REPORT NO. FAA-RD-73-122

AIRPORT INFORMATION RETRIEVAL SYSTEM (AIRS) SYSTEM SUPPORT MANUAL

Manuel F. Medeiros
Julie Sussman

OCTOBER 1973
FINAL REPORT

Prepared for:

DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION
SYSTEMS RESEARCH AND DEVELOPMENT SERVICE
Washington DC 20591

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FAA-RD-73-122		

4. Title and Subtitle	5. Report Date
AIRPORT INFORMATION RETRIEVAL SYSTEM (AIRS) SYSTEM SUPPORT MANUAL	October 1973
	6. Performing Organization Code
	8. Performing Organization Report No.
7. Author(s)	DOT-TSC-FAA-73-25
Manuel F. Medeiros and Julie Sussman	
9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Department of Transportation Transportation Systems Center Kendall Square Cambridge MA 02142	FA-406/R4111
	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Final Report
Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington DC 20591	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This handbook is a support manual for a prototype air traffic flow control automation system developed for the FAA's Systems Command Center. The system is implemented on a time-sharing computer and is designed to provide airport traffic load predictions and flow control support. The System Support Manual is designed for use by an experienced computer programmer. It contains instructions on performing the monthly AIRS data base updating including the Official Airline Guide data tape processing, the merging with the existing data base and the maintenance of the associated supporting data files. The manual describes the duties associated with monitoring nightly file checking and failsafe programs to assure data base integrity. The daily processing and troubleshooting of the system's usage records are also described. Other support functions involving data base maintenance are presented.

17. Key Words	18. Distribution Statement
Flow Control, Air Traffic Control, Central Flow Control Facility, Automation, Information Retrieval, Airport Information, Flow Control Procedures	DOCUMENT IS AVAILABLE TO THE PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22151.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	85	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

Appendices

ACRONYMS AND ABBREVIATIONS

AIRS	Airport Information Retrieval System
ARO	Airport Reservation Office
ARTCC	Air Route Traffic Control Center
CFCF	Central Flow Control Facility
CPU	Central Processing Unit
FAA	Federal Aviation Administration
GMT	Greenwich Mean Time
I/O	Input/Output
SCC	Systems Command Center
TSC	Transportation Systems Center
UFD	User File Directory

1. INTRODUCTION

AIRS (Airport Information Retrieval System) is a prototype computer program developed by the Transportation Systems Center (TSC) to aid the FAA's Systems Command Center (SCC) in air traffic flow control. Two groups within the SCC use AIRS: the Central Flow Control Facility (CFCF) uses it to find out traffic demands for airports, predict arrival delays, and compute flow controls; the Airport Reservation Office (ARO) updates the data base of flight schedules by entering and cancelling flights.

The "AIRS User's Guide" (Ref.2) tells what AIRS can do and how to use it. The "AIRS System Design" (Ref.1) explains how AIRS works. This manual contains instructions for performing certain support functions necessary for the smooth running of AIRS. These functions are shown in the table below. They all relate to creation and maintenance of the AIRS data base.

Function	Frequency
prepare new data base	monthly
truncate current data base	monthly (optional)
monitor AIRS usage	daily
check data base integrity	daily
back up data base	as needed (rare)
respond to AIRS failures	as needed (rare)

AIRS resides on a time-shared PDP-10 computer at First Data Corporation in Waltham, Mass. Very little knowledge of either AIRS or the PDP-10 is required to carry out the support functions under normal conditions. Basic information on how to use the computer can be found in the "Terminal User's Guide" (Ref.8). Before attempting to use the computer you should know:

how to log in (LOG);
how to log off (KJOB);
how to correct typing errors (rubout, control U);
how to interrupt a program (control C).

Appendix C contains a list of all monitor (time-sharing system) commands and system programs used in this manual, with references to their explanation in the "Terminal User's Guide" (Ref.8) and other documentation. For normal

operation, very little knowledge besides what is contained in this manual is needed.

The AIRS project has two different user numbers:
[57041,1161] - used by TSC - AIRS development and data base;
[57041,1162] - used by SCC - operational use.
You will need the passwords for both, as there are support functions done under both numbers.

All files for the AIRS project are contained on a private disk pack, called TSCA.

In this manual, underlining will be used to indicate commands or inputs typed by the user.

2. DATA BASE CREATION

2.1 INTRODUCTION

Every month a tape containing the next month's flight schedules is received. The tape must be processed and the schedules combined with still-effective schedules from the current AIRS data base to form a new AIRS data base. The data base creation process has 5 stages, described in the remaining 5 sections of this chapter as follows:

Stage 1 - Section 2.2
Should be done as soon as possible.
Tape is read. Relevant schedules are partially processed and stored on disk.

Stage 2 - Section 2.3
May be done any time between stage 1 and stage 3.
New codes are added to AIRS definition files.

Stage 3 - Section 2.4
May be done any time before current data base expires. Allow some leeway.
New dictionary is formed. New schedules are encoded, sorted, and merged with current schedules. Indexes are formed.

Stage 4 - Section 2.5
Must be done immediately after stage 3.
New data base replaces current one.

Stage 5 - Section 2.6
Should not be done until at least the day after stage 4.
Intermediate files are deleted.

All stages of data base creation are done while logged in under 57041,1161.

2.2 SCHEDULE TAPE

The primary source of flight schedules is a magnetic tape produced for the FAA by the Reuben H. Donnelley Corp., publishers of the Official Airline Guide. The first step in preparing the data base is to process this tape, transferring the relevant schedules to a disk file, on which the subsequent processing (2.3) is performed. This section covers the procedures for handling (section 2.2.1) and processing (section 2.2.2) the tape.

2.2.1 HANDLING THE TAPE

Each month, the tape is mailed to TSC, attention M.Medeiros. It is usually received around the middle of the month.

The tape should be taken out to First Data (see Appendix D for address) and the processing in section 2.2.2 done as soon as possible. Place the number "S31/M64" (ID number assigned by First Data) on the tape and take it to First Data with instructions to store it as numbered.

Keep the enclosed sheets of information on code changes.

When processing is completed, pick up the tape from First Data and send it to the address listed in Appendix D. Remove the "S31/M64" label and save it to use next month.

Our contact at Reuben Donnelley is listed in Appendix D, in case a new copy of the tape is required, etc.

TAPES AND THE PDP-10

This section describes the basic aspects of using the tape on the computer: mounting, rewinding, and dismounting it. These will be referred to from the processing description (2.2.2).

MOUNTING THE TAPE

Use the PLEASE command to ask the operator to mount the tape. You must tell him the tape number, that you need a 9 track drive, and that you want it write-locked. For example:

.PLEASE MOUNT TAPE S31/M64 9 TRACK WRITE-LOCK

(you may abbreviate TRACK to TRK and WRITE-LOCK to WRLK)

The operator will either:

1. Tell you no drive is available. You'll have to try again later.

or 2. Mount the tape. When it's ready, he'll tell you the tape drive number - for example, MTA2. You'll need this number to refer to the tape. In these instructions, "n" is used to represent the tape drive number - that is, we say "MTAn".

If the operator doesn't terminate the conversation ("." typed), do so by typing control C.

REWINDING THE TAPE

At certain stages in the processing you will have to rewind the tape. To do this, use the system program PIP, as follows.

```
.R PIP              [computer typed ".", you typed rest]
*MTAn:(MW)←         [computer typed "*", you typed rest]
*↑C                 [computer typed "*", you typed control C]
.
```

[Note: The characters "_" and "←" are equivalent - use whichever appears on your keyboard.]

DISMOUNTING THE TAPE

When done using the tape, notify the operator via the PLEASE command.
For example: .PLEASE I'M DONE WITH THE TAPE

The operator will dismount it and put it away.

Give up your claim to the tape drive by typing the command:

```
.REAS MTAn 1
```

2.2.2 PROCESSING THE TAPE

The tape should be processed as soon as possible after it is received. Then if there is anything seriously wrong with it, there is still time to get a new one. There are two steps in processing the tape: checking the tape for errors, and processing the data. The tape processing can take under half an hour if the system is lightly loaded or close to an hour if it is heavily loaded.

2.2.2.1 CHECKING THE TAPE FOR ERRORS

This check has nothing to do with the content of the tape, but simply its readability. The tape drive may be in need of maintenance or may simply be slightly incompatible with the tape. Or the tape itself may be unreadable.

The procedure is as follows:

1. Have the operator mount the tape (see 2.2.1).

2. Run the FILDMP program to check the tape.

PERFECT CASE SCENARIO

```
.R FILDMP
*TTY:←MTAn:(MI)(J#300W)/S
  BLOCK# 1,     SIZE=20 WORDS (36-BIT)
  BLOCK# 4      *** E O F ***

*TTY:←MTAn:(MI)(J#300W)/S
  BLOCK# 1,     SIZE=265 WORDS (36-BIT)
  BLOCK# m,     SIZE= w  WORDS (36-BIT)
  BLOCK# m+1    *** E O F ***

*↑C
.
```

(In the above, m is a number around 6000, and w is a rounded up multiple of 26.5 - for example 80.)

[Note: The characters "_" and "←" are equivalent - use whichever appears on your keyboard.]

WHO TYPES WHAT

In the above scenario:

You type R FILDMP to start the program.

When FILDMP is ready for a command, it types *. The characters after the *'s are commands you type to FILDMP. N.B. Colons, etc. are essential!! In the command, MTAn is whatever tape number the operator assigned you when he mounted the tape. For example, MTA2. Control C (↑C) is used to exit from FILDMP. The other lines, beginning with BLOCK, are output from FILDMP.

MEANING OF OUTPUT

FILDMP, when given the command shown above, reads a file from the tape, one block at a time. If the block size or error status changed from the previous block, it prints them out. Otherwise there is no output. It also indicates the end of file (EOF). Thus if the tape is error-free, the output should just be: the size of the first block; the size of the last block (if partial); the end of file. Since our tape has two files, this sequence happens twice. Note that if there is an error in a single block, the next block is also typed since its status is changed from the previous (erroneous) block.

NO ERRORS ON TAPE

If the tape is OK, the output should be as in the SCENARIO above. Except where the variables m and w appear (the length of the second file varies, and the amount of the last block that is filled varies), it should match exactly - form of first file, number of words per block in second file, etc. If there are no errors, go on to 2.2.2.2.

ERRORS ON TAPE

An error on the tape will show up either as a different block size (size not 265, except for last block) or an error code signalling a parity error or some other problem. The error code will appear after the size. See below for how to handle tape errors.

OTHER ERRORS

After you type the first command to FILDMP, if you get the message:

/OPEN/LOOKUP/ENTER FAILURE

You either mistyped the command, or the operator neglected to assign the tape drive to you. If you are sure you typed it right (try again, to be sure), exit FILDMP by typing control C, then notify the operator via the PLEASE command. For example,

.PLEASE YOU FORGOT TO ASSIGN MTAn TO ME

(of course, MTAn is the number the operator gave you earlier).

When the operator has fixed this, FILDMP should work OK.

HANDLING TAPE ERRORS

If there are no errors, go on to 2.2.2.2. Otherwise:

1st Attempt
See if the errors are repeatable.

Rewind the tape (as described in 2.2.1).

Try FILDMP again.
If it is still giving errors, you needn't let it run to completion, but may interrupt it by typing two control Cs. Then go to 2nd attempt. Otherwise (if no errors), go on to 2.2.2.2.

2nd Attempt
Try the other tape drive.

Via the PLEASE command, ask the operator to move the tape to the other drive.

If the other drive is not available, tell the operator to dismount the tape, and try again later. Find out from the operator what physical drive you were using, so you can ask for the other when you log in again. (MTAn is the logical, not physical name. It may be assigned to different drives at different times. Thus later, when you want a particular physical drive, you must specify it in your request for mounting the tape. For example, drive "A".)

If (or when) you do get the other drive, try FILDMP again (note, you may have a different MTA number).

If the tape is OK on this drive, go on to 2.2.2.2.

If the tape is still unreadable, have the operator dismount it (as in 2.2.1), and try 3rd attempt.

3rd Attempt
Get help from First Data

Call up First Data (ask for technical assistance - for phone number see Appendix D). They should be able to either fix the tape drives (if that is the problem) or make a readable copy of the tape. This may take till the next day. As soon as possible, redo the tape checking. (If they make a copy, it will have a new ID number, not S31/M64 - be sure you ask for the new tape by using the new number).

If the copy checks out, go on to 2.2.2.2.

4th Attempt
Get a new tape.

If First Data can't make a readable copy, the original tape may really be bad. You may need to contact Reuben Donnelley Corp. (see 2.2.1) about getting a new copy immediately. Use your judgment, however. If there are only a couple of bad blocks, you may as well process the tape as is; the bad blocks will simply be rejected during processing. Just be sure you read it on the tape drive that gave the least trouble.

2.2.2.2 PROCESSING THE DATA

Step 1
If the tape is already mounted (from doing the error checking), rewind it (as described in 2.2.1).

If the tape is not mounted, have the operator mount it (as described in 2.2.1).

Step 2
Type the command:

.ASSIGN MTAn TAPE

where MTAn is the tape number given when the tape was mounted.

Response:
normal: MTAn ASSIGNED
go to step 3

abnormal: ? ALREADY ASSIGNED TO JOB x
Try again to be sure you typed it right.
If the error persists, probably the operator forgot to assign the tape drive to you when he mounted the tape. Tell him, so he can correct it. Example:
.PLEASE YOU FORGOT TO ASSIGN MTAn TO ME

Then after he assigns it, repeat the ASSIGN command.

Step 3
Type the command:

.SET BLOCKSIZE MTAn 300

Response: none

Step 4
Run the processing program (RUBINE) as follows. We will first give the perfect error-free case, then go over the various error situations.

PERFECT CASE

you type	RUN RUBINE
computer types	START OR RESTART?
you type	S

computer types TAPE EFFECTIVE AUG01 1973
telling you the effective date of the tape (in this example, AUG 1)

computer types EFFECTIVE DATES=

You give start and end dates in "/" notation, separated by a space. For the AUG01 example, you would type:

8/1 8/31

!!BE SURE YOU TYPE THE RIGHT DATES (SEE FOLLOWING)!!

The dates are determined as follows.
Each tape contains one month of data. Except for the April, May, October, and November tapes, the effective date typed by the computer (as read from the tape) should be the first of the month, and you should type the first and last days of the month. The April tape (effective APR01), however, goes only through the last day before daylight savings time starts, and the May tape picks up where the April one left off.
Similarly, the October tape ends on the last day of daylight savings time and the November tape picks up where it left off. (Be careful to type this correctly. Watch out for leap years, "30 days hath September", etc. If you make a mistake, interrupt the program by typing two control C's and start over.)

The program requires no more input. During the tape processing it types the following messages.

a. "STATE SAVED" messages:
STATE SAVED ON SAVE.1
STATE SAVED ON SAVE.2
These two are typed alternately, about 20-25 messages in all.

b. new codes

Three types of messages may be interspersed with the "STATE SAVED" messages.

new airports: 3-letter code plus lat/long
xxx nnnnn mmmmm

new airlines: 2-letter codes
AIRLINE xx

new aircraft types: all start with J, P, or T

TYPE xxxx

c. statistics

Finally, statistics on the schedules are typed out. The totals given for airlines, airports, and aircraft types are cumulative, and will thus be the previous month's value plus the number of new ones typed out this time, if any.

If there are more than 2047 airports or more than 127 airlines, finish this stage (section 2.2) but do not go on to the next (section 2.3). The AIRS data base cannot accommodate more than these numbers, so an AIRS expert must do something about the situation before the data base creation can proceed.

After typing the statistics, RUBINE exits. You should go on to step 5.

DESCRIPTION

FILES
reads (and may update) MONTH.TAB
updates ACFT.RD,AIRLN.RD,AIRPT.RD
creates FLIGHT.RD,AIRBN.RD,SAVE.1,SAVE.2
may create SAVE.ERR

Reads schedule tape, rejecting irrelevant schedules and erroneous lines and blocks. Schedules retained are packed and output on FLIGHT.RD, 7 words per flight. After every 1000 flights output, state of program is saved to permit backing up processing after a crash or error. Approximate airborne flight counts are written on AIRBN.RD. New codes for aircraft types, airlines, and airports are appended to the existing code files (ACFT.RD, AIRLN.RD, AIRPT.RD).

ERROR MESSAGES

FATAL ERRORS - program exits

CANNOT OPEN FBREAD DEVICE

(typed before "START OR RESTART?")
You forgot to ASSIGN the tape.
Go back to step 2 of the data processing.

IFILE FAILED TO OPEN FILE MONTH.TAB FOR READING

See same error message in chapter 7, AIRS Errors, under fatal errors, then start RUBINE again.

OFILE FAILED TO OPEN FILE MONTH.TAB FOR WRITING

See same error message in chapter 7, AIRS Errors, under fatal errors, then start RUBINE again.

TABLE FULL

The hash table used to store airline, airport, and aircraft type codes has overflowed. An AIRS programmer must fix the program.

RAX ERROR #n ON UNIT u
STATE SAVED ON SAVE.ERR

See Appendix A for explanation of RAX error codes.
See Appendix B for what file is on that unit in RUBINE.

NON-FATAL ERRORS - program continues

DATELINE

meaning: A flight's local time is 12 hours offset from GMT. AIRS cannot know whether it is ahead or behind. It assumes the local and GMT dates are the same.

action: none

BAD LINE n
line...

Meaning: Types out number of bad line (position on tape) and contents of line. A line contains the fields listed below. Only some of these fields are checked for erroneous data. These are marked in the following table with a "*". If no other error message appears with the "BAD LINE" message, it means that one of the marked data fields was bad. The line is rejected, and processing continues.

Origin - 3-letter code
*Planned time of departure, local time - 4 digits
*Planned time of departure, GMT - 4 digits
Destination - 3-letter code
*Domestic/International carrier flag - 1-digit, 0-4
*Planned time of arrival, GMT - 4 digits
Aircraft type
Airline - 2 letters
Flight ID - 5 characters
Service class
*Departure day flags - 7 digits, 0 or 1.
Carrier code - blank, I, or T
*Estimated time enroute, in minutes - 4 digits
*Effective date - 4 digits - usually blank
*Discontinued date - 4 digits - usually blank
Latitude and longitude of origin and destination - four 5-digit numbers

Action: You may check whether the error is spurious by interrupting the program (type control C twice) and backing it up to the last saved state (see "RESTARTING THE PROGRAM" below). If the error doesn't repeat, you're in luck. If it does, ignore it. Losing a few flights is not tragic. But if very many lines are rejected (use your judgment) you should:
(a) If you are not sure you have the same tape drive that FILDMP successfully read the tape on (section 2.2.2.1) try the other drive, as described for "2nd Attempt" under "HANDLING TAPE ERRORS" in 2.2.2.1. You may "RESTART" RUBINE as mentioned above.
(b) If the tape is on a drive on which FILDMP reads it reasonably but RUBINE gives many of these errors, get a new tape from Reuben Donnelley Corp. (see 2.2.1).

TAPE INPUT NOT INTEGRAL NUMBER OF RECORDS
BAD LINE n
line...

Meaning: Block on tape had wrong number of words.
First line in block is typed out (format as for BAD LINE message), though it is not necessarily bad.
Whole block is rejected.

Action: If FILDMP (section 2.2.2.1) found blocks with wrong numbers of words, they are rejected here; just ignore this. But if FILDMP didn't, or more blocks are rejected than FILDMP complained about, interrupt the program (type control C twice). You can try backing up the program to see if the error repeats (see "RESTARTING THE PROGRAM" below), and may have to try the other tape drive, get help from First Data, or get a new tape as described under "HANDLING TAPE ERRORS" in 2.2.2.1.

TAPE I/O ERROR - K=n
BAD LINE n
line...

"TAPE I/O ERROR..." may also appear without BAD LINE message.

Meaning: System had some sort of trouble reading tape block. Value of K represents mag tape status bits 18-21, as described in section 5.5.5 of "Monitor Calls" in Ref. 3.
First line in block may be typed out (as above) though it is not necessarily bad.
Whole block is rejected.

Action: K=1 means tape block too large. If it happens at the beginning of RUBINE, probably you forgot to set the buffer size for reading the tape. Interrupt the program (type two control Cs) and go back to step 3 of the data processing.

If FILDMP (2.2.2.1) found bad blocks, they are rejected here; just ignore this. But if FILDMP didn't, or if more blocks are rejected than FILDMP complained about, act as described above for previous error message.

COMPUTER CRASHES

If the computer crashes, you can continue RUBINE another time from where it left off. To do so, perform steps 1-3 as usual. Then resume RUBINE as described below under "RESTARTING THE PROGRAM".

RESTARTING THE PROGRAM

The processing can be backed up to an earlier state to repeat some of it or continue from where the run is interrupted, as described above (ERROR MESSAGES and COMPUTER CRASHES). It can restart from either of the last two saved states. If for some reason you need to back it up farther than that, you'll have to simply start the program over as if it had never been run. The only difference from the initial run will be that new codes reported the first time may not be reported again.

To back up the program, first be sure the tape is mounted, rewound, etc. as in steps 1-3 of the data processing. Then run RUBINE as before (step 4) with the following differences:

(1) Answer "R" to "START OR RESTART".

(2) Instead of talking about effective dates, the computer will ask: "SAVE FILE 1 OR 2?"
In the interrupted run you are resuming, decide which point you want to back up to. The last time the state was saved on SAVE.1, or the last time it was saved on SAVE.2.
Answer "1" to pick up from the last SAVE.1.
Answer "2" to pick up from the last SAVE.2.

(3) The program will reposition the tape to the appropriate place (it may take a while), then do one of the following.

"TAPE REPOSITIONED"

Self-explanatory. The processing continues as usual.

"TAPE ADVANCED TO line..."
"INSTEAD OF line..."

Tape not correctly positioned.
Format of lines is same as in "BAD LINE" message above.
Program exits.
You should rewind the tape (as in 2.2.1) and try again.

Step 5
When RUBINE exits, dismount the tape (as in 2.2.1).

The tape processing is now complete.

Step 6
RUBINE creates a file containing approximate counts of airborne flights. Type and delete this file as follows.

```
.TYPE AIRBN.RD
```

(the file is typed - it is about 1 page long)

```
.DELETE AIRBN.RD
```

(the file is deleted)

Label the output with the month and year of the tape just processed and compare airborne counts with last month's. If no drastic differences (such as double the number of flights) appear, put it away with the data from other months. If there are drastic differences, then someone familiar with the RUBINE program and the data tape should try to evaluate whether the change is reasonable or whether something is wrong.

2.3 EDITING DEFINITION FILES

If no new airports or aircraft types were reported by RUBINE, go on to the next stage (2.4).

Otherwise the following definition files must be edited.

> New aircraft types must be put in definition of J(jet), P(prop), or T(turbo-prop) in the file CLASS.TAB, according to their first letter.
>
> New domestic airports must be added to the appropriate center definitions in the file CENTR.TAB.
>
> New foreign airports must be added to the appropriate foreign area definitions in the file FORIN.TAB.
>
> (A fourth definition file, TAXI.TAB, defines the group "TAXI" to include all commuter air carriers. All such airlines were typed in once, from the list in Ref.6, regardless of whether they had shown up in RUBINE. This file can also be updated as appropriate to reflect airline code changes.)

Find out the air traffic control center (ARTCC) or foreign area each new airport is in, as follows. See if the airport code is defined on the code sheet that came with the tape. This will give you the name of the city. If it isn't there, you can look it up either in the "Official Airline Guide" (Refs.6&7 - try the international guide if it isn't in the domestic guide) under "City (Airport) Codes", or in the FAA's code book "Location Identifiers" (Ref.5 - again, there are separate domestic and international sections). If the city is domestic, look it up in section A of the "Location Identifiers" (Ref.5) to get its air traffic control center ("Cntr" column - all start with Z). If it is international, assign it one of the following.

ZEUR - Europe
ZPAC - Pacific area
ZSOA - South America
ZSAM - South America and Mexico
ZSAB - South Atlantic and Bermuda
CYQM - Montreal
CYYZ - Toronto
CYWG - Winnipeg
CYVR - Vancouver
CYEG - Edmonton
CYUL - Montreal

If you can't find an airport's code, you can use its latitude and longitude together with a map of the ARTCCs to

locate it. The lat/long typed out by RUBINE are in 10s of seconds, and the latitude has 90 degrees added to it so that the South Pole is 0. There is no distinction made between East and West longitude.

If you absolutely can't locate some airport, give up and continue. It isn't critical.

To do the editing you must know how to use one of the text editors. COED (see Appendix C) is recommended.

Each definition file has the form:

```
first group name
number of items
first item
    .
    .
    .
last item
second group name
number of items
    .
    .
    .
```

Thus to add an item, you must:

locate the group name

go to the next line, the number of items

change the number to incorporate the one(s) you are adding to the group

insert the new group members after the number, one per line, left-justified

Note: These are text files which may be typed out (example: TYPE CLASS.TAB) if desired. If for any reason one is destroyed, it may be backed up (see chapter 6). N.B. The order of centers in the CENTR.TAB must never be altered.

Check that all airports have been assigned to a center or foreign group as follows.

```
.RUN CNTCHK
```

The CNTCHK program reads the file of airport names (AIRPT.RD) and the center and foreign area definition files (CENTR.TAB, FORIN.TAB). It reports any unknown, unassigned,

and multiply assigned terminals as follows.

Note: Strange errors consisting of messages listed below and/or messages starting with ? and aborting the program can result if there is an inconsistency in the number of items in a group definition - i.e. if the number after the group name does not match the number of items listed. Action: You must have made an error editing CENTR.TAB or FORIN.TAB. Go back and fix it. Then repeat CNTCHK.

UNKNOWN TERMINAL xxx

The given code is in the definition of a center or foreign area but it is not known to be an airport (does not appear in AIRPT.RD). You probably made a typo when editing the definition files. Correct it and try running CNTCHK again.

xxx ASSIGNED TO BOTH yyy AND zzz

The given terminal (xxx) is assigned to more than one center or foreign area (yyy and zzz). You probably made a mistake when editing the definition files. Correct it and try running CNTCHK again. You should not assign a terminal to more than one group on purpose.

UNASSIGNED TERMINALS:

Airports not assigned to any center or foreign area are listed. If you accidentally omitted to assign the terminal, re-edit the definition files and try running CNTCHK again. If you purposely omitted to assign the terminal (you don't know where it belongs, or for some reason don't want it included in any of the areas), you may ignore this message.

2.4 CREATING NEW DATA BASE

2.4.1 GENERAL INFORMATION

WHEN TO RUN

This stage can finish in a little over an hour if it is run at a time when the computer is not busy, and can take considerably longer (perhaps double) if the system is heavily loaded.

You should not start this stage unless there is time for it to finish. Also allow an extra 15 minutes or so for the next stage, which must follow immediately.

It is best to run the programs when the system is not busy. This will minimize the effects on AIRS users (see below).

The run should not be allowed to overlap the nightly batch job (chapter 5), which is normally scheduled for 11 p.m. Chapter 5.2.2 tells how to check when the job is scheduled.

EFFECT ON AIRS USERS

About one third of the way through this stage, the current AIRS schedule file is frozen (write-protected) so that it can't be modified. From this point until the new data base is made available (section 2.5), no ARO entries can be made.

HOW TO RUN

First prepare disk space by typing

.RUN PREPAR

This program simply deletes an old copy of the AIRS data base (the files SCHED2.CRF, TERM2.CRF, CENT2.CRF, TERM2.FRE, and CENT2.FRE).

The rest of this stage consists of a series of 10 programs. You start the first one running, then each runs the next automatically. Only the first one requires input from you.

To start the first program (which is named "FILES"), type the command:

.RUN FILES

The sequence of programs will be run.

Each program types:
     **ENTER (program name)   when it starts
 and **EXIT (program name)   when it finishes.
For all but the ninth program, FREFIL, the CPU and elapsed times are also reported.

If it is ever necessary to start the sequence at a program other than the first (as will be described for error recovery), type the command:

.RUN name

where "name" is the name of the program to be started.

Section 2.4.2 describes each program, explaining any inputs and outputs and any error messages produced by the program.

Error conditions and recovery procedures which apply to all programs in the sequence are described in section 2.4.3.

2.4.2 PROGRAM DESCRIPTIONS

The 10 programs in the automatic sequence will now be described. The following information will be given for each.

- -the approximate run time (elapsed time)
- -the normal (perfect case) input/output
- -error conditions recognized by the program, if any
- -brief description of what it's doing

For normal processing, you need not be familiar with the error conditions or description. This information is provided for the rare case when trouble arises. The descriptions may be useful in analyzing any weird problems not covered by the error conditions.

For ease of use, the normal case input and output is given at the left margin. Explanations and the error and descriptive information are indented.

Program #1: FILES

Run time: about 1-2 minutes

TERMINAL AND CENTER GROUPS IN FILE=FORIN.TAB

AIRCRAFT TYPE GROUPS IN FILE=CLASS.TAB

AIRLINE GROUPS IN FILE=TAXI.TAB

The file names you type are the definition files you edited in section 2.3.
If these are ever renamed, use the right name.
The file name should be at most 5 letters long.
If one of the definition files is ever eliminated, type NONE instead of a file name.

DESCRIPTION

files read: ACFT.RD,AIRLN.RD,AIRPT.RD
CENTR.TAB
FORIN.TAB,CLASS.TAB,TAXI.TAB
files created: TABLS.RD, TABL2.RD

Forms name tables, dictionary, and group definition tables for AIRS. The tables become current (replace the old ones and are used by

AIRS) immediately, not when the rest of the data base is handed over (section 2.5). (Tables are first created on files TABLS.RD2, TABL2.RD2; then current tables are saved as TABLS.OLD, TABL2.OLD; then new files are renamed to be current.)

ERRORS

Note: Strange errors consisting of messages listed below and/or messages starting with ? and aborting the program can result if there is an inconsistency in the number of items in a group definition - i.e. if the number after the group name does not match the number of items listed (see group definition file format in 2.3).
Action: You must have made an error editing the file being processed in 2.3. Go back and fix it. Then come back to 2.4.

(1) File name errors

BLANK - RETYPE
You entered a blank file name. Perhaps you hit the carriage return by accident. Just type the name again.

FILENAME CANT BE 6 CHARS - RETYPE
You typed a file name having more than 5 characters before the dot.

filename NOT FOUND
The given file could not be found. The request for the file name will be repeated.

(2) group file content errors

UNKNOWN PLACE xxxxx
Meaning: An item listed in the definition of a group is unknown.
Action: You should stop the program (type control C twice). You must have made an error editing the file in section 2.3. Go back and fix it. Then come back to 2.4.

xxxxx NOT A PLACE
Meaning: An item listed in a group definition is a known code, but isn't an airport or center.
Action: Stop the program (type control C twice).
You may have typed the wrong file name. Try running FILES again, and be careful to answer the right group file name to each question.
If the error persists with the correct file names, act as above for UNKNOWN PLACE errors (i.e. the file has an error).

UNKNOWN AIRLN xxxxx
Meaning: An item in a group definition is unknown.
Action: None.
There will normally be several of these.
The TAXI definition was typed to contain all commuter airlines, whether or not known to AIRS. These are simply ignored if not known.

xxxxx NOT A AIRLN
Meaning: An item listed in an airline group definition is a known code, but not an airline.
Action: Same as for NOT A PLACE.

UNKNOWN TYPE xxxxx
Meaning: An item in an aircraft type group definition is unknown.
Action: Same as for UNKNOWN PLACE.

xxxxx NOT A TYPE
Meaning: An item in an aircraft type group definition is a known code, but not an aircraft type.
Action: Same as for NOT A PLACE.

(3) Array overflow

After one of these messages, the program exits.
Someone knowledgeable about AIRS must work this out.

TERMS OVERFLOW
The file AIRPT.RD contains more than 2047 airports (terminals).

CENTS OVERFLOW
The file CENTR.TAB contains more than 50 center definitions.

TYPES OVERFLOW
The file ACFT.RD contains more than 127 aircraft types.

GROUP OVERFLOW
The group definitions (from the three group files) take too much space.

Program #2: PACKRD

Run time: about 10-15 minutes

output:

n FLIGHTS

Number of flights matches number of FLIGHTS WRITTEN OUT in statistics at end of RUBINE, section 2.2.2.2.

1000
2000
.
.
.

Counts to number of flights by 1000s.

DESCRIPTION

files read: TABLS.RD (from FILES)
FLIGHT.RD (from RUBINE)
files created: SCHED.PAK

Uses dictionary from TABLS.RD to encode schedules from FLIGHT.RD. Packs flights into final format and writes on SCHED.PAK. SCHED.PAK contains number of flights, then 8-word records.
size(SCHED.PAK)=8/7*size(FLIGHT.RD)

Program #3: SORTRD

Run time: about 15-20 minutes

output:

```
START SORT
 .
 .
 .
START SORT
DONE SORTING
```

Types "START SORT" once for every 2500 flights, then DONE message, then continues to run for a while.

DESCRIPTION

files read: SCHED.PAK (from PACKRD)
scratch file: SCRTCH
files created: ARRIVE

Sorts schedules from input file by destination and origin, writing sorted buffersful on scratch file. Merges sorted flights onto output file. Deletes SCRTCH when done. ARRIVE has same form and size as SCHED.PAK.

Program #4: XTRACT

Run time: about 8-10 minutes

output:

OMITTING FLIGHTS BEFORE m/d

Types a date about 15 days past. This date will be required for 2.5.2.

SCHED.CRF PROTECTED

Current schedule file has been write-protected. ARO users cannot make entries until sections 2.4 and 2.5 are completed.

m FLIGHTS RETAINED OUT OF n

n is number of flights in current schedule file

m is number still relevant (not omitted)

ERRORS

bad date in computer
  self-explanatory message
  program exits

DESCRIPTION

files read: SCHED.CRF (current schedules)
files created: OSCHED.OK

Changes protection of SCHED.CRF to <357>.
After this, no ARO entries can be made.
Copies current flights (omitting those cancelled earlier than the given date) from old data base to output file. Output file has same format as SCHED.PAK (in PACKRD program).

Program #5: OSORT

Run Time: about 10-15 minutes

output:

START SORT
.
.
.
START SORT
DONE SORTING

Types "START SORT" once for every 2500 flights, then DONE message, then continues to run for a while.

DESCRIPTION

files read: OSCHED.OK (from XTRACT)
scratch file: SCRTCH
files created: OSCHED.SRT

Same as SORTRD but different input and output files.

Program #6: MERGE

Run time: about 10-15 minutes

no output when doing data base creation (this chapter)

NO NEW FLIGHT FILE if doing chapter 3 processing (data base truncation)

DESCRIPTION

files read: OSCHED.SRT (from OSORT)
ARRIVE (from SORTRD)
files created: SCHEDN.CRF (final schedules)

Note: ARRIVE only read (only exists) when doing chapter 2 processing, not when doing chapter 3.

Merges old schedules (OSCHED.SRT) with new (ARRIVE), if any, onto SCHEDN.CRF. 1st word is number of flights, then 8-word entries start at word 17. Extends file with 500 blocks of 0s.

Program #7: TCREF

Run Time: about 5 minutes

output:

n FLIGHTS
1000
2000
.
.
.

Counts to number of flights by 1000s.

r RANGES
1000
2000
.
.
.

Counts to number of ranges by 1000s.

DESCRIPTION

files read: SCHEDN.CRF (from MERGE)
scratch file: PTR.TMP
files created: TERMN.CRF (terminal cross-ref)

Forms cross-reference of schedules by airport. First forms scratch file (while counting flights), then forms output file (while counting ranges). Scratch file is deleted.

Program #8: CCREF

Run Time: about 5 minutes

output:

n FLIGHTS
1000
2000
.
.
.

Counts to number of flights by 1000s.

r RANGES
1000
2000
.
.
.

Counts to number of ranges by 1000s.

ERRORS

UNKNOWN TERMINAL xxxxx
Meaning: Shouldn't happen.
Unknown airport code in a center definition on file CENTR.TAB. Should already have been caught and fixed when you ran CNTCHK in section 2.3.
Action: Stop the program (type control C twice).
You must have made an error editing CENTR.TAB in section 2.3. Go back and fix it. Then resume from this point by typing RUN CCREF.

DESCRIPTION

files read: SCHEDN.CRF (from MERGE)
TABLS.RD (from FILES)
CENTR.TAB
scratch file: PTR.TMP
files created: CENTN.CRF (center cross-ref)
files modified: PARAM.TRM (terminal parameters)

Uses dictionary from TABLS.RD and center definitions from CENTR.TAB. Forms cross-reference of schedules by center. Same stages (including scratch file use and deletion) as TCREF. Updates terminal/center correspondence on PARAM.TRM.

Program #9: FREFIL

Run time: about 1 minute

no output

DESCRIPTION

files read: TERMN.CRF (from TCREF)
CENTN.CRF (from CCREF)
files created: TERMN.FRE
CENTN.FRE

Each output file has 0 on word 1, length of associated CRF file on word 2, then 2 blocks of 0.

Program #10: EXTCRF

Run time: about 1 minute

no output

DESCRIPTION

files modified: TERMN.CRF (from TCREF)
CENTN.CRF (from CCREF)

Extends each file with 200 blocks of 0.

2.4.3 ERROR HANDLING

Errors associated with a particular program in the series were described in 2.4.2 under the program description. This section covers error handling which applies to all programs.

COMPUTER CRASHES

If the run is interrupted due to a crash, you can continue it later. You know from the "ENTER" and "EXIT" messages described in 2.4.1 what program was in progress at the time of the crash, and can restart from that program as described in 2.4.1. For example, if the system crashes after entering MERGE but before exiting it, you can restart it by typing "RUN MERGE".

If you will not be able to resume the run for a long time (for example, if the computer comes back up too late in the day and you can't continue till the next day), be careful not to lock out ARO entries the whole time. If the data base was not yet locked (XTRACT has not protected SCHED.CRF yet), you can simply resume when ready from the interrupted program. If the data base was locked (XTRACT protected SCHED.CRF) you should unlock it as follows:

```
.PROTECT SCHED.CRF <337>
```

Then when you are ready to resume processing, you must back up to XTRACT -- you cannot pick up from where you left off -- by typing

```
.RUN XTRACT
```

[EXCEEDING QUOTA ON TSCA]

All disk space allocated for [57041,1161] on disk TSCA has been exhausted. Unless all physical space on the disk is exhausted (in which case a different message will appear), the program will continue to run and files will be written correctly. However, a RAX output error will occur when the files are being closed and the program will bomb out. You should determine whether the program has run to completion. In some programs, you can tell by seeing whether it has completed its outputs and knowing that the file giving the output error (as determined from Appendix B) was the last one being written. You may have to do more snooping around, seeing what files have actually been created, etc., to decide whether the

program finished. Before you can continue the processing, you must free up some disk space. You can do this by deleting intermediate files - that is, files created during the processing which are not needed for any subsequent stages and which are not the data base being created (SCHEDN.CRF, TERMN.CRF, CENTN.CRF, TERMN.FRE, CENTN.FRE). (Normally these are deleted when the data preparation is complete - 2.6.) Do not delete files used by the program which was interrupted. If the interrupted program did not complete its job (or you are not sure), you should restart it. If enough space was freed, it should now run. If the interrupted program did run to completion, you may continue the sequence with the next program.

RAX ERRORS

If a program bombs out with a RAX error, you should delete any files created by the program (as listed in the program description) and restart the program. Hopefully the error was spurious and the program will work correctly when rerun. For example, if PACKRD died with a RAX error you would:

.DELETE SCHED.PAK
then .RUN PACKRD

If the error repeats, you will have to do some work to track down the problem. You can find out what the given RAX error code means from Appendix A, and what file is on the given unit from Appendix B. Actually, some programs open more than one file on the same unit (not at the same time!), so you may have to do some more sleuthing to figure out which file is at fault. The program description in section 2.4.2 may help you deduce the cause of the problem.

If the error was on a file created by the current program, you're in trouble, since simply deleting the file and rerunning the program didn't help.

If the error was on a file created in an earlier step of the sequence of programs, try deleting the file and restarting from the program that created it.
For example, if program MERGE is getting a RAX error on unit 1, you would

.DELETE OSCHED.SRT

to delete the bad file, then

.RUN OSORT

to back up the processing to the start of OSORT, which created OSCHED.SRT.

If the error was on a file not created by a program in this sequence, there may be serious trouble.

ERROR n TRYING TO RUN prog

The program failed to run the next one in the sequence, namely "prog". The error code, n, is given by the RUN UUO, as documented in section 4.2 of "Operating System Commands" in Reference 4 (p.780).

2.5 HANDING OVER DATA BASE

2.5.1 REPLACING OLD DATA BASE WITH NEW ONE

DATA BASE FILES

AIRS works from a data base consisting of (among others) the 5 files:
SCHED.CRF, TERM.CRF, CENT.CRF, TERM.FRE, CENT.FRE

The newly created data base consists of the files:
SCHEDN.CRF, TERMN.CRF, CENTN.CRF, TERMN.FRE, CENTN.FRE

An old copy of the data base is kept under the names:
SCHED2.CRF, TERM2.CRF, CENT2.CRF, TERM2.FRE, CENT2.FRE

STEP 1

First check the existence and status of the data base files by the command:

.DIR *.CRF, *.FRE /W

Complete data on all files having extensions "CRF" or "FRE" will be listed. This data includes the size, protection, creation (or modification) date and time, mode, and access date. the list should include the 10 files making up the current and new data bases (as named above). The old data base should not appear, since it was deleted by the program PREPAR, which was run before the automatic sequence of data preparation programs was begun (section 3 or 2.4.1).

If for some reason you forgot to run PREPAR, so that the old data base exists, you should run it now. I.e.

.RUN PREPAR

The current data base should have protection <337>, except for SCHED.CRF, which was given <357> protection by XTRACT in 2.4.2; the new data base should have protection <057>.

If any file has incorrect protection, you should fix it by the command:

.PROTECT filname <protection>

STEP 2

To hand over the new data base, type:

.RUN NEWDAT

NEWDAT renames the current data base to save it under the "old" name, then renames the new data base to make it the current one. File protections are set as appropriate. Thus when NEWDAT exits, the new data base has become the current one. It is available for AIRS use, including ARO entries (which have been prevented since XTRACT ran).

If for any reason NEWDAT fails to run to completion (which is unlikely, since it takes under a minute), you should re-run it immediately. I.e., repeat Step 2. DO NOT repeat Step 1. After a re-run, Step 3 is likely to show up correctly named but incorrectly protected files, which you will have to fix as described under Step 3.

STEP 3

Check the existence and status of the data base files by the command

.DIR *.CRF, *.FRE /W

as in Step 1. This time the list should include the 10 files making up the old and current data bases; no "new" data base should appear. The current data base should have protection <337> and the old one should be <057>. Check that all files have been correctly renamed. That is, the file listed here as SCHED2.CRF should be the one (have same size, creation date and time, etc.) listed as SCHED.CRF in Step 1, and the file listed here as SCHED.CRF should be the one listed as SCHEDN.CRF in Step 1. The same relationship should hold for the other files. The renames performed by NEWDAT are summarized in the table below.

If all the files have been correctly renamed and protected, go on to 2.5.2.

new name and protection		old name and protection	
SCHED2.CRF	<057>	SCHED.CRF	<357>
TERM2.CRF	<057>	TERM.CRF	<337>
CENT2.CRF	<057>	CENT.CRF	<337>
TERM2.FRE	<057>	TERM.FRE	<337>
CENT2.FRE	<057>	CENT.FRE	<337>
SCHED.CRF	<337>	SCHEDN.CRF	<057>
TERM.CRF	<337>	TERMN.CRF	<057>
CENT.CRF	<337>	CENTN.CRF	<057>
TERM.FRE	<337>	TERMN.FRE	<057>
CENT.FRE	<337>	CENTN.FRE	<057>

If any files are not correctly renamed or protected you must do them by hand, using the following commands. (Note: This is unlikely to occur, but is conceivable for reasons we won't go into.)

To protect a file, you

.PROTECT filename <protection>

To rename a file, you

.RENAME newname=oldname

(Note: RENAME will fail on a file with <337> or <357> protection. Such a file should first be given <057> protection, then renamed.)

You should then repeat the check of Step 3.

2.5.2 EDITING AIRS MESSAGE

The file AIRS.MSG contains a message which is typed out each time AIRS is run. The first line tells the effective dates of the data base. For example: "DATA FOR JUL.7 THRU AUG.31"

This should be edited to reflect the new dates as follows.

Use a text editor (such as COED - see Appendix C) to print the first line of the file. Change the start date to be the date typed out by XTRACT in 2.4.2 and the end date to be the end date entered in 2.2.2.2 when processing the tape. (Note: If you are doing chapter 3 processing you will not have processed a tape and will not need to change the end date.)

2.5.3 TESTING NEW DATA BASE

Just to be sure the new data base is OK, you should run AIRS and make some test requests. It would help to be familiar with AIRS traffic demand requests (see Ref.2).

Type: RUN AIRS

Make a few requests, using both airports and centers to check both cross-reference files. A sample session follows.

REQUEST=D BOS DCA 1100 1300
(departures from Boston to Washington between 1100 and 1300)

REQUEST=LIST
(flights listed should have origin BOS, destination DCA, departure time as appropriate)

REQUEST=D ZBW ZDC 1100 1300
(departures from Boston center to Washington center between 1100 and 1300)

REQUEST=LIST
(should include flights retrieved in previous request)

REQUEST=QUIT
(quit AIRS)

2.6 CLEANING UP

what: Intermediate files produced during the data processing of sections 2.2 and 2.4 were kept around so that the processing could be backed up to an earlier stage if necessary. In this step, these files are deleted.

when: This step should not be done until the new data base has been saved on a failsafe tape. This will usually be the day after it was created, since a failsafe is made each night. Thus, the day after the data base was created, if a good failsafe was obtained (see chapter 5), this cleanup should be done. Otherwise, wait another day, or as long as necessary. This way, if the data base becomes damaged and there is no backup copy available, it can at least be recreated.

how: Simply type: .RUN CLEAN
to run the cleanup program. No output is produced.

CLEAN deletes the following files:
SCHED.PAK, ARRIVE, OSCHED.OK, OSCHED.SRT, SAVE.1, SAVE.2, SAVE.ERR, TABLS.OLD, TABL2.OLD

If you are cleaning up after a data base creation (i.e. starting from a schedule tape), you should also

.DELETE FLIGHT.RD

NOTE: Do NOT do this deletion if you are cleaning up after a data base truncation (chapter 3).

3. DATA BASE TRUNCATION

INTRODUCTION

The support function described in this chapter is done once a month, but is optional. It can be done any time after the 15th of the month, before the new data base is created. It is probably worth doing if you won't be creating the new data base for a week or more. Otherwise the lifetime of the shortened data base will be rather short, and it may not be worth your while to create it.

A new data base is formed by omitting all flights discontinued at least 15 days ago from the schedules and re-sorting the schedules. Since most schedules are effective throughout a month, doing this after the 15th will remove about half the schedules (last month's) while doing it before the 15th would have a negligible effect.

PROCEDURE

The procedure is the same as part of the data base creation procedure (chapter 2). The information in 2.4.1 under "WHEN TO RUN" applies, except that the procedure takes about a half hour less.

First prepare disk space (as in 2.4.1) by typing

```
.RUN PREPAR
```

Then start from the XTRACT program (4th program in 2.4.2) by typing

```
.RUN XTRACT
```

The procedure continues through the rest of 2.4, 2.5, and 2.6, with the following differences:

-The MERGE program (2.4.2) types "NO NEW FLIGHT FILE", since there are no new schedules being incorporated.

-When you change the dates in the message file (2.5.2), only the start date changes.

Note that, just as in 2.4, the ARO is prevented from making entries during the processing. It is therefore desirable to do this when ARO is not busy making entries.

4. USAGE RECORDS

4.1 Introduction

Every time AIRS is run, it leaves a record of its use on a file. The file is named U.nn, where nn is 10 plus the number of the job which ran AIRS. These records are collected in order to see how AIRS is being used and to catch certain problems which may develop. The usage records should be typed out every day. Section 4.2 tells how to type them out and section 4.3 tells how to detect problems with AIRS by reading them.

4.2 Obtaining the Usage Records

The following steps should be followed every day to obtain the usage records.

(1) Log in on [57041,1162].

(2) Find out what usage files exist by the command:

```
.DIR U.*
```

This will list all usage files, if any (all files with first name U). If there are no files listed, you are done. Otherwise, go on to step 3.

(3) For each usage file, do the following:

a. Find out if the file is currently being recorded on by an AIRS job by the command:

```
.SYS n
```

where n is the job number which would write on that file; i.e. the second name of the file minus 10.

Example: If there is a file named "U.23",
type "SYS 13".

"SYS n" tells you the status of job n. If the response is just a ".", no one is using that U file (no one has that job number) so go on to step b. If the response is anything other than ".", go on to the next U file and leave this one for another time.

b. Type the file just checked by the command:

```
.TYPE U.nn
```

The file will be typed out.

c. Count ARO entries

ARO entries are recorded as "*R" (reservations) and "*X" (cancellations). If there are no ARO entries on the usage record, go on to the next step (d).

To count the ARO entries, type:

```
.RUN COUNT
```

You will be asked:

```
FILE?=
```

and should type the name of the usage file.
You will be asked

```
DIVIDING WORD=
```

If the usage record has only one date, just type carriage return. If it has two dates, type the first five characters of the line containing the second date. If the line starts with a blank, you must include the blank. For example: If the record starts with " 8-FEB-73" and switches to " 9-FEB-73", you should type " 9-FE". (Giving more than 5 characters doesn't hurt, but don't give fewer.)

The number of reservations, cancellations, and the total will be typed.
For example: R=200 X= 10 T=210.

If there was a dividing word, separate counts will be given for the two dates.

Record the counts on the usage record.

Note: There is no way to specify more than one dividing word to the COUNT program, in case the record has entries for more than two dates.

d. Delete the file

Now that you have a written record of the usage, the file should be deleted as follows.

.DELETE *.nn

This will delete all files with last name nn, which may include other scratch files with the job number + 10 as their second name, as well as the usage record.

All files deleted will be listed. The possible files have first names "U", "HASH", "FLTS", "CHAIN".

If you get an error message that the file is being modified and can't be deleted. This means that someone now has that job number and is appending to that same usage record. In other words, if you again did a SYS you would not get a dot. This means that next time you type out usage files, this one will still be there, containing the part you already typed out as well as new parts added on by the new job.

(4) When all usage files not in use have been typed and deleted, you may log out.

(5) Store the usage records, arranged by date. If a single record covers more than one date, separate it.

4.3 Interpreting Error Messages

We will not describe the content of a normal usage record. This section covers messages on the usage record which indicate that something abnormal has happened. Most of these errors have never occurred and probably never will. Those that do occur usually are self-healing, requiring no corrective action. Although an explanation is given along with each error message, some knowledge of AIRS may be needed to deal with errors.

Note: Many of the error messages refer to unit numbers. The file associated with each unit number can be determined from Appendix B.

*PTON
or
*PTOFF

Associated with TAPE, KEY, AIRS, or QUIT request in ARO mode. The TRMOP. UUO failed to turn on or off paper tape mode. This shouldn't happen.

*ECHO
or
*NOECHO

Associated with TAPE, KEY, AIRS, or QUIT request in ARO mode. The program failed to turn echoing on or off because it failed to find the channel the teletype was open on. This can only happen if no teletype I/O was done before the attempt to control echoing, which should not happen in AIRS.

FOREST OVERFLOW

There is not enough room in the FOREST array to represent the request. The only cure is to modify AIRS.

UNIT u RAX ERR 7
FIXED TIME= n

The unit will always be one of the permanent data base files, not a scratch file. AIRS got an input error on the given file, and tried to fix it, assuming it was a checksum error. It kept trying to read the file without errors. On the nth attempt, the file read error-free. Thus n-1 fixes were required. If n=1, the error was spurious - i.e. no

errors were found to be fixed.

UNIT u RAX ERR 7
CANT FIX

Same as above (FIXED TIME=n) but after 10 fixes the file still had errors. There may be serious damage in the file that can't be fixed, or there may simply be more than 10 fixable errors. In the latter case, rerunning AIRS may do the trick, as it may finish the fix the next time through.

UNIT u RAX ERR n

The given error was encountered on the given file. If the file is a scratch file (U.nn, HASH.nn, FLTS.nn, CHAIN.nn), there should be no problem, since it will be deleted and recreated when AIRS is run again. If the file is a data base file, then:
1.If n is 7, and neither of the above messages (FIXED or CANT FIX) appears, AIRS was interrupted while trying to fix the file. The error will still be there next time the file is read.
2.If n is not 7, then either the error is spurious, in which case it will not repeat when AIRS is rerun, or there is something wrong with the file.

FLIT NOT APPENDED -
FLIT, LAST, FIL, ADR, NW
n1 n2 n3 n4 n5

While processing an ARO entry, a flight pointer being appended to a cross-reference pointer list was not greater than the last pointer on the list.
The five numbers printed out are:
1.number of flight being appended
2.last flight pointer on cross-reference being appended to
3.unit number of cross-reference file (terminal or center)
4.address of start of cross-reference list in question
5.number of words in cross-reference list in question

BAD RANGE
FILE, ADR, SIZE=
f a s
RANGE, NFLITS=
r1 r2 n

A pointer range being used to retrieve flights seems to be bad. The retrieval pointers are on the file on unit f (either a cross-reference or scratch file), and consist of s words starting at address a. The range in question is r1-r2, and the number of flights on the schedule file is n. One of the following 3 cases holds: (1) r1 is negative (2) r1 is greater than r2 (3) r2 is more than 10 greater than n.

If (1) or (2):
(a) If f is a cross-reference file (TERM.CRF or CENT.CRF), either the error was spurious or the file is damaged. This can be checked by repeating the offending request and seeing what happens. If the file is damaged, the data base will probably have to be backed up (see chapter 6). The file should first be analyzed for errors using the checking programs described in chapter 5.4 (particularly CRFCHK and FRECHK).
(b) If f is a scratch file, ignore the problem. There was probably an error in writing to or reading from the file. When the request is repeated, the file is recreated and should be OK. If the error persists, the cross-reference of one of the places in the request may be damaged. The cross-reference can be checked and the data base backed up if necessary as in (a) above.

If (3):
A pointer should not be greater than the total number of flights. However, it may seem greater if ARO entries are in progress during the request processing, because the number of flights is read early in the processing, and new flights may be added (and show up in the cross-reference) before this test is made. The threshold of 10 in the test allows for 10 entries during the request. If r2-n is only a little more than 10, it may simply be that more entries were made. If r2 is much greater than n, we have the same situation as in case (1) and (2).

AREA NOT -1
FILE, ADR, NW=
u a n

An area on a cross-reference file which was listed as free and was going to be used is not filled with -1's. The cross-reference file is the one on unit u and the area consists of the n words starting at address a of the file. The offending area is not used. It is removed from the free list and the program quits.

```
BAD RANGE BEFORE WRITE
FILE, ADR, NW=
u     a    n
```

While copying a pointer list to a larger area of the cross-reference file on unit u, AIRS encountered a bad pointer. Either some pointer being copied was negative, zero, or greater than 1,000,000, or the pointers were not in increasing order. The bad pointer list consists of the n words at address a on the cross-ref file on unit u. The original (bad-looking) copy of the pointers is still the one in effect. Either the error is spurious or the file is damaged. See (a) under "BAD RANGE" message above.

```
BAD RANGE AFTER WRITE
FILE, ADR, NW=
u     a    n
```

After copying a pointer list to a larger area of the cross-reference file on unit u, AIRS checked the new copy and found a bad pointer as in the previous error message. The bad pointer list consists of the n words at address a on the cross-ref file on unit u. The original (non-erroneous) copy of the pointers remains in effect.

```
BAD CRF WRITE ADR OR # WORDS
ADR, NW1, NW2, NEXT=
a    n1   n2   n
```

AIRS is trying to move a cross-reference list to a larger area on one of the cross-reference files, but one of the values involved doesn't make sense. The free area to be used consists of the n2 words at address a, the list being moved is n1 words long, and the next free address at the end of the cross-reference file is n. The error detected was one of the following:
bad number of words: n1 is ridiculous (<0 or >1,000,000)
bad address: a is ridiculous (<0 or >1,000,000)
free area overlaps end of file: a, n2, and n don't make sense together (a>n or a<n but a+n2>n)

The free area is removed from the free list, nothing is moved, and the program quits.

5. DATA BASE INTEGRITY CHECK

5.1 INTRODUCTION

In order to maintain the integrity of the AIRS data base, a batch job is automatically run every night after 11:00 (2300). This job runs a series of programs to check the data base and makes a copy of our disk on magnetic tape. A record of the job's run is left in a file, called the log, which should be checked every morning to see if any problems were encountered.

This chapter is divided as follows:

Section 5.2 describes the mechanics of handling the batch job - i.e. how to obtain the log file, how to submit the job if necessary, etc.

Section 5.3 tells how to interpret the log by describing the content and flow of control of the batch job.

Section 5.4 describes the checking programs. These descriptions should enable you not only to interpret the log file, but also to run any of the programs by hand, as might be desirable if the AIRS data base seems to have been clobbered.

Note: All operations described in this chapter take place in [57041,1161].

5.2 MECHANICS

Our regular batch job is named AIRCHK. The commands and inputs which make up the job are contained in the control file AIRCHK.CTL, and the log of the job's run is contained in the file AIRCHK.LOG. A second batch job, run if necessary by AIRCHK, is named AIRCH2. Its control and log files are named AIRCH2.CTL and AIRCH2.LOG.

5.2.1 OBTAINING THE LOG

The following should be done every morning.

1) Log in on [57041,1161].

2) Type the log file by the command:

.TYPE AIRCHK.LOG

The file AIRCHK.LOG (the log file from the batch job AIRCHK) should be typed out.

If you get an error message that the file cannot be found, it probably means the batch job wasn't run. You should check the batch queue as in 5.2.2. If neither AIRCHK nor AIRCH2 is scheduled, you should resubmit AIRCHK (not AIRCH2!) as in 5.2.3.

3) Delete the log file by the command:

.DELETE AIRCHK.LOG

The system will confirm that it deleted the file.

4) If the log shows that AIRCH2 was submitted, you should repeat steps 2) and 3) for the file AIRCH2.LOG (i.e. "TYPE AIRCH2.LOG" and "DELETE AIRCH2.LOG").

5) Log out.

6) Interpret the log for abnormalities according to the information in 5.3 and 5.4.

Alternate Method:

It is not necessary to have a complete hard-copy record of the batch runs. You may view the log on a display terminal instead of printing it out. The procedure is exactly as above, except that you interpret the log as you

view it. In this case, you should keep a written record of when the job ran, any unusual events in the log, etc. If there are serious abnormalities it is probably a good idea to print out the log (at least the bad part) before deleting it.

5.2.2 CHECKING THE BATCH QUEUE

To check whether the job is scheduled, type:

```
.SUBMIT [57041,1161] /L
```

This will print out a list of batch jobs pending for [57041,1161]. If AIRCHK (or AIRCH2) is on the list, it will get run eventually. The last column tells when it is scheduled for in terms of hours, minutes, and seconds after the current time. If there is a value in the PTY column, the job is currently running.

5.2.3 SUBMITTING THE JOB

If it is necessary to resubmit the job (it is not on the queue), give the command:

```
.SUBMIT AIRCHK/AFTER:2300/OUTPUT:0/RESTART:1/TIME:8:00
```

This will put the job back on the queue for processing tonight as usual. (Note: you will never submit AIRCH2. AIRCH2 is only submitted by AIRCHK if necessary.)

If the SUBMIT fails with a non-existent file message, it means the control file for the batch job, AIRCHK.CTL, is missing. You should restore the file (see chapter 6) then try again to submit the job.

If you realize you have submitted the job erroneously (typed one of the switches wrong, etc.) you can remove it from the queue by typing

```
.SUBMIT AIRCHK=/K
```

then resubmit it as above. (If the job has already started running - for example, if you omitted the "AFTER" switch - you will be unable to remove it from the queue.)

5.3 THE BATCH JOB

The following references will be helpful in understanding the batch job and log file. We will omit from our discussion information which can readily be found in them.

1. First Data Corp. Technical Brief #67, "Batch For Time-Sharing Users"

2. "Beginner's Guide to Multiprogram Batch", in Ref.4

3. Chapter 3 of "Operating System Commands" in Ref.4

5.3.1 LOG FILES

BASIC STRUCTURE

A normal log file for a single run of the batch job contains the following basic sections:

1. Job startup:
   The batch controller logs in to run the job. Some batch system messages appear, then a login, then some standard initialization commands.

2. Body of job:
   The job is run as specified by the control file. The batch job commands are executed, and all commands, inputs, and outputs appear in the log. These are discussed in section 5.3.2.

3. Job termination:
   The job is logged out. The usual logout messages appear.

MULTIPLE LOG

If the job is run more than once and the log is not deleted in between (as happens over weekends), the log contains records of all the runs, appended one after the other.

INCOMPLETE LOG

If the log of a run is incomplete (does not include the logout), it probably means the system crashed during the run. The job is supposed to be automatically resubmitted after a crash (because of the "RESTART:1" parameter given when it is submitted).

If there is another log after the aborted one, then obviously the job was resubmitted and run.

If there is no appended log, you may have to resubmit the job yourself. You should check the queue as in 5.2.2. If neither AIRCHK nor AIRCH2 is scheduled, submit AIRCHK as in 5.2.3. It is probable that due to the crash the schedule file was left protected. Check this by the command

.DIR SCHED.CRF /P

If the protection is <337>, the file is OK. If it is <357>, then ARO entries are still locked out. If you are sure that there is no other reason for the file to be protected (i.e. neither data base creation (chap.2) nor data base truncation (chap.3) is in progress), you should

.PROTECT SCHED.CRF <337>

5.3.2 CONTENT AND FLOW OF CONTROL

Section 5.3.3 contains listings of the batch control files, AIRCHK.CTL and AIRCH2.CTL, which contain the monitor commands, batch system commands, and program inputs which make up our job. We will not go through a statement-by-statement explanation, as the batch commands are well described in the references listed at the start of 5.3. This section just tells in English what the job is doing.

AIRCHK

AIRCHK, which is run every night, does the following:

1. Tries to find out whether AIRS users are modifying the data base by protecting the schedule file against writing by AIRS users.

Normal: The file is unlikely to be busy when the job runs at its normal time, 11:00 p.m. If the protection succeeds, AIRCHK sets the file protection back to normal.

Abnormal: If the protection fails (the file is being written), the job resubmits itself to run again in 15 minutes and quits; there should then be another log for about 15 minutes later.

2. Asks the operator to mount the appropriate failsafe tape, write-enabled. Seven tapes are used in rotation, with one tape being saved each month.

Normal: The tape is mounted, possibly following a long wait.

Abnormal: The request fails, because no operator is present. AIRCHK submits the alternate job, AIRCH2, to run immediately, then quits.

3. Again protects the schedule file against writing by AIRS users. It was not left protected the first time so that ARO use would not be prevented if the tape mount took a long time.

Normal: The protection succeeds. The file is unlikely to be busy when the job runs at its normal time, 11:00 p.m.

Abnormal: If the protection fails (the file is being written), AIRCHK asks the operator to dismount the tape, then resubmits itself as in step 1 above.

4. The 6 data base maintenance programs (CHKSUM, EXTEND, FRECHK, CRFCHK, SCHCHK, UFDCHK) are run. These programs are described in section 5.4.

Normal: As described in 5.4.

Abnormal: As described in 5.4, plus the following. If an error occurs which is not handled by the program running (a "?" error which aborts the program), the rest of the maintenance programs are not executed and the job goes on to the next step. An example of this would be a RAX error which the program can't deal with; RAX errors can be interpreted by reference to Appendices A and B.

5. Parameters are set so that "?" errors will be ignored and "$" messages will be passed to the computer operator for action.

6. Directory 57041,1161 is saved on failsafe tape. Among other messages, "TSCA 57041,1161" (signalling the start of the save) and "$SAVE COMPLETED ... " messages should appear. The operator must then continue the job.

7. The schedule file is returned to its original protection.

8. Directory 57041,1162 is saved on failsafe tape. Messages similar to step 6 should appear.

9. Asks operator to dismount the failsafe tape.

10. Submits AIRCHK to run the next night after 11:00 p.m. Other "SUBMIT" parameters prevent line printer output, request automatic job restart in event of a crash, and set the CPU time limit for the job.

AIRCH2

AIRCH2 is basically the same as AIRCHK, but it only runs the maintenance programs and does not make a failsafe tape. It is run by AIRCHK when no operator is available to mount the tape.

1. Protects the schedule file against writing by AIRS users.

Normal: As in step 3 for AIRCHK.

Abnormal: As in step 1 for AIRCHK.

2. Runs the maintenance programs as in step 4 for AIRCHK.

Normal: As in step 4 for AIRCHK.

Abnormal: As in step 4 for AIRCHK. If a "?" error occurs, skips the rest of the 6 programs and goes on to the next step.

3. Same as step 7 in AIRCHK.

4. Same as step 10 in AIRCHK.

5.3.3 CONTROL FILE LISTINGS

LISTING OF AIRCHK.CTL

```
 .PROTECT SCHED.CRF<357>
 .IF(ERROR) .GOTO LATER
 .PROTECT SCHED.CRF<337>
 .MOUNT MTA:FAILSAFE/B /WEN /VID:'TODAYS TSCA FAILSAFE TAPE'
 .IF(ERROR) .GOTO NOOPER
 .PROTECT SCHED.CRF<357>
 .IF(ERROR) .GOTO LATER2
 .RUN CHKSUM
 .RUN EXTEND
 .RUN FRECHK
 *CENT.CRF
 *CENT.FRE
 *50
 *TERM.CRF
 *TERM.FRE
 *2047
 *NONE
 .RUN CRFCHK
 *CENT.CRF
 *50
 *(2I13)
 *TERM.CRF
 *2047
 *(2I13)
 *NONE
 .RUN SCHCHK
 .RUN UFDCHK
 .GOTO FSAFE
 FSAFE: .ERROR &
 .OPERATOR $
 .R FAILSAFE
 */G57041,1161
 */O57041,1161
 */U
 .PROTECT SCHED.CRF<337>
 .R FAILSAFE
 */G57041,1162
 */O57041,1162
 */U
 .NOOPERATOR
 .DISMOUNT FAILSAFE /R /B
 .SUBMIT AIRCHK/AFTER:2300/OUTPUT:0/RESTART:1 /TIME:8:00
 .GOTO END
 LATER2: .DISMOUNT FAILSAFE /R /B
 .GOTO LATER
 LATER: .SUBMIT AIRCHK /OUTPUT:0 /RESTART:1 /AFTER:+15 /TIME:8:00
```

```
.GOTO END
END: ;THAT'S ALL FOLKS!
%ERR: .BACKTO FSAFE
NOOPER:.SUBMIT AIRCH2 /OUTPUT:0 /RESTART:1 /TIME:8:00
```

LISTING OF AIRCH2.CTL

```
.PROTECT SCHED.CRF<357>
.IF(ERROR) .GOTO LATER
.RUN CHKSUM
.RUN EXTEND
.RUN FRECHK
*CENT.CRF
*CENT.FRE
*50
*TERM.CRF
*TERM.FRE
*2047
*NONE
.RUN CRFCHK
*CENT.CRF
*50
*(2I13)
*TERM.CRF
*2047
*(2I13)
*NONE
.RUN SCHCHK
.RUN UFDCHK
.ERROR &
.GOTO UNPRO
UNPRO: .PROTECT SCHED.CRF<337>
.SUBMIT AIRCHK/AFTER:2300/OUTPUT:0/RESTART:1 /TIME:8:00
.GOTO END
LATER: .SUBMIT AIRCH2 /OUTPUT:0 /RESTART:1 /AFTER:+15
.GOTO END
END: ;THAT'S ALL FOLKS!
%ERR: .BACKTO UNPRO
```

5.4 PROGRAM DESCRIPTIONS

The following data base maintenance programs are run by the batch job AIRCHK. They can also be run manually if desired. For each program, we give (at the left margin) the error-free run, then (indented) describe the various error conditions and what the program is doing. We also give any differences between a manual run of the program and a run by the batch job.

As usual, inputs are underlined.

Note: When the programs are run manually, the inputs appear as shown here. In the batch job, each input is preceded by a "*".

The programs are described in the order in which AIRCHK runs them.

Program CHKSUM

```
FILE SCHED.CRF
FILE TERM.CRF
FILE CENT.CRF
FILE TERM.FRE
FILE CENT.FRE
FILE PARAM.TRM
```

Types file name just before starting to process file.

DESCRIPTION

Fixes checksum errors on the 6 main data base files. Tries to read through the files without errors, making up to 10 attempts on each file. If a checksum error is encountered, it is fixed before the next attempt to read.

ERRORS

The error message refers to the previously named file. You can find out from Appendix A what the RAX error code means.

```
RAX ERROR 7
FIXING IT
```

Error 7 is usually a checksum error. CHKSUM fixes it and tries again. This message may repeat. If the "GIVING UP" message below does not appear, the error was successfully handled.

```
RAX ERROR n      (where n≠7)
TRYING AGAIN
```

CHKSUM hopes the error was spurious, and simply tries again. This message may repeat. If the "GIVING UP" message below does not appear, the error disappeared.

```
TRIED 10 TIMES BUT STILL ERRORS!
GIVING UP ON filename
```

Gives up and goes on to the next file.

If the stubborn error was not #13 (file busy), there may be serious trouble with the file. This must be analyzed. If a checksum error remains, the file will be truncated on the failsafe tape. In any case, the failsafe copy made this session is suspect.

If the error was #13, it means the file could not be checked because it was being written on (by AIRS). During AIRCHK this is only likely to occur on PARAM.TRM. During a manual run, it can happen with any of the files.

Program EXTEND

May have no output or may print any or all of the following:

```
EXTENDING SCHED.CRF
EXTENDING TERM.CRF
EXTENDING TERM.FRE
EXTENDING CENT.CRF
EXTENDING CENT.FRE
```

DESCRIPTION

Compares the physical and logical lengths of each of the 5 data base files named above. If a file is almost used up (its logical length is within some threshold of its

physical length), it is extended (physically) with zeros and a message is typed.

Program FRECHK

CRF FILE=CENT.CRF
FRE FILE=CENT.FRE
MAX=50

Maximum # of centers on CENT.CRF is 50. Proceeds to check center files.

CRF FILE=TERM.CRF
FRE FILE=TERM.FRE
MAX=2047

Maximum # of terminals on TERM.CRF is 2047. Proceeds to check terminal files.

CRF FILE=NONE

No more files to check.

DESCRIPTION

Checks free-space files and cross-reference files for consistency. Each cross-reference file contains an index to the areas on it which are in use; the associated free-space file contains an index of unused areas on the cross-ref file and a pointer to the (logical) end of the cross-reference file.

ERRORS

Note: all addresses refer to the cross-reference file being processed.

HOLE OF w WORDS AT ADDRESS a

The w words at address a are not pointed to by either index. This is unlikely to occur, but can happen if AIRS is interrupted just before listing as free an area it has vacated. No action is required, as there is no data damage; there is simply some wasted (unusable because not pointed to) space on the file.

**OVERLAP OF n WORDS
f1 POINTS TO w WORDS AT ADDRESS a1; f2 points to address a2

The area of w words at address a1 overlaps the area at address a2 by n words.

f1 tells which file points to the area at a1. f1 is either "CRF" (the cross-ref file lists the area as used) or "FRE" (the free file lists the area as free).

f2 tells which file points to the overlapping area at a2. f2 is either "CRF" or "FRE" (same meaning as for f1 above), or "END" (the free file lists a2 as the end of the cross-ref file - i.e. the next free address after all used areas).

If f1 and f2 are both "CRF", there is serious damage. Two airports (or centers) think their pointers are listed in the same place. This will probably also cause the program CRFCHK (see below) to find errors. The data base must be backed up (see chapter 6). (If running programs manually, see warning about spurious errors under "MANUAL RUN" below.)

If f1 and f2 are not both "CRF", no action is required. If AIRS tries to use the erroneous "free" area, it will recognize the problem and remove the part it tried to use from the free list. This will show up in error messages to the user and on the usage record.

MANUAL RUN

When running the program manually, you may do either pair of files without doing the other. If the ARO is making entries while you are running FRECHK, spurious FRECHK errors may result, due to the files being modified during the checking. This can be prevented by making sure ARO stops running AIRS while you run FRECHK. No interference can happen during the batch job, since AIRCHK locks out ARO entries.

Program CRFCHK

```
CRF FILE=CENT.CRF
MAX=50
                Maximum # of centers on CENT.CRF is 50
FORMAT=(2I13)
                Proceeds to check CENT.CRF
CRF FILE=TERM.CRF
MAX=2047
                Maximum # of terminals on TERM.CRF is 2047
FORMAT=(2I13)
                Proceeds to check TERM.CRF
CRF FILE=NONE
                No more files to check
```

Note: A different FORTRAN format may be entered, but must not exceed 10 characters.

DESCRIPTION

Checks the cross-reference files for reasonableness by checking the pointers for each terminal's (center's) departures and arrivals to be sure each list of pointers consists of positive numbers in increasing order.

ERRORS

RAX ERROR n

The last file name typed is the one the error refers to. Error codes are explained in Appendix A. The processing of the current file is aborted, and the program goes back to the "CRF FILE=" question.

If an illegal pointer or pointer range is found, the entire pointer list containing it is typed as follows:

```
p   a   w
n   n
  .
  .
n   n
```

where:

a is the address of the bad pointer list on the cross-reference file

w is the number of words in the pointer list (twice the number of pointer ranges)

p is the relative position of the entry (a,w) in the index of the cross-reference file. The entry (a,w) is at address 2p+3 on the file.

The center or terminal in question is number (2p+3)/4. If p is odd, the pointers are for departures; if p is even they are for arrivals.

The n's are the w words of pointers from address a. They are typed in pairs (pointer ranges) using the format that was typed in answer to "FORMAT=".

Either some pointer is negative or zero, or the pointers are not all in increasing order. This can result either from the pointers being clobbered or the address (a) or number of words (w) being bad.

If a file is bad, the data base must be backed up (see chapter 6). (If running programs manually, see warning about spurious errors under "MANUAL RUN" below.)

MANUAL RUN

When running the program manually, you may do either of the cross-reference files without doing the other. If the ARO is making entries while you are running CRFCHK, spurious CRFCHK errors may result, due to the cross-reference file being modified during the checking. This can be prevented by making sure ARO stops running AIRS while you run CRFCHK. No interference can happen during the batch job, since AIRCHK locks out ARO entries.

Program SCHCHK

no output

DESCRIPTION

Spot checks the schedule file by making sure the planned time of arrival of each flight is valid (less than 2400).

ERRORS

If any flight has a bad arrival time (greater than 2359), the relative position of the flight on the file is typed out (i.e. 241 means the 241st flight on the file), followed by the 8 words of the flight entry in octal. To interpret the entry, see schedule file format description in AIRS documentation.

Program UFDCHK

[1161:][1162:]

DESCRIPTION

Reports what error bits, if any, are set in the status word of each of our UFDs (file directories). If a directory registers an error, reports what error bits, if any, are on for specific files in the directory.

ERRORS

All errors that involve a UFD (file directory) refer to the one currently being processed (either 57041,1161 or 57041,1162).

These status bits are as described in the ".RBSTS" argument (argument 17) under extended arguments for LOOKUP and ENTER in Ref.3. These error bits may persist long after the error is fixed.

(9) Some file in the directory has had a software checksum error.

(10) Some file in the directory has had a hard data error while being written.

(11) Some file in the direcotory has had a hard data error while being read.

filename(27) This file has had a software checksum error. If the file is one of the 6 data base files listed under the CHKSUM program, the error either has been or will be fixed by CHKSUM or AIRS; however the error bit will remain set until a new data base is formed. If the file is not one of the above, it may be permanently damaged and may require backing up.

filename(28) Same as (10) above but for this file. The file may be damaged enough to require backing up. You may need help from First Data to analyze this.

filename(29) Same as (11) above but for this file. See (28) above for action.

Except where otherwise noted, the following errors abort the program.

!ERR LOOKUP FILE filename
Shouldn't happen.
The named file is listed in the UFD but no such file exists ("LOOKUP" failed).
Program continues processing with next file.

!ERR OPEN CHANNEL 1
Shouldn't happen.
"OPEN" failed to open channel for reading from disk TSCA

!ERR OPEN CHANNEL 2
Shouldn't happen.
"OPEN" failed to open channel for reading from disk TSCA

!ERR LOOKUP UFD
Shouldn't happen.
"LOOKUP" couldn't find the UFD being processed.

!STRANGE IN ERROR ON UFD

Shouldn't happen.
"IN" got input error reading the UFD being processed, but the status bits registered neither an error nor end-of-file.

!IN ERROR ON UFD
Shouldn't happen.
"IN" got input error reading the UFD being processed.

6. BACKING UP THE DATA BASE

This chapter deals with recovery of the AIRS data base in case of file damage. The nightly batch job described in chapter 5 uses the FAILSAFE program to save copies of all of our files (not just the AIRS data base) on magnetic tapes (called failsafe tapes). If for some reason a file becomes damaged, it can be restored from one of these tapes by the computer operator upon request.

In general, to recover a file, you simply ask the operator to restore it from the TSCA failsafe tape. You may have to tell him which night's failsafe to use and which directory, [57041,1161] or [57041,1162], the file is in. Communication with the operator is via the PLEASE command (see Appendix C).

Recovery of the AIRS data base is somewhat more complex than just described. The data base consists of six files, all in the disk area [57041,1161] on disk TSCA. The files are:

SCHED.CRF	PARAM.TRM
TERM.CRF	TERM.FRE
CENT.CRF	CENT.FRE

These six files are interrelated and must be consistent. Therefore, if any one needs to be backed up, all six must be backed up.

The procedure for backing up the AIRS data base is as follows:

1. Make sure no one uses AIRS while the backup is in progress.

   AIRS must not be run until the backup is completed. Otherwise it will be working with an inconsistent set of files (some backed up versions, others newer versions) and may cause further damage. Call up CFCF and tell them not to use AIRS and not to let the ARO use it until you notify them. (Phone numbers are in Appendix D.)

2. Determine how far to back up.

   From the records of the nightly batch job (chapter 5) which checks the six files then forms the failsafe tape, you can determine the most recent occasion on which the files were known to be good and were failsafed. That is the failsafe tape you should restore files from.

3. Log in on [57041,1161].

4. Ask the operator to restore the files.

   Using the PLEASE command (see Appendix C), tell the computer operator the 6 files you want backed up. Tell him to use the TSCA failsafe tape, and tell him which one (as determined in step 2). Restoring files is a slow process, so stress its urgency, since the SCC is unable to use AIRS until its completion. You may ask the operator to notify you when done (either by phone or, if you stay logged in, on the computer terminal).

5. Wait until the backup is completed.

   Check that it is done (even if the operator says it is) by getting a directory listing of the six files as follows:

   .DIR SCHED.CRF,...(type the 6 file names)... /CRE,TI,ACC

   This will print the names of the files and, for each, the date and time when it was last modified and the date when it was last accessed. The access date should match the date the files were checked and failsafed (from step 2). The modification date and time provide a double check: If they are more recent than the failsafe date and time the file has not been restored yet.

6. Check file protections.

   Fix the protection of the schedule file as follows:

   .PROTECT SCHED.CRF <337>

   (it is write-protected, <357>, when failsafed).

   Make sure all 6 files have protection <337> as follows:

   .DIR SCHED.CRF,...(type the 6 file names)... /P

   This prints out the protection of each file. If any is not <337>, fix it as above (.PROTECT file <337>).

7. Make AIRS available.

   Call CFCF and tell them they and ARO can resume use of AIRS. Inform them that all ARO entries made since the date/time to which the files were backed up have been lost. (Note: There may be some confusion as to which are lost - entries entered on those dates, or entries for flights flying on those dates. The answer is, those entered on those dates.)

7. AIRS ERRORS

This chapter explains certain error messages given by AIRS. It does not include errors in the form or content of a user's request (such as an unknown word, an illegal date, etc.), only errors encountered while trying to process a valid request. For each such error we give the message typed out by AIRS, the corresponding error message that is written on the usage record (chapter 4.3) when the error happens, and an explanation.

NON-FATAL ERRORS - AIRS continues after giving message

CANNOT PROCESS REQUEST WHILE NEXT MONTHS DATA BEING PREPARED PLEASE TRY AGAIN LATER

Usage record: no message written - but since request not processed, will show low (usually 0) CPU time. Generally appears as series of 0 CPU times in ARO processing.

Explanation: Message is given when AIRS tries to write on the schedule file and finds it write-protected. Thus it can happen only during a request that modifies the schedules. The only commonly used requests that do so are ARO entries. The only times the schedule should be write-protected are: (1) during data base creation (chapter 2) or truncation (chapter 3) after program XTRACT announces "SCHED.CRF PROTECTED" (2) During running of batch failsafe job (chapter 5). If this message occurs but neither of the above data base processes is running, the file must have accidentally been left protected. For example, if the system crashed during the batch job and the job was not rerun, the file would be stuck protected. If you are absolutely sure the file should not be protected you can fix its protection by the command:

.PROTECT SCHED.CRF <337>

CANT TURN ON PAPER TAPE MODE
or
CANT TURN OFF PAPER TAPE MODE

Usage record: "*PTON" or "*NOECHO" if first message above, "*PTOFF" or "*ECHO" if second message above.

Explanation: See associated usage record messages in 4.3.

FATAL EPRORS - AIRS quits after giving message

CANNOT PROCESS DUE TO PROGRAM SPACE LIMITATION

Usage record: FOREST OVERFLOW

Exlanation: There is not enough room in the FOREST array to represent the request. The only cure is to modify AIRS.

Note: This is not a normal quit. After giving the message, AIRS continues trying to process the impossible request, and eventually dies with a monitor error, probably an ILLEGAL UUO.

COMPUTER MALFUNCTION DETECTED
PLEASE 'RUN AIRS' AGAIN

Usage record: This corresponds to several different messages on the usage record - any of the last 6 errors listed in section 4.3.

Explanation: This corresponds to several different problems. See 4.3 for the more specific usage record message and explanation of each problem.

FILE ERROR FOUND AND CORRECTED
PLEASE 'RUN AIRS' AGAIN

Usage record: UNIT u RAX ERR 7
FIXED TIME= n

Explanation: See associated usage record message in 4.3.

IFILE FAILED TO OPEN FILE MONTH.TAB FOR READING

Usage record: no message

Explanation: The file MONTH.TAB in 57041,1161 is either missing or has incorrect protection. Check this by the following command (while logged in under 57041,1161):

.DIR MONTH.TAB /P

If MONTH.TAB is not listed, it is missing. You must have it restored from failsafe tape (see chapter 6). If MONTH.TAB is listed but the protection is not <007>, fix the protection as follows:

```
.PROTECT MONTH.TAB<007>
```

IFILE FAILED TO OPEN FILE TABLS.RD FOR READING

Same as above for the file MONTH.TAB, except that TABLS.RD should have protection <057>.

OFILE FAILED TO OPEN FILE MONTH.TAB FOR WRITING

Usage record: no message

Explanation: The file MONTH.TAB in 57041,1161 has incorrect protection. Correct this as above ("IFILE FAILED...").

UNIT u RAX ERROR n

Usage record:

```
   UNIT u RAX ERR n
or
   UNIT u RAX ERROR 7
   CANT FIX
```

Explanation: In the first case, the given error on the given file was encountered. If the file is a scratch file, all will probably be well when AIRS is rerun, since scratch files are deleted when AIRS starts up. If the file is not scratch, either the error is spurious, in which case it will not repeat when AIRS is rerun, or there is actually trouble with the given file. In the second case, the unit will always be one of the permanent data base files. As in the "FILE ERROR FOUND..." message above, an input error was encountered, but after 10 tries AIRS still couldn't fix it. There may be trouble with the given file.

REFERENCES

1. "Airport Information Retrieval System (AIRS) System Design", Manuel Medeiros and Julie Sussman, Transportation Systems Center, Report No. FAA-RD-73-77, July 1973

2. "Airport Information Retrieval System (AIRS) User's Guide", Manuel Medeiros and Julie Sussman, Transportation Systems Center, Report No. FAA-RD-73-121, August 1973

3. "DECsystem-10 Assembly Language Handbook", Second Edition, Digital Equipment Corporation, 1972

4. "DECsystem-10 User's Handbook", Second Edition, Digital Equipment Corporation, July 1972

5. "Location Identifiers", FAA, ORDER 7350.1U, Jan. 1972 (or a later update of Location Identifiers if available)

6. "Official Airline Guide, North American Edition", Reuben H. Donnelley, published monthly

7. "Official Airline Guide, International Edition", Reuben H. Donnelley, published monthly

8. "Terminal User's Guide", First Data Corporation, Feb. 1973

APPENDIX A - RAX ERROR CODES

This appendix explains the error codes given by RAX, the random access package used by many of the programs in this manual.

1 (Unused)

2 Output attempted to file that was never opened

3 Input attempted from file that was never opened

4 Monitor I/O channels all busy

5 RAX I/O channels all busy

6 Error during output

7 Error during input. Probably a checksum error.

8 Error during closing of file

9 Blank file name

10 File owner identification not valid

11 File write protected

12 File read protected

13 File busy (another job has it open for writing)

14 Non-local file not found, can't be created.

15 Monitor channel can't be opened (disk not available?)

16 File just created can't be found

17 Unassigned file # referenced in call to RAXDFW/RAXIMW

18 Unassigned file # referenced in call to RAXWRT

19 Attempt to write on file opened in READ-ONLY mode

20 File being opened in READ-ONLY mode was not found

21 Buffer allocation error. Probably exceeded maximum allowed core.

APPENDIX B - CORRESPONDENCE BETWEEN FILES AND UNIT NUMBERS

Correspondence between files and unit numbers for each program in this manual that reads or writes files.
Files are divided into those accessed via the RAX random access package and those accessed via FORTRAN I/O.

AIRS

RAX

1	SCHED.CRF
2	U.nn
3	TERM.CRF
4	FLTS.nn
5	CHAIN.nn
6	CENT.CRF
7	PARAM.TRM
8	HASH.nn
9	TERM.FRE
10	CENT.FRE
20-..	10mm.nn
97	DATA.airport
98	CONT.airport, STDX.airport
99	ZONE.airport

FORTRAN

20	AIRS.MSG, MAIL, MONTH.TAB *.HLP, TABLS.RD, TABL2.RD

CCREF

RAX

1	SCHEDN.CRF, TERMN.CRF
2	PTR.TMP
7	PARAM.TRM

FORTRAN

20	CENTR.TAB, TABLS.RD

CHKSUM

RAX

1	CENT.CRF, CENT.FRE, PARAM.TRM SCHED.CRF, TERM.CRF, TERM.FRE

```
CNTCHK

   RAX

     1  AIRPT.RD

   FORTRAN

    20  CENTR.TAB, FORIN.TAB

COUNT

   RAX

     1  U.nn

CRFCHK

   RAX

     1  CENT.CRF, TERM.CRF

EXTCRF

   RAX

     1  TERMN.CRF
     2  CENTN.CRF

EXTEND

   RAX

     1  SCHED.CRF
     3  TERM.CRF
     6  CENT.CRF
     9  TERM.FRE
    10  CENT.FRE

FILES

   RAX

     1  ACFT.RD, AIRLN.RD, AIRPT.RD

   FORTRAN

    20  CENTR.TAB, CLASS.TAB, FORIN.TAB, TAXI.TAB
        TABLS.RD2, TABL2.RD2
```

```
FRECHK

   RAX

     1  CENT.CRF, TERM.CRF
     2  CENT.FRE, TERM.FRE

FREFIL

   RAX

     1  TERMN.CRF
     2  TERMN.FRE
     3  CENTN.CRF
     4  CENTN.FRE

MERGE

   RAX

     1  OSCHED.SRT
     2  ARRIVE
     3  SCHEDN.CRF

OSORT

   RAX

     1  SCRTCH
     2  OSCHED.OK
     3  OSCHED.SRT

PACKRD

   RAX

     1  SCHED.PAK
     2  FLIGHT.RD

   FORTRAN

    20  TABLS.RD
```

```
RUBINE

   RAX

     1  FLIGHT.RD
     2  AIRPT.RD
     3  ACFT.RD
     4  AIRLN.RD

   FORTRAN

    20  AIRBN.RD, MONTH.TAB
    21  SAVE.1, SAVE.2, SAVE.ERR

SCHCHK

   RAX

     1  SCHED.CRF

SORTRD

   RAX

     1  SCRTCH
     2  SCHED.PAK
     3  ARRIVE

TCREF

   RAX

     1  SCHEDN.CRF, TERMN.CRF
     2  PTR.TMP

XTRACT

   RAX

     1  SCHED.CRF
     2  OSCHED.OK
```

APPENDIX C - MONITOR COMMANDS AND SYSTEM PROGRAMS

The following is an alphabetical list of all operating system commands and system programs used in this manual. Where the command is an abbreviation, the full name follows in [ ]. Each command name is followed by a brief description of its use in this manual, and by references describing it. In the references, "R" stands for reference, "p" stands for page, and "TB" stands for Technical Brief (these are memos put out by First Data Corp.). For example, a writeup of the PLEASE command can be found on page 14 of Ref.8 (the "Terminal User's Guide") and on page 602 of Ref.4 (the "DECsystem-10 User's Handbook"). In most cases where two references are given, the first is more beginner-oriented and the second more formal and complete.

ASSIGN	assign name to tape drive	R8,p50 R4,p470
COED	edit definition & message files	R8,p35 TB61
DELETE	delete files	R8,p16 R4,p517
DIR [DIRECT]	list file information	R8,p7 TB16A
DISMOUNT	dismount failsafe tape	R4,p525
FAILSAFE	save files on mag. tape	
FILDMP	check schedule tape	TB48
KJOB	kill job - log off	R8,p26
LOG [LOGIN]	log in	R8,p1 R4,p590
MOUNT	mount failsafe tape	R4,p593
PIP	rewind schedule tape	R8,p48
PLEASE	talk to computer operator	R8,p14 R4,p602
PROTECT	change protection of file	R8,p6 R4,p616
R	run COED,FAILSAFE,FILDMP,PIP	R4,p632
REAS [REASSIGN]	give up claim to tape drive	R4,p633
RENAME	rename data base files	R8,p16 R4,p638

RUN	run user programs	R8,p23 R4,p646
SET BLOCKSIZE	set mag. tape buffer size	R4,p653
SUBMIT	submit batch job	R4,p680
SYS [SYSTAT]	system status	R8,p13
TYPE	type out a file	R8,p20 R4,p702

APPENDIX D - NAMES, ADDRESSES, AND PHONE NUMBERS

ARO (Airport Reservation Office)

Phone: 202-426-8923

CFCF (Central Flow Control Facility)

Phone: 202-426-3797

First Data Corporation

400 Totten Pond Road
Waltham, Mass. 02154

Phone: 617-890-6701

Reuben H. Donnelley Corp.

2000 Clearwater Drive
Oakbrook, Illinois 60521

Phone: 312-654-4000

Contact:Chuck Cesal or Mary Kastory

After processing Reuben Donnelley tape, mail it to:

Mr. Eugene Falsetti
ATC Systems Command Center
FAA/AAT 370 Room 626
800 Independence Avenue, S.W.
Washington, D.C. 20590