NATIONAL RADIO ASTRONOMY OBSERVATORY

SOCORRO, NEW MEXICO

December 29, 1976

VLA COMPUTER MEMORANDUM NO. 134

VLA Correlator Data Formats for the DEC-10 System

by A. Braun, D. Ehnebuske, J. Hudson

1. Introduction

A revised data format is presented here, together with a description of the access programs. A tentative schedule for implementation is also included. The data format described here represents a number of viewpoints and satisfies what we hope is a majority of observing needs. Your comments are solicited.

Two major requirements are met by this choice of data structure: First, it will be possible for user programs to access the data base without recourse to more than a minimal package of routines; second, the structure provides a natural and simple interface to the CANDID data reduction system. While SAIL is to be the implementation language for the access programs, it should be possible at a later date to re-code the programs in the DEC-10 macro assembler language, once requirements are better understood. At that time, interfaces to other language processors can be provided.

2. Procedures

Let us outline here what we think will be a reasonable set of operating procedures for handling data files on the asynchronous subsystem.

Each observer on the VLA will be assigned a project/programmer number, where his visibility data files, indices, gain tables, and synthesized maps will be held. No guarantee is made by the NRAO staff that his directory and files will be immune to system errors, and so the

observer should take care to back up those files which he considers valuable. After a designated time period, the files will be deleted as a matter of course. Some data may be considered "public," and kept as files under the [11,1] directory. In this category fall observations of calibration sources and monitor data.

A program called FILLER generates the user's data base and also the public files, either from magnetic tape or fixed head disk. Prior to running FILLER, the user gives the program a list of sources to be written on certain files, and the ordering in which he wants the data on those files. Those sources designated as calibrators will also be written onto a public file, with a preset amount of time-averaging applied to the visibilities. The observer may choose between two major modes of ordering his data: (1) Time -- Baseline (natural), and (2) f(u,v). Records sorted according to some function f(u,v) will be ordered according to values of that key computed from the stored (u,v) coordinate in each record. For example, f(u,v) = |u| might be a reasonable sorting key for purposes of gridding for the Fast Fourier Transform. The observer also selects the degree of averaging desired for the records. It is understood that the ordering and averaging are constant for a given file. While the opportunity is obvious here for abusing file space, we suggest that the file lists be cleared with the systems staff prior to observing to ensure that excessive duplication does not take place, and that there is sufficient disk space available.

3. Data Structure

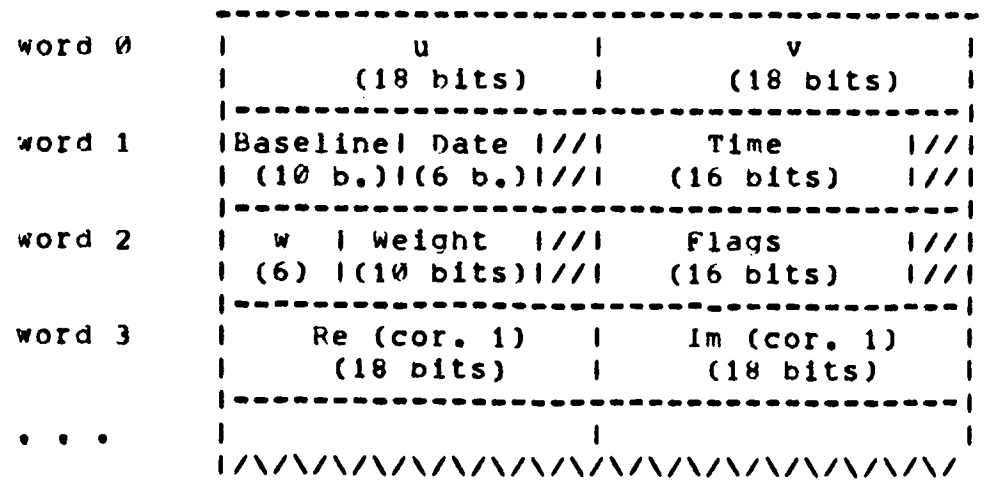
Conceptually, the data structure takes the form:

```
1 Observer,
  2 Visibility data set (several),
    3 Data record (1:Nrecs),
  2 Visibility data set index (1 for each vis. data set),
    3 Index record (1:Nobs),
  2 Gain table (1 for each vis. data set),
    3 Gain table record (1:Ntimes),
  2 Monitor data table abstract,

1 Public,
  2 Index of observations,
  2 Visibility data set (one for calibrators),
    3 Data record(1:Nrecs),
  2 Visibility data set index,
    3 Index record (1:Nobs),
  2 Monitor data file,
  2 Antenna station table;
```

Under the file directory for his project/programmer number, the observer's visibility data, index to the data, gain table, and monitor table are represented by different files. Visibility data are kept in separate files, principally according to whether the user intends to map the data or to use the observations for experimentation requiring access by time -- baseline. A facility will exist to permit breaking, say, one source out of a given file and copying it to another. Files intended for mapping are sorted according to some function of (u,v), say |u|. The index file pertaining to each visibility data set contains such information as source name and qualifier, start and stop times, and indices for retrieval from the visibility data set (which looks like a large array of data records). Other information, such as the definition of frequencies, bandwidths, and polarizations of the IF channels, is stored in the index. The gain table contains nominal antenna phases, sensitivities, calibrated gain factors, and polarization corrections. The antenna station table lists the antenna positions and station designations, with a separate record for each time period for which any one antenna was moved. The monitor data table contains an abstraction of monitor data which is of interest in the calibration and correction of the visibility data. It is kept in time-sequence with a legend describing the data that were logged.

Record layout for visibility records is as follows:



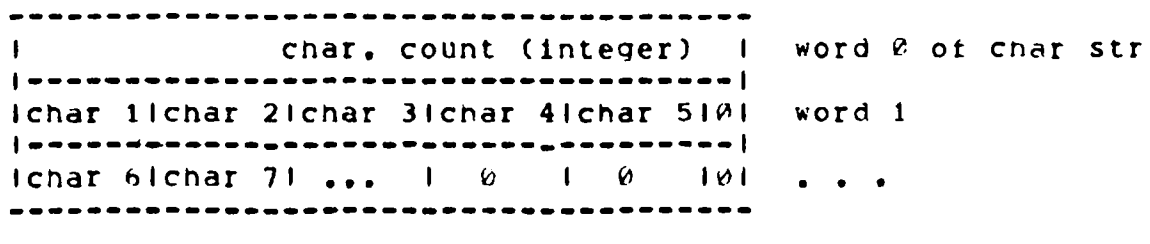
The 18-bit halfwords represent data which is only of 16-bit precision; hence, the entire format is readily translated into a Modcomp or DEC minicomputer storage format. (u,v) are kept in 16's of nanoseconds. The w component is stored with only sign and 5 most significant bits. The baseline number is a pair of 5-bit physical antenna numbers. The 6-bit date is the low order 6 bits in the Modified Julian Atomic Date. Time is seconds, IAT, past midnight, divided by 2. The weight field allows for averaging up to 1024 time-wise adjacent records together. The flags field is broken into groups of 2 bits for each correlator (assuming a maximum

of 8 correlator data words). It is suggested that only 2 classes of records be maintained: of lengths 7 and 11 words, to hold data for 2 and 4 active IF channels, respectively.

The visibility data index is organized as follows:

```
1 Visibility index (1:Nobs),
  2 Recsize          integer,
  2 Source name      char(10),
  2 Source qual      integer,
  2 Mode qual        integer,
  2 Calibr code      integer,
  2 Gain code        floating,
  2 Sort code        integer,
  2 Averaging time   floating,
  2 Source ptr1      integer,
  2 Source ptr2      integer,
  2 Antptr           integer,
  2 Gain ptr1        integer,
  2 Gain ptr2        integer,
  2 Source link      integer,
  2 RA1950           floating,
  2 dec1950          floating,
  2 RAobsdate        floating,
  2 decobsdate       floating,
  2 Start MJAD       floating,
  2 Start IAT        floating,
  2 Start LST        floating,
  2 Correl(1:4)      char(5),
  2 Band(1:4)        char(5),
  2 Freq(1:4)        floating,
  2 Bandw(1:4)       floating,
  2 SumLO(1:4)       floating;
```

All character strings in these various files will be stored in the format



where zero padding is used for the unused characters filling out the maximum string length (to allow word comparison of char. data). This will

thus pertain to such fields in the above structure as source name, correl, band, etc. The number of words occupied by a string of length N characters is 1 + [(N+4)/5].

The index file contains a number of records equal to the number of times the observer switched from one source to another in the case of one sub-array; more index entries will be made if the observer used multiple sub-arrays each observing different sources or with different frequency settings. Each index entry corresponds to a set of visibility data records, whose first and last numbers in the corresponding visibility data file are recorded in Source ptr1 and Source ptr2. Antenna positions used during the observation can be obtained from the antenna table file, at Antptr. The gain table entries pertinent to the observation are pointed to by Gain ptr1 and Gain ptr2. The mode qualifier is the same one used by the synchronous subsystem. If there are more observations of the same source at the same mode later on in the file, the Source link field will contain a pointer to the index record pertaining to that group of visibility data records. The calibrator code indicates whether the source was a calibrator (if non-zero); codes '101, '102, ... (the ASCII characters "A", "B", ...) stand for various types of calibrator. All coordinates are expressed in radians. Dates are modified Julian atomic dates. Start IAT is in seconds. LST is kept in radians. The field "correl" is used to name the 4 correlations in each correlator group of the visibility data (e.g.: "RR", "RL"). Band may be "L", "C", "K", "U". Frequencies and the field "SumLO" are kept in Hertz. "NrIFs" counts the number of IF frequencies used (2 or 4), and determines whether the logical record length of the visibility data is 7 or 11 words.

The gain tables are organized as follows:

```
1 Gain table (1:Nobs),
  2 nomgain(1:28, 1:4) halfword complex,   (112 words)
  2 corrgain(1:28, 1:4) halfword complex;  (112 words)
```

where nomgain is that indicated by the synchronous subsystem (perhaps feedback from the asynchronous subsystem based on past history of correlators), which has already been applied to the data, and corrgain is that which should be multiplied into the data whenever they are accessed.

The antenna station table is organized as follows:

```
1 Antstations (1:Nentries),
  2 Antennas (1:Nant),
    3 Station name  char (5),
    3 X             floating,
    3 Y             floating,
    3 Z             floating;
```

where there will be an entry each time the antennas are moved. The antenna array is kept in order of the physical antenna number. The

station name is by array, arm, and number, e.g.: "CW9." The station coordinates are in a right-handed system with Z toward the north celestial pole, X toward the E point on the horizon.

4. Processing Routines

Initially, the following routines, capable of being called from "stand-alone" SAIL programs, will be provided:

1. FILLER: Accepts raw data which have been read from fixed-head disk or tape, as generated by the Synchronous subsystem. Initially, FILLER is instructed as to what files are to receive what data, and how to sort it. FILLER constructs one or more visibility data files, with corresponding index files and gain tables.

2. SUMARY: Prepares a text file suitable for display on terminals or line printer giving a summary of the data in a given visibility data file. The report is based on the contents of the index file. Output may be sorted on source or time, at the user's option.

3. VLIST: Prepares a text file suitable for display on terminals or line printer giving the same information as SUMARY, followed by a listing of all records pertaining to a set of baselines requested by the user. Plots of Re(V), Im(V) or modulus(V), phase(V) will also be provided.

4. VGRID: Grid the data in a visibility data file for a given source, polarization component, and frequency. Produce an output file suitable for Fourier transforming via the FFT algorithm. Various options will be provided for weighting, tapering the data. Requires data in input file to be sorted according to iul.

5. COPY: Copy data for a given source in the input file onto an output file containing only that source.

6. OPNCOR: Open file for GET or ED functions, below.

7. GETCOR: Get one record of correlator data from input file for a given source, source qualifier, mode of observing.

8. EDCOR: Replace one record just obtained by GETCOR with its core image, after the user has modified it.

9. CLSCOR: Close correlator data file.

Of course, CANDID access is automatically provided by having all files formatted as external structures.

5. Accessing Algorithms

For files that are to be sorted on a particular key, (u,v say, either the sorted strings can be merged upon request, or at certain intervals. We propose not to merge until a file has been filled: this providing a simplification which will be useful in getting started. There is nothing in the data structure, however, which would preclude occasional merging of strings into a longer string, saving repetitious processing later on (Barry Clark's suggestion). If strings are kept sufficiently long (say, 2 minutes' data for 351 baselines, or 10 minutes' data for the interim), then merging of the 700 or fewer strings with, say, a merge order of 10, would only require 3 passes through the data, with sequential binary access. Especially now, the strings are apt to be short and few in number, so this is not regarded as a major inconvenience. Merging of data records containing, say 450 12-word blocks (5 min. of 10 sec. records from 15 baselines), for 3 passes, takes .3 to .5 sec./merge string, or 45 to 70 sec. for 12 hours' data.

Extracting data from (u,v) files in baseline-time order would proceed as follows: After the first index record for the source is found, the time is compared. If the time interval for the string overlaps the time interval desired, the entire sorted string is retrieved. (u,v) for the baselines are computed for the start time of the string, and used to help search the data records for all pertinent baselines and times to be extracted. Then the source link is pursued to the following time interval and the process repeated.

Listing of a file in strict time order is simply a matter of reading the index and visibility data in the order they were stored.

6. Timetable

It is proposed to start coding FILLER as soon as a data structure design is agreed upon. We estimate about 2 man-months of effort would be required, to be accomplished by Dave Ehnebuske and Jerry Hudson. During the same time, Dave will be working closely with Al Braun in creating necessary CANDID operators and routines for generating skeleton data structures and accessing them. Following completion of FILLER, it is estimated that the other routines would require another man-month of effort, principally by Jerry Hudson.

APPENDIX

File Headers for External Storage

All files used for data storage by the above programs are also organized as CANDID external storage files. The first few physical records on each file contain a description of the contents, which represent a CANDID structure or array. The same information is essential to non-CANDID access routines, since knowledge of the data organization is necessary for them, even if only to know the logical record size.

Two types of pointer are utilized in the file header: word and byte (9-bit). The zero pointer in either case refers to physical block zero, word (or byte) zero within that block. Since the first writable (by the user) block is numbered physical block 1, the word pointer to the first word in the file is 128; its byte pointer is 512. Conversion from pointer to physical block thus means dividing by 128 or 512, as appropriate; the remainder numbers the word or byte.

The first six words in every file hold word pointers out to the data origin, and to various file descriptors. These are organized as follows:

word pointer	contents	
128	DOWP	Data origin word ptr.
129	SYMTWP	Symbol table word ptr.
130	MOLTWP	Molecule table word ptr.
131	MOLTWL	# words in mol. table
132	ATMTWP	Atom table word ptr.
133	ATMTWL	# words in atom table

Macro names referring to the pointers are defined in EXTFIL.DCL [11,12]. DOWP always points to the top of a physical block. The symbol table is described below. The molecule and atom tables are essentially lists of standard data aggregates and primitives in use at the time the file was generated, and which are referred to by the symbol attribute word. Examples of molecules: vectors, matrices. Examples of atoms: real numbers, complex numbers.

The symbol table is organized as follows:

word pointer	contents	
SYMTWP+0	EFST.LENGTH	Length of this area
+1	EFST.PTO	Offset for arry hdr

```
            +2    EFST.LINK    Link to other symbols, if
                               this is a data structure.
            +3    EFST.ATTR    Symbol attributes
            +4    EFST.ESIZE   Length of arry elt (bytes)
            +5    EFST.SYMLEN  Length of name (chars)
            +6    EFST.SYMBOL  Name, 7-bit ASCII
                          .  .  .
+EFST.PTO                      Array Header
```

Strictly speaking, EFST.PTO may not point to the array header, but instead to the datum itself, if the datum is not dimensioned as an array. In all files that concern us, however, it will be an array header offset, relative to the beginning of the symbol table. The link contains a word offset within the symbol table where the next symbol can be found (if the file contains a CANDID data structure). The attribute word contains such information as the atomic and molecular type for the symbol, whether it is the node of a data structure, and whether it is dimensioned. The last of these is perhaps the only information of interest to non-CANDID programmers; bit 33 will be set (mask with '000000000004) for arrays. The array header is organized as follows:

word	contents
HDR + 0	Total size of array (bytes)
+ 1	Virtual origin (bytes)
+ 2	Number of dimensions
+ 3	(Upper bound 1 (leftmost)
+ 4	< Lower bound 1 (leftmost)
+ 5	(Multiplier 1 (leftmost)
+ 6	(Upper bound 2
+ 7	< Lower bound 2
+ 8	(Multiplier 2

. . .

The byte address for subscripts $(i_1, i_2, \ldots, i_n)$ is calculated to be

$$V.O. + \sum_{i=1}^{n} Mult_i \cdot Subscr_i$$

For the files described above, a singly-dimensioned array suffices to describe the structure.