Appendix C

Details of AIPS files

C.1 Introduction

This appendix contains the detailed descriptions of the various files used in AIPS both for user data and AIPS system functions. The details of the structure, contents and usage are given for each of these file types. These files are organized into 4 categories in this appendix; 1) AIPS system files, 2) user data files, 3) user tables and 4) task specific tables.

C.2 AIPS System files

In this section the files that the AIPS system uses are described.

C.2.1 Accounting (AC) file

Overview

The account file is used to accumulate information on the use of the AIPS system.

Details: All user programs in the AIPS system are required to place information on task name, user number, start date/time, elapsed time, and CPU time for each execution in the accounting file. This information is used by AIPS System Managers for a variety of purposes including the identification of problem programs and the maintaining of financial and usage accounting.

Name: The file name is ACr00000 where r is the system format revision number. It is a permanent file and must be created at AIPS system installation. A good initial size is about 100 blocks, but the file will expand itself one granule at a time as necessary.

File Structure

The file consists of logical records each 9 words long. There are 28 logical records in each 256-word physical record. The first logical record contains

```
FIELD    TYPE            DESCRIPTION

  1       I         Maximum logical record # now recorded
  2       I         Max logical record # which will fit in file
  3       I         number of words / logical record
  4       I         Number of logical records / physical record
  5       I(5)      Reserved
```

where fields 3 and 4 are clearly just for convenience. All other logical records have the structure:

FIELD	TYPE	DESCRIPTION
1	H(2)	Task root name (4 characters/word)
2	I	POPS number + 100*i (i=1,2,3 for OLD,NEW,TST)
3	I	Logon user number
4	I(2)	Start time (packed YY/MM/DD, HH/MM/SS)
5	I	Total I/O count (when possible)
6	R	Total real time in seconds
7	R	Total CPU time in seconds

When a program initializes its entry in the file, field 6 is set to zero and field 7 is set to the current CPU time. When a program closes its entry, field 6 is reset to the difference in the current time and that in field 4 and field 7 is reset to the difference between the current CPU time and that previously in field 7. Thus, system overhead times are not fully charged to the program and programs which abort are identifiable by field 6 equal to zero. I/O count is handled like CPU time.

Usage Notes

No normal program other than those listed below should access this file. Of course, AIPS System Managers may need to construct additional, special account management programs.

Routines and commons for writing the account task data file

The primary routine for accessing the file is ACOUNT (IOP, buffer), where IOP = 2 means the entry is being closed and all other values of IOP mean the entry is being opened. The AIPS, AIPSC, AIPSB, and BATER programs call ACOUNT directly at appropriate times. Tasks must call GTPARM to obtain their full identity and their adverb values. Therefore, as soon as the identity is known, GTPARM calls ACOUNT. When tasks end, they must issue a close message, resume the initiating AIPSxx, and close the account entry. The subroutine DIETSK (RETCOD, RQUICK, buffer) has been constructed to perform all of these operations. The last executable statement in every task must be a call to DIETSK. Subroutines DIE and TSKEND perform this call for some tasks.

The common /MSGCOM/ (DMSG.INC) now carries a parameter, NACOUN, which gives the position in the accounting file being used by the current program. No routine, but ACOUNT, should change this parameter.

Routines and commons for reading the task data file

A service program, PRTAC, has been written for use by AIPS System Managers and other interested users. It can be run either as an AIPS task or as a stand-alone program. It prints the contents of the account file in the forms: (1) sequential listing, (2) totals by POPS number, (3) totals by user number, and (4) totals by program name (sorted by CPU and sorted by number of occurrences). PRTAC accumulates tasks and AIPS-like programs separately for types (2) and (3). The user may limit which of these displays he receives. He may also limit the summing and displays to a specific program name, a range of POPS numbers, a range of user numbers, only programs using more than x seconds of CPU, and/or only programs starting after a specific date. The displays may be done separately for each AIPS version, or the versions may be lumped together into a single display. PRTAC has a separate operation code which will clear the file for AIPS Managers only.

C.2.2 Batch text (BA) file

Overview

Batch text files contain a list of 80-character lines to be used as input to the batch versions of AIPS. There are logically two types of batch text files: the work files used by AIPS and BATER to prepare jobs for submission and the files actually queued to AIPSBn.

Names: The former have names BAr00n0m where r is the AIPS format revision code, n is the queue number (1 <= n <= NBATQS) and m = 1 through NINTRN for the interactive AIPSs and m = NINTRN + 1 for BATER. These are permanent files which can grow as needed. The latter have names BAr0nn0m where nn is the lower two digits of the job number (01 <= nn <= 64) and m is the value of NPOPS used by the AIPSBm program (i.e. m = BATQUE + NINTRN + 1). These files are created by the checker version of AIPS and are destroyed by AIPSBm when the batch job terminates.

Data structures

The physical records in these files are each 256 words. The first four words contain special information (which is only used in record 1) and the rest of the record contains logical records. There are 11 logical records in each physical record. The two kinds of files use identical data and pointer structures.

The first four words of the first physical record contain:

```
FIELD    TYPE       DESCRIPTION

  1        I    User number given in the logon
  2        I    Next available logical record number
  3        I    Logical record number of last line in file
  4        I     Number of 256-word records now in file
```

The first four words of all other physical records are unused. Each logical record has the structure:

```
FIELD    TYPE       DESCRIPTION

   1      I     Logical record number of next line (0 if none)
   2      I     Logical record number of previous line (0 if none)
   3      H(20) Text of line in HOLLERITH form.
```

Usage Notes

The first line in the file is regarded as line 0 and is always blank and always located at logical record number 1. Its sole purpose is to point to the first real line of text. If this convention were not adopted, then it would not be possible to insert text in front of the existing text in a work file nor to delete the first line in the work file. The linked list structure of these files is not necessary for the files processed by AIPSBn. However, it is maintained in order to retain a single, somewhat simpler reading program. In particular, it is used to make BATLIST and JOBLIST essentially identical.

Routines and commons for reading batch text files

The subroutine PREAD performs all standard reads for POPS language processors. If the variable IUNIT in common /IO/ is set to 3, then the read is done from a batch text file. The operation is controlled by the common /BATCH/ as :

```
BATLUN   I      logical unit number of batch text file
BATIND   I      pointer to FTAB for the open file
BATREC   I      logical record number of the last line read
                  (<= 0 if none)
BATDAT   I(256) contents of last physical record read
```

The program which uses PREAD for reading batch files must open the file setting appropriate values in BATLUN, BATIND, and BATREC. If BATREC is set > 0, PREAD will assume that the contents of BATDAT are valid and contain the logical record referenced by BATREC. PREAD will use the pointer in the current BATREC to update BATREC and to obtain the next line, performing a read operation only when required. If the pointer in the current logical record (BATREC) indicates that there are no more lines, then, effectively, an error (end of file) condition has arisen. PREAD signals this by returning a text line of 'EXIT' or, for UNQUE in AIPS and BATER, 'ENDBATCH'.

The application subroutines AUB and CUB also read batch text files. The verb UNQUE uses the common /BATCH/ together with PREAD to read the text file being unqueued. The verbs BATEDIT, BATLIST, and JOBLIST use the common /BWTCH/ (see below) to read a file to locate the line to be editted and to locate the lines to be listed.

Routines and commons for writing batch text files

The subroutine PREAD can perform a write to a batch text file of the line which has just been read. It does this under control of the common /BWTCH/ as

```
BWTLUN   I       logical unit number of output text file
BWTIND   I       pointer to FTAB for the opened file
BWTREC   I       logical record number of the last line
                 added to the file (0 if none)
WASERR   L       .FALSE. => do the writing
BWTDAT   I(256)  contents of physical record containing
                 logical record BWTREC
```

Programs using PREAD and not wishing to write a batch text file must set WASERR = .TRUE.. As for reading, the text file must be opened and the common initialized by the program which uses PREAD. The program must also write the last record in BWTDAT to, and close, the file. It should also update the global pointers in record 1.

At present, the only program which uses PREAD for writing batch text files is AIPSCm. Because of certain problems with deletion and insertion as well as the need to check for the "magic" character strings 'RUN' and 'ENDBATCH', the batch preparation routines use the subroutine BBUILD to write into batch work files. BBUILD uses the common /BWTCH/ (except for WASERR) in a way which is similar to, but slightly messier than the way used by PREAD.

C.2.3 Batch queing (BQ) file

Overview

The batch queueing file is used to queue jobs to the batch versions of AIPS.

Name: The file is named BQr00000 where r is the system format revision code. It is a permanent file in area DA00 and must be large enough to hold NBATQS 256-word records.

Data structure

There is one 256-word record in the file for each batch processor: record n corresponding to BATQUE = n processed by AIPSB(m) (i.e., with NPOPS = m = n + NINTRN + 1).

Each 256-word block contains structures describing 64 jobs. These structures have the form:

```
FIELD    TYPE      DESCRIPTION

  1       I     User number given in the logon
  2       I     Submission time: 256 * 256 * (year-1900)
                                 + 256 * month + day
  3       I     Submission time: 256 * 256 * hours
                                 + 256 * minutes + seconds
  4       I     Desired version: 1-3 => OLD, NEW, TST resp.
```

In addition, the signs and contents of the first two words are used to convey the status of the job as:

```
word 1  = 0                    no job (vacant slot)
word 1  > 0  word 2  > 0       job waiting to run
```

```
word 1  > 0  word 2  < 0      job ran and failed
word 1  < 0  word 2  = 0      job is being submitted
word 1  < 0  word 2  > 0      job is running
word 1  < 0  word 2  < 0      job finished - may be reassigned
```

Usage Notes

Job numbers are formed as 100 * n + m , where m is the entry in queue n. It is simply the m'th 4-word block in record n. The queueing algorithm is first-in-first-out with a bias toward new users. Thus, the algorithm first looks for the oldest job submitted under any user number other than the previous three users of that queue. If it finds no jobs to run, it repeats the search excluding only the previous two user numbers. And so forth.

Routines and commons

The principal program to access this file is the subroutine BATQ which performs all normal operations on the file. It can "OPEN" a job by finding an available job number and setting the values in the slot to show that the job is being submitted. It takes the lowest numbered vacant slot or, if there is none, the lowest numbered job finished slot. BATQ can "RUNN" (queue) a job by converting the user number to a positive value and inserting the current time. It can "FIND" the next job to run using the algorithm mentioned above and setting the entry to job running. It can also "CLOS" a job by clearing the entry or, if it was running, by marking it finished. BATQ uses no commons and requires the calling program to maintain the list of previous users. BATQ does update the list on FIND, however. Other operations include "WHOO" to find the ID of the current job and "FAIL" to mark all currently running jobs as having failed.

The other subroutine which accesses this file directly is AUB (and the BATER version of AUB called CUB). These routines perform on the BQ file the user-oriented functions:

- QUEUES list job numbers and times in the queue
- JOBLIST list contents of submitted job
- UNQUE remove job from queue and transfer text of job back to a batch work file

C.2.4 GRIPE (GR) files

Overview

The gripe file contains gripes about the AIPS system; either bug reports or suggestions for improvements. versions of AIPS.

Name: The file is named GRr00000 where r is the system format revision code. It is a permanent file in area DA00.

Data structure

The first record of the GR file is a header record:

```
FIELD    TYPE      DESCRIPTION

  1       I     Current file size in blocks
  3       I     Current record number
  4       I     Current character position
```

The following records contain a series of arbitrary length hollerith entries. An entry is delimiter by the characters "{" and "}". Each gripe consists of 10 entries; these are:

1. Date and time in form {dd-mmm-yyyy hh:mm:ss}

2. system name {AIPS system name}
3. User number and release date {nnn 15JUL90}
4. User name {User Name}
5. Address { User address}
6. Phone number { User phone number}
7. Text of gripe { I can't get anything to work...}
8. One line summary { Everything is broken }
9. E-Mail address unused entry {bonkers@esu.edu }
10. unused entry { }

Usage Notes

Gripes are entered by users and may be accessed by users. Gripes are periodically copied to the AIPS gripes managment system.

Routines and commons

The AIPS GR file access is in routine AUC which create entries and give users access to the entered gripes. Gripes are removed from the GR file on a periodic basis and entered into the AIPS gripes managment system. This is currently done with the standalone utility routine GR2TEX. Routine AIPSUB:CHGRIP reads and writes entries in the GR file. CHGRIP accepts and returns character entries without the enclosing "{" and "}". AIPS verbs which allow user access to the GR file are:

- GRIPE enter a gripe.
- GRINDEX Index the gripes currently in the GR file.
- GRLIST Displays a sepected gripe.
- GRDROP Delete a specified gripe.

C.2.5 Help (HE) file

Overview

Help files are text files used to describe things to the user at run time.

Details: Help files are text files stored in a logical area called HLPFIL. A portion of the text in each help file is displayed on the user's terminal upon request.

Names: Each Help file is named with the name of the verb, task, adverb, pseudoverb, or procedure which it describes.

File Structure and contents

The Help files are text files prepared by the local text editor. Each Help file is a separate physical file in directory with logical name HLPFIL with the first part of the name being the name of the adverb, verb, task etc to which the file applies. The file name ends with ".HLP". E.g. the HELP file for help is HLPFIL:HELP.HLP in VMS notation or $HLPFIL/HELP.HLP in Unix notation.

Since help files are used only to provide run-time assistance to users, they may contain anything the programmer feels is relevant. Three general guidelines should be followed in order to obtain pleasing displays:

1. Text should occupy only card columns 1 - 64.

2. Text should be typed in lower case.
3. The file should not contain too many lines. 23 lines will fit on almost every terminal.

Help files are typed following a general pattern. To illustrate this pattern, consider the example:

```
; PRTMSG
;---------------------------------------------------------------
;! prints selected contents of the user's message file
;# Verb General
;-----------------------------------------------------------------------
;;  Copyright (C) 1995
;;  Associated Universities, Inc. Washington DC, USA.
;;
;;  This program is free software; you can redistribute it and/or
;;  modify it under the terms of the GNU General Public License as
;;  published by the Free Software Foundation; either version 2 of
;;  the License, or (at your option) any later version.
;;
;;  This program is distributed in the hope that it will be useful,
;;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;;  GNU General Public License for more details.
;;
;;  You should have received a copy of the GNU General Public
;;  License along with this program; if not, write to the Free
;;  Software Foundation, Inc., 675 Massachusetts Ave, Cambridge,
;;  MA 02139, USA.
;;
;;  Correspondence concerning AIPS should be addressed as follows:
;;         Internet email: aipsmail@nrao.edu.
;;         Postal address: AIPS Project Office
;;                         National Radio Astronomy Observatory
;;                         520 Edgemont Road
;;                         Charlottesville, VA 22903-2475 USA
;-----------------------------------------------------------------------
;---------------------------------------------------------------
PRTMSG    LLLLLLLLLLLLUUUUUUUUUUUU CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
PRTMSG:  Verb to print the message log file
PRIORITY           0.0       10.0 Print messages at or above
                                  this priority.
PRNUMBER          -1.0       15.0 AIPS number to be printed
                                     0 -> current, -1 -> all
PRTASK                            Restrict to task name(s)
PRTIME             0.0      999.9 Only messages younger than
                                  PRTIME (in days) printed
DOCRT             -1.0      132.0 > 0 => output on terminal
                                  > 72 => width of terminal
OUTPRINT
                                  Printer disk file to save
----------------------------------------------------------------
PRTMSG
Type: verb
Use:  Prints contents of message file including log of input
      commands and messages produced by AIPS and by tasks shed
```

```
      by AIPS.  This verb no longer deletes the messages.
      Please use CLRMSG to keep your message files small.
Adverbs:
  PRIORITY....Limit print to message having priority >= PRIORITY
              Message level 0 (user inputs) is also printed when
              PRIORITY <= 5.
  PRNUMBER....Selects the AIPS number of the messages to be
              printed.  0 => your current one, -1 => all.  See
              HELP PRNUMBER for some details.
  PRTASK......Selects the programs whose messages are to be
              printed.  Any program name whose first characters
              match the first non-blank characters of PRTASK
              will be selected.  Thus, PRTASK = 'UV' will cause
              the messages of UVMAP, UVSUB, UVCOP, etc. to be
              printed.  ' ' => all tasks by this rule.
              NOTE: messages from GO, INPUTS, and all other
              verbs are messages from AIPS not from any other
              associated task.
  PRTIME......Only messages younger than PRTIME days will be
              printed.   <= 0 => all times printed.
  DOCRT.......<= 0 -> print the messages on the line printer
              with times, priorities, etc. listed.  > 0 -> show
              the messages in slightly abbreviated form on Bthe
              terminal.  If you have a wide terminal, set DOCRT
              to its width in characters for a more complete
              display.
  OUTPRINT....Disk file name in which to save to line printer
              output (after printing).   ' ' => use scratch.
----------------------------------------------------------------
```

There are four parts to the help file.

1. Precursor lines to give the file name, a one line description, list of categories, and the user agreement statment.
2. The INPUTS section used by AIPS verbs INPUTS, GO, TPUT, etc. This has a fairly strict format. Adverb names begin in column 1, lower limits to numeric adverbs are free format in columns 11-22, upper limits to numeric adverbs are free format in columns 23-34, and a description of the adverb uses columns 36-64.
3. Separated from the INPUTS section by a row of 64 minus signs is the HELP section. This is displayed on the user terminal with verb HELP. The standard form is illustrated above.
4. Separated from the HELP section by another row of 64 minus signs is the EXPLAIN section. This provides more detailed user information, usually on the printer, with AIPS verb EXPLAIN.

The HELP file ends with a row of 64 minus signs. Some adverbs are used both for starting a task (GO) and for TELLing it things later. This is indicated by placing an astrisk ("*") in column 10 of the appropriate line of the inputs section. If an adverb is not used by GO but is used by TELL place a "?" in column 10. A line in the help file beginning with a semi-colon is a comment line, never displayed to the user.

Usage Notes

Help files are an important part of AIPS. They are the means by which users can find out how to use AIPS without leaving the terminal or leafing through thick manuals. Every task, verb, adverb, and pseudoverb must have a help file. Additional help files may also be created to explain special details. The existence of such files should be mentioned in the appropriate normal help file.

Routines and commons for reading Help files

Help files are opened and the member is located by the subroutine ZTOPEN. They are read sequentially by ZTREAD and closed by ZTCLOS. By convention, the logical unit number used for this is 11. These text file I/O routines are described elsewhere in this volume.

The subroutine which displays these files is called HELPS. The routine looks for special names which cause the symbol table to be examined and printed. Any other name is assumed to be a help file name and the file is opened (ZTOPEN), read (ZTREAD), printed (MSGWRT), and closed (ZTCLOS).

Routines and commons for writing Help files

Help files are prepared by programmers using their local source editors. No routines are to write such files at run time.

C.2.6 Image catalog (IC) file

Overview

The image catalog contains data for images stored on the TV device that identify the images, refer them back to their original map files, and specify scaling of the X-Y and intensity coordinates. There is a separate image catalog which performs the same functions for graphics devices (e.g. TEK4012 storage screens).

Name and Location: There is one image catalog file for each television device whose physical name corresponds to ICr0000n, where r = the system format revision number n = the device number (0 for graphics, 1 - n for TVs). They reside in area DA00 and must be created at AIPS installation.

Data Structures

General: For each grey-scale image plane of the TV device, the IC file contains N 1-block (256-word) records for cataloging up to N subimages, plus a (N-1)/51+1 block directory. The directory immediately precedes the catalog blocks for each image plane. For each TV graphics overlay plane there is one, undirectoried, catalog block. These blocks follow immediately after the last grey-scale block.

The IC for pure graphics devices (called TK devices) has one image catalog block for each device in the system including all "local" TK devices followed by all remote-entry devices. Record number n in this file is associated with TK device number n (NTKDEV in /DCHCOM/).

The image catalog blocks themselves are essentially duplicates of the map catalog blocks except that scaling information replaces the extension file index of the map catalog.

Record Formats:

Directory Block (Grey-scale image)

```
OFFSET LENGTH TYPE    DESCRIPTION
     0      2  I    Sequence number of last sub-image cataloged
                    on this plane
     2      2  I    Seq. no. of sub-image in slot 1; 0 if slot empty
     4      8  I(4) TV pixel positions of corners of 1st sub-image,
                    x1,y1,x2,y2
    12      2  I    Seq. no. of sub-image in slot 2; 0 if empty
    14      8  I(4) TV pixel positions of corners of 2nd sub-image
     .      .
     .      .
     .      .
```

Catalog Block for each image or subimage:

Most of the Image Catalog block is identical to the map catalog block of the source of the image. (See section on CB files.) Pointers for the Image catalog are found in the DHDR.INC common. The information on antenna pointing, alternate frequency/velocity axis descriptions, and extension files is replaced in the IC by:

```
OFFSET LENGTH TYPE  POINTER      DESCRIPTION

  472     8   R(2) IRRAN   Map values displayed as min & max
                           brightness (units are those of file,
                           not the physical ones)
  508     4   I    IIVOL   Disk volume from which map came
  512     4   I    IICNO   Catalog slot number of orig. map
  516    16   I(4) IIWIN   Map pixel positions of corners of
                           displayed image (rel. to orig. map)
  532    20   I(5) IIDEP   Depth of displayed image in 7 -
                           dimensional map (axes 3 - 7)
  552    16   I(4) IICOR   TV pixel positions of corners of
                           image on screen
  568     4   I    IITRA   2-char code for transfer function
                           used to compute TV brightness from
                           map intensity values.
  572     4   I    IIPLT   Code for type of plot.
  576   100   I(25)IIOTH   Misc. plot type dependent info.
                              (at the moment no more than 20 used)
```

Usage Notes

We assume that single images only are stored on graphics planes; they are not directoried.

When a grey-image plane is cleared, its directory is zeroed. As images are added to the plane, their coordinates are written into an open directory slot for that plane, along with the current value of the plane sequence number. The sequence number is then incremented. If an old image is completely overwritten by a new one, its directory slot is cleared. For partially overlapping images, the sequence number allows the user to select the one most recently loaded into a given part of the plane.

Subroutines:

- YCINIT clears the Image Catalog for a given plane
- YCOVER are there any overlapped images in each on quadrant?
- YCWRIT adds a new block to the catalog
- YCREAD returns the block corresponding to a given TV pixel
- YFIND determines desired image, asks user if > 1 visible

These routines expect the "plane number" as an argument. TV gray scale planes are numbered 1 - NGRAY, TV graphics overlay planes are numbered NGRAY+1 - NGRAY+NGRAPH, and TK devices are referenced by any plane number > NGRAY+NGRAPH.

Commons:

The COMMON /TVCHAR/ referenced by 'DTVC.INC' contains TV device characteristics such as:

```
NGRAY  =  # of grey-scale planes on this device
NGRAPH =  # of graphics planes
MAXXTV(2) Maximum number of pixels in x,y directions in image
```

The common /DCHCOM/ (DDCH.INC) contains two important parameters in this regard: NTVDEV and NTKDEV. The subroutine ZDCHIN sets these to the actual number of such devices present locally. Then, the routines ZWHOMI (in AIPS only) and GTPARM (in all tasks) reset them to the device number assigned to the current user. ZWHOMI determines these assignments.

C.2.7 TV lock (ID) file

Overview

This file is to establish current ownership of a TV display device. To becode the current "owner" of the device, a process opens the ID file with exclusive use. This prevents other users from attempting simultaneous access to the display.

Name and Location: There is one ID file for each television device whose physical name corresponds to IDr0000n, where r is the system format revision number, and n = the device number (0 for graphics, 1 - n for TVs). They reside in area DA00 and must be created at AIPS installation.

Data Structures

There is no meaningful data in the ID file.

Usage Notes

The ID file is opened by YTVOPN using ZOPEN and closed by YTVCLS using ZCLOSE to extablish exclusive use of the display device.

C.2.8 POPS memory (ME) file

Overview

POPS memory files are used to store copies of the POPS "environment" (procedure source code, symbol table, procedure executable code, adverbs values, various pointers).

Details: The files are permanent files and have a size which depends on the size of the LISTF and K arrays in the version of POPS being used. The formula is 5 times the size of LISTF plus 4 times the size of K.

Names: POPS memory files are named MEr0000n , where r is the system format revision number and n is NPOPS (the POPS identification number).

File Structure

The structure of the K and LISTF arrays is discussed elsewhere. Here we will show only how they are stored on disk. The files consist of sequential 256-word records. Using the sizes of the K and LISTF arrays in the 15APR90 version of AIPS, the files are laid out as

```
FIELD   TYPE          DESCRIPTION

  1     I(4096)    LISTF in use by the current user
  2     I(18944)   K array for initial values and RESTART
  3     I(4096)    LISTF array for initial values and RESTART
  4     I(18944)   K array for STORE 1 and RESTORE 1
  5     I(4096)    LISTF array for STORE 1 and RESTORE 1
  6     I(18944)   K array for STORE 2 and RESTORE 2
  7     I(4096)    LISTF array for STORE 2 and RESTORE 2
  8     I(18944)   K array for STORE 3 and RESTORE 3
  9     I(4096)    LISTF array for STORE 3 and RESTORE 3
```

User notes

It should not be necessary for users to read or write this file under normal circumstances. Standard POPS subroutines do all the operations which are required.

Routines which read and write POPS memory files

The memory files are given their initial values by the program POPSGN. This program does a special compilation followed by an AIPS-like compilation on text files called POPSDAT for interactive and batch AIPSs. POPSGN sets all areas of the memory file to the "virgin" K and LISTF arrays which it has computed.

In the POPS programs, the subroutine INIT moves field 3 ("virgin" LISTF) into field 1 (working LISTF) and moves field 2 into the core copy of the K array. The subroutine STORES updates the working copy of LISTF (field 1 on disk) and performs the pseudoverbs LIST, STORE, RESTORE, SAVE, GET, CORE, and SCRATCH. The variables in the common /POPS/ which aid this process are

```
LPAGE       number of 256-word blocks in LISTF (= 16)
MPAGE       number of 256-word blocks in K plus LISTF (= 90)
```

C.2.9 Message (MS) file

Overview

Message log files are used to record the input to, and messages from, all programs.

Details: Message files are user-owned files which record all messages except those with no time-variable information content (e.g. HELPs, interactive instructions) and those explicitly directed to the line printer (e.g. verbs PRTMSG and PRTHI, tasks PRTIM and PRTPL). All programs at all POPS numbers use the message file of the logon user.

Names: The message file is called MSfuuu00.uuu, where uuu is the user number in hexadecimal notation and f is the data format version code letter.

File Structure

Message files are sequential files consisting of 256-word records. Each of these physical records begins with two words of special information (which are used only in record 1). The remainder of each physical record is devoted to 10 logical records.

The special information in word 1 of the first record is simply the current number of messages recorded in the file and the number of physical records currently in the file. Each logical record has the structure

```
FIELD    TYPE          DESCRIPTION

  1      I       message priority (0 - 10) + 16 * POPS number
                 of task
  2      I       message date: YY/MM/DD packed
  3      I       message time: HH/MM/SS packed
  4      H(2)    name of task generating message (5 HOLLERITH
                 characters)
  5      H(20)   80-character message (HOLLERITH characters)
```

User notes

All programs should use this file for message display. They do this with the subroutine MSGWRT described below.

Routines and commons for writing message files

The subroutine MSGWRT is the only routine which should be used to write on these files. It is a somewhat complicated routine and very fundamental to the AIPS system. It uses the commons in include DMSG.INC as :

```
MSGCNT   I      number of messages in file (< 0 implies don't
                know and file is closed)
MSGREC   I      number of 256-word records in file at present
TSKNAM   C*6    name of current task
NPOPS    I      POPS identification number
NLUSER   I      user number from logon
MSGSUP   I      if = 32000, suppress level 6 & 7 messages
ISBTCH   I      if = 32000, treat this job as a batch job
NACOUN   I      position in the accounting file this job
MSGTXT   C*80   80-character message to be logged in file and/or
                displayed on terminal
```

MSGWRT opens the message file and initializes MSGCNT, puts the message into the file (unless inhibited), displays the message on the terminal (LUN=6, unless inhibited), and closes the file updating the recorded message count and resetting MSGCNT. MSGWRT will expand the file with no limit as needed. It will complain to interactive users when the number of messages is greater than 750.

Routines and commons for reading message files

The subroutine PRTMSG is used to print, and compress message files. It takes 7 input and 3 output arguments. The first is an opcode of 'PRIN' for print, 'DELE' for compress, or anything else for a typed summary. The next 6 arguments define the user number, the POPS number (0 => all), the priority lower limit for printing, the task name, a time cutoff for printing or deleting, and a logical to request the printing to be on the CRT terminal. The outputs are the number of messages printed/deleted, the number left in the file, and an error code. See the precursor remarks for the details. The physical file is compressed after messages are deleted.

C.2.10 Password (PW) file

Overview

The PW file contains AIPS user passwords. These are AIPS internal passwords and are not related to system passwords.

Details: AIPS user passwords are up to 12 characters kept as 3 hollerith variables. The first of these words in the (256 integer) record given by:

```
RECNO = (user_number - 1) / (256/3)) + 1
```

in word number:

```
WORD = MOD ((user_number - 1), (256/3)) * 3 + 1
```

Names: The password file is called PWf00000., where f is the system format version code letter.

File Structure

Password files are sequential files consisting of 256-word records. User passwords are stored in blocks of 3 hollerith variables (all in a given record) as described above.

Useage notes

All access to the PW file is through ZFIO. AIPS usage of the password file is limited to routine AUC. Several standalone programs also use the password file.

C.2.11 POPS Save-Get (SG) file

Overview

POPS memory files are used to store copies of the POPS "environment" (procedure source code, symbol table, procedure executable code, adverbs values, various pointers). Save/get files are "user-owned", rather than the "public" ones described in the ME portion of this appendix.

Details: The files are semi-permanent files and have a size which depends on the size of the LISTF and K arrays in the version of POPS being used. The file must be larger than one K plus one LISTF array. The files are at least 91 256-word blocks long. Each user having one or more SG files also has an SG directory file. On ordinary computers this file only requires 8 blocks.

Names: POPS save/get files are named SGfuuujj.uuu , where uuu is the user logon identification number in hex and jj is the sequence number in hex of the file in the SG directory. Sequence number 0 is reserved for the directory file. The data format revision letter (A, B, ...) is f.

File Structure

The SG directory file consists of 256 logical records, numbered 0 through 255. Each logical record consists of 7 words. There are 256 / 7 logical records per physical record (36). The first logical record (called no. 0) is reserved for control information and currently includes:

```
FIELD   TYPE          DESCRIPTION

  1     I         Maximum version number now in use (called max)
  2     I         Number of vacant entries with version # < max
  3     I(2)      Last GET time (packed YY/MM/DD, HH/MM/SS)
  4     I(2)      Last SAVE time (packed YY/MM/DD, HH/MM/SS)
  5     I         Reserved
```

Logical records 1 to 255 (correspond to file version numbers 1 - 255) contain

```
FIELD   TYPE          DESCRIPTION

  1     I         Number characters in user-supplied name ( <= 0
                  means empty) + 32 * SG format version number
  2     I(2)      Last SAVE time (packed YY/MM/DD, HH/MM/SS)
  3     H(4)      User supplied name, blank filled, 16 char.
```

The SG data files are described below:

The structure of the K and LISTF arrays is discussed elsewhere. Here we will show only how they are stored on disk. The files consist of sequential 256-word records. Using the sizes of the K and LISTF arrays in the Charlottesville version of AIPS, the files are laid out as

```
FIELD   TYPE          DESCRIPTION

  1     I(256)    Header record (see below)
  2     I(18944)  K array
  3     I(4096)   LISTF array
```

The header record at the present contains only 6 useful words:

```
FIELD    TYPE          DESCRIPTION

  1       I(3)     Date: year since 0, month, day
  2       I(3)     Time: hour, min, sec of the last SAVE op
```

User notes

It should not be necessary for users to read or write these files under normal circumstances. Standard POPS subroutines do all the operations which are required. They are created at run time by the user as required. They may be deleted by the user at run time or by the AIPS manager using FILINI or other utilities.

Routines which read and write POPS memory files

The subroutine STORES creates, writes, and reads the save/get files. The subroutine AU3A reports on disk usage including such files and can destroy all such files belonging to the logon user. The variables in the common /POPS/ which aid in this process are

```
LPAGE       number of 256-word blocks in LISTF (=16)
MPAGE       number of 256-word blocks in K plus LISTF (=90)
```

The subroutine SGLOCA, called by STORES, performs normal operations on SG directory files including creation. The verb SGDESTR performed by the subroutine AU2A handles destruction of individual SG data files and will destroy an empty directory. AU2A also performs the directory listing function (verb SGINDEX). Subroutine AU3A will destroy all the user's SG files at once (verb SAVDEST).

C.2.12 System parameter (SP) file

Overview

The system parameter file provides programs with information about the system resources that are available to AIPS.

Details: All AIPS programs read this file as one of the first steps of program execution. This is a permanent file created at AIPS installation time either with FILINI or FILAIP or as part of an automatic installation procedure that is available on a limited number of computers. The size needed is one 256 word block, rounded up to one granule.

Name: The file name is SPr00000. where r is the system format revision code letter.

File Structure

The file consists of one 256 word logical record.

```
FIELD   TYPE   DESCRIPTION

  1      I     Number of disk drives available.
  2      I     Number of tape drives available.
  3      I     Number of lines per CRT page.
  4      I     Number of lines per print page.
  5      I     Number of batch queues.
  6      I     Plotter no. of X dots per page.
  7      I     Plotter no. of Y dots per page.
  8      I     Plotter no. of X dots per character.
  9      I     Plotter no. of Y dots per character.
 10      I     Maximum no. of interactive users.
 11      I     Number of words in AP (in 1024 sections).
 12      I     Number of TV devices available.
 13      I     Number of graphics devices (like TEK 4012) available.
 14-63   I(50) Device table (see below).
 64      I     No. of users allowed access to TVs.
 65      I     No. of users allowed access to graphics devices.
 66      I     No. of entries in private catalogs. 0=> public catalog.
 67      I     Max. user number.
```

```
  68     I     Width of line printer in characters.
  69     R     # 1024s words of secondary AP memory.
  70     I     Shortest vector length to vectorize
  71     R     No. or dots per mm on printer
  72     R     No. of dots per mm on graphics screen.
  73     H(5)  System name or ID (4 characters/floating point).
  78     R(15) Min. TIMDEST time for each disk (days)
  93     R     Min. TIMDEST time for SAVE/GET files (days)
  94     R     Min. automatic destruction time for messages
  95     R     Min. automatic destruction time for scratch
  96     R     Min. destruction time for empty catalogs.
  97     R(4) Times during which AP Batch jobs cannot start.
               1, 2 start, stop times (hrs) on weekends
               3, 4 start, stop times (hrs) on weekdays
 101     R(3)  1 => time between rolls (min)
               2,3 polynomial terms for determining how long
               a job must wait before grabbing the AP.
 104     R(120) Lists of allowed users, 8 per disk for up to 15
                disks. 0=all, -1=scratch only otherwise user numbers.
                Ordered by disk.
 224     R(2)  Graphics screen size x,y
 226     R(2)  Graphics character size x,y

  REST OF BLOCK  Free
```

The device table is used to describe the characteristics of a file that can be opened on a specific AIPS I/O logical unit number. The logical unit numbers run from 1 to 50 and are described by its corresponding device table value. The codes for the device table values are listed below.

```
 DEVICE TABLE       TYPE OF FILE OPENED ON THE CORRESPONDING LUN
    VALUE
      0             File manager file with FTAB
      1             Fortran device (no FTAB)
      2             Non-file manager with FTAB
      3             Other, no FTAB
      4             television device with FTAB
```

If this isn't clear, don't be alarmed. The device table need not be changed unless significant alterations or additions to AIPS are made.

Routines and commons for reading the system parameter file.

The subroutine ZDCHIN (called by every program in the initial stages of execution) will open, read and then close the system parameter file. The information is then transfered by ZDCHIN to commons DCHCOM and FTABCM which are found in include file DDCH.INC.

Routine for initializing and updating the system parameter file.

Stand alone program SETPAR allows the AIPS manager to initialize the system parameter file during AIPS installation or to modify the file when conditions warrant (for example when another disk is made available). SETPAR is an interactive program with prompts. The first prompt that appears when SETPAR is run is:

```
Enter:  1=Init, 2=Change vals, 3=Change DEVTAB, 4=Quit
```

Option 1 (INIT). This option is normally chosen during AIPS installation. SETPAR will initialize the values in the SP file with the default values but SETPAR will not create an SP file. This must be done with program FILINI or FILAIP. After initializing the values SETPAR will immediately proceed to option 2 described below.

Option 2 (CHANGE VALS) This option allows the AIPS manager to change the current SP file values. When this option is chosen the current values and a description is printed on the CRT. Then the following prompt is printed.

```
Enter number to change or  0 = print,  -1 = Return
```

The “number to change” in the prompt refers to the number on the far left of the description of the value. For example if the user wants to change the number of disks available to AIPS from 2 to 3, then the user finds the following line on the CRT:

```
1  no. of AIPS data disks            2
```

In this case the 1 is the “number to change” and 2 is the current number of disks. Thus, the user enters 1 in free format. The next prompt is:

```
1  no. of AIPS data disks            2
```

The user then enters a 3 in free format. The program then reprints the prompt:

```
Enter number to change or  0 = print,  -1 = Return
```

At this point, the user can change another value by entering the proper positive number, or reprint the latest values by entering a 0, or return to the very first prompt (from which the quit command can be issued) by entering a -1. If values are changed SETPAR will ask for the password before changing the file.

Option 3 (CHG DEVTAB) This option allows the user to change the device table values described above. The details of operation are very similiar to option 2.

Option 4 (QUIT) This option will save the latest SP file values and stop the program.

C.2.13 Task Show and Tell (TC) file

Overview

The Show and Tell file is used to pass parameters to tasks which are already running and which can make some use of parameters during execution.

Details: The use of this file is for interactive, but asynchronous, control of tasks. Thus, the only program to write new parameters into the file is AIPS. Those tasks which can change parameters during execution (e.g. number of iterations), or which can sensibly be told to quit early may read the file at any time. If information intended for them is present, then the task should fetch that data and remove the indication of the communication to it.

Name: The file is named TCr00000 where r is the system format code. It is a permanent file and must contain at least (4 + 8 * NINTRN) 256-word records. It is created during system installation.

File Structure

The file consists of 4 directory records followed by up to 120 data records. The directory records contain 64-word logical records one for each value of NPOPS, 4 per physical record. Each 64-word record contain 8 8-word logical records, allowing up to 8 TELL operations to be queued for each POPS number at any one time. The structure of these 8-word records is

```
 WORD    TYPE             DESCRIPTION

   1     H(2)      Task name (Hollerith)
   3     I(2)      Time (YY/MM/DD, HH/MM/SS)
   5     I         User number
   6     I         Operation
   7     I(2)      Reserved
```

The data records follow, one for each operation, 8 for each POPS number. They contain adverb values specified in the Inputs section of the HELP file by am "*" or "?" in column 10, preceded by the value of the adverb OPTELL.

C.2.14 Task data (TD) file

Overview

The task data file is used to pass data to spawned tasks and to return a "return code" to the initiating tasks.

Details: The only programs in the AIPS system which are allowed to spawn tasks are AIPS, BATER, AIPSCn, and AIPSBm. These tasks use the task data file to pass binary parameter data to spawned tasks. This file and the TC file are the only mechanism for direct intertask communication. Other forms of intertask communication are accomplished indirectly via the changes produced in the various catalog, map, history, plot, *et. al.* files by the normal functioning of verbs and tasks. SHOW and TELL use file type TC to pass parameters to running tasks in a manner similar to that used for the TD file.

Name: The file is named TDr00000 where r is the system format version number. It is a permanent file and must contain at least 46 256-word records. It must be created during system installation.

File Structure

The file consists of 46 256-word physical records. They are ordered logically in the file as

```
FIELD    TYPE             DESCRIPTION

  1      I(256)    Control record
  2      I(768)   Data area: NPOPS = 1
  3      I(768)   Data area: NPOPS = 2
  4      I(768)   Data area: NPOPS = 3
  5      I(768)   Data area: NPOPS = 4
  6      I(768)   Data area: NPOPS = 5
  7      I(768)   Data area: NPOPS = 6
  8      I(768)   Data area: NPOPS = 7
  9      I(768)   Data area: NPOPS = 8
 10      I(768)   Data area: NPOPS = 9
 11      I(768)   Data area: NPOPS = A
 12      I(768)   Data area: NPOPS = B
 13      I(768)   Data area: NPOPS = C
 14      I(768)   Data area: NPOPS = D
 15      I(768)   Data area: NPOPS = E
 16      I(768)   Data area: NPOPS = F
```

The control record is divided into fifteen five-word entries and the remaining 181 words are ignored. Each entry has the structure:

```
WORD     TYPE             DESCRIPTION

 1       H(2)      Task name: (Hollerith)
 3       I         Return code
 4       I(2)      Reserved
```

The data areas are simply 768 words of binary data. Their structures are almost totally determined by the task being activated. The only standards are that all task-dependent values are in floating point and that the first N integer values convey

```
 1       I         The logon user number
 2       I         The assigned TV device number
```

3	I	The assigned TK device number
4	I	The MSGKILL parameter
5	I	The ISBATCH parameter
6	I	The DBGAIP parameter (controls print levels, debugger use...)
7-8	I	reserved
9	R	DOWAIT value
10	H	Version string (4 characters)

Words 11 through 768, as needed, contain the adverb values required by the task in binary form. Note: character adverbs are sent as Hollerith strings.

Usage Notes

This file is clearly important and somewhat dangerous. It must be handled carefully and kept available to the full system as much as possible. There are standard routines for handling the file. There should not be a need for new routines and/or new uses of this file.

Routines and commons for writing the task data file

There are two standard subroutines which write on the task data file: AU2 which spawns tasks for AIPS and AIPSBn and RELPOP which resumes the initiating programs. The subroutines AUA (called CUA in BATER) which does the verb SUBMIT, CHSTOP which can start up batch queues for AIPSCn, and GTPARM, which is described in the next section, also write on this file.

Except for the batch submission and activation process, task activation is done by the following process. AU2 writes the standard values (listed above) into the first 8 integers and 2 reals of the data area corresponding to the current value of NPOPS (physical records 3*NPOPS - 1 through 3*NPOPS + 1). The remainder is filled by AU2 with the adverb values for the task. Then AU2 places the root task name in record one in the entry corresponding to the current value of NPOPS (i.e. words 5*NPOPS - 4 through 5*NPOPS - 3) and the initial return code (-999) in the return code entry (5*NPOPS-2). Finally, AU2 releases control of the task data file, activates the task, and waits for a resumption signal from the spawned task using subroutine TASKWT. That signal is now given by changing the -999 return code to some other value (or by aborting). AU2 determines the order of the adverbs required by the task by reading adverb names from the Inputs section of the help text file associated with that task. Using the POPS language subroutines, AU2 locates the adverbs in the symbol table, determines from the symbol table the number of words of data corresponding to the adverb name and the location of those words in the K array, and moves those words to the data area. Note that this makes tasks impervious to changes in POPS and the structure of the K array.

RELPOP is the subroutine used by spawned tasks to resume AIPS, AIPSBn, and BATER. A one word "return code" is written on record 1 in the entry corresponding to the current value of NPOPS. At present, this return code has only a binary meaning. A value of zero means that the spawned task was happy, at least when it called RELPOP. Any other value implies problems and causes batch jobs to abort.

The activation of AIPSCm and AIPSBn is less complicated. AIPSC requires three adverbs: BATQUE, the desired batch queue number, DETIME, the desired delay time before starting the job, and VERSION, the desired release for the AIPSB to be used. AIPSB requires only the job number that QMNGR wants to be run. Thus, Inputs files and language routines are not needed. For reasons of speed and simplicity in dealing with the message file, the AIPSs now run their batch checkers (AIPSC) their own value of NPOPS. BATER and its checker both use NPOPS = NINTRN+1. The checker programs will activate the batch control program QMNGR if it is not already running. To do this, Checker uses the data area appropriate to the value of NPOPS under which the QMNGR is supposed to run (NINTRN+2). Other than these differences, the activation of AIPSCn and AIPSBm procedes by the same process as for other tasks. (Note: NINTRN is the maximum number of simultaneous interactive AIPSs allowed.)

Routines and commons for reading the task data file

The subroutine GTPARM is used by all spawned tasks to obtain the data passed by the initiating program. GTPARM reads the first record in the task data file and locates the entry containing the root task name.

This location implies the value of NPOPS which is to be used and the location in the task data file of the binary data. GTPARM then moves the requested number of words from the disk area to a buffer provided by the calling program and initializes MSGCNT, NPOPS, NLUSER, TSKNAM, MSGCNT, and DBGAIP in the include DMSG.INC. Finally, GTPARM zeros the entry containing the task name and updates the control record (no. 1) in the task data file. This last operation is needed in order to prevent interference between AIPSm's and AIPSBm's in initiating tasks with the same root name. GTPARM also returns a parameter (RQUICK) which instructs the task to resume the initiating program as soon as possible or only after the task has done its thing. It is this parameter which forces batch processes to be synchronous, but allows asynchronous processes for interactive users. The formula is simply

```
RQUICK  =  (.NOT. DOWAIT)  .AND.  (NPOPS .LE. NINTRN)
              .AND.  (ISBTCH.NE.32000)
```

GTPARM also picks up the assigned TV and TK device numbers and the code to suppress messages (if the user asserted MSGKILL TRUE).

C.2.15 Tape lock (TP) file

Overview

The tape dummy file is used as a means to take exclusive use of a tape drive.

Details: Many systems do not allow a sub-process to take exclusive use of a tape drive. The parent process (i.e. AIPS) or others can still perform a rewind in the middle of a job which is writing on the tape! In order to avoid this, all open operations on tapes also open exclusive a disk (TP) file.

Name: The files are named TPr0010n, where r is the format revision number and n is the tape drive number. They are permanent files, but need not contain any data. They must be created during system installation.

File Structure

The file contains no data and need not even have any records.

Usage Notes

These files are accessed by ZTPOPN and ZTPCLS and should not be accessed by any other programs.

C.2.16 Task Adverb Save (TS) file

Overview

The Task Save file stores the adverb values for the most recent execution of each task for each user.

Details: When AIPS, AIPSB, and AIPSC "spawn" tasks they pass adverb values to the task via the Task Data file. The values are also saved in a directoried Task Save file. The file contains only one area per task root name, but can hold areas for a large number of tasks. The adverb values used for the last execution of a given task may be retrieved from this file using the task's Inputs file and a process similar to GO. The adverb values may also be put into the file with the verb TPUT without actually spawning the task.

Names: The file is named TSfvvvnn.uuu, where, for all interactive AIPS, uuu and vvv are the user number in hex and nn is 0. For batch AIPS, vvv is 0 and nn is NPOPS. For batch, these files are temporary and last only through the execution of the batch (or Checker) job. For interactive however, they are created when first needed and remain until explicitly destroyed. f is the data format revision code letter.

File Structure

The first six 256-word blocks in the file consist of 5-word logical records, 51 per block. The first logical record (called logical record no. 0) contains

```
FIELD   TYPE          DESCRIPTION

  1     I         Number of 256-word blocks in file
  2     I         Number of tasks currently in directory
  3     I(2)      Most recent write access (packed YY/MM/DD,
                       HH/MM/SS)
  4     I         Reserved
```

Logical records 1 through 305 contain task information:

```
FIELD   TYPE          DESCRIPTION

  1     H(2)      Task name (Hollerith)
  2     I(2)      Last write access time (packed YY/MM/DD,
                       HH/MM/SS)
  3     I         Version code: 1-8 => OLD, NEW, TST,
                       OLDPSAP, NEWPSAP, TSTPSAP, LOCAL, PRIVATE
```

Note that we allow 8-character names here. This permits a user to save the adverb values for verbs (via GO *verb-name*) if so desired. The fact that a verb was specified will be trapped and the sequence TPUT *verb-name* ; *verb-name* will be substituted. Of course, the user can simply save the verb parameters with TPUT *verb-name* .

The remainder of the file contains task data in the form transmitted to the task in the TD file (see description of the TD file in this appendix). Only three records per task are allowed. If the task name occurs in the directory logical record IN, then the task data occur in physical blocks 3 * IN + 4 and, if needed, up to 3 * IN + 6.

Usage Notes

There should be no need for routines other than those listed below to access this file.

Routines which read and write the Task Save file

The subroutine AU2 performs the operations of GO and TPUT (see description of the TD file in this chapter). As it does so, it also opens the TS file (creating one if needed) and locates the task name in the directory (creating an entry and expanding the file, if needed). When it writes a record to the TD file, it writes the same data to the appropriate record of the TS file.

The subroutine AU2A performs the verb TGINDEX which lists the task names and last write times found in the TS file directory. AU2A also performs the verb-like portions of the pseudoverb TGET. Using code nearly identical to that of AU2, it locates the relevant Inputs file and locates the task name in the TS file directory. It then parses the Inputs file identifying the adverb names and properties via POPS language processing routines and transfers the adverb values from the TS file to the in-core K array.

C.3 User data files

This section describes files that are used for user data files other than tables.

C.3.1 Catalog directory (CA) file

Overview

Catalog files contain directory information for the AIPS files stored on a disk.

Names and Locations: There is a catalog file on each disk on which user data can be stored. The catalog refers only to maps, uv data, and scratch files on its own disk volume. The catalogs have physical names corresponding to "CAf00000", where f is the current fomat code letter (A, B, ...) and can be user owned (.uuu = user number in hex) or public (.uuu omitted).

Data Structures and usage notes

File Structure: Each catalog file contains a one block (256-word) header and a number of catalog directory blocks. The header block contains principally the number of catalog blocks in the file; this is set when the file is created or expanded. The directory blocks contain a reference to each catalog entry. The directory is used to speed catalog searches and also contains the status words that register file activity. A catalog to store N entries must have enough space for 1 + CEIL[N/NLPR] blocks (i.e. catalog blocks + directory), where NLPR is defined below and is 25 on normal machines.

Record Formats:

Header Block:

```
OFFSET LENGTH TYPE          DESCRIPTION
   0      1     I   Volume number of disk containing this catalog
   1      1     I   Unused
   2      1     I   Number of catalog blocks in this file
   3      3     I   Date (YYYY, MM, DD) create
   6      3     I   Time (HH, MM, SS) create
   9      3     I   Date (YYYY, MM, DD) last access
  12      3     I   Time (HH, MM, SS) last access
```

Directory Block:

The Mth directory block contains NLPR entries, each NWPL words, indexing the NLPR*(M-1)+1 to the NLPR*M-1 catalog blocks. In a file with N catalog blocks, the first directory block is the 2nd block in the file. The parameters are given by NWPL = 10, NLPR=256 / NWPL.

```
OFFSET LENGTH TYPE          DESCRIPTION
   0      1     I    User ID number; or -1 if slot is empty
   1      1     I    Map file activity status
   2      2     I(2) Date/Time file was last accessed
   4      1     I    User defined sequence number 1 to 9999
   5      5     H(5) Image ID as:
                          User defined name, 12 characters
                          User defined class, 6 characters
                          Program defined type, 2 characters
    ...
```

Directory Usage Notes:

Directory: Map name and class are user defined character strings of 12 and 6 characters that can be used to identify and locate a specific map. The strings are stored as Hollerith characters together with the 2-character string which identifies the "physical" map type, in their slots in the directory. The sequence number is similarly an arbitrary I reference number. The Map Status is an I number registering the activity of the map file itself.

```
STATUS = 0    => no programs are accessing the map file
       = n>0  => n programs are reading the map
       = -1   => one program is writing into the file
       = n<0  => 1 + n programs are reading the map, one
                 program is writing into the file.
```

File Type: This word describes file type. At present only 'MA' => map and 'UV' => single- or multi-source uv and 'SC'=> scratch files are allowed.

Usage protocols:

Maintaining the integrity of the catalog entries is essential to insure reliable access to the map files. Thus certain rules should be followed when using the catalog. These rules are coded in to the utility routines described below; these routines should be used when at all possible to access the catalog.

Rules:

1. Take exclusive use of the catalog whenever you access it. The required operation should be done quickly and then the catalog file should be closed and released.

2. The status word must be monitered to see if an intended catalog or map operation will disturb an (asynchronous) operation already in progress.

Specifically: Do not modify a catalog block, nor write into a map file which is not in a rest state (STATUS = 0).

If you intend to write into a map and STATUS = 0, change the status to "WRITE" (STATUS = -1) before releasing exclusive use of the catalog.

If you intend to read a map file or catalog block, check to see if someone else is writing on it (STATUS $<$ 0). If so decide whether this is acceptable to your program. If so modify the status to indicate use;

```
STATUS = 1 + STATUS if STATUS > 0
STATUS =-1 + STATUS if STATUS < 0.
```

Clear status when you have finished your operation. If you were reading, reverse the process just described. If you were writing; STATUS = - (1 + STATUS)

Subroutines

- CATDIR searches, lists, and modifies the catalog directory
- CATIO reads and writes catalog blocks and can modify status
- CATOPN opens the catalog file on a given volume
- MCREAT, MAPOPN, MAPCLS, and MDESTR handle most of the catalog bookkeeping while creating, opening, closing, or destroying map files.

C.3.2 Catalog header (CB) file

Overview

Image header files contain descriptions of the format and contents of standard image files.

Names and Locations: There is an image header file for every image stored on disk (MA, UV, SC, or whatever) and cataloged in the corresponding CA file. The header files have physical names corresponding to CBfccc01.uuu where f is the format revision code, ccc is the catalog number of the image file in the CA directory file, and uuu is the user number. Note that ccc and uuu are in hex.

Data Structures and usage notes

File Structure: Each catalog header file contains a one block (256-word) binary header and a number of keyword = value entries in as many following blocks as are required. A catalog file must be at least two blocks long.

Record Formats:

Catalog Blocks:

```
TYPE   POINTER      DESCRIPTION

H(2)    KHOBJ       Source name
H(2)    KHTEL       Telescope, i.e., 'VLA'
H(2)    KHINS       e.g., receiver or correlator
H(2)    KHOBS       Observer name
H(2)    KHDOB       Observation date in format 'DD/MM/YY'
H(2)    KHDMP       Date map created in format 'DD/MM/YY'
```

```
H(2)     KHBUN       Map units, i.e., 'JY/BEAM '
H(2)(7) KHPTP       Random Parameter types
         KIPTPN= 14  Max. number of labeled random paramaters
H(2)(7) KHCTP       Coordinate type, i.e., 'RA---SIN'
         KICTPN= 7   Max. number of axes
D(7)     KDCRV       Coordinate value at reference pixel
R(7)     KRCIC       Coordinate value increment along axis
R(7)     KRCRP       Coordinate Reference Pixel
R(7)     KRCRT       Coordinate Rotation Angles
R        KREPO       Epoch of coordinates (years)
R        KRDMX       Real value of data maximum
R        KRDMN       Real value of data minimum
R        KRBLK       Value of indeterminate pixel (real
                      maps only)
I        KIGCN       Number of random par. groups.
                     This is the number of uv data
                     records.
I        KIPCN       Number of random parameters
I        KIDIM       Number of coordinate axes
I(7)     KINAX       Number of pixels on each axis
I        KIIMS       Image sequence no.
H(3)     KHIMN       Image name (12 characters)
         KHIMNO= 1   Character offset in HOLLERITH string
H(2)     KHIMC       Image class (6 characters)
         KHIMCO= 13  Character offset in HOLLERITH string
H        KHPTY       Map physical type (i.e., 'MA','UV') (2 char)
         KHPTYO= 19  Character offset in HOLLERITH
I        KIIMU       Image user ID number
I        KINIT       # clean iterations
R        KRBMJ       Beam major axis in degrees
R        KRBMN       Beam minor axis in degrees
R        KRBPA       Beam position angle in degrees
I        KITYP       Clean map type: 1-4 => normal,
                     components, residual, points.
                     For uv data this word contains a
                     two character sort order code.
I        KIALT       Velocity reference frame: 1-3
                     => LSR, Helio, Observer +
                     256 if radio definition.
D        KDORA       Antenna pointing Right Ascension
D        KDODE       Antenna pointing Declination
D        KDRST       Rest frequency of line (Hz)
D        KDARV       Alternate ref pixel value
                     (frequency or velocity)
R        KRARP       Alternate ref pixel location
                     (frequency or velocity)
R        KRXSH       Offset in X (rotated RA) of phase center
R        KRYSH       Offset in Y (rotated Dec) from tangent pt.
H(20)    KHEXT       Names of extension file types (2 char)
         KHEXTN= 20  Max number of extension files
I(20)    KIVER       Number of versions of corresponding
                     extension file.
```

Comments

General

1. Standard names are given for the pointer variables. The values for the pointers are computed by the subroutine VHDRIN and are machine-dependent. The values are found in the common /HDRVAL/ via include DHDR.INC. The characters of each H*8 variable are packed separately (and left-justified) in as many hollerith variables as required. The image name, class, and physical type are as a 20-character string in as many Hollerith variables as required.

2. The header contains 256 words and should be contained in the arrays CATBLK(256), CATH(256), CATR(256) and CATD(128) which are INTEGER, HOLLERITH, REAL, and DOUBLE PRECISION resp. and are equivalenced. Pointers of the type KI... should refer to CATBLK locations, KH... to CATH locations, KR... to CATR locations and KD... to CATD locations. E.g. CATD(KDORA) contains the D pointing right ascension. The variable names used here are now standards and should be used wherever possible.

3. When used in the image catalog (IC file) instead of the CB file some parameters are given new meanings. See the IC file description for details.

Specific

- KHINS: Any special equipment etc. used during observations
- KHPTP: Random parameters are those associated with an irregularly gridded "array".
- KHCTP: Seven coordinates!!! Four will commonly be used; RA, DEC, FREQ and STOKES.
- KDCRV: In keeping with the FITS format convention, angles are expressed in degrees.
- KREPO: Somewhat astronomically specific. 1950.0 or 2000.0 are used.
- KRBLK: The value used to specify that a pixel is undefined; usually 'INDE'. If there are no blanked pixels, set to 0.0.
- KHEXT: The types of subsidiary files associated with the map are given by a two letter designation; eg.'HI' for history files, 'PL' plot file.
- KIVER: The current highest version number of the associated file type listed in the same relative array position in the previous type listing.

Keywords

The keyword section contains one or more records beginning in record two of the file. The first keyword record uses words one through six as

```
WORD      TYPE      USE
  1        I      Number of 256-integer records in file
  2        I      Number of keywords in file
3-6        I(4)   Reserved
```

Keyword N is found in disk record N/51 + 2 at word 5 * MOD (N,5) + 2. In other words, each keyword logical record requires 5 words, the first logical record in disk record 2 is logical record 0 used for the control data listed above, and the first word of all disk records (except the control) is ignored. A logical record contains:

```
WORD      TYPE      USE
1-2       H(2)    Keyword (8 HOLLERITH characters)
3-4        *        Keyword value (word 4 used for
                       type 3 and, if NWDPDP = 2,
```

```
                                for type 1)
         5         I         Keyword type:
                                1 double precision floating
                                2 single precision floating
                                3 character (8 HOLLERITH chars)
                                4 integer
                                5 logical
```

Subroutines

- CATCR creates CA files
- CATDIR searches and modifies the catalog directory, creates CB
- CATIO reads and writes catalog blocks and can modify status
- CATKEY reads/writes keywords in the CB file
- CATOPN opens the catalog file (CA) on a given volume
- MCREAT, MAPOPN, MAPCLS, and MDESTR handle most of the catalog bookkeeping while creating, opening, closing, or destroying map files.

C.3.3 Gain (GA) file

Overview

This extension file for a uv data set contains the gains resulting from ASCAL.

Details: GA files use the EXTINI-EXTIO file structure. With the rescaling factor in the file header. The logical record length is 256 words

Names: The file name is GArsssvv where r is the format revision code, sss=catalog number and vv = version number.

File structure.

The file header record contains the following:

```
   Location
Location  Type            Description
  255      R      GM=mean gain modulus.
```

Logical record structure.

```
  Location
 Addr  Type   Name           Description
   1    CX    GAIN(28,2)     IF gains
  113   I     IFLG(2)        Packed logical IF flags 1st 28=IF 1,
                             2nd 28=IF 2.
  115   I     BFLG(24)       Packed logical baseline flags,
                             1st 378 = IF 1, next 378 = IF 2.
                             The order number for baseline i-j
                             is j - 28 + i(55 - i)/2
  139   I     KV             Last vis number in the time range.
  140   R     T1             Start time of solution interval.
                             NOTE: times are in seconds.
  141   R     T2             End time of solution interval.
  142   I     IREF(2)        Reference antenna for the 2 IFs.
```

```
 144    I     GACAL(28,2,2) (ant, if, 1) = Tsys (used by VBANT)
                            (ant, if, 2) = Tant
```

User notes.

When calling EXTINI use LREC=256.

Routines to write GA files:

EXTINI and EXTIO, GA files are currently written by ASCAL.

Routines to access GA files:

EXTINI and EXTIO are used to access GA files.

C.3.4 History (HI) file

Overview

History files are used to record, in character form, the processing history of an image.

Details: Although they are in fact separate physical files, they are treated as if they were extensions of the file containing the image data. There must be one and only one history file for each image. The files are normally created with relatively modest size and extended when necessary.

Names: The file names are HIfsss01 , where f is the format version code and sss is the sequential position of the image header in the catalog file for that disk in hex. With user owned file systems, the names are HIfsss01.uuu, where uuu is the user number in hex.

File Structure

History files consist of sequential 256-word records. Each of these physical records begins with 4 words of special information (used only in the first record). The rest of each physical record consists of 72-character logical records in AIPS HOLLERITH form (4 characters per integer or floating). There are NL such logical records per physical record where NL = 252/(72/4) = 14.

The special data in the first record has the form

```
FIELD    TYPE         DESCRIPTION

  1       I       Number of logical records currently in file
  2       I       Number of logical records which can fit in file
                  before expansion is required.
  3       I(2)    Reserved.
```

The logical records consist of 72 HOLLERITH characters with no required format. However, the following FITS-like usages are recommended strongly:

1. The first 5 characters should be the left-justified name of the program causing this history record and the sixth character should be blank.
2. The rest of the card should show parameter values in the form <keyword> = <value> where more than one keyword may appear on a "card" (logical record). Character string values must be enclosed in single quotes (').
3. Information which should never be parsed (i.e. comments) should follow any parsable (i.e. keyword=value) information and should be preceded by the slash (/) character.

Thus,

```
IMLOD / Run at 14:23:06 on October 15, 1983
IMLOD INNAME='CASA' INCLASS='IPOL' INSEQ=3 / Source select
```

are good logical records for history files.

User notes

History files are important. Any program which produces an output image should copy the history file(s) of the input image(s) to the history file of the output image. It should then record in that file all parameters relevant to the function of the program. Parameters which were defaulted by the user should have the actually-used values inserted in the history file. History files need to be readable both to humans and to computers. Thus comments and parameter names should not be overly terse or crowded onto the cards. However, one should also try to avoid having the history files grow without bounds. Thus, excessive commenting and one keyword per card are poor practices. The parsing rules should be those of ANSII-standard Fortran 77 list- directed I/O.

Routines and commons for writing history files

There is a collection of basic routines to create, open, add to, read, write, and close history files. They all use the basic common called /HICOM/ in DHIS.INC:

```
NHIFIL  I                max number of history files open at once
NHIWRD  I                number of words/entry in HITAB
NHIWPL  I                # words / logical record
NHILPR  I                # logical records / physical record
HITAB   I(NHIFIL*NHIWRD)  table describing open history files
```

If HIPNT is the pointer to an entry in HITAB, then the structure is

```
HITAB(HIPNT+0)    logical unit number of file
HITAB(HIPNT+1)    pointer to FTAB for file
HITAB(HIPNT+2)    number of logical records now in file
HITAB(HIPNT+3)    number logical records which will fit in file
HITAB(HIPNT+4)    disk number for file
HITAB(HIPNT+5)    sequential catalog position for image
HITAB(HIPNT+6)    physical record # now in user-supplied buffer
```

The user of the standard routines must cause the HITAB to receive an appropriate size and to be initialized. This may be done by

```
INCLUDE 'DHIS.INC'
....

CALL HIINIT (n)
```

where n is the number of history files around and is less than 20. The include 'DHIS.INC' declares HITAB(140). If you really must have more than 20 simultaneous HI files, you will need to declare the COMMON and its variables in your own initialization routine. Do follow the example of DHIS.INC and HIINIT, however.

The user may create and open a history file with subroutine HICREA or open an old file with HIOPEN. Among the arguments to these routines is a 256-word buffer which is used as a working buffer by all of the standard history routines. The programmer should not modify the contents of this buffer between the call to HIOPEN (or HICREA) and the corresponding call to HICLOS. Separate buffers must be provided for each open history file. History records are normally added to history files via HIADD. This routine calls HIIO to perform the actual input/output operations (when required only) including expanding the file as needed.

HICLOS is used to complete the writing of the file including updating the pointers in record 1 and to close the file.

There are a number of less basic routines which provide history- related services. HISCOP copies the contents of one open history file into another. HENCO1, HENCO2, and HENCOO encode and write to files some of the basic parameters used by most tasks.

Routines and commons for reading history files

At this writing, there are no special routines for reading history files. Normally, one uses HIINIT, HIOPEN, and HICLOS and the common /HICOM/ as described above. HILOCT may be used to return the location of the entry in HITAB (called HIPNT above). With this pointer, one may use HIIO (or simply ZFIO) to read the file one physical record at a time.

C.3.5 Image (MA) file

Overview

Map data files contain the binary pixel values for the n- dimensional "map" array ("image").

Details: Each map file holds only one image. The header which describes that image is stored in the catalog file for the disk on which it resides. (See CB file description in this section.)

Names: Map file physical names are MArsss01, where r is the format revision code and sss is the sequential position of the header in the catalog file for that disk. The logical names for map files, used by users, have no relationship to this physical name.

Record formats

The pixel values are stored in binary form in map files, normally as real values. The image array data are ordered in the standard Fortran order in which the first axis counter varies fastest. Each "plane" of the array (one full cycle of the first two axis counters) must begin on a sector boundary and occupy one or more consecutive sectors as required. Thus, depending on row length and the number of bytes per sector on the local machine, some disk space may be wasted. This waste is tolerated in order to obtain improved IO performance on (the normal) row reads and writes. The full map occupies NGPMAP = CEIL ((NSROW * NAXIS2 * NAXIS3 ...) / NSPG) granules, where NBPS and NSPG are in the common /DCHCOM/ (DDCH.INC) described in Chapter 6 on Disk I/O.

Usage Notes

Map files are always read with the fast IO routines MINIT, MDISK, ZMIO, ZWAIT, et al..

Routines and commons for reading and writing map files

These routines and commons are described in the chapter on Disk I/O.

C.3.6 Plot (PL) file

Overview

Plot files are a generalized representation of a graphics display. They contain scaling information and commands for drawing lines, pixels, and characters. Plot files for a certain type of plot (contour maps, grey scales, etc.) are written by one routine and then another routine must read the plot file and write to a particular graphics device (plotter, graphics terminal, etc.). In AIPS these functions are performed by separate tasks. This two step approach offers several advantages. A plot file may exist for an extended period of time allowing plots to be written to different devices and copies to be generated at later times without duplicating the calculations needed in making the plot. Also only one program for a particular plot need be written instead of one program for each graphics device.

I/O to the plot files consists of reading and writing 256-word blocks. The logical records are of 9 types and vary in length. With the exception of the 'draw pixels' record, logical records do not span the 256-word blocks. Unused space at the end of a block consists of integer zeros. All values in the plot file are I variables or Hollerith characters. This aids in exporting plot files to other computers via tape. Plot files have names of the format PLrsssvv, where r is the format revision code, sss is the Catalog slot number of the associated map, and vv is the version number. Plot files are usually extension files associated with a cataloged map.

Record layouts

(0) The first physical record contains information about the task which created the file. It is not logically part of the "plot file", but is there to provide documentation of the file's origins should it be needed. The first logical record (see below) starts in word 1 of the second physical record in the file. The contents of the first physical record are task-dependent and have the form:

```
FIELD   TYPE        DESCRIPTION
  1.    H(3)       Task name
  2.    I(6)       Date/time of file creation YYYY,MM,DD,HH,MM,SS
  3.    I          Number of words of task parameter data
  4.    R(*)       Task parameter block as transmitted from AIPS
                     (preferably with defaults replaced by the values
                      used).
```

(1) Initialize plot record.
The first logical record in a plot file must be of this type.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equal to 1 for this record type).
  2.   I      User number.
  3.   I(3)   Date:  yyyy, mm, dd
  4.   I      Type of plot:  1 = miscellaneous
                             2 = contour
                             3 = grey scale
                             4 = 3D profile
                             5 = slice
                             6 = contour plus polarization lines
                             7 = histogram
```

(2) Initialize for line drawing record. This record provides scaling information needed for a plot. The plot consists of a 'plot window' in which all lines are drawn and a border (defined in terms of character size) in which labeling may be written. The second record in a plot file must be of this type.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equal to 2 for this record type).
  2.   I      X Y ratio * 100.  The Ratio between units on the X axis
              and units on the Y axis (X / Y).  For example if the
              decrement between pixels in the X direction on a map
              is twice the decrement in the Y direction the X Y ratio
              can be set to 2 to provide proper scaling.  Some
              programs may ignore this field.  For example IISPL when
              writing grey scale plots to the IIS.
  3.   I      Scale factor (currently 16383 in most applications).
              This number is used in scaling graph positions before
              they are written to disk.  BLC values in field 4 are
              represented on disk by zero and TRC values are
              represented by integers equal to the scale factor).
  4.   I(4)   The bottom left hand corner X and Y values and the top
```

```
              right hand X and Y values respectively in the plot
              window (in pixels).
  5.   I(4)   1000 * the fractional part of a pixel allowed to occur
              outside the (integer) range of BLC and TRC (field 4
              above) in line drawing commands
  6.   I(4)   10 * the number of character positions outside the plot
              window on the left, bottom, right, and top respectively
  7.   I(5)   Location of the X Y plane on axes 3,4,5,6,7.  This
                field is valid only for plots associated with a map.
```

(3) Initialize for grey scale record. This record if needed must follow the 'init for line drawing' record. This record allows proper interpretation of pixels for raster type display devices. Programs that write to line drawing type devices (e.g. the TEKTRONIX 4012) ignore this record.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equals 3 for this record type).
  2.   I      Lowest allowed pixel intensity.
  3.   I      Highest allowed pixel intensity.
  4.   I      Number of pixels on the X axis.
  5.   I      Number of pixels on the Y axis.
```

(4) Position record. This record tells a device where to start drawing a line, row/column of pixels or character string.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equals 4 for this record type).
  2.   I      scaled x position i.e. a value of 0 represents the
              BLC values defined in the 'init for line drawing'
              record, and a value equal to the scale factor
              represents the TRC value.
  3.   I      Scaled Y position.
```

(5) Draw vector record. This record tells a device to draw a line from the current position to the final position specified by this record.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equals 5 for this record type).
  2.   I      Scaled final X position.
  3.   I      Scaled final Y position.
```

(6) Write character string record. This record tells a device to write a character string starting at the current position.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equals 6 for this record type).
  2.   I      Number of characters.
  3.   I      Angle code: 0 = write characters horizontally.
                          1 = write characters vertically.
  4.   I      X offset from current position in characters * 100
  5.   I      Y offset from current position in characters * 100
                (net position refers to lower left corner of 1st char)
  6.   H(n)   Hollerith characters (n = INT((field2 + 3) / 4))
```

(7) Write pixels record. This record tells a raster type device to write an n-tuple of pixel values starting at the current position. Programs that write to line drawing type devices ignore records of this type.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equals 7 for this record type).
  2.   I      Number of pixel values.
  3.   I      Angle code: 0 = write pixels horizontally.
                          1 = write pixels vertically (up).
  4.   I      X offset in characters * 100.
  5.   I      Y offset in characters * 100.
  6.   I(n)   n (equal to field 2) pixel values.
```

(8) Write misc. info to image catalog record. This record tells the programs that write to interactive devices (TEKPL, IISPL) to put up to 20 words of miscellaneous information in the image catalog starting at word I2TRA + 2. This information is interperted by routines such as AU9A (TKSKYPOS, TKMAPPOS, etc.). Routines that write to non- interactive graphics devices (PRTPL) ignore this record.

```
FIELD  TYPE      DESCRIPTION
  1.   I       Opcode (equals 8 for this record type).
  2.   I       Number of words of information.
  3.   I(n)    Miscellaneous info (n=value of field 2).
```

(9) End of plot record. This record marks the end of a plot file.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equals 32767 for this record type).
```

Subroutines and Commons for Writing

- GINIT Creates and opens a plot file, initializes the graphics common GPHCOM and writes an 'Initialize plot' record.
- GINITL Writes an 'init for line drawing' record to the plot file.
- GINITG Writes an 'init for grey scale' record to a plot file.
- GPOS Writes a 'position' record to the plot file.
- GVEC Writes a 'draw vector' record to the plot file.
- GCHAR Writes a 'write character string' record to the plot file.
- GRAYPX Writes a 'write pixels' record to the plot file.
- GFINIS Writes an 'end of plot' record to the plot file and closes the file.
- GPHWRT This routine is called by the above routines if I/O is needed. It writes the current buffer to the plot file and zeros the buffer. Then the record count and buffer position pointer are updated in common GPHCOM.
- GPHCOM (common) contains variables used for inter-subroutine communication. It is declared by including DGPH.INC and has the structure:

```
GPHSIZ  I      File size in 256-word blocks
GPHLUN  I      Logical unit number
GPHIND  I      Pointer to FTAB
GPHPOS  I      Position in work buffer of last word used
GPHRRN  I      Number of records written
GPHVOL  I      Disk volume number
GPHNAM  I(12)  Physical file name
GPHX1   R      Leftmost (col) pixel position
                 (as passed in file logical record type 2,
```

```
                    field 4 above; i.e. integer part)
GPHX2   R      Rightmost (col) pixel position
GPHY1   R      Lowest (row) pixel position
GPHY2   R      Highest (row) pixel position
GPHTLO  I      Lowest allowed pixel value
GPHTHI  I      Highest allowed pixel value
```

This common is handled fully by the routines listed here. A user of these routines only needs to declare the common in his main program to insure that adequate space is reserved for it.

Subroutines and Commons for Reading.

- TVPL Reads a plot file and writes the corresponding commands to the tv device.
- TVSPCL (common) for inter-subroutine communication for IISDRW and its subroutines.
- PRTDRW Reads a plot file and builds a bit map representation on disk. This bit map is then written to a printer/plotter.
- CPRT (common) allows inter-subroutine communication for PRTDRW and its subroutines.
- TEKDRW Reads a plot file and writes the corresponding commands to the TEKTRONIX 4012 graphics screen.
- TVCHAR (common) for inter-subroutine communication for TEKDRW and its subroutines.
- IMAHDR (common) plot file catalog header information.used in IISDRW, PRTDRW, and TEKDRW.

C.3.7 Slice (SL) file

Overview

Slice files are usually extension files associated with a map file. The slice file contains interpolated data points that lie along a vector from one map pixel to another. End points may be fractions of a pixel. The slice file may also contain the parameters for a number of models of the slice file. Each model may contain up to 4 gaussian components.

Details: SL files use a modified EXTINI-EXTIO file structure. The logical record length is 256 words. The number of data points in the slice is determined by the following algorithm:

```
If the length of the vector is greater than 1024 then 4096.
If the length of the vector is greater than 512 then 2048.
If the length of the vector is greater than 256 then 1024.
otherwise the no. of points is 512.
```

Names: The system name for a slice file is SLrsssvv where r is the release code, sss is the catalog slot number of the associated map file, and vv is the version number (starting at 1) of the slice file.

Record formats

Slice files are sequential files consisting of 256 word records. The first record in the file is of the type created by EXTINI and contains useful miscellaneous information. The 2nd record contains the inputs the user specified when creating the slice file. The slice data points start in record three and continue for as many records as necessary to use up the data. Models, if extant, immediately follow the data using one record per model.

WORD	TYPE	DESCRIPTION
1	I	# 512-byte records in the existing file
2	I	# logical records to extend the file when req.
3	I	max. # of logical records

```
  4     I       current number of logical records
  6     I       # values per logical record.
  7     I       # of logical records per physical record, if neg then
                the # of physical records per logical record.
  8     H*6     Creation task name
 11     I(6)    Creation date, time
 17     H*48    File name
 29     I       Volume number on which file resides.
 30     H*6     Last write-access task
 33     I(6)    Last write-access time,date
 39     I(18)   reserved. (53-56 used by EXTIO.
 57     I       Number of slice data points.
 58     I       Number of model records currently in the file.
 59     I       The record that may contain the first slice model.
                Includes header record and numbers from one.
```

(1) The second physical record
All default values for the fields have been filled in.

```
WORD   TYPE     DESCRIPTION
  1     H*6     Program name (slice).
  4     I(3)    Date (year, month, day).
  7     I(3)    Time (hour, min, sec).
 10     I       Number of words in the following inputs
                section of the record (25 at this time).
 11     R       User number of the map file.
 12     H*12    Name of the associated map file.
 15     H*6     Class of the associated map file.
 17     R       Sequence number of the associated map file.
 18     R       Disk volume number of slice and map files.
 19     R       Type of the map file.
 20     R(7)    Starting pixel of the slice vector.
 27     R(7)    Ending pixel of the slice vector.
 34     R       Maximum value of slice pixels.
 35     R       Minimum value of slice pixels.
```

(1) Slice data points

```
WORD   TYPE     DESCRIPTION
  1     R(256)  Slice data points in floating point.
```

(2) Slice models.

```
WORD   TYPE     DESCRIPTION
  1     R(12)   Parameters for up to 4 gaussian components in the order
                maximum amplitude (physical units), position (slice
                point), half width (slice points), maximum amplitude ...
 12     R(12)   Errors for corresponding parameters.
 25     I       Slice points not fitted from the beginning of the slice.
 26     I       Slice points not fitted from the end of the slice.
 27     I       Number of Gaussians.
 28     I(3)    Date as year, month, day.
 31     I(3)    Time as hours, minutes, seconds.
 34     I(4)    Minus ones indicate initial guess for position of
                corresponding gaussian component held constant.
```

```
  38      I(4)     Minus ones indicate initial guess for maximum amplitudes
                   held constant.
  42      I(4)     Minus ones indicate initial guess for half widths held
                   constant.
```

User notes.

When calling EXTINI use LREC=256.

Routines to write SL files:

EXTINI and ZFIO, SL files are currently written by SLICE and SLFIT.

Routines to access SL files:

EXTINI and ZFIO are used to access SL files.

C.3.8 Uv data (UV) file

Overview

UV files contain interferometric data or single dish data randomly sampled on the sky. The files consists of a series of records each with a set of "random" parameters and a rectangular data array. The "random" parameters are descriptions of the associated data array and give such information as u, v, and w or feed number. The structure of these records is flexible and depends on the particular application.

Details: A UV file contains an arbitrary number of records in an arbitrary order. A description of the structure, number and order of records is given in the associated catalog header (CB) record. A number of extention tables may be need to fully interprete the data.

Names: UV file physical names are UVrsss01, where r is the format revision code and sss is the sequential position of the header in the catalog file for that disk. The logical names for UV files, used by users, have no relationship to this physical name.

Record formats

All data are stored as single precision values. The exception to this is "compressed" format data in which "visibility" data may be stored in a machine dependent form. Data are stored with random parameters first followed by the data array in Fortran order. There is no unused space between records and records may span disk sectors.

Usage Notes

UV files are always read with the fast IO routines UVINIT, UVDISK, ZMIO, ZWAIT, et al..

Routines and commons for reading and writing UV files

These routines and commons are described in the chapters on Disk I/O and Calibration.

C.4 Table details

This section describes the standard AIPS tables files. These are user data files using the conventions described in the chapter on Tables in this volume. These are used heavily in the calibration package.

C.4.1 Antenna (AN) table

Overview

This extension table for a uv data set contains information about the antennas and the array geometry including conversion from atomic time to sidereal time. The AN table defines the time system used in the data with header keywords DATUTC which gives the correction for the time system of the data to UTC and TIMSYS which is a label for the time system.

Name: The file name is ANrsssvv where r is the revision code (A, B...), sss=catalog number and vv = version number.

File Structure

Logical records consist of the information for a single antenna. In the case of an orbiting antenna the elements of the orbit are given. For each antenna two polarization feeds are assumed (A and B) For orbiting antennas the orbital elements are given by ORBPARM. The file header record contains the following KEYWORDs:

```
  Keyword type    Description
 --------- ---- ---------------------------------------------------
  ARRAYX   D    Array center X coordinate (meters, earth center)
  ARRAYY   D    Array center Y coordinate
  ARRAYZ   D    Array center Z coordinate
  GSTIA0   D    GST at time=0 (degrees) on ref. date
  DEGPDY   D    Earth rotation rate (deg/day)
  FREQ     D    Obs. Reference Frequency for subarray(Hz)
  RDATE    A    Reference date as 'DD/MM/YY'
  POLARX   E    Polar position X (meters) on ref. date
  POLARY   E    Polar position Y (meters)     "
  UT1UTC   E    UT1-UTC  (time sec.)          "
  DATUTC   E    Data time-UTC  (time sec.).  If 0 then the data has
                UTC time.
  TIMSYS   A    Time system (e.g. 'IAT', 'UTC').  Defaults to 'IAT'.
  ARRNAM   A    Array name
  NUMORB   I    Number of orbital parameters
  NOPCAL   I    Number of polarization calibration constants.
  POLTYPE  A    Feed polarization parameterization, only if the
                feed parameters have been entered.
                        'APPROX  ' => linear approximation
                        'ORI-ELP ' => orientation-ellipticity
                        'X-Y LIN ' => lin. approx. for lin.
                           polarized (X-Y) data.
                        'VLBI'     => R-L linear approximation with
                                      prior correction for parallactic
                                      angle.
```

Table entries:

```
     Title          Units   Code       Description
 ----------------- -------- ----  ------------------------------------
   ANNAME                    8A   Station name
   STABXYZ         meters    3D   X,Y,Z offset from array center
   ORBPARM                   *D   Orbital parameters (see note 1)
   NOSTA                     1I   Station number
```

```
MNTSTA                    1I   Mount type, 0=altaz, 1=equatorial,
                               2=orbiting
STAXOF           meters   1E   Axis offset
POLTYA                    1A   Feed A feed poln. type 'R','L','X','Y'
POLAA            degrees  1E   Feed A feed position angle.
POLCALA                   *E   Feed A poln. cal parameter. (note 2)
POLTYB                    1A   Feed B feed poln. type 'R','L','X','Y'
POLAB            degrees  1E   Feed B feed position angle.
POLCALB                   *E   Feed B poln. cal parameters.
```

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

1. ORBPARM is an array whose dimension is given by the header keyword NUMORB
2. POLCALA and POLCALB are arrays whose dimension is given by the header keyword NOPCAL.

Routines to access AN files

ANTINI and TABAN, AN files are currently written by UVLOD, VLBIN, MK3IN and SETAN.

C.4.2 Baseline dependent calibration (BL) table

Overview

This table contains information for baseline dependent (antenna pair) calibration. The complex gains for each baseline necessary to correct for non–antenna based errors. These errors are assumed to consist of a multiplicative and an additive portion.

Names: The file name is BLrsssvv where r is the format revision code, sss=catalog number and vv = version number.

File structure

Logical records consist of the information for a single baseline at a given time for all IF and polarizations. The file header record contains the following KEYWORDs:

```
Keyword  Code   Description
-------- ----  ----------------------------------------------------
NO_ANT   I     The number of antennas for which there is
               information (actually highest antenna number).
NO_POL   I     The number of polarizations
NO_IF    I     The number of IFs.
```

Table entries:

```
    Title          Units   code      Description
 ----------------- --------- ---- --------------------------------------------
  TIME             Days     1E   Time of center of interval since 0h on
                                 reference day.
  SOURCE ID                 1I   Identification number of the source used.
  SUBARRAY                  1I   Subarray number
  ANTENNA1                  1I   First antenna number.
  ANTENNA2                  1I   Second antenna number.
```

```
  FREQ ID                    1I   Frequency group id.
  REAL M1                    *E   Real part of multiplicative factor for
                                  first polarization. (see note 1)
  IMAG M1                    *E   Imag. part of multiplicative factor for
                                  first polarization.
  REAL A1                    *E   Real part of additive correction for
                                  first polarization.
  IMAG A1                    *E   Imag. part of additive correction for
                                  first polarization.
```

The following are present only if NO_POL = 2

```
     Title          Units  code      Description
----------------- -------- ----   ---------------------------------------
  REAL M2                    *E   Real part of multiplicative factor for
                                  second polarization. (see note 1)
  IMAG M2                    *E   Imag. part of multiplicative factor for
                                  second polarization.
  REAL A2                    *E   Real part of additive correction for
                                  second polarization.
  IMAG A2                    *E   Imag. part of additive correction for
                                  second polarization.
```

User notes

The “code” column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

1. “REAL M1”, “IMAG M1”, “REAL A1”, “IMAG A1”, “REAL M2”, “IMAG M2”, “REAL A2” and “IMAG A2” are arrays whose dimensions are given by the header keyword NO_IF.
2. This table will in practice be used only by standard interface routines.

Routines to access BL files

The AIPS routines BLINI and TABBL will create/read/write BL tables. Chapter 13 gives a detailed description of routines to access tables files.

C.4.3 Bandpass calibration (BP) table

Overview

This table contains information for bandpass calibration; this is frequency channel dependent calibration. The bandpass correction functions are factored into antenna based components.

Names: The file name is BPrsssvv where r is the format revision code, sss=catalog number and vv = version number.

File structure

Logical records consist of the information for a single antenna at a given time for all IF and polarizations. The file header record contains the following KEYWORDs:

```
Keyword  Code   Description
-------- ----  ---------------------------------------------------
NO_ANT   I     The number of antennas for which there is
```

```
                information (actually highest antenna number).
NO_POL    I     The number of polarizations
NO_IF     I     The number of IFs.
NO_CHAN   I     The number of spectral channels present
STRT_CHN  I     First channel number present in file.
```

Table entries:

Title	Units	code	Description
TIME	Days	1D	Time of center of interval since 0h on reference day.
INTERVAL	Days	1E	Time interval covered.
SOURCE ID		1I	Identification number of the source used.
SUBARRAY		1I	Subarray number
ANTENNA		1I	Antenna number.
BANDWIDTH	Hz	1E	Bandwidth of individual channels
IF FREQ	Hz	*D	Reference frequency for each IF
FREQ ID		1I	Frequency group id.
REFANT 1		1I	Reference antenna
REAL 1		*E	Real part of channel gains for first polarization. (see note 1)
IMAG 1		*E	Imag. part of channel gains for first polarization.

The following are present only if NO_POL = 2

Title	Units	code	Description
REFANT 2		1I	Reference antenna
REAL 2		*E	Real part of channel gains for second polarization. (see note 1)
IMAG 2		*E	Imag. part of channel gains for second polarization.

User notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

1. "REAL 1", "IMAG 1", "REAL 2", and "IMAG 2" are arrays whose dimensions are given by the product of the header keywords NO_CHAN, and NO_IF. They can be considered to be arrays dimensioned (NO_CHAN,NO_IF).

2. This table will in practice be used only by standard interface routines.

Routines to access BP files

The AIPS routines BPINI and TABBP will create/read/write BP tables. Chapter 13 gives a detailed description of routines to access tables files.

C.4.4 Clean Components (CC) table

Overview

CLEAN components are stored in this table by CLEAN tasks during the deconvolution process or by modeling programs. CLEAN point components represent a set of sources which when convolved with the dirty beam and added to the residual map gives the original dirty map. The Fourier transform of these POINT components should, after CLEAN converges, reproduce the observed data. Other tasks write CC files containing Gaussian or other model fits to images or UV data.

Names: The file name is CCdsssvv, where d is the update code and sss is the sequential position of the image header in the catalog file for that disk and vv is the version number.

File Structure

Logical records consist of a single model component. If only three columns are present then the file contains only point components. CC tables headers contain no KEYWORDs.

Table entries:

```
    Title          Units   code      Description
----------------- --------- ---- -------------------------------------------

  FLUX              Jy      1E  Flux density of component.
  DELTAX          degrees   1E  Offset from reference pixel in X direction
                                (RA for unrotated images)
  DELTAY          degrees   1E  Offset from reference pixel in Y direction
                                (Dec for unrotated images)
```

The following are optional:

```
  MAJOR AX        degrees   1E  Gaussians: Major axis size (FWHM)
                                Spheres: Radius
  MINOR AX        degrees   1E  Gaussian: Minor axis size.
  POSANGLE        degrees   1E  Gaussian: position angle (from N thru E)
  TYPE OBJ                  1E  Model type:
                                  0 => point model.
                                  1 => gaussian on sky.
                                  2 => gaussian convolved by observation.
                                  3 => uniform, optically thin sphere.
```

User notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

The positions given are those projected onto the same plane as the original image *i.e.* the number of pixels from the reference pixel times the pixel spacing. If accurate positions are desired the appropriate transformation must be made.

Routines for accessing CC files

AIPS routines CCINI, TABINI, TABIO allow read and write access to CC tables. The tasks APCLN, SDCLN and MX (regular cleaning), IMFIT, JMFIT and UVFIT may write in CC files. The tasks PRTCC and PRTAB will print the contents of CC tables. Chapter 13 describes the routines to access tables files.

C.4.5 Calibration (CL) table

Overview

This extension table for a uv data set contains calibration information and the total observed geometric observables. This will include both calibration information passed to AIPS from external sources and that derived from AIPS calibration software. The information in the CL table is composed of two kinds: 1) accountability information which tells what has been done to the data and 2) residual calibration data. The accountability data may have been either applied in earlier processing (e.g. the correlator model) or a correction applied to the residual calibration information. The residual calibration information gives the corrections to be applied to the residual data (the main uv data) to produce fully calibrated astronomical data. Times refer to the arrival time at the array center (defined in the AN table) of the given wavefront.

Modifications of the CL table may be made directly for corrections determined externally to the astronomical data or from solution (SN) tables determined from fits to the astronomical data. All modifications of the CL table should be done in a mannar to accurately maintain the accountability information. Corrections determined from external sources should be applied to the CLOCK, ATMOS, and DISP terms as well as to the residual calibration terms. Corrections from astronomical data (SN table) should be applied to the Multi-band delay residual as well as to the residual calibration terms.

The total correlator model, expressed as a delay, is divided into a number of parts. The purely geometric portion (GEODELAY) is expressed as a time series polynomial. This portion is assumed global for a given time and antenna and the number of terms in the polynomial is arbitrary. A second global portion of the model is the astmospheric delay (ATMOS) and it's time derivative (DATMOS). The atmospheric delay is the excess path length due to the neutral (gray) portion of the atmosphere.

There is a separate clock epoch and rate error for each of two possible polarizations (CLOCK 1, D-CLOCK 1, CLOCK 2, DCLOCK 2). Each possible polarization also has a separate dispersive delay and rate term (DISP 1, DDISP 1, DISP 2, DDISP 2) which describes the dispersive portion of the delay model. This dispersive delay is the delay in seconds at a wavelength of 1 meter and scales to other frequencies as the wavelength squared.

The residual calibration portion of the record is composed of several parts which are of the form of corrections to be applied to the data. Each of the possible polarizations has a "multiband delay" (MBDELAY1 and MBDELAY2). These are not actually applied in the process of calibration of the residual data but are used to keep track of the total multiband correction made. (Multiband delay corrections are incorporated in the IF phase corrections.) Each IF in each polarization has a set of real and imaginary parts of the complex gain calibration term (REAL1, IMAG1, REAL2, IMAG2), time derivatives of phase (RATE 1, RATE 2) and frequency derivatives of phase (DELAY 1, and DELAY 2). The time at which the phase is given by REALn and IMAGn is the label time of the CL table record and the frequency is that of the reference frequency channel of the given IF. The table header has the mean modulus of all complex gain corrections which can be used in self calibration to keep the flux density scale from shifting.

Each polarization and IF also has an associated WEIGHT n which is basically a signal to noise ratio determined from the uncertainty of the phase estimate and the number of the reference antenna used for the fits to astronomical data (REFANT n).

When all calibration is complete, the total observed delay for each polarization at the reference frequency and the label time is given schematically by the following:

$$Total\_delay = GEODELAY + ATMOS + CLOCK - MBDELAY\ n$$

(This probably needs some correction for the dispersive term).

Phase model

This section describes the phase model of the data assumed. The basic phase model is given by:

$$\phi(\nu, t) = 2\ \pi\nu\{D_t + \frac{c^2}{\nu_{i,c}^2} L_t\} + \Phi(\nu, t)$$

where ϕ is phase, ν is the frequency, t is time, D_t is the nondispersive portion of the delay, c is the speed of light, L_t is the purely dispersive portion of the delay (mostly due to the ionosphere), and Φ is residual phase. The first term of this equation is called the *a priori* and the second term the residual.

The *a priori* component of the phase can be further broken down as:

$$\phi_{i,c} = 2\ \pi\nu_{i,c}\{D^g + D^a + D^c + \frac{c^2}{\nu_{i,c}^2} L_t\}$$

where the index i indicates IF and c indicates frequency channel, D^g is the geometric portion of the delay (GEODELAY), D^a is the delay due to the neutral atmosphere (ATMOS), and D^c is the clock epoch error (CLOCK). It is assumed that D^c and L are functions of the polarization measured. The D^a term includes both the wet and dry components of the atmosphere.

The residual phase is assumed to be described by the following:

$$\Phi_{i,c}(\nu,t) = 2\,\pi\{\nu_{i,0} - \nu_{0,0}\}MB + 2\,\pi\{\nu_{i,c} - \nu_{i,0}\}SB_i + 2\,\pi\nu_{i,c}\{t - t_0\}RATE + \theta_{i,c}$$

where a subscript of 0 indicates the reference value, MB is the multiband delay, SB_i is the singleband delay (DELAY n), $RATE$ is the residual fringe rate (RATE n) and θ is the residual peculiar phase. MB, SB, $RATE$ and θ are assumed to be functions of the polarization and SB and θ are also functions of IF. The IF phases stored in the CL table (as REAL n and IMAG n) are the sum of the MB and θ terms in the above expression. The values of $RATE$ are stored in the CL table on an IF and polarization basis but for nondispersive effects these will be the same. The values of MBDELAYn, REALn, IMAGn, DELAY n and RATE n stored in the CL table are in the form of corrections to be applied to data so in general they have the opposite sign from the terms in the expressions above.

File Structure

Names: The file name is CLrsssvv where r is update code, sss=catalog number and vv = version number.

Logical records consist of the information for all IFs for a single antenna at a single time. The file header record contains the following KEYWORDs:

Keyword	Code	Description
NO_ANT	I	The number of antennas for which there is information (actually highest numbered antenna)
NO_POL	I	The number of polarizations.
NO_IF	I	The number of IFs.
NO_TERM	I	The number of terms in the GEODELAY polynomial.
MGMOD	E	The mean gain modulus for the entire table.
REVISION	I	Revision number of the CL table. (10 here)

Table entries:

Title	Units	code	Description
TIME	Days	1D	Time of center of interval since 0h on reference day.
TIME INTERVAL	Days	1E	Interval over which solution was obtained.
SOURCE ID		1I	Identification number of the source used.
ANTENNA NO.		1I	Antenna number.
SUBARRAY		1I	Subarray number
FREQ ID		1I	Frequency group identifier
I.FAR.ROT	rad/m**2	1E	Ionospheric Faraday rotation.
GEODELAY	Seconds	*D	Geometric model delay at the reference time as a time series polynomial.
DOPPOFF	Hz	*E	Doppler frequency offset. (see note 2)
ATMOS	Seconds	1E	"Atmospheric" delay
DATMOS	Sec/sec	1E	Time derivative of ATMOS
MBDELAY1	Seconds	1E	Multiband delay residual for pol. 1.
CLOCK 1	Seconds	1E	"Clock" delay for pol. 1.
DCLOCK 1	Sec/sec	1E	Time derivative of CLOCK 1.

DISP 1	Seconds	1E	Dispersive delay for pol. 1.
DDISP 1	Sec/sec	1E	Time derivative of DISP 1.
REAL 1		*E	Real part of the gain calibration factor; calibrated = raw * CALi * conj(CALj) for antennas i,j.
IMAG 1		*E	Imaginary part of the gain calibration. Phase refers to ref. channel in each IF.
RATE 1	Sec/sec	*E	Residual rate of pol. 1
DELAY 1	Seconds	*E	Residual delay of pol. 1 Note: DELAY n and RATE n are the residual values by which the data are to be corrected during calibration.
WEIGHT 1		*E	Weight of 1st polarization.
REFANT 1		*I	Reference antenna used for solution.

The following are present only if NO_POL = 2

Title	Units	code	Description
MBDELAY2	Seconds	1E	Multiband delay residual for pol. 2.
CLOCK 2	Seconds	1E	"Clock" delay for pol. 2.
DCLOCK 2	Sec/sec	1E	Time derivative of CLOCK 2.
DISP 2	Seconds	1E	Dispersive delay for pol. 2.
DDISP 2	Sec/sec	1E	Time derivative of DISP 2.
REAL 2		*E	Real part of the gain calibration factor for 2nd polarization.
IMAG 2		*E	Imaginary part of the gain calibration. Phase refers to ref. channel in each IF.
RATE 2	Sec/sec	*E	Residual rate of pol. 2
DELAY 2	Seconds	*E	Residual delay of pol. 2
WEIGHT 2		*E	Weight of 2nd polarization.
REFANT 2		*I	Reference antenna used for solution.

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

1. "GEODELAY" is an array of arbitrary dimension containing the terms of a time series polynomial. The number of terms is given by the header keyword NO_TERM.
2. "DOPPOFF", "REAL n", "IMAG n", "DELAY n", "RATE n", "TSYS n", "WEIGHT n" and "REFANT n" are arrays whose dimensions are given by the header keyword NO_IF.
3. "MBDELAY n" should be updated every time the file is updated from a Solution (SN) table.

AIPS routines to access CL files

CALINI and TABCAL read and write CL tables. Also chapter 13 gives a detailed description of routines to access tables files.

C.4.6 Frequency (CH) table

Overview

Note: this table is now replaced by the FQ table and is included here for historical completeness.

This extension table for a uv data set contains relevant information about the IFs in the raw uv data file. For these purposes an IF consists of the output from a receiver or baseband converter; all measured polarizations at the same frequency are considered part of the same IF. IFs can be arbitrarily spaced in frequency. Examples are the A-C and B-D IFs of the VLA and the output of independent video converters in VLBI recorders. Individual, regularly spaced frequency or delay channels derived from the correlation of such IFs are not themselves considered IFs.

Names: The file name is CHrsssvv where r is the format revision code (A, B,...) disk number, sss=catalog number and vv = version number.

File Structure

The table entries give the frequency offset to the center of the IF and the sideband of each IF. The file header record contains no KEYWORDs.

Table entries:

```
 Title               Units  code       Description
----------          ------- ----  ----------------------------------
  IF NO.                    1I    The IF pixel number of the entry.
  FREQUENCY OFFSET    Hz    1D    Frequency offset from ref. freq.
                                  True = Ref + Offset
  SIDEBAND                  1I    Sideband of each IF
                                  -1 => 0 video freq. is high freq. end
                                   1 => 0 video freq. is low freq. end
```

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

The true frequency of the observations are the signed sums of the reference frequency in the subarray antenna (AN) file, the peculiar source doppler offset from the source table (SU file) Any time dependent doppler correction from the CL table and the IF frequency offset from this file.

Routines to access CH files

AIPS utility routine CHNDAT will read of create/write these tables; CHNCOP will copy all or a subset of the table. Also see chapter 13 for a detailed description of routines to access tables files.

C.4.7 Single dish calibration (CS) table

Overview

This extension table for a single dish data set contains position, gain and baseline offset corrections to be applied to single dish data in a uv data like format.

Names: The file name is CSdsssvv where d = release code, sss=catalog number and vv = version number.

File structure

Logical records consist of the information for all IFs for a single beam at a give time. The file header record contains the following KEYWORDs:

```
  Keyword  Code   Description
  -------- ----  ---------------------------------------------------
```

```
NO_BEAM  I     The number of beams for which there is
               information
NO_POL   I     The number of polarizations.
NO_IF    I     The number of IFs.
```

Table entries:

Title	Units	code	Description
TIME	Days	1E	Time since 0h on reference day.
RA	Deg	1E	apparent RA at center time
DEC	Deg	1E	apparent Dec at center time
BEAM NO.		1I	Beam number.
SUBARRAY		1I	Subarray number
FACTOR 1		*E	Gain factor calibrated = factor * (raw + offset) for 1st polarization. (see note 1)
OFFSET 1		*E	offset
RAOFF 1	Degrees	*E	RA correction (to be added)
DECOFF 1	Degrees	*E	Declination correction

The following are present only if NO_POL = 2

Title	Units	code	Description
FACTOR 2		*E	Gain factor calibrated = factor * (raw + offset) for 2nd polarization. (see note 1)
OFFSET 2		*E	offset
RAOFF 2	Degrees	*E	RA correction
DECOFF 2	Degrees	*E	Declination correction

User notes.

The “code” column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

1. “FACTOR n”, “OFFSET n”, “RAOFF n” and “DECOFF n” are arrays whose dimensions are given by the header keyword NO_IF.

Routines to access CS files

Routine CSINI will create/initialize/open/read-keywords-from/ write-keywords-to a solution table and TABCS will do I/O. Also chapter 13 gives a detailed description of routines to access tables files.

C.4.8 Flag (FG) table

Overview

This extension table for a uv data set contains the editing information. This information may or may not have been applied to the data depending on the uv data file type. This file will contain a list of specifications of data to be flagged.

Names: The file name is FGrsssvv where r the AIPS update code, sss=catalog number and vv = version number.

File Structure

Each logical record consists of a specification of data to be flagged. These specifications are independent and may overlap. Data is to be rejected if it is specified in any flagging record that is currently selected. Any entry may be temporarily disabled by deselecting that table entry. The file header record contains no KEYWORDs.

Table entries:

```
    Title      Units   code       Description
------------- --------- ----  -------------------------------------------
  SOURCE                 1I   Source id number of the source to be
                              flagged.
                              0 => all sources
  SUBARRAY               1I   Subarray number, 0 => all
  ANTS                   2I   1st element = number of the first
                              antenna,  0 => all baselines to all
                              antennas flagged. 2nd element =
                              number of the second antenna,
                              0 => all baselines to ANTS(1) flagged
  TIMERANG      Days     2E   Beginning and end time of flagging in
                              the same system as the data is labeled.
                              Both = 0 => data flagged for all times.
  IFS                    2I   Numbers of first and last IF group
                              to be flagged.
  CHANS                  2I   First and last channel number in IF group.
  PFLAGS                 4X   Array of flags for the polarizations.
                              4 are kept even if fewer polarizations
                              are present. Flags in same order as data,
                              bit set => correlator value flagged.
  REASON                24A   Reason code for flagging (24 char)
```

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

Routines to access FG files

FLGINI and TABFLG do I/O to flag files. Chapter 13 gives a detailed description of routines to access tables files.

C.4.9 Frequency (FQ) table

Overview

This extension table for a uv data set contains relevant information about the IFs in the raw uv data file. A data file may contain data made with different sets of IF frequencies or bandwidths. Data containing more than one such frequency group (or set of bandwidths) will have a random parameter indicating an entry in the FQ table which gives the details of the frequencies and/or bandwidths.

Names: The file name is FQrsssvv where r is the format revision code (A, B,...), sss=catalog number and vv = version number.

File Structure

The table entries give the frequency offset to the center of the IF, bandwidth and the sideband of each IF. The file header record contains the following KEYWORD:

```
Keyword  Code   Description
-------- ----  ---------------------------------------------------
NO_IF    I     The number of IFs.
```

Table entries:

```
 Title              Units  code       Description
----------         ------- ---- -----------------------------------
  FRQSEL                    1I    Frequency group identifier
  IF FREQ             Hz    *D    Frequency offset from ref. freq.
                                  (see note 1)
  CH WIDTH            Hz    *E    Channel bandwidth (note 1)
  TOTAL BANDWIDTH     Hz    *E    Total bandwidth (note 1)
  SIDEBAND                  *I    Sideband of each IF (note 1)
                                  -1 => 0 freq. is high freq. end
                                   1 => 0 freq. is low freq. end
```

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

1. "IF FREQ", "CH WIDTH", "TOTAL BANDWIDTH", and "SIDEBAND" are arrays whose dimensions are given by the header keyword NO_IF.

2. The true frequency of the observations are the signed sums of the reference frequency in the subarray antenna (AN) file, the peculiar source doppler offset from the source table (SU file), any time dependent doppler correction from the CL table and the IF frequency offset from this file.

Routines to access FQ files

AIPS utility routine CHNDAT will read or create/write these tables; CHNCOP will copy all or a subset of the table. FQINI and TABFQ also provide access to FQ tables. Also see chapter 13 for a detailed description of routines to access tables files.

C.4.10 Index (NX) table

Overview

This extension table for a uv data set contains an index of the data file. There is an entry for each scan which gives the source, time range, range of visibility numbers and other necessary information. A scan is defined here to consist of a sequence of data from a given subarray which terminates when the source, observing band or observing mode changes or there is a gap in the data much longer than the integration time of the data.

Names: The file name is NXrsssvv where r is the format revision code, sss=catalog number and vv = version number.

File Structure

Logical records consist of the information for a single scan on a single subarray. The file header record contains no KEYWORDs.

Table entries:

```
   Title          Units   code      Description
----------------- --------- ---- -------------------------------------------
  TIME             Days    1E   Time of center of interval since 0h on
                                reference day.
  TIME INTERVAL    Days    1E   Interval over which solution was obtained.
  SOURCE ID                1I   Identification number of the source used.
  SUBARRAY                 1I   Subarray number
  START VIS                1I   First visibility number (0 relative) of
                                data in this scan.
  END VIS                  1I   Last visibility number (0 relative) of
                                data in this scan.
  FREQ ID                  1I   Frequency group identifier.
```

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

Routines to access NX files

The AIPS routines NDXINI and TABNDX will create/read/write NX tables. Chapter 13 gives a detailed description of routines to access tables files.

C.4.11 Solution (SN) table

Overview

This extension table for a uv data set contains gain solutions derived from calibration software in AIPS. This information may or may not have been applied to the data or calibration "CL" table depending on the uv data file type. For multi–source files, the SN table is to be applied to the CL table; for single-source data files, the SN table is applied directly to the data. The table keyword APPLIED indicates if the appropriate application has been made.

Names: The file name is SNrsssvv where r = format revision code, sss=catalog number and vv = version number.

File Structure

Logical records consist of the information for all IFs for a single antenna. The file header record contains the following KEYWORDs:

```
Keyword  Code   Description
-------- ----  ---------------------------------------------------
NO_ANT   I     The number of antennas for which there is
               information
NO_POL   I     The number of polarizations.
NO_IF    I     The number of IFs.
NO_NODES I     The number of interpolation nodes.
MGMOD    E     The mean gain modulus for the entire table.
APPLIED  L     If true then the table has already been applied.
REVISION I     Revision number of the SN table. (10 here)
RA_OFF1  E     The right ascension offset of the first interpolation
               node. (degrees)
DEC_OFF1 E     The declination offset of the first
 ...
```

RA_OFFn	E	The right ascension offset of the n th interpolation node n = NO_NODES.
DEC_OFn	E	The declination offset of the n th interpolation node n = NO_NODES.

Table entries:

Title	Units	code	Description
TIME	Days	1D	Time of center of interval since 0h on reference day.
TIME INTERVAL	Days	1E	Interval over which solution was obtained.
SOURCE ID		1I	Identification number of the source used.
ANTENNA NO.		1I	Antenna number.
FREQ ID		1I	Frequency group identifier
I.FAR.ROT	rad/m**2	1E	Ionospheric Faraday rotation.
SUBARRAY		1I	Subarray number
NODE NO.		1I	Node number
MBDELAY1	Seconds	1E	Multiband delay residual for pol. 1.
REAL 1		*E	Real part of the gain calibration factor; calibrated = raw * CALi * conjg(CALj) for antennas i,j, for 1st polarization. (see note 1)
IMAG 1		*E	Imaginary part of the gain calibration. Phase refers to ref. channel in each IF.
DELAY 1	Seconds	*E	Residual pair delay for pol. 1.
RATE 1	Sec/sec	*E	Residual phase delay rate for pol. 1.
WEIGHT 1		*E	Weight of 1st polarization
REFANT 1		*I	Reference antenna used for solution.

The following are present only if NO_POL = 2

Title	Units	code	Description
MBDELAY2	Seconds	1E	Multiband delay residual for pol. 2
REAL 2		*E	Real part of the gain calibration factor; calibrated = raw * CALi * conjg(CALj) for antennas i,j, for 2nd polarization.
IMAG 2		*E	Imaginary part of the gain calibration. Phase refers to ref. channel in each IF.
DELAY 2	Seconds	*E	Residual pair delay for pol. 2.
RATE 2	Sec/sec	*E	Residual phase delay rate for pol. 2.
WEIGHT 2		*E	Weight of 2nd polarization.
REFANT 2		*I	Reference antenna used for solution.

User Notes

The “code” column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

1. “REAL n”, “IMAG n”, “DELAY n”, “RATE n”, “WEIGHT n” and “REFANT n” are arrays whose dimensions are given by the header keyword NO_IF.

Routines to access SN files

Routine SNINI will create/initialize/open/read-keywords-from/ write-keywords-to a solution table and TABSN will do I/O. Also chapter 13 gives a detailed description of routines to access tables files.

C.4.12 Source (SU) table

Overview

This extension table for a uv data set contains relevant information about the sources in a raw uv data file. This includes positions.

Names: The file name is SUrsssvv where r is the format revision code, sss=catalog number and vv = version number.

File Structure

Each logical record consists of the position and other information about a source in the raw uv data file. Sources are distinguished in the data file by a source ID number.

For moving sources the apparent position is for 0 hours IAT on the reference day defined in the subarray AN file. The motion from this position is given by PMRA and PMDEC. The file header record contains the following KEYWORDs:

Keyword	code	Description
NO_IF	I	The number of IF pairs.
VELTYP	A	Velocity type
VELDEF	A	Velocity definition 'RADIO','OPTICAL'

Table entries:

Title	Units	code	Description
ID. NO.		1I	The source identification number.
SOURCE		16A	Name of the source to be flagged, (16 Char)
QUAL		1I	Source qualifier
CALCODE		4A	Calibrator code (4 char)
IFLUX	Jy	*E	Flux density at reference frequency (Ipol) (See note 1)
QFLUX	Jy	*E	Flux density at reference frequency (Qpol)
UFLUX	Jy	*E	Flux density at reference frequency (Upol)
VFLUX	Jy	*E	Flux density at reference frequency (Vpol)
FREQOFF	Hz	*D	Frequency offset (note 1)
BANDWIDTH	Hz	1D	Bandwidth of channel
RAEPO	degrees	1D	Right ascension at standard mean epoch
DECEPO	degrees	1D	Declination at standard mean epoch
EPOCH	years	1D	Date in years since year 0.0 of the standard epoch.
RAAPP	degrees	1D	apparent Right ascension at 0h IAT on reference day in uv file header.
DECAPP	degrees	1D	apparent declination at 0h IAT on reference day in uv file header.
LSRVEL	m/sec	*D	LSR velocity (see note 1)
RESTFREQ	Hz	*D	Line rest frequency (note 1)
PMRA	deg/day	1D	Proper motion in RA

```
PMDEC         deg/day 1D   Proper motion in Dec
```

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

1. Entries with dimensions marked '*' are arrays whose dimension is given by the header keyword NO_IF.
2. The true frequency of the observations are the signed sums of the reference frequency in the catalog header, the peculiar source frequency offset from this table a time dependent doppler offset from the CL table and the IF frequency offset from the FQ table.

Routines to access SU files

AIPS routines SOUINI and TABSOU create/read/write SU tables. Chapter 13 gives a detailed descriptions of routines to access tables files.

C.5 Task Specific Tables

A number of tasks use temporary tables to keep track of information. These are described in the following sections.

C.5.1 SPFLG baseline (BL) table

Overview

NOTE: This temporary table is not to be confused with the standard calibration BL table. This temporary extension table for a uv data set contains baseline numbers and is used internally by SPFLG.

Names: The file name is BLrsssvv where r the AIPS format revision code, sss=catalog number and vv = version number.

File Structure

The file header record contains no KEYWORD:

Table entries:

```
     Title      Units   code       Description
------------- --------- ---- ------------------------------------------
 ANTENNA1                1I   First antenna number
 ANTENNA2                1I   Second antenna number.
 BASELINE                1I   Baseline code number.
```

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

Routines to access SPFLG type BL files

uses TABINI and TABIO to read and write its BL tables.

C.5.2 SPFLG/TVFLG Temporary Flag (FC) table

Overview

This extension table for a uv data set contains a temporary copy of editing instructions. These are used internally by tasks TVFLG, and SPFLG.

Names: The file name is FCrsssvv where r the AIPS update code, sss=catalog number and vv = version number.

File Structure

The details of these files are identical to FG tables.

C.5.3 ANCAL System Temperature (TY) table

Overview

This extension table for a uv data set contains system temperatures and amplitude calibration factors and is used internally by ANCAL.

Names: The file name is TYrsssvv where r the AIPS update code, sss=catalog number and vv = version number.

File Structure

The file header record contains the following KEYWORD:

```
  Keyword  code     Description
---------- ----     ---------------------------------------------------
  NO_IF     I       The number of IFs.
```

Table entries:

```
    Title      Units   code       Description
------------- --------- ---- -------------------------------------------
 TIME          Day      1R   Time in days
 ANTENNA NO.            1I   Antenna number
 CALFAC                 *R   Amplitude calibration factor 1 / IF
 TSYS         Kelvin    *R   System temperature 1/IF
```

User Notes

The "code" column is element_count + basic type code. basic type codes: D=Double precision, E=single precision, A=character, I=integer, L=logical, X=bit.

"CALFAC" and "TSYS" are arrays of dimension NO_IF.

Routines to access TY files

TYINI and TABTY do I/O to TY files. Chapter 13 gives a detailed description of routines to access tables files.