Chapter 13

Tables in AIPS

13.1 Overview

This chapter is an attempt to describe the use of AIPS tables extension files and to describe the structure of these files. The next section describes general tables utility routines followed by routines which simplify the access to specific types of AIPS tables. The final section describes the structure of the tables files and the fundamental routines to access AIPS tables.

Table files consist of an extensive and rather flexible header and a table organized as rows and columns. An entry in a column may be either a single value or an array of values with the same data type. Each column has a specified format and is stored in the appropriate binary form for the local computer. The columns are ordered on disk in an order appropriate to computer addressing, but are accessed in any desired logical column order via a lookup list.

The extension file contains not only the rows and columns, but also a variety of other information. Each column has an associated 24-character column "title" and an 8-character "units" field. Each row has a "selection" flag which allows the user to access temporarily a subset of his table. The strings used to specify the current selection are stored in the file for display. The file may also contain general information applying to the full table in the form of keyword/value pairs. This information will be called the table "header" data.

13.2 General Tables Routines

There are a number of utility routines which perform operations on AIPS tables. Hopefully there will be many more of these as the use of tables in AIPS increases. The following list gives a short description of these routines; details of the call sequences are given at the end of this chapter. Also of interest to the programmer is the AIPS task PRTAB for printing the contents of a table file.

1. ALLTAB copies all tables, allows a list of exclusions.
2. ISTAB determines if an extension file type exists and if it is a table.
3. TABCOP copies the entire contents of one or more tables of a given type.
4. TABKEY reads or writes keyword/value pairs to a table header.
5. TABMRG merges or adds adjacent, like rows in a table.
6. TABSRT sorts the rows in a table file using up to 4 keys.

13.3 Specific Tables Routines

Because of the generality of the tables routines, the low-level use of tables is rather cumbersome. For this reason, there are a number of specialized routines which simplify the access to a given type of table. In general, these routines come in pairs - one to create/initialize the I/O and the other to read or write to the file. If there are keyword/value pairs associated with a given table type, they are processed by the initialization routine. These specialized routines usually return the contents of a row into properly named variables, which avoids the use of equivalencing in the calling routine.

Since many of the tables in AIPS are subject to modification, especially those associated with the calibration package; the number of columns in each table is likely to change. For this reason it is strongly recommended that the number of columns in these tables be taken from the PARAMETER include PUVD.INC especially to be used for declaring array sizes. Comments in the text of this include file, which is shown at the end of this chapter, tell which PARAMETER belongs to which table type.

The specific tables routines are briefly described in the following list; details of the call sequences are given at the end of this chapter.

1. ANTINI and TABAN access AN (Antenna) tables.
2. BLINI and TABBL access BL (baseline dependent calibration) tables.
3. BPINI and TABBP access BP (bandpass) tables.
4. CALINI and TABCAL access calibration (CL) tables.
5. CCINI creates/initializes CC (CLEAN component or gaussian model files).
6. CCMERG sums CC components at the same position including the two needed sorts.
7. CHNCOP copies selected portions of a Frequency (FQ) table.
8. CHNDAT reads/writes/creates the contents of FQ (Frequency descriptor) tables.
9. FLGINI and TABFLG access FG (Flag) tables.
10. FQINI and TABFQ access FQ (FreQuency) tables.
11. GETNAN determines the number of subarrays and the numbers of antennas in each from the AN table(s).
12. NDXINI and TABNDX access NX (Index) tables.
13. SNINI and TABSN access solution (SN) tables.
14. SOUINI and TABSOU access SU (Source) tables.

13.4 The Format Details

There are several distinct types of information kept in a table file. Most important is the data tabulated, referred to as “row data”. Associated with each column is label information; this includes a label, units, element count, and data type. There is also a provision for storing general information about the file in the form of keyword/value pairs. A keyword/value pair consists of a string of characters (Keyword) which gives a label to a value (Value) which may be any of a number of data types.

13.4.1 Row Data

The row data are stored as an integer number of rows per disk record (256 integers) or as an integer number of disk records per row. The columns are given a physical order appropriate to addressing on all computers. The logical order is carried in the file header record (physical record 1, see below) and in a set of array indices for addressing by the programs. The type of data is specified by code numbers. These codes and the physical ordering are as follows:

```
    ORDER                    ARRAY   BASIC CODE  +  LENGTH
double-precision floating    D           1       +  10 * LENGTH
single-precision floating    R           2       +  10 * LENGTH
hollerith (4 char / word)    H           3       +  10 * LENGTH
integer                      I           4       +  10 * LENGTH
logical                      L           5       +  10 * LENGTH
bit (NBITWD / integer)       I           7       +  10 * LENGTH
select flag                  I           9           -
```

```
Declarations:
      INTEGER   I(*)
      LOGICAL   L(*)
      HOLLERITH H(*)
      REAL      R(*)
      DOUBLE PRECISION D(*)
      EQUIVALENCE (I, L, H, R, D)
```

13.4.2 Physical File Format

The data, control, and header information are written in the Table file via ZFIO in 512-byte (256-integer) blocks. The order on disk, by physical record number, is:

```
record   1     : Control information and lookup table (see later)
         2     : DATPTR(128) subscript of the appropriate array for
                       logical column n
                 DATYPE(128) type code for logical column n
         3 - 4 : Selection strings now in force
         5 - m : Titles (24 HOLLERITH) in physical column order
       m+1 - i : Units (8 HOLLERITH) in physical column order
       i+1 - k : Table header (keyword/value pairs, see below)
       k+1 - * : Row data in n rows/record or n records/row
where
       m =     5 +  (NCOL-1) / (256 / 6)
       i = m + 1 +  (NCOL-1) / (256 / 2)
       k = i + 1 +  (NKEY-1) / (256 / 5)
       NCOL = number logical columns not including the select column
       NKEY = maximum number of keyword/value pairs
```

13.4.3 Control Information

Physical record one contains file control data needed to do the I/O operations and maintain the physical file. It is prepared by subroutine TABINI and modified by TABIO. The latter subroutine returns the record to disk on OPCODE = 'CLOS'. Its contents are:

```
  1         Number 512-byte records now in file
  2
  3         Max number of rows allowed in current file
  4
  5         Number of rows (logical records) now in file
  6
  7
  8         # values/logical (# words/row incl. selection flag)
  9         > 0 => number rows / physical record
```

```
             < 0 => number physical records / row
 10          Number logical columns/row (not including selection
             column)
 11 - 16     Creation date: ZDATE(11), ZTIME(14)
 17 - 28  H  Physical file name (set on each TABINI call)
 29 - 30  H  Creation task name
 31
 32          Disk number
 33 - 38     Last write access date: ZDATE(33), ZTIME(36)
 39 - 40  H  Last write access task name
 41
 42          Number logical records to extend file if needed
 43          Sort order: logical column # of primary sorting
 44          Sort order: logical column # of secondary sorting
                  0 => unknown, < 0 => descending order
 45          Disk record number for column data pointers (=2)
 46          Disk record number for row selection strings (=3)
 47          Disk record number for 1st record of titles (=5)
 48          Disk record number for 1st record of units
 49          Disk record number for 1st record of keywords
 50          Disk record number for 1st record of table data
 51          DATPTR (row selection column)
 52          Maximum number of keyword/value pairs allowed
 53          Current number of keyword/value pairs in file
 54 - 56  H  "*AIPS TABLE*" string to verify that this is a table
 57 - 59
 60          If 1 then the table CANNOT be written as FITS ASCII
 61          Number of selection strings now in file
 62          Next available address for a selection string
 63          First address of selection string 1
 64          First address of selection string 2
 65          First address of selection string 3
 66          First address of selection string 4
 67          First address of selection string 5
 68          First address of selection string 6
 69          First address of selection string 7
 70          First address of selection string 8
********** for TABIO / TABINI use only **********
 71          IOP : 1 => read, 2 => write
 72          Number words per logical record (incl. select)
 73          Current table row physical record in BUFFER
 74
 75          Current table row logical record in BUFFER
 76
 77          Type of current record in BUFFER (0 - 5)
 78          Current control physical record number in BUFFER
 79          Current control logical record number in BUFFER
 80          Type of current control record in BUFFER
 81          File logical unit number (LUN)
 82          FTAB pointer for open file (IND)
***********
 83 -100         Reserved
***********
101 -128  H  Table title.
```

```
129 -256     lookup table as COLPTR (logical column) = physical column
```

13.4.4 Keyword/value records

The keyword/value pairs are stored in 5 single-precision floating locations, 256 / 5 per physical record. The keyword is an 8-character string stored as a HOLLERITH. It is left justified. The value is stored left justified in the 3rd and 4th reals using as many words as needed (see table below). The keyword type is given in a integer following the 4th word.

The keyword data type codes which specify the type of the binary value are:

```
1    double-precision floating point
2    single-precision floating point
3    8-character HOLLERITH string  (4 chars / element)
4    integer
5    logical
```

In the call sequence to TABIO, the variable RECORD is an integer array used to convey the data to the I/O operations. For keyword/value pairs, RECORD is divided as follows:

```
RECORD(1)  1st 4 chars of the keyword
RECORD(2)  2nd 4 chars of keyword
RECORD(3)  value
RECORD(5)  type code
```

```
where the value occupies the following number of integer words:
         type 1    NWDPDP
              2    1
              3    2
              4    1
              5    1
```

13.4.5 I/O buffers

The call to TABINI specifies two buffers, one for I/O scratch and control and the other for the data pointers which will be used by the calling program to access the column data. The first, called BUFFER, is used as

```
BUFFER(  1)-BUFFER(128) control pointers
BUFFER(129)-BUFFER(256) lookup table
BUFFER(257)-BUFFER(***) current physical record(s) of table data
                        where *** = 512 if there are >= 1 rows/rec,
                        *** = (n+1)*256 if there are n recs/row.
```

The call sequence of TABINI has an argument, NBUF, which gives the length of BUFFER. This is used solely to check that BUFFER is large enough to handle the present table file. BUFFER is also provided by the programmer to TABIO, which will modify the control and data portions. The programmer should not modify BUFFER between the call to TABINI and the call to TABIO with OPCODE 'CLOS', except to insert a title for the table in words 101 - 128, or to correct the sort order information.

The second buffer, called DATP, is used by the non-I/O portions of the table package. DATP(1,1) - DATP(128,1) contains the subscript of the appropriate array for the logical columns. DATP(1,2) - DATP(128,2) contains the data type / element count for each logical column. When TABINI is to create the table extension file, the programmer must fill in DATP(1,2) - DATP(NCOL,2) before calling TABINI. A complete set of DATP will be returned by TABINI under all circumstances.

13.4.6 Fundamental Table Access Subroutines

There is a set of basic table handling routines which apply to all tables files. The following list gives a short description; the details of the call sequences and usage are found at the end of this chapter.

1. TABINI creates/opens/catalogs an AIPS table.
2. TABIO does I/O to a tables file. Row data, keyword/value pairs and control information are passed through this subroutine.
3. GETCOL returns the value and value type at a specified row and column from an open table.
4. FNDCOL locates the logical column number for a column with a specified label.
5. PUTCOL puts a specified the value into a specified row and column from an open table.

13.4.7 Table Reformating Subroutines

Several of the tables used with the calibration package are subject to change as new things are added. In order to minimize the confusion there are a number of routines which check if the table uses the current format and if not reformats the table adding any new columns. A short description of these routines is given here and a more complete description is given at the end of this chapter.

1. BLREFM reformats a BL (BaseLine dependent calibration) table.
2. BPREFM reformats a BP (BandPass calibration) table.
3. CLREFM reformats a CL (CaLibration) table.
4. SNREFM reformats a SN (calibration SolutioN) table.

13.5 Includes

13.5.1 PUVD.INC

A number of generally useful parameters including the number of columns in each of the calibration tables is given in this include as PARAMETERS.

```
C                                                          Include PUVD
C                                       Parameters for uv data
      INTEGER   MAXANT,  MXBASE,  MAXIF, MAXFLG, MAXFLD, MAXCHA
C                                       MAXANT = Max. no. antennas.
      PARAMETER (MAXANT=45)
C                                       MXBASE = max. no. baselines
      PARAMETER (MXBASE= ((MAXANT*(MAXANT+1))/2))
C                                       MAXIF=max. no. IFs.
      PARAMETER (MAXIF=15)
C                                       MAXFLG= max. no. flags active
      PARAMETER (MAXFLG=1000)
C                                       MAXFLD=max. no fields
      PARAMETER (MAXFLD=16)
C                                       MAXCHA=max. no. freq. channels.
      PARAMETER (MAXCHA=512)
C                                       Parameters for tables
      INTEGER MAXCLC, MAXSNC, MAXANC, MAXFGC, MAXNXC, MAXSUC,
     *   MAXBPC, MAXBLC, MAXFQC
C                                       MAXCLC=max no. cols in CL table
      PARAMETER (MAXCLC=41)
```

```
C                                       MAXSNC=max no. cols in SN table
      PARAMETER (MAXSNC=20)
C                                       MAXANC=max no. cols in AN table
      PARAMETER (MAXANC=12)
C                                       MAXFGC=max no. cols in FG table
      PARAMETER (MAXFGC=8)
C                                       MAXNXC=max no. cols in NX table
      PARAMETER (MAXNXC=7)
C                                       MAXSUC=max no. cols in SU table
      PARAMETER (MAXSUC=21)
C                                       MAXBPC=max no. cols in BP table
      PARAMETER (MAXBPC=14)
C                                       MAXBLC=max no. cols in BL table
      PARAMETER (MAXBLC=14)
C                                       MAXFQC=max no. cols in FQ table
      PARAMETER (MAXFQC=5)
C                                                          End PUVD.
```

13.6 Routines

Following are the descriptions of the call sequences and usage notes for the routines discussed in this chapter.

13.6.1 Routines Applying to All Tables

ALLTAB

Copies all Table extension file(s). The output files must be new — old ones cannot be rewritten. The output file must be opened WRIT in the catalog and will have its CATBLK updated on disk.

```
ALLTAB (NONOT, NOTTYP, LUNOLD, LUNNEW, VOLOLD, VOLNEW, CNOOLD,
  *   CNONEW, CATNEW, BUFF1, BUFF2, IRET)
Inputs:
   NONOT      I     Number of "Forbidden" types to copy.
   NOTTYP(*)  C*2   Table types to ignore (2 char meaningful, blank
                    filled)
   LUNOLD     I     LUN for old file
   LUNNEW     I     LUN for new file
   VOLOLD     I     Disk number for old file.
   VOLNEW     I     Disk number for new file.
   CNOOLD     I     Catalog slot number for old file
   CNONEW     I     Catalog slot number for new file
In/out:
   CATNEW(256)I     Catalog header for new file.
Output:
   BUFF1(1024) I     Work buffer
   BUFF2(1024) I     Work buffer
   IRET        I     Return error code  0 => ok, otherwise TABCOP
                                       or 10*CATIO error.
```

FNDCOL

Is used with AIPS Table extension files. It locates the logical column number(s) which are titled with specified strings.

```
FNDCOL (NKEY, KEYS, LKEY, LORDER, BUFFER, KOLS, IERR)
Inputs:
   NKEY    I            Number columns to be found
   KEYS    C*24(NKEY)   Column titles to locate
   LKEY    I            Number characters to check in each of KEYS
                        (legal values 1 through 24)
   LORDER  L            T => logical order desired, else phys.
In/out:
   BUFFER  I(>512)      TABINI/TABIO buffer/ header/ work area
Output:
   KOLS    I(NKEY)      Logical column numbers: 0 => none,
                              -1 => more than one (!)
   IERR    I            Error code: 0 => ok, 1 - 10 from ZFIO
                             >10 = 10 + # of failed columns
```

GETCOL

Returns the value and value type found in an open table file at the specified logical column and row.

```
GETCOL (IRNO, ICOL, DATP, BUFFER, RTYPE, RESULT, SCRTCH, IERR)
Inputs:
   IRNO    I         Table row number: n.b. I
   ICOL    I         Table column number
   DATP    I(256)    Pointer array returned by TABINI
In/out:
   BUFFER  I(*)      Control area set up by TABINI, used in TABIO
Output:
   RTYPE   I         10 * length + Type of column:
                     1 -> D,  2 -> R,  4 -> I,  5 -> L*?,  6 -> I.
                     Characters hollerith, bits packed.
   RESULT  ?(*)      Value of column: use D, R, H, I equivalenced
                     arrays, note: may be an array; if length = 0,
                     then no value and RESULT is unchanged.
   SCRTCH  I(*)      Scratch large enough to hold a row
   IERR    I         Error code: 0 => OK.
                       -1 => OK, but row is flagged
                        1 file not open,  2 input error
                        3 I/O error,      4 read past EOF
                        5 bad data type
```

ISTAB

Given an extension type, volume, version, determine if the extension file exists, and if so, is it a standard table.

```
ISTAB (ETYPE, IVOL, ISLOT, IVER, LUN, BUFFER, TABLE, EXIST, FITASC,
  *   IERR)
Inputs:
  ETYPE  C*2  Extension type like 'CC'.
  IVOL   I    Disk Volume.
  ISLOT  I    Map catalog slot number.
  LUN    I    AIPS LUN to use for opening and reading a test block.
In/Out:
  TABLE  L    True if this extension type is a table or this
```

```
                 version does not exist.
     EXIST  L    True if the extension file exists, else false.
     FITASC L    True if the file can be written as a FITS ASCII table.
  Output:
     IERR   I    Error code, 0=>OK else failed
```

PUTCOL

Enters the value and value type into an open table file at the specified logical column and row.

```
PUTCOL (IRNO, ICOL, DATP, BUFFER, VALUE, SCRTCH, IERR)
Inputs:
   IRNO    I         Table row number
   ICOL    I         Table column number
   DATP    I(256)    Pointer array returned by TABINI
   VALUE   ?(*)      Value of column: use D, R, H, I equivalenced
                     arrays, note: may be an array; if length = 0,
                     then no value and VALUE is unchanged.
In/out:
   BUFFER  I(*)      Control area set up by TABINI, used in TABIO
Output:
   SCRTCH  I(*)      Scratch large enough to hold a row
   IERR    I         Error code: 0 => OK.
                       -1 => OK, but row is flagged
                        1 file not open,  2 input error
                        3 I/O error,      4 read past EOF
                        5 bad data type
```

TABCOP

Copies Table extension file(s). The output file must be a new extension - old ones cannot be rewritten. The output file must be opened WRIT in the catalog and will have its CATBLK updated on disk.

```
TABCOP (TYPE, INVER, OUTVER, LUNOLD, LUNNEW, VOLOLD, VOLNEW,
  *   CNOOLD, CNONEW, CATNEW, BUFF1, BUFF2, IRET)
Inputs:
   TYPE     C*2     Extension file type (e.g. 'CC','AN')
   INVER    I       Version number to copy, 0 => copy all.
   OUTVER   I       Version number on output file, if more than one
                    copied (INVER=0) this will be the number of the
                    first file.  If OUTVER = 0, it will be taken as
                    1 higher than the previous highest version.
   LUNOLD   I       LUN for old file
   LUNNEW   I       LUN for new file
   VOLOLD   I       Disk number for old file.
   VOLNEW   I       Disk number for new file.
   CNOOLD   I       Catalog slot number for old file
   CNONEW   I       Catalog slot number for new file
In/out:
   CATNEW   I(256)  Catalog header for new file.
Output:
   BUFF1    I(256)  Work buffer
   BUFF2    I(256)  Work buffer
   IRET     I       Return error code  0 => ok
```

```
                              1 => files the same, no copy.
                              2 => no input files exist
                              3 => failed
                              4 => no output files created.
                              5 => failed to update CATNEW
                              6 => output file exists
```

TABINI

Creates/opens a table extension file. If a file is created, it is cataloged by a call to CATIO which saves the updated CATBLK.

```
TABINI (OPCODE, PTYP, VOL, CNO, VER, CATBLK, LUN, NKEY,
  *   NREC, NCOL, DATP, NBUF, BUFFER, IERR)
Input:
   OPCODE   C*4       Operation code, 'READ' => read only,
                                      'WRIT' => read/write
   PTYP     I         Physical extension type (eg. 'CC')
   VOL      I         Disk volume number
   CNO      I         Catalog slot number
   CATBLK   I(256)    Catalog block of cataloged file.
   LUN      I         Logical unit number to use.
   NREC     I         Number of logical rec. for create/extend
   NBUF     I         Number I   words in BUFFER
In/out:
   VER      I         Version number: (<= 0 => write a new one,
                      read the latest one), returns one used.
   NKEY     I         Maximum number of keyword/value pairs
                      input: used in create, checked on write old
                      (0 => any); output: actual
   NCOL     I         Number of logical columns (does not include
                      selection column).  Input: used in create,
                      checked on write old (0=>any); output: actual
   DATP     I(128,2)  DATP(*,1) address pointers (output only)
                      DATP(*,2) column data type codes. Input:
                      used in create only; output: actual.
   BUFFER   I(*)      Work buffer, at least 1024 bytes in size,
                      more if logical record longer than 512 bytes
                      Output: control info, lookup table, ...
Output:
   IERR     I         Return error code. 0 => OK
                                     -1 => OK, created new file
                                      1 => bad input.
                                      2 => could not find or open
                                      3 => I/O problem.
                                      4 => create problem.
                                      5 => not a table file
Usage notes:
For sequential access, TABINI leaves pointers for TABIO such that,
if IRNO <= 0, reads will begin at the start of the file and writes
will begin after the last previous record.  Cataloged file should
be marked 'WRIT' if the file is to be created.
```

TABIO

Does random access I/O to Tables extension files. Mixed reads and writes are allowed if TABINI was called 'WRIT'. Writes are limited by the size of the structure (i.e. no. columns for units and titles) or to the current maximum logical record plus one. Files opened for WRITe are updated and compressed on CLOS.

```
TABIO (OPCODE, IRCODE, IRNO, RECORD, BUFFER, IERR)
Inputs:
   OPCODE   C*4      Opcode 'READ','CLOS'
                        'WRIT' : write data as selected
                        'FLAG' : write data as de-selected
   IRCODE   I        Type of information
                        0 => Table row
                        1 => DATPTR/DATYPE record
                        2 => data selection string
                        3 => title
                        4 => units
                        5 => keyword/value pair
   IRNO     I        Logical record number. 0 => next (can work
                     with row data and latest IRCODE > 0 only)
                        IRNO is row number (IRCODE = 0)
                        IRNO is ignored (IRCODE = 1)
                        IRNO is string number (IRCODE = 2)
                        IRNO is column number (IRCODE = 3)
                        IRNO is column number (IRCODE = 4)
                        IRNO is keyword number (IRCODE = 5)
   RECORD   I(*)     Array containing record to be written
   BUFFER   I(*)     Work buffer = 512 bytes + enough 512 byte
                     blocks for at least one full logical record.
                     Must be the same one given TABINI.
Output:
   RECORD   I(*)     Array containing record read.
   BUFFER   I(*)     buffer.
   IERR     I        Return error code 0 => OK
                         -1 => on READ: row read is flagged
                          1 => file not open
                          2 => input error
                          3 => I/O error
                          4 => attempt to read past end of data
                               or write past end of data + 1
                          5 => error on expanding the file
IMPORTANT NOTE:  the contents of BUFFER should not be changed
except by TABIO between the time TABINI is called until the file
is closed.
```

TABKEY

Reads or writes KEYWORDs from or to an AIPS table file header. The order of the keywords is arbitrary. Table file must have been previously opened with TABINI.

```
TABKEY (OPCODE, KEYWRD, NUMKEY, BUFFER, LOCS, VALUES, KEYTYP, IERR)
Inputs:
   OPCODE   C*4        Operation desired, 'READ', 'WRIT',
                          'ALL ' => Read all.
   KEYWRD   C(*)*8     Keywords to read/write
```

```
   BUFFER   I(*)        Buffer being use for table I/O >= 512 words.
In/out:
   NUMKEY   I           Number of keywords to read/write.
                           Input on OPCODE='ALL' = max. to read.
                           Output on OPCODE='ALL' = no. read.
   LOCS     I(NUMKEY)   The word offset of first integer
                        word of keyword value in array VALUES.
                        Output on READ, input on WRIT.
                        On READ this value will be -1 for keywords
                        not found.
  VALUES    I           The array of keyword values; due to word
                        alignment problems on some machines values
                        longer than a integer should be copied,
                        eg. if the  5th keyword (XXX) is a R:
                             IPOINT = LOCS(5)
                             CALL COPY (1, VALUES(IPOINT), XXX)
                        Output on READ, input on WRIT
  KEYTYP    I(NUMKEY)   The type code of the keywords:
                           1 = Double precision floating
                           2 = Single precision floating
                           3 = Character string (8 HOLLERITH chars)
                           4 = integer
                           5 = Logical
Output:
   IERR     I           Return code, 0=>OK,
                           1-10 =>TABIO error
                           19   => unrecognized data type.
                           20   => bad OPCODE
                           20+n => n keywords not found on READ.
```

TABMRG

Merges an AIPS general table-format file. It does not sort the file - so sorts must be done first if merging is to make any sense. TABMRG compares row N with row N+1 and merges the two, summing columns SKOL and using row N for the others, if each of columns EKOL are within TKOL of each other.

```
TABMRG (DISK, CNO, TYPE, INVER, OUTVER, EKOL, SKOL, TKOL, ITABL,
  *   INBUF, OTABL, ONBUF, CATBLK, OUTNUM, IERR)
Inputs:
   DISK    I         Disk number to use
   CNO     I         Primary file catalog number
   TYPE    C*2       Extension file type (e.g. CC)
   INVER   I         Input extension file number (0 -> high)
   EKOL    I(*)      Columns which must be "equal" by logical
                     column number.  A zero MUST terminate the
                     list unless it is 25 long.  Must have >0.
   SKOL    I(*)      Columns which will be summed on a merge by
                     column number.  A zero MUST terminate the
                     list unless it is 25 long.  May be 0.
   TKOL    R(*)      Tolerance for equality by logical column
                     number (parallel to EKOL list)
   INBUF   I         Number I   words in ITABL
   ONBUF   I         Number I   words in OTABL
In/out:
```

```
   OUTVER  I         Output extension file number (0 -> high+1)
   CATBLK  I(256)    Primary file header
Output:
   ITABL   I(>512)   Scratch for input table IO
   OTABL   I(>512)   Scratch for output table IO
   OUTNUM  I         Number rows in output file.
   IERR    I         Error code: 0 -> ok
                        1 -> input file open error
                        2 -> input parameter error
                        3 -> output file create/open error
                        4 -> ZFIO IO error
                        5 -> TABIO IO error
```

TABSRT

Subroutine to sort an AIPS table extension file. First key changes the most slowly. A linear combination of two columns or a substring of a bit or character string may be used. The columns and factors are specified in KEY and FKEY, the first (slower varying key) is:

```
KEY_VALUE1 = COL_VALUE(KEY(1,1)) * FKEY(1,1) +
             COL_VALUE(KEY(2,1)) * FKEY(2,1)
```

The faster changing key value is:

```
KEY_VALUE2 = COL_VALUE(KEY(1,2)) * FKEY(1,2) +
             COL_VALUE(KEY(2,2)) * FKEY(2,2)
```

In the case of bit or character strings only one column is used to generate the key values.

```
A KEY(m,n) < 0 => use ABS (COL_VALUE(-KEY(m,n))).

   TABSRT (DISK, CNO, TYPE, INVER, OUTVER, KEY, FKEY, BUFFER, BUFSZ,
     *   TABUFF, NBUF, CATBLK, IERR)
   Inputs:
      DISK         I    Disk number of the file.
      CNO          I    Catalog slot number.
      TYPE         I    Two character type code (e.g. 'CC')
      INVER        I    Input version number
      OUTVER       I    Output version number
      KEY(2,2)     I    Sort keys: may be linear combination of two
                        numeric value columns.  KEY contains the column
                        numbers and FKEY contains the factors.  If the
                        column is a string (bit or char.) then
                        FKEY(1,n)=first char/bit and FKEY(2,n)=number
                        of char/bit and KEY(2,n) is ignored.
                        KEY(2,n)=0 => ignore, <0 => use abs. value.
                        Column no. is the logical number.
      FKEY(2,2)    R    Key  coefficients, 0=>1, see above.
      BUFSZ        I    Size of BUFFER in bytes.
      NBUF         I    Size of TABUFF in (I)   words.
      CATBLK(256)  I    Catalog header record.
   Output:
      TABUFF(*)    I    Buffer large enough to handle I/O to table.
      BUFFER(*)    R    I/O work buffer
      IERR         I    Error code, 0 => OK, else error.
```

```
                     10 => Couldn't find or open file.
Useage Notes:
     Normally the keys are sorted into ascending order, to sort into
descending order negate the values of FKEYn.
     TWO standard scratch files will be created and entered into the
/CFILES/ common.  These scratch files will be deleted on normal
termination.  Include DFIL.INC should be included in the main
routine and a call made to DIE rather than DIETSK should
be made at the end of the program execution.  The values in BADD
(adverb BADDISK) in the /CFILES/ common should be initialized.
     IF a disk based sort is required, then a 4-way merge sort will
be used.
     Since keys are converted into floating point numbers some
accuracy may be lost sorting on character or bit strings.
     For a 1 key sort use KEY2(1) = 0.
NB: This routine will modify the contents of common /MAPHDR/
(DCAT.INC)
```

13.6.2 Routines Applying to Specific Tables

ANTINI

Creates and initializes antenna (AN) tables.

```
ANTINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   IANRNO, ANKOLS, ANNUMV, ARRAYC, GSTIA0, DEGPDY, SAFREQ, RDATE,
  *   POLRXY, UT1UTC, IATUTC, ANAME, NUMORB, NOPCAL, IERR)
Inputs:
   OPCODE          C*4  Operation code:
                        'WRIT' = create/init for write or read
                        'READ' = open for read only
   BUFFER(512)     I    I/O buffer and related storage, also defines
                        file if open.
   DISK            I    Disk to use.
   CNO             I    Catalog slot number
   VER             I    GA file version
   CATBLK(256)     I    Catalog header block.
   LUN             I    Logical unit number to use
Input/output (file keywords):
   ARRAYC(3)       D    Array center X coord. (meters, earth center)
   GSTIA0          D    GST at IAT=0 (degrees) on ref. date
   DEGPDY          D    Earth rotation rate (deg/IAT day)
   SAFREQ          D    Obs. Reference Frequency for subarray(Hz)
   RDATE           C*8  Reference date as 'DD/MM/YY'
   POLRXY(2)       R    Polar position X,Y (meters) on ref. date
   UT1UTC          R    UT1-UTC  (time sec.)          "
   IATUTC          R    IAT-UTC  (time sec.)          "
   ANAME           C*8  Array name
   NUMORB          I    Number of orbital parameters
   NOPCAL          I    Number of polarization calibration constants.
   ANNUMV(MAXANC)  I    Element count in each column. On input only
                        used if the file is created.
Output:
   IANRNO          I    Next scan number, start of the file if READ,
                        the last+1 if WRITE
```

```
  ANKOLS(MAXANC)  I    The column pointer array in order, ANNAME,
                       STABXYZ, ORBPARM, NOSTA, MNTSTA, STAXOF,
                       POLTYA, POLAA, POLCALA, POLTYB, POLAB, POLCALB
  IERR            I    Return error code, 0=>OK, else TABINI or TABIO
                       error.
Useage NOTE: use the include 'DANT.INC' for the declarations in
ANTINI and TABAN.
```

BLINI

Creates and initializes baseline correction (BL) extension tables.

```
BLINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   IBLRNO, BLKOLS, BLNUMV, NUMANT, NUMPOL, NUMIF, IERR)
 Inputs:
  OPCODE       C*4 Operation code:
                   'WRIT' = create/init for write or read
                   'READ' = open for read only
  BUFFER(1024) I   I/O buffer and related storage, also defines file
                   if open.
  DISK         I   Disk to use.
  CNO          I   Catalog slot number
  VER          I   BL file version
  CATBLK(256)  I   Catalog header block.
  LUN          I   Logical unit number to use
 Input/output
  NUMANT       I    Number of antennas
  NUMPOL       I    Number of polarizations.
  NUMIF        I    Number of IFs
 Output:
  IBLRNO       I   Next scan number, start of the file if 'READ',
                   the last+1 if WRITE
  BLKOLS(MAXBLC) I   The column pointer array in order, TIME, SOURID,
                   SUBARRAY, ANTENNA1, ANTENNA2, FREQ. ID,
                   REALM1, IMAGM1, REALA1, IMAGA1,
                   Following used if 2 polarizations per IF
                   REALM2, IMAGM2, REALA2, IMAGA2.
  BLNUMV(MAXBLC) I   Element count in each column.
  IERR         I   Return error code, 0=>OK, else TABINI or TABIO
                   error.
```

BLREFM

Routine to change the format of the BL table from an old format to the current one if necessary.
NOTE: routine uses LUN 45 as a temporary logical unit number.

```
BLREFM (DISK, CNO, VER, CATBLK, LUN, IRET)
Inputs:
   DISK           I       Volume number
   CNO            I       Catalogue number
   VER            I       Version to check/modify
   CATBLK(256)    I       Catalogue header
   LUN            I       LUN to use
Output:
```

```
  IRET            I       Error, 0 => OK
Note, routine will leave no trace of its operation, i.e. BL table
will be closed on output and will have same number as one specified.
Difference will be only that number of columns has changed if that
is required.
```

BPINI

Creates and initializes bandpass (BP) extension tables.

```
BPINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   IBPRNO, BPKOLS, BPNUMV, NUMANT, NUMPOL, NUMIF, NUMFRQ,
  *   BCHAN, IERR)
Inputs:
  OPCODE       C*4 Operation code:
                   'WRIT' = create/init for write or read
                   'READ' = open for read only
  BUFFER(4096) I   I/O buffer and related storage, also defines file
                   if open.
  DISK         I   Disk to use.
  CNO          I   Catalog slot number
  VER          I   BL file version
  CATBLK(256)  I   Catalog header block.
  LUN          I   Logical unit number to use
 Input/output
  NUMANT       I    Number of antennas
  NUMPOL       I    Number of polarizations.
  NUMIF        I    Number of IFs
  NUMFRQ       I    Number of frequency channels
  BCHAN        I    Start channel number
Output:
  IBPRNO       I   Next scan number, start of the file if 'READ',
                   the last+1 if WRITE
  BPKOLS(MAXBPC) I   The column pointer array in order:
                     TIME, INTERVAL, SOURID,
                     SUBARRAY, ANTENNA,
                     BANDW (of individual channel), IFFREQ,
                     FREQ. ID,
                     REFANT1, REAL1, IMAG1,
                     Following used if 2 polarizations per IF
                     REFANT2, REAL2, IMAG2.
  BPNUMV(MAXBPC) I   Element count in each column.
  IERR         I   Return error code, 0=>OK, else TABINI or TABIO
                   error.
```

BPREFM

Routine to change the format of the BP table from an old format to the new one if necessary.
NOTE: routine uses LUN 45 as a temporary logical unit number.

```
BPREFM (DISK, CNO, VER, CATBLK, LUN, IRET)
Inputs:
  DISK            I       Volume number
  CNO             I       Catalogue number
```

```
     VER            I       Version to check/modify
     CATBLK(256)    I       Catalogue header
     LUN            I       LUN to use
  Output:
     IRET           I       Error, 0 => OK

  Note, routine will leave no trace of its operation, i.e. BP table
  will be closed on output and will have same number as one specified.
  Difference will be only that number of columns has changed if that
  is required.
```

CALINI

Creates and initializes calibration (CL) extension tables.

```
  CALINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
    *   ICLRNO, CLKOLS, CLNUMV, NUMANT, NUMPOL, NUMIF, GMMOD, IERR)
  Inputs:
    OPCODE       C*4 Operation code:
                     'WRIT' = create/init for write or read
                     'READ' = open for read only
    BUFFER(512)  I   I/O buffer and related storage, also defines file
                     if open.
    DISK         I   Disk to use.
    CNO          I   Catalog slot number
    VER          I   CL file version
    CATBLK(256)  I   Catalog header block.
    LUN          I   Logical unit number to use
   Input/output
    NUMANT       I    Number of antennas
    NUMPOL       I    Number of IFs per pair
    NUMIF        I    Number of IF pairs
    GMMOD        R    Mean gain modulus
   Output:
    ICLRNO       I   Next scan number, start of the file if 'READ',
                     the last+1 if WRITE
    CLKOLS(MAXCLC) I   The column pointer array in order, TIME,
                       TIME INT., SOURCE ID., ANTENNA NO., SUBARRAY,
                       FREQID, ROT.MEAS. GEODELAY, GEOPHASE, GEORATE,
                       DOPOFF, CLKGD 1, DCLKGD 1, CLKPD 1, DCLKPD 1,
                       ATMGD 1, DATMGD 1, ATMPD 1, DATPGD 1,
                       REAL1, IMAG1, RATE 1, DELAY 1, TSYS1, WEIGHT1,
                       REFANT 1
                       Following used if 2 polarizations per IF
                       CLKGD 2, DCLKGD 2, CLKPD 2, DCLKPD 2,
                       ATMGD 2, DATMGD 2, ATMPD 2, DATPGD 2,
                       REAL2, IMAG2, RATE 2, DELAY 2, TSYS2, WEIGHT2,
                       REFANT 1
    CLNUMV(MAXCLC) I   Element count in each column.
    IERR         I   Return error code, 0=>OK, else TABINI or TABIO
                     error.
```

CCINI

Creates and/or opens for writing (and reading) a specified CC (components table) file.

```
CCINI (LUN, NCOL, VOL, CNO, VER, CATBLK, BUF, IERR)
Inputs:
   LUN     I         Logical unit number to use
   VOL     I         Disk number
   CNO     I         Catalog number
In/out:
   NCOL    I         Number of columns: 3 or 7 are allowed.
   VER     I         Input: desired version number 0 -> new
                     Output: that used
   CATBLK  I(256)    File catalog header block
Output:
   BUF     I(768)    First 512 words required for later calls to
                     TABIO
   IERR    I         Error codes from TABINI or TABIO
```

CCMERG

Sorts AIPS CC tables to bring all components at the same cell together, then it sums them, and finally it resorts the file into the original order (by flux of the new components).

```
CCMERG (DISK, CNO, INVER, OUTVER, INPCMP, OUTCMP,
  *   JBUFS, BUFFER, IRET)
Inputs:
   DISK    I     File disk number
   CNO     I     File catalog number
   JBUFS   I     Number R   words in BUFFER
In/out:
   INVER   I     Input CC version number: 0 => MAXVER
   OUTVER  I     Output CC version number: 0 => MAXVER+1
Output:
   INPCMP  I     Number components on input.
   OUTCMP  I     Number components on output.
Common: /MAPHDR/ CATBLK for the affected image file
The routine assumes that the CATBLK is in this common already and
that the file has been opened in the catalog for WRITE.  (The image
file itself does not need to be open.)  The routine assumes that
the DFIL.INC common is initialized especially IBAD (BADDISK).
```

CHNDAT

Routine to create/fill/read CH/FQ extension tables. We are phasing out CH tables, so this routine will read them, but will only write FQ tables.

```
CHNDAT (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   NIF, FOFF, ISBAND, FREQID, IERR)
Inputs:
   OPCODE   C*4      Operation code:
                     'WRIT' = create/init for write or read
                     'READ' = open for read only
   BUFFER   I(512)   I/O buffer and related storage, also defines
                     file if open.
   DISK     I        Disk to use.
   CNO      I        Catalog slot number
   CATBLK   I(256)   Catalog header block.
```

```
   LUN      I        Logical unit number to use
   FREQID   I        Frequnecy ID #, if FQ tables exists
Input/Output:
   VER      I        CH file version
   NIF      I        Number of IFs.
   FOFF     D(*)     Frequency offset in Hz from ref. freq.
                        True = reference + offset.
   ISBAND   I(*  )   Sideband of each IF.
                     -1 => 0 video freq. is high freq. end
                      1 => 0 video freq. is low freq. end
Output:
   IERR     I        Return error code, 0=>OK, else TABINI or TABIO
                     error, -1 => tried to create/write an FQ table
```

CHNCOP

Copies selected portions of a Frequency (FQ) table extension file.

```
CHNCOP (INVER, OUTVER, LUNOLD, LUNNEW, VOLOLD,
  *   VOLNEW, CNOOLD, CNONEW, CATOLD, CATNEW, BIF, EIF, FREQID,
  *   BUFF1, WORK1, WORK2, IRET)
Inputs:
   LUNOLD     I     LUN for old file
   LUNNEW     I     LUN for new file
   VOLOLD     I     Disk number for old file.
   VOLNEW     I     Disk number for new file.
   CNOOLD     I     Catalog slot number for old file
   CNONEW     I     Catalog slot number for new file
   BIF        I     First IF to copy to output.
   EIF        I     Last IF to copy.
   FREQID     I     FREQ ID to copy.
   CATOLD(256)I     Catalog header for old file.
In/out:
   INVER      I     Version number to copy, 0 => copy all.
   OUTVER     I     Version number on output file, if more than one
                    copied (INVER=0) this will be the number of the
                    first file.  If OUTVER = 0, it will be taken as
                    1 higher than the previous highest version.
   CATNEW(256)I     Catalog header for new file.
Output:
   BUFF1(256) I    Work buffer
   WORK1(512) D    Work buffer to hold frequency table.
   WORK2(512) I    Work buffer to hold sideband table.
   IRET       I    Return error code  0 => ok
                                      6 => asked for too many IFs.
                                      other = CHNDAT error.
```

CLREFM

Routine to change the format of the CL table from an old one to a new format if necessary. NOTE: routine uses LUN 45 as a temporary logical unit number.

```
CLREFM (DISK, CNO, VER, CATBLK, LUN, IRET)
Inputs:
```

```
   DISK            I       Volume number
   CNO             I       Catalogue number
   VER             I       Version to check/modify
   CATBLK(256)     I       Catalogue header
   LUN             I       LUN to use
Output:
   IRET            I       Error, 0 => OK

Note, routine will leave no trace of its operation, i.e. CL table
will be closed on output and will have same number as one specified.
Difference will be only that number of columns has changed if that
is required.
```

FLGINI

Creates and initializes FLAG (FG) extension tables.

```
FLGINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   IFGRNO, FGKOLS, FGNUMV, IERR)
 Inputs:
  OPCODE         C*4 Operation code:
                     'WRIT' = create/init for write or read
                     'READ' = open for read only
  BUFFER(512)     I   I/O buffer and related storage, also defines file
                     if open.
  DISK            I   Disk to use.
  CNO             I   Catalog slot number
  VER             I   FG file version
  CATBLK(256)     I   Catalog header block.
  LUN             I   Logical unit number to use
 Output:
  IFGRNO          I   Next scan number, start of the file if 'READ',
                     the last+1 if WRITE
  FGKOLS(MAXFGC)  I   The column pointer array in order, SOURCE,
                     SUBARRAY, ANTS, TIMERANG, IFS, CHANS, PFLAGS,
                     REASON
  FGNUMV(MAXFGC)  I   Element count in each column.
  IERR            I   Return error code, 0=>OK, else TABINI or TABIO
                     error.
```

FQINI

Creates and initializes frequency (FQ) extension tables.

```
FQINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   IFQRNO, FQKOLS, FQNUMV, NUMIF, IERR)
Inputs:
  OPCODE         C*4 Operation code:
                     'WRIT' = create/init for write or read
                     'READ' = open for read only
  BUFFER(4096)    I   I/O buffer and related storage, also defines file
                     if open.
  DISK            I   Disk to use.
  CNO             I   Catalog slot number
```

```
  VER             I   FQ file version
  CATBLK(256)     I   Catalog header block.
  LUN             I   Logical unit number to use
Input/output
  NUMIF           I    Number of IFs
Output:
  IFQRNO          I   Next row number, start of the file if 'READ',
                      the last+1 if WRITE
  FQKOLS(MAXFQC)  I   The column pointer array in order:
                      FQID, IFFREQ, IFCHW, IFTBW, IFSIDE
  FQNUMV(MAXFQC)  I   Element count in each column.
  IERR            I   Return error code, 0=>OK, else TABINI or TABIO
                      error.
```

GETNAN

Determines the number of subarrays in a data set from the number of AN files and returns the highest antennas number in each subarray If no antennas are found, one subarray with 28 antennas assumed. If an error occurs, information about subarrays from AN files found is returned; although an error code is returned.

```
GETNAN (DISK, CNO, CATBLK, LUN, BUFFER, NUMAN, IRET)
Inputs:
   DISK     I        Disk to use.
   CNO      I        Catalog slot number
   CATBLK   I(256)   Catalog header block.
   LUN      I        Logical unit number to use
   BUFFER   I(512)   I/O buffer and related storage.
Output:
   NUMAN    I(51)    1st element = no. subarrays followed by
                     the highest antenna number in each subarray.
   IRET     I        Return error code, 0 => ok,
                        else TABINI or TABIO error.
                        10 = no AN files.
```

NDXINI

Creates and initializes INDEX (NX) extention tables.

```
NDXINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   INXRNO, NXKOLS, NXNUMV, IERR)
Inputs:
  OPCODE       C*4 Operation code:
                   'WRIT' = create/init for write or read
                   'READ' = open for read only
  BUFFER(512)  I   I/O buffer and related storage, also defines file
                   if open.
  DISK         I   Disk to use.
  CNO          I   Catalog slot number
  VER          I   NX file version
  CATBLK(256)  I   Catalog header block.
  LUN          I   Logical unit number to use
Output:
  INXRNO         I   Next scan number, start of the file if 'READ',
```

```
                     the last+1 if WRITE
 NXKOLS(MAXNXC) I   The column pointer array in order, TIME,
                     TIME INTERVAL, SOURCE ID, SUBARRAY, START VIS,
                     END VIS, FREQID.
 NXNUMV(MAXNXC) I   Element count in each column.
 IERR           I   Return error code, 0=>OK, else TABINI or TABIO
                     error.
```

SNINI

Creates and initializes solution (SN) extension tables.

```
SNINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   ISNRNO, SNKOLS, SNNUMV, NUMANT, NUMPOL, NUMIF, NUMNOD, GMMOD,
  *   RANOD, DECNOD, ISAPPL, SNTYPE, IERR)
Inputs:
  OPCODE      C*4 Operation code:
                  'WRIT' = create/init for write or read
                  'READ' = open for read only
  BUFFER(512) I   I/O buffer and related storage, also defines file
                  if open.
  DISK        I   Disk to use.
  CNO         I   Catalog slot number
  VER         I   SN file version
  CATBLK(256) I   Catalog header block.
  LUN         I   Logical unit number to use
Input/output
  NUMANT      I    Number of antennas
  NUMPOL      I    Number of IFs per group
  NUMIF       I    Number of IF groups
  NUMNOD      I    Number of interpolation nodes. Will handle
                   up to 25 interpolation nodes.
  GMMOD       R    Mean gain modulus
  RANOD(*)    R    RA offset of interpolation nodes (deg.)
  DECNOD(*)   R    Dec. offset of interpolation nodes (deg.)
  ISAPPL      L    True if this SN table has been applied to
                   the  CL table.
  SNTYPE      I    "solution" type, 1=>"Clock", 2=>"atmosphere"
Output:
  ISNRNO      I   Next scan number, start of the file if 'READ',
                  the last+1 if WRITE
  SNKOLS(MAXSNC) I  The column pointer array in order, TIME,
                    TIME INT., SOURCE ID., ANTENNA NO., SUBARRAY,
                    FREQ. ID., IFR, NODE NO.,
                    REAL1, IMAG1, DELAY1, RATE1, WEIGHT1, REFANT 1,
                    Following used if 2 polarizations per IF
                    REAL2, IMAG2, DELAY2, RATE2, WEIGHT2, REFANT 2
  SNNUMV(MAXSNC) I  Element count in each column.
  IERR        I   Return error code, 0=>OK, else TABINI or TABIO
                  error.
```

SNREFM

Routine to change the format of the SN table from an old one to the new one if necessary. NOTE: routine uses LUN 45 as a temporary logical unit number.

```
SNREFM (DISK, CNO, VER, CATBLK, LUN, IRET)
Inputs:
   DISK           I       Volume number
   CNO            I       Catalogue number
   VER            I       Version to check/modify
   CATBLK(256)    I       Catalogue header
   LUN            I       LUN to use
Output:
   IRET           I       Error, 0 => OK

Note, routine will leave no trace of its operation, i.e. SN table
will be closed on output and will have same number as one specified.
Difference will be only that number of columns has changed if that
is required.
```

SOUINI

Creates and initializes SOURCE (SU) extension tables.

```
SOUINI (OPCODE, BUFFER, DISK, CNO, VER, CATBLK, LUN,
  *   NUMIF, VELTYP, VELDEF, ISURNO, SUKOLS, SUNUMV, IERR)
Inputs:
  OPCODE          C*4 Operation code:
                      'WRIT' = create/init for write or read
                      'READ' = open for read only
  BUFFER(512)     I   I/O buffer and related storage, also defines file
                      if open.
  DISK            I   Disk to use.
  CNO             I   Catalog slot number
  VER             I   SU file version
  CATBLK(256)     I   Catalog header block.
  LUN             I   Logical unit number to use
Input/Output:
  NUMIF           I   Table keyword, gives the number of IFs
  VELTYP          C*8 Velocity type,
  VELDEF          C*8 Velocity defination 'RADIO','OPTICAL',
Output:
  ISURNO          I   Next scan number, start of the file if 'READ',
                      the last+1 if WRITE
  SUKOLS(MAXSUC)  I   The column pointer array in order, ID. NO.,
                      SOURCE, QUAL, CALCODE, IFLUX, QFLUX, UFLUX,
                      VFLUX, FREQO, BANDWIDTH, RAEPO, DECEPO, EPOCH,
                      RAAPP, DECAPP, LSRVEL, LRESTF, PMRA, PMDEC
  SUNUMV(MAXSUC)  I   Element count in each column.
  IERR            I   Return error code, 0=>OK, else TABINI or TABIO
                      error.
```

TABAN

Does I/O to Antenna (AN) tables. Usually used after setup by ANTINI.

```
TABAN (OPCODE, BUFFER, IANRNO, ANKOLS, ANNUMV, ANNAME,
  *   STAXYZ, ORBPRM, NOSTA, MNTSTA, STAXOF, POLTYA, POLAA, POLCA,
  *   POLTYB, POLAB, POLCB, IERR)
```

```
Inputs:
   OPCODE   C*4        Operation code:
                       'READ' = read entry from table.
                       'WRIT' = write entry in table.
                       'CLOS' = close file, flush on write
   BUFFER   I(512)     I/O buffer and related storage, also defines
                       file if open. Should have been returned by
                       ANTINI or TABINI.
   IANRNO   I          Next scan number to read or write.
   ANKOLS   I(MAXANC)  The column pointer array in order, ANNAME,
                       STABXYZ, ORBPARM, NOSTA, MNTSTA, STAXOF, POLTYA,
                       POLAA, POLCALA, POLTYB, POLAB, POLCALB
   ANNUMV   I(MAXANC)  Element count in each column.
Input/output: (written to or read from antenna file)
   ANNAME   C*8        Station name
   STAXYZ   D(3)       X,Y,Z offset from array center
   ORBPRM   D(*)       Orbital parameters.
   NOSTA    I          Station number
   MNTSTA   I          Mount type, 0=altaz, 1=equatorial, 2=orbiting
   STAXOF   R          Axis offset
   POLTYA   C*2        Feed A feed poln. type 'R','L','X','Y'
   POLAA    R          Feed A feed position angle.
   POLCA    R(*)       Feed A poln. cal parameter. (note 2)
   POLTYB   C*2        Feed B feed poln. type 'R','L','X','Y'
   POLAB    R          Feed B feed position angle.
   POLCB    R(*)       Feed B poln. cal parameters.
Output:
   IANRNO   I          Next GAIN number.
   IERR     I          Error code, 0=>OK else TABIO error.
                       Note: -1=> read but record deselected.
Usage NOTE: use the include 'DANT.INC' for the declarations in
ANTINI and TABAN.
```

TABBL

Does I/O to baseline (BL) extention tables. Usually used after setup by BLINI.

```
TABBL (OPCODE, BUFFER, IBLRNO, BLKOLS, BLNUMV,
  *   NUMPOL, TIME, SOURID, SUBA, ANT1, ANT2, FREQID,
  *   FACMUL, FACADD, IERR)
Inputs:
  OPCODE       C*4 Operation code:
                   'READ' = read entry from table.
                   'WRIT' = write entry in table.
                   'CLOS' = close file, flush on write
  BUFFER(1024) I   I/O buffer and related storage, also defines file
                   if open. Should have been returned by BLINI or
                   TABINI.
  IBLRNO       I   Next entry number to read or write.
  BLKOLS(MAXBLC)   I   The column pointer array in order, TIME,
                   SOURID, SUBARRAY, ANTENNA1, ANTENNA2, FREQID,
                   REALM1, IMAGM1, REALA1, IMAGA1,
                   Following used if 2 polarizations per IF
                   REALM2, IMAGM2, REALA2, IMAGA2.
```

```
  BLNUMV(MAXBLC)   I   Element count in each column.
  NUMPOL        I   Number of polarizations per IF.
Input/output: (written to or read from baseline file)
  TIME          R   Center time of record (Days)
  SOURID        I   Source ID number.
  SUBA          I   Subarray number.
  ANT1          I   First antenna number.
  ANT2          I   Second antenna number.
  FREQID        I   Freqid #
  FACMUL(2,2,m)R   Multiplicative correction,  m IFs
                    second dimension is polarization,
                    (1,*,*) = real, (2,*,*) = imag.
  FACADD(2,2,m)R   Additive correction, m IFs
Output:
  IBLRNO        I   Next solution number.
  IERR          I   Error code, 0=>OK else TABIO error.
                    Note: -1=> read but record deselected.
```

TABBP

Does I/O to bandpass (BP) extention tables. Usually used after setup by BPINI.

```
TABBP (OPCODE, BUFFER, IBPRNO, BPKOLS, BPNUMV,
  *   NUMIF, NUMFRQ, NUMPOL, TIME, INTERV, SOURID, SUBA, ANT,
  *   BANDW, IFFREQ, FREQID, REFANT, REAL, IMAG, IERR)
Inputs:
  OPCODE        C*4 Operation code:
                    'READ' = read entry from table.
                    'WRIT' = write entry in table.
                    'CLOS' = close file, flush on write
  BUFFER(4096) I   I/O buffer and related storage, also defines file
                    if open. Should have been returned by BPINI or
                    TABINI.
  IBPRNO        I   Next entry number to read or write.
  BPKOLS(MAXBPC) I   The column pointer array in order,
                      TIME, INTERVAL, SOURID,
                      SUBARRAY, ANTENNA,
                      BANDW (of individual channel), IFREQ, FREQID,
                      REFANT1, REAL1, IMAG1, .. etc for all channels
                      Following used if 2 polarizations per IF
                      REFANT2, REAL2, IMAG2.
  BPNUMV(MAXBPC) I   Element count in each column.
  NUMIF         I   Number of IF's
  NUMFRQ        I   Number of chns
  NUMPOL        I   Number of polarizations per IF.
Input/output: (written to or read from baseline file)
  TIME          D   Center time of record (Days)
  INTERV        R   Time interval of record (Days)
  SOURID        I   Source ID number.
  SUBA          I   Subarray number.
  ANT           I   Antenna number.
  BANDW         R   Bandwidth of an individual channel (Hz)
  IFFREQ(m)     D   Reference frequency for each IF (Hz)
  FREQID        I   Freq. id number
```

```
  REFANT(2)    I   Reference Antenna; one for each poln
  REAL(2,n,m)  R   Real part of complex bandpass
                   m IFS; n channels; 2 polns
  IMAG(2,n,m)  R   Imag part of complex bandpass
                   m IFS; n channels; 2 polns
Output:
  IBPRNO       I   Next solution number.
  IERR         I   Error code, 0=>OK else TABIO error.
                   Note: -1=> read but polzn #1 flagged
                         -2=> read but polzn #2 flagged
                         -3=> both flagged
```

TABCAL

Does I/O to CALIBRATION (CL) extention tables. Usually used after setup by CALINI.

```
TABCAL (OPCODE, BUFFER, ICLRNO, CLKOLS, CLNUMV,
  *   NUMPOL, NUMIF, TIME, TIMEI, SOURID, ANTNO, SUBA, FREQID, IFR,
  *   GEODLY, GEOPHA, GEORAT, DOPOFF, CLKGD, DCLKGD, CLKPD, DCLKPD,
  *   ATMGD, DATMGD, ATMPD, DATMPD, CREAL, CIMAG, DELAY, RATE, TSYS,
  *   WEIGHT, REFA, IERR)
Inputs:
  OPCODE       C*4 Operation code:
                   'READ' = read entry from table.
                   'WRIT' = write entry in table.
                   'CLOS' = close file, flush on write
  BUFFER(512)  I   I/O buffer and related storage, also defines file
                   if open. Should have been returned by TABINI or
                   TABINI.
  ICLRNO       I   Next scan number to read or write.
  CLKOLS(MAXCLC) I   The column pointer array in order, TIME,
                     TIME INT., SOURCE ID., ANTENNA NO., SUBARRAY,
                     FREQID, IFR (Ionesph. Faraday Rot.),
                     GEODELAY, GEOPHASE, GEORATE, DOPPOFF,
                     CLKGD 1, DCLKGD 1, CLKPD 1, DCLKPD 1,
                     ATMGD 1, DATMGD 1, ATMPD 1, DATPGD 1,
                     REAL1, IMAG1, RATE 1, DELAY 1, TSYS1, WEIGHT1,
                     REFANT 1
                     Following used if 2 polarizations per IF
                     CLKGD 2, DCLKGD 2, CLKPD 2, DCLKPD 2,
                     ATMGD 2, DATMGD 2, ATMPD 2, DATPGD 2,
                     REAL2, IMAG2, RATE 2, DELAY 2, TSYS2, WEIGHT2,
                     REFANT 2
  CLNUMV(MAXCLC) I   Element count in each column.
  NUMPOL       I   Number of polarizations per IF.
  NUMIF        I   Number of IFs.
Input/output: (written to or read from CAL file)
  TIME         D   Center time of CAL record (Days)
  TIMEI        R   Time interval covered by record (days)
  SOURID       I   Source ID as defined in the SOURCE table.
  ANTNO        I   Antenna number.
  SUBA         I   Subarray number.
  FREQID       I   Freqid #
  IFR          R   Ionospheric Faraday Rotation (rad/m**2)
```

```
  GEODLY       D   Geometric delay at TIME (sec)
  GEOPHA       D   Phase of sinusoid (turns)
  GEORAT       D   Time rate of change of GEOPHA (Hz)
  DOPOFF(*)    R   Doppler offset for each IF (Hz)
  CLKGD(2,*)   R   "Clock" Group delay (sec) 1/poln/IF
  DCLKGD(2,*)  R   Time derivative of "Clock" Group delay (sec/sec)
  CLKPD(2,*)   R   "Clock" Phase delay (sec) 1/poln/IF
  DCLKPD(2,*)  R   Time derivative of "Clock" Phase delay (sec/sec)
  ATMGD(2,*)   R   "Atmos" Group delay (sec) 1/poln/IF
  DATMGD(2,*)  R   Time derivative of "Atmos" Group delay (sec/sec)
  ATMPD(2,*)   R   "Atmos" Phase delay (sec) 1/poln/IF
  DATMPD(2,*)  R   Time derivative of "Atmos" Phase delay (sec/sec)
  CREAL(2,*)   R   Real part of the complex gain, 1/poln/IF
  CIMAG(2,*)   R   Imag part of the complex gain, 1/poln/IF
  DELAY(2,*)   R   Residual group delay (sec), 1/poln/IF
  RATE(2,*)    R   Residual fringe rate (Hz), 1/poln/IF
  TSYS(2,*)    R   System temperature (K), 1/poln/IF
  WEIGHT(2,*)  R   Weight of solution, 1/poln/IF
  REFA(2,*)    I   Reference antenna use for cal. solution.
Output:
  ICLRNO       I   Next CAL number.
  IERR         I   Error code, 0=>OK else TABIO error.
                   Note: -1=> read but record deselected.
```

TABFLG

Does I/O to FLAG (FG) extention tables. Usually used after setup by FLGINI.

```
TABFLG (OPCODE, BUFFER, IFGRNO, FGKOLS, FGNUMV,
  *   SOURID, SUBA, ANTS, TIMER, IFS, CHANS, PFLAGS, REASON, IERR)
Inputs:
  OPCODE         C*4 Operation code:
                     'READ' = read entry from table.
                     'WRIT' = write entry in table (must have been
                     opened with 'WRIT'.
                     'FLAG' = like 'WRIT' but entry deselected.
                     'CLOS' = close file, flush on write
  BUFFER(512)     I   I/O buffer and related storage, also defines file
                     if open. Should have been returned by FLGINI or
                     TABINI.
  IFGRNO          I   Next FLAG entry number to read or write.
  FGKOLS(MAXFGC)  I   The column pointer array in order, SOURCE,
                     SUBARRAY, ANTS, TIMERANG, IFS, CHANS, PFLAGS,
                     REASON
  FGNUMV(MAXFGC)    I   Element count in each column.
Input/output: (written to or read from FLAG file)
  SOURID          I   Source ID as defined in the SOURCE table.
  SUBA            I   Subarray number.
  ANTS(2)         I   Antenna numbers, 0=>all
  TIMER(2)        R   Start and end time of data to be flagged (Days)
  IFS(2)          I   First and last IF numbers to flag. 0=>all
  CHANS(2)        I   First and last channel numbers to flag. 0=>all
  PFLAGS(4)       L   Polarization flags, same order as in data.
                     .TRUE. => polarization flagged.
```

```
    REASON          C*24 Reason for flagging
  Output:
    IFGRNO          I    Next scan number.
    IERR            I    Error code, 0=>OK else TABIO error.
                         Note: -1=> read but record deselected.
```

TABFQ

Does I/O to frequency (FQ) extension tables. Usually used after setup by FQINI.

```
  TABFQ (OPCODE, BUFFER, IFQRNO, FQKOLS, FQNUMV,
    *   NUMIF, FQID, IFFREQ, IFCHW, IFTBW, IFSIDE, IERR)
  Inputs:
    OPCODE          C*4 Operation code:
                         'READ' = read entry from table.
                         'WRIT' = write entry in table.
                         'CLOS' = close file, flush on write
    BUFFER(4096)    I    I/O buffer and related storage, also defines file
                         if open. Should have been returned by FQINI or
                         TABINI.
    IFQRNO          I    Next entry number to read or write.
    FQKOLS(MAXFQC)  I    The column pointer array in order,
                         FQID, IFFREQ, IFCHW, IFTBW, IFSIDE
    FQNUMV(MAXFQC   I    Element count in each column.
    NUMIF           I    Number of IF's
  Input/output: (written to or read from frequency table)
    FQID            I    Frequency ID number, is random parameter
                         in uv-data.
    IFFREQ(*)       D    Reference frequency for each IF (Hz)
    IFCHW(*)        R    Bandwidth of an individual channel (Hz)
    IFTBW(*)        R    Total bandwidth of the IF (Hz)
    IFSIDE(*)       I    Sideband of the IF (-1 => lower, +1 => upper)
  Output:
    IFQRNO          I    Next row number.
    IERR            I    Error code, 0=>OK else TABIO error.
```

TABNDX

Does I/O to INDEX (NX) extention tables. Usually used after setup by NDXINI.

```
  TABNDX (OPCODE, BUFFER, INXRNO, NXKOLS, NXNUMV,
    *   TIME, DTIME, IDSOUR, SUBARR, VSTART, VEND, FREQID, IERR)
  Inputs:
    OPCODE       C*4 Operation code:
                      'READ' = read entry from table.
                      'WRIT' = write entry in table.
                      'CLOS' = close file, flush on write
    BUFFER(512)  I    I/O buffer and related storage, also defines file
                      if open. Should have been returned by NDXINI or
                      TABINI.
    INXRNO       I    Next scan number to read or write.
    NXKOLS(MAXNXC)  I    The column pointer array in order, TIME,
                         TIME INTERVAL, SOURCE ID, SUBARRAY, START VIS,
                         END VIS, FREQID.
```

```
  NXNUMV(MAXNXC)  I   Element count in each column, set by NDXINI.
Input/output: (written to or read from INDEX file)
  TIME          R   Center time of the scan (Days)
  DTIME         R   Duration of scan (Days)
  IDSOUR        I   Source ID as defined in then SOURCE table.
  SUBARR        I   Subarray number.
  VSTART        I   First visibility number in file.
  VEND          I   Last visibility number in file.
  FREQID        I   Freqid of scan
Output:
  INXRNO        I   Next scan number.
  IERR          I   Error code, 0=>OK else TABIO error.
                    Note: -1=> read but record deselected.
```

TABSN

Does I/O to solution (SN) extention tables. Usually used after setup by SNINI.

```
TABSN (OPCODE, BUFFER, ISNRNO, SNKOLS, SNNUMV,
  *   NUMPOL, TIME, TIMEI, SOURID, ANTNO, SUBA, FREQID, IFR,
  *   NODENO, CREAL, CIMAG, DELAY, RATE, WEIGHT, REFA, IERR)
Inputs:
  OPCODE        C*4 Operation code:
                    'READ' = read entry from table.
                    'WRIT' = write entry in table.
                    'CLOS' = close file, flush on write
  BUFFER(512)   I   I/O buffer and related storage, also defines file
                    if open. Should have been returned by TABINI or
                    TABINI.
  ISNRNO        I   Next scan number to read or write.
  SNKOLS(MAXSNC) I    The column pointer array in order, TIME,
                      TIME INT., SOURCE ID., ANTENNA NO., SUBARRAY,
                      FREQID, IFR, NODE NO.,
                      REAL1, IMAG1, DELAY1, RATE1, WEIGHT1, REFANT 1,
                      Following used if 2 polarizations per IF
                      REAL2, IMAG2, DELAY2, RATE2, WEIGHT2, REFANT 2,
  SNNUMV(MAXSNC) I    Element count in each column.
  NUMPOL        I   Number of polarizations per IF.
Input/output: (written to or read from solution file)
  TIME          D   Center time of solution record (Days)
  TIMEI         R   Time interval covered by record (days)
  SOURID        I   Source ID as defined in the SOURCE table.
  ANTNO         I   Antenna number.
  SUBA          I   Subarray number.
  FREQID        I   Freqid #
  IFR           R   Ionospheric Faraday Rotation (rad/m**2)
  NODENO        I   Interpolation node number
  CREAL(2,*)    R   Real part of the complex gain, 1 /Poln/IF
  CIMAG(2,*)    R   Imag part of the complex gain, 1 /Poln/IF
  DELAY(2,*)    R   Residual group delay (sec), 1 /Poln/IF
  RATE(2,*)     R   Residual fringe rate (Hz), 1 /Poln/IF
  WEIGHT(2,*)   R   Weight of solution, 1 /Poln/IF
  REFA(2,*)     R   Ref. ant. of solution, 1 /Poln/IF
Output:
```

```
  ISNRNO       I   Next solution number.
  IERR         I   Error code, 0=>OK else TABIO error.
                   Note: -1=> read but record deselected.
```

TABSOU

Does I/O to SOURCE (SU) extention tables. Usually used after setup by SOUINI.

```
TABSOU (OPCODE, BUFFER, ISURNO, SUKOLS, SUNUMV, IDSOU, SOUNAM,
  *   QUAL, CALCOD, FLUX, FREQO, BANDW, RAEPO, DECEPO, EPOCH,
  *   RAAPP, DECAPP, LSRVEL, LRESTF, PMRA, PMDEC, IERR)
Inputs:
  OPCODE          C*4 Operation code:
                      'READ' = read entry from table.
                      'WRIT' = write entry in table.
                      'CLOS' = close file, flush on write
  BUFFER(768)     I   I/O buffer and related storage, also defines file
                      if open. Should have been returned by SOUINI or
                      TABINI.
  ISURNO          I   Next scan number to read or write.
  SUKOLS(MAXSUC)  I   The column pointer array in order, ID. NO.,
                      SOURCE, QUAL, CALCODE, IFLUX, QFLUX, UFLUX,
                      VFLUX, FREQO, BANDWIDTH, RAEPO, DECEPO, EPOCH,
                      RAAPP, DECAPP, LSRVEL, LRESTF, PMRA, PMDEC
  SUNUMV(MAXSUC)  I   Element count in each column.
Input/output: (written to or read from SoUrce file)
  IDSOUR          I   Source ID as defined in the SOURCE table.
  SOUNAM          C*16 Source name
  QUAL            I   Source qualifier.
  CALCOD          C*4 Calibrator code
  FLUX(4,*)       R   Total flux density I, Q, U, V pol, (Jy)
                      1 set per IF.
  FREQO(*)        D   Frequency offset (Hz) from IF nominal.
  BANDW           D   Bandwidth (Hz)
  RAEPO           D   Right ascension at mean EPOCH (degrees)
  DECEPO          D   Declination at mean EPOCH (degrees)
  EPOCH           D   Mean Epoch for position in yr. since year 0.0
  RAAPP           D   Apparent Right ascension (degrees)
  DECAPP          D   Apparent Declination(degrees)
  LSRVEL(*)       D   LSR velocity (m/sec) of each IF
  LRESTF(*)       D   Line rest frequency (Hz) of each IF
  PMRA            D   Proper motion (deg/day) in RA
  PMDEC           D   Proper motion (deg/day) in declination
Output:
  ISURNO          I   Next source number.
  IERR            I   Error code, 0=>OK else TABIO error.
                      Note: -1=> read but record deselected.
```