TABLE OF CONTENTS

1. Introduction 1
2. cmenu 3
   2.1 cmenu Overview 3
   2.2 Key-letters 4
   2.3 Browse Function 5
   2.4 Detailed Inventory Browse 6
   2.5 Select Data and Indices 7
   2.6 Select Programs and Documents 9
   2.7 Copy Selections 10
   2.8 Batch Mode 13
   2.9 cmenu and gdat 14
3. Simplest Task 15
   3.1 Identifying a Data Volume 15
   3.2 Copying Mass Store Volumes 17
   3.3 rdtp 18
   3.4 is2asc and back 22
4. Geophysical Indices and Data Subsetting 24
   4.1 getndcs 24
   4.2 gdat use 27
   4.3 gdat and apxntrp Installation 34
5. Plot Programs 37
   5.1 pltcyc 38
   5.2 pltvst 39
   5.3 pltvsh 43
   5.4 binhnt 48
6. Programming 55
   6.1 Subroutine Library 57
   6.2 Program Modification 62
   6.2.1 binhnt Overview 63
   6.2.2 binhnt Modification 64
   6.2.3 gdat Overview 66
   6.2.4 gdat Modification 68
   6.3 Models 69

Appendix A: UNIX Help 72
   A.1 Login Procedures 72
   A.2 Computing Environment 73
   A.3 UNIX Introduction 74
   A.4 Remote File Transfer 77

A.5 Visual Editor 78
A.6 References 82
Appendix B: Logical Data Record Format 83

CEDAR DATABASE USER GUIDE

R. Barnes
National Center for Atmospheric Research

November 1998

CHAPTER 1

INTRODUCTION

This describes use of the CEDAR (Coupling Energetics and Dynamics of Atmospheric Regions) database at NCAR. The database consists of the data archive, summary information, access programs and computer. Readers are assumed to have a login on the database computer (or be using database software elsewhere). The computer is accessible on the internet via the NCAR gatekeeper computer; login procedures are described in Appendix A. Services provided include:

- Browse current versions of the inventory tables.
- Display data with various print and plot utilities.
- Display geophysical indices.
- Select data and convert to ASCII character "flat file" format.
- Transfer data to another computer over the internet.
- Write programs either using the executable subroutine library or ANSI Fortran-77 source code.
- Transfer and install access programs on another computer. Some database plot programs use the GKS based NCAR Graphics library which must be purchased separately. IDL (Interactive Display Language) is also used for some plotting.

Conventions used in this text mimic those employed in **man** pages. **Bold** text is to be taken literally. *Italics* imply a variable which is to be replaced by a supplied value. Arguments bounded by brackets, e.g., '[*arg*]' are optional. Arguments separated by a vertical bar '|' are exclusive; e.g., '[*arg1* **|** *arg2*]' means either *arg1* or *arg2* but not both. Arguments followed by an ellipsis '...' may be repeated.

Except for cmenu, a basic knowledge of the UNIX operating system is required. If not UNIX literate, the tutorial program **learn** will be instructional. The command summary in Appendix A may also be helpful. Reference material is available via **man** pages, for which an introduction may be obtained by entering **man man**. Thereafter, try **man learn** before invoking **learn**. A list of documented locally developed database commands may be obtained by entering **mancedar** (no intervening space). With the exception of the menu driven utility (**cmenu**), it is eventually necessary to use an editor. The choices are: the standard UNIX editor **vi**, **emacs**, or **re**. Each is described in **man** pages and a summary of **vi** commands is in Appendix A.

Related database documents include:

CEDAR DATABASE FORMAT	describes the details of the binary and character versions of the format. The construction of logical records is defined. Codes are defined for instruments, parameters, record types, etc.
CEDAR DATABASE CATALOGUE	published annually, summarizes the archive and describes how to request data or an account on the database computer.
AKAVSNS	(database volume aliases) Historically, the oldest form of archive documentation, this table contains a brief text description of each volume including caveats.

These are available on cedar in directory **~bozo/is/docs**. The format description is in two forms, PostScript or plain ASCII character directives (**cedar.fmt.ps, cedar.fmt.chr**). The catalogue tables (**catprt.list, invpxt{.fpi,.isr,yr}.out, parcods.list**) are updated more frequently than the annually printed version which is also available (**tables.***yr***, text.***yr*). Most of this information is also available on the World Wide Web (**http://hao.ucar.edu/public/research/tiso/cedar/cedar.html**) and a list of related web sites is in **cedar.urls**.

The database appearance can be affected by the limited amount of disk space on cedar and the NCAR computing environment. The complete database archive resides in the mass store system from which portions are copied to cedar into directory **/usr/madrigal/files/stage**. Since 1994, there has been sufficient disk space that the entire database is on on cedar, however, with future growth it may be necessary to reinstitute the copy procedure.

Users are initially allocated a nominal amount of permanent space (2 MB) in their home directory (**/home/cedar/***your.login*) which is backed-up to avoid loss in the event of a disk failure. A larger quota may be allocated when a need is demonstrated. Additional permanent file space with no quota is available in the **/local/d/***your.login*, although it is not backed-up.

Users with logins on HAO computers other than cedar will discover that home file systems are mounted on most other machines, thus making the database programs available. However, the data are not commonly mounted in this manner, so it is often necessary to **rlogin cedar** to gain access. Once logged-in to cedar, both the data directory and home directories are accessible.

The next chapter describes the **cmenu** utility for subsetting and downloading data which requires no knowledge of UNIX or the computer systems at NCAR. Also capable of providing a read program and format description, cmenu is sufficient for those wishing to analyze data on their own home computer.

For those wishing to do more work on cedar, Chapter 3 describes ways to select a dataset and display it. Chapter 4 introduces a program to display geophysical indices (**getndcs**), and a program to generate flat-file data subsets (**gdat**). Chapter 5 covers other plot utilities. Chapter 6 describes how to write programs incorporating database software and how to customize database programs. Appendix A summarizes login procedures, account setup parameters, and introduces the UNIX operating system commands. Appendix B, a description of the logical data record organization, is an excerpt from the **CEDAR Database Format** and included for convenient reference. Appendix C describes MADRIGAL, an alternative data access procedure provided by Millstone Hill personnel. MADRIGAL's isprint will generate flat-file character subsets, and is analogous to **gdat** but with additional options.

If you encounter problems, consulting assistance is available by contacting Roy Barnes (303/497-1230 or **bozo@ucar.edu**). Problems with the operating system should be described in an email sent to **trouble@cedar.hao.ucar.edu**.

Users may be interested in a review of the instruments and models in the CEDAR Data Base at URL: **http://www.haystack.edu/cedardb.html**. The CEDAR web page **http://www.hao.ucar.edu/public/research/tiso/cedar/cedar.html** has a copy of most database documents, including the current catalogue. Also of potential interest are the time series plots. Now availabile as a prototype is the a web data access; it will soon be in full production (perhaps eliminating the need for this document).

CHAPTER 2

CMENU

The easiest way to obtain software, model output, indices or other data, is to use the cmenu utility. cmenu does not require detailed knowledge of UNIX or an interactive editor. There are interactive and batch modes. Neophytes can rely on the prompts and help summaries to guide them them through an interactive session; yet with experience they will prefer the convenience of cmenu's batch mode for data access.

For those already familiar with cmenu, this version has some notable differences from the original. Interactive sessions employ a different file browse utility (to better accommodate window size changes). Shortcuts are now possible: At most interactive prompts one may enter any key-letter argument pair, including unsolicited parameters. The addition of key-letter **-a** (display all current settings) simplifies setting up or switching to batch mode. Now all settings, including prior batch executions, are remembered between sessions. One feature (main menu: Leave message) was eliminated; no-one used it.

In addition to the following summary, an on-line description is available via the "man page", which may be accessed by entering:

source ~bozo/is/bin/envcdb
man -F cmenu

when on the cedar computer; see Appendix A.

2.1. cmenu Overview

An interactive session is started by entering

source ~bozo/is/bin/envcdb
cmenu

The main menu is then displayed as:

```
                CEDAR DATABASE PROMPT SYSTEM MAIN MENU
Browse inventories and extract data, programs and documents (enter J for more)

     Browse general data summaries
  A.   All instruments (and models) monthly
  B.   Incoherent Scatter Radar combined daily
  C.   Fabry-Perot combined daily

  D. Browse parameter codes
  E. Browse instrument (kinst) codes

  F. Detailed inventory browse - test effect of data/indices selection
     specifications (time interval, kinst codes, kindat codes)

  G. Select data and indices
  H. Select programs and documents

  I. Copy selections (to a permanent location)

  J. Help
  K. Quit (also try 'q' or interrupt - usually 'control c')

  Enter the desired letter (plus any key-letter options):
```

The top items (**A-F**) display various database summaries and may be used to identify available data. The next two items select data (**G**) and select programs and documentation (**H**). All selections, including optionally saved browse summaries are held temporarily until using Copy selections (**I**). The next item (**J**) is a help summary. Last (**K**) is a graceful means of quitting, for which synonyms are **Q** or **q**; main menu item selection is case insensitive. Abrupt termination also results from an interrupt (usually 'control-c'), but it is immediate, whereas entering **k** first checks to see if any files have been accumulated during the session and issues a reminder with an offer to start Copy selections.

Browse data summary items are ordered in increasing detail. **All instruments** (**A**) displays which months have data. The next two (**B** and **C**) display dates with data for two popular types of instruments. And the Detailed inventory browse (**F**) displays hours with data for all instruments.

The **Parameter codes List** (**D**) and **Instrument codes list** (**E**) are prominently displayed with the data summaries because they are heavily used in the inventory browse and data selection. To avoid repeated references to these tables, it is recommended that a copy of each be obtained prior to running the detailed inventory browse or selecting data. Note that these codes lists are also published in the annual Catalogue and are available in the public ftp directory (**/jabba/ftp/archive/cedar**).

The **Detailed inventory browse** (**F**) and **Select data and indices** (**G**) use the same information: time interval, instrument codes, and kind-of-data (kindat) codes). One uses these criteria to subset the database inventory and the other subsets the actual data. These criteria are remembered to facilitate iterations when searching. For instance, one could initially browse the inventory for a given time period and all instruments, then refine criteria when selecting the data; eliminating unwanted record types by requiring specific instrument codes and kind-of-data (kindat) codes. The detailed inventory lists continuous time intervals, spatial extent and parameters for each available record type (defined uniquely by the combination of instrument and kindat code).

Select programs and documents (**H**) displays a menu of various database documents, access programs, and models. Any files chosen are held temporarily until **Copy selections** (**I**) is used for disposition. Copy selections supports copying to a local permanent directory, or copying to a remote host (via UNIX **rcp** or **ftp**).

2.2. Key-letters

With experience, one may wish to shortcut prompting. A consistent set of key-letter argument pairs are recognized throughout this utility; e.g, **-t 19960101** selects everything on 1 January 1996. These key-letter argument pairs may be entered on the command line (at the outset), at the main menu, and whenever dash arguments are parsed; viz., the Detailed inventory browse, Select data, Select programs and Copy selections. On these occasions, it is possible to complete the session with no further prompting by entering the appropriate combination to define the selection and file disposition. These options are remembered too; for example, if files are always put in the same permanent location (host, login and directory), the file disposition parameters need not be re-entered. The key-letter argument pairs are case sensitive; they include

Batch processing:	
-b	begin batch processing; i.e., use the current key-letter argument settings to finish the current session without more prompts
Data selection:	
-t *yyyymmdd[:hhmm]-yyyymmdd[:hhmm]*	time interval
-i *kinst1,kinst2,...*	instrument codes
-k *kindat1,kindat2,...*	kindat codes
-h *hcropt*	header/catalog records [**0-3**]
-f *format*	data format [**f, c, b, fc, cc** ,or **bc**]
-n *name*	data file name prefix
-s *size*	max Mbytes per data file
-m *mxmbt*	max Mbytes total

	Detailed inventory browse:
-v *verbosity*	parameter code verbosity [**0-2**]
	Permanent file disposition:
-r *remote host*	destination computer name or IP address
-u *user*	login
-p *passwd*	password
-d *dir*	directory
	Interactive option:
-q	(quiet) shut off initial display of browse function options
-a	display current key-letter arguments

Detailed descriptions of these arguments are provided in the help display available after starting the relevant main menu item. For instance, requesting help (entering **H**) after starting the Detailed inventory browse produces a display defining key-letters: **-t -i -k -h** and **-v**. The following sections also describe their use.

2.3. Browse Function

The browse function is used for all summary displays. It uses the notion of a table header, page header and the table body. Unless turned off by the **t** option, the table header (if any) is displayed prior to each page header which is displayed for all options except: Forward 1 line, Forward half screen. The following commands are recognized when browsing:

COMMAND	**EFFECT**
h, H	Display a help summary
q, Q	Quit browse
f, space	Forward one screen
b	Back one screen
d	Forward half screen
u	Back half screen
e, j, RETURN	Forward 1 line
y, k	Back 1 line
r	Redraw screen
t	Toggle off/on table header display
number	Go to line *number* (or as close as possible). When *number* is preceded by '+' or '-', the move is relative to the current line (bottom of screen)
/pattern	Search forward in text body for line containing */pattern*
?pattern	Search backward in text body for line containing */pattern*
n	Repeat previous search

Some displays exceed 130 characters width, so to avoid wrap-around it may necessary to increase the window width. The browser accommodates such adjustments, although the change does not take effect until the next command after the window change.

Upon quitting the browse display, the prompt:

```
Save this browse file (catprt.1 (2,042 bytes)? (y/n)
```

offers to retain the summary table. If the response is **y**, a reminder is issued to later use main menu **Copy selections** to save this file in a permanent location.

2.4. Detailed Inventory Browse

Selecting main menu item **F**, starts the detailed inventory browse. This displays a table where each line summarizes a continuous period for one type of data record; i.e, gaps are less than 6 hrs. A data record type is defined by the combination of instrument (kinst) and kind-of-data (kindat) codes for which there is a fixed parameter list. Some or all of these parameters are displayed on each line along with the time interval. Each line is part or all of an inventory record. An inventory record is displayed on multiple lines when there are gaps 6 to 48 hours duration. Longer gaps always start a new line and inventory record. The intermediate length gaps are evident when minutes of the hour are blank (because the exact time of the hour is not retained). Inventory records are ordered by kinst, begin date, and kindat.

Between the time interval and the parameters list are fields summarizing spatial extent and resolution. The cycle time is an estimate; it is relevant to a repeated sampling sequence such as radar operation, but not TIGCM model output of the following example. Spatial parameters are incomplete for many instrument types, such as Fabry-Perots where height is inferred. In the example, magnetic latitude is omitted because the time consuming magnetic latitude calculation is not justified when the model output spans the entire globe.

BEGIN		END				CYCLE	MIN TIME	MIN VERT	HEIGHT (KM)		MAG LAT (DEG)			
DATE	TIME	DATE	TIME	KINST	KINDAT	TIME (MIN)	RES (MIN)	RES (KM)	MIN	MAX	MIN	MAX	VSN ALIAS (AKAVSN)	PARAMETERS
790321	0000	790321	2400	310	11031		60.0	5.0	100	500			GCM760320A	f107a epow pcp bygsm tn vnep1 vnnp1
790321	0000	790321	2400	310	11031		60.0	5.0	100	500			GCM760320C	f107a epow pcp bygsm tn vnep1 vnnp1
790321	0000	790321	2400	310	11032			5.0	100	500			GCM760320B	f107a epow pcp bygsm tna tn24a tn24p tn12a tn12p tn08a tn08p tn06a tn06p vnea vne2a1 vne24p vne12a vne12p vne08a vne08p vne06a vne06p vnna vnn2p1 vnn24p vnn12a vnn12p vnn08a vnn08p vnn06a vnn06p
790321	0000	790321	2400	310	11032			5.0	100	500			GCM760320C	f107a epow pcp bygsm tna tn24a tn24p tn12a tn12p tn08a tn08p tn06a tn06p vnea vne2a1 vne24p vne12a vne12p vne08a vne08p vne06a vne06p vnna vnn2p1 vnn24p vnn12a vnn12p vnn08a vnn08p vnn06a vnn06p

This example is the result of a search for TIGCM model output (kinst=310) and was obtained using selection criteria

-t 19790318-19790325 -i 310 -k 11031,11032,11040

Actually, only the instrument code was given on the first iteration; subsequent refinements eliminated extraneous times and record types. When first starting the detailed inventory browse, the most recent selection criteria and the browse file size is displayed:

```
Selecting inventory records matching current criteria ...


Applying the following criteria produces an inventory summary containing

          17 lines

 Time interval: 19790301:0000 - 19790331:2400
        KINST: 310 =  NCAR TGCM/TIGCM Model Output
       KINDAT: (any)
     VERBOSITY: 0

CHANGE inventory record selection by entering at least one of

   -t yyyymmdd:hhmm-yyyymmdd:hhmm -i kinst1,... -k kindat1,... -v verbosity

ACCEPT selection by pressing 'Return' (or 'h'= help; 'r'=return; 'q'=quit)
>>-->
```

Prompts for key-letter argument inputs work by confirmation: Current values are displayed, then input is requested. If there is a disagreement on syntax of the response, a diagnostic is issued and the opportunity is given to try again. Prompts continue until until no changes are made; i.e., response is only a carriage return. Thereafter, the browse function commences the table display.

Recognized key-letter argument pairs include:

-t *yyyymmdd:hhmm-yyyymmdd:hhmm*
: Time interval. Dates are 8-digits: a 4-digit year, then month and day. Appending a time of day (:hhmm) is optional, defaulting to 0000 (begin) or 2400 (end); minutes may be omitted. If only one date is provided the other is assumed equivalent; e.g., **-t -19920102:12** is 00:00 to 12:00 UT 2 Jan 1992. If no time interval is given, all times are acceptable.

-i *kinst1,kinst2,...* Acceptable instruments. Data must match the specified instrument code(s). If no codes are entered, any instrument is acceptable. The instrument codes list is a main menu browse option.

-k *kindat1,kindat2,...* Acceptable kindat codes. Data must match the specified kind-of-data code(s). kindat codes are intended to identify the data processing algorithm. They are defined separately for each instrument; i.e., they may not be unique unless combined with kinst code. kindats are used effectively to exclude some data types after an initial browse (to identify all kindat codes). If no codes are entered, any kindat code is acceptable.

-v *verbosity* Which types of parameter codes are included and which version is used.

-v0 (default) omits parameters related to time, position, and data quality
-v1 omits time and position
-v2 omits nothing.

Two versions of the parameter codes may be listed: alphabetic (default) and integer. To display the integer version, append an **i** to the verbosity level; e.g., **-v0i**. Both versions of parameter codes are defined in the main menu item: **Parameter codes list**.

Entering a key-letter with no argument resets it to default. A space between the key-letter and the argument is optional.

Selection criteria are remembered. They may have been defined in a previous session or numerous places during this session. Prompts for selection criteria are issued here and in the main menu item: Select data. In both places, the response is verified via prompts. Selection criteria may be entered elsewhere too; any time there is a prompt for dash key-letter arguments and at the main menu prompt. At those places, one may display the status of all key-letter arguments by entering **-a** (for all arguments).

2.5. Select Data and Indices

Selecting main menu item **G**, starts Select data and indices. Prompts for a time interval, kinst (instrument) codes, and kind-of-data (kindat) codes are the same as in the Detailed inventory browse.

```
Scanning the inventory with current selection criteria ...

Applying the current record selection criteria will produce approximately
      12,000,000 bytes (if formatted flat character)
      21,000,000 bytes (if formatted CEDAR Database character)
       6,900,000 bytes (if formatted CEDAR Database binary)

         Time interval: 19790301:0000 - 19790331:2400
                 KINST: 310 =  NCAR TGCM/TIGCM Model Output
                KINDAT: 11031,11032,11034
HEADER/CATALOG RECORDS: 2

CHANGE data selection by entering at least one of

   -t yyyymmdd:hhmm-yyyymmdd:hhmm -i kinst1,... -k kindat1,... -h header opt

ACCEPT selection by pressing 'Return' (or 'h'= help; 'r'=return; 'q'=quit)
>>-->
```

The first portion of the display is an estimate of the amount of data requested. Amounts are given for each of three basic formats: flat character, database character, or database binary. Flat character is usually the largest volume but easiest to use. It is made up of ASCII characters and has no hierarchy; i.e., each line contains a value for each variable. Database character reduces the volume by applying some hierarchy but retains easily viewable ASCII characters. Database binary is the most compact, replacing ASCII with 16-bit integers. Binary is also most efficient for program input/output execution time, but binary format requires more effort to install access programs than character formats. The database formats document and Fortran access programs may be obtained from the main menu item: Select programs and documents. Data selection is at the logical record level; i.e., the smallest amount one may select is a logical record. Hence, the format description may be a useful reference to understand selection even when choosing the flat character format.

The new key-letter (**-h**) is the header/catalog option:

-h *header opt* defines which of these documentation record types are to be included during data selection. Choices are

- **-h0** none
- **-h1** (default) those bounded by the time interval
- **-h2** all header/catalog records encountered in data volumes containing the requested data records.
- **-h3** only header/catalog records, excluding all data records

After resolving the data and header record selection criteria, a second screen is displayed prompting for output file name, format and maximum file size.

```
Selected data will be formatted flat character and named

   data.flt

If more than

   100 Mbytes

are selected, a new output file will be named by inserting a unique number
(the second file would be 'data2.flt').

CHANGE THE NAME prefix, SIZE limit, or write FORMAT by entering at least one of

   -n name -s size -f format

ACCEPT THE NAME,SIZE,FORMAT by pressing 'Return'  (or 'h', 'b', 'r', 'q')
>>-->
```

The new key-letters are:

-n *name* The name prefix to use when writing data. Appended to *name* may be a number, to insure uniqueness, followed by a suffix, to identify format:

- **.flt** flat character data
- **.fld** flat character documentation
- **.hcr** header or catalog records accompanying **.flt** files when hcropt>0
- **.chr** database character
- **.cbf** database binary
- **.doc** notes about database files read (extracted from akavsns)

-s *size* Limit the (uncompressed) size of an output data file to *size* MBytes (10**6 bytes). If no size is given, it becomes 100 MBytes.

-f *format* Define the output data format:

- **-ff** (default) flat character

-fc CEDAR database character version
-fb CEDAR database binary version

Compression is indicated by appending **c** to the format letter; e.g., **-ffc** means compressed flat file format. If no format letter is given, the format defaults to flat character. When files are compressed, names are altered to conform with UNIX convention by appending **.Z**. DOS convention is followed by omitting the other '**.**' from the suffix; e.g., **data.flt** becomes **dataflt.Z**. The size option does not adjust when compression is selected because the effect is not easily predicted: Often it is about 50% reduction but it can exceed 90%. Thus, one might conservatively compensate for this effect by doubling the maximum size.

Once these output criteria are satisfactorily defined, data selection starts. A warning is issued when a large amount of data are to be selected or it may take a while; e.g.,

```
WARNING:  Data selection may take roughly 5 minutes.  Proceed? (y/n)
```

Messages are issued when data selection commences and completes:

```
Starting data selection ...
        writing: data.fld  (flat character documentation)
        writing: data.hcr  (header/catalog records)
        writing: data.flt  (flat character data)
        writing: data.doc  (notes regarding input files)
```

Before returning to the main menu, a reminder is issued to use **Copy selections** to save the temporary files in a permanent location.

2.6. Select Programs and Documents

The main menu option to select programs and documents displays a menu of choices:

```
                  SELECT SOURCE CODE AND DOCUMENTS MENU
    A. Access Program:   Database Character Format
    B. Access Program:   Database Binary    Format
    C. Model:   International Geomagnetic Reference Field
    D. Model:   Apex magnetic coordinates
    E. Model:   International Reference Ionosphere (IRI-90)
    F. Model:   Neutral composition (MSIS-90)
    G. Model:   Neutral winds (HWM-93)
    I. Model:   Conductances (HPI)
    J. Model:   Electric Potential (Richmond's quiet-day solar min)
    K. Model:   Electric Potential (HMR)
    L. Model:   Electric Potential (IZMEM)
    M. Model:   Electric Potential (MHI/MHS)
    N. Model:   Electric Potential (WEIMER)
    O. Document:   CEDAR Database Format: PostScript
    P. Document:   CEDAR Database Format: Character
    S. Document:   CEDAR Database User Guide: PostScript
    T. Document:   CEDAR Database User Guide: Character

ADD    to      SELECTIONS by entering letters matching desired items
ACCEPT current SELECTIONS by pressing Return key
OR ENTER:  'r' (immediate return main menu)  'h' (help) 'q' (quit)
>>->
```

Some menu items consist of multiple files. When the number of files becomes confusing, a **readme** file is included to introduce the others.

To select an item, enter the corresponding letter, either lower or upper case. To remove selections, enter an exclamation (!) followed by the letter(s); e.g, **! bc** removes previously selected items **B** and **C**. Selections are held in a temporary location; use the main menu option **Copy selections** to put them in a permanent directory.

Source code is Fortran-77 compatible; such files are indicated by a **.f** suffix. Some are subroutines with a separate driver program while others are complete programs with separate input job control files. All models include an example with associated input and output. Some models include IDL source and output graphics. Embedded comments are frequently the only documentation, however, journal references are included when available. See also the description of **rdtp** in Chapter 3 and models in Chapter 6. Operating system specific installation instructions are given when appropriate. Reproduction of the test case results is the first indication of successful installation, however, test cases seldom exhaustively exercise all logic paths.

Documentation files come in two forms, ASCII character and PostScript. The PostScript version should be selected when possible because typsetting nicities are lost when translating to a simple character set. PostScript versions may require cut-and-paste after printing if the figures are separate or when the table of contents is printed last.

Database file names follow UNIX convention for suffixes by default, but may be shortened to comply with DOS convention (maximum eight characters plus an optional suffix consisting of '.' and three characters).

2.7. Copy Selections

The main menu option to dispose accumulated files to a permanent location starts by displaying a list of accumulated files and then prompts for permanent disposition key-letter argument pairs:

```
The following files have been accumulated:
  Character:  data.doc       1,746
              data.fld       6,714
              data.flt  74,294,352
              data.hcr     462,900
                        74,765,712 bytes

Currently rhost = huron.scd
           user = bozo
         passwd =
            dir = /huron/ws/bozo

CHANGE COPY CRITERIA by entering at least one of

      -r rhost -u user -p passwd -d dir

ACCEPT COPY CRITERIA by pressing 'Return' (or 'h'= help; 'r'=return; 'q'=quit)
>>-->
```

The new key-letters are:

-r *rhost* — *rhost* is the name of the computer to receive files. Default is the current host.

-u *user* — *user* is the login on *rhost*. Default is the current login.

-p *passwd* — *passwd* is the password for *user* on *rhost*. When *passwd* is given, ftp (file transfer protocol) is used to deliver files. If no password is given or it is reset by entering only **-p**, then either UNIX rcp (remote copy) or mv (move) is used depending on *rhost*. Beware of snoops: *passwd* is displayed when it is entered. It is not necessary to add apostrophes before and after a password containing special characters (to protect the shell from interpreting them; i.e., globbing is off).

-d *dir* — *dir* is the directory to which files will be delivered. If dir is not an absolute path (beginning with '/') then it is assumed to be relative to *user*'s home directory on *rhost*; e.g., for user **bozo** whose home directory is **/home/cedar/bozo**, **-d wk1** means **/home/cedar/bozo/wk1**.

Entering a key-letter with no argument resets it to default.

The mechanism for file transfer (ftp, rcp, or mv) is determined by the combination of *rhost* and *passwd*. When choosing this mechanism, security may be a consideration. For remote file transfer, ftp is universally accepted but it requires a password to be sent to the remote machine which is a security risk. rcp is an alternative on UNIX systems which avoids this, but rcp requires prior set up of permissions (**~/.rhosts** file; see **man rsh**). Another issue can be the transfer rate. ftp is faster than rcp at least on some systems and, thus, preferred when moving larger amounts.

Traditionally, PC's are not configured to allow remotely initiated **ftp** (put), so, when connecting from a PC, it is probably necessary to save cmenu output files on local disk then later run ftp (get) from the PC; it will be necessary to use the gatekeeper as a proxy (see Appendix A for details). If the quota on your home directory on this computer is too restrictive to hold cmenu's output, use **/d/***your login* or **/tmp** to hold local files output from cmenu. **/tmp** is scrubbed daily, but **/d** has no automatic removal mechanism, so be kind to other users and remove old files you no longer need.

When the file disposition arguments are satisfactory and the response is only a carriage return, a message is displayed indicating the activity:

```
Initiating remote copy (rcp) to huron.scd:/huron/ws/bozo/ ...
```

Thereafter, control is returned to the main menu.

In the example, four files were transferred, the rather large data file and others documenting its contents:

data.hcr contains header and catalogue records from the original datasets accompanying the selected data. These contain descriptions of the the data taking, the data analysis, and define database data record layout.

data.doc contains any notes provided with the original datasets and may contain caveats qualifying the data. This is a catch-all for information not otherwise recorded in the dataset. In this case three datasets were involved:

```
GCM760320A Specific Notes
  TIGCM (TIGCM 8) model output for 18 conditions: (winter, equinox, summer)
  * (low, medium, high magnetic activity) * (low, high) solar activity.
  KINDAT 10s digit reflect the magnetic activity 30, 60 or 90 kV and the
  date reflects the season and solar activity.

GCM760320B Specific Notes
  TIGCM (TIGCM 8) model output harmonic analysis version for 18 conditions:
  (winter,equinox,summer) * (low,medium,high) magnetic activity * (low,high)
  solar activity.  KINDAT 10s digit reflect the magnetic activity 30, 60 or
  90 kV and the date reflects the season and solar activity.

GCM760320C Specific Notes
  TIGCM model output for every 15 degrees longitude and 12 combinations:
  (winter,equinox,summer) * (low,medium) magnetic activity * (low,high)
  solar activity.  Two record types: winds and temps (files 1-12) and
  harmonic analysis (files 13-24).  This is similar to GCM760320A and
  GCM760320B, except they are only for 70 W longitude (but also include
  the additional case of high magnetic activity).  More documentation in
  directory ~bozo/is/docs/tigcm includes a README file, a paper and plots.
```

data.fld defines **data.flt**'s format:

```
----INTRODUCTION
This describes the accompanying "flat character" format data.
If only one record type was selected, then each record (line)
in the data file has the same number of six character integer
fields and each field occupies the same location, containing
the same parameter's values.  All values are in the range
-32768 to 32767 (an artifact of the database binary format).
Two special values are noteworthy: missing data are represented
by -32767; and an assumed value is indicated when it's error
value is -32766.

The order, position and content of each field in the record is
described in the following table.  The entry labeled "DB Format
Section" identifies the original location in the Database
format record.  Infrequently, the "Description" is not known
because a non-standard parameter code has been used.  In this
case, a definition should be available in Database "header"
records (which may also be extracted (use the "-h" option when
```

```
specifying selection criteria).

A record type is defined by the combination of instrument
(KINST) and kind-of-data (KINDAT) codes.  For each record type,
a "RECORD TYPE DESCRIPTION" is provided.  If multiple types are
interleaved in an undesirable manner, they can be reordered by
sorting the data file.


----BEGIN RECORD TYPE DESCRIPTION:
 Field  Begin  DB Format
  No.  Column   Section    Code          Description
   1       1   prologue   KINST =   310  NCAR TGCM/TIGCM Model Output
   2       7   prologue  KINDAT = 11031
   3      13   prologue    IBYR = Begin year (4-digit) - UT
   4      19   prologue    IBDT = Begin month and day of month (mmdd)
   5      25   prologue    IBHM = Begin hour and minute of day (hhmm)
   6      31   prologue    IBCS = Begin centisecond of the hour
   7      37   prologue    IEYR = End year (4-digit)
   8      43   prologue    IEDT = End month and day of month (mmdd)
   9      49   prologue    IEHM = End hour and minute of day (hhmm)
  10      55   prologue    IECS = End centisecond of the hour
  11      61   1D array     160 = Geodetic latitude of measurement (deg * 10**-2)
  12      67   1D array     170 = Geodetic longitude of measurement (deg * 10**-2)
  13      73   1D array      44 = Local solar time (hour * 10**-3)
  14      79   1D array      42 = Local solar time diff (=SLT-UT) +E lon (hhmm)
  15      85   1D array      21 = Day number of year (universal time) (day)
  16      91   1D array     350 = F10.7 solar flux (Sa) (W/m2/Hz * 10**-23)
  17      97   1D array     365 = Estimated Hemispheric Power Input (W * 10**8)
  18     103   1D array    2301 = Polar cap potential difference (V * 10**1)
  19     109   1D array    2206 = Interplanetary Mag Field By GSM (T * 10**-11)
  20     115   2D array     110 = Altitude (height) (km)
  21     121   2D array     810 = Neutral temperature (K)
  22     127   2D array    1411 = Direction 1 Neutral wind (eastward) (m/s * 10**-1)
  23     133   2D array    1421 = Direction 2 Neutral wind (northward) (m/s * 10**-1)


----BEGIN RECORD TYPE DESCRIPTION:
 Field  Begin  DB Format
  No.  Column   Section    Code          Description
   1       1   prologue   KINST =   310  NCAR TGCM/TIGCM Model Output
   2       7   prologue  KINDAT = 11032
   3      13   prologue    IBYR = Begin year (4-digit) - UT
   4      19   prologue    IBDT = Begin month and day of month (mmdd)
   5      25   prologue    IBHM = Begin hour and minute of day (hhmm)
   6      31   prologue    IBCS = Begin centisecond of the hour
   7      37   prologue    IEYR = End year (4-digit)
   8      43   prologue    IEDT = End month and day of month (mmdd)
   9      49   prologue    IEHM = End hour and minute of day (hhmm)
  10      55   prologue    IECS = End centisecond of the hour
  11      61   1D array     160 = Geodetic latitude of measurement (deg * 10**-2)
  12      67   1D array     170 = Geodetic longitude of measurement (deg * 10**-2)
  13      73   1D array      42 = Local solar time diff (=SLT-UT) +E lon (hhmm)
  14      79   1D array      21 = Day number of year (universal time) (day)
  15      85   1D array     350 = F10.7 solar flux (Sa) (W/m2/Hz * 10**-23)
  16      91   1D array     365 = Estimated Hemispheric Power Input (W * 10**8)
  17      97   1D array    2301 = Polar cap potential difference (V * 10**1)
  18     103   1D array    2206 = Interplanetary Mag Field By GSM (T * 10**-11)
  19     109   2D array     110 = Altitude (height) (km)
  20     115   2D array     937 = Mean neutral temperature (K * 10**-1)
  21     121   2D array     942 = 24-h neutral temperature amplitude (K * 10**-2)
  22     127   2D array     947 = 24-h max neutral temperature phase (hr * 10**-3)
  23     133   2D array     952 = 12-h neutral temperature amplitude (K * 10**-2)
  24     139   2D array     957 = 12-h max neutral temperature phase (hr * 10**-3)
  25     145   2D array     962 = 8-h neutral temperature amp (K * 10**-2)
  26     151   2D array     967 = 8-h max neutral temperature phase (hr * 10**-3)
  27     157   2D array     992 = 6-h neutral temperature amplitude (K * 10**-2)
  28     163   2D array     997 = 6-h max neutral temperature phase (hr * 10**-3)
  29     169   2D array     935 = Mean eastward neutral wind (m/s * 10**-2)
  30     175   2D array     943 = 24-h eastward neutral wind amplitude (m/s * 10**-1)
```

```
31      181    2D array     945 = 24-h max eastward neutral wind phase (hr * 10**-3)
32      187    2D array     950 = 12-h eastward neutral wind amplitude (m/s * 10**-2)
33      193    2D array     955 = 12-h max eastward neutral wind phase (hr * 10**-3)
34      199    2D array     960 = 8-h eastward neutral wind amplitude (m/s * 10**-2)
35      205    2D array     965 = 8-h max eastward neutral wind phase (hr * 10**-3)
36      211    2D array     990 = 6-h eastward neutral wind amplitude (m/s * 10**-2)
37      217    2D array     995 = 6-h max eastward neutral wind phase (hr * 10**-3)
38      223    2D array     936 = Mean northward neutral wind (m/s * 10**-2)
39      229    2D array     948 = 24-h northward neutral wind amplitude (m/s * 10**-1)
40      235    2D array     946 = 24-h max northwrd neutral wind phase (hr * 10**-3)
41      241    2D array     951 = 12-h northward neutrl wind amplitude (m/s * 10**-2)
42      247    2D array     956 = 12-h max northward neutrl wind phase (hr * 10**-3)
43      253    2D array     961 = 8-h northward neutral wind amplitude (m/s * 10**-2)
44      259    2D array     966 = 8-h max northward neutral wind phase (hr * 10**-3)
45      265    2D array     991 = 6-h northward neutral wind amplitude (m/s * 10**-2)
46      271    2D array     996 = 6-h max northward neutral wind phase (hr * 10**-3)
```

data.flt is the data file from which the following are the first few lines:

```
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   100   179    42
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   105   236   113
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   110   320   116
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   115   395    16
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   120   455  -141
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   125   510  -290   1
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   130   550  -394   1
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   135   585  -472   1
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   140   621  -541   1
310 11031  1979   321     0     0  1979   321     0     0 -8750 -7000 19333  -440    80  1950    50  3000     0   145   657  -606   1
```

The entire file contains 514,800 lines; the first 495,000 are the first format type described in **data.fld** (KINDAT=11031) and the remaining 19,800 lines are the second (KINDAT=11032). This example illustrates a few points worth consideration. It is possible to obtain a lot of data in multiple record formats in the same file; records may be nicely ordered or interleaved in a haphazard manner. Even so, it is always possible when reading the file to distinguish these types by virtue of the contents of the fields. In most cases, it is sufficient to use the instrument (KINST) and kind-of-data (KINDAT) codes, however, that is not true in this case. Both the 70° west analysis and the multi-latitude analysis used the same KINDAT codes, so the distinction here is the latitude. Perhaps a more familiar example where selection requires inspection of a parameter involves incoherent scatter radars and the pulse length.

2.8. Batch Mode

When starting cmenu, batch mode is invoked by using key-letter arguments. This precludes browsing files or accessing programs and documents. However, one could start an interactive session, select the interactively accessible files and then complete the session in batch mode. Either way, batch mode is started by using the key-letter **-b**. For instance, the data selection example described in the preceding text could be accomplished by entering:

cmenu -b -t19790318-19790325 -i310 -k11031,11032,11034 -rhuron.scd -d/huron/ws/bozo

In addition to **-b**, there is one key-letter that is intended primarily for batch mode:

-m *mxmbt* where *mxmbt* is the maximum MBytes data to write. If this limit is reached a warning is given and execution terminates.

When **-a** is the sole argument given at the outset, the current key-letter argument settings are displayed without starting; thus, entering

cmenu -a

results in:

```
Current key-letter argument settings are
               Batch processing
    -b off                          (batch option)
               Data Selection
    -t 19790318:0000-19790325:2400  (time interval)
    -i 310                          (instrument codes)
```

```
-k 11031,11032,11034          (kindat codes)
-h 2                          (header/catalog record option)
-m 100                        (max MBytes total)
-n data                       (name, actually data file name prefix)
-s 100                        (size, maximum MBytes per data file)
-f f                          (format of data output)
           Detailed inventory browse
-v 0                          (verbosity of parameter codes)
           Permanent File disposition
-r huron.scd                  (remote host)
-u bozo                       (user)
-p                            (password)
-d /huron/ws/bozo             (directory)
           Interactive
-q off                        (quiet option)
```

Effective use of batch mode requires knowing what is available. This can be accomplished by having inventory information on hand when setting up batch commands. There are a number of versions of the detailed inventory available. Using interactive features to obtain an inventory allows subsetting, however, there are other versions available in directory **/jabba/ftp/archive/cedar**:

catmad.list is an abridged version using alphabetic parameter codes. Geophysical indices datasets are omitted. Time, position and data quality parameters are omitted.

catprt.list is like **catmad.list** except integer parameter codes are used. For general reference, this is probably the best compromise between detail and length.

invsrtf.out is a full listing using integer parameter codes. Long lists of parameter codes are kept on the same line, which is convenient for program input.

All detailed inventories are sorted by instrument code, begin time, and kindat code.

2.9. cmenu and gdat

Some creative users have found it convenient to combine cmenu with gdat, the batch data subsetting program. cmenu delivers binary subsets of logical records to a home computer, then gdat is used on the home computer to read the binary data and produce output containing only the desired fields in a simple character format. gdat provides greater control of data selection: It requires specification of which parameters are output, for which selection may be contingent on another parameter's values; e.g., one can select electron density only when height is between 200km and 300km and the radar pulse length is 320 microsec. Other advantages result from installing gdat locally: The binary files are more compact than the character version, so there is less to transfer, no conversion is required during transfer, and less disk space is used. Execution time is also saved because gdat's binary input is faster than character (formatted) reads. gdat also offers checksum verification to assure data file integrity.

IDL users have made gdat a callable function in order to ingest data for plotting.

A variation which avoids installing gdat locally, is to execute gdat on cedar, then transfer the output data file.

Refer to Chapter 4 for a description of gdat installation and use.

CHAPTER 3

SIMPLEST TASK

There are alternatives to using cmenu (see Chapter 2) to access data. This chapter describes the basic steps needed to identify a data volume, obtain a copy from the NCAR Mass Store system (in case it is not already in **/usr/madrigal/files/stage**) and run an access program - sort of a tape dump - to look at the data file contents. For small datasets, it is often convenient to use the character version of the format. A description of this conversion is given in the last section.

3.1. Identifying a Data Volume

The data archive volumes are inventoried and summaries prepared characterizing the available parameters in time and location. These summary tables are stored in the documents directory, **~bozo/is/docs**, in the annual printed catalogue, and in the public ftp directory (**/jabba/ftp/archive/cedar**). There are advantages to each medium. The files may be browsed with an editor using string searches and the files always reflect the most recent additions, but printed matter may be easier to page through. Thus, it may become convenient to print current versions (after copying to a local computer).

The inventory summary tables descriptions are arranged here from general to specific. Only the last four tables, **cat*** and **akavsns**, list data volumes and, therefore, must ultimately be referenced. The first tables introduce the available instruments and model outputs in increasing time resolution and may be skipped if a specific instrument and time have already been selected.

invpxtyr.out Inventory print by time, year, shows for each instrument in the archive, which years have data available by displaying the number of separate months with data. This is the only table which displays all instruments in the archive and it is compact, currently fitting on one page.

It is important to note that the archive contributions are only a fraction of the data taken by these instruments. So, if these summaries list no data for your selected instrument and time interval, it may be possible to obtain data directly from the instrument operators. See the list of contacts in the annual catalogue.

invpxt.isr.out Inventory print by time, incoherent scatter radars, shows specific dates of available I.S. radar data. A measure of geomagnetic activity is provided by listing the daily Ap index for each date. Data available from multiple radars for the same date(s) are combined on one line of the table for convenient identification of coincident periods. At least one observation is available for each site for each date listed. If available, the type of data are indicated by a combination of **NTV**, corresponding to number density, temperature and velocity. Alternatively, an **X** indicates that data do not fit into one of these categories. An **F** appended indicates availability of coincident Fabry-Perot data.

invpxt.fpi.out Inventory print by time, Fabry-Perot, is similar to the I.S. radar table except the parameter **E** is used to indicate emission intensity (replacing number density).

catprt.list Catalogue print list is the most detailed inventory summary. Ordered by instrument code (see the Catalogue or file **~bozo/is/docs/kinst.file**) and time, each row in this table represents a continuous period of data for a single record type, i.e., logical data records with the same *KINDAT* code and parameter configuration. In addition to times, geographic coverage and specific parameter codes are listed. Each field is defined in Table 3-1.

catprt.full is a variation on catprt.list where all parameter codes are included in the table and they are all kept on one line. This version is convenient for use with scripts or programs.

catmad.list is another variation on catprt.list where alphabetic parameter names are used instead of the integer parameter codes. This is convenient when unfamiliar with the integer codes, yet one still may need to reference the parameter codes file for some definitions.

akavsns (**a**lso **k**nown **a**s **v**olume **s**erial **n**umber) historically provided the translation between the database naming convention and actual physical storage media and volumes. Such aliases have been removed (since the implementation of the NCAR mass store system) resulting in some superfluous fields. Nevertheless, for each database volume used **akavsns** should be consulted for any warnings of idiosyncrasies. Other unique information provided is the delineation of data volume contents by file number (which may be required when using **gmsfile**). The first few lines of the table contain a list of site abbreviations in alphabetical order so that it also outlines the contents. The format of the remainder of the table is described in Table 3-2.

TABLE 3-1. **catprt.list** Fields

DATE TIME	Begin and end time of each data interval, formatted as *yymmdd hhmm*, corresponding to year, month, day, hour and minute UT. Minutes are omitted when there are data gaps of six to 48 hours duration.
KINST	Instrument code: see the catalogue or **~bozo/is/docs/kinst.file**.
KINDAT	(kind of data) analysis code. An explanation may be provided in the header record(s) with the same *kindat* code.
CYCLE TIME	Estimated time (minutes) to complete one observing sequence.
MIN TIME RES	Minimum time resolution (minutes) between two data samples.
MIN VERT RES	Minimum vertical resolution or height difference (kilometers).
HEIGHT	Minimum and maximum height (kilometers).
MAG LAT	Minimum and maximum magnetic latitude (degrees).
VAR TYP	A combination of the letters **NTV**, signifying availability of number densities, temperatures, and velocities.
VSN ALIAS	The volume name used by the database.
PARAMETER CODES	Parameters in this type data of record, excluding error, time, and position codes. Parameter codes are defined in associated header records or in file **~bozo/is/docs/parcods.list**.

catprt.list Example

```
Jicamarca Peru I.S. Radar                                    Min    Min     Height      Mag Lat
    Begin            End                             Cycle   Time   Vert     (km)        (deg)
-----------    -----------                           Time    Res    Res    ---------   -------   Var    VSN Alias
 Date   Time    Date   Time   KINST   KINDAT         (min)   (min)  (km)   Min   Max   Min  Max  Typ    (AKAVSN)     Parameter Codes
------  ----   ------  ----   -----   ------         -----   -----  -----  ---  ----   ---  ---  ---   ----------   -----------------
661111  0922   661112  00        10    30003                   5.0   10.0  145   840     9   20  NT    JRO661111A   512  550  560
```

TABLE 3-2. **akavsns** Fields

Alias	The volume name used by the database (e.g., in the inventory) in the format *iiiyymmddu*, a three letter instrument abbreviation, the first date of data, and a uniqueness character.
# Fls	Number of files.
# Blocks	Number of blocks.
Period of Record	The first and last dates with data.
Inventory File(s)	Position of the associated file(s) in the master inventory.

akavsns Example

*Alias File Name in *Inventory Records	# Fls	# Blocks	−−Period of Record− 1st Date	Last Date	Inventory File(s)
*−−−−−−−−−−−−−−−−−−	−−−	−−−−−−−	−−−−−−−−−	−−−−−−−−−	−−−−−−−−
ARO660714A	1	100	14 Jul 66	19 Jun 70	1

3.2. Copying Mass Store Volumes

Volumes must be copied from the complete archive on the NCAR mass store system to cedar before database programs can access the data. A relatively large amount of space has been reserved to hold copies on cedar in directory **/usr/madrigal/files/stage**. As of this writing there is sufficient space to contain the entire database, so check this directory for the desired data before using the procedures described below. If not already in **stage**, there are two utilities which may be used for copying, **migsget** and **gmsfile**. MIGS (MASnet Internet Gateway) is called upon to do the file staging including spooling and notification via email. (**cedar** users are routinely added to the MIGS database, however, if you receive an error diagnostic indicating this is not the case, send email to **bozo@ucar.edu** or **trouble@ucar.edu**).

migsget is simplest and easiest to use. The single required argument is the database volume name as is listed in **catprt.list** and **akavsns**. For example,

migsget aro740810a

requests MIGS to copy the mass store volume **/ISRADAR/ARO740810A** to **cedar** as **/usr/madrigal/files/stage/aro740810a.cbf.** **migsget** automatically changes the name entered to match the convention of database volumes on the mass store by converting any lower case characters to upper case and prefixing the directory **ISRADAR**. A different convention is employed on **cedar**, where the name is converted to lower case and **.cbf** appended (indicating the data volume is in a COS blocking envelope). Typically, the request is processed in less than five minutes at which time an email status message is issued. If the copy was successful, the data volume is ready for access by database programs.

In order to reasonably share the available space in the staging directory, **migsget** will not copy volumes exceeding 80 MB. Most in the archive are smaller, but there are exceptions which may be circumvented by using **gmsfile** to copy a fraction of the files in a mass store volume. Prerequisite to using **gmsfile** during each login session it is necessary to enter

source /usr/madrigal/prep

This establishes an alias to the actual command; Appendix A goes into more detail and describes how you may automate this by adding it to your **~/.login** file.

gmsfile aei780101a 6 7

requests only the sixth and seventh files instead of the entire volume. The man page has more examples. **akavsns** identifies the contents of each volume by file number; for example, it currently indicates **AEI780101A** contains 10 files each holding one calendar year of 1-minute Ae data, starting with 1978, so the **gmsfile** request is for years 1983 and 1984. An email status message is issued upon completion. The copy is put in directory **/usr/madrigal/files/stage** with a name of the form

gfil*usernnnn***.cbf**

where *usernnnn* is a combination of your login name and a sequence number. If the entire dataset is requested (by omitting the file number arguments), the copy is named using the **migsget** convention.

Archive volumes typically contain a multiple experiments data in separate files. The file selection feature of **gmsfile** is useful when a small fraction of an archive volume is of interest: The benefits are shorter transfer time from the mass store and longer retention time on **cedar**. The staging directory is periodically scrubbed, removing the oldest and largest items first.

When planning disk space usage (perhaps for eventual downloading) there are two methods to obtain Mass Store archive volume size. (File sizes on **cedar** are displayed by the UNIX command **ls -l** or **wc** *file*.)

mssize aro740810a aei780101a

sends a request to MIGS for the size of two data volumes. Responses are emailed. A quicker, worst case estimate may be made using the block count listed in **akavsns**. Although variable size blocks are used, under no circumstance do they exceed 16000 bytes, so an order of magnitude estimate can be made by multiplying by 8000 and a worst case estimate obtained by multiplying by 16000.

3.3. rdtp

Once copied to **cedar**, data are ready for program input. This section describes use of program **rdtp** to print logical records. The following discussion assumes familiarity with the database logical record structure and parameter codes; explanations given here are incidental. Complete definitions are available in the "CEDAR Database Format" or files **~bozo/is/docs/cedar.fmt.ps**, **~bozo/is/docs/parcods.list** and **~bozo/is/docs/kinst.list**).

rdtp reads binary (COS blocked) data and produces a character print file, **rdtp.out**, according to directives in file **rdtp.ctrl**. Logical records are printed by specifying a time interval or record position (number). Other options define when to print header or catalogue records, the fraction of each logical record printed, and additional selection criteria. **rdtp.ctrl** can take two basic forms depending on the logical record selection option:

NAME CKSMOP IHCROP PRTAMT IOKNST IOKNDT
´time´
IBYR IBDT IBHM IBCS IEYR IYDT IEHM IECS

or

NAME CKSMOP IHCROP PRTAMT IOKNST IOKNDT
´position´
NEOFS NPRT1ST NSKPPRT NNTHPRT NTOTPRT NXTRAS EOFOPT
EXTRA(1) EXTRA(2) ... EXTRA(NXTRAS)

where

NAME — is the name of the (COS blocked) CEDAR binary formatted file as obtained from the mass store with **migsget** or **gmsfile**.

CKSMOP — is the checksum option. If **.TRUE.** block checksums are verified on binary input. Programs execute faster when this is **.FALSE.**, which is okay unless suspicious of a corrupted data file.

IHCROP — The header/catalogue record print option may be **0,1,2,**or **3**, corresponding to: never printing, printing if also satisfying the time or position criteria, always printing, or printing only header and catalogue records.

PRTAMT — the print amount controls how much of each record to print. **´prol´** prints only prologues; **´part´** also prints the 1D array and 2D parameter codes; and **´full´** also prints the 2D values or the entire logical record.

IOKNST — Okay KINST (instrument) code. Restricts logical records printed to only those with a KINST value matching *IOKNST*. **-99** means no restriction. If *IOKNST* is a KINST code for any of a number of instruments, each is considered; e.g., *IOKNST* = **30** will select logical records with KINST = **30, 31, 32, or 33**. This behavior differs from cmenu where all KINSTs must be explicitly listed.

IOKNDT — Okay KINDAT (kind of data) code. Restrict logical records printed to only those with a KINDAT value matching *IOKNDT*. **-9999** means no restriction.

IBYR ... IECS — The begin and end time, each specified as four integers: year, month and day, hour and minute, and hundredth seconds. All times are UT. The begin and end times in logical record prologues are compared with this interval and accepted if partially bounded.

NEOFS The number of end-of-files to read before terminating. This may be used to decrease program execution time. If arbitrarily large, e.g., **99999**, execution continues until all data are read.

NPRT1ST Number of logical records to print initially.

NSKPPRT NNTHPRT Number of logical records to iteratively skip then print. Useful for sampling an entire volume without printing everything.

NTOTPRT The total number of logical records to print can be used to avoid printing too much.

NXTRAS The number of extra logical records to print which could not otherwise be printed. This is seldom necessary because combinations of time and position options are usually adequate.

EXTRA(1) EXTRA(2) ... EXTRA(NXTRAS)
NXTRAS integers, each of which indicates position (logical record number) of the extra logical record to be printed.

EOFOPT The EOF print option. If **.TRUE.**, after each end-of-file read the initial *NPRT1ST* logical records are printed. Useful to identify the file stratification in a volume.

An example may be obtained by entering from a directory which you have write permission

gpgm rdtp.ctrl

gpgm searches program directories for the named file and delivers a copy to the current directory (enter **man gpgm** for a detailed description). **rdtp.ctrl**, thus obtained, has a brief description appended to the input fields to jog the memory when editing. Fields are not column dependent and character strings must be apostrophe delimited since **rdtp** uses list directed read statements. Fields must match the data type implicit in the name: The Fortran convention distinguishes type integer from real by names beginning with I, J, K, L, M, or N.

Having satisfactorily modified **rdtp.ctrl**, execution is started by entering

rdtp

which will create, or overwrite, **rdtp.out**.

The following two examples of **rdtp.ctrl** and **rdtp.out**, show the progression of profiling a data volume using the *EOFOPT* and printing prologues then printing all details for a short time interval. In the first example, each prologue shows the record type (2nd value), followed by the KINST code, KINDAT code, and the begin and end time. The last three values are the number of 1D parameters, 2D array parameters and number of rows in the 2D array. 1D parameters are constant for the time interval given in the prologue while 2D parameters have multiple values (often a height profile). The second example prints an explanatory header record which has a similar prologue format with a different record type code. The text portion defines the parameter codes used in the data records. To obtain the physical units, parameter values in the data record are interpreted as the product of the integer and the scale factor; thus, the first row indicates that at 1 Jan 1981 00:00.00 UT the 1-minute Ae is 57 X (1. nT) or 57 nT. (For a complete description, see "CEDAR Database Format or **~bozo/is/docs/cedar.fmt.ps**).

Example 1: **rdtp.ctrl**

```
´/usr/madrigal/files/stage/aei780101a.cbf´
´position´
99999    3 99999  0 99999 0 .TRUE.
```

Example 2: **rdtp.ctrl**

```
'/usr/madrigal/files/stage/aei780101a.cbf'
'time'
1981  101    0    0 1981  101   30    0
```

Example 1: **rdtp.out**

```
1 PROGRAM READTP JOB CONTROL VARIABLES     Thu Oct 15 17:38:19 1992
 Dataset input file name = /usr/madrigal/files/stage/aei780101a.cbf
 Checksums in input blocks will NOT be verified (CKSMOPT)
 Header or catalog records encountered are ALWAYS printed (IHDPOP=2).
 For each logical record printed, only the prologue is displayed
 Logical records will be printed for any KINST (site id) code
 Logical records will be printed for any KINDAT or MODEXP code
 PRINT BY POSITION OPTION SELECTED:
        # of EOFs to read before stopping =   99999
          Initial # logical records printed =       3
 # logical records skipped between prints =   99999
 # logical records printed after skipping =       0
    Total # logical records to be printed =   99999
 Repeat print of initial logical records after each EOF encountered (EOFOPT)
 No extra lrecs will be printed (NXTRAS=0)

 #     1 Prol:   1640   3002    210  30008   1978    101      0      0   1978   1231   2359      0     16      5      8      0
 #     2 Prol:    514   1002    210  30008   1978    101      0      0   1978    101     59      0     16      5      8     60
 #     3 Prol:    514   1002    210  30008   1978    101    100      0   1978    101    159      0     16      5      8     60

 EOF       1 encountered after lrec    8761

 #  8762 Prol:   1640   3002    210  30008   1979    101      0      0   1979   1231   2359      0     16      5      8      0
 #  8763 Prol:    514   1002    210  30008   1979    101      0      0   1979    101     59      0     16      5      8     60
 #  8764 Prol:    514   1002    210  30008   1979    101    100      0   1979    101    159      0     16      5      8     60

 EOF       2 encountered after lrec   17522

 # 17523 Prol:   1640   3002    210  30008   1980    101      0      0   1980   1231   2359      0     16      5      8      0
 # 17524 Prol:    514   1002    210  30008   1980    101      0      0   1980    101     59      0     16      5      8     60
 # 17525 Prol:    514   1002    210  30008   1980    101    100      0   1980    101    159      0     16      5      8     60

    :    :       :     :       :     :      :       :       :      :      :       :      :       :      :      :      :      :

 EOF       8 encountered after lrec   70136

 # 70137 Prol:   1600   3002    210  30008   1986    101      0      0   1986   1231   2359   5999     16      5      8      0
 # 70138 Prol:    514   1002    210  30008   1986    101      0      0   1986    101     59   5999     16      5      8     60
 # 70139 Prol:    514   1002    210  30008   1986    101    100      0   1986    101    159   5999     16      5      8     60
 RDBLK6:  EOD encountered on unit 10 after reading    9 files.
 EOD encountered after lrec   78897 in file   9 tot blks    10605
```

Example 2: **rdtp.out**

```
1 PROGRAM READTP JOB CONTROL VARIABLES     Thu Oct 15 18:19:06 1992
 Dataset input file name = /usr/madrigal/files/stage/aei780101a.cbf
 Checksums in input blocks will NOT be verified (CKSMOPT)
 Header or catalog records encountered are printed if satisfying
          other criteria (IHDPOP=1).
 For each logical record printed, all values are displayed
 Logical records will be printed for any KINST (site id) code
 Logical records will be printed for any KINDAT or MODEXP code
 PRINT BY TIME OPTION SELECTED
       Begin time = 1981  101    0    0
           End time = 1981  101   30    0


 EOF       1 encountered after lrec    8761


 EOF       2 encountered after lrec   17522


 EOF       3 encountered after lrec   26307


 HEADER  record   26308
 PROLOGUE:   1640   3002    210  30008   1981    101      0      0   1981   1231
             2359      0     16      5      8      0      0      0      0      0
                0      0      0      0      0      0      0      0      0      0
                0      0      0      0      0      0      0      0      0      0
 TEXT:
 COLS:1-8 9-16    17-24   25-64                                   65-72   73-80
```

```
CKEYWORD DRWDNO  VALUE   DESCRIPTION                                 UNITS
C       DRWDNO IS DATA RECORD WORD NUMBER
KRECH              3002 HEADER RECORD, VERSION 2
KINST          3     210 GEOPHYSICAL INDICES FROM WDC
KINDAT         4   30008 KINDAT DEFINITION FOLLOWS:
CKINDAT  All data available on the original tape were retained when
CKINDAT  converting these indices into the NCAR I.S. Format; i.e., no
CKINDAT  data were omitted.
CHIST    The original tape from which these data were extracted was
CHIST    obtained from the World Data Center A (Boulder, Colorado USA.
IBYRT              1981 BEGINNING YEAR FOR THIS DATA SET
IBDT                101 BEGINNING MONTH AND DAY
IBHMT                 0 BEGINNING UT HOUR AND MINUTE
IBCST                 0 BEGINNING CENTISECOND
IEYRT              1981 ENDING YEAR FOR THIS DATA SET
IEDTT              1231 ENDING MONTH AND DAY
IEHMT              2359 ENDING UT HOUR AND MINUTE
IECST                 0 ENDING CENTISECOND
LPROL         13      16 DATA RECORD PROLOGUE LENGTH
JPAR          14       5 NUMBER OF SINGLE-VALUED PARAMETERS
MPAR          15       8 NUMBER OF MULTIPLE-VALUED PARAMETERS
NROW          16      60 NUMBER OF ROWS OF MULTIPLE-VALUED PARA METERS
C        NROW MAY VARY FROM THE ABOVE VALUE IN DATA RECORDS
KODS( 1)      17     324 Ae Index (hourly mean)                 1.        nT
KODS( 2)      18     325 Al Index (hourly mean)                 1.        nT
KODS( 3)      19     326 Au Index (hourly mean)                 1.        nT
KODS( 4)      20     327 Ao Index (hourly mean)                 1.        nT
KODS( 5)      21      70 Sampling interval (time between sampls) 1.       s
KODM( 1)      27      10 Year (universal time)                  1.        yr
KODM( 2)      28      20 Month/day (universal time)             1.       mmdd
KODM( 3)      29      30 Hour/min (universal time)              1.       hhmm
KODM( 4)      30      31 Additional increment to hour/min UT    1.E-02    s
KODM( 5)      31     320 Ae Index (1 or 2.5 min sample)         1.        nT
KODM( 6)      32     321 Al Index (1 or 2.5 min sample)         1.        nT
KODM( 7)      33     322 Au Index (1 or 2.5 min sample)         1.        nT
KODM( 8)      34     323 Ao Index (1 or 2.5 min sample)         1.        nT
C       MISSING PARAMETER ARE ASSIGNED A VALUE OF -32767
C       ASSUMED PARAMETERS ERROR VALUES ARE ASSIGNED A VALUE OF  -32766
CANDATE  January 1988.  NCAR

Logical record    26309
Prologue    514   1002    210  30008   1981    101      0      0   1981    101     59      0     16      5      8     60
1D cods:    324    325    326    327     70
1D vals:     43    -14     29      7     60
2D cods:     10     20     30     31    320    321    322    323
2D vals:   1981    101      0      0     57    -12     45     16
           1981    101      1      0     53    -13     40     13
           1981    101      2      0     51    -13     38     12
           1981    101      3      0     47    -11     36     12
           1981    101      4      0     52    -11     41     14
              :      :      :      :      :      :      :      :
           1981    101     59      0     35    -14     20      3


EOF       4 encountered after lrec   35068


EOF       5 encountered after lrec   43829


EOF       6 encountered after lrec   52590


EOF       7 encountered after lrec   61375


EOF       8 encountered after lrec   70136

RDBLK6:  EOD encountered on unit 10 after reading    9 files.
EOD encountered after lrec   78897 in file   9 tot blks    10605
```

rdtp relies on subroutine RDBLK6 to do the unblocking and logical record unpacking as do other database programs. RDBLK6 generates a number of diagnostics when suspiciously formatted blocks are encountered. Perhaps the most commonly issued message,

`RDBLK6: Warning: Block` *xxxx* had *yyy* 16-bit values after the cksum; *zzz* are not 0

can be ignored. It tries to explain that the back end of a block contains something which is currently being interpreted as garbage. This is okay except for the rare (and illegal) instance where logical records span blocks and whatever was skipped may not be insignificant. Other friendly diagnostics generated by RDBLK6 report status at an end of file or the end of data as illustrated above; therein the abbreviation “lrec” means logical record.

When the program terminates early, it usually generates diagnostics. This evidence may be crucial to interpreting what went wrong. If assistance is required, please include the output with a description of what happened and send it to bozo@ucar.edu.

3.4. is2asc and back

For small datasets, it may be most convenient to convert from the (COS blocked) binary format to the character version; it is easily viewed by an editor or with UNIX commands (**cat, more,** or **less**) and there is a version of rdtp for it. The character version of a data file is usually about three times larger than the binary version. Thus, although there is no maximum size restriction, large character files can become unwieldy, too big to conveniently view interactively.

A copy of the example csh script which executes **is2asc** may be obtained by entering:

gpgm is2asc.run

This script is more flexible than rdtp's job control file. Indeed, it contains such a file in the form of a UNIX "here document" and it allows file renaming.

```
#!      /bin/csh -f
#       csh script to execute program "is2asc"
#        (1) Change lfn to the appropriate input "cbf" file full path name
#        (2) Change the job control parameters
#        (3) Maybe change the output file names:  ofl, ofd
#        (4) Run job by typing the name of this script; e.g., "is2asc.run"
#        (5) Review output listing; e.g., "more *.out"
#        (6) Review data file; e.g., "more *.data"

set tfa = ${cwd}/is2asc.in$$
set ofl = ${cwd}/is2asc.out
set ofd = ${cwd}/is2asc.data
set lfn = asc2is.data

cat << EOF >! $tfa
EOF
#  Job control file arguments are read list directed (free formatted)
#       NAME ISTF NEOFS
#  where
#       NAME  = Binary input file name (apostrophe delimited)
#       ISTF  = The number (position) of the first file in NAME to convert
#       NEOFS = The number files in NAME to convert
#  Additional files will be converted and appended to output until
#  all input lines have been read.

is2asc.exe < $tfa >& $ofl

if (-e fort.11) then
  if (! -z fort.11) then
    mv fort.11 $ofd
  endif
endif

rm -f $tfa fort.11
```

The first line clues the UNIX interpreter that this is a csh script. Lines beginning with a **#** are comments. Each **set** defines a csh variable; these are the names of files: the job control input file (**$tfa**), the output diagnostics file (**$ofl**), the output data file (**$ofd**), and the input (COS blocked) binary data file (**$lfn**). **$cwd**, the current working directory, is defined automatically. The contents of **$tfa** are defined between **cat** and **EOF**; in such "here documents", there is variable substitution, so **$lfn** becomes **asc2is.data**. This naming scheme was designed for convenience: there is one place in the script where all file names are specified. Such organization becomes useful when running the job many times; its convenience is also apparent in more complex programs discussed later. In this simple case, one could run is2asc without the script by entering:

is2asc.exe
/usr/madrigal/files/stage/aro860604a.cbf 1 99999

The first line starts the program. It expects to read one text line from the command line (a.k.a. stdin) which contains three fields, so that gobbles up the second line. As it executes, diagnostics are written to the screen (a.k.a. stdout) and the character version data are written to fortran unit number 11. The script redirects stdin (**< $tfa**) and stdout (**> $ofl**). When done executing, the "**if**" block checks to see if the output data file exists and is not zero length, before renaming the output data file. Finally the script cleans up scratch files. The output diagnostics file (not shown) lists inputs and results are summarized: The first full logical record and the next few prologues serve to profile the data volume. It finishes with a summary of the number of logical records converted and number of text lines created in the character version file.

The complete description of the character version of the format is given in "The CEDAR Database Format". An excerpt, describing logical data records (Appendix B), may be sufficient to interpret header and catalogue records too because the prologues of all three record types are almost the same.

The character version of **rdtp** is used like the binary version with one additional (compilation) step:

gpgm -g rdtpchr.f
f77 -O -o rdtpchr rdtpchr.f
gpgm rdtpchr.ctl
edit **rdtp.ctrl**
rdtpchr.

One could adopt this version of the format for any database program which currently reads the binary version by replacing the binary read routine (RDBLK6) with RDCHR from **rdtpchr.f**.

There is also a program to convert back from the character version: **asc2is**. It has the same sort of driver script (**asc2is.run**) which looks just like its counterpart.

CHAPTER 4

GEOPHYSICAL INDICES AND DATA SUBSETTING

This chapter delves deeper into the database, introducing two batch access programs. These tools provide the means to display geophysical indices and dissect any data file. First, plotting is introduced when describing **getndcs**, a program to obtain solar and geomagnetic conditions. Next, the focus switches to a general data access program (**gdat**) which is the the most robust data subsetting program offered, with capability to subset logical records and compute ancillary variables. gdat provides more selection control than cmenu, but its use requires more knowledge of the data record contents.

The files output by these programs may be of sufficient interest to transfer back to the home computer. Commands for this purpose are **ftp** and **rcp**; Appendix A, UNIX help, contains a summary of their use and describes complications associated with the “gatekeeper” security scheme used on NCAR computers.

The last section is only for those who want to use a binary access program on their home computer. It describes how to copy **gdat** source code and install it. This includes installation of the Apex geomagnetic coordinate interpolation code (**apxntrp.f**).

4.1. getndcs

Having selected a time interval for analysis, it is often useful to quantify the solar and geomagnetic conditions. **getndcs** displays tables and plots of most indices available in the database.

The archive contains a number of indices datasets which are periodically updated (see **~bozo/is/docs/akavsns** for the most recent dates):

- One minute and hourly AE, AL, AU, AO starting 1 Jan 1978 (from the NGDC, National Geophysical Data Center).
- Hourly disturbance storm time, Dst, index starting in 1 Jan 1957 (from NGDC).
- Estimated Hemispheric Power Input (as described in Foster et al, *Geophys. Res. Letters*, Vol 13 No 7, Jul 1986) starting 2 Nov 1978.
- Hourly interplanetary magnetic field, solar wind density and speed starting 27 Nov 63 (from NSSDC, the National Space Science Data Center).
- 1-minute average interplanetary magnetic field and solar wind density and speed for most coordinated "world day" campaigns starting 13 Apr 83 (from NSSDC with additional reduction by SRI).
- Assorted indices from the "Lenhart Tape" starting 1 Jan 1960, including 3-hour Kp, 3-hour Ap and daily Ap, F10.7 Flux corrected to 1 AU, and the sunspot number (from NSSDC).
- Estimates of the midnight equatorward boundary of the aurora from DMSP (Defense Meteorological Satellite Program) satellites start 1 Jan 1983.

Updates for these datasets are obtained once or twice a year, so more comprehensive up-to-date versions may be obtained directly from these sources; for a current list of references, refer to the annual Database Catalogue or file **/jabba/ftp/archive/cedar/cedar.urls**. Any publications using these indices should acknowledge these data centers.

cmenu, **rdtp** or **gdat** could be used to access any of the indices datasets, but **getndcs** is more convenient: A single execution can generate tables and plots of all database indices except the 1-minute samples and the midnight equatorward boundary. **getndcs** computes 81 day F10.7 (used by the MSIS model) which is not otherwise available in the data files. **getndcs** also optionally computes cross-tail potential producing additional tables, plots, and a “lexical read” file (for TIGCM and AMIE model inputs). **getndcs** output is not intended to be read by another program; **gdat** or **cmenu** are better suited for that purpose: both produce “flat file” character formats.

getndcs may be executed using the example csh script **getndcs.run**. To obtain a copy and and run it with default criteria, enter:

gpgm getndcs.run
getndcs.run

Before initiating execution, **getndcs.run** names input and output files and defines the job control file. Embedded comments (e.g., lines beginning with “**#**”) indicate possible modifications. Such comments have been removed from the following example:

```
#!     /bin/csh -f

set tfa = "${cwd}/getndcs.in$$"
set ofl = "${cwd}/getndcs.out"
set ofp = "${cwd}/getndcs.cgm"
set ofx = "${cwd}/getndcs.lex"

set fld = "/usr/madrigal/files/stage"
set lf1 = $fld"/"ehp781102a.cbf
set lf2 = $fld"/"gpi600101a.cbf
set lf3 = $fld"/"imf631127a.cbf
set lf4 = $fld"/"aei780101a.cbf
set lf5 = $fld"/"dst570101a.cbf

cat << EOF >! $tfa
  1 1 1 1 1 0
´$lf1' '$lf2' '$lf3' '$lf4' '$lf5'
1979  321    0    0 1979  321 2400    0
EOF

getndcs.exe < $tfa >& $ofl

if (-e fort.61) then
  if (! -z fort.61)then
    mv fort.61 $ofx
  endif
endif

if (-e gmeta) then
  if(! -z gmeta) then
    mv gmeta $ofp
  endif
endif

rm -f $tfa gmeta fort.61
```

The first four executable lines define shell variables naming the job control input file, diagnostic output, plot output, and lexical read output. Sometimes it is convenient to rename the latter three. The next lines similarly define the names of the CEDAR binary (COS blocked) indices files by first defining the directory in which they reside. (Usually these files are resident on cedar; if not, refer to Chapter 3 to obtain a copy from the mass store system.) The job control input file consists of the lines between “cat ...” and “EOF”. “getncs.exe”. starts execution with input and output file naming (fortran units 5 and 6). Next are two conditional blocks which rename output files if they exist and are not zero length. The last line cleans up unneeded intermediate files.

Naming input files using shell variables is a convenience. It is only in the job control file that the names are significant. The job control file has the following form:

IGETHP IGETGPI IGETIMF IGETAE IGETDST ICALCT
NAMEHP NAMGPI NAMIMF NAMAE NAMDST
IBYR IBDT IBHM IBCS IEYR IYDT IEHM IECS LEXBDAY

Fields are not column dependent and character strings must be apostrophe delimited because they are input using list directed read statements. The first line consists of option switches which may be **1** (on) or **0** (off). The second line contains the names of indices volumes. The third line defines a time interval for which indices are to be displayed. Additional time interval lines may be appended.

IGETHP Estimated Hemispheric Power option.

IGETGPI 3-hr Kp, 3-hr ap, 24-hr AP, daily F10.7 cm solar flux, daily sunspot number, and 81-day F10.7 (computed from daily values) option.

IGETIMF hourly interplanetary magnetic field (GSE and GSM coordinates) and solar wind speed and density option.

IGETAE hourly AE option.

IGETDST hourly Dst option.

ICALCT option to calculate the crosstail potential via various formulae.

NAMEHP ... NAMDST are the names of the (COS blocked) CEDAR binary formatted indices files. If the corresponding option is switched off (*IGETxxx*=**0**), the indices file name is irrelevant, and any name may be used; e.g. ' '.

IBYR ... IECS The begin and end time, each specified as four integers: year, month and day, hour and minute, and hundredth seconds. All times are UT.

LEXBDAY The UT day number of the begin day for the lexical output. This field is read only when *ICALCT* is **1**).

The files **getndcs.out** and **getndcs.cgm** were generated using the example input. The input control parameters are printed followed by separate listings for each requested option. Note the use of the database missing data code (**-32767**) in the IMF data.

getndcs.out

```
1 PROGRAM GETNDCS - OPTIONS SELECTED:
    IGETHP IGETGPI IGETIMF  IGETAE IGETDST  ICALCT
       1       1       1       1       1       0
    Hemispheric power data are read from /usr/madrigal/files/stage/ehp781102a.cbf
   GPI (SS#,F107,Kp,Ap,..) are read from /usr/madrigal/files/stage/gpi600101a.cbf
           Hourly IMF data are read from /usr/madrigal/files/stage/imf631127a.cbf
        Hourly AE indices are read from /usr/madrigal/files/stage/aei780101a.cbf
           Hourly DST data are read from /usr/madrigal/files/stage/dst570101a.cbf
 Begin time (year moday hrmin centisec) = 1979  321    0    0
   End time                             = 1979  321 2400    0
 --------- AE INDEX ----------
  Time (mid-point)  Ae(hourly)
  year  mmdd  hrmn      nT
  1979   321    30      65.
  1979   321   130      62.
  1979   321   230      52.
    :     :     :        :
  1979   321  2330      60.


 ------ ESTIMATED HEMISPHERIC POWER INPUT -------
         Requested Time Interval        # Values
 -------Begin------- --------End--------   Found
 1979  321    0    0 1979  321 2400    0      28
  UT Date/Time  Power(GW) PwrIndex  PwrQualifier
 1979  321   41     4.1      3            0
 1979  321  133     3.8      2            1
 1979  321  225     5.3      3            0
 1979  321  317     4.8      3            1
   :    :    :       :       :            :
 1979  321 2339     3.0      2            1


 ------------ GEOPHYSICAL INDICIES ------------
  Time (mid-point)  Daily   81day   Daily  Daily
  year  mmdd  hrmn   SS#    F10.7   F10.7    AP
  1979   321  1200   134.   184.8   182.2     4.
```

```
 Time (mid-point)       3hr    3hr
 year  mmdd  hrmn        Kp     ap
 1979   321   130       0.0     0.
              430       0.3     2.
              730       1.7     6.
             1030       1.0     4.
             1330       1.3     5.
             1630       1.0     4.
             1930       2.3     9.
             2230       1.7     6.


----------------------------------- IMF AND SOLAR WIND -----------------------------------
     Mean     ------- GSM [nT] -------  --- GSE [nT] ---       Solar Wind
---- Time ----      Bx        By        Bz        By        Bz       [#/cm3]  [km/s]    Qualifier
1979  321   30    -3.85     -0.96      3.28      1.02      3.27          8.6    342.9   1313
1979  321  130-32767.00-32767.00-32767.00-32767.00-32767.00    -32767.0 -32767.0      0
1979  321  230-32767.00-32767.00-32767.00-32767.00-32767.00    -32767.0 -32767.0      0
1979  321  330-32767.00-32767.00-32767.00-32767.00-32767.00    -32767.0 -32767.0      0
1979  321  430-32767.00-32767.00-32767.00-32767.00-32767.00    -32767.0 -32767.0      0
1979  321  530-32767.00-32767.00-32767.00-32767.00-32767.00    -32767.0 -32767.0      0
1979  321  630    -1.25      4.51      2.26      5.00      0.72          6.7    360.0   5050
1979  321  730    -1.10      3.74      0.02      3.60     -1.01          7.2    360.0   5050
  :    :    :       :         :         :         :         :            :        :       :
1979  321 2330    -0.89      2.51      3.07      3.81      1.10          9.1    390.0   1350


--------- DST INDEX ---------
 Time (mid-point) DST(hourly)
 year  mmdd  hrmn      nT
 1979   321    30       7.
 1979   321   130       7.
 1979   321   230       6.
   :     :     :        :
 1979   321  2330      11.
```

getndcs.cgm, the plot output adheres to ANSI CGM (Computer Graphics Metacode) and may be displayed on appropriate graphics terminals using the NCAR Graphics utility

ctrans -d*device* **getndcs.cgm**

Frames are advanced by pressing return. A list of supported *device*s is obtained by entering

gcaps

The next chapter (Plot Programs) and Appendix A have more information about displaying NCAR Graphics GCM files.

The following example shows IMF components in three successive frames for a three day interval. Note that the curve is discontinuous when values are missing. A single point isolated by missing data is represented by "+".

4.2. **gdat** use

Program gdat can be used to read the (COS blocked) binary format, select a subset, and produce simple character output.

Of course, the first step toward running gdat is to choose a binary version Database format data file. This may be accomplished using methods described in Chapter 3, or, one might use **cmenu** (see Chapter 2) to produce such a file. When preparing a **gdat** job it is often necessary to determine the data file's record type(s) and associated parameters: The selection behavior depends partially on where desired parameters are located in the data record, the 1D or 2D array; see Appendix B or, preferably, the CEDAR Database Format for definition of logical record components. Program rdtp (Chapter 3) can be used to dump some example logical records or cmenu can produce such parameter lists.

getndcs.cgm IMF frames

gdat reads the chosen dataset sequentially, comparing each data record with selection criteria. Fields in the prologue of the record are tested first: the KINST code, the KINDAT code and the the time interval. If any prologue test fails, then the whole record is tossed; otherwise, any derivable parameters needed are calculated before searching for the desired "primary" parameter code. If it is found, then each "row" in the record is considered further. Spatial parameters are located or calculated and then tested as well as optional additional requested test parameters. 1D array parameters are considered to have the same value for each row in the 2D array. Values are written, one line for each row that was not rejected; included in that line are four time fields and any any additional parameters requested. If only 1D parameters are involved then only one line will be output. If the primary parameter is 2D, then values from those 2D rows passing all tests are output. If the primary parameter is 1D but extra parameters are 2D, then a line is output for each acceptable row, thereby repeating the primary parameter value in each line. If the primary parameter is 1D and tests restrict some 2D rows but no 2D parameters are requested, then the first okay row is output.

The derivable parameters are as follows:

Category	Code	Description	Units
Temperature	550	ion	°K
	560	electron	°K
	570	ratio (electron/ion)	
Electron Density	530	maximum electron density	10**9/m**3
	535	maximum electron density	log10(Ne/m**3)*10**-3
	536	maximum uncorrected electron density	log10(Ne/m**3)*10**-3
	540	height of maximum	km
Geographic position	110	height	km
	111	height, additional increment	m*10**-1
	120	range	km
	121	range additional increment	m*10**-1
	130	azimuth	degrees*10**-2
	140	range	degrees*10**-2
	160	geodetic latitude	degrees*10**-2
	170	geodetic longitude	degrees*10**-2
Magnetic position	220	dip latitude	degrees*10**-2
	222	invariant latitude	degrees*10**-2
	224	centered dipole latitude	degrees*10**-2
	225	PACE latitude	degrees*10**-2
	226	Apex latitude	degrees*10**-2
	246	Apex longitude	degrees*10**-2

The height and range additional increment codes (111 and 121) are an optional means to obtain extra precision than that provided by the basic codes (110 and 120). Geomagnetic latitude is complicated by potential substitution: Whatever magnetic latitude available in the record will be substituted when requesting any of codes 220, 222, 224, or 225; a warning is issued in such cases. There is no substitution for Apex latitude and it is usually available; it can be determined as long as the geodetic location and time are known.

The text file output begins with a user provided header line, followed by lines defining each parameter to be output. The data lines follow, each containing four date/time fields, primary parameter value and any extra parameters.

gdat may be executed using the example csh script **gdat.run**. To obtain a copy and and run it with default criteria, enter:

gpgm gdat.run
gdat.run

gdat.run names input and output files and defines the job control file. Definitions of input fields are given in embedded comments (i.e., lines beginning with "#"); however, they have been omitted from the following example, which requests electric field components, height, and magnetic latitude whenever magnetic latitude is 80° to 85° for a one hour period from a Sondre Stromfjord IS Radar database volume.

```
#!     /bin/csh -f

set tfa = $cwd/gdat.in$$
set ofl = $cwd/gdat.out
set ofd = $cwd/gdat.data

set lfn = /usr/madrigal/files/stage/son961210c.cbf
set pcf = ~bozo/is/docs/parcods.file
set kif = ~bozo/is/docs/kinst.file
set apf = /cedar/d/bozo/apxntrp.grid.1965-2000

cat << EOF >! $tfa
$lfn  0 99999 .FALSE. $pcf $kif $apf
GDAT test data header record - max 80 characters long
1640    1650 110 222 226 246
-99
-9999
1996 1210 2300    0 1996 1210 2400    0
GMLAT 80. 85.

EOF

gdat.exe < $tfa >&! $ofl

if (-e fort.85) then
  if (! -z fort.85) then
    mv fort.85 $ofd
  endif
endif

rm -f $tfa fort.85
```

The first three executable lines define names for the job control input file, diagnostic output, and data output file, using shell variables so that they are convenient to revise. The next lines similarly define the names of the CEDAR binary (COS blocked) data file, the database parameter codes file, the instrument codes file and the Apex magnetic coordinates look-up file. The job control file is next in the form of a here-document; it contains fields:

IBNAM NF2SK NEOFS CKSMOP PCFNAM KIFNAM APXNAM
HDRLBL
IPC,(IXCD(I),I=1,NXCD)
(IOKNST(I),I=1,NOKNST)
(IOKNDT(I),I=1,NOKNDT)
IBYR IBDT IBHM IBCS IEYR IYDT IEHM IECS
(SPAR(I),SMIN(I),SMAX(I),I=1,NSPAR)
(ITCD(I),ITMN(I),ITMX(I),I=1,NTCD)

Field positions are not column dependent; i.e., they must be separated by one or more spaces. The first line contains the input volume names and data input file parameters. The second line is a text label which will be written to output. The remaining lines are the selection criteria: the primary parameter code, any extra parameters, KINST codes, KINDAT codes, the time interval, any spatial parameters and their limits, and any additional test parameters plus their limits.

IBNAM is the name of the (COS blocked) CEDAR binary formatted data volume.

NF2SK is the number of end-of-files to skip before starting selecting data. To start at the beginning use **0**. However, if it is known that first files of the volume are irrelevant, skipping them will save execution time. If execution time becomes an issue, refer to **akavsns** to determine file contents by volume.

NEOFS is the number of end-of-files to read before terminating. To read to the end of data, use **99999**. **gdat** will finish sooner when avoiding unblocking, unpacking and testing irrelevant files within a volume.

CKSMOP is the checksum option. If **.TRUE.** block checksums are verified. Programs execute faster when this is **.FALSE.**, which is usually okay (except when suspicious of a corrupted data file).

PCFNAM is the database parameter codes file name.

KIFNAM is the database instrument codes file name.

APXNAM is the the Apex interpolation lookup tables file. If omitted, no interpolation is performed and Apex coordinates are calculated directly.

HDRLBL is a header label not exceeding 80 characters; this becomes the first record output in the data file.

IPC is the primary parameter code for data selection. This code must be in the selected data record or derivable from the record, including:

- the third temperature parameter of Ti (550), Te (560), or Tr (570) when the other two are available;
- Nmax (530,535,536), Hmax (540) from electron density height profiles;
- location (space) parameters: height (110,111), range (120,121), azimuth angle (130), elevation angle (140), geodetic latitude (160), geodetic longitude (170), geomagnetic latitude (220, 222, 224, or 225), Apex latitude (226), or Apex longitude (246). Additional increment codes, height (111) and range (121), are an optional means to obtain extra precision. With the exception of Apex latitude, the other geomagnetic latitude codes may be filled with whatever magnetic latitude is available in the record; if none are available, then Apex latitude will be used when it can be derived.

XCD(i) are extra parameter codes whose values are to be added to each output data record. These codes must exist in the input data or may be one of the specially computed parameters spatial codes (110, 111, 120, 121, 130, 140, 160, 170, 220, 222, 224, 225, 226, 246) temperature codes (550, 560, and 570) and maximum electron density codes (530, 535, 536, and 540).

IOKNST(i) Only logical records whose KINST value matches *IOKNST(i)* are accepted. A blank line or *IOKNST(1)*=**-99** signifies no KINST restriction.

IOKNDT(i) Only logical records whose KINDAT value matches *IOKNDT(i)* are accepted. A blank line or *IOKNDAT(1)*=**-9999** signifies no KINDAT restriction.

IBYR ... IECS The acceptable begin and end time, each specified as four integers: year, month and day, hour and minute, and hundredth seconds. All times are UT.

SPAR(i) SMIN(i) SMAX(i)
Three fields are entered for each spatial parameter: the name, minimum and maximum value. Recognized names are: **HT** (height), **R** (range) **EL** (elevation angle) **AZ** (azimuth angle) **GDLAT** (geodetic latitude) **GDLON** (geodetic longitude) **GMLAT** (geomagnetic latitude). Lengths are in kilometers; angles are in degrees. Numbers may be integer or floating point.

ITCD(i) ITMN(i) ITMX(i)
Three integer fields are entered for each additional parameter to be tested: the parameter code, its minimum and maximum. Units are those defined for the parameter code; e.g.,

```
520 9000 13000
```

defines electron number density (code 520) values whose units are log10(Ne/m3) * 1.E-03) must be not less than 10**9 nor greater than 10**13 to accept the corresponding *IPC* value. In this example, the blank line signifies no additional parameters are to be tested. *ITCD(i)* must be in the record or it will be rejected.

Appending additional sets of control parameters causes **gdat** to run again, appending anything new to the same output file.

The following tables were generated using the example input. **gdat.out** contains the execution diagnostics including input control parameters. The data file, **gdat.data**, contains the input header record followed by one line declaring how many parameters follow. Next are lines describing each parameter and sort of a header labeling the fields in the subsequent data lines.

gdat.out

```
 PROGRAM GDAT                             Mon Oct  5 11:22:25 1998

 Fortran
   Unit      Contents          File Name
    84   Input data          /huron/ws/bozo/t/son961210c.cbf
                                    No. files to skip     0
                                    No. files to read 99999
                                    Block check-sums will not be verified
    81   Parameter codes     /fs/scd/home0/bozo/is/docs/parcods.file
    82   Instrument codes    /fs/scd/home0/bozo/is/docs/kinst.file
    83   Apex interp grids   /huron/ws/bozo/apxntrp.grid.1965-2000
    85   Data write

   The header line written:
   GDAT test data header record - max 80 characters long

   Primary parameter code:   1640
  Additional codes:   1650    110    222
      Code     ---- Units ----   ------------ Description -----------
      1640     V/m       1.E-05  Direction 4 electric field (perp east)
      1650     V/m       1.E-05  Direction 5 electric field (perp north)
       110     km        1.E+00  Altitude (height)
       222     deg       1.E-02  Invariant latitude in measurement vol
       226     deg       1.E-02  Apex latitude in measurement volume
       246     deg       1.E-02  Apex longitude in measurement volume

   Selection Criteria:
       Any KINST (site id code) is okay
       Any KINDAT (site id code) is okay
       Time interval: 1996 1210 2300    0 to  1996 1210 2400    0
       Acceptable GMLAT interval is   80.0000 to   85.0000
       No additional parameters are tested

GXCDV: Warning requested extra parameter     110 resides in the 2D array
       but the primary parameter   1640 is in the 1D array; returning more
       than one value for this extra parameter may be inappropriate.

GXCDV: Warning requested extra parameter     222 resides in the 2D array
       but the primary parameter   1640 is in the 1D array; returning more
       than one value for this extra parameter may be inappropriate.

Warning: Requested parameter   1640 is in the 1D array (it is single valued)
         yet its value is written more than once when requested extra
         codes are in the 2D array and multiple rows are selected.

 RDBLK6:  EOD encountered on unit 84 after reading    1 files.
EOF    1 encountered after reading     228 lrecs

Wrote       4 data lines from      1 records selected of     228 records read
```

gdat.data

```
GDAT test data header record - max 80 characters long
    10 Parameters:
        10     yr        1.E+00  Year (universal time)
        20     mmdd      1.E+00  Month/day (universal time)
        30     hhmm      1.E+00  Hour/min (universal time)
        31     s         1.E-02  Additional increment to hour/min UT
```

```
    1640     V/m       1.E-05  Direction 4 electric field (perp east)
    1650     V/m       1.E-05  Direction 5 electric field (perp north)
     110     km        1.E+00  Altitude (height)
     222     deg       1.E-02  Invariant latitude in measurement vol
     226     deg       1.E-02  Apex latitude in measurement volume
     246     deg       1.E-02  Apex longitude in measurement volume
  10    20    30    31  1640  1650   110   222   226   246
1996  1210  2331  5450 -2558  4099   399  8011  7986  3485
1996  1210  2331  5450 -2558  4099   427  8048  8024  3419
1996  1210  2331  5450 -2558  4099   461  8092  8069  3338
1996  1210  2331  5450 -2558  4099   496  8135  8112  3248
```

This example illustrates how a single value in the input data can be selected and written multiple times. This occurs when a mixture of 1D and 2D array parameters are requested. In this case, the electric field components are in the 1D array while magnetic field and height are in the 2D array. It turns out that multiple rows in the 2D array pass the selection tests, so one line is written for each okay row, each of which has the same electric field values but different magnetic coordinates and height.

This example also demonstrates that Apex coordinates may be obtained even when the data contain another magnetic latitude.

gdat.data is intended to be easily read with simple format statements. The following fortran reads the introductory information and the first data record from the example. The header text is saved and the parameter codes, labels and scale factors. Except for the time parameters, data values are converted to floating point using the scale factors. Although removing the log scaling is "hard wired" here, subroutine **~bozo/is/src/fixlogs.f** may be used to automate this task.

```
      PARAMETER (NPC=8)
      CHARACTER*80 LBL
      CHARACTER*40 PCDES(NPC)
      CHARACTER*8  PCULBL(NPC)
      DIMENSION IPC(NPC) , SFCTR(NPC)

      OPEN (1,'gdat.data')

      READ (1,'(A)') TEXT
      READ (1,'(I6)') NPCCK
      IF (NPCCK .NE. NPC) STOP

      READ (1,'(I10,5X,A8,F8.0,2X,A40)')
     +       (IPC(I),PCULBL(I),SFCTR(I),PCDES(I),I=1,NPC)
      READ (1,'(I6)') JUNK

      READ (1,'(8I6)') IYR,IMD,IHM,ICS,NE,NEE,IHT,IHTI

      RNE  = 10. ** ( REAL(NE) *SFCTR(5) )
      RNEE = 10. ** ( REAL(NEE)*SFCTR(6) )
      HT = REAL(IHT) * SFCTR(7) + REAL(IHTI) * SFCTR(8)
```

After running gdat, it is possible to discover that no data were written. What went wrong? Usually the selection criteria were too restrictive. Sometimes the primary parameter or additional test parameters are not available. When this occurs, it is time to consult various inventory summaries (cmenu output, **catprt.list** or **invsrtf.out**), but beware that some parameter lists are incomplete; viz., **catprt.list** omits time, space and quality assurance parameters; cmenu output may be similarly limited depending on the setting of the verbosity **-v**. This confusion is eliminated if **rdtp** is used to display a sample of the data records. Another potentially useful tool for determining look directions for steerable antenna is program **pltcyc**, which produces tables and plots of elevation and azimuth angles. For data containing elevation and azimuth information (e.g., radar), it may be beneficial to use program **pltcyc** to display the pointing sequences and, thus, determine the spatial coverage. Refer to Chapter 3 for use of rdtp and Chapter 6 for use of pltcyc.

A long running job may cause concern. Once it is started, there is no further cursor movement until job completion. How long should this take? In a test, gdat took 600 seconds to extract one day of 1 minute Ae, Al, Au indices from the 9 year, 81 MB volume. This is is one of the largest files in the database; in contrast, more than 90%

of the volumes are smaller than 40MB. This is representative of the longest execution times because gdat typically spends many more cycles reading than writing. What can one do about it? On UNIX, one can simply kill the job (enter **control c**, press control and c simultaneously) or it can be put in background: First suspend the current foreground process (enter **control z**), then restart it in background (enter **bg**), or terminate it (enter **kill %**). It could have been started in background in the first place by appending an ampersand (**&**), thus:

gdat.run &

Refer to **man csh** for more information on job manipulation, including use of **jobs**, **ps** and **kill**.

4.3. gdat and **apxntrp** Installation

In Chapter 2, the **cmenu** description completed with a recommendation to use it in conjunction with **gdat**; i.e., use **cmenu** for initial data access and delivery, then switch to **gdat** to further subset the data and produce a simple (easy to read) format. These two utilities complement each other: **cmenu** can not break apart logical records, whereas, **gdat** does just that; it even adds to records with additional parameter calculation. Conversely, the busywork of identifying names and contents of database volumes (required inputs to **gdat**) can be simplified using **cmenu**. Although one can use **gdat** on cedar, it is advantageous to install and use **gdat** on your home computer: **cmenu** can simplify file transfer; binary files contain block checksums which can be crucial when validating intact file transfer; binary files are compact so they are efficient to transfer and store; and they contain other information (header and catalog records, neighboring data) potentially useful in an analysis.

Before proceeding with the installation, one must decide how Apex geomagnetic coordinate calculations are to be performed; whether to calculate the apex of the geomagnetic field line each time or to use interpolation tables. This involves a tradeoff between computing speed and additional set-up requirements. The Apex calculation is an iterative procedure tracing the field line equatorward which is time consuming in comparison to interpolation. However, the interpolation tables must first be prepared and this takes about an hour cpu time for the current default. If this is too much trouble, then simply provide a blank Apex interpolation file name in program inputs.

There are a number of ways to install gdat. Perhaps the most convenient approach for UNIX platforms is to copy all files from both source directories (**~bozo/is/src** and **~bozo/apex**) and use **make** to create executables from each **Makefile**. Equally effective would be to simply copy the 28 gdat source files and compile them. Either way, one must also decide how Apex coordinates are determined: They can be calculated from first principals (**apex.f**), which involves iteratively tracing a field line, or they can be interpolated from a previously calculated grid (**apxntrp.f**). The interpolation scheme is faster but it requires extra effort to create the look-up tables. The installation sequence is as follows: (1) transfer files; (2) make any operating system dependent changes; (3) optionally prepare the Apex interpolation tables file; (4) compile gdat; and (5) transfer a data file and execute an example. All possible files involved are listed in Table 4-1. If the Makefiles are to be used, then one should first prepare the Apex directory executables. All files in the Apex directory should be copied. Before using the **Makefile**, refer to comments in it and in **README** and make any changes appropriate for the operating system; e.g., **IRLF** in **apxntrp.f** must be properly defined. It is probably easiest to create all executables by entering **make all**, thus creating the magnetic field routine object library (**libmag.a**), two test programs (**test.run, tstarr.run**), and the lookup tables creation program (**mkglob.run**). The library is referenced when making the gdat executable. Both test programs should be run. And **mkglob.run** must be executed to create the lookup tables file. Of course, if the operating system matches cedar's then executing mkglob can be avoided by also obtaining a copy of the interpolation tables file.

There are other reasons one might wish to execute **mkglob.run**: The file size, grid resolution or epochs used may not be acceptable. The current default configuration for mkglob might be considered resource intensive, exceeding some PC Fortran compilers memory and taking about an hour to produce a 16MByte look-up tables file which has global grids (2° lat by 2.4° lon) for heights up to 1274km and for each five-year interval from 1965 to 2000. This grid size was chosen to provide adequate precision without being unwieldy. The maximum height may appear to be too low, but the access routine in gdat (**gapx.f**) switches to a direct apex calculation whenever the limit is exceeded. **gapx.f** also automatically switches the interpolation time whenever it changes significantly (120 days). The grid is currently defined by subroutine ggrid.f, but this is easily replaced in program **mkglob.f** with an arbitrary set of grid coordinates; they may have irregular spacing or only cover a portion of the globe.

TABLE 4-1. **GDAT AND APXNTRP FILES**

File	Contents
(GDAT Source in dir: **~bozo/is/src**)	
gdat.f	GDAT main program
rdblk6.f	read and unpack COS blocked binary format logical records
cbfopn.f	underlying read (COS blks)
swap4.f	byte reversal (needed for DEC computers and PCs)
cbfskf.f	COS blocked file skip
gbytes.fortran	unpack and fields; includes gbyte, gbytes, sbyte, sbytes
cvti2r.f	convert time: four integers to double precision
cvtr2i.f	convert time: double precision to four integers
cvt2dn.f	convert time: year, month, day to day number
cvt2md.f	convert time: day number to year, month, day
hours.f	used by some time routines
gspac.f	get spatial location parameters from a data record
gtime.f	get time parameters from a data record
makt.f	add a temperature parameter (Te, Ti or Tr) to a record
maknm.f	add Nmax and Hmax to a data record
quadmx.f	calculate a maximum for a quadratic (used by maknm)
gxcdv.f	get extra code values from a record
cspac.f	check space parameters against selection criteria
newtim.f	calculate a new time from an old (4 integer) and a difference
timdif.f	determine the difference between two (4 integer) times
findpc.f	find parameter code location in a data record
rdpcf.f	read parameter codes file into memory; includes gpcod, gpcods
rdkif.f	read instruments codes into memory; includes gknst, gknsts
parsb.f	find the location of the first and last non-blanks in a string
parsbs.f	find the locations of all blank delimited fields in a string
gapx.f	get Apex coordinates using either interpolation or field line tracing
(APEX Source in dir: **~bozo/apex**)	
apxntrp.f	Apex interpolation main program
apex.f	Apex calculation
magfld.f	geomagnetic field
(APEX Interpolation Source in dir: **~bozo/apex**)	
Makefile	compiler directives for use by UNIX **make**
README	directions for installation and use
apex.f	Apex calculation via field line tracing
apxntrp.f	Apex calculation via interpolation from a previously calculated grid
cossza.f	cosine solar zenith angle calculation
ggrid.f	subroutine to generate look-up tables
magfld.f	DGRF/IGRF (Definitive or Interim Geomagnetic Ref. Field) definition
magloctm.f	magnetic local time calculation
mkglob.f	main program to make look-up tables
subsol.f	sub-solar point calculation
test.f	example program 1 (general)
test.out	example program 1 output
tstarr.f	example program 2 (look-up tables)
tstarr.out	example program 2 output
(CODE DEFINITIONS TABLES in dir: **~bozo/is/docs**)	
parcods.file	parameters
kinst.file	instruments
(GDAT DRIVER SCRIPT in dir: **~bozo/is/run**)	
gdat.run	Csh script which executes gdat, defining job control and file names
(APEX INTERPOLATION TABLES FILE)	
/d/bozo/apxntrp.grid.1965-2000	lookup tables file (binary apropos only the current OS)

If unable to use the Apex **Makefile**, then compilation must be explicitly keyed in; for instance, to create the executable **mkglob.run** on a UNIX system, enter:

f77 -O -o mkglob.run mkglob.f ggrid.f apxntrp.f magfld.f apex.f

If the interpolation files are not to be created, then only the three files listed for Apex source in Table 4-1 are needed; they should be added to the list of source files when compiling gdat. Don't forget to verify proper definition of **IRLF** in **apxntrp.f**. Also make sure the interpolation file name (defined in the script **gdat.run**) is blank when executing gdat.

Some preparations may be necessary before creating the gdat executable using its Makefile. While only those files listed for gdat source in Table 4-1 are needed from that directory, it is easiest to copy the entire contents and then remove the extraneous files; this includes all old intermediate object files (***.o**) and old versions of source code (***.old**). Next, edit gdat's **Makefile**. Therein are directions for accommodating different operating systems. It is also likely that the definition of **DIRBIN** (the directory to contain executables) and **DIRMAG** (the geomagnetic field directory) need to be changed to match the directories to be used. **DIRBIN** will need to be created if it does not yet exist.

Depending on the operating system, some changes may be required to the source files comprising **gdat**. Embedded comments identify operating system specifics. A detailed description of the binary read routine installation is also given in the **Cedar Database Format**. In **gdat.f**, locate and maybe redefine **LMWD** and **IRLF**. In **cbfopn.f**, locate **Crecl** and **Cbyte**) and make the changes prescribed for your operating system. In **gbytes.fortran** at the beginning of subroutine **GBYTES** and **SBYTES**, statement functions definitions may need alteration for **MASKS**, **OR**, **AND** and **NOT.**

After all of that, the gdat executable may be created by entering **make gdat**. If not able to use the Makefiles, one can still create the executables manually by including all of the source files as arguments to **f77**.

It is convenient to use the Csh script **~bozo/is/run/gdat.run** to execute **gdat** after it has been compiled. This defines the job control file, redirects it to standard input (Fortran unit 5), redirects standard output (Fortran unit 6) to a file named **gdat.out**, and renames the data output file (**fort.30** from fortran unit 30) to **gdat.data**. If UNIX is not available, **gdat.run** could be replaced by defining these file linkages in OPEN statements added to the source.

CHAPTER 5

PLOT PROGRAMS

This chapter describes access utilities which render the data as time series plots (**pltvst**), individual height profiles (**pltvsh**), or binned height and time averages (**binhnt**). Any parameter in the input data volume may be plotted as well as various extensions for implicitly available parameters. Data selection options impose bounds in time and space; e.g., height must be between 100 and 200 km. Additional restrictions may be imposed on data record type as defined by KINST and KINDAT.

pltcyc is described prior to the more general parameter plotting programs. Plot-cycle only selects azimuth and elevation angle, which is useful for identifying geographical coverage and antenna motion.

If not familiar with the previous two chapters, one may wish to review two topics relevent to these plot programs: The details of Csh job execution script (see **is2asc and back**) and debugging selection criteria in **gdat use**.

A copy of job execution scripts may be obtained and executed by entering

gpgm *name***.run**
*name***.run**

where *name* is **pltcyc**, **pltvst**, **pltvsh**, or **binhnt**. These scripts name input and output files, define the job control file, execute the program, and clean up intermediate files upon completion. An example of these scripts is given in the **getndcs** section where there is also a description of how to use **ctrans** to display plot metacode files. If using a windows environment, **idt** is a more sophisticated alternative requiring additional initialization steps:

- Add to your file **~/.cshrc** or enter once each login session:

  setenv GRAPHCAP X11
  setenv DISPLAY *wkstname***:0**

  where *wkstname* is the name of your workstation; e.g., mine is **ypsilon.scd.ucar.edu**. (Also requisite is definition of the environment variable **NCARG_ROOT** as **/opt/local**; this is established for each login in the system Cshrc file and may be confirmed by entering **env**).

- Optionally predefine where **idt** opens windows on your screen. To do this, add to or make a file in your home directory named **.Xdefaults**; e.g., to stack the three windows vertically on the right side of a screen with dimension 1152x900 (width x height in pixels):

  idt.geometry: 410x90+600+0
  idt*TopLevelShell*geometry: 600x600+540+115
  ctrans*geometry: 600x600

  (These settings are apropos NCAR Graphics Version 3.2; settings for earlier versions differ to accommodate different window configurations.) Locations are of the form *W***x***H*[+|-]*X*[+|-]*Y*, where, *W* is width, *H* is height, positive signs indicate window offset from the left (*X*) or top (*Y*) and negatives from the right or bottom.

- It is also possible to tailor the print menu button in **idt** to your printer environment; enter **man ncarv_spool** for details.

- Each windows session, give permission for **idt** running on **cedar** to open a window on your workstation by entering (or adding to your file **~/.xinitrc**)

  xhost +cedar.hao.ucar.edu

5.1. pltcyc

Radar data represent a significant fraction of the archive, most of which are from steerable antennae. When using these data, it is often important to know the scan sequence, for example, when determining vector velocities from line-of-sight. **pltcyc** was designed to aid cycle time determination by displaying the antenna(e) pointing sequence and times. Although the text output file contains all relevant information, the cycle pattern is most evident in the plot output, so it is best to use both.

pltcyc reads binary (COS blocked) files and extracts elevation, azimuth angle and time from logical data records. If begin and end values are present in a logical record, they are averaged. The values are then plotted in polar coordinates as radial lines where the length is proportional to zenith angle (90°-elevation angle) and the orientation is the azimuth angle (North or 360° is up, East or 90° is to the right). A sequence number is written beyond the end of each line which may be cross referenced in the text output. If a pointing direction is repeated, the sequence number is written radially outward enough to avoid overwriting the previous number. This continues until the edge of the plot is reached. In order to avoid too much clutter on the plot, sampling time intervals should generally not exceed an hour.

The job control file in **pltcyc.run** takes one of two forms depending on whether data are to be selected for display by time or logical record position:

1 *NAME*
IBYR IBDT IBHM IBCS IEYR IYDT IEHM IECS IOKNST IOKNDT

or

2 *NAME*
NF2SK IST NPTS IOKNST IOKNDT

Fields are not column dependent; character strings must be apostrophe delimited because **pltcyc** uses list directed read statements.

NAME is the name of the CEDAR binary (COS blocked) format file.

IBYR ... IECS The begin and end time, each specified as four integers: year, month and day, hour and minute, and hundredth seconds. All times are UT.

IOKNST The okay KINST (instrument) code restricts logical data records to only those with a KINST value matching *IOKNST*. **-99** means no restriction. If *IOKNST* is a KINST code for any of a number of instruments, each is considered; e.g., *IOKNST* = **30** will select logical records with KINST = **30, 31, 32, or 33**. This behavior is different than cmenu where each KINST must be explicitly listed.

IOKNDT The okay KINDAT (kind of data) code restricts logical records examined for data to only those with a KINDAT value matching *IOKNDT*. **-9999** means no restriction.

NF2SK is the number of end-of-files to skip before selecting data. To start at the beginning use **0**. Effective use of this option requires reference to **akavsns** to determine data volume contents by file (usually each file contains a separate experiment period).

IST is the position (number) of the first logical data record to examine.

NPTS is the number of azimuth, elevation angle pairs to display before terminating.

Repeating all control parameters appends additional tables and plot frames to the output files as in the following example.

```
2 '/usr/madrigal/files/stage/mlh851014a.cbf'
2 1 100 -99 -9999
1 '/usr/madrigal/files/stage/mlh851014a.cbf'
1985 1112  1300 0 1985 1112 1503 0   31  -9999
```

The same data are read first using the position option then using the time option. The second part of the output would be identical except for the restriction of only the Millstone Hill steerable antenna (KINST = **31**).

pltcyc.out

```
      Program Plot Cycle
   Position option selected; az and el angle  are extracted from data
   records by position (or number):
                    Skip forward :       2 files
             then start at record:       1
     Total number of points to get:     100
   Additional data record selection criteria:
 Logical records will be used regardless of KINST (site id) code
 Logical records will be used regardless of the KINDAT (data processing algorithm) code
 Terminating after finding     100 points

1  /usr/madrigal/files/stage/mlh851014a.cbf
      DEL T                              AZIMUTH        ELEVATION
   #  (min)        TIME          AVG   BEG   END   AVG  BEG  END  KINST KINDAT
 ---- ----- ------------------- ----- ----- ----- ---- ---- ---- ----- ------
    1   0.0 1985 1112 1356  900  180.  180.  180. 88.0 88.0 88.0    32   3001
    2   1.4 1985 1112 1357 3200  180. -180. -180. 88.0 88.0 88.0    31   3001
    3   2.7 1985 1112 1358 5300  180.  180.  180. 88.0 88.0 88.0    32   3001
    :    :    :    :    :     :    :     :     :    :    :    :      :     :
  100  66.8 1985 1112 1503    0  351.   -6.  -11.  6.0  6.0  6.0    31   3001
1       Program Plot Cycle
   Time option selected; az and el angle data are extracted from data records
   for the time interval:
             Begin time:  1985 1112 1300    0
               End time:  1985 1112 1503    0
   Additional data record selection criteria:
 Logical records will be used only if
       the KINST (site id) code is      31
 Logical records will be used regardless of the KINDAT (data processing algorithm) code
 RDBLK6:  EOD encountered on unit  1 after reading   12 files.
 End of data encountered after finding      93 points of up to    1000 requested.

1  /usr/madrigal/files/stage/mlh851014a.cbf
      DEL T                              AZIMUTH        ELEVATION
   #  (min)        TIME          AVG   BEG   END   AVG  BEG  END  KINST KINDAT
 ---- ----- ------------------- ----- ----- ----- ---- ---- ---- ----- ------
    1   0.0 1985 1112 1357 3200  180. -180. -180. 88.0 88.0 88.0    31   3001
    2   2.7 1985 1112 1400 1500  180. -180. -180. 88.0 88.0 88.0    31   3001
    3   5.4 1985 1112 1402 5850  180. -180. -180. 88.0 88.0 88.0    31   3001
    :    :    :    :    :     :    :     :     :    :    :    :      :     :
   93  65.5 1985 1112 1503    0  351.   -6.  -11.  6.0  6.0  6.0    31   3001
```

5.2. pltvst

Plot versus time may be used to plot any parameter in the input data file as a function of time. An implicit temperature parameter may also be plotted, providing that two of the three (electron temperature, ion temperature, and temperature ratio) are available (see **gdat** for details). Given a desired time interval and parameter code, **pltvst** identifies "rows" in logical data records (see **CEDAR Database Format** or **~bozo/is/docs/cedar.fmt.ps**) containing the parameter and passing the time criterion. The "row" selection may be further constrained by spatial criteria. Logical records may also be restricted by KINST and KINDAT values. For each accepted "row", the datum is plotted and optionally written to a character file. Each plot frame consists of an X-axis (time) and at least one Y-axis. Multiple Y-axes are stacked vertically in the plot frame.

The job control file in **pltvst.run** has the following form:

```
IBNAM NEOFS CKSMOP PCFNAM WRTOP
FRMLBL
IBYR IBDT IBHM IBCS IEYR IYDT IEHM IECS          NYAXS
YMIN(1)    YMAX(1) NMAJRY(1) NMINRY(1) ZROOPT(1)
IPARCD(1) IOKNST(1) IOKNDT(1) NSPAR(1)
    SPAR(1,1)          SMIN(1,1)          SMAX(1,1)
    SPAR(2,1)          SMIN(2,1)          SMAX(2,1)
         .                  .                  .
    SPAR(NSPAR(1),1) SMIN(NSPAR(1),1) SMAX(NSPAR(1),1)
YMIN(2)    YMAX(2) NMAJRY(2) NMINRY(2) ZROOPT(2)
IPARCD(2) IOKNST(2) IOKNDT(2) NSPAR(2)
```

pltcyc.cgm

```
     SPAR(1,2)          SMIN(1,2)          SMAX(1,2)
     SPAR(2,2)          SMIN(2,2)          SMAX(2,2)
          .                  .                  .
     SPAR(NSPAR(2),2)   SMIN(NSPAR(2),2)   SMAX(NSPAR(2),2)
  .
  .
  .
YMIN(NYAXS)   YMAX(NYAXS)   NMAJRY(NYAXS)   NMINRY(NYAXS)   ZROOPT(NYAXS)
IPARCD(NYAXS)   IOKNST(NYAXS)   IOKNDT(NYAXS)   NSPAR(NYAXS)
     SPAR(1,NYAXS)               SMIN(1,NYAXS)               SMAX(1,NYAXS)
     SPAR(2,NYAXS)               SMIN(2,NYAXS)               SMAX(2,NYAXS)
          .                           .                           .
     SPAR(NSPAR(NYAXS),NYAXS)   SMIN(NSPAR(NYAXS),NYAXS)   SMAX(NSPAR(NYAXS),NYAXS)
```

For each Y-axis in a frame, job control lines beginning with *YMIN* must be specified. Additional Y-axes are positioned above predecessors. A plot frame is produced for each set of job control lines starting with *FRMLBL*. Fields are not column dependent and character strings must be apostrophe delimited because **pltvst** uses list directed read statements.

IBNAM is the name of the (COS blocked) CEDAR binary formatted data volume.

NEOFS is the number of end-of-files to read before terminating. To read to the end of data, use **99999**. **pltvst** will finish sooner when avoiding unblocking, unpacking and testing irrelevant files within a volume.

CKSMOP is the checksum option. If **.TRUE.** block checksums are verified on binary input. Programs execute faster when this is **.FALSE.**, which is okay unless suspicious of a corrupted data file.

PCFNAM is the name of the parameter codes information file.

WRTOP is the write option. If **.TRUE.** plot points are written to a text file.

FRMLBL is the text string to be used as a label centered in the top of the plot frame.

IBYR ... IECS The acceptable begin and end time, each specified as four integers: year, month and day, hour and minute, and hundredth seconds. All times are UT.

NYAXS is the number of Y-axes to include in this plot frame. The current (dimension) limit is four axes.

YMIN YMAX NMAJRY NMINRY
are the minimum and maximum Y-axis values, the number of labeled tick-mark intervals and the number of unlabeled tick-mark intervals between each labeled tick-mark. These are chosen automatically if *NMAJRY* < 0.

ZROOPT The zero line option. If **.TRUE.**, the horizontal line Y=0 is drawn.

IPARCD is the code of the parameter to be plotted.

IOKNST The okay KINST (instrument) code restricts logical data records to only those with a KINST value matching *IOKNST*. **-99** means no restriction. If *IOKNST* is a KINST code for any of a number of instruments, each is considered; e.g., *IOKNST* = **30** will select logical records with KINST = **30, 31, 32, or 33**. This behavior is different than cmenu where each KINST must be explicitly listed.

IOKNDT The okay KINDAT (kind of data) code restricts logical records examined for data to only those with a KINDAT value matching *IOKNDT*. **-9999** means no restriction.

NSPAR is the number of space parameters to be used for *IPARCD* selection.

SPAR(i,j) SMIN(i,j) SMAX(i,j)
Spatial selection parameter name, minimum and maximum value. Recognized *SPAR* are: **´HT´ ´R´ ´EL´ ´AZ´ ´GDLAT´ ´GDLON´ ´GMLAT´** for height, range, elevation angle, azimuth angle, geodetic latitude, geodetic longitude, and magnetic latitude, respectively. Lengths are in km, angles are in degrees.

The following example reads all files in the data volume. The time intervals 04:30-05:30 and 08:30-09:30 UT 17 February 1982 are plotted in separate frames. Both frames have two Y-axes of electron density (code 505) values at two height intervals. Data in the second frame are taken from logical records with KINST=20 and KINDAT=30014, as is indicated in the Y-axis label. **pltvst.out** displays the job control parameters and other execution diagnostics including the number of points in each plot.

Evident in **pltvst.data** and **pltvst.cgm** is the repetition of data at each time: Six values passed the height selection criterion from each height profile in the first (lowest) Y-axis while only three passed the second (upper) Y-axis. Note also that the plot symbol changes with Y-axis so outliers may be identified.

pltvst.run

```
#!      /bin/csh -f

set tfa = ${cwd}/pltvst.in$$
set ofl = ${cwd}/pltvst.out
set ofp = ${cwd}/pltvst.cgm
set ofd = ${cwd}/pltvst.data

set lfn = /usr/madrigal/files/stage/aro810114b.cbf
set pcf = /home/cedar/bozo/is/docs/parcods.file
```

```
cat << EOF >! $tfa
´$lfn' 9999 .FALSE. '$pcf' .TRUE.
´(1) Heights 300-305km (upper) 250-260km (lower) Y-axis'
1982  217  430 0 1982  217  530 0         2
   8.    14.  4 5 .FALSE.
   505 -99 -9999   1
      'HT   ' 250. 260.
   8.    14.  4 5 .FALSE.
   505 -99 -9999   1
      'HT   ' 300. 305.
´(2) Same as frame 1, but automatic Y-labels'
1982  217  830 0 1982  217  930 0         2
   8.    14. -1 5 .FALSE.
   505  20 30014   1
      'HT   ' 250. 260.
   8.    14. -1 5 .FALSE.
   505  20 30014   1
      'HT   ' 300. 305.
EOF

pltvst.exe < $tfa >& $ofl

if (-e fort.30) then
  if (! -z fort.30) then
    mv fort.30 $ofd
  endif
endif

if (-e gmeta) then
  if (! -z gmeta) then
    mv gmeta $ofp
  endif
endif

rm -f $tfa gmeta fort.30
```

---

pltvst.out

```
 PROGRAM PLOT-VERSUS-TIME                              Thu Nov 12 11:35:45 1992
  Plot data will be read from unit  11 reading /usr/madrigal/files/stage/aro810114b.cbf
   Number of files to read =  9999
   Check-sums will not be verified.
   Plot data will be written to unit 30
   Parameter codes will be read  from unit  90 opened as file /home/cedar/bozo/is/docs/parcods.file
 -------------------------------------------------------------------------

    FRAME  1  Header label: (1) Hts 300-305km (upper) 250-260 km (lower) Y-axis
             Time interval: 1982  217  430    0 to
                            1982  217  530    0
          Number of Y-axes:    2

 ---------- Y-axis  1 plot value minimum:  8.0000E+00
                      plot value maximum:  1.4000E+01
            # major (labelled) divisions:    4
          # minor (unlabelled) divisions:    5
                  Parameter code plotted:   505, lg10(rawNe) [lg(m-3) ]
                  To be plotted on frame:    1 as Y-axis:  1
       Any KINST (site id code) will be accepted.
       Any data processing algorithm code (KINDAT) will be accepted
                  Space selection option:    1
     ( 1 space parameters are used for plot values selection)
                                         HT      from  250.0000 to   260.0000
      .
      .
      .

  After reading     6 files,
```

```
Yaxis Frame  # Plot Points
   1     1         192
   2     1          96
   1     2         192
   2     2          96
```

pltvst.data

```
  1    1     192  1.E-03  lg10(rawNe)lg(m-3)  (frm,axis,npts,sclfctr,par)
11676 1982  217  450  100
11735 1982  217  450  100
11833 1982  217  450  100
11844 1982  217  450  100
11885 1982  217  450  100
11945 1982  217  450  100
11595 1982  217  451 1500
11671 1982  217  451 1500
11743 1982  217  451 1500
11801 1982  217  451 1500
11843 1982  217  451 1500
11889 1982  217  451 1500
  :     :    :    :   :
11768 1982  217  529 2900
  1    2      96  1.E-03  lg10(rawNe)lg(m-3)  (frm,axis,npts,sclfctr,par)
12236 1982  217  450  100
12268 1982  217  450  100
12252 1982  217  450  100
12226 1982  217  451 1500
12226 1982  217  451 1500
12220 1982  217  451 1500
  :     :    :    :   :
11825 1982  217  529 2900
11781 1982  217  529 2900
11787 1982  217  529 2900
  2    1     192  1.E-03  lg10(rawNe)lg(m-3)     20 30014  (frm,axis,npts,sclfctr,par)
11676 1982  217  450  100
  :     :    :    :   :
```

pltvst.data excerpts reveal the beginning of three of the four Y-axes data. Each Y-axis is represented by an initial record followed by one record for each plot point. Read fortran might be:

```
    READ (IUNIT,'(2I5,I8,2X,1PE6.0,2X,A37)') IFR,IAX,NPTS,SFCTR,PCDES
    DO 100 I=1,NPTS
    READ (IUNIT,'(I7,4I5)') IVAL,IYR(I),IMD(I),IHM(I),ICS(I)
100 RVAL(I) = REAL (IVAL) * SFCTR
```

READ (IUNIT,'(2I5,I8,2X,1PE6.0,2X,A37)') IFR,IAX,NPTS,SFCTR,PCDES

where

IFR is the frame number, beginning with 1.

IAX is the Y-axis number, where 1 is the lowest (first requested) axis of the frame.

NPTS is the number of data points on the Y-axis.

SFCTR is the parameter scale factor, which is used to convert integers (**IVAL**) to real values in the units described by **PCDES**.

5.3. pltvsh

Plot versus height may be used to produce height profiles of any parameter in the input data. An implicit temperature parameter may also be plotted, providing that two of the three (electron temperature, ion temperature, and temperature ratio) are available. Another extension is the ability to add Chiu, IRI (Version 9), or MSIS (1986) model output to a plot. (More information on models is in Section 6.3).

pltvsh data selection logic is similar to **pltvst**. Given a desired time interval and parameter code, **pltvsh** identifies "rows" in logical data records (see **CEDAR Database Format** or **~bozo/is/docs/cedar.fmt.ps**) containing the

pltvst.cgm

---

parameter and passing the time criterion. The “row” selection may be further constrained by spatial criteria. Logical records may also be restricted by KINST and KINDAT values. For each accepted “row”, the datum is plotted and optionally written to a character file. Each plot frame consists of a Y-axis (height) and at least one X-axis. Additional X-axes are added horizontally to the right of the initial one.

Geophysical indices needed as inputs for the models are obtained from the database volume **GPI600101A**. The mean of the selection time interval is used to obtain sunspot number, F10.7, and Ap. Models are not run when indices are not available (see **~bozo/is/docs/akavsns** for current holdings). Geographic latitude and longitude required by models must be specified in the job control file. Calculations are performed for heights from 100 to 1000km, unless the data height interval is smaller.

The job control file in **pltvsh.run** has the following form:

```
IBNAM NEOFS CKSMOP WRTOP PCFNAM IRINAM GPINAM
FRMLBL
IBYR IBDT IBHM IBCS IEYR IYDT IEHM IECS     YMIN YMAX     NXAX
XMIN(1) XMAX(1) NMAJRX(1) NMINRX(1) ZROOP(1) CHIUOP(1) IRIOP(1) MSISOP(1) GGLAT(1) GGLON(1)
IPARCD(1) IOKNST(1) IOKNDT(1) NSPAR(1)
   SPAR(1,1)        SMIN(1,1)        SMAX(1,1)
```

```
    SPAR(2,1)          SMIN(2,1)          SMAX(2,1)
        .                  .                  .
    SPAR(NSPAR(1),1) SMIN(NSPAR(1),1) SMAX(NSPAR(1),1)
XMIN(2) XMAX(2) NMAJRX(2) NMINRX(2) ZROOP(2) CHIUOP(2) IRIOP(2) MSISOP(2) CGLAT(2) CGLON(2)
IPARCD(2) IOKNST(2) IOKNDT(2) NSPAR(2)
    SPAR(1,2)          SMIN(1,2)          SMAX(1,2)
    SPAR(2,2)          SMIN(2,2)          SMAX(2,2)
        .                  .                  .
    SPAR(NSPAR(2),2) SMIN(NSPAR(2),2) SMAX(NSPAR(2),2)
  .
  .
  .
XMIN(NXAX) XMAX(NXAX) NMAJRX(NXAX) NMINRX(NXAX) ZROOP(NXAX) CHIUOP(NXAX) IRIOP(NXAX) MSISOP(NXAX) CGLAT(NXAX) CGLON(NXAX)
IPARCD(NXAX) IOKNST(NXAX) IOKNDT(NXAX) NSPAR(NXAX)
    SPAR(1,NXAX)          SMIN(1,NXAX)          SMAX(1,NXAX)
    SPAR(2,NXAX)          SMIN(2,NXAX)          SMAX(2,NXAX)
        .                     .                     .
    SPAR(NSPAR(2),NXAX) SMIN(NSPAR(2),NXAX) SMAX(NSPAR(2),NXAX)
```

For each X-axis in a frame, job control lines beginning with *XMIN* must be specified. Additional X-axes are positioned to the right of predecessors. A plot frame is produced for each set of job control lines starting with *FRMLBL*. Fields are not column dependent and character strings must be apostrophe delimited because **pltvsh** uses list directed read statements.

IBNAM is the name of the (COS blocked) CEDAR binary formatted data volume.

NEOFS The number of end-of-files to read before terminating. This may be used to decrease program execution time. If arbitrarily large, e.g., **99999**, execution continues until all data are read.

CKSMOP is the checksum option. If **.TRUE.** block checksums are verified on binary input. Programs execute faster when this is **.FALSE.**, which is okay unless suspicious of a corrupted data file.

WRTOP is the write option. If **.TRUE.** plotted data points are written to a text file.

PCFNAM IRINAM GPINAM
: is the name of the parameter codes information file, the IRI model coefficients file, and the geophysical indices file. If IRI or MSIS model options are not used, any non-blank name will suffice for *IRINAM GPINAM*.

FRMLBL is the text string to be used as a label centered in the top of the plot frame.

IBYR ... IECS The acceptable begin and end time, each specified as four integers: year, month and day, hour and minute, and hundredth seconds. All times are UT.

YMIN YMAX are the minimum and maximum heights (km) to show on this frame. If either is negative, it is automatically selected.

NXAX is the number of X-axes to include in this plot frame. The current (dimension) limit is three axes.

XMIN XMAX NMAJRX NMINRX
: are the minimum and maximum X-axis values, the number of labeled tick-mark intervals and the number of unlabeled tick-mark intervals between each labeled tick-mark. These are chosen automatically if *NMAJRX* < 0.

ZROOPT The zero line option. If **.TRUE.**, the vertical line X=0 is drawn.

CHIUOP IRIOP MSISOP
: are the option switches for the Chiu, IRI, and MSIS model parameter options. If **.TRUE.**, then the model parameter (*IPARCD*) is added to the plot of this X-axis. Chiu or IRI model electron density profiles may be added when *IPARCD* is 510 or 520. Other IRI model parameters include ion temperature (550), electron temperature (560), neutral temperature (810), O^+/Ne (620), NO^+/Ne (630), O_2^+/Ne (640), He^+/Ne (650). MSIS model parameters include neutral temperature (810), N_2 (850), O_2 (860), O (870), AR (880), He (890), H (900).

GGLAT GGLON
: are the geographic latitude and longitude (degrees, positive towards north and east) to be used for model calculations. These fields must be filled even if not using a model option.

IPARCD is the code of the parameter to be plotted.

IOKNST The okay KINST (instrument) code restricts logical data records to only those with a KINST value matching *IOKNST*. **-99** means no restriction. If *IOKNST* is a KINST code for any of a number of instruments, each is considered; e.g., *IOKNST* = **30** will select logical records with KINST = **30, 31, 32, or 33**. This behavior is different than cmenu where each KINST must be explicitly listed.

IOKNDT The okay KINDAT (kind of data) code restricts logical records examined for data to only those with a KINDAT value matching *IOKNDT*. **-9999** means no restriction.

NSPAR is the number of space parameters to be used for *IPARCD* selection.

SPAR(i,j) SMIN(i,j) SMAX(i,j)
: Spatial selection parameter name, minimum and maximum value. Recognized *SPAR* are: **´HT´ ´R´ ´EL´ ´AZ´ ´GDLAT´ ´GDLON´ ´GMLAT´** for height, range, elevation angle, azimuth angle, geodetic latitude, geodetic longitude, and magnetic latitude, respectively. Lengths are in km, angles are in degrees.

In the following example, the time interval was chosen to select three height profiles comprising a single scan at Arecibo. **pltvsh** only reads the number of fields specified in the input list, so comments such as the frame and axis notes may be appended without affecting the program. X-axis limits are specified in the first frame and automatically selected in the second. For both frames the Y-axis minimum is specified to be zero but the maximum is automatically selected. Points are plotted using symbols (e.g., "+") which change with X-axis so outliers may be identified. Model parameters are drawn as curves, labelled "C" (Chiu), "I" (IRI), or "M" (MSIS).

pltvsh.run

```
#!      /bin/csh -f

set tfa = "${cwd}/pltvsh.in$$"
set ofl = "${cwd}/pltvsh.out"
set ofp = "${cwd}/pltvsh.cgm"
set ofd = "${cwd}/pltvsh.data"

set lf1 = "/usr/madrigal/files/stage/aro740810a.cbf"
set lf2 = "/usr/madrigal/files/stage/gpi600101a.cbf"
set pcf = "/home/cedar/bozo/is/docs/parcods.file"
set icf = "/usr/madrigal/files/stage/iricofcnts"

cat << EOF >! $tfa
´$lf1' 9999 .TRUE. .TRUE. '$pcf' '$icf' '$lf2'
´(1) No spatial restriction, choose X-ticks'
1974 918 1725 0 1974 918 1835 0            0. -1.       2
10. 13.  3 5 .F. .T. .T. .F. 18.345 293.247   (left X-axis frm 1)
  520 -99 -9999 0
0. 1000. 5 2 .F. .F. .T. .T. 18.345 293.247   (rght X-axis frm 1)
  810 -99 -9999 0
´(2) Add Model Te and Ti, auto X-ticks'
1974 918 1725 0 1974 918 1835 0            0. -1.       2
0. 2500. 5 5 .F. .F. .T. .F. 18.345 293.247   (left X-axis frm 2)
  560 -99 -9999 0
0. 2500. 5 5 .F. .F. .T. .F. 18.345 293.247   (rght X-axis frm 2)
  550 -99 -9999 0
EOF

pltvsh.exe < $tfa >& $ofl

if (-e fort.21) then
  if (! -z fort.21) then
    mv fort.21 $ofd
  endif
```

```
endif

if (-e gmeta) then
  if (! -z gmeta) then
    mv gmeta $ofp
  endif
endif

rm -f $tfa gmeta fort.21
```

pltvsh.out

```
         PROGRAM PLOT-VERSUS-HEIGHT                   Mon Nov 16 16:41:14 1992
    I.S. Formatted data will be read from unit  11
 I.S. Formatted file name = /usr/madrigal/files/stage/aro740810a.cbf
        Number of files to read 9999
 Checksums will be verified during input
 Plot points are written to unit   21
 Other named input files:
       Parameter codes = /home/cedar/bozo/is/docs/parcods.file
      IRI coefficients = /usr/madrigal/files/stage/iricofcnts
    Geophysical indices = /usr/madrigal/files/stage/gpi600101a.cbf
    (coefficients and indices files are read if a model a option is selected)

 --------------------------------------------------------------------

    FRAME  1  Header label: (1) No spatial restriction, choose X-ticks
             Time interval: 1974  918 1725    0 to
                            1974  918 1835    0
 Y-axis (height) label min:    0.00
 Y-axis (height) label max:  will be selected automatically
          Number of X-axes:    2

 ---------- X-axis  1 plot value minimum:  1.0000E+01
                      plot value maximum:  1.3000E+01
            # major (labelled) divisions:    3
          # minor (unlabelled) divisions:    5
        Chiu model Ne values are plotted on this X-axis as a dashed curve
        labelled with C
        IRI model values will be plotted on this X-axis as a solid curve
        labelled with I
        The model will use latitude   18.345
                      and longitude  293.247
                 Parameter code and name:   520,  lg10(Ne)   [lg(m-3) ]
        Any site id code (KINST) will be accepted.
        Any data processing algorithm code (KINDAT) will be accepted
                  Space selection option:    0
     ( 0 space parameters are used for plot values selection)
      :
      :
  After reading     1 files,
  X-axis Frame  # Plot Points
      1     1         360
      1     2           0
      2     1          78
      2     2          78
```

pltvsh.data

```
   1    1     360  1.E-03  lg10(Ne)  lg(m-3)  (frm,axis,npts,sclfctr,par)
  10260    83.70 1974  918 1727 5600
  10247    84.90 1974  918 1727 5600
    :       :     :     :    :    :
  12000   476.60 1974  918 1816  900
   1    2       0  1.E+00  Tn        K  (frm,axis,npts,sclfctr,par)
```

```
 2    1      78  1.E+00  Te        K  (frm,axis,npts,sclfctr,par)
 165   101.00 1974   918 1742  400
 177   104.00 1974   918 1742  400
  :       :     :     :    :    :
1206   493.00 1974   918 1826 3750
 2    2      78  1.E+00  Ti        K  (frm,axis,npts,sclfctr,par)
 165   101.00 1974   918 1742  400
 177   104.00 1974   918 1742  400
  :       :     :     :    :    :
1102   435.00 1974   918 1826 3750
```

pltvsh.data format parallels **pltvst.data**. Each Y-axis is represented by an initial record followed by one record for each plot point. However, model values are excluded, so there are no data records for the right Y-axis of the first frame. Read fortran might be:

```
    READ (IUNIT,'(2I5,I8,2X,1PE6.0,2X,A37)') IFR,IAX,NPTS,SFCTR,PCDES
    DO 100 I=1,NPTS
    READ (IUNIT,'(I7,F9.2,4I5)') IVAL,HT(I),IYR(I),IMD(I),IHM(I),ICS(I)
100 RVAL(I) = REAL (IVAL) * SFCTR
```

IFR is the frame number, beginning with 1.

IAX is the Y-axis number, where 1 is the left (first requested) axis of the frame.

NPTS is the number of data points on the Y-axis.

SFCTR is the parameter scale factor, which is used to convert integers (**IVAL**) to real values in the units described by **PCDES**.

5.4. binhnt

Program bin height and time differs from **pltvsh** by averaging selected data in height and time. **binhnt** may be used to plot any parameter in the input data. An implicit temperature parameter may also be plotted, providing that two of the three (electron temperature, ion temperature, and temperature ratio) are available.

binhnt data selection logic is similar to the other programs. Given a desired time interval and parameter code, **binhnt** identifies "rows" in logical data records (see **CEDAR Database Format** or **~bozo/is/docs/cedar.fmt.ps**) containing the parameter and passing the time criterion. The "row" selection may be further constrained by spatial criteria. Logical records may also be restricted by KINST and KINDAT values. For each accepted "row", the datum is accumulated in the bin matching the height and time. When done selecting data, bins are averaged either as simple arithmetic means or weighted by error values. In order to compute weighted means, the error of the parameter (**-***IPARCD*) is also accumulated when selecting data. After averaging, another option can fill holes in the plot caused by empty bins by linear interpolation over one or two adjacent empty time bins at the same height. Bin averages may be rendered as monochrome contours, color contours, or color cell array (where each bin is individually colored). Another option allows log scaling. Bin averages and other statistics are optionally written to a character file.

The job control file in **binhnt.run** has the following form:

IBNAM NEOFS CKSMOP PCFNAM WRTOP
FRMLBL
IBYR IBDT IBHM IBCS IEYR IYDT IEHM IECS
IPARCD IOKNST IOKNDT NSPAR
  SPAR(1) SMIN(1) SMAX(1)
  SPAR(2) SMIN(2) SMAX(2)
  . . .
  SPAR(NSPAR) SMIN(NSPAR) SMAX(NSPAR)
SIZHBN SIZTBN IWTOPT TNTPL8
IPLTOP LOGPLT ICTROP

Depending on the selected contour option (*ICTROP* > 0), additional job control parameters may be required in either of two forms:

pltvsh.cgm

FLO FHI FINC

or

NLIN CTRLIN(1) CTRLIN(2) . . . CTRLIN(NLIN)

Fields are not column dependent and character strings must be apostrophe delimited because **binhnt** uses list directed read statements.

IBNAM is the name of the (COS blocked) CEDAR binary formatted data volume.

NEOFS The number of end-of-files to read before terminating. This may be used to decrease program execution time. If arbitrarily large, e.g., **99999**, execution continues until all data are read.

CKSMOP is the checksum option. If **.TRUE.** block checksums are verified on binary input. Programs execute faster when this is **.FALSE.**, which is okay unless suspicious of a corrupted data file.

PCFNAM is the name of the parameter codes information file.

WRTOP is the write option. If **.TRUE.**, the binned data are written to a text file.

FRMLBL is the text string to be used as a label centered in the top of the plot frame.

IBYR . . . *IECS*
The acceptable begin and end time, each specified as four integers: year, month and day, hour and minute, and hundredth seconds. All times are UT.

IPARCD is the code of the parameter to be plotted.

IOKNST The okay KINST (instrument) code restricts logical data records to only those with a KINST value matching *IOKNST*. **-99** means no restriction. If *IOKNST* is a KINST code for any of a number of instruments, each is considered; e.g., *IOKNST* = **30** will select logical records with KINST = **30, 31, 32, or 33**. This behavior is different than cmenu where each KINST must be explicitly listed.

IOKNDT The okay KINDAT (kind of data) code restricts logical records examined for data to only those with a KINDAT value matching *IOKNDT*. **-9999** means no restriction.

NSPAR is the number of space parameters to be used for *IPARCD* selection.

SPAR(i , j) *SMIN(i , j)* *SMAX(i , j)*
Spatial selection parameter name, minimum and maximum value. Recognized *SPAR* are: **´HT´ ´R´ ´EL´ ´AZ´ ´GDLAT´ ´GDLON´ ´GMLAT´** for height, range, elevation angle, azimuth angle, geodetic latitude, geodetic longitude, and magnetic latitude, respectively. Lengths are in km, angles are in degrees. If not explicitly set, height limits default to 0 to 1000 km.

SIZHBN SIZTBN
are the height (km) and time (min) bin sizes.

IWTOPT the weighting option may be **1** or **2**: **1** indicates bins are averaged as simple arithmetic means; **2** indicates bins are averaged using 1/error^2 weighting.

TNTPL8 is the time interpolation option. If **.TRUE.**, one or two empty bins adjacent in time are filled by linear interpolation of bounding bins at the same height.

IPLTOP the plot option flag, may be one of four values: **1** for monochrome contours, **2** for monochrome contours with shade fill, **3** for color filled contours, or **4** for cell array.

LOGPLT is the log plot option. If **.TRUE.**, the plotted values are log10 scaled. This option has no effect on binned data text output (when *WRTOP* is **.TRUE.**).

ICTROP the contour option may be one of three values: **0** indicates contour levels are automatically selected; **1** indicates contours are linear and specified by *FLO FHI* and *FINC*; **2** contours are arbitrarily specified by *NLIN CTRLIN(1) CTRLIN(2)* . . . *CTRLIN(NLIN)*.

FLO FHI FINC
are the maximum the lowest interval, the minimum of the highest interval, and the size of a contour interval; e.g., for contour options, the values of the lowest and highest contour lines. Color plot options (*IPLTOP* > **2**) are restricted to no more than 14 intervals. These fields are only included when *ICTROP* is **1**.

NLIN CTRLIN(1) CTRLIN(2) . . . *CTRLIN(NLIN)*
are the number of contour lines and their values. Color plot options (*IPLTOP* > **2**) are restricted to no more than 13 lines. These fields are only included when *ICTROP* is **2**.

A plot frame is produced for each set of job control lines starting with *FRMLBL*.

binhnt will extend the last time or highest height bin if it is significantly smaller than the others. Bin maxima are computed from the selection criterion minimum and the bin size. The selection criterion maximum is adjusted, if it does not match the computed maximum. A warning message is printed for a significant adjustment (more than 1% of the bin size).

A diagnostic is also generated if the product of the number of height and time bins is too large (currently the limit is 15000). In this case there are three corrective actions possible: increase bin sizes, reduce selection limits (maybe make two plot frames for the time interval), or increase the array dimension (see Chapter 6: Program Modification).

In the following example, two days of high resolution uncorrected electron density data are binned and plotted in two frames to demonstrate display options. *S1ZTBN* was chosen to match the antenna cycle time, so each time bin should contain one azimuth scan and represent average conditions overhead. *S1ZHBN* was chosen as a compromise between showing fine scale height features and smoothing (**catprt.list** reveals the height resolution is 1.7km).

binhnt.out displays all input job control parameters and contains summary statistics about the binning, including the maximum bin average and the height of that maximum for each time bin.

binhnt.run

```
#!      /bin/csh -f
set tfa = "${cwd}/binhnt.in$$"
set ofl = "${cwd}/binhnt.out"
set ofp = "${cwd}/binhnt.cgm"
set ofd = "${cwd}/binhnt.data"

set lfn = "/usr/madrigal/files/stage/aro860114a.cbf"
set pcf = "/home/cedar/bozo/is/docs/parcods.file"

cat << EOF >! $tfa
´$lfn’ 99999 .FALSE. ’$pcf’ .TRUE.
´(1) IPLTOP=1 , ICTROP=0’
986  403    0    0 1986  404 2359 5999
   505 20  30016  1
´HT’  0. 599.9999
10.0 16.0  2 .FALSE.
1 .TRUE. 0
´(2) IPLTOP=1 , ICTROP=1 , TNTPL8=T’
986  403    0    0 1986  404 2359 5999
   505 20  30016  1
´HT’  0. 599.9999
10.0 16.0  2 .TRUE.
1 .TRUE. 1
            9.25E00   12.25E00   .25E00
EOF
if (-e fort.30) then
  if (! -z fort.30) then
    mv fort.30 $ofd
  endif
endif

if (-e gmeta) then
  if (! -z gmeta) then
    mv gmeta $ofp
  endif
endif

rm -f $tfa gmeta fort.30
```

binhnt.out

```
1         PROGRAM BIN HEIGHT AND TIME                              Fri Nov 20 16:17:57 1992
  Plot data will be read from unit  11 reading /usr/madrigal/files/stage/aro860114a.cbf
   Number of files to read = 99999
   Check-sums will not be verified.
   Binned data results will be written to unit 30
   Parameter codes will be read  from unit  90 opened as file /home/cedar/bozo/is/docs/parcods.file

 ----------------------------------------------------

 PLOT HEADER LABEL: (1) IPLTOP=1 , ICTROP=0

 PARAMETER SELECTION CRITEREA:
   Acceptable time range: 1986- 403-   0:   0 TO 1986- 404-2359:5999
```

```
  Parameter code searched:   505, lg10(rawNe) (units: lg(m-3)  )
    Log10 scaling will be removed prior to binning.  New labels are:
     long: Electron density (no temp correction)     Abbreviated:   Raw Ne
    units:  1/m**3
  KINST (site id code) must be      20
  KINDAT (data processing algorithm code) must be    30016
  Space selection option:    1
    ( 1 space parameters are used for plot values selection)
    HT    from    0.0000 to   599.9999

 BINNED ARRAY DEFINITION:
  2D array height bin width:    10.0000 (km)
  2D array time   bin width:    16.0000 (min)
  (Note that the acceptance criteria define the beginning value of the first
   bins and the number of bins)
      Number of height bins:    60
      Number of time   bins:   180
  Bin averages will be calculated using weights = 1/error**2.
  Empty bins will not be filled by linear interpolation in time.

 PLOT OPTIONS:
  The binned array will be plotted using contours (IPLTOP=1).
  The plot will be log10 scaled.
  Contour (or cell array) levels are automatically selected based on the
  plot min/max, a linear increment, and chosing 10 to 30 levels (ICTROP=0).

 RDBLK6:  EOD encountered on unit 11 after reading    1 files.

 BINNED PLOT DATA ARRAY
  Number of values input to bins =       34548
                  Total # bins =       10800
          Number of empty bins =        4831
     Number of bins with 1 value =           0
             Minimum bin value =  8.1008E+00
                   @ bin mean time   =  1986  404  408    0
                              height =  65.0
             Maximum bin value =  1.2594E+01
                   @ bin mean time   =  1986  404  352    0
                              height = 325.0
  Maximum values for each column (bin time):
         Time          Hmax(km)    Max
  1986  403    8    0   315.00  1.1554E+01
  1986  403   24    0   315.00  1.1554E+01
  1986  403   40    0   315.00  1.1550E+01
  1986  403   56    0   305.00  1.1533E+01
  1986  403  112    0
  1986  403  128    0
  1986  403  144    0   325.00  1.1474E+01
  1986  403  200    0
    .    .    .     .     .      .
```

binhnt.data

```
(1) IPLTOP=1 , ICTROP=0
Electron density (no temp correction) [1/m**3]
    0.0000  600.0000                          (min/max hts - km)
 1986  403    0    0 1986  404 2359 5999      (min/max times - UT)
        60       180                          (number ht,time bins)
     5.000                                    (middle first ht bin)
    15.000
      :
      :
   595.000
 1986  403    8    0                          (middle first time bin)
 1986  403   24    0
   :    :     :    :
   :    :     :    :
 1986  404 2352    0
  -3.2767E+04  -3.2767E+04  -3.2767E+04  -3.2767E+04       0
  -3.2767E+04  -3.2767E+04  -3.2767E+04  -3.2767E+04       0
  -3.2767E+04  -3.2767E+04  -3.2767E+04  -3.2767E+04       0
  -3.2767E+04  -3.2767E+04  -3.2767E+04  -3.2767E+04       0
  -3.2767E+04  -3.2767E+04  -3.2767E+04  -3.2767E+04       0
```

```
3.4159E+08   6.4782E+07   3.6752E+07   5.4143E+07       2
3.4676E+08   4.8324E+07   4.0968E+07   1.0630E+08       6
4.3976E+08   6.1436E+07   9.1039E+07   1.7540E+08       5
6.7755E+08   5.8694E+07   1.9087E+08   3.7208E+08       6
    ⋮            ⋮            ⋮            ⋮            ⋮
```

The binned array is written such that all height bins for the first time are written first, starting with the lowest height. Fortran to read **binhnt.data** might be

```
      READ (IUNIT,'(A80)' FRMLBL
      READ (IUNIT,'(A100)' LBLZ
      READ (IUNIT,'(2F10.4)' HTB,HTT
      READ (IUNIT,'(8I5)' (ITIM(I),I=1,8)
      READ (IUNIT,'(2I10)' NUMHBN,NUMTBN
      READ (IUNIT,'(F10.3') (HT(I),I=1,NUMHBN)
      READ (IUNIT,'(4I5)') ((IT(I,J),J=1,4),I=1,NUMTBN)
      READ (IUNIT,'(4E13.4,I8)') ((YBAR(I,J),SIGMA(I,J),S(I,J),V(I,J),
     +                          N(I,J),I=1,NUMHBN),J=1,NUMTBN)
```

where

FRMLBL is *FRMLBL*.

LBLZ is a complete description of *IPARCD*.

HTB HTT are the minimum height of the lowest bin and maximum of the highest.

NUMHBN NUMTBN are the number of height and time bins.

IH is an array containing the mean for each height bin, lowest first.

IT is an array containing the mean for each time bin, earliest first.

YBAR is the bin average in units described by LBLZ. The first dimension is height, lowest first, and the second dimension is time, earliest first.

$$\bar{y} = \frac{\sum_i w_i y_i}{\sum_i w_i}$$

where y_i is the $i\,th$ datum from parameter *IPARCD* and w_i is the $i\,th$ weight. If *IWTOPT* = **1**, $w_i = 1$. If *IWTOPT* = **2**, $w_i = 1/\sigma_i^2$, where σ_i is the $i\,th$ datum from parameter **-***IPARCD*.

SIGMA is the error of YBAR obtained from input error values.

$$\bar{\sigma} = \left[\frac{\sum_i w_i^2 \sigma_i^2}{(\sum_i w_i)^2} \right]^{\frac{1}{2}}$$

S is the error of YBAR obtained from the standard deviation.

$$S = \left[\frac{\sum_i w_i^2 (y_i - \bar{y})^2}{(\sum_i w_i)^2 - \sum_i w_i^2} \right]^{\frac{1}{2}}$$

V is the variability of YBAR for constant weights obtained from standard deviation.

$$V = \left[\frac{\sum_{i=1}^{N} (y_i - \bar{y})^2}{N - 1} \right]^{\frac{1}{2}}$$

For large N and $w_i = 1$, $V = N^{\frac{1}{2}} S$.

N is the number of samples in each bin in YBAR, SIGMA, S, and V.

binhnt.cgm

CHAPTER 6

PROGRAMMING

This chapter introduces programming with the database software. This introduction summarizes what is available and where it is located. The next section describes the binaries library and how it may be used in programs; examples are given for a time routine (**cvti2r.f**), reading parameter codes (**rdpcf.f**), reading instrument codes (**rdkif.f**), and two binary accesss routines (**rdblk6.f and vbkopn.f**). The following sections describe program modification; first with a simple example, then with detailed examples for programs **binhnt** and **gdat**. The last section summarizes available models.

The data access programs reside in subdirectories below **~bozo/is/**. **is/README** introduces the contents of these directories. Subdirectory **is/run** contains batch job scripts (***.run**) and rdtp job control input file (**rdtp.ctrl**). Ancillary files are in subdirectory **is/docs**, including **parcods.file** and **kinst.file**, parameter and instrument codes definitions.

Subdirectory **is/src** contains source code files. Fortran source code files are distinguished by the "**.f**" suffix; C source routines with "**.c**". Also in **src**, **Makefile** contains rules for compiling and linking the source files; this is used by the UNIX command **make**. Source file names mimic the name of the program or first subroutine in the file; e.g., **rdtp.f** contains the source for the main program **rdtp**. Source files often contain multiple routines. Subroutines unique to the program are included with the main routine source but shared routines are kept separately, thus, avoiding duplication.

Executables in **~bozo/is/bin/SunOS** reflect this organization, with subroutines used in multiple programs stored in a linkable object library. Most common subroutines are in **~bozo/is/bin/SunOS/libis.a**, but another directory contains geomagnetic field subroutines; its library is **~bozo/apex/bin/SunOS/libmag.a**. Entire programs are identifiable by the suffix **.exe**; one exception is **rdtp**.

The dump utility, **rdtp**, is configured more simply than the other routines. It is invoked by name and execution is controlled by a small "job control" file. Other programs involve a Csh script to accomplish this, which provides the flexibility to rename files as well as defining the job control file. Most programs, including **rdtp**, are composed of more than one source file. This practise avoids duplication. Comments near the beginning of each main program source list the additional files required. For instance, **rdtp** is made up of files:

File Name	Routines Contained
rdtp.f	the main routine and subroutine JOBCTL
rdblk6.f	RDBLK6
cbfopn.f	CBFOPN, CBFCLS, CBFRD, CBFWR, CBFEOF
gbyts.c or gbytes.fortran	GBYTE, GBYTES, SBYTE, SBYTES
prtr.f	PRTR
hdrprt.f	HDRPRT
lrprt.f	LRPRT
cktim.f	CKTIM
timdif.f	TIMDIF
hours.f	HOURS

Table 6-1 introduces most of the files in the primary source directory. These files may be useful when modifying database programs or to add function to other code.

Explanatory remarks are embedded in all source files and **Makefile**. These comments are the first and, in some cases, only form of documentation. Sufficient information is provided in each routine to use it: The algorithm

TABLE 6-1. **Source files in dir: ~bozo/is/src**

File Name	Description
a162ch.f	Character string unpacker for header and catalog records
binhnt.f	Main program for **binhnt**
cbfopn.f	COS block access subroutine with read and write entries
cbfskf.f	COS blocked skip file subroutine.
ch2a16.f	Reverse of a162ch
chuiri.f	The Chiu and IRI models
cktim.f	Determine whether a time (4 integers) is in an interval
cnvrnh.f	Old coordinate conversion routine; gspac.f is now preferred
cspac.f	Check space parameters against selection criteria (used by gdat)
cvt2dn.f	Convert time (four integers) to day number of the year
cvt2md.f	Convert year and day number to month and day
cvti2r.f	Convert time (four integers) to real (actually double precision) representation
cvtr2i.f	Reverse of cvti2r.f
efield.f	Richmond's electric field model
findpc.f	Given a logical data record, search for a specified parameter code
fixlgs.f	Remove log10 scaling from parameter code labels
gbyts.c	Bit manipulation routines for unpacking and packing fields
gchiu.f	Driver for the Chiu model
gdat.f	Main program for **gdat**
getae.f	AE indices access subroutine (reads **AEI780101A**)
getdst.f	Dst index access subroutine (reads **DST570101A**)
getehp.f	Estimated hemispheric power access subroutine (reads **EHP781102A**)
getgpd.f	Access one days worth of Kp, ap, Ap, F10.7, and sunspot number (reads **GPI600101A**)
getgpi.f	Like getgpid, but returns one value per call; also computes 81 day F10.7
gethpd.f	Similar to getehp.f but returns a days worth of values
getimf.f	Hourly IMF and solar wind data access subroutine (reads **IMF631127A**)
getndcs.f	Main program for **getndcs**
ghttic.f	Get height axis tick marks and labels for plotting
giri.f	Driver for the IRI model
gmsis.f	Driver for the MSIS-83 or MSIS-86 model
gspac.f	Get spatial parameters for a given logical data record
gtime.f	Get times for a given logical data record
gtmtic.f	Get time tick marks and axis labels for plotting
gxcdv.f	T{Get extra parameter code values; used by **gdat** to extract additional data from a logical record
hdrprt.f	Print logical header and catalogue records
igrf.f	Obtain DGRF/IGRF geomagnetic field for geographic location
is2asc.f	Main program for format conversion program: binary to character
lcprt.f	Print character version of header/catalog records
lrprt.f	Print logical data records
makhdr.f	Make a header record based on the contents of a data record
maklr.f	Produce test logical data records
maknm.f	Make electron density Nmax and Hmax and add them to a logical data record
makt.f	Make the third temperature from the other two (Te, Ti, or Tr)
msis.f	The 1983 and 1986 MSIS model
newtim.f	Compute a new time from an old time and an increment
paklr.f	Pack a logical record (in preparation for writing with vbkopn.f)
parsb.f	Parse a character string for leading and trailing blanks
parsbs.f	Parse a character string for all blank delimited text
pltcyc.f	Main program for **pltcyc**
pltvsh.f	Main program for **pltvsh**
pltvst.f	Main program for **pltvst**
pltyvt.f	Plot y versus time
prtr.f	Print a binary version logical record; drives lrprt.f and hdrprt.f
prtrc.f	Print a character version logical record; drives lrprt.f and prtrc.f
quadmx.f	Determine the maximum using a parabolic fit
rdasc.f	Read the CEDAR format character version (superseded by rdchr.f)
rdblk6.f	Read the binary version of the data format
rdchr.f	Read the CEDAR format character version
rdkif.f	Read the instrument (KINST) codes file
rdpcf.f	Read the parameter codes file
rdtp.f	Main program for **rdtp**
selht.f	Select parameter values from a logical record (superseded by gspac.f and cspac.f)
swap4.f	Byte reversal subroutine for DEC computers and PCs
timdif.f	Compute the time difference
ut2lt.f	Convert UT to local standard time; currently set up only for I.S. Radars
vbkopn.f	CEDAR format binary version access routine with entries to read and write
wrchr.f	CEDAR format character version write routine

is summarized, all input and returned arguments are defined, and any alternate entries are similarly described.

Parallel to **src** and below **is** is the main binaries directory **~bozo/is/bin/SunOS** or **~bozo/is/bin/Lunux**. The files suffixed by "**.exe**" are the executable programs. The general subroutine library is **libis.a**. Routines related to geomagnetic coordinates are in another library (**~bozo/apex/bin/{SunOS,Linux}/libmag.a**)

6.1. SUBROUTINE LIBRARY

The general library (**~bozo/is/bin/{SunOS,Linux}/libis.a**) contains a number of subroutines which may be used directly by other programs. Most geomagnetic related routines are stored separately in **~bozo/is/bin/{SunOS,Linux}/libmag.a**; the previous section describes its use with gdat. Empirical models are also stored in separate directories; see the next section. Table 6-2 groups these by function; refer to Table 6-1 for individual descriptions.

TABLE 6-2. **General Library Subroutines**

Function	Source File Name
Codes	rdkif.f rdpcf.f fixlgs.f
Indices access	getae.f getdst.f getehp.f getgpd.f getgpi.f gethpd.f getimf.f
Read/write records	rdblk6.f vbkopn.f paklr.f cbfopn.f rdasc.f rdchr.f wrflat.f
Logical Record Manipulation	cktim.f findpc.f gxcdv.f maknm.f makt.f ch2a16.f a162ch.f
Models	gapx.f chuiri.f efield.f gchiu.f giri.f gmsis.f msis.f
Plot	gaxtic.f ghttic.f gtmtic.f pltyvt.f clrbar.f
Print	prtr.f lrprt.f hdrprt.f
Location	gspac.f gapx.f gspace.f cnvgdm.f cnvrnh.f
Time	cvt2dn.f cvt2md.f cvti2r.f cvtr2i.f gtime.f newtim.f timdif.f ut2lt.f

There is some functional overlap in two categories. Under “Read/write records”, **rdblk6.f** and **vbkopn.f** each read logical records, but **rdblk6.f** also unpacks them. **vbkopn.f** has other differences, including a write entry, often used in conjunction with **paklr.f** to first pack records. **vbkopn.f** is appropriate for writing and situations where efficiency is gained by only unpacking needed fields. The other reduncancy is in category “Location”. This time the difference is old versus new code design: **gspace.f**, **cnvrnh** and **cnvgdm** are the original spatial location code still used in some programs. In time, they will be replaced by the more robust combination of **gspac.f** and **gapx.f**.

Each subroutine may be used by adding a CALL statement matching the subroutine or alternate entry name and argument list as is described in comments embedded at the beginning of the source code (**~bozo/is/src/*name*.f**). Information passed into and out of these routines always goes through call arguments (i.e., no common blocks are used), thus one must only consider the argument list when studying parameter passing. Once modified, the new program can be compiled and linked with

f77 -O -o*myexecutable mysource.f* **~bozo/is/bin/SunOS/libis.a ~bozo/apex/bin/SunOS/libmag.a**

If using the Linux computer, substitute **Linux** for **SunOS**. If using NCAR graphics, substitute **ncargf77** for **f77**. Library routines can be modified by obtaining a copy of the source, changing it and explicitly listing it during compilation:

f77 -o*myexecutable mysource.f modified lib routine.f* **~bozo/is/bin/SunOS/libis.a ~bozo/apex/bin/libmag.a**

Explicitly listed files are used before searching libraries.

The following example illustrates the additions required to use a time subroutine. Suppose time comparisons are to be made and the input data are in the standard database four integer representation (year mmdd hhmm centisec) when a single number representation is desired. Table 6-1 indicates **cvti2r.f** provides this function, converting from integer to the double precision representation (*yyddd.fraction*). The arguments description in the beginning of file **~bozo/is/src/cvti2r.f** indicates the following declarations and subroutine call would suffice:

```
DOUBLE PRECISION DTIM
PARAMETER (MSGUN=6)
DIMENSION ITIM(4)
    ⋮
CALL CVTI2R (MSGUN,DTIM,ITIM)
```

If the embedded comments do not adequately explain usage, it is possible to find examples by searching the source code files. The UNIX command **grep** can identify examples:

cd ~bozo/is/src
grep "CALL GETAE" *.f

examines all fortran source files for occurrences of the string "CALL GETAE". **grep** returns

getndcs.f: CALL GETAE(MSGUN,ITMP1,324,AE(I) , ISTAT ,

indicating **getndcs.f** has the desired example illustrating declarations, subroutine inputs, and returned arguments.

Some routines use multiple entries which are to used in a prescribed sequence, often involving an initialization.

rdpcf.f The parameter codes file must be loaded into memory (RDPCF) before calling GPCOD (get parameter code descriptors).
rdkif.f The instrument codes file must be loaded into memory (RDKIF) before calling GKNST (get instrument code descriptors).
rdblk6.f Establish a file name unit number correspondence and initialize work buffer with OPENBLK before reading and unpacking logical records (RDBLK6) or rewinding (REWBLK).
vbkopn.f Similarly, VBKOPN must be called before before reading (VBKRD), writing (VBKWR, VBKEOF) or positioning (VBKREW).

The following examples identify calling sequence, call variables, and their declarations.

rdpcf.f has three entries: RDPCF opens and loads the parameter codes file into arrays so subsequent calls to GPCOD or GPCODS can return one or all codes:

```
C         Declarations for GPCOD (entry in RDPCF)
      CHARACTER*40 PCLNAM
      CHARACTER*11 PCSNAM
      CHARACTER*8  PCULBL

C         Declarations for GPCODS (entry in RDPCF)
      PARAMETER (MXNPC=1500)
      CHARACTER*40 PCLNS(MXNPC)
      CHARACTER*11 PCSNS(MXNPC)
      CHARACTER*8  PCULS(MXNPC)
      DIMENSION     IPCS(MXNPC) , PCSFS(MXNPC)
        ⋮
      CALL RDPCF (PCFNAM,IPCUN,MSGUN,IST)

      CALL GPCOD (IPCOD,PCLNAM,PCSNAM,PCULBL,PCSFCT,IST)

      CALL GPCODS (MXNPC,NPC,IPCS,PCLNS,PCSNS,PCULS,PCSFS,IST)
```

where

PCFNAM	=	the parameter codes file name
IPCUN	=	the fortran unit number
MSGUN	=	the fortran unit number to print any diagnostics

IST	=	the return status, where 0 is okay.
IPCOD	=	the parameter code for which information is requested
PCLNAM	=	the long description of IPCOD
PCSNAM	=	the short description of IPCOD
PCULBL	=	the units label
PCSFCT	=	the scale factor portion of the units
MXNPC	=	the dimension of arrays provided to GPCODS for assignment
NPC	=	the number of assigned elements in the arrays returned by GPCODS
IPCS	=	the parameter codes array
PCLNS	=	the long descriptions array
PCSNS	=	the short array
PCULS	=	the unit labels array
PCSFS	=	the scale factors array

Parameter codes attributes are needed when extracting values from a data record. Two more routines are helpful for that purpose: **findpc.f** and **fixlgs.f**. In the following code segment, it is assumed that RDPCF has been called and RDBLK6 has fetched a logical record (LREC). FINDPC returns the parameter code location (LOCPC) and FIXLGS determines when IPC is log10 scaled; If it is, FIXLOG = .TRUE. and the descriptor strings (PCLNAM, PCSNAM and PCULBL) are modified to remove the log scaling text. Note that the subsequent code must do the actual parameter value scaling based on FIXLOG.

```
C          If this is a data record, search for the desired parameter
      IF (LREC(2) .EQ. 1002) THEN
         CALL FINDPC (MSGUN,LREC,IPC,LOCPC)

         IF (LOCPC .NE. 0) THEN
C          Get the scale factor from the parameter codes file and, if
C          appropriate, remove log scaling (changing labels)
           CALL GPCOD (IPC,PCLNAM,PCSNAM,PCULBL,PCSFCT,IST)
           IF (IST .NE. 0) STOP
           CALL FIXLGS (IPC,PCLNAM,PCSNAM,PCULBL,FIXLOG)
C          Determine whether it is a 1D or 2D array parameter
           ISKP = LREC(13)
           NVAL = 1
           IF (LOCPC .GT. LREC(13) + LREC(14)) THEN
             ISKP = LREC(14)
             NVAL = LREC(16)
           ENDIF

C          Convert to real, scaling all but missing data values
           L = LOCPC
           DO 200 I=1,NVAL
           PCVALS(I) = RMISS
           L = L + ISKP
           IF (LREC(L) .NE. MISS) PCVALS(I) = REAL (LREC(L)) * PCSFCT
           IF (FIXLOG) PCVALS(I) = 10.**PCVALS(I)
  200     CONTINUE
         ENDIF
      ENDIF
```

where

LREC	=	a logical record obtained by RDBLK6
MSGUN	=	the fortran unit number for FINDPC to print any diagnostics
IPC	=	the parameter code sought
LOCPC	=	the location of IPC in LREC (returned by FINDPC) or LOCPC=0 when unknown
FIXLOG	=	logical variable returned by FIXLGS. If IPC was found to be log10 scaled, FIXLOG is **.TRUE.** and log10 scaling text was removed from the labels (PCLNAM, PCSNAM, and PCULBL).

NVAL	=	the number of assigned values in PCVAL; NVAL=NROW (the number of rows in the 2D array) or NVAL=1 when IPC is in the 1D array.
MISS	=	**-32767**, the standard missing data code
RMISS	=	**-32767.**, the real version of MISS
PCVALS	=	the array of real values as described by PCLNAM with units PCULBL, except missing values which are assigned RMISS

rdkif.f is analogous to **rdpcf.f**: RDKIF opens and loads the instrument (KINST) codes file into arrays so subsequent calls to GKNST or GKNSTS can return one or all codes:

```
C          Declarations for GKNST (entry in RDKIF)
      CHARACTER*40 KINAM

C          Declarations for GKNSTS (entry in RDKIF)
      PARAMETER (MXNKI=100)
      CHARACTER*40 KINSTS(MXNKI)
      DIMENSION     KINMS(MXNKI), RLATS(MXNKI), RLONS(MXNKI),
                   RELS(MXNKI)

        ⋮

      CALL RDKIF (KIFNAM,IKIUN,MSGUN,IST)

      GKNST (KINST,KINAM,RLAT,RLON,REL,IST)

      GKNSTS (MXNKI,NKI,KINSTS,KINMS,RLATS,RLONS,RELS,IST)
```

where

KIFNAM	=	the instruments codes file name
IKIUN	=	the fortran unit number
IST	=	the return status, where 0 is okay
KINST	=	the instrument code for which information is requested
KINAM	=	the instrument description
RLAT RLON REL	=	are the geodetic latitude, longitude (degrees positive north and east) and elevation (km MSL). If the instrument is mobile, IST = 1 implying the instrument locations embedded in the data (parameter codes 153 and 156) should be used. If the location is unknown, IST = 2.
MXNKI	=	the dimension of arrays provided to GKNSTS for assignment
NKI	=	the number of assigned elements in the arrays returned by GKNSTS
KINSTS	=	the instrument codes array
KINMS	=	the instrument descriptions array
RLATS RLONS RELS	=	are the location arrays. If the instrument is mobile and the locations are in the data records, the positions are returned -998; if unknown, they are returned -999.

rdblk6: In order to read the (COS blocked) binary version of the format, it is necessary to call OPNBLK before RDBLK6. The OPNBLK entry establishes the file name unit number alias and initializes pointers and buffers. Since RDBLK6 works sequentially, it may be necessary to use REWBLK to rewind the data volume. The other entry, CLSBLK, may be used to disconnect dataset and release the unit number. The following example, taken from **rdtp.f**, also shows how large to make the work arrays:

```
      CHARACTER*200 NAME
      PARAMETER (IBUN=10 ,
     +           LMWD=32 , MXVALS=10000 ,
     +           MXWDS=MXVALS*16/LMWD , MXLWRK= 17+20 + 2*512*64/LMWD)
      DIMENSION LREC(MXVALS) ,IUNPK(MXVALS) , IBLK(MXWDS) , IWRK(MXLWRK)

                  ⋮

      CALL OPNBLK (IBUN,NAME,NEOFS,IOCKSM,MSGUN,LMWD,MXVALS,MXWDS,
     +             IWRK,MXLWRK)
```

```
                 :
100 CALL RDBLK6(IWRK,LREC,IBLK,IUNPK,ISTAT)
                 :
    CALL REWBLK (IWRK)
    CALL CLSBLK (IWRK)
```

where the arguments to OPNBLK are all inputs:

IBUN	=	the fortran unit number
NAME	=	the file name
NEOFS	=	the number of files to read before RDBLK6 sets ISTAT to indicate completion. There are two ways to use this. If NEOFS=*n*, where *n* is positive, each RDBLK6 call returns the next logical record and ISTAT=0, even when an end-of-file is encountered. There are two exceptions where no new logical record is returned: upon encountering the nth end-of-file, ISTAT=1; or encountering the end-of-data, ISTAT=3. Usually it is most convenient to set NEOFS = 99999, something larger than the maximum number of files in a volume. NEOFS=0 invokes the other option where RDBLK6 returns control immediately upon encountering end-of-files (ISTAT=1) or the end-of-data (ISTAT=3). This is useful for examining the last LREC in a file.
IOCKSM	=	the checksum flag. If IOCKSM=1 (0), block checksums are (not) verified
MSGUN	=	the fortran unit number to print any diagnostics
LMWD	=	the number of bits in a computer word
MXVALS	=	the maximum number of (16-bit) values in a logical record
MXWDS	=	the number of words needed to hold the largest packed binary block
IWRK	=	a work array provided to OPNBLK and used by RDBLK6, REWBLK, and CLSBLK. A diagnostic is issued if the dimension is insufficient. The formula computing MXLWRK in the parameter statement is adequate if LMWD is properly defined. The contents of IWRK are not normally of interest with the possible exception of the first 17 elements (see also **rdblk6.f** and **cbfopn.f**):

IWRK(1)	=	IBUN
IWRK(2)	=	NEOFS
IWRK(3)	=	MSGUN
IWRK(4)	=	LMWD
IWRK(5)	=	MXVALS
IWRK(6)	=	MXWDS
IWRK(7)	=	Switch (on=1, off=0) to trigger initialization of pointers, counters and arrays. This is set by an OPNBLK or REWBLK call.
IWRK(8)	=	IOCKSM
IWRK(9)	=	Pointer to the beginning of the next logical record in IUNPK
IWRK(10)	=	The number of assigned values in IUNPK
IWRK(11)	=	The number of assigned values in IBLK
IWRK(12)	=	The number of blocks read
IWRK(13)	=	The number of warning messages issued about blocks containing zero filled 16-bit values before the checksum
IWRK(14)	=	The number of blocks which had non-zero 16-bit values after the checksum
IWRK(15)	=	The number of blocks with parity errors
IWRK(16)	=	The number of end-of-file marks read
IWRK(17)	=	The number of logical records read

RDBLK6 input arguments, IWRK, IUNPK and IBLK must match those in OPNBLK; conversely, a single routine may use RDBLK6 to read simultaneously from multiple datasets as long as each uses separate arrays. RDBLK6 return arguments are the logical record (LREC) and the status flag (ISTAT). The layout of LREC is described in the previous **binhnt** example as well as the **CEDAR Database Format** and Appendix B.

vbkopn.f: With entries to read and write, **vbkopn.f** is more generally applicable than RDBLK6. Like RDBLK6, it relies on **cbfopn.f** to handle the COS blocking; unlike RDBLK6, it does not unpack records. Like RDBLK6, an initialization call (VBKOPN) is required before sequential reading (VBKRD) or writing (VBKWR). When writing, it is also important to call VBKCLS to properly terminate the file. Other entries in **vbkopn.f** support rewinding (VBKREW) and writing end-of-file marks (VBKEOF). The following example is excerpted from program asc2is:

```
      PARAMETER(MSGUN=6, IWRUN=12)
C               MSGUN = Fortran unit number to print any diagnostic
C                    messages.
C               IWRUN = Fortran unit number to write the Binary version

C          Declarations for PAKLR and VBKWR (including VBKOPN, VBKCLS)
C            IOCKS  = verify (1) or not (0) check-sums when reading
C            LMWD   = Number of bits in a computer word.
C            MXNVAL = Max # of values in an lrec (2 16 bit values < 16000 bytes)
C            MXLBK  = Maximum packed write block size in (no. words length lmwd)
C            LWK    = Length of work array needed by VBKOPN
      CHARACTER*80 WRNAM
      PARAMETER (WRNAM='asc2is.data.cbf',
     +    IOCKS = 0,
     +    LMWD  = 32 , MXNVAL=7998,
     +    MXLLR = (MXNVAL*16 + LMWD-1)/LMWD ,
     +    MXLBK = ((MXNVAL+2)*16 + LMWD-1)/LMWD ,
     +    LWK   =  13 + MXLBK + 20+2*512*64/LMWD , IRW=2)
      DIMENSION LREC(MXNVAL) , LPKD(MXLLR), IWK(LWK)


      CALL VBKOPN (IWRUN,WRNAM,MSGUN,LMWD,IRW,IWK,LWK,MXLBK,IOCKS)

      ⋮

C          Pack the record and write it
      CALL PAKLR (MSGUN,LREC,LPKD)
      CALL VBKWR (IWRUN,IWK,LPKD,LREC(1),IST)

      ⋮

      CALL VBKCLS (IWRUN,IWK)
```

The pack and write statements are in the middle of a loop in which each iteration formulates a new logical record (LREC) for output.

6.2. PROGRAM MODIFICATION

The database programs should be considered as templates which can be modified to do what you really had in mind: Often a few tweaks make the difference. The basic procedure is to copy and modify the source files requiring change and link the remaining externals during compilation by referencing the object libraries. This is illustrated in a number of examples. First, is a simple case where a dimension must be increased. The next two examples are more complex, showing how one might have to change program **binhnt** and then repeating the exercise with program **gdat**. **binhnt** is described because it is similar to other plot routines (**pltvsh** and **pltvst**). **gdat** is described because it has improvements over the old design; it is the new paradigm.

Before getting involved in a complicated example, consider the situation where **binhnt** was run, but the combination of height and time bins needed exceeded current limits. An error message was issued:

```
0Dimension exceeded!  The product of the number of height and time bins is too large.
 Either increase the bin size or reduce the acceptance criterea range(s).
```

Neither suggestion is tolerable; instead, the program can be changed to increase the number of bins allowed. The procedure mimics the subroutine library example: First obtain a copy of the program

gpgm binhnt.f

then find out where dimensions are specified. Locating the error diagnostic message, shows it was triggered by a test:

```
 IF (NUMHBN*NUMTBN .GT. MXNBNS) THEN
    WRITE (MSGUN,'(''0Dimension exceeded!  The product of the '',
+                                                 ''number '',
+ ''of height and time bins is too large.'',/,'' Either increase ''
+ ,''the bin size or reduce the acceptance criterea range(s).'')')
   STOP
 ENDIF
```

Tracing MXNBNS back to its definition, identifies the statement that must be changed. Comments support this contention:

```
C          Declarations for binning stuff
      PARAMETER (MXNBNS=15000)
C                MXNBNS=Max # of bins (#hts*#tims) allowed in each 2D array;
C                see ACUML8 for an index definition.
```

Bump up **MXNBNS** appropriately and recompile, linking to the unchanged routines via the libarary:

ncargf77 -O -o my.ver binhnt.f ~bozo/is/bin/SunOS/libis.a

If using the Linux machine, substitute Linux for SunOS. Before rerunning, make sure the new executable is referenced in **binhnt.run**: change **binhnt.exe** to **my.ver**.

6.2.1. binhnt Overview

binhnt is the best example of the plot programs because it has essentially the same structure but more complexity than the other plot programs. All get their job control parameters from a subroutine JOBCTL and all employ a main ingest loop involving RDBLK6 and SELHT, but binhnt also has accumulation and averaging logic, more complex display diagnostics and numerous output options. These modifications could be applied to the other plot routines.

Suppose binhnt is to be used to plot electron density height profiles as a function of time but an input dataset contains bad data. (This is taken from real examples). Changes are needed to: (1) correct missing data values from **32767** to **-32767**; (2) multiply the densities by 10 to correct the scale factor from 10^2 to 10^3 for parameter code 520 ($\log_{10}$(N_e/m^3)); and (3) to filter out short pulse length profiles to avoid weak signal to noise ratios from top-side data.

The first step is to obtain a copy of the source code and determine where to insert modifications.

gpgm binhnt.f

will fetch a copy of the main program. The main routine in **binhnt** contains the following key subroutine calls and logic elements (subroutine argument lists are omitted for brevity):

```
      declarations ...
      CALL OPNGKS
      CALL JOBCT1
      CALL OPNBLK
   10 CALL JOBCT2
      IF (DONE) GO TO 9000
      CALL REWBLK
      initialize counters, accumulators for this plot frame ...
  100 CALL RDBLK6
      IF (ISTAT .EQ. 1) GO TO 1000
      reject this lrec if it is not a data record ...
      reject this lrec if wrong KINST ...
      reject this lrec if wrong KINDAT ...
      add third temperature parameter if needed ... (CALL MAKT)
      CALL SELHT
      IF (NFOUND .EQ. 0) GO TO 100
      if appropriate, remove log10 scaling (do 500) ...
      CALL ACUMUL8
      GO TO 100

 1000 CALL AVERAG
      IF (TNTPL8) CALL NTRPY
      IF (LOGPLT) ... log10 scale the plot array (do 2000)
```

```
          compute and print bin counts statistics (label 2200)
          if some bins are not empty then
             identify and print Hmax, max for each time bin (labels 2300,2400)
             CALL PLTBNS
             IF (WRTOP) CALL WRBNS
          endif
          GO TO 10

     9000 CALL CLSGKS
          IF (WRTOP) CLOSE (IWUN)
          END
```

Following the declarations, OPNGKS initializes NCAR Graphics GKS plotting routines, including opening Fortran unit 2 with the name **gmeta**. JOBCT1 is called which reads the first line of the job control file obtaining information for opening the binary data file. OPNBLK, an entry in subroutine RDBLK6, initializes pointers, buffers and opens the data file for reading; actually, a lot of this is done by calling the underlying routine CBFOPN, the COS blocked file access routine. Next, job-control-2 (JOBCT2) reads criteria for one plot from the job control file. JOBCT2 signals there are no more plots to produce by setting DONE to .TRUE.; consequently, control branches to the termination logic at statement label 9000. Otherwise, preparations are completed for reading data needed for the plot: The data file is rewound by REWBLK (a RDBLK6 which calls CBFREW, an entry in CBFOPN) and variables are initialized. The data read loop starts at statement label 100. Each call to RDBLK6 returns the next logical record (LREC) and an okay status condition (ISTAT=0). Data input continues, until the desired number of end-of-files are read, the status flag changes (ISTAT=1) and control transfers to the plotting logic at statement label 1000.

LREC is a single dimensioned integer array containing a logical record (see Appendix B, or better, the **CEDAR Database Format**). The first 16 elements always contain the prologue. If it is a data record, the 1D and 2D arrays follow, one value per element in LREC. In both arrays the parameter codes are listed prior to the data and the 2D array is laid out row-wise; i.e., all values for the first row come first. If it is a header or catalogue record, the prologue is extended to 40 elements (equivalent to one card image in packed form) and the text portion follows, stored as two ASCII characters per element, right justified into the lowest 16 bits.

While ISTAT=0, each record is compared against selection criteria. Sometimes additional parameters are derived. The first tests reject non-data (i.e., catalog or header) records, or the wrong instrument (KINST) code or kind-of-data (KINDAT) code. Next, SELHT is called to assemble and test any spatial parameters. When time parameters are in the data record (as well as the prologue), SELHT calls GTIME to get times for each row in the 2D array and the SELHT tests them. SELHT treats spatial parameters similarly, using GSPACE to make arrays then testing each row. SELHT returns NFOUND acceptable values (VALS), errors (ERRS), heights (HTS) and times (TIMS). VALS and ERRS are floating point numbers derived by multiplying the integer by the power of ten multiplier portion of the parameter definition. TIMS (UT) are the double precision form of the time ($yyddd.fraction$ composed of a two-digit yr (fIyy), day number (*ddd*) and fraction of day ($fract$). HTS (km) are real numbers. Back at the main routine any $\log_{10}$ scaling is removed from VALS and ERRS and these arrays are passed into ACUMUL8. ACUMUL8 determines which (height X time) bins are to be updated and accumulates intermediates needed for weighted mean and variance calculations.

When data input is complete, control branches to statement 1000 where AVERAG is called to complete the binning calculations. Now the arrays can be adjusted optionally interpolating some bins in time (NTRPY) and log10 scaling (DO 2000 and DO 2100 loops). Next, summary tables are printed that describe the binning, before plotting (PLTBNS) and optionally writing (WRBNS) the bins.

6.2.2. **binhnt** Modification

There are a number of ways to fix the problems in the example. Anywhere after the call to RDBLK6 and before ACUMUL8, logic can be inserted to modify the selection process. Rejection of entire logical records, such as the pulse length testing is best located prior to the SELHT call:

```
      DO 300 I=LREC(13)+1,LREC(13)+LREC(14)
      IF (LREC(I) .NE. 402) GO TO 300
      IF (LREC(I+LREC(14)) .LT. 300) GO TO 100
      GO TO 301
  300 CONTINUE
  301 CONTINUE
```

LREC(13) is the length of the prologue and LREC(14) is the number of 1D array parameters. The 1D array follows the prologue and is constructed with all parameter codes preceding values. So “DO 300” loops over the parameter codes in the 1D array, searching for the pulse length (code 402). If the code is found, the datum, located LREC(14) elements later, is tested and if it is less than 300 μsec, control branches to statement label 100 where RDBLK6 fetches another LREC.

After the SELHT returns some values, i.e., following

```
      IF (NFOUND .EQ. 0) GO TO 100
```

is a convenient time to adjust the selected data. SELHT applies the scale factor (power of ten) portion of the parameter code units, when assigning the integers of LREC to the real array VALS. The log scaling for parameters such as code 520 is removed in the DO 500 loop of the main routine. It is in this loop that the power of 10 correction can be accomplished by adding directly after "DO 500..."

```
      VALS(I) = VALS(I)*10.
```

The third program modification, correction of the improper missing data code (32767 -> -32767), could be applied to the entire LREC. Since (16-bit twos complement) integers may be in the range -32767 to 32767, the correction involves testing for one extreme and converting to the other. In this case, it is probably okay to add before the SELHT call:

```
      DO 400 I=1,LREC(1)
      IF (LREC(I) .EQ. 32767) LREC(I) = -32767
  400 CONTINUE
```

LREC(1) is the length of the logical record (the number of assigned elements in LREC), so every element in LREC is tested. It does not matter if some elements are incorrectly changed, because it is not reasonable for the parameters of interest to have values as large as 32767 (an electron density of 10^{32} or height of 32767 km?).

One alternative to changing all occurrences of 32767 would be to add logic to test and eliminate such elements from the VALS array. The commented out loop, “DO 600”, is an example of this elimination logic. In this case, however, it is simplest to modify subroutine SELHT.

gpgm selht.f

will fetch a copy of the source code. SELHT has the following key subroutine calls and logic elements:

```
declarations ...
establish pointers to 1D and 2D arrays ...
initialize returned arguments to nothing found (do 1) ...
locate desired parameter (statement labels 5-30) ...
compare acceptable time interval to prologue times ...
initialize NGROW to all acceptable (do 100) ...
CALL GTIME
test each row for time (do 200) ...
CALL GSPACE
test each row for spatial parameters (statement labels 310-500) ...
assign returned acceptable "rows" values (do 1000) ...
```

Initially SELHT attempts to eliminate the entire LREC if the prologue begin or end time is unacceptable or if the desired parameter is not available. If those tests are passed, GTIME and GSPACE are called to build times and spatial parameters for each row. NGROW(I) is set to 1 for any row failing a selection test. Those rows surviving all tests are assigned to the returned arrays; ERRS units match VALS, HTS in kilometers, and TIMS.

The 32767’s in the example could be avoided by adding one line to the DO 1000 loop.

```
      IF (LREC(LV) .EQ. 32767) NGROW(I) = 1
```

In general, modifications involving testing each row are easily inserted any time after NGROW is initialized and before the final assignment loop (DO 1000). Suppose electron density is not acceptable when the signal to noise ratio (parameter code 402) is below a threshold (MINOK), then the following might be added just before "DO

1000":

```
      LSNR = 0
      DO 600 I=IB2DC,IE2DC
      IF (LREC(I) .EQ. 402) LSNR = I
  600 CONTINUE
      IF (LSNR .NE. 0) THEN
         DO 610 I=1,NV
         LSNR = LSNR + LREC(15)
         IF (LREC(LSNR) .LT. MINOK) NGROW(I) = 1
  610   CONTINUE
      ENDIF
```

The first loop locates the desired parameter. IB2DC IE2DC and NV are all defined by SELHT to be the beginning of 2D parameter codes, the end of 2D parameter codes and the maximum number of values to return (either the number of rows in the 2D array or 1 when there is only a 1D array). The second loop increments LSNR by one row (the number of 2D parameters) at the same time the loop index, I, is incremented, such that LSNR is the location of the signal to noise ratio in the Ith row.

Once the program and any subroutines have been modified, they may be compiled and linked to routines in the general subroutine library and NCAR Graphics libraries with

ncargf77 -O -o binhnt.exe binhnt.f selht.f ~bozo/is/bin/SunOS/libis.a

If using the Linux computer swap Linux for SunOS. The name (**binhnt.exe**) must match that used in the job execution script. If the name is not changed from the original, as in this example, then the execution script should be in the same directory as the executable so that the new version is found before searching other directories. (This assumes the variable **$path** is defined to look in the current directory first; see Appendix A.)

In order to make such changes it is often necessary to gain an intimate knowledge of the data volume. Beyond examining **~bozo/is/docs/akavsns** for caveats and studying detailed inventory tables (such as **~bozo/is/docs/catprt.list** or a **cmenu** detailed inventory output), this implies at least running **rdtp** to obtain samples of logical records. It may also be necessary to use **pltvst** to examine time series; e.g., the threshold acceptable signal to noise ratio could be deduced this way.

Changing source code is error prone. Compiler errors are usually easily identified via print diagnostics but execution time errors may not be so trivial to locate. Those familiar with **dbx** will find the interactive debugger useful, however, it is often sufficient to add a few well placed debug print statements; e.g.,

```
     PRINT *,'counter I=',I,' and array X=',X
```

Executable files tend to be fairly large, **binhnt.exe** is almost 1 MB. So it is possible that the current disk allocation is insufficient. In this case, attempts to compile, will result in an error diagnostic and the compilation will fail. The current allocation can be determined by entering

quota -v

This problem may be circumvented by working in **/d/***yourlogin* or **/tmp**, but neither directory is backed up and **/tmp** is scrubbed daily. The permanent solution is to request more disk space.

6.2.3. **gdat** Overview

The data selection logic in program **gdat** (get data) used to be similar to the plot programs, but recent changes make it a better starting point for making custom access code.

- Apex coordinates are now available via the fast interpolation logic. This means Apex latitude and longitude may be used for testing and may also be selected for output.
- One may now specify multiple KINST and multiple KINDAT codes for data selection, whereas binhnt allows only one of each.
- One may now specify additional parameters and acceptable limits for each. binhnt had no such option, so in the above example it was necessary to modify the program to select on pulse length.
- The new selection logic is more forgiving when selecting parameters located both 1D and 2D arrays. One had to be careful with the old code and some combinations were impossible, such as a 1D primary parameter with 2D

test parameters.

- The old spatial parameters calculation scheme (subroutine SELHT) has been replaced and instrument position can now vary. The new version can compute sample location (latitude, longitude and height) from a record containing instrument latitude, longitude, elevation angle, azimuth angle and slant range to the target. (The addition of airplane borne lidar data motivated this change.)
- The selection and testing logic formerly in SELHT has been moved so it is mostly accessible from the main routine. Subroutines GTIME and GSPAC and their returned time and space variables are no longer buried two calls deep. The changes in the binhnt example can all be made in the main program of gdat.

As with the binhnt description, the following outline of gdat's main routine depicts the branching logic by eliminating some details (subroutine argument lists) and substituting brief descriptions for short logic blocks:

```
      declarations ...
      NRUN = 0
   10 NRUN = NRUN + 1
      CALL JOBCTL
      if NMBIN is all blanks, go to 2000
      IF (NRUN .EQ 1) THEN
         CALL RDKIF
         CALL GAPXI
      ENDIF
      CALL GDWRI
      CALL GPCG
      CALL SETK
      CALL OPNBLK
      CALL REWBLK
      CALL CBFSKF
  100 CALL RDBLK6
      IF (ISTAT .EQ. 1) GO TO 1000
      reject this lrec if it is not a data record
      reject this lrec if wrong KINST ...
      reject this lrec if wrong KINDAT ...
      reject this lrec if completely out of okay time interval ...

      IF (NNTC  .NE. 0) CALL ADTC
      IF (NNNMC .NE. 0) ... CALL GSPAC and ADNMC

      CALL FINDPC
      IF (LPC .EQ. 0) THEN
         When it is calculated space/Apex, add it to the list for GXCDV
      ELSE IF ...
         Otherwise, establish pointers to the code values in the rec
      ENDIF
      initialize NGROW (do 300)
      CALL GTIME
      test times (do 310)
      IF (KSPAC .NE. 0) THEN
         IF (KSPAC .EQ. 1) CALL GSPAC
         CALL CSPAC
      ENDIF
      IF (NTCD.NE.0) CALL CTCD
      IF (NXCD .NE. 0) CALL GXCDV

      set first and last okay row (do 330, do 340)
      write the values for all ok rows in this rec (do 410)
      GO TO 100

 1000 CALL CLSBLK
      GO TO 10

 2000 CLOSE (IWUN)
      END
```

Following the declarations, a counter (NRUN) is incremented and JOBCTL is called to read the next batch of job control parameters. If none are found, the binary read file name (NMBIN) returned is blank, and the program branches to 2000 and quits.

Otherwise, preparations are made for data selection. If this is the first run (NRUN = 1), the instrument codes file is loaded into memory by RDKIF and the Apex interpolation routines are initialized by GAPXI; the parameter codes file is opened elsewhere (in JOBCTL). Next, the list of parameters to be output is formed and GDWRI writes this list with parameter descriptions to the data output file. GCPG (get parameter code groups) is called to to see whether temperature or Nmax,Hmax calculations may be needed. SETK performs similar tasks, setting flags indicating whether spatial position parameters may need to be calculated. Finally, OPNBLK is called to open the binary data file, then REWBLK and CBFSKF position the file forward the desired number of files.

The RDBLK6 call starts the main logic loop on statement label 100 by reading a data record. If the end of data is encountered instead (ISTAT = 1), then control branches to statement 1000 where the data read unit is closed and the process starts again with a branch to statement 10. If a data record was read, then tests are made to reject the entire record based on prologue fields KINST, KINDAT and begin and end time. If any of these fail, logic branches back to 100 to read another record. So far, the logic closely resembles binhnt; it is the rest of the main routine that has the most change. What used to be buried in SELHT in binhnt has been relocated here, beginning with CALL FINDPC.

The primary parameter code is sought in subroutine FINDPC and its position (LPC) established in the record (LREC). The location of the data value (LV) and the number of elements to skip to the next datum (ISK) are defined to distinguish 1D and 2D array values. If the parameter code is not in the record, logic branches to statement 100 unless the code is a calculated spatial location (KIPC = 1); in this case the code is added to the list input to GXCDV (get extra code values). Next, the no-good-row flag array (NGROW) is initialized; it contains flags indicating whether (NGROW(I) = 0) or not (NGROW(I) = 1), row I, passes the selection tests. After building times (GTIME) and spatial parameters (GSPAC), testing is performed on each row. Switch KSPAC keeps from calling GSPAC twice, in case it was called previously to build heights for the Nmax,Hmax calculation. Testing each row for time is done in the DO 310 loop. Testing space parameters is done in subroutine CSPAC. Testing of any extra parameter code values is in subroutine CTCD.

GXCDV normally just fetches the other parameters requested from the record but it also has logic to extract values from the spatial position arrays produced by GSPAC: height (HT), range (R), elevation angle (EL), azimuth angle (AZ), geodetic latitude (GDLAT), geodetic longitude (GDLON), geomagnetic latitude (GMLAT), Apex latitude (ALAT) and Apex longitude (ALON). Basically, for each code in IXCD, GXCDV assigns values to array IXCDV.

After GXCDV, the last task before writing is to establish the first okay row (IBRO) and the last row (IERO) to write. Finally, all acceptable rows from the record are written in the DO 410 loop. In that loop, the second version of the write statement is used when the desired code is a calculated spatial parameter.

6.2.4. gdat Modification

Given the conditions from the previous example, **gdat** is easier to change. To reiterate, the three conditions are to: (1) correct missing from 32767 to -32767; (2) multiply densities by 10; and (3) filter out short pulse records. Filtering data by pulse length can be accomplished by adding this criterion to the job control inputs; see Chapter 4 and use of *ITCD(I)*, *ITMN(I)*, and *ITMX(I)*. The other two conditions are addressed by modifying the DO 410 loop; the two added lines are in bold print:

```
    DO 410 I=IBRO,IERO
    LV = LV + ISK
    IF (NGROW(I) .EQ. 1) GO TO 410
    IF (LREC(LV) .EQ. 32767) GO TO 410
    LREC(LV) = LREC(LV) * 10
    NLW = NLW + 1
    IF (LV .EQ. 0) GO TO 400
    WRITE (IWUN,DFMT)(IAVGT(J,I),J=1,4),LREC(LV),(IXCDV(K,I),K=1,NXCD)
    GO TO 410
400 WRITE (IWUN,DFMT)(IAVGT(J,I),J=1,4),         (IXCDV(K,I),K=1,NXCD)
410 CONTINUE
```

The new IF test, identifies the bad missing codes (32767's) and avoids writing those lines. The power of ten scaling can be corrected on the next line because all bad values have been rejected by the previous two lines.

Compiling and executing the modified code is differs from the **binhnt** example: The geomagnetic library is added and **f77** is used because there is no need to link to NCAR graphics libraries:

f77 -O -o gdat.exe gdat.f ~bozo/is/bin/SunOS/libis.a ~bozo/apex/bin/SunOS/libmag.a

If using the Linux computer, swap Linux for SunOS. The executable name (**gdat2.exe**) is chosen to match that specified in the script so that it will be found and supercede the standard version (~bozo/is/bin/SunOS/gdat.exe). This presumes the directory search path has been established to search the local directory first.

6.3. MODELS

Some models have already been introduced. **pltvsh** (in Chapter 5) optionally plots profiles of MSIS, Chiu and IRI models and **gdat** (in Chapter 4) employs the Apex geomagnetic field model. This section reviews the other models or model output available, but first references for all models are listed in Table 6-4. Source code is available on **cedar** except for FLIP, TGCM, and VSH:

APEX
: Directory **~bozo/apex** contains the Apex model including routines for time saving interpolation, modified Apex coordinates and quasi-dipole coordinates. This package employs the International Geomagnetic Reference Field (IGRF) using DGRF (Definitive) coefficients for 5 year intervals from 1965 to 1990 and IGRF (Interim) for 1995. A **README** file introduces each file and explains how to install and test the programs.

Chiu and IRI-86
: File **~bozo/is/src/chuiri.f** is a series of subprograms containing the Chiu and IRI-86 (a.k.a. Version 9) models where IONDENZ and IRIDRV are the respective interface routines. Alternatives, subroutine GCHIU (**~bozo/is/src/gchiu.f**) or GIRI (**~bozo/is/src/giri.f**), obtain geophysical indices before IONDENZ or IRIDRV. The coefficients required by IRI-86 are in **/usr/madrigal/files/stage/iricofcnts**.

Efield
: Subroutine EFIELD (**~bozo/is/src/efield.f**) gives quiet-day ionospheric electrostatic pseudo-potential and **E** X **B** drifts at 300 km for solar minimum conditions.

FLIP
: The Field Line Interhemispheric Plasma model solves for plasma parameters, production and cooling rates, and emissions along a field line. An account on the Cray computer is required. Copy the contents of the shavano directory **/u2/richards/demo**, then execute **fliphelp.csh** to obtain further instructions. The source code is not available. Questions should be directed to Phil Richards (University of Alabama at Huntsville, Computer Science Department, Research Institute C-10, Huntsville, AL 35899, phone: (205) 895-6238x375, email: richards@cspara.dnet.nasa.gov).

Glow
: **~stans/glow** is a directory containing the Glow model where **glow.doc** introduces use of the various routines, and describes the implementation of IRI-90 routines, and requests notification of usage.

HMR
: The Heppner Maynard Rich model computes the electrostatic potential and electric fields poleward of ±50° magnetic latitude. The model source, a test example and directions are in directory **~bozo/pgms/hmr**.

HPI
: The Hemispheric Power Index auroral model computes conductances given magnetic latitude, magnetic local time and hemispheric power index. The model source, a test example and directions are in directory **~bozo/pgms/hpi**.

HWM
: Two versions of the Horizontal Winds model (HWM-90 and HWM-93) by Al Hedin are available in directory **~bozo/hedin**. File **readme** introduces the model source files, example driver and test output files.

IGRF
: (**~bozo/apex/src/magfld.f**) are the IGRF with DGRF coefficients for 5 year intervals from 1965 to 1990 plus IGRF for 1995. Program IGRF (**~bozo/is/src/igrf.f**) illustrates the use of COFRM and FELDG. **~bozo/is/run/igrf.run** is an example script to execute IGRF.

IRI-86
: See Chiu (above).

IRI-90
: **~bozo/pgms/iri** is a directory containing IRI-90 source (**iri90.f**), a test driver (**testiri90.f**), coefficients files (**ccir.cofcnts ursi.cofcnts**) and documentation (**readme**).

TABLE 6-4. Models

Acronym	Reference
Apex	Van Zandt, T. E., W. L. Clark, and J. M. Warnock, Magnetic apex coordinates: A magnetic coordinate system for the ionospheric F2 layer, J. Geophys. Res., 77, pp 2406-2411, 1972.
	Richmond, A. D., Ionospheric Electrodynamics Using Magnetic Apex Coordinates, J. Geomag. Geoelectr., 47, 191-212, 1995.
Chiu	Chiu, Y. T., An improved phenomenological model of ionospheric density, J. Atmos. Terr. Phys., 37, pp 1563-1570, 1975.
Efield	Richmond, A. D., M. Blanc, B. A. Emery, R. H. Wand, B. G. Fejer, R. F. Woodman, S. Ganguly, P. Amayenc, R. A. Behnke, C. Calderon, and J. V. Evans, An empirical model of quiet-day ionospheric electric fields at middle and low latitudes, J. Geophys. Res., 85, pp 4658-4664, 1980.
FLIP	Torr, M. R., D. G. Torr, P. G. Richards, and S. P. Yung, Mid- and low-latitude model of thermospheric emissions 1. O+(2P) 7320 A and N2 (2P) 3371A, J. Geophys. Res., 95, pp 21147-21168, 1990.
	Richards, P. G., An improved algorithm for determining neutral winds from the height of the F2 peak electron density, J. Geophys. Res., 96, pp 17839-17846, 1991.
Glow	Solomon, S. C. and V.J. Abreu, The 630 nm Dayglow, J Geophys. Res., 94, 6817, 1989.
	Solomon, S. C., and R. G. Roble, Simulation of the global thermospheric airglow, 1, methodology, J. Geophys. Res., submitted, 1993.
HMR	J.P. Heppner and N.C. Maynard, Empirical high-latitude electric field models, JGR, 92, 4467-4489, 1987.
	F.J. Rich and N.C. Maynard, Consequences of using simple analytical functions for the high-latitude convection electric field, JGR, 94, 3687-3701, 1989.
HPI	T.J.Fuller-Rowell and D.S. Evans, Height integrated Pedersen and Hall conductivity patterns inferred from the TIROS-NOAA satellite data, JGR 92, pp 7606-7618, 1987.
HWM-87	Hedin, A.E., N.W. Spencer, and T.L. Killeen, Empirical global modal of upper thermosphere winds based on Atmosphere and Dynamics Explorer satellite data, J. Geophys. Res., 93, pp 9959-9989, 1988.
HWM-90	Hedin, A. E., M. A. Biondi, R. G. Burnside, G. Hernandez, R. M. Johnson, T. L. Killeen, C. Mazaudier, J. W. Meriwether, J. E. Salah, R. J. Sica, R. W. Smith, N. W. Spencer, V. B. Wickwar, and T. S. Virdi, Revised global model of thermosphere winds using satellite and ground-based observations, J. Geophys. Res., 96, pp 7657-7688, 1991.
IGRF	Kodama, K.P. ed., Revision of International Geomagnetic Reference Field Released, EOS 77 16, p 153, 16 April 1996.
IRI-86	Bilitza, D., International reference ionosphere: Recent developments, Radio Sci., 21, pp 343-346, 1986.
IRI-90	Bilitza, D., International Reference Ionosphere 1990, National Space Science Data Center/World Data Center A for Rockets and Satellites, NSSDC/WDC-A-R&S 90-22, Nov 1990.
IZMEM	V.O. Papitashvili, et. al., Electric potential patterns in the northern and southern polar regions parameterized by the interplanetary magnetic field, JGR 99, pp 13251-13262, 1994.
MHI	Foster, Holt, Musgrove and Evans, GRL, 13, pp 656-659, 1986.
MHS	Foster, Proceedings of the International Symposium on Quantitative Modeling of Magnetosphere-Ionosphere Coupling Processes, editors Kamide and Wolf, Kyoto, Japan, pp 71-76, 1987.
MSIS-83	Hedin, A. E., A revised thermospheric model based on mass spectrometer and incoherent scatter data: MSIS-83, J. Geophys. Res., 88, pp 10170-10188, 1983.
MSIS-86	Hedin, A. E., MSIS-86 thermospheric model, J. Geophys. Res., 92, pp 4649- 4662, 1987.
MSIS-90	Hedin, A. E., Extension of the MSIS thermosphere model into the middle and lower atmosphere, J. Geophys. Res., 96, pp 1159-1172, 1991.
PESIMP	Richards, P. G. and D. G. Torr, Ratios of photoelectron to EUV ionization rates for aeronomic studies, J. Geophys. Res., 93, 4060, 1988.
TGCM	Dickinson, R. E., E. C. Ridley, and R. G. Roble, A three-dimensional general circulation model of the thermosphere, J. Geophys. Res., 86, pp 1499-1512, 1981.
	Roble, R. G., R. E. Dickinson, and E. C. Ridley, Global circulation and temperature structure of thermosphere with high-latitude plasma convection, J. Geophys. Res., 87, pp 1599-1614, 1982.
	Dickinson, R. E., E. C. Ridley, and R. G. Roble, Thermospheric general circulation with coupled dynamics and composition, J. Atmos. Sci., 41, pp 205-219, 1984.
TIGCM	Roble, R. G., E. C. Ridley, A. D. Richmond, and R. E. Dickinson, A coupled thermosphere/ionosphere general circulation model, Geophys. Res. Lett., 15, pp 1325-1328, 1988.
TIEGCM	Richmond, A. D., E. C. Ridley, and R. G. Roble, A thermosphere/ionosphere general circulation model with coupled electrodynamics, Geophys. Res. Lett., 19, pp 601-604, 1992.
VSH	VSH Model Documentation, Version 2.1, Space Physics Research Laboratory, The University of Michigan, October, 1991.
Weimer	D.R. Weimer, Models of high-latitude electric potentials derived with a least error fit of spherical harmonic coefficients, JGR 100, pp 19595-19607, 1995.
	D.R. Weimer, A flexible IMF dependent model of high latitude electic potentials having “space weather” applications, GRL, 23, 18, pp 2549-2552, 1996.
Model Compendium	Bilitza, D., Solar-Terrestrial Models and Application Software, National Space Science Data Center/World Data Center A for Rockets and Satellites, NSSDC/WDC-A-R&S 91-26, 1991.

IZMEM The IZMIRAN electrodynamic model produces electrostatic potential and electric fields poleward of ±57° magnetic latitude based on ground-based magnetometer data. The model source, a test example and directions are in directory **~bozo/pgms/izmem**.

MHI and MHS

Two Millstone Hill electric potential, electric field models are based on incoherent scatter data. One model requires hemispheric power index input while the other requires IMF. The model source, a test example and directions are in directory **~bozo/pgms/mlh**.

MSIS-83 and MSIS-86
: File **~bozo/is/src/msis.f** is a series of subprograms containing the versions MSIS-83 and MSIS-86. GTSG4 and GTSG5 are the respective interface subroutines. Alternative subroutine GMSIS (**~bozo/is/src/gmsis.f**) obtains geophysical indices before calling GTSG4 or GTSG5.

MSIS-90
: **~bozo/hedin** is a directory containing MSIS-90 HWM-90 and HWM-93. File **readme** introduces the model source files, example driver and test output files.

PESIMP
: Photo-Electron SIMPle model. A single file (**~bozo/pesimp.f**) contains the source code, instructions for use, and test output.

T*GCM
: There are various versions of Thermosphere General Circulation Models (TGCM) under the control of Dr. R. G. Roble at HAO/NCAR which are available for the use of the community. The present models are:

TIGCM	Thermosphere/Ionosphere General Circulation Model
TIEGCM	Thermosphere/Ionosphere/Electrodynamics General Circulation Model
TIMEGCM	Thermosphere/Ionosphere/Mesosphere/Electrodynamics General Circulation Model

The TGCM models constantly change (updating heating/cooling rates, using different parameterizations, etc.) so model output is generally not retained for more than one year. Exceptions include published work. Specific runs done for a campaign interval should be revised periodically in order to reflect recent improvements. Requests for simulations of specific periods or generic conditions may be directed to R.G. Roble or B.A. Emery (see Cedar Database Catalogue for addresses). Various post-processors are available to produce print tables or plots of model output: Capabilities include location processors, longitude/latitude slices, global views, and determination of intermediate parameters.

Weimer
: The Weimer electric potential, electric field model adds dipole tilt to required inputs IMF component and location to produce seasonal electric potential patterns. The model source, a test example and directions are in directory **~bozo/pgms/weimer**.

The model compendium is included in Table 6-4 as a general reference summary of models. It may be obtained from

Space Science Data Center (NSSDC)
Code 933.4
Goddard Space Flight Center
Greenbelt, MD 20771
(301) 286-6695
telnet: nssdca.gsfc.nasa.gov (128.183.36.23)
user: NSSDC or NODIS

The Vector Spherical Harmonic (VSH) Model is included because of potential interest even though it is not available on cedar. (Also, it is not described in the Model Compendium.) VSH is a compact representation of multiple TIGCM runs created on a Cray which will fit on smaller computers. VSH defaults to MSIS below 110 km and above 500 km. VSH can be obtained by contacting:

Rob Raskin
Space Physics Research Laboratory
University of Michigan
2455 Hayward St
Ann Arbor MI 48109-2143
(313) 747-3450

APPENDIX A

UNIX HELP

This is a summary of environment criteria and commands related to the database implementation at NCAR on a computer running UNIX (SunOS 5.5.1 as of November 1998). Login procedures and account setup criteria are discussed followed by an introductory section summarizing common commands. The next section goes into details of transferring files to another computer. The last section summarizes the **vi** editor, supplied with all UNIX operating systems. Two other editors are available on cedar but not described here: **emacs** and **re**. Both have **man** descriptions plus **emacs** commands are summarized in file **~ftp/pub/gnu-emacs.bindings**.

A.1 Login Procedures

As of September 1998, a security measures were installed at NCAR to minimize exposure to the rest of the internet. A "gatekeeper" security system filters internet traffic coming into NCAR computers. This affects access from outside, including login procedures, changing passwords, email addresses, X window system permissions and file transfer via rcp and ftp. The consequences are most noticeable during login from an external host; you must first connect to the gatekeeper:

telnet gate.ucar.edu

The response will be:

```
Trying 128.117.7.112 ...
Connected to gate.ucar.edu.
Escape character is '^]'.

Username:
```

Respond to this with your login, then give your password when it is prompted. Thereafter, the prompt will be

```
ucar-telnet-proxy>
```

Switch to the cedar computer, by entering

c cedar.ucar.edu

The response will be:

```
Trying 128.117.16.90 ...
Connected to 128.117.16.90.
Escape character is '^]'.


UNIX(r) System V Release 4.0 (cedar)

login:
```

Enter your login once more and then your cedar password, which is is probably different than the gateway password. After successfully connecting, most interactions should occur as if the gatekeeper were not there. For instance, **rcp** and **rsh** initiated from cedar will work as before. **ftp** is similar providing the external computer honors the PASV (passive) mode; if it doesn't, then it will be necessary to use the gatekeeper as a proxy. Recipes for this and other gatekeeper related procedures are described in URL **http://www.hao.ucar.edu/public/home/gatekeeper/** a version of which is in files on cedar formatted in Postscript (**/home/sitongia/adm/Projects/security/HAO-model.ps**) or PDF (**/home/sitongia/adm/Projects/security/HAO-model.pdf**).

To change your cedar password, use **nispasswd** when logged in on cedar. To change your gatekeeper password, use one of two methods. When you are logged into **gate.ucar.edu** (e.g. during the login process), you can enter **password** to start a prompt and response sequence. Otherwise during a cedar login session, you can access a menu driven system by entering:

rlogin -l directory directory.ucar.edu

then select

```
C.  Change Gatekeeper Password
```

and respond to prompts to complete the change.

Occasionally a bug affects the login procedure resulting in replacing current screen size (rows and columns) with an incorrect default. The problem is intermittent, so logging out and in may correct it, however, it is recommended that you first execute a simple script to fix this problem: enter **rs** (to execute **~bozo/bin/rs assuming your $path** variable is appropriately set). Failing that, rows and columns may be set manually: Before logging in, determine your screen size (by entering **stty -a** on UNIX systems). Once logged in, establish the correct size by entering

stty rows *n*
stty columns *m*

Alternatives for C shell and Borne users are explained in the resize man page; for instance, in an interactive C shell enter

set noglob; eval ‘resize‘

Odd terminal response may also be caused by incorrect terminal definition. The current setting may be determined by entering

env | grep TERM

If suspicious of a mismatch, request help by sending email to **bozo@ucar.edu**.

A.2 Computing Environment

UNIX provides a number of ways to tailor the computing environment. One can automate customizations each login session by defining them in two special files in your home directory: **~/.login** and **~/.cshrc**. “**~**” is an abbreviation for your home directory; see **man csh**. Use **ls -a** to list such files beginning with a dot; otherwise they are not displayed by **ls**. When logging in, first **.cshrc** is read then **.login**; in fact **.cshrc** is read before starting any new process while **.login** is read only once. Attributes established in **.login** are apropos interactive activities, such as terminal characteristics. Attributes in **.cshrc** are also passed to non-interactive tasks, such as aliases used in scripts. If either **.login** or **.cshrc** are modified during a session, changes are not effective until the commands in the files are executed. This may be achieved using **source**, as in

source ˜bozo/is/bin/envcdb

which coincidently defines environment variables and search path and $MANPATH for CEDAR database commands. Environment variables are listed by entering **env** as in

env | sort | more

Three NCAR Graphics plotting utilities (**ctrans**, **ictrans**, and **idt**) use environment variables. **ctrans** is the basic computer graphics metacode file (cgm) translator. **ictrans** is the interactive version for graphics terminals and **idt** is the X windows version. All three need to know where executables are located and a plot device type. The executable location is established in the system Cshrc file:

setenv NCARG_ROOT /opt/local

The device type may be defined either by adding to the command an argument (e.g., **-d X11**) or an environment variable:

setenv GRAPHCAP *display*

where *display* is the terminal type. A list of supported terminals is produced by entering **gcaps**. Settings may be automated by inclusion in **.cshrc**, or just entered each time they are needed. Thus, **ctrans** is invoked by entering

ctrans -d *device my.cgm*

or, if GRAPHCAP has been defined,

ctrans *cgm.file*

whereupon the first frame in *cgm.file* is displayed. Frames are advanced by pressing return. **ictrans** is invoked similarly, but allows more options such as arbitrary ordering of plot frames to display; once invoked enter **help** for information about these options. **idt** is invoked in the same manner; details of additional initializations are given in the introduction to Chapter 5 (Plot Programs).

A few example customizations added to a **.login** file are:

```
#          For Rand editor
setenv EDITOR  /opt/local/bin/re
#          Redefine backspace, interrupt and kill
stty  erase '^h' intr '^?' kill '^u'
echo 'erase=BS(^h)  intr=DEL  kill=NAK(^u)'
#          Change the fancy prompt (with current directory) to just hostname
set prompt     = "`uname -n`: "
unalias setprompt
```

Lines beginning with "#" are comments. EDITOR establishes which editor to use when using **mailx**. **stty** overrides the default backspace, delete, and kill key definitions and **echo** displays a reminder when logging in.

Some useful alias customizations in the **.cshrc** file are:

```
#          aliases which need to be available in every shell:
alias dir   ls -laF
alias ldir  'set a=-l ; if ("*" != "") set a = -ld ; ls $a * | grep "^d"'
alias lfil  "ls -lF * | awk 'NF==0 {exit} ; /^[^d]/'"
```

A list of the current aliases is produced by entering **alias**.

Some initializations may have hidden side effects. For instance, Madrigal includes

limit coredumpsize 0

which truncates diagnostic **core** files. This is convenient unless debugging (e.g., with **dbx**). **limit** displays current limits and the restriction may be removed by entering, for example

unlimit coredumpsize

A.3 UNIX Introduction

Complete descriptions of commands may be obtained using **man** as in

man *command*

By default **man** sends its output to be displayed by **less** (as established by the environment variable PAGER). There are a number of ways to print man pages: If physically located at HAO,

man -t *command*

will send output to the printer defined by the environment variable PRINTER. If the desired printer is on another UNIX machine accessible on internet (and **.rhosts** permissions are established)

setenv TCAT cat
man -t *command* **| /usr/lib/lp/postscript/dpost | rsh** *my.host* **lp -d***my.printer*

will convert the man page documentation for *command* to Postscript and send it to *my.printer* on *my.host*. If *my.host* is not UNIX or it is not a Postscript printer, there are other options, the simplest of which may be to produce an ASCII rendition with simulated underlines

man *command* **| ul -i > temp.file**

and subsequently transfer **temp.file** using **ftp**.

The following lists summarize commonly used features of UNIX commands. There are four alphabetically ordered categories. The descriptions are brief in order to introduce as many commands as possible, thus, masking complexity of some commands.

Command arguments which are file names may be generalized by using pattern matching characters (*****, **?**, **[...]**) or substitution characters (**~** or **{...}**). ***** matches any string, including no characters; e.g., **ls old*** will list any files in the current directory beginning with **old**. **?** matches any single character. **[...]** matches any characters in the list (**...**) in which a lexical range of characters is indicated by **-**, so **[abd-f]** would match **a**, **b**, **d**, **e**, or **f**. **~** at the beginning of a file name refers to a home directory. Standing alone, it is expanded to the invoker's home directory but when followed by a name it is expanded to another user's home directory, so **~ken** might expand to **/usr/ken**, **~ken/file** to **/usr/ken/file** and **~/file** would expand to the full path name of **file** in the invoker's home directory. Substitutions designated by curly brackets present alternatives; e.g., **a{b,c,d}e** is shorthand for **abe ace ade**. See **man csh** for more details.

File and Directory Management Commands

cat *file*	display (concatenate) *file*
cat *files* > *combined.file*	Concatenate list of *files* and redirect output to create *combined.file*
cd *directory*	Change to *directory*
cd	Change to home directory
cd ..	Change to parent directory (up the hierarchy)
chmod 754 *name*	Change file or directory (*name*) permission mode to user: read, write and execute (7); group: read and execute (5); other: read (4).
cp *file newfile*	Copy *file* to *newfile*
cp *files directory*	Copy list of *files* to *directory*
file *file*	Identify the nature of *file* (e.g., ascii, binary)
find *path.list expression*	Find all files matching *expression* in directories in and below *path.list*; e.g., **find ~ -name "*.f" -print** searches recursively through directories starting with home and prints path names for any files ending in "**.f**".
head *file*	Display first few lines of *file*
less *file*	Fancy version of **more**
ln *file1 file2*	Link files: Create a second name (*file2*) for *file1*
ls	List files and subdirectories in the current directory
ls -alF	List in long format including names beginning with "."; and append "/" to directories, "=" to sockets, "@" to links, and "*" to executables.
ls -R	Recursively list contents of this directory and those below it
more *file*	Display contents of *file* one screenfull at a time
mv *file newname*	Rename (move) *file* to *newname*
mv *files directory*	Move list of *files* to *directory*
mkdir *dirname*	Make new (sub)directory named *dirname*
pwd	Display full current pathname (print working directory)
rm *files*	Remove *files*
rm -rf *dirname*	Forcibly remove directory *dirname* and recursively remove all its contents including any subdirectories
rmdir *dirname*	Remove empty directory *dirname*
tail *file*	Display last few lines of *file*
tar *key name*	Tape archiver saves and restores multiple files on a single file
tee *files*	Read from standard input and write to standard output and *files*

Text Processing Commands

awk *string files*	Execute awk commands in *string* on *files*. Commands are of the form *pattern* {*action*}; e.g., **awk '$1~/CALL/ {PRINT $2}' file** displays the second blank delimited string of each line containing "CALL" in the first string.

cmp *file1 file2*	Compare two files and list the first location of a difference
diff *file1 file2*	List the differences between two text files
emacs *file*	Start the emacs (eats memory and crashes system) editor on *file*. Exit without saving changes by entering **^xc** (press control and x then control and c). If there were changes, a prompt will follow asking if if they are to be saved and a backup file named **#***file***#** will be created.
enscript *file*	Convert text file to PostScript and send to default printer
grep *string file*	List lines from *file* which contain *string*
lpstat	Show printer queue for default printer
lp *file*	Spool and print file to default printer
cancel *jobnumber*	Remove job from printer queue (*jobnumber* may be ascertained using **lpq**).
pr *file*	Paginate and print *file*
sed *options files*	Stream editor, noninteractive text editor; e.g., **sed -n 10,30p file** prints lines 10 to 30 of **file**.
re *file*	Start the Rand Editor on *file*. Exit without saving changes by entering **^a** (control a), then entering **ex a**
sort *file* > newfile	Sort the lines in *file* and redirect output to *newfile*
spell *file*	List incorrect spellings in *file*
tr *options str1 str2*	Translate characters: **tr** reads from standard input and writes to standard output, translating *str1* to *str2*; e.g., **tr "[a-z]" "[A-Z]" < input.file > output.file** converts alphabetic characters to uppercase.
troff *options file*	Also **nroff**. Text formatting and typesetting utility based on directives embedded in *file*; e.g., this document was produced by **troff -me -eqn -tbl user.guide | lpr -t**.
vi *file*	Start "visual" text editor on *file*. Exit without saving changes by entering **:q!**
wc *file*	Word count (lines, words, characters) of *file*

Software Development Commands

^C	(control C) Kill program, the current foreground process; can not restart
^Z	(control Z) Suspend program which may be restarted in foreground (**fg**) or background (**bg**)
a.out	Run executable file "a.out", the default name for compiled programs. **ar** *options library*!T{ Archive and library maintainer; e.g., **ar t lib.a** produces a table of object files in **lib.a**.
at *time executable*	Run a program at a given time
bg *n*	Restart in background job number *n* (as displayed by **jobs**)
cc *sourcefile***.c**	Compile C source program; the suffix (**.c**) is UNIX convention
dbx	Interactive debugger
f77 *sourcefile***.f**	Compile Fortran source program; the suffix (**.f**) is UNIX convention **jobs**!List active jobs **kill** *pid*!Kill process ID *pid* (as displayed by **ps**) **make**!T{ Maintain program groups: A formalism for describing dependencies when creating executables from multiple files.
ncargf77 *sourcefile***.f**	Compile Fortran source program and link to the NCAR Graphics library.
nice *executable*	Run a program at low priority
od -*options file*	Display octal, decimal, hexadecimal, or ascii representations of **file**; e.g., **od -c file**, displays bytes as ascii characters (or octal numbers when the parity bit is set).
pc *sourcefile***.p**	Compile Pascal program; the suffixed (**.p**) is UNIX convention
ps	Display process status including process ID numbers

System Status and Miscellaneous Commands

cal 12 1994	Display calendar for December 1994
clear	Clear screen
dc	Start desk calculator; quit by entering **q**
du *directory*	Recursively display disk usage in kilobytes for files in *directory* and any below it. When *directory* is omitted, the current directory is used.
df	Disk free: Display table of free disk space by partition
date	Display date and time
env	Display current environment settings
finger *user@address*	List information about *user@address*
mailx	Check for, read, and respond to e-mail messages. Help is available after starting **mail** by entering **?**.
mailx *login@address*	Compose and send an e-mail message
man *command*	Display manual page for *command*
man -k *keyword*	Show one line summaries of UNIX commands related to *keyword*.
nispasswd	Change password
stty *options*	Display or set terminal attributes; e.g., **stty -a** displays all current settings.
talk *login@address*	Live, on-screen typing to other user
tset *options*	Utility to define terminal capabilities via setting environment variables TERM and TERMCAP.
w	List who is logged in and what they are doing; also display the current system load
which *command*	Show the complete path name for *command*

A.4 Remote File Transfer

It is likely that files will need to be transferred to your home computer in the course of the analysis. While cmenu automates this task, there will be situations where an independent mechanism is required. This section reviews standard UNIX commands for that purpose: **ftp** and **rcp**. A complete reference for each command may be obtained via **man** pages. These commands have components executing on both computers to coordinate the file transfer; i.e., your home computer must also support these commands. ftp is more generally available but rcp is more convenient to use once permissions have been established; therefore, it is recommended that you go to the trouble to set up and use rcp when there will be many occasions to transfer files.

Bear in mind the restriction that these commands must be invoked from cedar (instead of the remote host) as a consequence of the "gatekeeper" security system. And that assumes the external computer honors the PASV (passive) mode; if it doesn't, then it will be necessary to use the gatekeeper as a proxy. This procedure is explained in URL **http://www.hao.ucar.edu/public/home/gatekeeper/** or in a PostScript file (**/home/sitongia/adm/Projects/security/HAO-model.ps**) or PDF file (**/home/sitongia/adm/Projects/security/HAO-model.pdf**).

ftp is sensitive to the transfer type (binary or character); it must be correctly specified to avoid incorrect translation. It can be determined using the command: **file**. Examples of binary files common in the database are data files (with names ending in **.cbf**) or compressed files (**.Z**); examples of character files are source code (**.f**) or character data files (**.chr** or **.fld**).

When on cedar, an interactive session may be started by entering

ftp *myhost*

where *myhost* is the recognized name or IP address of the remote computer. After responding to prompts for login and password, if the the file to be copied is binary, change from the default (**ascii**) by entering:

binary

Change to directory *dir* on *myhost*, initiate the transfer, and end the session by entering:

cd *dir*
put *cedar.file new.file.name*
quit

If *new.file.name* is not provided, the new copy will be put in directory *dir* with the name *cedar.file*. *dir* must have appropriate write permission (see **man** pages for **ls** and **chmod**). Multiple files of the same type (**binary** or **ascii**), may be transferred via:

> **prompt**
> **put** *cedar*files*
> **quit**

where *cedar*files* is a blank separated list of files or a string with file expansion characters (e.g., ***.?**, see **man csh**) matching the files to be transferred. It may be important to set the representation type correctly; the default (**ascii**) involves a translation needed when transferring to some operating systems.

Remote copy (**rcp**) requires permissions previously established on the remote computer. For example, to copy from **cedar** to *myhost*, your home directory on *myhost* must have a file named **.rhosts** with lines

> **cedar.hao.ucar.edu** *login*
> **128.117.16.90** *login*

where *login* is your login name on **cedar**. The second line is necessary when the host names is not recognized but the IP address is recognized.

If your *login* is the same on both computers, then *cedar.file* may be copied to *myhost* and put in directory *~/dir* be entering

> **rcp** *cedar.file myhost***:***dir*

If your logins differ use the form

> **rcp** *cedar.file user@myhost***:***dir*

where *user* is your login on *myhost*. If *myhost* is not recognized by name, substitute the IP address (your system administrator knows this, or it may be determined using **nslookup** *myhost*). rcp does no conversion or translation during transfer, so binary and character files are copied without regard to content. Argument list variations allow renaming the file (by adding a file name suffix after *dir*/) or copying multiple files to a directory by providing more than one *cedar.file* argument.

A security policy at NCAR started September 1998 forces incoming internet traffic through a “gate-keeper” computer. This is bypassed when ftp and rcp are initiated from cedar, so these commands should behave as described above. This also assumes ftp can be run in PASV (passive) mode. If this is not true or it is necessary to initiate the transfer from outside the firewall, then the connection must go through the gate-keeper; refer to the description of this special procedure given on the web at URL

> **http://www.hao.ucar.edu/public/home/gatekeeper/**).

It may be important to know the size of the file to be transferred. (Space must be reserved on the remote machine and transfer times are proportional to file size.) On cedar, one may use

> **ls -l** *file*

to determine the size of *file*. Transfer times should be acceptably short for files on the order of 10’s MB. When transferring larger files it may be important to avoid peak internet traffic times.

A.5 Visual editor

vi is described here because it is commonly available on UNIX machines. **vi** is a full screen embellishment of the line oriented editor **ex**. When editing, it is possible to switch between the command line mode of **ex** to the screen oriented **vi** commands. The editor works on a copy of the file held in memory, so changes are not saved until directed. **vi** will misbehave if terminal capabilities are not properly defined via the TERM environment variable. Usually this is established automatically via the **.login** file; see Login Procedures above and **man tset**.

ENTERING vi

vi *file*	begin editing *file*; create it if necessary
vi + file	begin editing at last line of *file*
vi +*n file*	begin editing *file* at line *n*
vi +/*string file*	begin editing *file* at *string*
vi -r *file*	recover edits after system crash
vi -R *file*	read only mode for displaying *file*

LEAVING vi

ZZ	exit from vi, saving changes; **:x** is synonymous
:e *file*	edit *file*
:e #	edit alternate file (when two files are toggled)
:n	edit next file in argument list; e.g., after starting with **vi ***
:n *files*	specify a new argument list
:q!	exit from vi, discarding changes.
:w	Write file to disk (make changes permanent)
:w *name*	Write file *name*
:w! *name*	Overwrite file *name*
:wq	Write changes and quit
:*x, y***w** *file*	Write lines *x* to *y* to *file*
:ta *tag*	edit file containing tag *tag* at *tag*

CURSOR POSITIONING

h	move left one character
j	move down one line.
k	move up one line.
l	move right one character
w	move right one word
b	move left one word
e	move to end of word
0	(zero) move to the beginning of line
$	move to end of line
f*x*	move right in line to character *x*
F*x*	move left in line to character *x*
t*x*	move right in line positioning to left of character *x*
T*x*	move left in line positioning to right of character *x*
;	repeat last **f**, **F**, **t** or **T**
,	inverse of last **f**, **F**, **t** or **T**
^u	(control u) move up 1/2 screen
^d	move down 1/2 screen
^b	move up (back) one screen
^f	move down (forward) one screen
G	move to last line
*n***G**	move to line number *n*
^G	display file name and current line number
^l	clear screen and re-draw
}	move down one paragraph

{	move up one paragraph
H	go to home position of screen
M	go to middle of screen
L	go to last line of screen
zCR	current line to top of screen (**CR** = return)
z.	current line to middle of screen
z-	current line to bottom of screen
/*string*	search forward for *string*
?*string*	search backward for *string*
n	repeat last string search
m*x*	save the current cursor position and mark it *x*
'*x*	return to beginning of line marked *x*
`*x*	return to cursor position marked *x*
''	return to previous position (counteracting **G** or simple search)
%	move to matching (,), [,], {, or }

TEXT MODIFICATION

i*text***ESC**	insert *text* to the left of the cursor; terminate with **ESC** (escape)
I*text***ESC**	insert *text* at beginning of line
a*text***ESC**	append *text* to right of the cursor
A*text***ESC**	append *text* to end of line
o*text***ESC**	open line below current and add *text*
O*text***ESC**	open line above current and add *text*
r	replace the current character
R*text***ESC**	replace characters at the cursor position
:*x***,***y***s/***old***/***new***/***flags*	For lines *x* to *y*, substitute string *new* for *old*; optional *flags* are **c** (confirm before substituting) and **g** (global - replace all occurrences in every line instead of only the first); substitution is on current line only when *x***,***y* is omitted; **1,$** means the first to the last lines which is synonymous to **%** as in **:%s/***old***/***new***/**
cw*text***ESC**	change word, replacing it with *text*
C*text***ESC**	change to end of line, replacing it with *text*
cc*text***ESC**	change this line
*n***cc***text*ESC	change *n* lines
x	delete the character at the cursor (into unnamed buffer); same as **d***space*
*n*x	delete *n* characters including and to the right of the cursor
dw	delete the next word
D	delete characters from the cursor position to the end of the line
dd	delete the current line
*n***dd**	delete *n* lines
d'*x*	delete current line to line marked *x*
*x***,***y***d**	delete lines *x* to *y*
*bn***dd**	delete *n* lines and place them in buffer named *b*; 26 definable buffer names are **a**-**z**
y	yank the current character (save a copy in unnamed buffer)
*n***y**	yank *n* characters after the cursor
yw	yank word
*n***yw**	yank *n* words after the cursor
yy	yank current line; same as **Y**
*n***yy**	yank *n* lines; same as *n***Y**

y'*x*	yank current line to line marked *x*
x,*y***y**	yank lines *x* to *y*
*bn***Y**	yank *n* lines and place them in buffer named *b*
p	put deleted or yanked characters or words after cursor on current line; or put deleted lines below current line
"*b***p**	place text from buffer *b* below current line; *b* may be named (**a**-**z**) or numbered (**1**-**9**). Numbered buffers are automatically loaded with the last nine deletions, so **"1** is the same as the "unnamed" buffer - try **"1pu.u.u.**
P	put text before cursor position or current line
"*b***P**	place text from buffer *b* above current line
u	undo the last change
U	undo last changes in current line only
J	join current line with the next (split is **rCR**)
*n***J**	join *n* lines
.	(period) repeat last change
~	change case of current character
!!*cmd*	execute UNIX *cmd* and put output after current line

MISCELLANEOUS

:g/*string***/***cmd*	Command *cmd* only affects lines containing *string*
:v/*string***/***cmd*	Command *cmd* affects lines not containing *string*
:!*cmd*	escape to shell, execute *cmd*, and return on prompt
:set allCR	display current settings (**CR** = return)
:map *lhs rhs***CR**	map *lhs* into *rhs* (redefine key sequences)

In addition to being established during an edit session, certain **ex** features may be initialized by prior definition of the EXINIT environment variable or in the file **~/.exrc**. Some options are

Name	Default	Description
autoindent	**noai**	supply indentation automatically
autowrite	**noaw**	automatic write before **:n**, **:ta**, **control-^**,**!**
ignorecase	**noic**	ignore case when searching
list	**nolist**	tabs print as ^I; end of lines marked with $
magic	**nomagic**	the characters **.** **[** and ***** are special in searches
number	**nonu**	lines are displayed prefixed with numbers
shiftwidth	**sw=8**	shift distance for **<**, **>** and input **^D** and **^T**
showmatch	**nosm**	show matching **(** or **{** as **)** or **}** is typed
slowopen	**slow**	postpone display updates during inserts
term	**dumb**	the kind of terminal you are using
wrapmargin	**wm=0**	automatically break lines (insert newlines) when a space is typed no further than **wm** characters from right edge of screen

Numeric and string options are defined using the format

set *opt=val*

Toggle options are switched by:

set *opt*
set no*opt*

The **map** feature may also be initialized as in

setenv EXINIT 'set ai aw wm=10|map @ dd|map # x'

where autoindent and autowrite are toggled on, wrapmargin is 10, @ is defined to delete the current line, and **#** is defined to delete the current character.

The above command summary is necessarily incomplete: There is no substitute for reading the manual. The **vi man** page is only a brief introduction. More complete on line documentation is available if using a windows environment on cedar: **AnswerBook** is invoked by entering

answerbook

A.6 References

Anderson G. and P. Anderson, *The UNIX C Shell Field Guide*, Prentice Hall Englewood Cliffs NJ, 1986.

Sonnenshein, D., *A Guide to VI Visual Editing on the UNIX System*, Prentice Hall Englewood Cliffs NJ, 1987.

various authors, *UNIX User's Reference Manual (URM) 4.3 Berkeley Software Distribution, Virtural VAX-11 Version*, Computer Systems Research Group, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California Berkeley CA, April 1986 (third printing June 1988).

various authors, *UNIX User's Supplementary Documents (USD) 4.3 Berkeley Software Distribution, Virtural VAX-11 Version*, Computer Systems Research Group, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California Berkeley CA, April 1986 (second printing June 1987).

APPENDIX B

LOGICAL DATA RECORD FORMAT

The “CEDAR Database Format” contains a complete description of the binary and character versions of the three types of logical records: catalogue, header and data. The logical data record table is repeated here for convenient reference:

FORMAT OF DATA RECORDS

Binary Word or Character Field	Name	Definition	Comments
Prologue			
1	LTOT	Length of this record.	Binary version: Length is the number of 16-bit words including this one, or LTOT = LPROL + 2*JPAR + MPAR*(NROW+1) Character version: Length is the number of 120 byte lines including this one, or LTOT = 1 + ((JPAR+19)/20)*2 + ((MPAR+19)/20)*(NROW+1)
2	KREC	Kind of record.	Binary version: Value is 1002 Character version: Value is 1101
3	KINST	Instrument code for these data	See Table 3
4	KINDAT	Kind-of-data code (Same value as for associated data records)	Each code indicates a particular data analysis algorithm. KINDAT must differ for each different type of interleaved records for the same instrument.
5	IBYR	Beginning year	
6	IBDT	" date	
7	IBHM	" hour & min	
8	IBCS	" centisecond	For data covered by this record.
9	IEYR	Ending year	Format in Table 4.
10	IEDT	" date	Universal Time is used.
11	IEHM	" hour & min	
12	IECS	" centisecond	
13	LPROL	Length of this prologue	Number of words or fields. Must be at least 16.
14	JPAR	Number of single-valued parameters	2*JPAR is length of 1-D data array. Zero is permissible.
15	MPAR	Number of multiple-valued parameters	Number of columns in 2-D data array. Zero is permissible.
16	NROW	Number of entries for each multiple-valued parameter	(NROW+1)*MPAR is length of 2-D data array.

Binary Word Number	Name	Definition	Comments
1-D data array:			
LPROL+1 · · · LPROL+JPAR	KODS(1) · · · KODS(JPAR)	Code for first single-valued parameter Code for last single-valued parameter	Codes from Table 8 Character version starts the 1-D array on a new line using a 20I6 format, or up to 20 codes per line
LPROL+JPAR+1 · · · LPROL+2*JPAR	IPARS(1) · · · IPARS(JPAR)	Value of first single-valued parameter · · Value of last single-valued parameter	Missing values are entered as -32767. Character version starts 1-D array values on a new line using 20I6 format, or up to 20 values per line
2-D data array:			
LPROL+2*JPAR+1 · · · LPROL+2*JPAR +MPAR	KODM(1) · · · KODM(MPAR)	Code for first multiple-valued parameter · · Code for last multiple-valued parameter	Codes from Table 8 Character version starts the 2-D array on a new line using a 20I6 format, or up to 20 codes per line.
LPROL+2*JPAR +MPAR+1 · · ·	IPARM(1,1) · · ·	First value of first multiple-valued parameter · ·	Character version starts each row of array values on a new line using 20I6 format, or up to 20 values per line.
LPROL+2*JPAR +2*MPAR · · ·	IPARM(1,MPAR) · · ·	First value of last multiple-valued parameter · ·	Data are stored row by row, rather than column by column
LPROL+2*JPAR +NROW*MPAR+1 · · ·	IPARM(NROW,1) · · ·	Last value of first multiple-valued parameter · ·	
LPROL+2*JPAR +(NROW+1)*MPAR	IPARM(NROW,MPAR)	Last value of last multiple-valued parameter	