VLB ARRAY MEMO No. 588

(870109)

OBSERVER'S INTERFACE TO THE VLBA

Dale King

7 January 1987

This memo discusses the observer's interface to the VLBA control system as it is currently programmed, which should be very nearly the final state. Previous memos have presented details of the syntax of the observer's input text stream (VLBA Memo #549), and of the data structure which is loaded from that text (VLBA Memo #505). This memo summarizes that information, discusses changes and corrections to those memos, and provides additional, detailed information about each observer-enterable item.

SECTION 1

The observation description file

The observer's desired sequence of operations will be called his observing schedule. This schedule is communicated to the control computer in the form of an ordinary, human readable text stream. The text is normally prepared in advance and stored as an observation description file.

Each individual observation within the schedule is represented in the control computer by a data structure which contains all of the information necessary to identify and control that specific observation. The data structure is sometimes called an "observ block," where "observ" (no "e" at the end) is from the programmer's name for the data structure. By analogy, a segment of the text stream which contains all of the information needed to load one "observ" data structure is also called a block.

The text consists of name and value pairs which are organized into observation blocks by metacommands.

The name and value pairs have two basic forms:

```
        name = value   [for unsubscripted items];
    and   name = (subscript1, value1), (subscr2, value2), ... .
```

The syntax of these pairs is more fully described in VLBA Memo 549, and in the documentation for module LDSTRUCT. (Documentation is available in VAX3::[VLBSOFT.DOC].) This syntax has been designed to be easy to read, easy to write, and to allow almost any reasonable punctuation.

Metacommands are delimited by exclamation points ("!") at beginning and end. This is the only legal use of the exclamation point within the observation description file, as currently programmed. There are currently five metacommands:

!NEXT! Ends the current observation block. Any name-value pairs after this metacommand will be loaded into the next block.

!BEGIN LOOP! The current block is designated as the start of a group of observations which are to be executed repeatedly.
!LOOP BACK! The current block is the last in the loop group. Loops may not be nested. See section 3 for more details.
!QUIT! End of this schedule. As currently programmed, anything in the file beyond !QUIT! will be ignored.
!* User-entered comment *!

Metacommands must be in upper case. They may be abbreviated. If the terminating "!" is missing, the end of the line terminates the metacommand. Comments, delimited by blanks or asterisks, may be embedded within the metacommands; for example, !BEGIN pointing calibration LOOP!.

The user need not specify every control parameter for every observation block within the schedule. All items which are unspecified will default to their value in the previous block, except for the comments in OBSTXT, which are cleared. The defaults for the first block are mostly zeros. An important exception is the declination, whose initial default is minus 90 degrees, which should cause the control system to complain if one actually attempts to use it. Other initial defaults are given in Appendix C.

SECTION 2

Description of user-enterable items in the observing schedule

Following are the index number for each item, a brief description of the item, the name of the item, the "minimum matching" name (i.e.: the shortest abbreviation of the name that can be used) for the item, a description of the expected input, and a description of the stored value, including its C data type. Subscript ranges are noted, where relevant. Appendix B contains an alphabetic list of the item names.

In the descriptions of the expected inputs, "decimal" means an optionally signed digit string containing an optional decimal point and/or an optional exponent. "Integer" means an optionally signed digit string without decimal point or exponent. "Hexadecimal" means an unsigned integer string, which will be interpreted as decimal; or a string starting with "$", "0x", or "0X" and containing the digits and letters 'a' thru 'f' or 'A' thru 'F', which will be interpreted as hexadecimal.

In the descriptions of the stored values, "short," "long," "float," and "double" have compiler and machine dependent meanings. For our Motorola MC680x0-based machines, and also the VAX, "short" indicates 16-bit integers, while "long" means 32-bit integers. "Float" means single precision floating-point (number and exponent packed into one 32-bit datum). "Double" normally indicates double precision floating-point, but with our current C compiler for the Motorola "double" is also 32 bits, indistinguishable from "float." We retain the programming distinction between double and float in case we should change compilers.

("int" is very compiler-dependent, and has been avoided in this memo. We will try to continue to avoid it in future documents. In our current C compiler for the Motorola, "int" is 16 bits, while on the VAX it is 32 bits. We hope to change the compiler for the Motorola to 32-bit "int"s in the near future.)

* An asterisk following the item name indicates that, while it is recognized in the input string, code to implement its usage will NOT be provided in the initial versions of the control program.

```
    case  1:     /*  Source Name
  SNAME          **    Input:   arbitrary string.
  SN             **    Stored:  String not longer than 12 chars,
                 **               nul-terminated if less than 12 chars,
                 **               truncated on right if more than 12 chars.

    case  2:     /*  Numeric Qualifier for Source Name
  QUAL           **    Input:   integer, arbitrary meaning.
  Q              **    Stored:  (long)  same meaning.

    case  3:     /*  2-character Qualifer for Source Name
  CALIB          **    Input:   arbitrary string.
  CA             **    Stored:  string not longer than 2 characters.

    case  4:     /*  Flux
  FLUX *         **    Input:   decimal, Janskys.
  FL             **    Stored:  (float)  Janskys.

    case  5:     /*  Right Ascension (J2000 epoch)
  RA             **    Input:   Any format accepted by FANGLE.
  RA             **    Stored:  (double)  Radians.

    case  6:     /*  Declination (J2000 epoch)
  DEC            **    Input:   Any format accepted by FANGLE.
  DE             **    Stored:  (double)  Radians.

    case  7:     /*  Derivative of RA for moving source.
  DRA            **    Input:   decimal, time-seconds per day.
  DR             **    Stored:  (float)  Radians per radian.

    case  8:     /*  Derivative of Dec for moving source.
  DDEC           **    Input:   decimal, arc-seconds per day.
  DD             **    Stored:  (float)  Radians per radian.

    case  9:     /*  Time of epoch for position of moving source.
  EPOCHT         **    Input:   time in format acceptable to FANGLE.
  EPOCHT         **    Stored:  (float)  Radians (!!) (24 hr = 2*pi radians)

    case 10:     /*  Date of epoch for position of moving source.
  EPOCHD         **    Input:   Year, Month, Day; in format for FDATE.
  EPOCHD         **    Stored:  (long)  Modified Julian day-number.
```

```
    case 11:       /* Diurnal Parallax
DPARAL *           **   Input:   decimal, arc-seconds.
DP                 **   Stored:  (float)  Radians.

    case 12:       /* Feed collimation error in Azimuth.
AZCOLIM            **   Input:   decimal, arc-minutes (NOT min, sec).
AZC                **   Stored:  (float)  Radians.

    case 13:       /* Feed collimation error, elevation
ELCOLIM            **   Input:   decimal, arc-minutes (NOT min, sec).
ELC                **   Stored:  (float)  Radians.

    case 14:       /* Lateral focus pointing correction, azimuth.
AZLAT *            **   Input:   decimal, arc-minutes (NOT min, sec).
AZL                **   Stored:  (float)  Radians.

    case 15:       /* Lateral focus pointing correction, elevation.
ELLAT *            **   Input:   decimal, arc-minutes (NOT min, sec).
ELL                **   Stored:  (float)  Radians.

    case 16:       /* Subreflector focus setting.
FOCUS              **   Input:   decimal, centimeters.
FOC                **   Stored:  (float)  centimeters.

    case 17:       /* Subreflector rotation setting.
ROTATION           **   Input:   decimal, degrees (NOT deg, min, sec).
RO                 **   Stored:  (float)  degrees (NOT converted to radians).

    case 18:       /* Subreflector focus change with position.
RFOCUS *           **   Input:   decimal, centimeters.
RF                 **   Stored:  (float)  centimeters.

    case 19:       /* Subreflector rotation change with position.
RROTATION *        **   Input:   decimal, degrees (NOT deg, min, sec).
RR                 **   Stored:  (float)  degrees (NOT converted to radians).

    case 20:       /* 2-12 GHz Synthesizer setting.
SYNTH              **   Input:   decimal, GHz.
SY                 **   Stored:  (float)  Mhz (note units conversion).
                   Subscript range:  1 - 2 (there are two synthesizers).
```

NOTE for items 21 to 32: The functions of the specific bits in these control words have not yet been defined. Please refer to the documentation for the various devices, when available.

```
    case 21:       /* Local Oscillator transfer switch.
LOXFER             **   Input:   Hexadecimal, bit settings.
LO                 **   Stored:  (short)  bit settings.
```

```
    case 22:     /*  Noise Source control.
NOISE            **    Input:   Hexadecimal, bit settings.
NO               **    Stored:  (short)  bit settings.

    case 23:     /*  Coherent calibration control.
PCAL *           **    Input:   Hexadecimal, bit settings.
P                **    Stored:  (short)  bit settings.

    case 24:     /*  Front End control.
FECNTRL          **    Input:   Hexadecimal, bit settings.
FEC              **    Stored:  (short)  bit settings.

    case 25:     /*  I-F selector switch control.
IFSEL            **    Input:   Hexadecimal, bit settings.
IFS              **    Stored:  (short)  bit settings.
                 Subscript range:  1 - 4 (there are four I-Fs).

    case 26:     /*  I-F distributor control.
IFDISTR          **    Input:   Hexadecimal, bit settings.
IFD              **    Stored:  (short)  bit settings.
                 Subscript range:  1 - 4 (there are four I-Fs).

    case 27:     /*  Formatter control.
FORMAT           **    Input:   Hexadecimal, bit settings.
FOR              **    Stored:  (short)  bit settings.
                 Subscript range:  1 - 2 (there are two formatters).

    case 28:     /*  Tape transport control.
TAPE             **    Input:   Hexadecimal, bit settings.
TA               **    Stored:  (short)  bit settings.
                 Subscript range:  1 - 2 (there are two tape drives).

    /****  Cases 29 - 32 reserved for renaming of elements of 'miscsw'  ****/
       ('miscsw' is a 4-word array reserved for additional control words.)

    case 33:     /*  Number of recorded channels.
NCHAN            **    Input:   integer in range 0 to 16.
NC               **    Stored:  (short)  number of channels.


       NOTE for items 34 to 43:  These items are all subscripted, and
       the meaningful subscript range is from 1 to NCHAN.  However, the
       program only checks these subscripts against the maximum allowed
       value of NCHAN (currently 16), not the actual value in NCHAN.

    case 34:     /*  Baseband Synthesizer setting.
BBSYNTH          **    Input:   decimal, MHz, in range 500 to 1000.
BBS              **    Stored:  (float)  MHz.
                 Subscript range:  1 - NCHAN.
    case 35:        /*  Bits per sample.
BITS                **    Input:   integer from set {1,2}.
BI                  **    Stored:  (short)  number of bits per sample.
                    Subscript range:  1 - NCHAN.

    case 36:        /*  Clock Divisor.
CLOCK               **    Input:   integer from set {1, 2, 4}
CL                  **    Stored:  (short)  clock divisor.
                    Subscript range:  1 - NCHAN.

    case 37:        /*  Baseband Filter selection.
BBFILTER            **    Input:   hexadecimal in range 0 to 8, filter number.
BBF                 **               Bandwidth = 16 MHz / 2**(filter number).
                    **    Stored:  (short)  filter number.
                    Subscript range:  1 - NCHAN.

    case 38:        /*  Track Assignment of channel.
TRACK               **    Input:   ===> NOT YET DETERMINED <===
TR                  **    Stored:  (short)  ===> ??? <===
                    Subscript range:  1 - NCHAN.

    case 39:        /*  Sideband selector.
SIDEBAND            **    Input:   integer from set {-1, 1}.
SI                  **    Stored:  (short)  sideband code.
                    Subscript range:  1 - NCHAN.

    case 40:        /*  Baseband Mixer selector.
BASEBAND            **    Input:   integer in range 1 to 16.
BA                  **    Stored:  (short)  mixer code.
                    Subscript range:  1 - NCHAN.

    case 41:        /*  I-F channel.
IFCHAN              **    Input:   ===>  NOT YET DETERMINED  <===
IFC                 **    Stored:  (short)   ===>   ???   <===
                    Subscript range:  1 - NCHAN.

    case 42:        /*  Front End code.
FE                  **    Input:   string giving wavelength of receiver:
FE                  **               "4m", "90cm", "50cm", "20cm", "13cm",
                    **               "6cm", "4cm", "3cm", "2cm", "1.3cm",
                    **               "8mm", or "4mm".
            **               "13mm" may be used for "1.3cm".
                    **    Stored:  (short) code [0 - 15] corresponding to rcvr.
                    Subscript range:  1 - NCHAN.

    case 43:        /*  Level control.
LEVEL               **    Input:   Hexadecimal, in range 0 to 0xff (0 to 255).
LE                  **    Stored:  (short)  level setting.
                    Subscript range:  1 - NCHAN.
    case 44:        /*  Termination time for current observation.
NEXTSTOP            **     Input:   time in format acceptable to FANGLE.
NEXTS               **     Stored:  (float)  Radians (!!) (24 hr = 2*pi radians).

    case 45:        /*  Termination date for current observation.
NEXTDAY             **     Input:   date (yr, mon, day) in FDATE format.
NEXTD               **     Stored:  (long)  Modified Julian day-number.

    case 46:        /*  Termination time for loop.
LASTSTOP            **     Input:   time in format acceptable to FANGLE.
LASTS               **     Stored:  (float)  Radians (!!) (24 hr = 2*pi radians).

    case 47:        /*  Termination date for loop.
LASTDAY             **     Input:   date (yr, mon, day) in FDATE format.
LASTD               **     Stored:  (long)  Modified Julian day-number.

    case 48:        /*  Length of time to observe current block.
DURATION            **     Input:   time in format acceptable to FANGLE.
DU                  **     Stored:  (float)  Radians (!!) (24 hr = 2*pi radians).

    case 49:        /*  Text area for comments, or whatever.
OBSTXT              **     Input:   Arbitrary sting.
O                   **     Stored:  Nul-terminated string shorter than 256 chars.

    case 50:        /*  Termination date for loop, or current observation if
DATE                **     not in loop.
DA                  **  (Sets both NEXTDAY and LASTDAY to the value-string.)
                    **     Input:   date (yr, mon, day) in FDATE format.
                    **     Stored:  (long)  Modified Julian day-number.

    case 51:        /*  Termination time for loop, or current observation.
STOP                **  (Sets both NEXTSTOP and LASTSTOP to the value-string.)
ST                  **     Input:   time in format acceptable to FANGLE.
                    **     Stored:  (float)  Radians (!!) (24 hr = 2*pi radians).
```

SECTION 3

More about loops

In the context of an observing schedule, a loop is a sequence of observations which is executed repeatedly until a specific date and time (in the sense of "time-of-day") has been exceeded. Loops may not be nested.

The first observation in the loop is designated by including the metacommand !BEGIN LOOP! in the text block which describes that observation. The metacommand may appear anywhere within the text block, but it is most readable if it is at the beginning of the block.

Similarly, the last observation in the loop is designated by the metacommand !LOOP BACK!, which is most readable if it appears at the end of its block.

Since each observation in the loop will be executed more than once, the items NEXTSTOP and NEXTDAY are not appropriate for designating the end of the observation. DURATION should be used to specify the length of time to observe. (During execution, the computer will actually use DURATION to constuct a new NEXT{STOP,DAY} pair for each iteration.)

Execution of the loop terminates when the date and time specified in LASTSTOP and LASTDAY has been reached or exceeded. As currently programmed, LAST{STOP,DAY} must be specified in the last block of the loop (that is: the block designated by !LOOP BACK!).

SECTION 4

Example of observing file (courtesy of B. Clark)

This is almost the same example as in VLBA Memo 549. Typographic errors have been corrected, and changes have been made to correspond to the actual program and data structure. This file has been run through the current version of the load-schedule program. The result was a set of correctly loaded 'observ' data structures in the global common area of memory.

```
!*  Setup for four recorded channels *!

nchan=4
fe = (1,6cm),(2,6cm),(3,6cm),(4,6cm)          !* 6cm front end *!
ifchan = (1,1),(2,1),(3,1),(4,1)              !* IF channel (RCP) *!
clock = (1, 4),(2, 4),(3, 4),(4, 4)           !*  4 MHz sample rate *!
baseband = (1,1),(2,1),(3,2),(4,2)            !* Two baseband channels, *!
sideband = (1,1),(2,-1),(3,1),(4,-1)          !* upper and lower sidebands *!
bbfilter = (1, 1),(2, 1),(3, 1),(4, 1)        !* 8 MHz BW all channels *!
bbsynth = (1,600),(2,600),(3,632),(4,632)     !* 64 MHz contiguous recorded *!
bits = (1,1),(2,1),(3,1),(4,1)                !* One-bit recording *!

!* Setup for receiver *!

azcolim = 1.3, elcolim = 0.85                 !* Collimation, arcminutes *!
focus = 11.7, rotation = 177.5                !* Focus rotation settings *!
synth = (1,4.00)                              !* A synthesizer 4 GHz *!

!* Begin observing schedule *!
```

```
date = 88aug08
sname =  3C286
qual = -1
ra = 13h28m53.287s                           !* J2000 coordinates *!
dec = 34d08'22.23"
flux = 12.3
stop = 12h00m                                !* Observe until this UT *!

!NEXT!

!BEGIN LOOP!
laststop = 17h24m                              !* loop termination time  *!
sname = 3C274
ra = 12h28m17.263, dec=12d22'17.8"
    duration = 20m                           !* 20 minutes on this source *!
flux = 75

!NEXT!
sname = 2C273
ra = 12h27m18.345s, dec=02d18'05.7"
duration = 2m                                !* and 2 minutes on this one *!

!LOOP BACK!

!NEXT!

!QUIT!
```

APPENDIX A

Changes and extensions to previous memos.

There have been a few changes to the 'observ' structure, as defined in file OBSERV.H. (The current version of all program files mentioned in these memos may be found in VAX3::[VLBSOFT.CODE]).

Member 'thisday' (data type long) has been ADDED after 'thistime'.
Member 'thisstop' has been RENAMED to 'nextstop'.
Member 'nextday' (data type long) has been ADDED after 'nextstop'.
Member 'thisduration' has been RENAMED to 'duration'.

A four-character string has been added to the beginning of the structure to contain the version number of the structure definition. This item is not available for observer entry. However, the user may wish to check it at data-reduction time, to ensure that the reduction programs are properly matched to the data.

Various array sizes and status bits are now defined symbolically, so that programs can avoid 'hard-wired' numerical constants.

The text stream handler recognizes two names which are not defined in OBSERV.H. These serve as convenient synonyms for two common multiple operations.

'STOP' loads both 'nextstop' and 'laststop' with the same stop time, as extracted from the (single) value-string.
'DATE' loads both 'nextday' and 'lastday' with the same date.

APPENDIX B

Alphabetic list of observer-enterable items.

This appendix contains no new information. It reorganizes the information in Section 2 for more convenient reference.

"Minmatch" is the minimum set of characters which uniquely identify the name. If the user enters more than this minimum, the additional characters must be correct.

"Index" is the position of the item within the namelist used by the name-matching function. The observer need not be concerned with the index; it is transparent to him. It is presented here for programmer convenience.

Name	Minmatch	Index	Description
AZCOLIM	AZC	12	Feed collimation error in azimuth.
AZLAT	AZL	14	Lateral focus pointing correction in azimuth.
BASEBAND	BA	40	Baseband mixer selector.
BBFILTER	BBF	37	Baseband filter selector.
BBSYNTH	BBS	34	Baseband synthesizer setting.
BITS	BI	35	Bits per sample.
CALIB	CA	3	2-character qualifier for source name.
CLOCK	CL	36	Clock divisor.
DATE	DA	50	Loads both NEXTDAY and LASTDAY with same value.
DDEC	DD	8	Derivative of declination for moving source.
DEC	DE	6	Declination (J2000 epoch).
DPARAL	DP	11	Diurnal parallax.
DRA	DR	7	Derivative of right ascension, moving source.
DURATION	DU	48	Length of time to observe current block.
ELCOLIM	ELC	13	Feed collimation error in elevation.
ELLAT	ELL	15	Lateral focus pointing correction in elevation.
EPOCHD	EPOCHD	10	Date of epoch for position of moving source.
EPOCHT	EPOCHT	9	Time of epoch for position of moving source.
FE	FE	42	Front-end code.
FECNTRL	FEC	24	Front-end control.
FLUX	FL	4	Flux.
FOCUS	FOC	16	Subreflector focus setting.
FORMAT	FOR	27	Formatter control.
IFCHAN	IFC	41	I-F channel.
IFDISTR	IFD	26	I-F distributor control.
IFSEL	IFS	25	I-F selector switch control.
LASTDAY	LASTD	47	Termination date for loop.
LASTSTOP	LASTS	46	Termination time for loop.
LEVEL	LE	43	Level control.
LOXFER	LO	21	Local-oscillator transfer switch.
miscsw	-none-	29-32	(Reserved for additional control words.)
NCHAN	NC	33	Number of recorded channels.
NEXTDAY	NEXTD	45	Termination date for current block.

NEXTSTOP	NEXTS	44	Termination time for current block.
NOISE	NO	22	Noise-source control.
OBSTEXT	O	49	Text area for comments, etc.
PCAL	P	23	Coherent calibration control.
QUAL	Q	2	Numeric qualifier for source name.
RA	RA	5	Right ascension (J2000 epoch).
RFOCUS	RF	18	Subreflector focus change with position.
ROTATION	RO	17	Subreflector rotation setting.
RROTATION	RR	19	Subreflector rotation change with position.
SIDEBAND	SI	39	Sideband selector.
SNAME	SN	1	Source name.
STOP	ST	51	Loads both NEXTSTOP and LASTSTOP.
SYNTH	SY	20	2 - 12 GHz synthesizer setting.
TAPE	TA	28	Tape transport control.
TRACK	TR	38	Track assignment of channel.

APPENDIX C

Initial Defaults

Before a user's observing schedule is processed, the internal data structure, or "block," which will control the first observation in the schedule is initialized to a standard set of defaults. The information in the user's observation description file modifies these values. Any item which is not specified by the user will remain set to its initial default block (except for the text in OBSTXT, which is nulled). Thus, if the user never declares a value for an item, its initial default will be used throughout the entire observing schedule.

Following is a list of all non-zero initial defaults.

SNAME and CALIB are set to all blanks (no nul terminator).

OBSTXT is set to nul (actually, all characters are set to zero).

DEC = -1.57 (minus 90 degrees) -- will cause antenna control to generate a complaint if execution is attempted.

SYNTH(1) = SYNTH(2) = 4.85 GHz.

NEXTDAY = LASTDAY = 100000 -- big number causes infinitely long observation.

NCHAN = 4.

Even though the number of channels is initialized to 4, the channel-specific data is initialized for all 16 channels.

In the following, let I be the channel number, counting from 1 to 16. Then let

```
   J = (I - 1) if I is less than or equal to 8, and
   J = (I - 9) if I is greater than 8.
(I.e.:  J runs from 0 to 7, then from 0 to 7 again.)
```

```
  BBSYNTH(I)   = (500 + (J * 16))   (MHz)
  TRACK(I)     = J
  BASEBAND(I)  = (I - 1)
```

And, for all channels:

```
  BITS = CLOCK = SIDEBAND   = 1
  BBFILTER = IFCHAN = LEVEL = 0
  FE = "3CM".
```

APPENDIX D

Allowed ranges of various items

```
/*    Baseband synthesizer (MHz)   */
#define BBSYN_LO    500.0
#define BBSYN_HI   1000.0
/*    Baseband selector            */
#define BBSEL_LO    1
#define BBSEL_HI   16
/*    Baseband filter numbers      */
#define BBFIL_LO    0
#define BBFIL_HI    8
/*    Level Control                */
#define LEVEL_LO    0
#define LEVEL_HI 0xFF
/*    Synthesizer (GHz)            */
#define SYNTH_LO    2.0
#define SYNTH_HI   12.0
```

See also OBSERV.H for other size limits.