The AIPS Database Structure and the VLBA

W. D. Cotton
12 January 1983

I. Introduction.

Most of the astronomical data obtained with the VLBA will be processed through the AIPS system or its successor so it is desirable to have a post-correlation database compatable with that used in AIPS. From the standpoint of software managment and simplicity of operation the most desirable solution to the VLBA post-processing problem is to have all operations run as AIPS tasks using the AIPS database structure. This approach would eliminate the problems involved with maintaining separate database formats and with translating data between formats. Using a format incompatable with AIPS databases would result in a serious duplication of effort and needlessly complicate the software effort required to process VLBA data. The purpose of this memo is to describe the current form and use of the AIPS database and to give suggestions for its use for VLBA post-correlation processing.

II. AIPS database structure.

AIPS data files are structures very much like FITS files on tape. There are two basic types of data sets 1) regularly spaced arrays (ie. maps) and 2) irregularly spaced arrays (ie. uv data). Additional types may be added if necessary. Since VLBA data will be predominantly of the second type most of the following comments will be directed towards this type of dataset. There are three distinguishable parts of the AIPS database structure: the catalog header, the main data file and extension files.

A. Catalog header.

The catalog header contains information about a database such as source name, observing date, the amount of data, details of the structure of the main data file and the existance of and number of any extension files. The catalog header record currently has a fixed structure which, although more general than those of previous data reduction systems, is the least flexible portion of the AIPS database.

The current AIPS binary, fixed size header format was adopted for reasons of efficiency. The details of the structure and contents have been changed several times in the past in response to changing needs; the most recent change was done in January 1983. The structure of the AIPS catalog header will undoubtedly respond to the needs of the VLBA. The current description of the catalog header from the AIPS manual is given in Appendix A.

B. Main Data file.

The form of the uv data files in AIPS is a sequence of logical records containing a regular, rectangular array of data (eg. correlator lags, spectral channels, time etc.) and a number of "random" parameters describing the array (u, v, w, time, baseline etc.). The number and order of the random parameters is given in the catalog header. At the present time there is no limit on the number of random parameters but there is space in the catalog header for the labels for only the first seven. Thus with more than seven random parameters the order is not completely specified. (Currently AIPS does not use more than seven random parameters).

The data array in each record is also described in the catalog header which gives the number of axes, the axis type, the dimension of each axis, the axis value increment, a reference pixel (needs be neither integer nor in the bounds of the array given) and the axis value at the reference pixel. The format of the current catalog header allows up to seven dimensions. The current convention is to have RA, Dec, Stokes type and Frequency as an axis even if that axis is degenerate. This allows a convenient way to specify position, frequency etc. This structure requires that there be a uniform spacing along each axis which may present problems for the VLBA (eg. when observations are made in the bandwidth synthesis mode). This limitation of the current AIPS might be circumvented by use of channel number instead of frequency.

The order of logical records in the data file can be changed if necessary. The AIPS manual description of the contents of and access to uv databases is given in Appendix B.

C. Extension Files

Extension files are used to store information not contained in the catalog header or the main data file. An example of an extension file is the History file which is carried along with all AIPS data files. This file contains ASCII records describing all processing which has been done on the data in the file. Other current extension files are the antenna files, the self-calibration solution files and the VLBI fringe fitting solution files.

Extension files contain a header record containing general information; for instance Antenna file headers carry time information like the Greenwich Sidereal Time at IAT midnight for the reference date. Extension file entries consist of fixed length logical records which may be complex data structures. VLBA data bases could use extension files to store antenna logs, correlator logs, calibration tables, correlator models etc. The structure of a typical extension file, the Antenna file, as described in the AIPS manual is shown in Appendix C.

III. AIPS Database Access

AIPS programs generally access the main data files sequentially which allows reading large blocks of data at a time and overlapping I/O and computation by means of double buffering. In order to increase I/O speed, DMA transfer requests are sent directly to system utilities rather that using FORTRAN I/O. This allows programs to directly access the I/O buffer removing one core-to-core copy of the data. The I/O routines are quite fast and are probably comparable in speed with mapping virtual memory onto the data base. The AIPS I/O routines have been designed to maximize speed and flexibility at the cost of increasing the complexity of their use. (The uv data file access routines are described in Appendix B). It may sometime be desirable to add a higher level of more programmer-friendly routines at a cost of reduced preformance.

I/O to extension files is generally sequential but the routines which handle the I/O are capable of random access and mixed reads and writes. This increased flexibility comes at a cost of reduced speed. However, since the extension files are generally much smaller than the main data file the reduced I/O speed is usually not serious.

IV. Limitations of the AIPS database for the VLBA.

- The AIPS conventions and file handling are inelegant from the point of view of a computer scientist (AIPS was designed by astronomers for astromomers). In the development of AIPS, elegance was sacrificed in order to minimize machine dependence and to maximize efficiency. The need for reduced machine dependence and efficient I/O exist for VLBA as well as VLA processing. Operations on VLBA datasets will be very similar to current AIPS uv data access, i.e. sequential access to substantial fractions of or all of a database. The AIPS approach is quite efficient for this type of operation.

- The current catalog header is designed for efficiency but is rather inflexible. In the current AIPS changes in the catalog header require corresponding changes in all affected programs. Several possible solutions have been suggested, the most elegant but most difficult of which is to change the AIPS catalog header to partially or wholly consist of a FITS type header containing ASCII entries of the form "KEYWORD = value". This would give databases the great flexibility of the FITS tape format.

A modification of the scheme described above is to keep the current catalog header but add an ASCII extension file which contains keywords and values not contained in the header. This would be an adequate although unsatisfying solution. A third solution is to use the history file to keep the additional information. The current convention in AIPS is to make history entries in the form "KEYWORD =value"

so that this information is easily available.

- Logical records in both main data files and extension files are fixed length. This might result in inefficient use of storage media for instance if scan oriented records were used and not all antennas were observing all the time. If visibility (baseline) oriented records were used this problem would be much less serious.

- AIPS uv databases currently do not have an associated gain table. Changes made to an AIPS database are applied directly to the data and a new database is created. This scheme protects the original database and allows more efficient I/O in uv data handling routines but probably uses too much disk space to be used for unaveraged VLBA data. There is no fundamental problem with adding VLA-like gain table extension files to AIPS-like databases.

- AIPS uv data is not indexed, thus to find a specific record the file must be searched. This has not proven to be a serious problem in AIPS because most operations are done on a sufficiently large fraction of the database. In several applications (eg. averaging, self-calibration, mapping) the data sets are sorted so that data needed at the same time will be contiguous. For most prefringe fitting applications baseline-time ordering should be sufficient for VLBA processing. If a fixed data order can be adopted prior to fringe fitting, data files could be indexed; however, sorting would destroy the meaning of such an index.

V. Strengths of the AIPS database.

- With the exception of the catalog header the AIPS database is rather flexible with a self documenting format.

- Use of the AIPS data base structure will allow use of AIPS fast I/O routines which will reduce software development costs and managment problems.

- Use of the AIPS database structure and routines will be relatively machine independent as opposed to schemes which involve mapping virtual memory onto the disk data file. The VLBA will likely be used for much longer than the useable lifetime of a given model of computer so it is important to minimize the problems involved with changing computers. Since there are an increasing number of non-NRAO AIPS systems in use such an approach also makes it more feasable for a user with special processing needs to use a computer at his home institution.

- If VLBA post-correlation processing is incorporated into AIPS, one post-correlation control program could handle all astronomical processing. This could avoid the problems facing a VLA user encountering different systems on the DEC 10, the PDPs and the VAXes. The interface to the correlator might be implemented as a special AIPS task which accepts

correlator output and creates an AIPS uv database. This task could run as an AIPS batch job. Use of this technique would allow the full power of the AIPS command language and all of its utilities to be used for the data acquisition and would allow immediate interactive monitoring and preliminary analysis of the data while acquisition proceeds.

VI. Conclusions

It appears to be both desirable and feasable to use the AIPS database for VLBA post-correlation data. Since AIPS or its successor will be used for a substantial fraction of the astronomical processing use of an AIPS-like data base would result in a better integrated system as well as reduce software management problems. Adoption of the AIPS approach will minimize hardware dependence which will allow delaying decisions about the specific computers to be used and will reduce the problems of upgrading the computer.

The fixed record length in AIPS data files will put some constraints on how the data is stored to optimize disk use. But, due to the simplicity of a fixed record length data access may be much faster than, for example, linked list data structures.

Most operations on a VLBA dataset will involve sequential access of a large fraction of the data. The AIPS database structure is well suited for this type of operation and allows the use of fast I/O routines.

APPENDIX A

Map Catalog Files: CAn00000

A. Overview

Function: Map catalog files contain descriptions of the format and contents of standard map files.

Names and Locations: There is a map catalog file on each disk on which maps can be stored. The catalog refers only to maps and associated files on its own disk volume. The catalogs have physical names corresponding to "CAn00000", where n is the disk volume (1,2,...9) where the catalog resides.

B. Data Structures and usage notes

File Structure: Each catalog file contains a one block (256-word) header, a number of one block catalog entries, and a number of catalog directory blocks at the end. The header block contains principally the number of catalog blocks in the file; this is set when the file is initialized. The directory blocks contain a 32-byte reference to each catalog block. The directory is used to speed catalog searches and also contains the map status words that register map file activity.

A catalog to store N entries must have enough space for N + 1 + CEIL[N/NLPR] blocks (i.e. catalog blocks + header + directory), where NLPR is defined below and is 16 on normal machines.

Catalog Blocks:

OFFSET	LENGTH	TYPE	POINTER	DESCRIPTION
0	8	C*8	K4OBJ= 1	Source name
8	8	C*8	K4TEL= 3	Telescope, i.e. 'VLA'
16	8	C*8	K4INS= 5	e.g. receiver or correlator
24	8	C*8	K4OBS= 7	Observer name
32	8	C*8	K4DOB= 9	Observation date in format 'DD/MM/YY'
40	8	C*8	K4DMP= 11	Date map created in format 'DD/MM/YY'
48	8	C*8	K4BUN= 13	Map units, i.e. 'JY/BEAM '
56	7*8	C*8(7)	K4PTP= 15 (K2PTPN= 7)	Random Parameter types
112	7*8	C*8(7)	K4CTP= 29 (K2CTPN= 7)	Coordinate type, i.e. 'LL'
168	8	R*8	K8BSC= 22	Map scaling factor
176	8	R*8	K8BZE= 23	Map offset factor: Real value = BSCALE * pixel + BZERO
184	56	R*8(7)	K8CRV= 24 (K2CTPN= 7)	Coordinate value at reference pixel

240	28	R*4(7) K4CIC= 61 (K2CTPN= 7)	Coordinate value increment along axis
268	28	R*4(7) K4CRP= 68 (K2CTPN= 7)	Coordinate Reference Pixel
296	28	R*4(7) K4CRT= 75 (K2CTPN= 7)	Coordinate Rotation Angles
324	4	R*4 K4EPO= 82	Epoch of coordinates (years)
328	4	R*4 K4DMX= 83	Real value of data maximum
332	4	R*4 K4DMN= 84	Real value of data minumum
336	4	R*4 K4BLK= 85	Value of indeterminate pixel (real maps only)
340	4	I*2(2) K2GCN=171	Number of random par. groups given as a Pseudo-I*4 number
344	2	I*2 K2PCN=173	Number of random parameters
346	2	I*2 K2DIM=174	Number of coordinate axes
348	14	I*2(7) K2NAX=175 (K2CTPN= 7)	Number of pixels on each axis
362	2	I*2 K2BPX=182	Code for pixel type: 1 integer, 2 real, 3 dbl prec, 4 complex, 5 dbl prec complex
364	2	I*2 K2INH=183	For integer maps: < 0 the value of an indeterminate pixel, > 0 the number of bits used to represent noise est. = 0 no blanking of pixels
366	2	I*2 K2IMS=184	Image sequence no.
368	12	C*12 K4IMN= 93 (K4IMNO= 1)	Image name Character offset in packed string
380	6	C*6 K4IMC= 93 (K4IMCO=13)	Image class Character offset in packed string
386	2	C*2 K4PTY= 93 (K4PTYO=19)	Map physical type (i.e. 'MA','UV') Character offset in packed string
388	2	I*2 K2IMU=195	Image user ID number
390	2	I*2 K2NIT=196	# clean iterations
392	4	R*4 K4BMJ= 99	Beam major axis in degrees
396	4	R*4 K4BMN=100	Beam minor axis in degrees
400	4	R*4 K4BPA=101	Beam position angle in degrees
404	2	I*2 K2TYP=203	Clean map type: 1-4 => normal, components, residual, points
406	2	I*2 K2ALT=204	Velocity reference frame: 1-3 => LSR, Helio, Observer + 256 if radio definition.
408	8	R*8 K8ORA= 52	Antenna pointing Right Ascension
416	8	R*8 K8ODE= 53	Antenna pointing Declination
424	8	R*8 K8RST= 54	Rest frequency of line (Hz)
432	8	R*8 K8ARV= 55	Alternate ref pixel value (frequency or velocity)
440	4	R*4 K4ARP=111	Alternate ref pixel location (frequency or velocity)
444	20	I*2(10) K2EXT=227 (K2EXTN=10)	Names of subsidiary file types (i.e. 'PL') 2 char unpacked form
464	20	I*2(10) K2VER=237 (K2EXTN=10)	Number of versions of corresponding subsidiary file
484	28	I*2(14)	Reserved

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 given above are for standard 16-bit machines. The values are found in the common /HDRVAL/ via includes 'DHDR.INC' and 'CHDR.INC'. The characters of each C*8 variable are packed separately (and left-justified) in as many real variables as required. The image name, class, and physical type are packed as a 20-character string in as many real variables as required.

2. The header contains 256 words and should be contained in the arrays (e.g.) HEAD2(256),HEAD4(128) and HEAD8(64) which are equivalenced Pointers of the type K2... should refer to HEAD2 locations, K4... to HEAD4 locations, and K8... to HEAD8 locations.
E.g. HEAD8(K8BSC) contains the R*8 map scaling factor.

Specific

Byte 16: Any special equipment etc. used during observations
Byte 56: Random parameters are those associated with an irregularly gridded "array". See latest FITS (u-v) paper for details.
Byte 112: Seven coordinates!!! Four will commonly be used; RA, DEC, FREQ and STOKES.
Byte 184: In keeping with the FITS format convention, angles are expressed in degrees.
Byte 324: Somewhat astronomically specific. 1950.0 is used.
Byte 328: 332 The real value of the max/min, not the pixel value is used.
Byte 336: The value used to specify that a pixel is undefined. Used only for floating point maps.
Byte 340: Cannot use an I*4 format in some machines, hence this double I*2 format.
Byte 344: See Byte 56:
Byte 364: Confused at the moment. If negative, then this is the value used as the pixel value for indeterminate pixels. If > 0, then this is the number of bits used to represent the quality of a pixel value. In this case, the true pixel value = (stored / 2 ** #bits) * BSCALE + BZERO. The noise at each pixel is represented by the lsb's, with zero meaning the least reliable.
bits in a map pixel to represent the inherent signal to noise of the data point.
Byte 366: The user mapname-sequence number
Byte 368: The user mapname-name
Byte 380: The user mapname-class
Byte 444: The types of subsidiary files associated with the map are given by a two letter designation; eg.'HI' for history files, 'PL' plot file.
Byte 464: The current highest version number of the associated file type listed in the same relative array position in the previous type listing.
Byte 484: Beginning of 14 spare words.

D. 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.

APPENDIX B

DESCRIPTION OF AIPS UV DATA BASE 25 Feb. 82

This document is to describe the current implementation of a UV data base in AIPS.

The bulk of the UV data is stored as reals in a form which is similar to the FITS random data format. Necessary information about the data is kept in the catalog header record and one or more antenna (AN) files. In particular, the order of the random and uniform parameters is given in the catalog header. This information can be easily obtained via the subroutine UVPGET (see below). Data from one or more arrays may be combined in one data base so the information specific to each array is kept in the antenna (AN) files, one per array. The information in these AN files includes frequency, date of observation, location of array center, location of the antennas etc. The array number is included in the baseline information (see below). The time in each array is offset from the date given in the AN file by 5 days times (array # - 1) to avoid confusion.

Other extension files are history (HI), gain tables (GA) and plot (PL) files. Other types may be added if necessary.

1) Order of the data in a record.

As for FITS, uv data records contain two kinds of data; first are the random parameters (u, v, w, time, etc). The number and order of the random parameters are given in the catalog header. The types currently in use are:

'U '	u spatial frequency coordinate (wavelength at the reference frequency in the catalog header.
'V '	v spatial frequency coordinate.
'W '	w spatial frequency coordinate.
'BASELINE'	ant #2 + 256*(ant #1) + 0.01*(array # - 1)
'TIME1 '	relative time in IAT days + 5.*(array # -1)
'DELAY '	(VLBI) group delay residual used (microsec.)
'RATE '	(VLBI) Fringe rate residual used (mHz)

Other random parameters may be added as needed although there is a limit of 7 which can be named in the catalog header.

Following the random parameters comes the data array. The order, number of values on each axis, reference pixel, reference value at the reference pixel, and the increment are given in the catalogue header as in the case of images. In principle the order is completely arbitrary but in many tasks it is assumed that the 'COMPLEX' axis is first. 'STOKES','FREQ','RA' and 'DEC' axes must be present even if the dimension is 1. The types currently in use are:

'COMPLEX'	Visibility, (Real, Imaginary, Weight)
'STOKES '	Polarization (1=RR, 2=LL, 3=RL, 4=LR)
'FREQ '	Frequency (Hz)
'RA '	Right Ascension (degrees)
'DEC '	Declination (degrees)
'DELAY '	(VLBI) group delay residual
'RATE '	(VLBI) fringe rate residual

In order to facilitate use of the catalogue header subroutine UVPGET is available. This subroutine determines the order of the data and other useful information and places this information in common /UVHDR/. The use of UVPGET is explained in the precursor comments for UVPGET which follow:

```
C-----------------------------------------------------------------------
C   UVPGET determines pointers and other information from a UV CATBLK.
C   The address relative to the start of a vis record for the real part
C   for a given spectral channel (CHAN) and stokes parameter (ICOR)
C   is given by :
C      NRPARM+(CHAN-1)*INCF+(ICOR-ICOR0)*INCS
C   Inputs: From common /MAPHDR/
C      CATBLK(256)    I*2  Catalogue block
C      CAT4           R*4  same as CATBLK
C      CAT8           R*8  same as CATBLK
C   Output: In common /UVHDR/
C      SOURCE(2)      R*4  Packed source name.
C      ILOCU          I*2  Offset from beginning of vis record of U
C      ILOCV          I*2                    "                     V
C      ILOCW          I*2                    "                     W
C      ILOCT          I*2                    "                     Time
C      ILOCB          I*2                    "                   Baseline
C      JLOCC          I*2  Order in data of complex values
C      JLOCS          I*2  Order in data of Stokes' parameters.
C      JLOCF          I*2  Order in data of Frequency.
C      JLOCR          I*2  Order in data of RA
C      JLOCD          I*2  Order in data of dec.
C      INCS           I*2  Increment in data for stokes (see above)
C      INCF           I*2  Increment in data for freq. (see above)
C      ICOR0          I*2  Stokes value of first value.
C      NRPARM         I*2  Number of random parameters
C      LREC           I*2  Length in values of a vis record.
C      NVIS(2)      P I*4  Number of visibilities
C      FREQ           R*8  Frequency (Hz)
C      RA             R*8  Right ascension (1950) deg.
C      DEC            R*8  Declination (1950) deg.
C      NCOR           I*2  Number of correlators
C      ISORT          C*2  Sort order
C      IERR           I*2  Return error code: 0=>OK,
C                             1, 2, 5, 7 : not all normal rand parms
C                             2, 3, 6, 7 : not all normal axes
C                             4, 5, 6, 7 : wrong bytes/value
C-----------------------------------------------------------------------
```

2) UV data specific catalogue header values

The following catalogue header locations (using the notation of the /HDRVAL/ common) have uses specific to uv databases:

CAT4(K4CRT+JLOCD)	Any rotation of UV from the normal RA-Dec relation. >0 => ccw.
CAT8(K8BSC)	Not used.
CAT8(K8BZE)	Not used.
CAT4(K4PTY)	'UV'
CATBLK(K2GCN)	NVIS(1) NVIS = pseudo I*4 # vis. rec.
CATBLK(K2GCN+1)	NVIS(2)
CATBLK(K2TYP)	Sort order (2 char. see below) (sort order '**' indicates unsorted data)

Following is the HELP file explaining the sort order code:

```
SORT
Type:  Adverb (character string, 2 bytes)
Use:   Specify which order data are to be sorted into.  Two
       are packed into SORT with the second key to vary fastest.
       Legal values are:
             B => baseline number
             T => time order
             U => u spatial freq. coordinate
             V => v spatial freq. coordinate
             W => w spatial freq. coordinate
             R => baseline length.
             P => baseline position angle.
             X => descending ABS(u)
             Y => descending ABS(v)
             Z => ascending ABS(u)
             M => ascending ABS(v)
        For example SORT='XY' is required by APMAP.
Tasks:
  UVSRT  Sorts UV data.
```

3) Routines to access uv data bases.

Routines UVINIT and UVDISK are the uv database analogues of MINIT and MDISK used for maps. Their use is described in their precursor comments which follow:

```
      SUBROUTINE UVINIT (OP, LUN, FIND, NVIS, VISOFF, LREC, NPIO,
     *   BUFSZ, BUFFER, BO, BP, BIND, IERR)
C-----------------------------------------------------------------
C   UVINIT sets up bookkeeping for the UV data I/O routine UVDISK.
C   I/O for these routines is double buffered (if possible)
C   quick return I/O.  UVDISK will run much more efficiently if
C   on disk LREC*NPIO*BP is an integral number of blocks.  Otherwise
C   partial writes or oversize reads will have to be done.
C   Minimum disk I/O is one block.  Smaller calls to UVINIT may be
C   as long as the buffer is large enough (double buffer req).
C   The buffer size should include an extra NBPS bytes for each
C   buffer for non tape read if NPIO records does not
C   correspond to an integral number of disk sectors (NBPS bytes).
C   2*NBPS extra bytes required for each buffer for write.
C
C  Inputs:
C  OP         R*4  OP code, 'READ' or 'WRIT' for desired operation.
C  LUN        I*2  Logical unit number of file.
C  FIND       I*2  FTAB pointer for file returned by ZOPEN.
C  NVIS     P I*4  Number of visibilities to be transfered.
C  VISOFF   P I*4  Offset in vis. rec. of first vis. rec. from BO.
C  LREC       I*2  Number of values in a visibility record.
C  NPIO       I*2  Number of visibilities per call to UVDISK.
C                  Determines block size for tape I/O
C  BUFSZ      I*2  Size in bytes of the buffer.
C                  If 32767 given, 32768 is assumed.
C  BUFFER( )  I*2  Buffer
C  BO       P I*4  Block offset to begin transfer from (1-relative)
```

```
C  BP         I*2  Bytes per value in the vis. record.
C
C  Output:
C  NPIO       I*2  For WRITE, the max. number of visibilities
C                  which can be accepted.
C  BIND       I*2  Pointer in BUFFER for WRITE operations.
C  IERR       I*2  Return error code:
C                      0 => OK
C                      1 => file not open in FTAB
C                      2 => invalid input parameter.
C                      3 => I/O error
C                      4 => End of file.
C                      7 => buffer too small
C
C  Note: VISOFF and BO are additive.
C  UVINIT sets and UVDISK uses values in the FTAB:
C
C  FTAB(FIND+0) = LUN
C            1  = # Bytes per I/O
C          2-3  = # vis. records left to transfer.
C                 For double buffer read, 1 more I/O will have
C                 been done than indicated.
C          4-5  = Block offset for next I/O.
C            6  = byte offset of next I/O
C            7  = bytes per value
C            8  = Current buffer #, -1 => single buffering
C            9  = OPcode 1 = read, 2 = write.
C           10  = Values per visibility record.
C           11  = # vis. records per UVDISK call
C           12  = max. # vis. per buffer.
C           13  = # vis. processed in this buffer.
C           14  = Buffer pointer for start of current buffer.(values)
C                 Used for WRIT only; includes any data carried over
C                 from the last write.
C           15  = Buffer pointer for call (values)
C-----------------------------------------------------------------------
      SUBROUTINE UVDISK (OP, LUN, FIND, BUFFER, NIO, BIND, IERR)
C-----------------------------------------------------------------------
C  UVDISK reads and writes records of arbitrary length especially
C  uv visibility data.  Operation is faster if blocks of data
C  are integral numbers of disk blocks.  There are three operations
C  which can be invoked: READ, WRITE and FLUSH (OPcodes READ,
C  WRIT and FLSH).
C
C  READ reads the next sequential block of data as specified to
C  UVINIT and returns the number of visibilities in NIO and
C  the pointer in BUFFER to the first word of this data.
C
C  WRIT arranges data in a buffer until it is full.  Then as
C  many full blocks as possible are written to the disk with the
C  remainder left for the next disk write.  For tape I/O data
C  is always written with the block size specified to UVINIT;
C  one I/O operation per call.  For disk write, left-over data
C  is transfered to the beginning of buffer 1 if that is the
C  next buffer to be filled.  Value of NIO in the call is the
C  number of vis. rec. to be added to the buffer and may be fewer
C  than the number speecified to UVINIT.  On return NIO is the
```

```
C  maximum number which may be sent next time.  On return BIND
C  is the pointer in BUFFER to begin filling new data.
C
C  FLSH writes integral numbers of blocks and moves any data left
C  over to the beginning of buffer 1.  One exception to this is
C  when NIO => -NIO or 0, in which case the entire remaining data
C  in the buffer is written.
C  After the call BIND is the pointer in BUFFER for new data.
C  The principal difference between FLSH and WRIT is that
C  FLSH always forces an I/O transfer.  This may cause trouble
C  if a transfer of less than 1 block is requested.  A call
C  with a nonpositive value of NIO should be the last call and
C  corresponds to a call to MDISK with opcode 'FINI'.
C
C  NOTE:  A call to UVINIT is REQUIRED prior to calling UVDISK.
C
C  Inputs:
C  OP         R*4  Opcode 'READ','WRIT','FLSH' are legal
C  LUN        I*2  Logical unit number
C  FIND       I*2  FTAB pointer returned by ZOPEN
C  BUFFFER()  I*2  Buffer for I/O
C  NIO        I*2  No. additional visibilities to write.
C
C  Output:
C  NIO        I*2  No. visibilities read.
C                  Max. no. vis. for next write.
C  IERR       I*2  Return error code.
C                  0 => OK
C                  1 => file not open in FTAB
C                  2 => input error
C                  3 => I/O error
C                  4 => end of file
C                  7 => attempt to write more vis than specified
C                       to UVINIT or will fit in buffer.
C
C-----------------------------------------------------------------
```

APPENDIX C

Antenna files ANdssvv

A. Overview

Function: This extension file for a uv data set contains information about the antennas and the array geometry including conversion from atomic time to sidereal time.

Details: AN files use the EXTINI-EXTIO file structure with various global parameters kept in the header record. The logical record length is 64 R*4 words.

Names: The file name is ANdsssvv where d is disk number, sss=catalog number and vv = version number.

B. File structure.

The file header record contains the following:

Location I*2	Location R*4	Location R*8	Size	Description
57	29	15	R*8	Array center X coord. (meters, earth center)
61	31	16	R*8	Array center Y coord.
65	33	17	R*8	Array center Z coord.
69	35	18	R*8	GST at IAT=0 (degrees)
73	37	19	R*8	Earth rotation rate (deg/IAT day)
77	39	20	R*8	Frequency (Hz)
81	41	21	C*4(2)	Reference date as 'DD/MM/YY'
85	43		R*4	Polar X (meters)
87	44		R*4	Polar Y (meters)
89	45		R*4	UT1-UTC (time sec.)
91	46		R*4	IAT-UTC (time sec.)
93	47		C*4	Sideband 'UP','LOW','DOUB'

Logical record structure.

Location I*2	Location R*4	Location R*8	Size	Name	Description
1	1	1	C*4	STANAM(2)	8 char. station name
5	3	2	R*8	STABX	X offset from array center (meter)
9	5	3	R*8	STABY	Y offset from array center (meter)
13	7	4	R*8	STABZ	Z offset from array center
17	9	5	I*2	NOSTA	Station number
18			I*2	MNTSTA	Mount type, 0=altaz, 1=equit.
19	10		R*4	STAXOF	Axis offset (meters)
21	11	6	R*4	ANTSPl(18)	Spare words
57	29	15	R*8	CLKIFA	IF A clock offset
61	31	16	R*8	LOIFA	IF A lo-offset
65	33	17	R*4	BPFRA	IF A effective bandpass (fraction)
67	34		C*4	POLTYA	IF A feed poln. type 'R','L','X','Y'
69	35	18	R*4	POLAA	IF A feed position angle.
71	36		R*4	AMPA	IF A JY/K

73	37	19	R*4	POLA1	IF A poln. cal. parameter
75	38		R*4	POLA2	"
77	39	20	R*4	POLA3	"
79	40		R*4	ANTSP2(7)	Spare words
93	47	24	R*8	CLKIFB	IF B clock (sec)
97	49	25	R*8	LOIFB	IF B lo offset (Hz)
101	51	26	R*4	BPFRB	IF B effective bandpass (fractional)
103	52		C*4	POLTYB	IF B poln. type. (see above)
105	53	27	R*4	POLAB	IF B feed position (deg)
107	54		R*4	AMPB	IF B JY/K
109	55	28	R*4	POLB1	IF B poln. cal. parameter
111	56		R*4	POLB2	"
113	57	29	R*4	POLB3	"
115	58		R*4	ANTSP3(7)	Spare words

C. User notes.

When calling EXTINI make sure the calling routine has the common /DCHCOM/ (includes DDCH.INC and CDCH.INC) and use BP = 2 and
LREC = 2 + 7 * NWDPDP + 49 * NWDPFP.

For ease in accessing values in a logical record use the includes DANT.INC and CANT.INC which declare a common /ANTCOM/ in which the names given in section B are in the appropriate location.
When STANAM is given to EXTIO as the location to put the record, the values will be correctly filled in.

D. Routines to write AN files:
EXTINI and EXTIO, AN files are currently written by UVLOD and TOAIP.

E. Routines to access AN files: EXTINI and EXTIO.