and Control Parameters.- This file is thousands of words long! Representative of the data contained therein besides typical numbers is the map of the Center's Airspace. A sample of the map is shown below (internal storage is in this map format and so it can thus be visualized directly from the printed out file). Each number covers a square area dependent on the map scale, (i.e., for the U.S. the area is 26 x 26 miles).



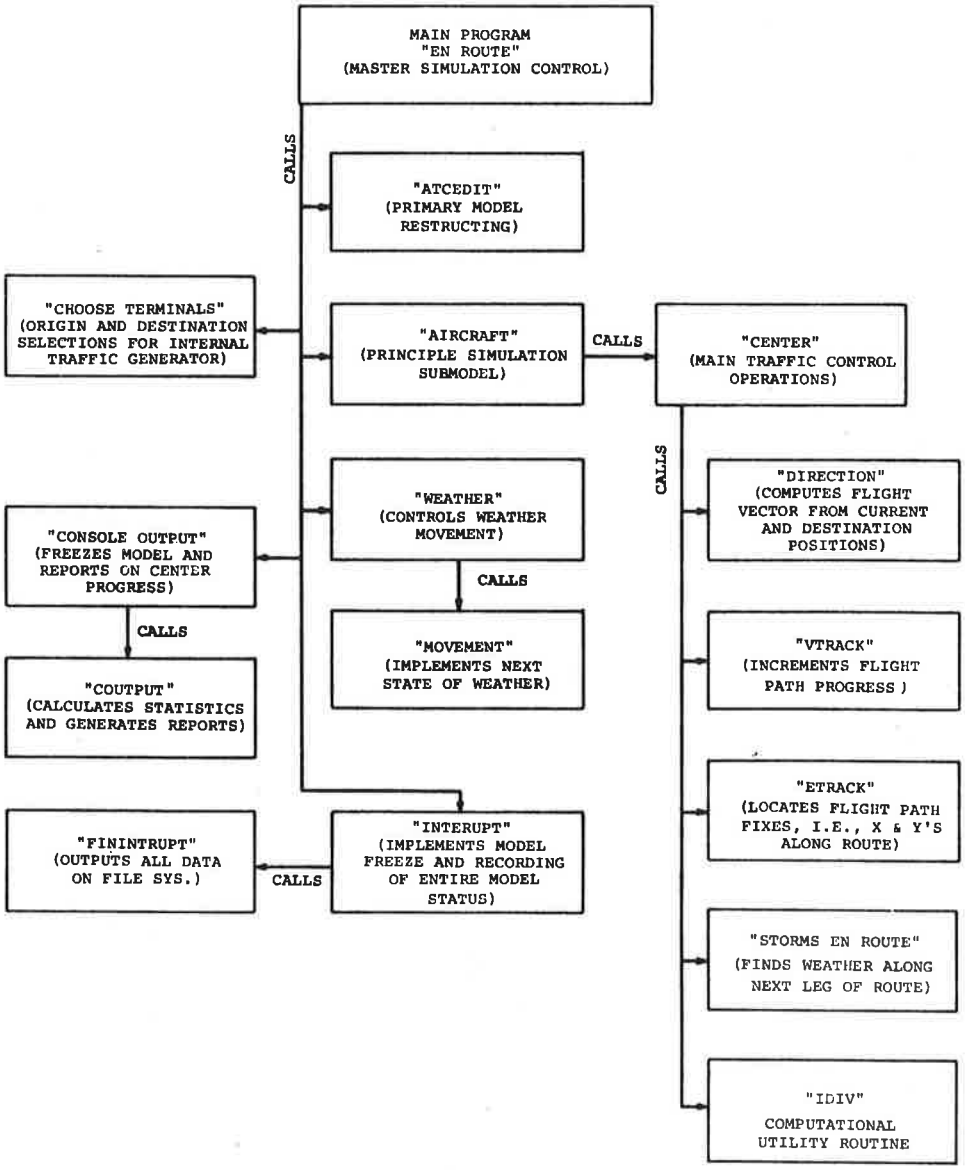
Appendix B

Detailed Program Information - En Route Air Traffic Flow Simulation

APPENDIX B

DETAILED PROGRAM INFORMATION
EN ROUTE AIR TRAFFIC FLOW SIMULATION

The program can be better understood if one can see the program elements and how they are used or called by the higher level elements. The following diagram pictures the elements and their calling hierachy:



```
enroute: proc;
/* ********************Run Time SIMPL Declarations*********************************************** */
dcl kywd$schedule entry (ptr, float bin);
dcl kywd$schedule_ab entry (ptr, fixed bin, ptr);
dcl kywd$reschedule entry (ptr, float bin);
dcl kywd$reschedule_ab entry (ptr, fixed bin, ptr);
dcl kywd$suspend entry (ptr);
dcl kywd$cancel entry (ptr);
dcl kywd$event_time entry (ptr) returns (float bin);
dcl kywd$terminate entry (ptr);
dcl setpak$number entry(ptr) returns(fixed bin(17));
dcl setpak$member entry (ptr, ptr) returns (bit(1));
dcl setpak$file entry (ptr, ptr);
dcl setpak$file_ab entry (ptr,ptr, fixed bin,ptr);
dcl setpak$remove entry (ptr,ptr);
dcl setpak$clear entry (ptr);
dcl setpak$void entry (ptr);
dcl setpak$order entry(ptr,ptr) returns(fixed bin(17));
dcl setpak$first ext entry (ptr) returns (ptr);
dcl setpak$exist ext entry (ptr) returns (bit(1));
dcl setpak$last ext entry (ptr) returns (ptr);
dcl setpak$predecessor (ptr) returns (ptr);
dcl setpak$successor (ptr) returns (ptr);
dcl statpak$uniform entry (float bin, float bin) returns (float bin);
dcl statpak$i_uniform entry (fixed bin, fixed bin) returns (fixed bin);
dcl statpak$expon entry (float bin) returns (float bin);
dcl statpak$poisson entry (float bin) returns (fixed bin);
dcl statpak$normal entry (float bin,float bin) returns (float bin);
dcl statpak$erlang entry (float bin,fixed bin) returns (float bin);
dcl ioa_ external entry;
dcl cu_$stack_frame_ptr entry external;
dcl (addr, null) builtin;
dcl delay entry (float bin);
dcl proc_storage entry (char(*), fixed, bit(1));

dcl (current ptr,                                   /* ptr to currently active process */
     time  float bin,                                     /* system time */
First ptr,
Last ptr) ext static;

dcl 1 S ext static,                                 /* system utilities */
    2 tref float bin,                                     /* time referenced */
    2 ( p,mainp) ptr,                               /* ptrs to process and main prog data structure */
    2 ( sqs_ptr,                                    /* sqs_pointer */
     p1,                                  /* ptr to proc_tbl entry */
     temp_ptr,
     temp_el) ptr,                                  /* temporary cells */
    2 (ex_sw,                                       /* external sw */
     lab_sw,                                        /* label sw */
     temp_bit) bit(1),                              /* temp bit */
    2 (RP,                                          /* Reactivation ptr */
```

Declarations

```
     temp_label) label,
    2 temp_char char(128) varying,
    2 temp_float float,
    2 temp_fixed fixed,
    2 temp_bin bit(72) varying,
    2 SP ptr;
dcl LABEL(3) based ptr;

dcl 1 set based,
    2 process_reference ptr init(null),
    2 pred ptr init(null),
    2 suc ptr init(null);

dcl 1 process_table based,
    2 process_name char(32) init((32)" "),
    2 process_rp label,
    2 data_ptr ptr init(null),
    2 sm ptr init(null),
    2 rp_sw bit(1) aligned init ("0"b),
    2 ext_sw bit(1) aligned init("0"b),
    2 proc_id fixed init (0);

dcl 1 sqs based,
    2 timeref float init(0.0e0),
    2 process_table_ptr ptr init(null),
    2 suc ptr init (null),
    2 pred ptr init (null);

dcl 1 set_membership based,
    2 set_ptr ptr init(null),
    2 suc ptr init (null);

/* global parameters */
dcl (ac_in_proc1,ac_in_proc2) file input;

dcl 1 enroute_data based (S,mainp),
    2 tgtime float,
    2 acrate fixed,
    2 frate float,
    2 intrp bit(1),
    2 intrp_thru bit(1),
    2 nextoutput float,
    2 interupttime float,
    2 iswg fixed,
    2 resp_char(1),
    2 tg(48) fixed,
    2 nb fixed,
    2 nc fixed,
    2 tgptr fixed,
    2 itg fixed,
    2 jtg fixed,
    2 rec1 float,
    2 cons1 float,
    2 filetime float,
    2 chvd char(1),
    2 wh bit(1),
    2 fwrd float,
    2 integfile fixed,
    2 recno fixed,

/* temp label */


/* temp cells for tracing */


    2 consno fixed,
    2 gid fixed,
    2 gdx fixed,
    2 gdy fixed,
    2 gx fixed,
    2 gy fixed,
    2 gdata(11) float,
    2 gdtime float,
    2 gairt fixed,
    2 genrdelay float,
    2 gct fixed,
    2 rsettime float,
    2 last_id fixed,
    2 scale float,
    2 xsize fixed,
    2 ysize fixed,
    2 pwd fixed,
    2 ac_intrupt fixed,
    2 rset bit(1),
    2 intime float,
    2 i fixed,
    2 j fixed,
    2 k fixed,
    2 gcall_in float,
    2 rn1 float,
    2 rn2 float,
    2 rn3 fixed,
    2 cno fixed,
    2 tbase float,
    2 iu fixed,
    2 ip fixed;
```

Declarations

```
        call ioa_("Give number of hours per cycle,4,8,12,16,20,or 24?");
        call read_list_(i);
        j = 1;
m5:     if j>i*2 then go to m4;
        call ioa_("Give 4 half hour quantities of aircraft to be generated  per line,");
        call read_list_(tg(j),tg(j+1),tg(j+2),tg(j+3));
        j = j+4;
        go to m5;
m4:     tgptr = 1;
        tgtime = ((time-rsettime)/3.6e3)+tbase;
        go to m14;
                                          /* following reads interupted aircraft and traffic gen in if needed */
m8:     call ioa_("Which input file,ac_in_proc'1' or ac_in_proc'2' ?");
        call read_list_(iu);
        if iu = 1 then do;
              open file(ac_in_proc1);
           go to m7;
        end;

        if iu = 2 then do;
             open file(ac_in_proc2);
             go to m7;
        end;

m10:    call ioa_("No file");
        go to m6;
m7:     i = 0;
        call read_list_(ip);
        if ip = 0 | ac_intrupt<1 then do;
             go to m10;
        end;

        rset = "1"b;
m9:     i = i+1;
        if iu = 2 then go to m9a;
        get file(ac_in_proc1) list(gid,gdx,gdy,gx,gy,gitime,gairt,gct,gcall_in);
        get file(ac_in_proc1) list(gdata(1),gdata(2),gdata(3),gdata(4),gdata(5),gdata(6),gdata(7),gdata(8),gdata(9),
        gdata(10),gdata(11));
        go to m9b;
m9a:    get file(ac_in_proc2) list(gid,gdx,gdy,gx,gy,gitime,gairt,gct,gcall_in);
        get file(ac_in_proc2) list(gdata(1),gdata(2),gdata(3),gdata(4),gdata(5),gdata(6),gdata(7),gdata(8),gdata(9),
        gdata(10),gdata(11));
m9b:    gdtime = gdtime+gcall_in+time;
        genrdelay = 0.0e0;
        if gcall_in>0.0e0 then genrdelay = 1.0e0;
        do;call kywd$schedule(current,time);
                  lab_sw="1"b;
        call aircraft(gid,gdx,gdy,gx,gy,gdata,gairt,gct,genrdelay);
                  lab_sw="0"b;
```

if changing traffic generator, read in proper amount of new values

when restarting initiate reading of data on "frozen" airborne aircraft

read in an aircraft data

reactivate that aircraft

```
                tref=gdtime;
                call kywd$schedule(p1,tref);
                RP=mm9a;
                return;
          mm9a:end;
          if i<ac_intrupt then go to m9;
          rset = "0"b;
                                                    /* reads traffic gen parameters */
          if iu = 2 then go to m9c;
          get file(ac_in_proc1) list(tgptr,tgtime);
          go to m9d;
m9c:      get file(ac_in_proc2) list(tgptr,tgtime);
m9d:      do i = 1 to 6;
               j = i*8-7;
               if iu = 1 then do;
                    get file(ac_in_proc1) list(tg(i),tg(i+1),tg(i+2),tg(i+3),
                    tg(i+4),tg(i+5),tg(i+6),tg(i+7));
               end;

               if iu = 2 then do;
                    get file(ac_in_proc2) list(tg(i),tg(i+1),tg(i+2),tg(i+3),
                    tg(i+4),tg(i+5),tg(i+6),tg(i+7));
               end;

          end;

          if iu=1 then close file(ac_in_proc1);if iu=2 then close file(ac_in_proc2);
          go to m3;
m14:      call ioa_("Set interupt in '?' sec, '0' means no interuptions");
          call read_list_(interupttime);
          do;call kywd$schedule(current,time);
                    lab_sw="1"b;
                    call weather;
                    lab_sw="0"b;
                    tref=time+3.6e3;
                    call kywd$schedule(p1,tref);
                    RP=mm9c;
                    return;
          mm9c:end;
          if interupttime = 0.0e0 then go to m11;
          do;call kywd$schedule(current,time);
                    lab_sw="1"b;
                    call interupt;
                    lab_sw="0"b;
                    tref=time+interupttime;
                    call kywd$schedule(p1,tref);
                    RP=mm9d;
                    return;
          mm9d:end;
m11:      call ioa_("Console report in '?' sec, '0' means none.");
```

at the end of reading in aircraft, read in the previous traffic generator and its "frozen" status

activate the weather movement in one hour simulated time

activate the interrupt function at the requested time

```
          call read_list_(nextoutput);
          if nextoutput = 0.0e0 then go to m12;
                    do;call kywd$schedule(current,time);
                    lab_sw="1"b;
                    call console_output;
                    lab_sw="0"b;
                    tref=time+nextoutput;
                    call kywd$schedule(p1,tref);
                    RP=mm9e;
```

activate the console report generator at the request time

```
                    return;
          mm9e:end;
m12:      call ioa_("Record every '?' and console trace every '?' aircraft?");
          call read_list_(recno,consno);
          rno = 10000;if recno=0 then rno=0;
          cno = 10000;if consno=0 then cno=0;
                                                        /* aircraft generator */
genac:    gdtime = ((time-rsettime)/3.6e3)+tbase;
          if (gdtime-tgtime)>1.8e3 then do;
               tgptr = tgptr+1;
               if tg(tgptr)^=0 then go to m15;
               tgptr = 1;
m15:           acrate = tg(tgptr);
               tgtime = gdtime;
          end;

          iswg = 1;                                     /* for m17 */
                                                        /* test switch for branching */
m15:      go to swgo(iswg);
                                                        /* delay until time to generate a new aircraft */
m17:      frate = 1.8e3/float(acrate);
          frate = frate+statpak$uniform(-frate,frate);

          do;
               call delay(frate);
               RP = enroute_3;
               return;
enroute_3: end;

          rn1 = statpak$uniform(1.0e0,5.0e1);           /* determining origin and destination,unif dist from terminals */
          rn2 = statpak$uniform(1.0e0,5.0e1);
          call choose_terminals(gx,gy,gdx,gdy,rn1,rn2);
          last_id = last_id +1;
```

```
gid = last_id;
rno = rno+1;
gairt = 0;
if recno<rno then do;
     rno = 1;
     gairt = 1;
end;

cno = cno+1;
gct = 0;
if consno<cno then do;
     cno = 1;
     gct = 1;
end;

genrdelay = 0.0e0;
do;call kywd$schedule(current,time);
          lab_sw="1"b;
call aircraft(gid,gdx,gdy,gx,gy,giata,gairt,gct,genrdelay);
          lab_sw="0"b;
          tref=time;
          call kywd$schedule(p1,tref);
          RP=mm9b;
          return;
mm9b:end;
```

tag the aircraft if it is to be recorded on the files

tag the aircraft if it is to be traced on the console

the aircraft is activated here

the interupt flag

```
if intrp then go to m16;

do;
     call delay(7.0e2);
     RP = enroute_4;
     return;
enroute_4: end;

/* allows all aircraft to be interupted */
call finintrupt(filetime,last_id,scale,xsize,ysize,pwh,ac_intrupt);
go to m1;
```

delay long enough for all aircraft to be frozen

finish recording the models complete data

```
dcl  aircraft_count fixed ext static init(1);


*/

 delays, */


aircraft:proc(id_,dx_,dy_,x_,y_,data_,airt_,ct_,enrdelay_);

          dcl busy(21) fixed ext static;
dcl 1 aircraft_data based (S,p),
    2 id fixed,
    2 dx fixed,
    2 dy fixed,
    2 x fixed,
    2 y fixed,
    2 data(11) float,
    2 dtime float,
    2 airt fixed,
    2 ct fixed,
    2 enrdelay float,
    2 totime float,
    2 alt float,
    2 speed float,
    2 vector float,
    2 call_in float,
    2 call_time float,
    2 comdel float,
    2 hold float,
    2 ii fixed,
    2 jj fixed

    2 cent fixed;
              call cu_$stack_frame_ptr(S,SP);
              addr(S,RP)->LABEL(2) = S,SP;

              if lab_sw then S,RP = aircraft_1;
              go to S,RP;
aircraft_1:   allocate aircraft_data set (p);
              S,RP = aircraft_2;

              call proc_storage("aircraft",aircraft_count,"0"b);
              aircraft_count = aircraft_count+1;
dcl (id_,dx_,dy_,x_,y_,airt_,ct_) fixed,
          (data_(10),enrdelay_) float,iqz fixed;
          S,p->id=id_;
```

```
/* test if restarted ,rset=true. */
/* else initialize */
/* data array=,1 time in hours,2 x,3 y,(both with +100 bias),4 alt,5 speed,

/* 6 vector(+100),7 communication cum delays,8 enroute cum delays,9 gnd hold

/* 10 take off time in hours,and 11 aircraft id number */
/* if was interupted in middle,restarts at a2 */
```

declarations

```
/* test if restarted ,rset=true. */
,
```

```
          S.p->dx=dx_;
          S.p->dy=dy_;
          S.p->x=x_;
          S.p->y=y_;
          do iqz=1 to 10;
                    S.p->data(iqz)=data_(iqz);
          end;
          S.p->airt=airt_;
          S.p->ct=ct_;
          S.p->enrdelay=enrdelay_;
               return;
aircraft_terminate: entry;
               free S.p->aircraft_data;
               return;
aircraft_2:    if rset then go to restor;
                                                        /* else initialize */
               do ii = 1 to 10;
                    data(ii) = 0.0e0;
               end;

               data(1) = ((time-rsettime)/3.6e3)+tbase;
               alt = 0.0e0;
                    speed=0.0e0;
                    vector=1.0e2;
               data(11) = float(ii)+.01e0;
               go to a1;
                                                        /* data array=,1 time in hours,2 x,3 y,(both with +100 bias),4 alt,5 speed,
*/
                                                        /* 6 vector(+100),7 communication cum delays,8 enroute cum delays,9 gnd hold
 delays, */
                                                        /* 10 take off time in hours,and 11 aircraft id number */
restor:        alt = data(4);
               speed = data(5);
               vector = data(6);
               totime = data(10);
               if enrdelay>0.0e0 then go to a2;
                                                        /* if was interupted in middle,restarts at a2 */
a1:            call_in = 0.0e0;
               dtime = ((time-rsettime)/3.6e3)+tbase;
               hold = 0.0e0;
               enrdelay = 0.0e0;
               if ct = 1 then call ioa_("Aircraft number ~i activated at ~f,destination x=~d,y=~d",id,dtime,dx,dy);
callagain:     call center(time,id,rsettime,tbase,totime,x,y,alt,speed,vector,data,dx,dy,call_time,comdel,hold,call_in,enrdelay,cent);
               data(2) = float(x);
               data(3) = float(y);
               data(4) = alt;
               data(5) = speed;
               data(6) = vector;
               if comdel>0.0e0 then do;
                    data(7) = data(7)+comdel;
```

console trace

aircraft contacts center for clearance and movement to next check point

update flight status

check for communication delays

```
                    do;
                        call delay(comdel);
                        RP = aircraft_3;
                        return;
aircraft_3:             end;
                        if ct = 1 then call ioa_("Aircraft ^d,Center ^d communication channels busy.",id,cent);
                        go to callagain;
                end;
```

delay until channel available

console trace

```
                if call_time>0.0e0 then do;
                    call delay(call_time);
                    RP = aircraft_4;
                    return;
aircraft_4:         end;
                    if ct = 1 then call ioa_("Aircraft ^d,communication time= ^f sec,with Center ^d",id,call_time,cent);
                    data(10) = totime;
            busy(cent)=busy(cent)-1;
                    if hold>0.0e0&speed = 0.0e0 then do;
                        data(9) = data(9)+hold;
                        if ct = 1 then call ioa_("Aircraft ^d,takeoff clearence delayed ^f sec.",id,hold);
                        hold = 0.0e0;
                    end;

                    data(8) = data(8)+enrdelay;
                    enrdelay = 0.0e0;
                    if ct = 1 then call ioa_("Aircraft ^d,call in ^f sec.",id,call_in);
                    dtime = 6.1e2;
a3:                 if intrp then go to a4;
                    if call_in<6.1e2 then dtime = call_in;

                    do;
                    call delay(dtime);
                    RP = aircraft_5;
                    return;
```

allow for communication time

console trace

release the communication channel to the center

test if aircraft not cleared from ground.

hold console trace

record data

console trace

test if simulation being interrupted

continue until the next time to check in (checking every 610 sec. for occurance of interruption)

```
aircraft_5:     end;
                call_in = call_in-itime;
                if call_in >0.0e0 then go to a3;                    record data
a2:             call_in = 0.0e0;
                enrdelay = 0.0e0;
                data(1) = ((time-rsettime)/3.5e3)+tbase;
                if airt = 1 then do;
                     call fo("aircraft_progress");
                     call ioa_$nnl(""f "f "f "f "f",data(11),data(1),data(2),data(3),data(4));
                     call ioa_(" "f "f "f "f "f",data(5),data(6),data(7),data(8),data(9),data(10));
                     call co;
                end;

                if ct = 1 then call ioa_("Aircraft "d,time="f,x="d,y="d,alt="f,speed="f,vector="f",id,data(1),x,y,alt,speed,vector);
                if alt = 0.0e0&speed>0.0e0 then go to a5;
                go to callagain;
                                                    /* interupt saving of aircraft still flying */
a4:             dtime = time-intime;
                                                    /* finds time beYond initiation of interupt */
                                                    /* Write out the status ot the interupted aircraft */
                if integfile = 1 then do;
                     call fo("ac_in_proc1");
                     go to a6;
                end;

                if integfile = 2 then do;
                     call fo("ac_in_proc2");
                     go to a6;
                end;

                go to a5;
a6:             call ioa_(""d "d "i "d "d "f "d "d "f",id,dx,iy,x,y,dtime,airt,ct,call_in);

                call ioa_$nnl(""f "f "f "f "f",data(1),data(2),data(3),data(4),data(5));
                call ioa_(" "f "f "f "f "f "f",data(6),data(7),data(8),data(9),data(10),data(11));
                call co;
                ac_intrupt = ac_intrupt+1;    increment the count of aircraft interrupted
                call kywdSterminate(current);
                free S.p->aircraft_data;
                return;
a5:         end aircraft;                     end of "aircraft" activity
```

if tagged record flight progress info.

if tagged print out progress on console

interruption, record data and stop

"Interupt" Activity

```
dcl  interupt_count fixed ext static init(1);
interupt: proc;


dcl 1 interupt_data based (S.p),
    2 inti fixed,
    2 r fixed;
               call cu_$stack_frame_ptr(S.SP);
               addr(S.RP)->LABEL(2) = S.SP;

               if lab_sw then S.RP = interupt_1;
               go to S.RP;
interupt_1:    allocate interupt_data set (p);
               S.RP = interupt_2;

               call proc_storage("interupt",interupt_count,"3"b);
               interupt_count = interupt_count+1;
               return;
interupt_terminate: entry;
               free S.p->interupt_data;
               return;
interupt_2:    filetime = (time-rsettime)/3.6e3+tbase;
               intime = time;

               ac_intrupt = 0;
               intrp = "1"b;
               call ioa_("Which output file,ac_in_proc '1' or '2' ?");
               call read_list_(integfile);
          if integfile=1 then call fo("ac_in_proc1");
          if integfile =2 then call fo("ac_in_proc2");
          call ioa_("1");
          call co;

               do;
                    call delay(6,5e2);
                    RP = interupt_3;
                    return;
interupt_3:    end;


               if integfile = 1 then do;
                    call fo("ac_in_proc1");
                    go to a1;
               end;

               if integfile = 2 then do;
                    call fo("ac_in_proc2");
                    go to a1;
               end;
```

/* Set aircraft termination flag and find which output file to use */

/* delay long enough for all other activities to terminate */
/* write out traffic generator parameters */

```
                call ioa_("No files");
                go to a2;
a1:             call ioa_(""d ^f",tgptr,tgtime);
                do int1 = 1 to 6;
                     r = int1*8-7;
                     call ioa_$nnl(""d ^d ^d ^d ^d",tg(r),tg(r+1),tg(r+2),tg(r+3),tg(r+4));
                     call ioa_(" ^d ^d ^d",tg(r+5),tg(r+6),tg(r+7));
                end;

                call co;
                intrp_thru = "1"b;
                call kywd$terminate(current);
                free S.p->interupt_data;
                return;
a2:        end interupt;

dcl  weather_count fixed ext static init(1);
weather:  proc;


dcl 1 weather_data based (S.p),
    2 dt float;
dcl  movement entry ext;
                call cu_$stack_frame_ptr(S.SP);
                addr(S.RP)->LABEL(2) = S.SP;

                if lab_sw then S.RP = weather_1;
                go to S.RP;
weather_1:      allocate weather_data set (p);
                S.RP = weather_2;

                call proc_storage("weather",weather_count,"0"b);
                weather_count = weather_count+1;
                return;
weather_terminate: entry;
                free S.p->weather_data;
                return;
weather_2: a1: call movement(tbase,rsettime,time,dt);
                if intrp then go to a2;

                do;
                     call delay(dt);
                     RP = weather_3;
                     return;
weather_3:      end;
                go to a1;
                a2:call kywd$terminate(current);
                free S.p->weather_data;
                return;
          end weather;
```

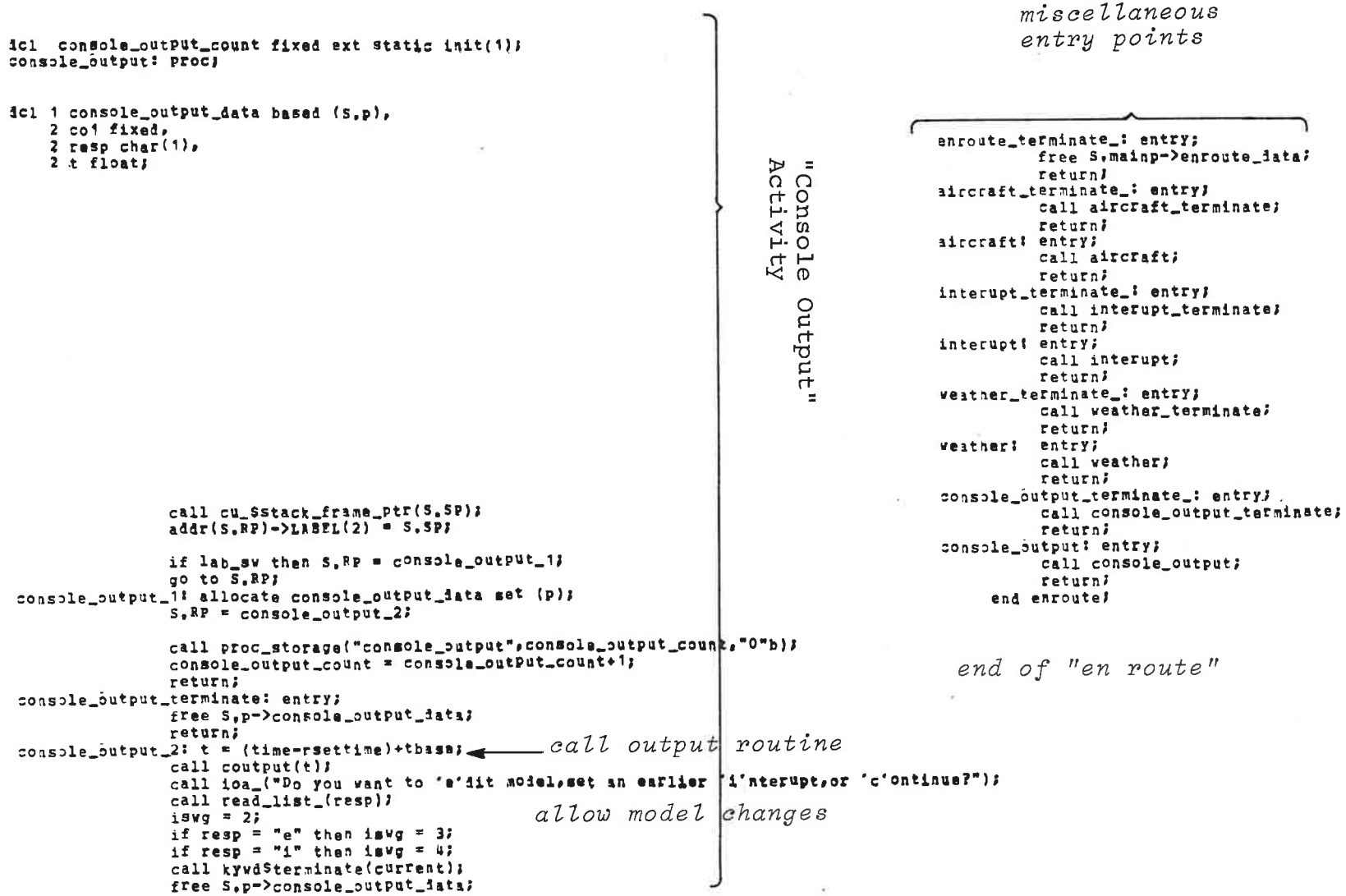
write out traffic generator info.

end "interupt" activity

move the weather an increment

recycle every hour

"Weather" Activity



```
dcl  console_output_count fixed ext static init(1);
console_output: proc;

dcl 1 console_output_data based (S,p),
    2 co1 fixed,
    2 resp char(1),
    2 t float;

                call cu_$stack_frame_ptr(S,SP);
                addr(S,RP)->LABEL(2) = S,SP;

                if lab_sw then S,RP = console_output_1;
                go to S,RP;
console_output_1: allocate console_output_data set (p);
                S,RP = console_output_2;

                call proc_storage("console_output",console_output_count,"0"b);
                console_output_count = console_output_count+1;
                return;
console_output_terminate: entry;
                free S,p->console_output_data;
                return;
console_output_2: t = (time-rsettime)+tbase;
                call coutput(t);
                call ioa_("Do you want to 'e'dit model,set an earlier 'i'nterupt,or 'c'ontinue?");
                call read_list_(resp);
                iswg = 2;
                if resp = "e" then iswg = 3;
                if resp = "i" then iswg = 4;
                call kywd$terminate(current);
                free S,p->console_output_data;
                return;
          end console_output;
enisim:   sqs_ptr = null;
          return;

enroute_terminate_: entry;
          free S,mainp->enroute_data;
          return;
aircraft_terminate_: entry;
          call aircraft_terminate;
          return;
aircraft: entry;
          call aircraft;
          return;
interupt_terminate_: entry;
          call interupt_terminate;
          return;
interupt: entry;
          call interupt;
          return;
weather_terminate_: entry;
          call weather_terminate;
          return;
weather:  entry;
          call weather;
          return;
console_output_terminate_: entry;
          call console_output_terminate;
          return;
console_output: entry;
          call console_output;
          return;
     end enroute;
```

"Console Output" Activity

call output routine

allow model changes

miscellaneous entry points

end of "en route"

```
atcedit:proc(filetime,last_id,lscale,lxsize,lysize,pvh,ac_intrupt);
        dcl aa(100,50) fixed ext static,ww(100,50) fixed ext static,tl(100,50) fixed ext static,
                scale float ext static,xsize fixed ext static,ysize fixed ext static,
                avail(21) fixed ext static,busy(21) fixed ext static,ctrs(21,21) float ext static,
                oper(21) fixed ext static,ctl(300) fixed ext static,ho(21,21) float ext static,
                ho_last(21,21) float ext static,terms(51) fixed ext static;
        dcl interupttime float,last_id fixed,cx(4) fixed,cy(4) fixed,filetime float,pvh fixed,
                resp char(1),word char(10),word1 char(10),word2 char(10),fword1 float,fword2 float,
                jword fixed,jword1 fixed,jword2 fixed,i fixed,j fixed, k fixed, l fixed, m fixed,
                id fixed,ac_intrupt fixed,vh bit(1),bi bit(1),lxsize fixed,lysize fixed,lscale float;
        dcl center ext entry,direction ext entry returns (float),storm_enroute ext entry,vtrack ext entry,
                etrack ext entry,idiv ext entry returns(fixed);
        dcl iq fixed,(restart1,restart2) file input,jword fixed;
/*pass local arguements to external parameters*/
        xsize=lxsize;
        ysize=lysize;
        scale=lscale;
        call ioa_("'n'ew or 'o'ld data base?");
                call read_list_(resp);
        vh="0"b;
        pvh=0;
        if resp="n" then go to e1;
        call ioa_("Read from restart'1' or restart'2' ?");
        call read_list_(resp);
        if resp="2" then go to e2;
        iq=1;open file(restart1) input;
        get file(restart1) list(jword);
        if jword=0 then go to e2;
        get file(restart1) list(filetime);
        go to readfile;
er1:call ioa_("Both restart files are empty,switched to new data base.");
        go to e1;
e2:iq=2;open file(restart2) input;
        get file(restart2) list(jword);
        if jword=0 then go to er1;
        get file(restart2) list(filetime);
        go to readfile;
```

```
readfile:if iq=2 then go to read2a;
         get file(restart1) list(xsize,ysize);
         get file(restart1) list(scale,last_id);
         go to read2b;
         read2a:get file(restart2) list(xsize,ysize);
         get file(restart2) list(scale,last_id);
         read2b:do i=1 to 21;
         if iq=1 then get file(restart1) list(qword,avail(i),busy(i),oper(i));
         if iq=2 then get file(restart2) list(qword,avail(i),busy(i),oper(i));
         end;
         do i=1 to 3;
         j=i*7-6;
         do m=1 to 3;
         if iq=1 then get file(restart1) list(dd,dd,dd,id,di,id,dd);
         if iq=2 then get file(restart2) list(dd,dd,dd,id,di,id,dd);
```

read in the restart info. from file

```
         do k=1 to 21;
         if iq=2 then go to read2c;
         if m=1 then
get file(restart1) list(dd,ctrs(k,j),ctrs(k,j+1),ctrs(k,j+2),ctrs(k,j+3),ctrs(k,j+4),ctrs(k,j+5),ctrs(k,j+6));
         if m=2 then
get file(restart1) list(dd,ho(k,j),ho(k,j+1),ho(k,j+2),ho(k,j+3),ho(k,j+4),ho(k,j+5),ho(k,j+6));
         if m=3 then
get file(restart1) list(dd,ho_last(k,j),ho_last(k,j+1),ho_last(k,j+2),ho_last(k,j+3),ho_last(k,j+4),ho_last(k,j+5),
                        ho_last(k,j+6));
         go to read2d;
read2c:if m=1 then
 get file(restart2) list(dd,ctrs(k,j),ctrs(k,j+1),ctrs(k,j+2),ctrs(k,j+3),ctrs(k,j+4),ctrs(k,j+5),ctrs(k,j+6));
         if m=2 then
 get file(restart2) list(dd,ho(k,j),ho(k,j+1),ho(k,j+2),ho(k,j+3),ho(k,j+4),ho(k,j+5),ho(k,j+6));
         if m=3 then
 get file(restart2) list(dd,ho_last(k,j),ho_last(k,j+1),ho_last(k,j+2),ho_last(k,j+3),ho_last(k,j+4),ho_last
         (k,j+5),ho_last(k,j+6));
         read2d:end;
end;
end;
```

```
          do i=1 to 50;
          j=i*6-5;
          if iq=1 then get file(restart1) list(dd,ctl(j),ctl(j+1),ctl(j+2),ctl(j+3),ctl(j+4),ctl(j+5));
          if iq=2 then get file(restart2) list(dd,ctl(j),ctl(j+1),ctl(j+2),ctl(j+3),ctl(j+4),ctl(j+5));
          end;
/*following reads 10 strips of the map*/
          do i=1 to 10;
          j=i*10-9;
          do m=1 to 3;
          if iq=1 then get file(restart1) list(dd,dd,dd,dd,dd,dd,dd,dd,dd,dd);
          if iq=2 then get file(restart2) list(dd,dd,dd,dd,dd,dd,dd,dd,dd,dd);
          do k=1 to 50;
          l=51-k;
          if iq=2 then go to read2e;
          if m=1 then
get file(restart1) list(dd,aa(j,l),aa(j+1,l),aa(j+2,l),aa(j+3,l),aa(j+4,l),aa(j+5,l),aa(j+6,l),aa(j+7,l),
                                  aa(j+8,l),aa(j+9,l));
          if m=2 then
get file(restart1) list(dd,vv(j,l),vv(j+1,l),vv(j+2,l),vv(j+3,l),vv(j+4,l),vv(j+5,l),vv(j+6,l),vv(j+7,l),
                                  vv(j+8,l),vv(j+9,l));
          if m=3 then
get file(restart1) list(dd,tl(j,l),tl(j+1,l),tl(j+2,l),tl(j+3,l),tl(j+4,l),tl(j+5,l),tl(j+6,l),
                                  tl(j+7,l),tl(j+8,l),tl(j+9,l));
          go to read2f;
          read2e:if m=1 then
get file(restart2) list(dd,aa(j,l),aa(j+1,l),aa(j+2,l),aa(j+3,l),aa(j+4,l),aa(j+5,l),aa(j+6,l),
          aa(j+7,l),aa(j+8,l),aa(j+9,l));
          if m=2 then
 get file(restart2) list(dd,vv(j,l),vv(j+1,l),vv(j+2,l),vv(j+3,l),vv(j+4,l),vv(j+5,l),vv(j+6,l),
          vv(j+7,l),vv(j+8,l),vv(j+9,l));
          if m=3 then
 get file(restart2) list(dd,tl(j,l),tl(j+1,l),tl(j+2,l),tl(j+3,l),tl(j+4,l),tl(j+5,l),tl(j+6,l),
          tl(j+7,l),tl(j+8,l),tl(j+9,l));
          read2f:end;
          end;
end;
          if iq=1 then get file(restart1) list(terms(1));
          if iq=2 then get file(restart2) list(terms(1));
          do i=2 to 50 by 2;
```

reads in more restart info.

```
          if iq=1 then get file(restart1) list(terms(i),terms(i+1));
          if iq=2 then get file(restart2) list(terms(i),terms(i+1));
          end;
```

```
          if iq=1 then get file(restart1) list(aq_intrupt);
          if iq=2 then get file(restart2) list(aq_intrupt);
          if iq=1 then close file(restart1);
          if iq=2 then close file(restart2);
          wh="1"b;
          pwh=1;
/*wh and pwh are true when a restarted data base was used*/
          e1:if ^wh then go to e25;
e4:call ioa_("Do you want to change maps or parameters?'y' or 'n'");
          call read_list_(resp);
          if resp="n" then go to sim1;
e5:call ioa_("Edit which?'c'enter map,'v'eather map,'t'erminal map,'h'andover restrictions,or'o'perations");
          call read_list_(resp);
          if resp="c" then go to e11;
          if resp="v" then go to e7;
          if resp="t" then go to e8;
          if resp="h" then go to e9;
          if resp="o" then go to e10;
          go to e24;
/*edit map of center*/
e3:do i=1 to xsize;
          do j=1 to ysize;
                    aa(i,j)=0;
          end;end;
e11:call ioa_("Map input-Center number?");
          call read_list_(dword);
          if dword=0 then
                    do;if wh then go to e5;
                              go to e12;
                              end;
          call ioa_("Give blx,brx,bly,tly of a rectangular portion");
          call read_list_(cx(1),cx(2),cy(1),cy(2));
          do i=cx(1) to cx(2);
          do j=cy(1) to cy(2);
                    aa(i,j)=dword;
          end;end;
          go to e11;
          e30:if oper(21)=0 then do;
          ctl(1)=ctl(1)+1;
                    oper(21)=ctl(1)+1;
                    k=oper(21);
                    ctl(k)=1;
                    ctl(k+1)=20;
                    ctl(k+2)=40;
                    ctl(k+3)=3;
                    ctl(k+4)=5;
                    ctl(k+5)=20;
                    ctl(k+6)=4;
                    ctl(k+7)=20;
                    ctl(k+8)=120;
                    ctl(1)=ctl(1)+9;
          end;
          go to e10;
```

optional choices when restarted run - (no choice when new data base, you are sent to each major question block)

map edit

nominal procedural list for Centers not specified otherwise

```
/*weather edit*/
e7:call ioa_("Clear existing weather?'Y' or'n'");
          call read_list_(resp);
          if resp="n" then go to e13;
e12:do i=1 to xsize;
          do j=1 to ysize;
                    ww(i,j)=0;
          end;end;
e13:call ioa_("Weather input-Give severity(1,2,3 or 4)");
          call read_list_(dword);
          if dword=0 then do;
                    if wh then go to e5;
                    go to e14;
                    end;
          call ioa_("Give blx,brx,bly,tly portion of rectangular shape");
          call read_list_(cx(1),cx(2),cy(1),cy(2));
          do i=cx(1),cx(2);
          do j=cy(1),cy(2);
                    ww(i,j)=dword;
          end;end;
          go to e13;
/*terminal edit*/
e14:do i=1 to 51;
          terms(i)=0;
          end;
e8:do i=1 to 100;
          do j=1 to 50;
                    tl(i,j)=0;
          end;end;
          if wh then go to e15;
          e16:terms(1)=terms(1)+1;
          call ioa_("Give new terminal location x y ");  /*0 0 answer ends input here*/
          i=terms(1)*2;
          call read_list_(terms(i),terms(i+1));
                    if terms(i) ^=0 then go to e16;
                    terms(1)=terms(1)-1;
```

weather edit

terminal edit

```
/*fill map*/
          do i=1 to terms(1);
                    j=i*2;
                    tl(terms(j),terms(j+1))=1;
          end;
          if wh then go to e5;
          go to e19;
e15:call ioa_("Do you want to remove a terminal?'y' or 'n'");
          call read_list_(resp);
          if resp="n" then go to e16;
e17:call ioa_("Give old terminal location x y ");
          call read_list_(cx(1),cy(1));
          if cx(1)=0 then go to e15;
          do i=1 to terms(1);
                    j=i*2;
                    if terms(j)=cx(1)&terms(j+1)=cy(1) then go to e18;
          end;
          call ioa_("No terminal at that location");
          go to e17;
e18:do k=i to terms(1);
          l=k*2;
          terms(l)=terms(l+2);
          terms(l+1)=terms(l+3);
```

rest of terminal edit

```
          end;
          terms(1)=terms(1)-1;
          go to e17;
 /*handover restriction edit*/
 e19:do i=1 to 21;
           do j=1 to 21;
                    ho(i,j)=0.0e0;
           end;end;
 e9:call ioa_("Handover restriction input-Give center imposing restriction");
           call read_list_(dword1);
           if dword1 =0 then
                    do;if wh then go to e5;
                              go to e20;
           end;
           call ioa_("Give center receiving restriction and the min time (sec) between aircraft(mit)");
           call read_list_(dword2,fword1);
           if dword2=0 then go to e9;
           ho(dword1,dword2)=fword1;
           go to e9;
```

Center-to-Center handover restrictions edit

```
/*center operations edit*/
e20:do i=1 to 300;
          ctl(i)=0;
          end;
          do i=1 to 21;
          oper(i)=0;
          end;
go to e30;
/*no core recovery fo this array*/
e10:call ioa_("Procedure input-Give center number for new operating structure");
          call read_list_(dword1);
          if dword1=0 then go to e24;
          k=ctl(1)+1;
          oper(dword1)=k;
          call ioa_("Give takeoff operation number(1 or 2)");
          call read_list_(dword2);
          if dword2=2 then go to e21;
          call ioa_("Give call in time and delay time to next call (sec)");
          call read_list_(m,dd);
          ctl(1)=ctl(1)+3;
          ctl(k)=1;
          ctl(k+1)=m;
          ctl(k+2)=dd;
          go to e22;
e21:call ioa_("Give ils capability(0,1,2,3 or5),call in time and delay time to next call(sec)");
          call read_list_(dword2,m,dd);
          ctl(1)=ctl(1)+4;
          ctl(k)=2;
          ctl(k+1)=dword2;
          ctl(k+2)=m;
          ctl(k+3)=dd;
e22:call ioa_("Give landing operation number(only 3 currently)");
          call read_list_(dword2);
          call ioa_("Give ils capability (0,1,2,3 or 5) and call in time(sec)");
          call read_list_(dd,m);
          k=ctl(1)+1;
          ctl(1)=ctl(1)+3;
          ctl(k)=dword2;
          ctl(k+1)=dd;
          ctl(k+2)=m;
```

takeoff procedures set to #1

takeoff procedure set to #2

landing, set to 3

edit Center's procedural function lists. (inputs the number of the desired procedure, and any supporting data.)

```
e23:call ioa_("Give enroute operation number(only 4 currently)");
          call read_list_(dword2);
          call ioa_("Give call in time and delay to next call(sec)");
          call read_list_(m,dd);
          k=ctl(1)+1;
          ctl(1)=ctl(1)+3;
          ctl(k)=dword2;
          ctl(k+1)=m;
          ctl(k+2)=dd;
          go to e10;
e25:scale=1.15e-6;
          filetime=0.0e0;
          rsettime=0.0e0;
          tbase=0.0e0;
          last_id=0;
          xsize=100;
          ysize=50;
          avail(21)=1000;
          do i=1 to 20;
          busy(i)=0;
                    avail(i)=10;
          end;
          go to e3;
/*parameter edit*/
e24:call ioa_("Do you want to change remaining parameters?'y' or 'n'");
          call read_list_(resp);
          if resp="n" then go to e4;
          call ioa_("Edit which?'s'cale,'a'vailable center communication channels");
          if ^wh then call ioa_("or 't'ime base");
          call read_list_(resp);
          if resp="s" then go to e26;
          if resp="a" then go to e27;
          if ^wh&resp="t" then go to e28;
          go to e4;
/*edit scale*/
e26:call ioa_("Give new scale factor.");
          call read_list_(scale);
          go to e24;
/*edit communication channels*/
e27:call ioa_("Give center number and number of communication channels");
          call read_list_(dword1,dword2);
          if dword1=0 then go to e24;
          avail(dword1)=dword2;
          go to e27;
e28:call ioa_("Give time base in hours");
          call read_list_(tbase);
          filetime=tbase;
          go to e24;
  sim1:lscale=scale;
          lxsize=xsize;
          lysize=ysize;
          return;
end atcedit;
```

en route procedures set to 4

initialize some parameters if new model

edit the key parameters

```
center:proc(time,id,rsettime,tbase,totime,x,y,alt,speed,vector,data,dx,dy,call_time,
          comdel,hold,call_in,enrdelay,cent);
     dcl aa(100,50)fixed ext static,ww(100,50) fixed ext static,tl(100,50) fixed ext static,
          scale float ext static,xsize fixed ext static,ysize fixed ext static,avail(21) fixed ext static,
               busy(21) fixed ext static,ctrs(21,21) float ext static,oper(21) fixed ext static,
          ctl(300) fixed ext static,ho(21,21) float ext static,ho_last(21,21) float ext static,
          terms(51) fixed ext static;
     dcl (time,rsettime,tbase,comdel)float,(id,x,y,dx,dy)fixed,(alt,speed,vector,totime,call_time,call_in,
          enrdelay)float,(ii,jj,cent,contrl,sx,sy,mdis,mptr,dist,kk,rx,ry)fixed,(adis,resfuel,
          data(10),vr,tvr,ftime,dvr)float,(path(300),xx,yy,thunder,where(2))fixed,weathsw bit(1),
          bran(10 )  label initial  (cc1,cc2,cc3,cc4,cc5,cc6,cc7,cc8,cc9,cc10 ) ;
     dcl idiv ext entry returns(fixed),direction ext entry returns(float),
          storm_enroute ext entry,vtrack ext entry,etrack ext entry,choose_terminals ext entry;
     cent=aa(idiv(x,100),idiv(y,100));
     if cent<=0 | cent> 21 then cent=21;
     comdel=0.0e0;
cx1:if busy(cent)>avail(cent) then                                          } if communication
          do;comdel=10.0e0;                                                 } channels busy -
               call vtrack(x,y,vector,speed,path,10,0e0);                   } notify aircraft
          return;
          end;
     ctrs(cent,1)=ctrs(cent,1)+1.0e0;
     busy(cent)=busy(cent)+1;
/*central branching point*/
nxt:contrl=oper(cent);                                                      } main interpretor for translating
     if contrl=0 then                                                       } the procedure numbers to specific
          do;contrl=oper(21);                                               } function in this routine
          end;
next:ii=ctl(contrl);
     if ii<=0 | ii>10 then do;
          call ioa_("Program operation control error,branched to ^d",ii);
          return;
          end;
     contrl    =contrl+1;
     go to bran(ii);
/*1st check to see if aircraft is taking off*/
     /*takeoff clearance not given is shown as speed=0*/
     cc1:if speed>0.0e0 then                                                } take off function #1 control algorithm
          do;contrl=contrl+2;
               go to next;
          end;
```

```
cc1a:alt=18.0e3;
          speed=5.5e2*scale;
          vector=direction(x,y,dx,dy);
          call_time=ctl(contrl);
          contrl=contrl+1;
          call_in=ctl(contrl)+hold;
          contrl=contrl+1;
          totime=(time-rsettime)/3.6e3+tbase;
          ctrs(cent,2)=ctrs(cent,2)+1.0e0;
          if hold<=0.0e0 then return;
          ctrs(cent,6)=ctrs(cent,6)+1.0e0;


          ctrs(cent,7)=ctrs(cent,7)+hold;
          return;
/*takeoff clearance dependent on weather     -imposes delays*/
cc2:if speed>0.0e0 then
          do;contrl=contrl+3;
                    go to next;
          end;
          xx=iiiv(x,100);
          yy=iiiv(y,100);
          hold=0.0e0;
/*rain-15 min delay*/
          if ww(xx,yy)=1 then hold=6.0e2;
/*fog-30 min delay*/
          if ww(xx,yy)=2 then hold=1.8e3;
/*thunder-1.5hrs delay*/
          if ww(xx,yy)=3 then hold=5.4e3;
/*snow and ice*/
          if ww(xx,yy)=4 then hold=1.4e4;
          if ctl(contrl)=5 then hold=0.0e0;
          contrl=contrl+1;
          call_time=ctl(contrl);
          contrl=contrl+1;
          call_in=hold+ctl(contrl);
          enrdelay=0.0e0;
busy(cent)=busy(cent)-1;
          return;
```

take off function #2 control algorithm (delays take off as a function of weather)

```
/*next check to see if aircraft is landing*/
/*landinding clearance is weather dependent-if fuel is running low
 will divert to the nearest airport which has better weather(+2 better) and which
is within the fuel reserves of the aircraft,and between which there is
no thunder storms */
cc3:if dx^=x|dy^=y then
          do;contrl=contrl+2;
          go to next;
          end;
          xx=idiv(x,100);
          yy=idiv(y,100);
/*will the terminal accept?*/
          if tl(xx,yy)=0 then go to not_accapt;
          enrdelay  =0.0e0;
/*optional removal of weather as a factor*/
          cc3a:if ctl(contrl)=5 then go to land;
/*is weather permitting?*/
          if ww(xx,yy)=1 then enrdelay=6.0e2+enrdelay;
          if ww(xx,yy)=2 then enrdelay=1.8e3+enrdelay;
          if ww(xx,yy)=3 then enrdelay=5.4e3+enrdelay;
          if ww(xx,yy)=4 then enrdelay=1.4e4+enrdelay;
/*test for adequate fuel reserve-assumes a min of one hour for short flights*/
          resfuel=6.0e-1*((totime-tbase)*3.6e3-data(8));
          if resfuel<3.6e3 then resfuel=3.6e3;
          if enrdelay>resfuel then go to new_destination;
land:alt=0.0e0;
          contrl=contrl+1;
          call_time=ctl(contrl);
          call_in=enrdelay;
          ctrs(cent,3)=ctrs(cent,3)+1.0e0;
          if enrdelay<=0.0e0 then return;
          ctrs(cent,8)=ctrs(cent,8)+1.0e0;
          ctrs(cent,9)=ctrs(cent,9)+enrdelay;
```

landing algorithms

```
          return;
/*a terminal may delay acceptance in multiples of 5 min*/
not_accept:enrdelay=3.0e2*float(t1(xx,yy));
          go to cc3a;
/*find closest alternate terminal*/
new_destination:dist=9000;
          mptr=0;
          do kk=1 to terms(1);
/*weather must be two points better*/
          jj=kk*2;
          if x=terms(jj)&y=terms(jj+1) then go to cc3b;
          if ww(terms(jj),terms(jj+1))>=(ww(xx,yy)-1) then go to cc3b;
/*tests weather on route for thunder storms*/
  sx=terms(jj)*100;
          sy=terms(jj+1)*100;
          call etrack(x,y,sx,sy,path);
          call storm_enroute(thunder,path,where);
          if thunder=3 then go to cc3b;
          sx=terms(jj)-xx*100;
          sy=terms(jj+1)-yy*100;
          mdis=sx*sx+sy*sy;
          adis=float(mdis);
          if adis=0.0e0 then go to cc3b;
          adis=sqrt(adis);
          mdis=fixed(adis);
          if dist<mdis then go to cc3b;
          dist=mdis;
          mptr=jj;
cc3b:end;
/*no terminals near enough*/
          if mptr>0 then go to cc3c;
          enrdelay=9.5e0*resfuel;
          adis=(time-resettime)/3.6e3*tbase;
          call ioa_("Emergency landing ,aircraft %d,low fuel,time= ^f",id,adis);
          go to land;
/*destination changed-switch to enroute flight status*/
cc3c:dx=terms(mptr)*100;
          dy=terms(mptr+1)*100;
          vector=direction(x,y,dx,dy);
          contrl=contrl+1;
          ctrs(cent,10)=ctrs(cent,10)+1.0e0;
          go to next;
```

rest of landing operation algorithms

```
/* the following are the enroute operations*/
/*time division ,area navigation operation=sets x&y to new position,rerouting for weather if necessary*/
cc4:call_time=ctl(contrl);weathsw="0"b;
/*total center handover restrictions between adjacent centers is implemented*/
          contrl=contrl+1;
          vr=vector;
          vector=direction(x,y,dx,dy);
/*calculate next flight increment*/
          cc4a:dvr=ctl(contrl);
          contrl=contrl+1;
          call  vtrack(x,y,vector,speed,path,dvr);
/*check for thunder storms enroute*/
          do jj=1,path(1);
          kk=jj*2;
/*test weather*/
          if wv(path(kk),path(kk+1))=3 then go to change;
/*test for new center encountered*/


          if cent=21 then go to cxc;
          if aa(path(kk),path(kk+1))^=cent then go to new_ctr;
          cxc:end;
/*no problems,continue on course*/
          call_in=dvr;
          return;
/*weather obstruction,choose direction of vector with 50/50 p of left or right,unless if aircraft off
 course then adjust back if weather permits*/
change:if weathsw then go to changva;
          weathsw="1"b;
          adis=1.0e1;
          if (vr-vector)<0.0e0 then do;
          adis=-10.0e0;
          go to changva;
          end;
          call random_$uniform(tvr);
          if (vr-vector)=0.0e0&tvr>5.0e-1 then adis=-10.0e0;
changva:vector=vector+adis;
          contrl=contrl-1;
          ctrs(cent,11)=ctrs(cent,11)+1.0e0;
          go to cc4a;
```

en route operational algorithms

```
/*find if restrictions prohibit handover (ho array)*/
new_ctr:jj=aa(path(kk),path(kk+1));
        tvr=ho(cent,jj);
/*time of last handover*/
        ftime=ho_last(cent,jj);
/*determine new call_in time*/
        if path(1)=1 then  go to new_ctrb;
new_ctra:if kk=path(1)*2 then go to new_ctrb;
        if path(1)=1 then go to new_ctrb;
        path(1)=path(1)-1;
        if speed=0.0e0 then do;
                erx:call ioa_("speed zero error in enroute operation");
                call_in=60.0e0;
                return;
                end;
        dvr=dvr=1.0e0/speed+5.0e0;
        go to new_ctra;
/*hold or handover */
new_ctrb:if (time+dvr-ftime)<tvr then go to holdar;
new_ctrc:ctrs(cent,4)=ctrs(cent,4)+1.0e0;
        ctrs(cent,5)=ctrs(cent,5)+1.0e0;
        call_in=dvr;
        ho_last(cent,jj)=time+dvr;
        return;
new_ctrd:x=path(kk)*100+99;
        y=path(kk+1)*100+99;
        return;
holdar:if path(1)=1 then go to hdr;
/*hold aircraft for required time*/
holdera:if speed =0.0e0 then go to erx;
        dvr=dvr=1.0e0/speed;
        if path(1)=1 then go to hdr;
        path(1)=path(1)-1;
        kk=path(1)*2;
        if jj=aa(path(kk),path(kk+1)) then go to holdera;
hdr:enrdelay=ftime+tvr-time-dvr;
        if enrdelay<0.0e0 then enrdelay=0.0e0;
        call_in =enrdelay+dvr;
        ctrs(cent,12)=ctrs(cent,12)+1.0e0;
        ctrs(cent,13)=ctrs(cent,13)+enrdelay;
        go to new_ctrd;
/*dummy labels,not used yet*/
cc5:cc6:cc7:cc8:cc9:cc10:return;
end center;
```

rest of en route operational algorithms

end "center"

"direction"

```
direction:proc(x,y,dx,dy) returns(float);
dcl (x,y,dx,dy,dlx,dly)fixed,angl float;
          dly=dy-y;
          dlx=dx-x;
          if dlx=0 then go to vert;
          angl=float(dly)/float(dlx);
          angl=atand(angl);
          if dlx<0 then angl=angl+1.8e2;
          angl=angl+1.0e2;
          return(angl);
vert:angl=1.9e2;
          if dly>=0 then angl=3.7e2;
          return(angl);
end direction;
```

computes aircrafts vector

"vtrack"

```
(nounderflow):vtrack:proc(x,y,vector,speed,path,ftime);
dcl etrack ext entry,path(300) fixed,(it,jt,xt,yt,dx,dy)fixed,(vct,ftime,dvct)float,ans bit(1);
          on underflow;
dcl (x,y) fixed,(speed,vector)float,(xsize,ysize)fixed ext static;
/*vector is biased by 100*/
/*note vector should updated before calling vtrack*/
          vct=vector-1.0e2;
again:dvct=sind(vct)*1.0e2*ftime*speed+0.1e0;
          yt=y+fixed(dvct);
          dvct=cosd(vct)*1.0e2*ftime*speed+0.1e0;
          xt=x+fixed(dvct);
          if xt<100|xt>xsize*100|yt<100|yt>ysize*100 then go to cut_time;
          call etrack(x,y,xt,yt,path);
          x=xt;
          y=yt;
          return;
cut_time:ftime=ftime/2.0e0;
          go to again;
end vtrack;
```

computes next position and gets path (list of grid points)

"etrack"

```
etrack:proc(x,y,dx,dy,path);
dcl path(300) fixed,(x,y,dx,dy,rx,rdx,ry,rdy,le,je,cx,cy,dir)fixed,(slope,constant,fy)float,
        idiv ext entry returns(fixed);
        rx=idiv(x,100);
        ry=idiv(y,100);
        rdx=idiv(dx,100);
        rdy=idiv(dy,100);
        path(1)=1;
        path(2)=rx;
        path(3)=ry;
        cx=dx-x;
        cy=dy-y;
        if cx=0 then go to vertical;
        slope=float(cy)/float(cx);
        constant=float(y)-float(x)*slope;
        dir=1;
        if rx>rdx then dir=-1;
        if rx=rdx then go to vertical;
        do le=2 to 149;
        je=le*2;
        rx=rx+1*dir;
        fy=float(rx)*slope*100.e0+constant*0.1e-1;
        ry=fixed(fy);
        ry=idiv(ry,100);
        path(je)=rx;
        path(je+1)=ry;
        path(1)=path(1)+1;
        if rx=rdx then return;
        end;
        errortcall ioa_("track too long for path array=etrack subr.");
        return;
        verticalif cy=0|ry=rdy then return;
        slope=float(cx)/float(cy);
        constant=float(x)-float(y)*slope;
        dir=1;
        if ry>rdy then dir=-1;
        do le=2 to 149;
        je=le*2;
        ry=ry+1*dir;
        fy=float(ry)*slope*100e0+constant*0.1e-1;
        rx=fixed(fy);
        rx=idiv(rx,100);
        path(je)=rx;
        path(je+1)=ry;
        path(1)=path(1)+1;
        if ry=rdy then return;
        end;
        go to errort;
end etrack;
```

computes/ finds the grid inter-sections crossed by the aircraft going from one point to another

"choose terminals"

```
choose_terminals:proc(x,y,dx,dy,rn1,rn2);
dcl (x,y,dx,dy,i,j)fixed,(rn1,rn2,fl)float,terms(51) fixed ext static;
dcl t float,td fixed;
          rn1=rn1/2.0e0;
          rn2=rn2/2.0e0;
          if terms(1)=0 then return;
          td=terms(1)*2;
          fl=float(terms(1))/25.0e0;
          rn1=rn1*fl+0.1e0;
          rn2=rn2*fl+0.1e0;
          i=fixed(rn1)*2;
          if i=0 then i=2;
          if i>td then i=td;
          x=terms(i)*100;
          y=terms(i+1)*100;
          i=fixed(rn2)*2;
rep:      if i=0 then i=2;
          if i>td then i=td;
          dx=terms(i)*100;
          dy=terms(i+1)*100;
          if dx ^=x|dy ^=y then return;
          i=i+2;
          if i>td then i=0;
          go to rep;
end choose_terminals;
```

uniformly/randomly pick the origin and destination from the list of terminals

"storm en route"

```
storm_enroute:proc(thunder,path_list,where);
dcl (thunder,path_list(300),where(2),iw,jw)fixed,
          ww(100,50) fixed ext static;
          where(1)=0;
          where(2)=0;
          thunder=0;
/*thunder encountered would return 3 in thunder*/
          do iw=1 to path_list(1);
          jw=iw*2;
          if thunder>=ww(path_list(jw),path_list(jw+1))|
                    ww(path_list(jw),path_list(jw+1))>3 then go to st1;
          thunder=ww(path_list(jw),path_list(jw+1));
          where(1)=path_list(jw);
          where(2)=path_list(jw+1);
st1:      end;
          return;
end storm_enroute;
```

checks path for level of weather

"movement"

```
movement:proc(x,y,z,t);
dcl(x,y,z,t) float,ww(100,50) fixed ext static,(i,j,k,l)fixed,
          count fixed internal static;
/*this is the early version,the arguements are not being used except for t*/
          do i=1 to 99;
          do j=1 to 49;
          k=100-i;
          if k=1 then ww(k,l+1)=0;
          l=50-j;
          if l=1 then ww(k+1,l)=0;
          ww(k+1,l+1)=ww(k,l);
          end;
          end;
          ww(1,1)=0;
          t=3.6e0;
          return;
end movement;
```

increments entire weather map one grid increment NE (every hour)

"idiv"

```
idiv:proc(ivalue,idvs) returns(fixed);
dcl ivalue fixed,idvs fixed,nvalue float,rvalue fixed;
          if idvs=0 then do;nvalue=ivalue;return(ivalue);end;
          nvalue=float(ivalue)/float(idvs)+0.1e-1;
          rvalue=fixed(nvalue);
          return(rvalue);
end idiv;
```

an integer divide and zero check utility program

```
finintrupt:proc(filetime,last_id,lscale,lxsize,lysize,pvh,ac_intrupt);
        dcl aa(100,50) fixed ext static,vv(100,50) fixed ext static,tl(100,50) fixed ext static,
                scale float ext static,xsize fixed ext static,ysize fixed ext static,
                avail(21) fixed ext static,busy(21) fixed ext static,ctrs(21,21) float ext static,
                oper(21) fixed ext static,ctl(300) fixed ext static,ho(21,21) float ext static,
                ho_last(21,21) float ext static,terms(51) fixed ext static;
        dcl interupttime float,last_id fixed,cx(4) fixed,cy(4) fixed,filetime float,pvh fixed,
                resp char(1),word char(10),word1 char(10),word2 char(10),fword1 float,fword2 float,
                dword fixed,dword1 fixed,dword2 fixed,i fixed,j fixed, k fixed, l fixed, m fixed,
                dd fixed,ac_intrupt fixed,vh bit(1),bi bit(1),lxsize fixed,lysize fixed,lscale float;
dcl iq fixed,(j1,j2,j3,j4,j5,j6,j7,j8,j9,j10)fixed;
/*pass local arguements to external parameters*/
        xsize=lxsize;
        ysize=lysize;
        scale=lscale;
        onc:call ioa_("Write on restart'1' or restart'2'?");
        call read_list_(iq);
        j=0;
        if iq=1 then
                do;call fo("restart1");
                go to i1;
        end;
        if iq=2 then
                do;call fo("restart2");
                go to i1;
        end;
        call ioa_("No file?");
        j=j+1;
        if j>2 then return;
        go to onc;
i1:call ioa_("^/^f^/^d ^d ^/^f ^d",filetime,lxsize,lysize,lscale,last_id);
        do i=1 to 21;
        call ioa_("^d ^d ^d ^d",i,avail(i),busy(i),oper(i));
        end;
        do i=1 to 3;
        j=i*7-6;
        j1=j+1;
        j2=j+2;
        j3=j+3;
        j4=j +4;
        j5=j+5;
        j6=j+6;
        do m=1 to 3;
        call ioa_("    ^d    ^d    ^d    ^d    ^d    ^d    ^d",j,j1,j2,j3,j4,j5,j6);
        do k=1 to 21;
        if m=1 then
call ioa_("^d ^f ^f ^f ^f ^f ^f ^f",k,ctrs(k,j),ctrs(k,j1),ctrs(k,j2),ctrs(k,j3),ctrs(k,j4),
        ctrs(k,j5),ctrs(k,j6));
```

performs the Center and maps actual data recording when interruption has occurred

```
            if m=2 then
call ioa_(""d ^f ^f ^f ^f ^f ^f ^f",k,ho(k,j),ho(k,j1),ho(k,j2),ho(k,j3),ho(k,j4),
            ho(k,j5),ho(k,j6));
            if m=3 then

call ioa_(""d ^f ^f ^f ^f ^f ^f ^f",k,ho_last(k,j),ho_last(k,j1),ho_last(k,j2),ho_last(k,j3),ho_last(k,j4),
            ho_last(k,j5),ho_last(k,j6));
            end;
            end;
            end;
            do i=1 to 50;
            j=i*6-5;
            call ioa_(""d ^f ^f ^f ^f ^f ^f",j,ctl(j),ctl(j+1),ctl(j+2),ctl(j+3),ctl(j+4),ctl(j+5));
            end;
            do i=1 to 10;
            j=i*10-9;
            j1=j+1;
            j2=j+2;
            j3=j+3;
            j4=j+4;
            j5=j +5;
            j6=j+6;
            j7=j+7;
            j8=j+8;
            j9=j+9;
            do m=1 to 3;
            call ioa_("      ^d   ^d   ^d   ^d   ^d   ^d   ^d   ^d   ^d   ^d",j,j1,j2,j3,j4,j5,j6,j7,j8,j9);
            do k=1 to 50;
            l=51-k;
            if m=1 then call ioa_(""d ^d ^d ^d ^d ^d ^d ^d ^d ^d ^d",l,aa(j,l),aa(j1,l),aa(j2,l),
            aa(j3,l),aa(j4,l),aa(j5,l),aa(j6,l),aa(j7,l),aa(j8,l),aa(j9,l));
            if m=2 then
            call ioa_(""d ^d ^d ^d ^d ^d ^d ^d ^d ^d ^d",l,vv(j,l),vv(j1,l),vv(j2,l),vv(j3,l),vv(j4,l),vv(j5,l),
            vv(j6,l),vv(j7,l),vv(j8,l),vv(j9,l));
            if m=3 then
            call ioa_(""d ^d ^d ^d ^d ^d ^d ^d ^d ^d ^d",l,tl(j,l),tl(j1,l),tl(j2,l),tl(j3,l),tl(j4,l),tl(j5,l),
            tl(j6,l),tl(j7,l),tl(j8,l),tl(j9,l));
            end;
            end;
            end;
            call ioa_(""d",terms(1));
            do i=2 to 50 by 2;
            call ioa_(""d ^d",terms(i),terms(i+1));
            end;
            call ioa_(""d",ac_intrupt);
            call co;
            return;
end finintrupt;
```

```
coutput:proc(t);
dcl (t,peak(21,2),sums(21),av(21))float,(i,j,k,m,li(22))fixed,
        ctrs(21,21) float ext static,a(16) char(16) initial
        ("com sesions","a/c origins","a/c termins","a/c exits",
        "a/c entered","a/c del/TOFF","bl","a/c del landg","bl","a/c destin chg","a/c weath-rerut",
        "a/c ctr restr","bl","avg Toff dely","avg landg del","avg enr delay");
        do i=1 to 21;
        sums(i)=0.0e0;
                li(i)=0;
        av(i)=0.0e0;
        peak(i,1)=0.0e0;
        peak(i,2)=0.0e0;
        end;
        m=0;
        li(22)=0;
        call ioa_("ARTCC PROGRESS REPORT"/      Time= "f",t);
c2:do i=1 to 20;
        if ctrs(i,1)=0.0e0 then go to c4;
        m=m+1;
        if ctrs(i,6) ^= 0.0e0 then ctrs(i,14)=ctrs(i,7)/ctrs(i,6);
        if ctrs(i,8) ^= 0.0e0 then ctrs(i,15)=ctrs(i,9)/ctrs(i,8);
        if ctrs(i,12) ^= 0.0e0 then ctrs(i,16)=ctrs(i,13)/ctrs(i,12);
        do j=1 to 16;
        sums(j)=sums(j)+ctrs(i,j);
        if ctrs(i,j)<= peak(j,2) then go to c3;
        peak(j,1)=i;
        peak(j,2)=ctrs(i,j);
c3:end;
c4:end;
        do i=1 to 21;
        if m ^=0 then av(i)=sums(i)/float(m);
        end;
        call ioa_("Delays:^2/  Enroute: ^f aircraft delayed ^f sec. avg.",sums(12),av(16));
        call ioa_("  Takeoff:  ^f aircraft delayed ^f sec. avg.",sums(6),av(14));
        call ioa_("  Landing:  ^f aircraft delayed ^f sec. avg.",sums(8),av(15));
        call ioa_("Avg number of aircraft originating in centers= ^f,terminating at centers= ^f",av(2),av(3));
        call ioa_("    entering center= ^f, and leaving center= ^f",av(5),av(6));
        call ioa_("Avg no. of aircraft vectored to different destinations= ^f",av(10));
        call ioa_("Avg no. of aircraft rerouted because of weather= ^f",av(11));
        call ioa_("^2/Peak values"/descripti center no.   peak value");
        do i=1 to 16;
        if i=7|i=9|i=13 then go to c5;
        call ioa_("^a  ^f    ^f",a(i),peak(i,1),peak (i,2));
        c5:end;
```

computes statistics and outputs reports on console

total system statistics and report

```
          i=1;
do;
c1:call ioa_("Give the center number you wish to see progress(?>21 gives all,=0 concludes list)");
          call read_list_(li(i));
          if li(i)=0 then go to c9;
          if li(i)>21 then
                    do;do j=1 to 21;
                    li(j)=j;
```

select specific Center reports

```
                              end;
                    go to c9;
                    end;
          i=i+1;
          if i<22 then go to c1;
          c9:do i=1 to 21;
          if li(i)=0 then go to c6;
          call ioa_("Center  "d  Progress report"/ description      value",i);
          do j=1 to 16;
          if j=7|j=9|j=13 then go to c7;
          call ioa_(""a    "f",a(j),ctrs(i,j));
c7:end;
c6:end;
c8:end;
end coutput;
```

output selected Center reports