VEX Definition

Rev 2.0 August 1, 2021

Preface to the wiki version This document starts as a nearly direct merging of two documents (VEX File Definition/Example and VEX Parameter Tables) and conversion to Twiki wiki format. Along the way a very few typos were corrected and typewriter font has been employed to represent literal text. Subsequent additions in anticipation of an eventual VEX 2.0 standard have caused a version number bump to 1.99 and subsequently to larger numbers. Parameter tables with new options or new parameters have been colored blue for identification. Information to be deleted appears in a ~~strikethrough~~ font.

Links to Sections

~~$PASS_ORDER Block~~ (Not in VEX2)
$PHASE_CAL_DETECT Block
$PROCEDURES Block (Deprecated in VEX2)
~~$ROLL Block~~ (Not in VEX2)
$SCHEDULING_PARAMS Block
$SEFD Block
$SITE Block (Minor changes and augmented for VEX2)
$SOURCE Block (Augmented for VEX2)
$TAPELOG_OBS Block (Minor change for VEX2)
$TRACKS Block (Substantially changed for VEX2)

Comments on VEX2

VEX2 is an update of the original VEX (VEX1), developed to more logically accommodate the sampling-before-channelization (aka `IF sampling') paradigm represented by modern digital-backend (DBE) systems, as well as updates that allow better equipment description, setup and signal connections. VEX2 is not backward compatible with VEX1; obsolete features have been removed and some small changes in existing usage were made. Although normally we strive to maintain backward compatibility, in this case we believe that need to remove obsolete features of VEX1 as well as the advantages of not having two different ways of representing the same features outweigh the advantages of backward compatibility. As a result the transition period from VEX1 to VEX2 will allow both formats to be used while users switch to the new format.

In broad terms, the changes from VEX1 to VEX2 are as follows:

1. `$BBC` block: Unchanged for legacy systems, but somewhat different usage model for DBE-type systems.
2. `$BITSTREAMS` block: This block was used internally by the JIVE correlator for use Mark 5B recordings with VEX1 and is now included in VEX2 with extensions to allow specification of sample rates and more than one recorder
3. `$DAS` block: Substantial removal of old features and augmentation to allow detailed definitions of equipment, equipment settings, and signal connections between pieces of equipment. The latter capability allows a detailed specification of the signal path and equipment settings of the entire data-acquisition system from the IF to data recording/transmission subsystem.
4. `$DATASTREAMS` block: New $block in VEX2 to define VDIF data streams.
5. `$EXTENSIONS` block: A new block intended to provide a way to specify site/system-specific information and/or settings that are not otherwise covered in VEX.
6. `$FREQ` block: The sample_rate parameter has been moved to the `$TRACKS` block. This is consistent with the sample rates being specified in the `$BITSTREAMS` and `$DATASTREAMS` block for those recording formats. This makes the approach in VEX2 consistently that the sample rate and bits per sample and type (real/complex) are always specified in the recording format blocks.
7. `$IF` block: Augmented to specify IF sample rate and remove Physical Connection. Also added optional parameters to further specify IF configuration at station.
8. `$PROCEDURES` block: Deprecated.
9. `$SCHED` block: Significantly augmentated to provide new features, particularly `intent` statements.
10. `$SOURCE` block: Augmented to provide new non-sidereal source types.
11. `$TRACKS` block: Many obsolete features were removed and the `sample_rate` parameter for recording Mark IV data was moved here.
12. `$HEAD_POS`, `$PASS_ORDER`, `$ROLL` blocks were deleted.
13. VEX2 is significantly different from VEX1 in that it allows much more flexibility in the number of back-ends and recorders and how they are interconnected. While VEX2 allows virtually any details of the configuration and

wiring from the IF to the recording to be described, it is expected that this flexibility will only be used to specify necessary details of the systems that are not normally static. Few details need to be specified for most systems. In fact it is possible that systems are described adequately by a "virtual" configuration with no details of the physical system in use. What needs to be described is system specific.

14. The following parameters exist in both VEX1 and VEX2, but the VEX2 version is not backward compatible with VEX1:
    - pointing_sector
    - chan_def
    - if_def
    - station
    - sample_rate (changed blocks)
    - fanout_def

Known Software Limitations

There are some known software limitations with existing VEX implementations that affect which VEX files can be used operationally. While these are outside of the definition of VEX, they are collected in the known software limitations document for reference.

Overall Organization

1. The first line in the VEX file identifies the file type and must be of the form `VEX_rev =` *Revision* `;` followed by an optional comment (prefaced with a '*').
2. Except for the first line, the active (i.e. non-comment) statements in a VEX file consist entirely of blocks separated (and named) by *$BlockName* statements. For convenience, these are referred to as $blocks or sections.
3. The *$BlockName;* statement is the first statement in a $block and names the $block.
4. A $block is terminated by the specification of another *$BlockName;* statement or the end of the file.
5. *$BlockName* may be comprised of characters from the 'Legal Character Set' (see 'Syntax Rules' below).
6. $blocks may be in any order, though they are normally organized for convenience and readability.
7. Additional $blocks may be added as necessary.
8. The maximum length of $blockname is specified under 'Syntax Rules' below.

$block Types

$blocks are divided into three types:

1. 'Primitive' $blocks - each primitive $block serves to define a set or sets of parameters. Most blocks in a VEX file are primitive. They consist entirely of keyword 'def' blocks, which define low-level station, source, and recording parameters. No `ref` statements to other blocks are permitted except to pick up parameters from an external file (`ref` *externalFilename* `:` *$blockName* = *keyword* `;`). Links, however, are sometimes required between blocks (for example, between `$FREQ`, `$BBC`, `$IF` and `$TRAKS` blocks). Each primitive block may contain *parameter* = *values* statements only for parameters which are native to that block. A list if primitive $blocks in given in the next section.
2. 'High-level' $blocks - made up entirely of references to primitive-block keywords to construct the full suite of parameters necessary to specify an experiment. The $GLOBAL, $STATION, and $MODE blocks are the currently-defined 'high-level' blocks.
    - $GLOBAL block: Specifies global/general experiment parameters as 'refs' to primitive-block keywords.

    - $STATION/$MODE blocks: Define station and mode keywords in 'def' blocks which contains 'refs' to primitive-block keywords. The combination of a $STATION key, $MODE key, plus the $GLOBAL parameters specify the detailed configuration for an observation at a particular station.
3. $SCHED block - entries necessary to drive the observation schedule. Has a special format and includes both parameters and keyword references to $STATION, $MODE and $SOURCE blocks. This block specifies a detailed time-ordered list of observations. (Time-ordering requirement may be removed in future.)

Additional $block types may be added for other or special purposes. The order of the $blocks is irrelevant except for convenience and readability. In this definition document, the numerous primitive $blocks are simply ordered alphabetically.

Primitive $blocks

The use list of primitive $blocks and their usage is given in the following table.

$block	Contents	Scheduling	Data-taking	Correlation
$ANTENNA	antenna parameters	X		X
$BBC	BBC/IF assignments	X	X	X
$BITSTREAMS	VSI-H recording details	X	X	X
$CLOCK	clock-synchronization model for correlation			X
$DAS	data-acquisition system information	X	X	X
$DATASTREAMS	VDIF recording details	X	X	X
$EOP	earth-orientation information			X
$EXPER	general experiment information	X	X	X
$EXTENSIONS	institution controlled extensions	X	X	X
$FREQ	channel frequencies, bandwidths, sample rates, etc.	X	X	X
$PHASE_CAL_DETECT	phase-cal frequencies to be detected at station	X	X	~~X~~
$IF	IF bands/sidebands, 1st LO freq, phase-cal freqs	X	X	X
$SEFD	SEFD info	X		
$SITE	antenna site details	X		X
$SCHEDULING_PARAMS	parameters for scheduling program	X		
$SOURCE	source names, positions, etc.	X	X	X
$PROCEDURES	general-procedure timing parameters	X	X	
$TRACKS	Mark IV recording multiplex details	X	X	X
$TAPELOG_OBS	'As-observed' VSN information			X
~~$HEAD_POS~~	~~headstack-positioning information~~	~~X~~	~~X~~	~~X~~
~~$PASS_ORDER~~	~~tape pass order~~	~~X~~	~~X~~	~~X~~
~~$ROLL~~	~~recording barrel-roll details~~	~~X~~	~~X~~	~~X~~

Additional blocks that may be defined some day include:

$block	Contents
$ANTENNA_CAL_OBS	'As-observed' antenna calibration measurements
$CABLE_CAL_OBS	'As-observed' cable-calibration measurements
$CLOCK_OBS	'As-observed' clock-sync measurements
$CORR	Correlation parameters
$SCHED_OBS	'As-observed' schedule
$PHASE_CAL_OBS	'As-observed' phase calibration measurements
$TSYS_OBS	'As-observed' system-temperature measurements
$WX_OBS	'As-observed' weather measurements

Syntax

General

1. Upper/lower case is relevant in VEX statements.
2. All VEX statements start with *$blockName*, `def`, `enddef`, `ref`, `scan`, `endscan`, `start_literal()`, `end_literal()` or *parameter*.
3. VEX punctuation characters are:
    - `=` - assignment of a keyword or parameter value
    - `:` - deliminator between fields in the assignment of parameter value(s).
    - `;` - terminates a VEX statement
    - ' ' - (white space) in special cases. See White Space Rules below.
4. Multiple statements may occur on a single line.
5. A single statement may span multiple lines.
6. The maximum length of block names (after the leading `$`), link names (after the leading `&`), parameter names, parameters values, keywords, comments (including the leading *), and file names is 128 characters.
7. All blocknames (and only blocknames) begin with the `$` character.
8. All link words (and only link words) begin with the `&` character.
9. White Space Rules
    - No white space is permitted within block names, keywords, file names, parameter names or values **except**: at least one white space must separate a numerical parameter value from its units (e.g., `10 sec`), and string values may include white space when enclosed by quotes (see below).
    - At least one white space must follow `def`, `ref` or `scan` (e.g., `ref $FREQ`).
    - No white space (or anything else) may occur before the `VEX_rev` statement at the beginning of the VEX file.
    - With the above exceptions, an arbitrary amount (including zero) of white space is allowed anywhere within a VEX file.
    - For purposes of parsing, an end-of-line character is equivalent to one white-space character, except that an end-of-line unconditionally terminates a comment.
10. There are no column specifications in the VEX format, other than those imposed by the discipline of the user for easy readability. A 'literal block', however, may have column restrictions/specifications which fall outside of the VEX format specification.

11. Keywords, link names, and scan_ID's are arbitrary and do not contain information. If is helpful if they are descriptive, but they are there just for bookkeeping. All actual information is in VEX parameter statements and the use of 'refs'.

Legal Character Set

1. The character set is confined to ASCII, except that the null character is prohibited anywhere, including any literal blocks, comments and quoted text. 7-bit ASCII is preferred, though not required, in VEX statements; comments may freely use 8-bit ASCII.
2. Block names, following the initial `$`, can include any characters except `\t\n;:=` and space (6 characters excluded).
3. Parameter names may contain any characters except `\t\n;:=` and space (6 characters excluded), and in addition may not start with any of `*$&"`. (`*` starts a comment; `$` starts a block name; `&` starts a link name; `"` surrounds a quoted string.) There are some excluded strings as well: `ref`, `def`, `enddef`, `scan` and `endscan`.
4. Parameter values and keywords may contain any characters except `\t\n;:=&*$"` and space (10 characters excluded).
5. Link names, following the initial '&', may contain any characters except `\t\n;:=&*$` and space (9 characters excluded).
6. Quoted character strings are allowed as field values in parameter-assignment statements, except for the first field of the `VEX_rev` parameter. However, such use should be avoided when possible. Quoted character strings are allowed, but ~~not encouraged~~ are strongly discouraged, as ~~block names,~~ link names (following the initial ~~=$= or~~ `&` ~~, parameter names~~ and keywords.
7. Quoted character strings must start and end with a double-quote, `"`. Quoted strings follow the usual C quoting conventions for character strings, except that a null character may not be specified. The length of a quoted string is the length of the resulting characters, not the quoted representation.
8. Literal blocks fall outside of the normal VEX construction rules and may contain any characters except null.

Comments

1. Any non-quoted asterisk, `*`, begins a comment. A comment is terminated at the first trailing new-line. A comment may be preceded by optional white space.
2. A comment may not be embedded within a VEX statement.
3. A comments may contain any ASCII character except null.
4. For purposes of parsing, a comment is considered to be equivalent to a single space character.
5. A point of philosophy: Software should strive to retain comments whenever a VEX file is processed. (If optional white space before a comment contains any new-lines, software should retain exactly one new-line.)
6. The maximum length of any comment is 128 characters, including the leading asterisk.

`def` Construct

Each `def` block begins with a line such as

`def` *keyword* `;`

And ends with

`enddef;`

In between, statements such as the following are allowed:

1. *parameter* = *values* ; (Direct setting of parameter values; permitted only in primitive $blocks)
2. ref *externalFilename* : *$blockName* = *keyword* ; (Set parameter values by reference to external source; permitted only in primitive $blocks; not permitted in schedule files for observing, which must be self-contained)
3. ref *$blockName* = *keyword* ; (Set parameters according to referenced def block; *$blockName* must refer to a primitive block; permitted only in $GLOBAL, $STATION and $MODE blocks)
4. ref *$blockName* = *keyword* : *qualifier* ; (Qualify the referenced def block parameters according to station, time, etc., only in $MODE -- see ref construct rules)

Notes:

1. The def block is used to assign a keyword name to a set of specified parameters values.
2. The def block is the ONLY way to specify parameter values.
3. def blocks are allowed in most $block sections, but specifically prohibited in the $GLOBAL and $SCHED blocks.
4. At least one space (or end-of-line character) must follow the word def.
5. *keyword* must be unique within the $block.
6. Any number of *parameter* = *values* constructs or ref statements may occur within the def.
7. The order of statements within a def is irrelevant.
8. A def without a corresponding ref may as well not exist.
9. The order of def blocks within a $block is irrelevant.

ref *externalFilename:$blockName* = *keyword* Construct

1. Permitted only within primitive def blocks to refer to lists of *parameter* = *values* statements from an external source.
2. *$blockName* is the $block name to be searched in the external file (Normally, this would be the same as the VEX file primitive *$blockName* in which this statement occurs).
3. *keyword* is a defined keyword in the *$blockName* block of the external file.
4. The syntax rules and construct rules for the external file are identical to the rules for primitive blocks.
5. The order of refs within a def is irrelevant.
6. At least one space (or end-of-line character) must follow the word ref.

ref *$blockName* = *keyword* [: *qualifier*] Construct

1. Permitted only within $GLOBAL, $STATION, $MODE blocks.
2. *keyword* is a defined def keyword within the specified primitive *$blockName*.
3. *qualifier* is an optional parameter that may be used only in the $MODE block to qualify a ref by station. *qualifier* may be a single $STATION def keyword, or may be multiple $STATION keywords separated by colons. Note: Current operational considerations require that **all** $MODE ref statements be qualified by the complete list of stations to which they apply. This is necessary so that useful consistency checking can be applied. In the future, checking may be re-written to relax this constraint.
4. Within a $GLOBAL, $STATION, or $MODE def block, there may be multiple ref statements to the same primitive block (with distinct keywords). In this case, the parameter values associated with the multiple ref statements are simply concatenated. No conflicting/duplicate parameters are permitted. The statements in this note have two important consequences:
    - Parameters (in particular for a mode) may be specified either globally, by station, or specific to particular mode. The last option may also be qualified by station. However for any given station in any given mode,

there cannot be conflicting/duplicate parameters from any source.
    - In interpreting the parameter information (including for a mode), no information is retained about which `def` the parameter was in or which block the `ref` statement that included was in. There is simply a list of parameters that apply.
5. The order of `ref` statements within a `def` or within the `$GLOBAL` block is irrelevant.
6. At least one space (or end-of-line character) must follow the word `ref`.

parameter = *values* Construct

1. Permitted only within `defs` defined within primitive blocks, except for special `VEX_rev` statement required as the first VEX statement in a VEX observation file.
2. The specification of any particular parameter is permitted only within the specific primitive block to which it is assigned (see table definitions below). A parameter cannot be used in more than one primitive block.
3. The number and type of fields must be consistent with the definition specified in the VEX parameter tables. A parameter can used more than once in a primitive block for different purposes as long as that does not create a conflict in meaning. All cases must use the same number of fields and/or types, including which are optional.
4. Fields within the *values* are separated by colon (`:`) characters. Each such colon may be surrounded by zero or more white spaces.
5. Each value must be of one of the following types:
    - **Real/Integer** Examples of acceptable values: -1.234, +5.67e-12, -.987E+04, 0.267e2, -871. Values may be followed by units specification if required. Unit specification, if present, must be separated by one or more spaces.
    - **Character** A character string may be specified either with or without surrounding quotes. As is usual in VEX, case is significant in all character strings.
        - If with quotes: All characters between the quotes, including white space, will be parsed as part of the character string (example: `"Character string"`). Legal characters are upper and lower case letters, numbers, and the following symbols: underscore (`_`), slash (`/`), minus (`-`) and plus (`+`). Normal C-language quoting conventions are observed, except that no null bytes may be present. Quoted text is not allowed in the first field of the `VEX_rev` parameter.
        - If without quotes: No embedded spaces are allowed. Character string is assumed to start with first non-space character and end with last non-space character before terminating colon or semi-colon.
    - **Keyword** A keyword value follows the same rules as a character value, but refers to the keyword of a `def`. This is currently used exclusively in the `$SCHED` block.
    - **Epoch** An epoch is specification of a particular instant of time is and is always expressed as UTC `xxxxyxxxdxxhxxmxx.xxxs` where `x` is a numeral, `xxxxy` is the four digit year, `xxxd` is day-of-year (1-366), `xxh` is hour (0-23), `xxm` is minute (0-59) and `xx.xxxs` are seconds (0-59.999..). Only the seconds sub-field may contain a non-integer (i.e., real) value. ~~The year may be either 4-characters (e.g., `1997y`) or 2-characters (e.g., `97y`).~~ Leading zeros within fields may be dropped. If appropriate, epoch may be truncated from the right (e.g., `1997y263d12h`) with implied 0's for unspecified fields.
    - **RA/Dec** Source position specified in ra/dec use an abbreviated form of units, of the form `ra=xxhxxmxx.xxxs; dec=xxdxx'xx.xxx";` (e.g., `ra=05h23m2.56s; dec=-20d45'12.2";`). Leading zeros of fields may be omitted, but zero fields (e.g,. `0h`), are required. A leading plus sign (+) is optional for positive declinations.
    - **&link** See Links section below.
6. Unit Specifications are provided as follows. Integer or real values with units of time, time rate, time acceleration, time jerk, frequency, sample rate, length, angle, angular rate, flux density, and byte count may require the explicit specification of units, as specified in the VEX Parameter Tables. Unit specifications follow, separated by one or

more spaces, the numerical value to which they refer. In cases where a parameter statement ends with a variable-length list of numerical values which take the same units, the first field of the list must have specified units, but the following values may omit them, causing them to default to the same units. [Note, however, for parameter statements with a fixed number of fields, units must always be specified. Thus, for example, `source_model` and `pointing_sector` require units for all relevant fields, but `horizon_map_az`, `ut1-utc` and `switching_cycle` do not.] The list of supported units follows:

- **time** 'psec', 'nsec', 'usec', 'msec', 'sec', 'min', 'hr', 'day', 'yr'
- **time rate** *time1/time2* where *time1* and *time2* are any valid time units with *time2* >= *time1*, units may be omited if *time1* = *time2*
- **time acceleration** *time1/time2^2* where *time1* and *time2* are any valid time units with *time2* >= *time1*
- **time jerk** *time1/time2^3* where *time1* and *time2* are any valid time units with *time2* >= *time1*
- **frequency** 'mHz', 'Hz', 'kHz', 'MHz', 'GHz'
- **sample rate** 'ks/sec', 'Ms/sec'
- **length** 'um', 'mm', 'cm', 'm', 'km', 'in', 'ft'
- **velocity** *length/time*, where *length* and *time* are any valid units
- **angle** 'mdeg', 'deg', 'amin', 'asec', 'rad' (exception - source position: see *RA/Dec* above ~~below~~)
- **angular rate** *angle/time*, where *angle* and *time* are any valid units
- **angular acceleration** *angle/time^2*, where *angle* and *time* are any valid angle and time units, respectively
- **flux density** 'mJy', 'Jy'
- ~~*bit density* 'bpi', 'kbpi'~~
- **byte count** `B', 'kB', 'MB', 'GB', 'TB', 'PB' (uses powers of 10)

Links

Links are cross-references made through the use of link words in the *values* part of various *parameter* = *values* statements. They serve to link together the myriad of details regarding station and recording configurations in the selected `def` blocks. There is, for example, heavy reliance on links between the `$FREQ`, `$BBC`, `$IF` and `$TRACKS` blocks in order to specify the detailed RF, IF, BBC and configurations. For clarity, link words require an `&` as the first character of their name, but are otherwise arbitrary using the normally-valid character set. Embedded spaces are not permitted. To understand the concept of links it may useful to examine the following $blocks in order: `$FREQ`, `$BBC`, `$IF`, `$FREQ`, `$TRACKS`.

Literal Block Construct

A block of text may be declared as literal if the block is preceded by a `start_literal();` statement and terminated by an `end_literal();` statement as follows:

`start_literal(` *string* `);`

first line of literal text

second line of literal text

...

`end_literal(` *string* `);`

Here *string* is optional (i.e. may be empty) and may be arbitrarily chosen to avoid any spurious interpretation of an embedded `end_literal()` string. *string* may not contain `)`, null, or new-line characters, but all other characters are

valid including white space.

Notes:

1. The `start_literal()` statement must be the last VEX statement on the line.
2. The `end_literal()` statement must be the first VEX statement on the line.
3. The literal block starts on the first line following the `start_literal` statement and ends on the last line before the `end_literal` statement.
4. Any ASCII characters except null are allowed within a literal block.
5. A literal block may only occur within a `def` block.
6. Parsing of a literal block is outside the VEX-format specification.
7. At this time, the literal block is used only in the `$SCHEDULING_PARAMS` block.

High-level $block Definitions

The $blocks that do not contain parameters: `$GLOBAL`, `$MODE`, and `$STATION` consist entirely of refs to primitive blocks. In the case of `$MODE` and `$STATION` these are contained in defs.

$GLOBAL Block

The purpose of the `$GLOBAL` block is to specify global/general information as appropriate. This block must follow the following rules:

1. Only `ref` statements to primitive-block keywords are permitted.
2. `ref` statements to any primitive $blocks are permitted and are concatenated with `ref` statements in the `$STATION` and `$MODE` blocks. No conflicting or duplicated parameters are permitted in the concatenation.

Example $GLOBAL block

```
$GLOBAL
    ref $EXPER = EXP1387;              *general experiment information
    ref $SCHEDULING_PARAMS = SKED1;    *choose scheduling program/parameters
    ref $PROCEDURES = STANDARD1        *general-procedure timing parameters
    ref $EOP = EOP129;                 *earth-orientation parameters
```

$MODE Block

The `$MODE` block must obey the following rules:

1. Consists exclusively of `def` blocks.
2. Within `def` blocks, only `ref` statements to primitive-block keywords are permitted.
3. For each observation in the `$SCHED` block, the set of parameters specified by the selected `$STATION` and `$MODE` keywords are concatenated with the parameters specified in the `$GLOBAL` block to form the full set of observing parameters for each station in that observation. Conflicting or duplicate parameters within the concatenated parameter set are not permitted.
4. Multiple `ref` statements to a given `def` block within the same primitive $block are simply concatenated.

5. Parameter values implied by the `ref` statements in the `$GLOBAL`, `$STATION` or `$MODE` blocks must not conflict. ~~(i.e., there must be no more than one value specified for a single parameter)~~

Notes:

1. The requirement that only `ref` statements are allowed in the `$MODE` block may seem somewhat awkward since some of the `ref` statements refer to single-parameters primitive `def` blocks. The advantage, however, is that this rule forces all parameter values to be set in the primitive $block to which they are uniquely assigned, eliminating any possible confusion about their meaning.
2. The `$MODE` block is intended to specify the detailed station setup parameters that may change from observation to observation.
3. A powerful capability of the `ref` statements in the `$MODE` block is the ability to qualify them by station, so that the specification of a single `$MODE` key automatically configures each station according to the details of its particular observing equipment and capabilities.
4. A `ref` *$blockName* = *keyword:stationKey(s)* statement is applied only to the station(s) specified.
5. Unless more than one recorder/remote data sink is involved, each antenna described in a `$MODE` block can reference only one of `$TRACKS`, `$BITSTREAMS`, or `$DATASTREAMS` blocks, but multiple blocks of that single type may be referenced.
6. Current operational consideration require that **all** `$MODE` `ref` statements be qualified by the complete list of stations to which they apply. This is necessary so that useful consistency checking can be applied. In the future, checking may be re-written to relax this constraint.

Example `$MODE` `def` block

```
  def SX;
  *As an illustration, this 'def' sets up data and recording parameters for one
station
    ref $FREQ = S4:HS;                      *station HS (Mark 5A)
    ref $BBC = HS/S4:HS;
    ref $IF = CDP/SX_WIDE:HS;
    ref $SEFD = HS:HS;
    ref $TRACKS = MARK5A/XX-4-2/8:HS;
    ref $TRACKS = MARK4_TRK_FORMAT:HS;
    ref $PROCEDURES = STANDARD_CAL:HS;
    ref $DAS = EARLY_START_20:HS;
  enddef
```

`$STATION` Block

The `$STATION` block must obey the following rules:

1. Consists exclusively of `def` blocks.
2. Within `def` blocks, only `ref` statements to primitive-block keywords are permitted.
3. For each observation in the `$SCHED` block, the set of parameters specified by the selected `$STATION` and `$MODE` keywords are concatenated with the parameters specified in the `$GLOBAL` block to form the full set of observing parameters for each station in that observation. Conflicting or duplicate parameters within the concatenated parameter set are not permitted.

4. Multiple `ref` statements to a given `def` block within the same primitive $block are simply concatenated.
5. Parameter values implied by the `ref` statements in the `$GLOBAL`, `$STATION` or `$MODE` blocks must not conflict (i.e., there must be no conflicting parameter/field combinations ~~more than one value specified for a single parameter~~).

Notes:

1. The requirement that only `ref` statements are allowed in the `$STATION` block may seem somewhat awkward since some of the `ref` statements refer to single-parameters primitive `def` blocks. The advantage, however, is that this rule forces all parameter values to be set in the primitive $block to which they are uniquely assigned, eliminating any possible confusion about their meaning.
2. A **station** consists of an **ANTENNA** on a **SITE** with a data-acquisition system (DAS). The `$STATION` block is intended for the specification of parameters which truly are associated with each station and will remain fixed for the duration of the experiment at each station.
3. The `$FREQ def` selected for each station, along with the corresponding selected `$BBC` and `$IF def` blocks, define the channels recorded from each antenna. Normally, all stations with same acquisition system type ~~will,~~ reference the same `def` blocks in the `$TRACKS, $BITSTREAMS, and $THREADS` ~~`$TRACK, $ROLL, $HEAD_POS, =$PASS_ORDER=`~~ blocks (depending on type of DAS), but care must be taken to ascertain that the combination of specified recordings is compatible with the DAS.

Special cases:

1. **Phased array** A phased array is represented as a single station, with a single site, antenna, and DAS.
2. **Multiple beams** Each beam is represented as a separate station, each of which may have the same site, antenna and DAS. Which stations use which source can be specified in the `source` parameter of the `$SCHED` block.
3. **Multiple antennas at one location (antenna cluster)** Each antenna is defined separately in the `$STATION` block and must have its own `$SITE def`. If the antennas are identical, they may reference the same `$ANTENNA def`.
4. **Multiple stations sharing a single recorder (antenna cluster or multiple beams)** If two or more stations share the same recorder, only the `def` blocks necessary for correct set-up at the stations and for correlation are required. This could include the `recording_system_ID` parameter (which identifies the particular DAS). If it is useful, one station may be declared as the recording controller with the inclusion of a `record_controller=1;` statement within the referenced `$DAS def` blocks for that station (this would indicate that recording is to be controlled according to the schedule of that station; the others will simply point). Note that some correlators will not allow data from multiple stations being sourced from a single piece of media and those that can don't necessarily have the ability to apply independent clock models. Proceed with caution! ~~*Multiple antennas sharing a single data-acquisition system (antenna cluster)* If two or more antennas share the same DAS, the `$STATION def` blocks for the corresponding antennas must have `ref` statements to exactly the same set of $DAS keywords, including particularly the `recording_system_ID` parameter (which identifies the particular DAS), except that one (and only one) of the stations must declare itself as the tape-control master with the inclusion of a `tape_control=master;` statement within the referenced `$DAS def` blocks for that station (this will cause the tape to be controlled according to the schedule of that station; the others will simply point).~~

Example $STATION def blocks

```
def EF;
  ref $SITE = EFLSBERG;
  ref $ANTENNA = EFLSBERG;
  ref $DAS = VLBA/1_DRIVE;
  ref $DAS = 33KBPI;
  ref $DAS = 8820FT;
  ref $DAS = EF_VLBA_ID;
  ref $PHASE_CAL_DETECT = STANDARD;
  ref $CLOCK=EF;
enddef;

def FD;
  ref $SITE = VLBA-FD;
  ref $ANTENNA = VLBA;
  ref $CLOCK=FF;
  ref $DAS = VLBA/2_DRIVES;
  ref $DAS = 33KBPI;
  ref $DAS = 17640FT;
  ref $DAS = FD_VLBA_ID;
  ref $PHASE_CAL_DETECT = STANDARD;
enddef;

def HS;
  ref $SITE = HAYSTACK;
  ref $ANTENNA = HAYSTACK;
  ref $CLOCK=HS;
  ref $DAS = MARK5A;
  ref $DAS = HS_MARK5A_ID;
  ref $PHASE_CAL_DETECT = STANDARD;
enddef;

def JB;
  ref $SITE = JODRELL_MK2;
  ref $ANTENNA = JODRELL_MK2;
  ref $CLOCK=JB;
  ref $DAS = MARK4;
  ref $DAS = 56KBPI;
  ref $DAS = 17640FT;
  ref $DAS = JB_MARK4_ID;
enddef;
```

$SCHED Block Definition (Augmented for VEX2)

The purpose of the `$SCHED` block is to specify the sequence of actual scans to observe. Each scan specified in the `$SCHED` block stands completely on its own as far as all details of station and recording configurations are concerned. This allows complicated switching between frequencies and/or recording modes to be executed with relative ease, and also allows a rearrangement of the schedule without the potential of lost configuration information.

The `$SCHED` block must adhere to the following rules.

1. Each scan is specified by a scan block consisting of a `scan` *scanID*; ... ; `endscan`; statement sequence. This (arbitrary) *scanID* will carry through into the log files and log-summary files to serve as a convenient reference to identify particular scans. Each *scanID* within the `$SCHED` block should be unique. [Note that the *scanID* is primarily for the convenience of the correlator and not of direct relevance to scheduling.]
2. Each scan block must include the appropriate `start`, `source`, `mode` and `station` parameter statements.
3. The scan blocks must be in time order by `start` times, although adjacent scans may have the same `start` time, if for example, each of two antennas is to be pointed at different sources starting at the same time. [Note that the time ordering constraint may be removed in the future.]
4. Each station participating in a scan is specified in a separate `station` statement.
5. The set of stations specified for a single scan must include all stations needed to form the desired set of baselines to be correlated (though not all possible baselines will necessarily be correlated). In cases of complicated clustering or subnetting, the correlator may, in some cases, be required to create composite scans spanning more than one `scan` block.

The parameters for the `$SCHED` block:

Parameter	Field	Description	Type	Allowed values	Units	Comments
`start`	1	Nominal scan start time	epoch		epoch	The time must be chosen to be the expected time of the first good data at any station in the 'scan'.
`mode`	1	Mode	keyword			Points to `def` in the `$MODE` block
`source`	1	Source	keyword			Points to `def` in the `$SOURCE` block. Also, see Note 6 in the `$SOURCE` block.
	(2)	Is pointing center?	int	`0` | `1`		If not set, assumed to be true. Only one source per station per scan block can be the pointing center.
	(3)	Correlate?	int	`0` | `1`		If not set, assumed to be true. Zero implies "don't correlate".
	(4,5,...)	Station list	keywords			Stations that this applies to. If none are present, it applies to all.
`station`	1	Station	keyword			Points to `def` in `$STATION` block
	2	Data good	real		time	Time after 'start' that data will be good
	3	Data stop	real		time	Time after 'start' that good data stops
	(4)	Media position	real		length **or** time **or** byte count	Estimated location of media at beginning of this scan. It can be NULL for dynamic media ~~tape~~ allocation. Using `0` indicates new media is requested, which may be ignored by some stations, particularly those without removable media. If not `0` for a disk recording, it is only informational since recordings can only be appended. It may be useful for estimating the media volume needed.
	(5)	~~Pass~~	~~char~~			Always null.
	(6)	Wrap_ID	link			Links to `pointing_sector` statements in the `$ANTENNA` block. Also, see Note 2 in the `$ANTENNA` block.
	(7)	~~Drive number~~	int	`0` | `1` | `-1`		`0` implies no recording. Defaults to `1`, recording, if not present. Some

		Record Flag				post-processing software packages use `-1` to indicate that the data was not correlatable for some reason.
`data_transfer`	1	Station	keyword			Points to `def` in `$STATION` block.
	2	Method	char	`disk2file` | `in2net`		
	3	Destination	char			File name, correlator, or blank
	4	Data start	real		time	Time of first data after 'start' to transfer. If null, data good time at station.
	5	Data stop	real		time	Time of last data after 'start' to transfer. If null, data stop time at station.
	(6)	options	char			Future use or blank or omitted
`intent`	(1)	Source	keyword			Points to `def` in the `$SOURCE` block. If not present, `intent` applies to all sources. See note 2.
	2	Identifier	char			Specify a purpose of the scan
	3	Value	char			The value of the intent.
`pointing_offset`	(1)	Station	keyword			Points to `def` in the `$STATION` block. If not present, `pointing_offset` applies to all stations. See note 3.
	2	coord1	char	`ra_ref` | `ra` | `ha` | `az` | `x`		First coordinate defining direction of offset; usually this will be `ra`, or `az` or the first axis of antenna. See note 4.
	3	offset1	real		angle	The angular offset along *coord1*.
	4	coord2	char	`dec_ref` | `dec` | `el` | `yns` | `yew`		Second coordinate defining direction of offset; usually this will be `dec` or `el` or the second axis of antenna.
	5	offset2	real		angle	The angular offset along *coord2*.

Notes:

1. `data_transfer` was an undocumented extension to VEX 1.5c to automate data transfers for real time fringe tests and other uses. The specified method and destination may be overridden (or the transfer eliminated) at the station to adapt to changing requirements.

2. Within a scan, any given `intent` Identifier may be set either for all sources (null keyword) or individual sources (possibly a subset), but not both. There may be no duplicate or conflicting `intent` statements. See text below for details on intents.
3. Offsets apply to the pointing source of the scan. Within a scan, `pointing_offset` statements may either be for all stations (null keyword) or individual stations (possibly a subset), but not both. Only one `pointing_offset` parameter may apply for a given station.
4. Axes of offsets must be commensurate,e.g., `ra_ref` with `dec_ref` **or** `ra` (or `ha`) with `dec` **or** `az` with `el`. Individual antennas may not support offsets of a particlar type. Pointing offsets for `ra_ref` with `dec_ref` apply to the coordinates of the `ref_coord_frame`, all others are "of date" or in the local frame of the station as appropriate.

Multiple phase centers

Multiple `source` parameters pointing to members of the `$SOURCE` block can be used to drive correlation at additional points within the antenna primary beam. Intents (below) are used to describe how these sources are to be used. The `source` parameter that defines the pointing center specifies the position used for pointing the antenna and phasing any phased array antennas. There are a multitude of interesting cases where the assumption that all of the action occurs at this single point in the sky for a given scan is violated. Examples include correlating (perhaps simultaneously) at multiple phase centers within the antennas' primary beams, forming more than one phased array beam toward more than one points within single element primary beams, and maintaining array phasing based on the phases derived from a source within the primary beam that is not at the phase center. By default, all sources will be correlated. Sources that need not be correlated (such as a pointing center where there may not be any compact radio emission) can have their third parameter set to 0.

Intents

The optional `intent` parameters provide a means of conveying the purpose (or "intent") of a particular scan to both downstream software and (human) users of the data. An example purpose would be designating a particular scan as useful for bandpass calibration. Each intent has three fields, the Identifier, or "key" which specifies the intent class, the Value, and an optional link to the SOURCE table. This key=value system offers great flexibility as there are no restrictions, other than those that would otherwise violate assumptions about character usage within vex, on the settings of these two fields. To be useful, it is suggested that the Identifiers, to the extent possible, are chosen from a standard list. The list below is standard in the sense that it is to be supported by the CASA data reduction software and is based on usage within the EVLA and ALMA projects.

Schedule writers should not assume, without verifying, that any given station or correlator supports any particular `intent`.

While originally conceived as a mechanism to convey hints to post-processing pipelines, intents can be useful at several stages during a projects life, including: observing, correlating, archiving, and data reduction. It would be prefered to keep usage of intents as uniform as possible. Intents that end up being useful for observing or correlating should be considered as recommendations for new formal vex parameters. It is highly discouraged too use the intent system to store information that is available through more rigorously defined parameters within the vex file.

Identifier	Origin	Expected values	Comments
CALIBRATE_AMPLI	EVLA	`True` | `False`	Is this source usable as an amplitude calibrator?
CALIBRATE_AUTOPHASE	EVLA	`True` | `False`	For interferometer stations, use this source to phase the array. Additional station-specific intents may be required.
CALIBRATE_BANDPASS	EVLA	`True` | `False`	Is this source usable as a bandpass calibrator?
CALIBRATE_DELAY	EVLA	`True` | `False`	Is this source expected to be useful (strong and pointlike enough) for delay (fringe) determination?
CALIBRATE_PHASE	EVLA	`True` | `False`	Is this source usable as a phase calibrator?
CALIBRATE_POINTING		`True` | `False`	This scan is used to hone the pointing of the antenna and should not be recorded
CALIBRATE_POLARIZATION_ANGLE	EVLA	`True` | `False`	Is this source suitable as a polarization angle calibrator?
CALIBRATE_POLARIZATION_LEAKAGE	EVLA	`True` | `False`	Is this source suitable as a D-term calibrator?
CALIBRATOR_CODE	NRAO	char	Any single uppercase letter can be specified. The blank character is the default for each scan if none is provided. This value will make it into a FITS-IDI file and will be visible within AIPS. See discussion and Calcode table below.
CLOCK_CHECK		`True` | `False`	An intent of this scan is for use as a real-time (or near-real-time) check of delays
OBSERVE_TARGET	EVLA	`True` | `False`	This identifies a source as being an astronomical target of the observation
QUALIFIER_CODE	NRAO	int	An integer to indicate the position in a sequence of scans that may have otherwise similar identifiers. This value will make it into a FITS-IDI file and will be visible within AIPS.
SPECTRAL_LINE_SOURCE		`True` | `False`	Is this source expected to be more strongly detected as a spectral line source?
CALIBRATE_ARRAY_PHASING		`True` | `False`	Should array phasing information be computed from measurements made in this scan?
APPLY_PREVIOUS_ARRAY_PHASING		`True` | `False`	Should array phasing information from the previously determined (last scan when CALIBRATE_ARRAY_PHASING was True) be applied?

APPLY_CURRENT_ARRAY_PHASING	`True` | `False`	Should array phasing information be determined from measurements made within this scan (determine the array phasing and apply it during the same scan)?
FRINGE_FINDER	`True` | `False`	Source is recommended and/or intended to facilitate finding fringes?

The CALIBRATOR_CODE has historically been used by the VLA and VLBA to encode something about the scan, usually the quality of a calibrator or the purpose of the scan, or the origin on the source (such as which survey defined it). Due to the fact that this field has been used for different purposes on different arrays and is rather limited in its information content, the concept of intents is becoming more widely accepted although is not yet supported by many observation scheduling and post-processing software packages and data formats. Some existing CALIBRATOR_CODE meanings

Calcode	Context	Meaning
A	VLA	Calibrator with position accurate to < 2 mas
B	VLA	Calibrator with position accurate to 2-10 mas
C	VLA	Calibrator with position accurate to 10-150 mas
G	VLBI	Pulsar gating is requested for this scan
L	VLBI	Perform line minus continuum for pointing
M	VLBI	A source from the Merlin source list
T	VLA	Calibrator with position accurate to worse than 150 mas
V	VLBI	A source from the VLBA calibrator survey
Y	VLBI	A calibrator taken from the VLA calibrator survey
Z	VLBI	Object is ephemeris driven

Multiple stations ~~antennas~~ feeding one recorder ~~DAS~~

Additional care must be taken in the special case of multiple stations sharing the same recorder (possibly in an antenna cluster or for multiple beams). In this case, it may be useful to leave the Media Position and Record Flag fields in the `station` statement null for the stations not controlling recording. If there is no conflict caused by mutliple station controlling recording, each of the stations can act independently and be scheduled as such within the `$SCHED` block. What can be scheduled is dependent on what the stations and the correlator can handle. Please also see the item for this topic in the Special cases section for `$STATION`. ~~Additional care must be taken in the special case of multiple antennas sharing the same DAS (possibly in an antenna cluster). One of the stations in the cluster must be declared as the master tape controller with the inclusion of a `tape_control=master;` statement included within referenced `$DAS def` statement for that station. This will cause the tape to be controlled according to the schedule of that station (i.e., the other stations in the cluster will only point). In this case, the *strt pos*, *pass* and *drive* sub-fields in the `station=` statement will be null. Apart from the tape-control aspect mentioned above, each of the antennas in a cluster act as independent stations and scheduled as such within the `$SCHED` block. Note that some correlators will not allow data from multiple stations being sourced from a single piece of media and those that can don't necessarily have the ability to apply independent clock models. Proceed with caution!~~

Example $SCHED scan blocks

```
  *                      stn   data      data      strt     point  Record
  *                      key   strt      stop       pos     sectr    Flag
  scan 263-061500;
     start=1995y263d06h15m00s;
     mode=SX;
     source=HD123456;
     station=             EF:    0 sec:  180 sec:    0 ft : :  &n  :      1 ;
     station=             FD:    0 sec:  180 sec:    0 ft : :  &cw :      1 ;
  endscan;
  scan 263-062000;
     start=1995y263d06h20m00s;
     mode=SX;
     source=HD123456;
     station=             HS:   20 sec:  120 sec:         : :  &n  :      1 ;
     station=             JB:    0 sec:  150 sec:    0 ft : :      :      1 ;
  endscan;
  scan 263-063000;
     start=1995y263d06h30m00s;
     mode=SX;
     source=3C123;
     station=             EF:   10 sec:  300 sec: 2000 ft : :  &n  :      1 ;
     station=             FD:    0 sec:  300 sec: 2000 ft : :  &np :      1 ;
*'plunge' mode
  endscan;
  scan 263-063500;
     start=1995y263d06h35m00s;
     mode=SX;
     source=3C123;
     station=             HS:   20 sec:  300 sec:         : : &ccw :      1 ;
     station=             JB:    0 sec:  300 sec: 1700 ft : :      :      1 ;
  endscan;
```

VEX Parameter Tables

Each VEX primitive $block , and the $SCHED $block, is allowed to specify only parameters that are defined for that $block. The VEX Parameter Tables define in detail these parameters for each $block (and for the VEX_rev parameter).

Each table lists the allowed parameter names and the associated fields. Field numbers in parenthesis are optional. Parameter type '&link' is a VEX linkword as defined in the VEX documentation. If 'units' are specified, the field must have a units label of the proper type; 'units' enclosed in parenthesis are assumed and should not be labeled.

Not all parameters need to be specified for a given experiment, only those that are 'relevant'. This listing does not attempt to specify the set of 'relevant' parameters, but is instead left to the assumed expertise of the scheduling-software writer.

This listing is not to be regarded as necessarily complete, either with regard to $blocks or the set of native parameters within the $blocks. Many $blocks, particularly the $ xxxx _OBS blocks, are not yet specified. These $blocks may be defined in the future. Additional parameters will undoubtedly be added as the need arises.

VEX_rev Parameter

Every VEX file starts with a VEX_rev parameter statement, which occurs outside of any $block. There may be no whitespace before the parameter.

Parameter	Field	Description	Type	Allowed values	Units	Comments
VEX_rev	1	Revision	char	2.0		This field may not be quoted text.

Example VEX_rev statement

```
VEX_rev = 2.0 ; *revision
```

$ANTENNA Block (Minor changes for VEX2)

The $ANTENNA block is a primitive block. It specifies the detailed characteristics of the antenna itself, including the type of mount and pointing characteristics. This block describes the physical characteristics of an antenna but not its location, so a single entry can be used for multiple identical antennas, as for the VLBA.

Parameter	Field	Description	Type	Allowed values	Units	Comments
`antenna_diam`	1	Antenna diameter	real	> 0	length	
`antenna_name`	1	Antenna name	char			Useful, if needed, to distinguish particular antenna from site (portable antenna, for example)
`axis_type`	1	Primary axis type	char	`az` | `ha` | `x` | `fixed`		Typically, `axis_type=az:el;` or `axis_type=ha:dec;` or `axis_type=x:yns;` (N/S orientation) or `axis_type=x:yew;` (E/W orientation), etc...
	(2)	Secondary axis type	char	`el` | `dec` | `yns` | `yew`		Pairing of parameters 1 and 2 must make sense (see comment immediately above)
	~~2~~ (3)	azimuth of +90 of the secondary axis ~~axis orientation~~	real ~~char~~		angle	Only used if significantly different than the nominal value. For an `x/y` antenna, the nominal values are: 0 deg - north/south orientation; 90 deg - east/west orientation
	(4)	elevation of +90 of the secondary axis	real		angle	Only used if significantly different than the nominal value. Could be used for a `ha:dec` antenna at the wrong latitude. Values > +90 may be useful.

axis_offset	2	real		length		
antenna_motion	1	Axis type	char	(axis type)		Must be one antenna_motion statement for each motion axis.
	2	Slew rate	real		angular rate	
	3	Constant ~~Settling time~~	real		time	See Note 4
	(4)	Acceleration	real		angular acceleration	
pointing_sector	1	Sector_ID or Wrap_ID	&link			May be used to specify pointing sector or wrap in $SCHED block.
	2	Primary axis type	char	az | ha | x | fixed		
	3	Lower limit	real		angle	Upper limit > Lower limit
	4	Upper limit	real		angle	
	(5)	Secondary axis type	char	el | dec | yns | yew		See Note 2
	(6)	Lower limit	real		angle	Upper limit > Lower limit. See Note 2
	(7)	Upper limit	real		angle	See Note 2
	8	Sector name	char	ccw | n | cw | ccwp | np | cwp		Any values supported by antennas are allowed. Conventionally all Az/El antennas can use ccw | n | cw. For 'plunged' regions (elevation &gt +90, use with p, Other values may be added. See Note 3
nasmyth	1	Band link	&link			
	2	Platform	char	left | right		Feed Location

Notes:

1. pointing_sector is typically only used for Az/El antennas.
2. All fields for the secondary axis of pointing_sector must have values or be null. Nulls can be used either for a single axis antenna or a dual axis antenna with no need to define the secondary axis sector limits.
3. VEX1 readers used the 'Sector_ID', referenced in the $SCHED block station parameter, as the 'Sector name'. This was necessary because VEX1 did not have a 'Sector name'. VEX2 readers should use the 'Sector name' as the value sent to the telescopes. To reduce the risk of losing data, VEX2 writers are encouraged to use the same string for the 'Sector_ID' and 'Sector name' to the extent possible.

4. The Constant term represents all fixed delays in the slewing model. Among other things, these may include the time required for a new source to be commanded and the settling time. In addition, if there is no Acceleration term in the model, the Constant may include a contribution to account for the acceleration and deceleration of the antenna.

Example $ANTENNA def blocks

```
  def EFLSBERG;
    antenna_diam = 100 m;
    axis_type = az : el;
    axis_offset = 0 m;
  *                 type   slew     settle
    antenna_motion = az: 90 deg/min: 2 sec;
    antenna_motion = el: 30 deg/min: 1 sec;
  *                   Sector  Axis   LoLimit  HiLimit  Axis  LoLimit HiLimit
  *                     ID
    pointing_sector = &ccw   : az :  -90 deg:   90 deg: el :  0 deg:  88 deg;
*ccw cable wrap zone
    pointing_sector = &n     : az :   90 deg:  270 deg: el :  0 deg:  88 deg;
*neutral cable wrap zone
    pointing_sector = &cw    : az :  270 deg:  450 deg: el :  0 deg:  88 deg;
*cw cable wrap zone
  enddef;

  def VLBA;
    antenna_diam = 25 m;
    axis_type = az : el;
    axis_offset = .123 m;
                    type   slew     settle
    antenna_motion = az: 90 deg/min: 2 sec;
    antenna_motion = el: 30 deg/min: 1 sec;
  *                   Sector  Axis   LoLimit  HiLimit  Axis  LoLimit HiLimit
  *                     ID
    pointing_sector = &ccw   : az : -90 deg:   90 deg: el :  0 deg:  88 deg;
*ccw cable wrap zone
    pointing_sector = &n     : az :  90 deg:  270 deg: el :  0 deg:  88 deg;
*neutral cable wrap zone
    pointing_sector = &cw    : az : 270 deg:  450 deg: el :  0 deg:  88 deg;     *cw
cable wrap zone
    pointing_sector = &ccwp  : az : -90 deg:   90 deg: el : 92 deg: 120 deg;
*ccw cable wrap zone, plunge   mode
    pointing_sector = &np    : az :  90 deg:  270 deg: el : 92 deg: 120 deg;
*neutral cable wrap zone,      plunge mode
    pointing_sector = &cwp   : az : 270 deg:  450 deg: el : 92 deg: 120 deg;     *cw
cable wrap zone, plunge    mode
  enddef;

  def HAYSTACK;
    ref <external_file>:$ANTENNA = HAYSTACK;
  enddef;
```

$BBC Block (Minor changes for VEX2)

The $BBC block connects physical BBC's to the 'logical' BBC's defined in the $FREQ section, and also specifies the connection of the BBC to a 'logical' IF.

Parameter	Field	Description	Type	Allowed values	Units	Comments
BBC_assign	1	Logical BBC 'link' with chan_def statement in $FREQ block	&link			
	2	Physical BBC# or DBE channel#	int	~~1~~ ~~16~~ >=0		See Note 3
	3	Logical IF 'link' with if_def statement in $IF block	&link			

Notes:

1. All logical BBC's defined in the selected $FREQ 'def' must be present in the selected $BBC 'def', but not necessary vice-versa.
2. The ~~$BBC block is~~ BBC# or DBE channel# field maybe ignored by the correlator. It is necessary to use the $BBC block to connect the channels in the $FREQ block to the input IFs in the $IF block.
3. For traditional analog BBC systems, field 2 is the physical position of the BBC#; for DBE systems it is the DBE channel#. Whether the DBE channel# is within an IF or global across all IFs is system dependent. The index origin for this field may depend on the specific device. Typically for BBC# the origin is 1; for DBE channel# it is 0.

Example $BBC def blocks

```
def EF/S4X4;
  ref External.File:$BBC = EF/SX;
enddef;

def VLBA/S4X4;
*                BBC   Physical     IF
*                 ID     BBC#       ID
  BBC_assign = &BBCa  :   1    : &IF_XR1;
  BBC_assign = &BBCb  :   2    : &IF_XL1;
  BBC_assign = &BBCc  :   3    : &IF_SR1;
  BBC_assign = &BBCd  :   4    : &IF_SL1;
enddef;

def VLBA/S4X4_DBE; *DBE system
*(emulating 4 dual-sideband BBCs, 2 each in 2 DBEs)
*               &BBC      DBE       &IF
*                lnk      Chn#       ID
  BBC_assign =  &BBC0a :   0    : &IF_XR1;
  BBC_assign =  &BBC0b :   1    : &IF_XR1;
  BBC_assign =  &BBC1a :   0    : &IF_SR1;
  BBC_assign =  &BBC1b :   1    : &IF_SR1;
enddef;
```

$BITSTREAMS Block (New for VEX2)

The $BITSTREAMS block is used to define a VSI-H data format (such as Mark 5B) within VEX. This is a new block defined in VEX2, adapted from JIVE's extension to VEX1.

Parameter	Field	Description	Type	Allowed values	Units	Comments
`stream_def`	1	Channel link	&link			
	2	Sign or magnitude bitstream	char	`sign` | `mag`		The samples are always real.
	(3)	Input bitstream number	int	0-31		Designates input pin on VSI-H connector, defaults to on-disk bit-stream number. See note 1
	4	On-disk bitstream number	int	0-31		Designates bitstream number in the recording
	(5)	Bitstream link	&link			See Note 2
`stream_sample_rate`	1	Sample rate	real		sample rate	This is the sample rate per bit-stream.
	(2)	Bitstream link	&link			See Note 2
`stream_label`	1	Stream label	char			See Note 3
	(2)	Bitstream link	&link			See Note 2

Notes:

1. The input bitstream number is necessary to calculate the bitstream mask that determines which bitstreams are to be recorded, as used with the Mark 5B mode command.
2. The bitstream link can be used with a `connection` statement in `$DAS` to specify the destination of a bitstream, if it is useful, e.g., to distinguish two bitstreams from a DBBC going to two different destinations.
3. The Stream label can be used to specify identifying information that may be needed to keep track of different streams recorded by one recorder, possibly either a name or the recording group size for a Mark 6.
4. For Mark 5B data over Ethernet: The native 10016-byte Mark 5B data frame is divided into two consecutive 5008-byte Ethernet packets; on playback, each packet pair is concatenated to form a normal 10016-byte Mark 5B data frame.

Example $BITSTREAMS def block

```
def MK5B.8Ch2bit;
     stream_def = &CH01 : sign : 16 :  8;
     stream_def = &CH01 :  mag : 17 :  9;
     stream_def = &CH02 : sign : 18 : 10;
     stream_def = &CH02 :  mag : 19 : 11;
     stream_def = &CH03 : sign :  0 :  0;
     stream_def = &CH03 :  mag :  1 :  1;
     stream_def = &CH04 : sign :  2 :  2;
     stream_def = &CH04 :  mag :  3 :  3;
     stream_def = &CH05 : sign : 20 : 12;
     stream_def = &CH05 :  mag : 21 : 13;
     stream_def = &CH06 : sign : 22 : 14;
     stream_def = &CH06 :  mag : 23 : 15;
     stream_def = &CH07 : sign :  4 :  4;
     stream_def = &CH07 :  mag :  5 :  5;
     stream_def = &CH08 : sign :  6 :  6;
     stream_def = &CH08 :  mag :  7 :  7;
     stream_sample_rate = 4 Ms/s;
enddef;
```

$CLOCK Block (Augmented for VEX2)

The $CLOCK section specifies the necessary clock parameters for proper correlation of the data. Normally, this information will be taken from the station logs, transcribed manually or, as a last resort, determined from a fringe search at the correlator. Obviously, the $CLOCK section is needed only by the correlator.

Parameter	Field	Description	Type	Allowed values	Units	Comments
`clock_early`	(1)	Starting epoch for validity of this clock model	epoch		epoch	Model in this statement assumed valid until next specified epoch, if any. May be null if ~~of~~ model is valid for entire experiment.
	2	'clock_early' - offset of station clock ~~formatter 1-pps tick~~ with respect to UTC	real		time	>0 for station clock ~~formatter tick~~ early. See note 2
	(3)	Epoch of origin of clock model	epoch		epoch	Needed if clock rate, ~~or acceleration~~ higher terms, or 'fmout2gps' are specified.
	(4)	Clock rate	real		time rate	For example, `usec/sec`. See note 3
	(5)	Second order polynomial coefficient	real		time acceleration	coefficient of t^2 in in polynomial expansion. See note 3
	(6)	Third order polynomial coefficient	real		time jerk	coefficient of t^3 in polynomial expansion. See note 3
	(7)	fmout2gps	real		time	Station measured value of how much the formatter 1-pps leads the GPS 1-pps at the epoch of the model. See note 4

Notes:

1. In case of 'clock breaks', where multiple sets of clock parameters must be specified for a station during a single experiment, multiple `clock_early` statements may be used, each specifying the starting epoch of validity for the model in the corresponding statement.
2. The 'clock_early' offset is the amount to subtract from the recorded time-tags to correct them for signal path delays at the station.
3. All clock terms up to the highest order used must be specified.
4. The 'clock_early' offset differs from 'fmout2gps' by a station dependent 'peculiar offset': 'clock_early' = 'fmout2gps' + 'peculiar offset'. The 'peculiar offset' will typically vary by a small amount for each experiment, but should remain roughly constant unless there are changes that affect: (i) the signal path delay from the intersection of axes to the point where the data is time-tagged, or (ii) the wiring of the 'fmout2gps' measurements.
5. The sign conventions for all clock components and 'fmout2gps' are the same; a positive number means the station clock is running fast.

Example $CLOCK def blocks

```
  def EF;
*                   Valid from     clock_early  clock_early_epoch    rate
    clock_early = 1995y263d06h00m :  2.5 usec :  1995y132d00h08m0s : 1.2e-12;
    clock_early = 1995y263d12h00m :  1.5 usec;                                *clock
'break'
  enddef;

  def FD;
    ref External.File:$CLOCK = VLBA-FD;
  enddef;

  def HS;
    clock_early= : -3.5 usec;                    *Valid though entire experiment
  enddef;
```

$DAS Block (Substantially changed for VEX2)

The $DAS block is intended to define the recording hardware present at a station. Because of the many combinations possible, various elements of the hardware are separately specified.

Parameter	Field	Description	Type	Allowed values	Units	Comments
equip	1	Equipment type	char	rack | recorder | remote | software		Defines equipment type (does not cover sub-equipment). The remote type is for remote data sinks used in place of or in additon to recorders. Additional types may be added.
	2	Device	char	For rack: Mark4 | Mark5 | VLBA | VLBA4 | VLBAG4 | VLBA5 | VLBAG5 | K4 | LBA | RDBE_PFB | RDBE_DDC | DBBC_PFB | DBBC_DDC | ALMA | DVP | WIDAR | none For recorder: K5 | Mark5A | Mark5B | Mark5C | LBADR | PC-EVN | Mark6 | FlexBuff | XCube | none For remote: JIVE | VLBA | Haystack | Bonn | WACO | Curtin | none		Defines device (or remote destination). The none devices are for cases where the site equipment is unknown or is known to not be supported by automated processing at the station. Additional devices may be added
	3	Equipment link	&link			Defines link for equipment (not for sub-equipment)
	(4)	Label	char			Optional local station defined unit label, if useful. Arbitrary strings are acceptable, but typically should be a sequentially

Parameter	Field	Description	Type	Allowed values	Units	Comments
						increasing value: 1, 2, 3, ... or A, B, C ...
composite_equip	1	Equipment	&link			Equipment link for overall unit. Each piece of equipment may be defined in terms of sub-units
	2,3(,...n)	Sub-equipment link	&link			Equipment link for sub-equipment, at least two required.
equip_set	1	Equipment	&link			Equipment (or sub-equipment) link for device. This parameter is used to specify equipment settings when needed.
	2	Function	char	AGC		Feature to be set. Additional functions may be added
	3(,...n)	Settings		For AGC: on | off		Setting fields depend on the Function and may include any type of field data including char, values (with or without units, non-standard units allowed), and links. At least one is required, but there can be an arbitrary number. Additional settings may be added
equip_info	1	Equipment	&link			Equipment (or sub-equipment) link for device. This parameter is used to provide information to identify the device when needed, not settable information.
	2	Name	char	model | serial | address		Name of indentifying information. Additional names may be added

Parameter	Field	Description	Type	Allowed values	Units	Comments
	3(,...n)	Values		For `model`, `serial`, and `address`: device dependent string.		Value fields depend on the `Name` and may include any type of field data including char, values (with or without units, non-standard units allowed), and links. At least one is required, but there can be an arbitrary number. For `address`, arbitrary strings are acceptable, but they should be appropriate for the device, typically sequentially increasing value: `1`, `2`, `3`, ... or `A`, `B`, `C` .... Additional values may be added
`connection`	1	Signal	&link			Signal to be connected. This parameter is used to specify signal connections, when needed.
	2	Equipment	&link			Equipment (or sub-equipment) to be connected to.
	3	Label	char	For `Mark4` and `Mark5` rack inputs: `1N`, `1A`, `2N`, `2A`, `3` ~~`3N`, `3A`~~ For `VLBA`, `VLBA4`, `VLBAG4`, `VLBA5`, `VLBAG5` rack inputs: `AN`, `AE`, `BN`, `BE`, `CN`, `CE`, `DN`, `DE` For `DBBC_DDC` and `DBBC_PFB` rack inputs: `A1`, `...`, `A4`, `B1`, `...`, `D4`		Equipment (or sub-equipment) defined label of connector to connect to.

Parameter	Field	Description	Type	Allowed values	Units	Comments
				For LBA rack inputs: 1N, 1A, 2N, 2A		
	(4)	Direction	char	in | out | bi		Optional direction of signal relative to equipment, if needed.
	(5)	Type	char	analog | digital | ETH		Optional signal type, if needed. Additional types may be added
recording_system_ID	1	Recording-system serial number	int			Must be unique for each system of a given type within an experiment. This information ~~is often recorded on tape~~ may be included with the data in some cases and used at the correlator. See Notes 2 and 3 below.
record_method	1	Specifies recording pattern	char	start+stop | continuous | adaptive		
	(2)	Early-start time	int	>=0	time	Relevant for start+stop and adaptive motion and first scan for continuous; specifies how soon recording should start before expected good data.
	(3)	Minimum gap for stopping	int		time	Specifies minimum valid-data time gap during which recording will be stopped. See Note 4 below.
record_control	1	Define record control	int	0 | 1		Relevant only when more than one station is recorded on a single recording system. Defines which antenna controls recording

Parameter	Field	Description	Type	Allowed values	Units	Comments
						(with value 1). Others simply point.
record_transport_type	1	Transport type	char	Mark3A|Mark4|VLBA|VLBAG|S2|K4|%Mark5A		Additional types may be added
	(2)	Revision level	char			Example: '1.0'
electronics_rack_type	1	Type of electronics rack	char	Mark3A|Mark4|VLBA|VLBAG|S2|K4		
number_drives	1	Number of drives (transports)	int			Number of physical drives connected to system
headstack	1	Headstack#	int	1-4		Specifies correspondence between physical headstack and logical drive. Relevant for MkIIIA, MkIV, VLBA, Mark 5A.
	2	Headstack function	char	read|write|read/write		
	3	Drive-number offset with which this headstack is associated	int			Added to drive# specified in the station statement in $SCHED block to determine physical drive#. Normally 0 for single-drive systems.
record_density	1	Longitudinal bit density along track	int		(bpi)	Usually 33,333 or 56,000 (bpi unit is assumed). Used to compute tape speed. Relevant only for MkIII, MkIV, VLBA systems.
tape_length	1	Tape length	int		length|time	For Mark IIIA, Mark IV, VLBA: actual tape length. For S2, K4: recording time at data

Parameter	Field	Description	Type	Allowed values	Units	Comments
						rate specified in field 2.
	(2)	S2 tape speed	char	slp | ep	(Mbps)	Relevant only for S2
	(3)	Number of S2 cassettes	int	1-8		Must be consistent with total date rate (each S2 cassette records 16 Mbps)
record_transport_name	1	Recording-transport name	char			Identifies a particular recording transport. Mostly useful for transportable recording systems.
electronics_rack_ID	1	Electronics rack serial number	int			Identifies particular electronics rack
electronics_rack_name	1	Electronics rack name	char			Like a serial number, but a name
tape_motion	1	Specifies how tape is to be controlled	char	start&stop | continuous | adaptive		
	2	Tape early-start time	int	>=0	time	Relevant for start&stop and adaptive motion; specifies how soon tape should start before expected good data.
	(3)	Late-stop time	int		time	Relevant for S2 only. Specifies length of time tape should continue running after end of valid data. Typically 0.5 minutes.
	(4)	Minimum gap for stopping tape	int		time	Relevant for S2 only. Specifies minimum valid-data time gap during which tape will be stopped. Typically 3 minutes.

Parameter	Field	Description	Type	Allowed values	Units	Comments
tape_control	1	Define master station in cluster	char	master		Relevant only in antenna cluster where data from more than one station is recorded on a single recording system. Defines which antenna controls tape motion. Others are assumed to be slaves.

Notes:

1. The DAS parameters are intended to be sufficient to describe any of the many variants of Mark3A, Mark4, Mark5 and VLBA systems currently deployed. Additional parameters values may be added in the future as systems evolve.
2. The recording_system_ID statement for some systems identifies the particular DAS used, akin to a serial number, which is usually written in the aux-data field of the recorded data. The *recording_system_ID* of each DAS within a given type (e.g., Mark4, VLBA, K4, etc) must be different for each DAS within an experiment; the *recording_system_ID* of a particular DAS is usually assigned at the time of manufacture. The recording_system_ID can be used by the correlator to positively identify the particular DAS upon which a recording was made. tape was written. The parameter electronics_rack_ID is similar.
3. If two or more stations antennas share the same recorder DAS, the $STATION 'defs' for the corresponding stations antennas must have 'refs' to compatible exactly the same set of $DAS keywords, including if appropriate, particularly the recording_system_ID parameter. One except that one (and only one) of the stations, at a time, may be declared must declare itself as the recording controller tape-control master with the inclusion of a record_controller=1; tape_control=master; statement within the referenced $DAS 'defs' for that station.
4. The 'Minimum gap time for stopping' refers to the difference between the last 'data end' time of the previous scan and the "start' time of the new scan. A consequence of this is that either all stations or no stations that are in both scans record continuously. Typically, the Field System omits the 'set-up', PREOB, POSTOB, and "check' procedures between continuously recorded scans. This approach is typically used for phase-referenced observations.

1. For Mark IIIA, Mark IV, VLBA and Mark 5A systems, the headstack statement indicates which 'headstack outputs' from the formatter are connected to the recording system(s). There must be one headstack statement for each formatter 'headstack output' to be recorded. Multiple recording systems may be connected to a single formatter.
2. If multiple headstacks on same drive, each headstack must have a different headstack# (e.g. Mark IV). Systems with multiple simultaneously-recording headstacks on 2 drives (e.g. VLBA with 2 drives) must have two 'headstack=' statements with a different headstack# and logical drive linkword. The headstack #'s must correspond to the headstack #'s in the $TRACKS block.
3. A 'drive-number offset' is associated with each headstack in the recording system. This number is added to the tape drive number specified in the station statement in the $SCHED block in order to determine the physical drive number. For single-drive systems, the drive-number offset will normally be 0, so that the drive number assigned in the $SCHED section is the actual physical drive number. In the case of multiple simultaneous drives, the drive-number offset must be different for each headstack so that the drive specification in the $SCHED block properly maps into the physical drive numbers being used.

Example $DAS `def` blocks

```
def DBE-A; *Define dual-DBE unit DBE-A with VDIF output
*
*         type   device     &equip   unit
  equip = rack : RDBE_PFB : &DBE-A :   A  ;
*
*                   &equip   &sub-equip1   &sub-equip2
  composite_equip = &DBE-A : &DBE-A1     : &DBE-A2     ; *Define sub-equipment
*
*
*              &equip/    name         value1
*            &sub-equip
  equip_info = &DBE-A  : model   : RDEH1.4 ;
  equip_info = &DBE-A1 : model   : PFB1.4  ;
  equip_info = &DBE-A1 : address : 1       ;
  equip_info = &DBE-A2 : model   : PFB1.4  ;
  equip_info = &DBE-A2 : address : 2       ;
*
*             &equip/    Function  Setting1
*           &sub-equip
  equip_set = &DBE-A :    AGC   : on       ; *Turn AGC on
*
*               &signal  &equip/  physical  direc-   type
*                      &sub-equip  label    tion
  connection =  &IF_XR : &DBE-A  :  in1 : in  : analog ;
  connection =  &IF_XL : &DBE-A  :  in2 : in  : analog ;
  connection =  &DS1   : &DBE-A  :  out : out : ETH    ;
enddef;

def MARK5C; *Define Mark 5C unit
*          type      device   &equip unit
  equip = recorder : Mark5C : &MK5C1 : 1 ;
*
*              &equip      name     value
               &sub-equip
  equip_info = &MK5C1    : model  : MARK5C-1.6 ;
  equip_info = &MK5C1    : serial : 655 ;
*
*              &signal &equip/    physical direc- type
*                      &sub-equip label    tion
  connection = &DS1   : &MK5C1   : in     : in   : ETH ;
enddef;
```

$DATASTREAMS Block (New for VEX2)

The `$DATASTREAMS` block describes VLBI Data Interchange Format (VDIF) Ethernet-based VLBI datastreams and is new in VEX2. The definiton for VDIF can be found at https://vlbi.org/vlbi-standards/vdif/. The channel links connect back to corresponding `chan_def` statements in the `$FREQ` block

Parameter	Field	Description	Type	Allowed values	Units	Comments
datastream	1	Datastream link	&link			
	2	Datastream format	char	VDIF | VDIF_legacy		VDIF_legacy is the 16-byte dataframe header
	(3)	Datastream label	char			See note 5
thread	1	Datastream link	&link			
	2	Thread link	&link			
	3	Thread number	int	>= 0		See note 6
	4	Number of channels in thread	int	> 0		
	5	Sample rate	real		Sample rate	Sample rate of this thread
	6	Bits per sample	int	>= 1		Bits per sample of this thread
	7	Sample type	char	real | complex		Sample type for this thread
	8	Data bytes/packet	int	>0		
channel	1	Datastream link	&link			
	2	Thread link	&link			
	3	Channel link	&link			
	3	Channel number	int	>=0		Channel number in thread
merged_datastream	(1)	Merged datastream link	&link			Used to indentify merged datastreams. See note 7
	(2)	Merged datastream label	char			See note 5
	3,4(,...n)	Constituent datastream links	&link			Links for constituent datastreams, must be at least two. See Note 8

Notes:

1. The $DATASTREAM def blocks are to be referenced from within a $MODE block in lieu of $TRACKS def blocks.
2. VDIF requires the number of channels within a thread to be a power of 2. Future formats may not have this restriction.

3. Multiple `channel` parameters can take data from the same channel link. A possible use for this capability is to define two threads, one with 1 bit per sample and one with 2 bits per sample and to, on a frame-by-frame basis, choose which frames to emit based on the capacity of an eVLBI data link.
4. See VDIF specification: http://www.vlbi.org/vsi/docs/VDIF%20specification%20Release%201.0%20ratified.pdf
5. Datastream labels can be used to specify meta-information that may be needed to keep track of different streams recorded by one recorder.
6. All thread numbers for a given station in a given mode must be unique.
7. A merged datastream link can be used with a `connection` statement in `$DAS` to identify the destination of a datastream, if it is useful.
8. When datastreams are merged from more than one station, the `merged_datastream` parameter appears for the recording control station. In this case, all datastream labels from the individual stations being merged must be unique.

Example $DATASTREAM def blocks

```
def DS12;
* Merged Merged Constituent Constituent
*                     datastream Datastream datastream1 datastream2
*                         link     label        link        link
  merged_datastream =           :  SX       :   &DS1   :    &DS2   ;
*
enddef;
*
*Define VDIF datastream
def DS1; *X-band VDIF datastream(1024Mbps total,2 threads, 4 chns/thread)
*
*           datastream   format  datastream
*              lnk                 label
  datastream= &DS1     :  VDIF  :   X     ;
*
*Define threads within datastream
*          datastream thrd   thrd  #    sample    b/s  type data bytes/
*            lnk      link     #  chns   rate                 packet
  thread=    &DS1 : &thread0 : 0 : 4 : 64 Ms/sec : 2 : real : 8000 ;
  thread=    &DS1 : &thread1 : 1 : 4 : 64 Ms/sec : 2 : real : 8000 ;
*
*Define channels within each thread
*        &datastream   &thread    &chn  chn# in
*           lnk         lnk       link   thrd
  channel=   &DS1   : &thread0 : &CH-XR0 : 0 ;
  channel=   &DS1   : &thread0 : &CH-XR1 : 1 ;
  channel=   &DS1   : &thread0 : &CH-XR2 : 2 ;
  channel=   &DS1   : &thread0 : &CH-XR3 : 3 ;
  channel=   &DS1   : &thread1 : &CH-XL0 : 0 ;
  channel=   &DS1   : &thread1 : &CH-XL1 : 1 ;
  channel=   &DS1   : &thread1 : &CH-XL2 : 2 ;
  channel=   &DS1   : &thread1 : &CH-XL3 : 3 ;
enddef ;
*
def DS2 *S-band VDIF datastream(1024Mbps total,2 threads, 4 chns/thread)
*
*           datastream format datastream
*              lnk              label
  datastream= &DS2    : VDIF :    S ;
*
*Define threads within datastream
*       datastream    thrd   thrd  #    sample    b/s  type data bytes/
*         lnk         link     #  chns   rate                 packet
 thread=   &DS2   : &thread0 : 0 : 4 : 64 Ms/sec : 2 : real : 8000 ;
```

```
 thread=   &DS2   : &thread1 : 1 : 4 : 64 Ms/sec : 2 : real : 8000 ;
*
*Define channels within each thread
*        &datastream &thread     &chn  chn# in
*           lnk         lnk      link   thrd
 channel=   &DS2   : &thread0 : &CH-SR0 : 0 ;
 channel=   &DS2   : &thread0 : &CH-SR1 : 1 ;
 channel=   &DS2   : &thread0 : &CH-SR2 : 2 ;
 channel=   &DS2   : &thread0 : &CH-SR3 : 3 ;
 channel=   &DS2   : &thread1 : &CH-SL0 : 0 ;
 channel=   &DS2   : &thread1 : &CH-SL1 : 1 ;
 channel=   &DS2   : &thread1 : &CH-SL2 : 2 ;
 channel=   &DS2   : &thread1 : &CH-SL3 : 3 ;
enddef ;
```

$EOP Block (Augmented for VEX2)

The `$EOP` block specifies the earth-orientation parameter to be used by the correlator or possibly needed for data-taking in the case of large antennas operating at high frequency (and hence narrow beam on the sky).

Parameter	Field	Description	Type	Allowed values	Units	Comments
TAI-UTC	1	Ephemeris TAI-UTC	real		time	Normally fixed for entire experiment
A1-TAI	1	Ephemeris A1-TAI	real		time	Normally fixed for entire experiment
eop_ref_epoch	1	Epoch of first 'EOP point'	epoch		epoch	
num_eop_points	1	Number EOP points	int			Number of points over which interpolation is done
eop_interval	1	Time space of EOP points	real		time	Typically 24 hrs
ut1-utc	1 - *num_eop_points*	Time series of ut1-utc values	real		time	Must be *num_eop_points* values in this statement. Units specification may be omitted after first field.
x_wobble	1 - *num_eop_points*	Time series of x-pole values	real		angle	Must be *num_eop_points* values in this statement. Units specification may be omitted after first field.
y_wobble	1 - *num_eop_points*	Time series of y-pole values	real		angle	Must be *num_eop_points* values in this statement. Units specification may be omitted after first field.
eop_origin	1	Type of EOP information	char	PREDICT | RAPID | FINAL		Used to indicate the expected quality of the EOP values
	(2)	Date	epoch		epoch	Version date associated with these values
	(3)	Source/version of EOP information	char			Arbitrary string identifying source and version, possibly analysis center and/or solution
nut_ref_epoch	1	Epoch of first 'nutation offset point'	epoch		epoch	See Note 2
num_nut_points	1	Number nutation offset points	int			Number of points in table. See Note 2
nut_interval	1	Time spacing of nutation offset	real		time	Typically one to a few days. See Note 2

Parameter	Field	Description	Type	Allowed values	Units	Comments
		points				
`delta_x_nut`	1 - *num_nut_points*	Time series of delta X values	real		angle	Must be *num_nut_points* values in this statement. Units specification may be omitted after first field.
`delta_y_nut`	1 - *num_nut_points*	Time series of delta Y values	real		angle	Must be *num_nut_points* values in this statement. Units specification may be omitted after first field.
`nut_model`	1	Reference model that deltas are added to	char			Some possible values: IAU1980, IAU2000A. See Note 2
`nut_origin`	1	Source/version of nutation offset information	char			Arbitrary string identifying source and version, possibly analysis center and/or solution
~~`delta_eps`~~	~~1 - *num_nut_points*~~	~~Time series of delta epsilon values~~	~~real~~		~~angle~~	~~Must be *num_nut_points* values in this statement. Units specification may be omitted after first field. See note 2~~
~~`delta_psi`~~	~~1 - *num_nut_points*~~	~~Time series of delta psi values~~	~~real~~		~~angle~~	~~Must be *num_nut_points* values in this statement. Units specification may be omitted after first field. See note 2~~

Notes:

1. EOP values can either be combined together all into one `def` block, or be spread over multiple (as per the example below). The later is more generally useful as it allows EOP values to have different leap second values (if the range of values spans a leap second) and allows accountability of the origin of each value.
2. Nutation parameters: ~~`delta_eps`~~, ~~`delta_psi`~~, `nut_ref_epoch`, `num_nut_points`, `nut_interval`, and `nut_model` were undocumented extensions to VEX 1.5c.

Example $EOP def blocks

```
def EOP55259;
  TAI-UTC= 34 sec;
  A1-TAI= 0 sec;
  eop_ref_epoch=2010y063d;
  num_eop_points=1;
  eop_interval=24 hr;
  ut1-utc  = 0.047389 sec;
  x_wobble = -0.03825 asec;
  y_wobble = 0.26195 asec;
  origin = FINAL : 2010y082d04h30m04s;
enddef;
def EOP55260;
  TAI-UTC= 34 sec;
  A1-TAI= 0 sec;
  eop_ref_epoch=2010y064d;
  num_eop_points=1;
  eop_interval=24 hr;
  ut1-utc  = 0.046017 sec;
  x_wobble = -0.03906 asec;
  y_wobble = 0.26394 asec;
  origin = RAPID : 2010y082d04h30m04s;
enddef;
```

$EXPER Block (Augmented for VEX2)

The $EXPER block contains general information useful for the success of the VLBI administrative process.

Parameter	Field	Description	Type	Allowed values	Units	Comments
exper_name	1	Experiment name	char			Typically will be the standard 6-char experiment designator INCLUDING the segment code (example. BW096A)
	(2)	Segment code	char			A specifier of the "segment" or "epoch" of a particular experiment (example: A)
exper_description	1	Experiment description	char			
exper_nominal_start	1	Epoch of nominal experiment start	epoch			
exper_nominal_stop	1	Epoch of nominal experiment end	epoch			
PI_name	1	PI name	char			
PI_email	1	PI e-mail address	char			
contact_name	1	Contact name	char			
contact_email	1	Contact e-mail address	char			
scheduler_name	1	Scheduler name	char			
scheduler_email	1	Scheduler e-mail address	char			
target_correlator	1	Target correlator	char	VLBA | VSOP | JIVE | Haystack | ~~etc.~~ Bonn | WACO | GSI		Others may be added.
scheduling_software	1	Program used to write schedule	char	SCHED | SKED | VIE_SCHED		Others may be added.
	(2)	Version	char	Program dependent string		Example for SKED: 2013May09

Parameter	Field	Description	Type	Allowed values	Units	Comments
	(3)	Epoch schedule was created	epoch			
VEX_file_writer	1	Program used to write VEX file	char	SCHED | SKED | VIE_SCHED		Others may be added. See note 2.
	(2)	Version	char	Program dependent string		Example for SKED: 2013May09
	(3)	Epoch VEX file was created	epoch			

Notes:

1. If a segment code is provided, the nominal name of the running project would be the concatenation of `exper_name` and the provided segment code.
2. The VEX file itself can be created by a different program than the program that creates the schedule. This allows for the existence of an external scheduling file format that is not VEX. If the `VEX_file_writer` parameter is omitted, it is understood to have the same values as the `scheduling_software` parameter.

Example $EXPER `def` block

```
$EXPER;                       *general experiment information
  def EXP1387;
    exper_name=RDWPS1;
    exper_description="This description can be up to 128 characters long."
    exper_nominal_start=1995y132d05h00m;
    exper_nominal_stop=1995y132d12h00m;
    PI_name=Joe_Schmoe;
    PI_email=jschoe@mycomputer.pu.edu;
    contact_name=John_Doe;
    contact_email=jdoe@myplace.edu;
    scheduler_name=Jane_Smith;
    scheduler_email=jsmith@abc.school.edu;
    target_correlator=EVN;
    scheduling_software= VIE_SCHED : 2013Sep21 : 2013y301d16h26m58s;
    VEX_file_writer= SKED : 2013May09 : 2013y302d12h34m32s;
  enddef;
```

$EXTENSIONS Block (New for VEX2)

The `$EXTENSIONS` block is intended to provide a way to specify site/system specific information and/or settings that are not covered by regular VEX2 prescriptions (e.g. VLBA stations having a control that must be set for some specific mode). Each `extension` statement must have an 'owner' for whom this extension is specific as well as a unique 'extension name' that identifies that particular extension within the owner's library of extensions. The 'extension name'

and list of specified parameters are meaningful **only to the 'owner'** and can be ignored by all others. The use of extensions is discouraged. Whenever possible other mechanisms should be used. There are useful options for some issues in the `$DAS` block. Extension 'defs', like any other primitive block defs, may be referenced in `$STATION`, `$MODE` or `$GLOBAL` blocks.

Parameter	Field	Description	Type	Allowed values	Units	Comments
`extension`	1	Owner	char			Specifies the name of the institution that this statement is used by.
	2	Name	char			Name of extension used by owner
	(3...n)	Values				Value fields depend on the `Name` and may include any type of field data including char, values (with or without units, non-standard units allowed), and links. At least one is required, but there can be an arbitrary number.

Example `$EXTENSIONS` `def` block

```
def NRAO_synthesizer ;
*            owner       name            param1      param2   param3
  extension = NRAO : X300_synthesizer : 12.35 GHz : -12 dBM :       ;
enddef;
```

$FREQ Block (Minor changes for VEX2)

The `$FREQ` block describes the ~~signal and sampling~~ RF sky Frequency, net sideband, and bandwidth characteristics of the channels recorded ~~on the tape~~, where a 'channel' is defined as a single USB or LSB output from a BBC. ~~This includes such information as total RF sky frequency, sideband, channel bandwidth, sampling rate and bits/sample.~~ The `$FREQ` block does not attempt to describe the recording mode (including format, sample rate, bits per sample, and sample type), since the same set of channels may be recorded in different recording modes (or on different equipment) at different stations. Each frequency channel is defined by a `chan_def` statement with at least ~~8~~ seven fields in each statement. The capability of specifying frequency switching is also built into the `chan_def` statement by assigning each frequency channel to one or more numbered 'states'. The `switching_cycle` statement then specifies the length of time spent in each of the states.

Parameter	Field	Description	Type	Allowed values	Units	Comments
chan_def	(1)	'Band_ID': RF band name	&link			Link to selected $SOURCE 'def' where it may be used to describe source-structure characteristics for scheduling purposes. May be omitted if not relevant.
	2	RF sky frequency at 0Hz in the BBC output	real		freq	This frequency, in combination with the LO frequency specified in the $IF block allows computation of the BBC LO frequency for this channel.
	3	Net sideband of this BBC channel	char	U | L | UC | LC		Note that this will be opposite the labeling on the BBC itself if the IF going into the BBC is net LSB. UC and LC indicate complex sampled data; see note 4
	4	BBC Channel bandwidth	real		freq	
	~~5~~ (5)	'Chan_ID': Logical channel name	&link			Must be different for each channel. Used as link to selected $TRACKS, $BISTREAMS, or $DATASTREAMS 'def'. See note 5
	6	'BBC_ID': Logical BBC name	&link			Link to selected $BBC 'def where connection to physical BBC is made.
	(7)	'Phase-cal_ID': Logical phase-cal name	&link			Link to selected $PHASE_CAL_DETECT 'def' block to specify details of phase-cal tone(s). Null specifies no phase-cal to be detected.
	(8)	Channel name	char			For correlation use. See note 6.
	~~(8,9,...)~~ (9,10,...)	Frequency-switched state numbers in which this	int			Used only with frequency switching. Specified state numbers are separated by colons. States must be ordered 1,2,...

Parameter	Field	Description	Type	Allowed values	Units	Comments
		channel is active				
switching_cycle	1	Phasing of frequency-switch cycle	char	wrt_obs_start | wrt_min_mark		Relevant only if frequency-switching being used. Timing of switching cycle begins according to this specification.
	2	State 1 period	real		time	Interval over which state 1 is active
	3...	State 2 period, etc	real		time	Interval over which state 2 is active, etc.
~~sample_rate~~	~~1~~	~~Sample frequency~~	~~real~~		~~sample rate~~	~~Ignored for S2.~~

Notes:

1. There must be one chan_def statement for each BBC channel to be recorded.
2. If one or more stations in an experiment observe different sets of frequency channels, there must be a separate 'def' for each different set of channels.
3. A resolved link must exist for every specified 'linkword' in each chan_def statement.
4. In the case of complex data, the RF frequency refers to the center of the band with the full sampled band ranging from half of the bandwidth below this to half of the bandwidth above this. Sideband designation UC indicates upper-side-band I/Q encoding, while LC indicates lower-side-band I/Q encoding. LC data is identical to UC except for the sign of the imaginary component.
5. If the channel link is null it indicates that the channel should be set-up, but not recorded. It is not necessary to specify chan_def statements when whether the channels are set up is not important. This feature of omitting the channel link can be used for example to specify the frequency setting of an unused analog BBC to avoid accidental interference in a recorded channel.
6. The channel name field is provided as a convenience for processing by correlators. This field is intended to be populated and used only by the correlators. Its suitability for any given application is beyond VEX. The providing user is responsible for how it is interpreted. It is not intended to be used to pass information to other users.

Example $FREQ def blocks

```
def X4;
*          Band     Sky freq     Net      Chan    Chan    BBC     Phase-cal
*           ID     at 0Hz BBC     SB       BW      ID      ID        ID
  chan_def= &X :   8210.99 MHz :  U  :   2 MHz : &CH1  : &BBCa  : &USB_CAL  ;
  chan_def= &X :   8210.99 MHz :  L  :   2 MHz : &CH2  : &BBCa  : &LSB_CAL  ;
  chan_def= &X :   8212.99 MHz :  U  :   2 MHz : &CH3  : &BBCb  : &USB_CAL  ;
  chan_def= &X :   8212.99 MHz :  L  :   2 MHz : &CH4  : &BBCb  : &LSB_CAL  ;
enddef;

def S4;
*          Band     Sky freq     Net      Chan    Chan    BBC     Phase-cal
*           ID     at 0Hz BBC     SB       BW      ID      ID        ID
  chan_def= &S :   2210.99 MHz :  U  :   2 MHz : &CH5  : &BBCc  : &USB_CAL  ;
  chan_def= &S :   2210.99 MHz :  L  :   2 MHz : &CH6  : &BBCc  : &LSB_CAL  ;
  chan_def= &S :   2212.99 MHz :  U  :   2 MHz : &CH7  : &BBCd  : &USB_CAL  ;
  chan_def= &S :   2212.99 MHz :  L  :   2 MHz : &CH8  : &BBCd  : &LSB_CAL  ;
enddef;
```

~~$HEAD_POS Block~~ (Not in VEX2)

~~The $HEAD_POS block defines the headstack positioning as a function of 'headstack-position reference number' for Mark IIIA, Mark IV, and VLBA systems; the $HEAD_POS block is irrelevant for other types of recording systems.. One headstack_pos statement is required for each potential headstack position (or set of positions if multiple simultaneous recording headstacks).~~

~~Parameter~~	~~Field~~	~~Description~~	~~Type~~	~~Allowed values~~	~~Units~~	~~Comments~~
~~headstack_pos~~	~~1~~	~~Position reference number~~	~~int~~	~~>0~~	~~-~~	~~-~~
~~-~~	~~2~~	~~Headstack 1 position~~	~~int~~	~~range of headstack motion~~	~~length~~	~~Allowed range depends on system, but is typically -1000 to +1000 um.~~
~~-~~	~~(3)~~	~~Headstack 2 position~~	~~int~~	~~range of headstack motion~~	~~length~~	~~Required only if headstack 2 is being used.~~
~~-~~	~~(4)~~	~~Headstack 3 position~~	~~int~~	~~range of headstack motion~~	~~length~~	~~Required only if headstack 3 is being used.~~
~~-~~	~~(5)~~	~~Headstack 4 position~~	~~int~~	~~range of headstack motion~~	~~length~~	~~Required only if headstack 4 is being used.~~

Example $HEAD_POS block

```
def MARK4/2_HDSTKS;
*                    pos   hdstk1    hdstk2    hdstk3    hdstk4
*                    ref#   pos       pos       pos       pos
   headstack_pos =    1  : -319 um : -271 um;
   headstack_pos =    2  :   31 um :   79 um;
   headstack_pos =    3 :  -223 um : -175 um;
   headstack_pos =    4  :  127 um :  175 um;
   headstack_pos =    5  : -127 um :  -79 um;
   headstack_pos =    6  :  223 um :  271 um;
enddef;
```

$IF Block (Minor changes and augmented for VEX2)

The $IF block defines the characteristics of the IF bands used in the observations and is linked to the $BBC block (which specifies the detailed BBC-to-IF connections). An if_def statement must be defined for each of the IF 'links' specified in the selected $BBC 'def'.

Parameter	Field	Description	Type	Allowed values	Units	Comments
if_def	1	'IF_ID' link word	&link			One if_def statement must be present for each input connector (channelize before sample) or connector/sampler (sample before channelize) ~~separate IF.~~
	2	~~Physical IF name~~	~~char~~			Always null. See note 2 ~~System dependent. Used to create procedures to select proper IF. See Notes.~~
	3	Polarization	char	R | L | X | Y | H | V		
	4	Total effective LO of IF (just before signal enters BBC or sampler)	real	>=0	freq	A value of 0 (with units) indicates direct RF sampling. ~~Positive number~~
	5	Net sideband of IF	char	U | L | D		
	(6)	Phase-cal frequency interval	real		freq	Typically 1 or 5 MHz. Null or omission indicates no phase-cal.
	(7)	Phase-cal base frequency	real		freq	Usually 0, in which case may be null or omitted.
	(8)	IF sample rate	real		sample rate	Applicable only for sample before channelize systems.
receiver_name	1	'IF_ID' link word	&link			One receiver_name statement may be present for each separate IF.
	2	Receiver name	char			The name of the receiver, e.g., 6cm
sub_lo_frequencies	1	'IF_ID' link word	&link			One sub_lo_frequencies statement may be present for each separate IF.
	2..n	List of 1st..(n-1)th sub LO frequencies	real		freq	Units specification may be omitted after first field.
sub_lo_sidebands	1	'IF_ID' link word	&link			One sub_lo_sidebands statement may be present for each separate IF.

Parameter	Field	Description	Type	Allowed values	Units	Comments
	2..n	List of sidebands of the 1st..(n-1)th conversion	char	U | L | D		
switched_power	1	'IF_ID' link word	&link			One switched_power statement may be present for each separate IF.
	2	Amplitude	char	High | Low | Off		
	(3)	Switching frequency	real		freq	Full cycle of on and off.

Notes:

1. The 'total effective LO' is used in conjunction with the total sky frequency specified for each channel in the $FREQ block to calculate the local-oscillator setting in each individual BBC.
2. The second field of if_def is always null. If needed, the physical connection is specified in a connection statement for the IF_ID link in $DAS. There may be only one connection statement per IF_ID link. ~~The 'Physical IF Name' is a system-dependent designation specifying which IF is selected. For the VLBA system, IF's A, B, C, and D may each be selected with either a 'Normal' or 'External' input, leading to designations AN, AE, BN, BE, CN, CE, DN, and DE. For the Mark III system, IF's 1, 2, and 3 may each be selected with either a 'Normal' or 'Alternate' input, leading to the designations 1N, 1A, 2N, 2A, 3N, 3A.~~
3. The sub_lo_... parameters can be used to specify intermediate conversion steps that lead to the net effect given by if_def.
4. Stations support switched_power on a best effort basis. If the requested configuration is not supported, some other, possibly default, configuration will be used.
5. A switched_power switching frequency of 0 implies constant added noise.

Example `$IF def` blocks

```
  def VLBA/X;
  *           IF       Pol    Total     Net    Phase-cal     P-cal base
  *           ID                 LO        SB   freq spacing      freq
    if_def= &IF_XR1 : : R :   7600 MHz :  U  :  1 MHz      :  0 Hz  ;
    if_def= &IF_XR2 : : R :   9600 MHz :  L  :  1 MHz      :  0 Hz  ;
    if_def= &IF_XL1 : : L :   7600 MHz :  U  :  1 MHz      :  0 Hz  ;
    if_def= &IF_XL2 : : L :   9600 MHz :  L  :  1 MHz      :  0 Hz  ;
  enddef;

  def VLBA/S;
  *           IF       Pol    Total     Net
  *           ID                 LO        SB
    if_def= &IF_SR  : : R :   2900 MHz :  L  ;           *no phase-cal
    if_def= &IF_SL  : : L :   2900 MHz :  L  ;           *no phase-cal
    receiver_name      = &IF_SR : 13cm ;
    sub_lo_frequencies = &IF_SR : 2900 MHz ; * this is a trivial example
    sub_lo_sidebands   = &IF_SR : L ;    * this is a trivial example
    receiver_name      = &IF_SL : 13cm ;
    sub_lo_frequencies = &IF_SL : 2900 MHz ; * this is a trivial example
    sub_lo_sidebands   = &IF_SL : L ;    * this is a trivial example
    switched_power     = &IF_SR : Low : 80 Hz ; * not sure there is any other
configuration available at VLBA
    switched_power     = &IF_SL : Low : 80 Hz ; * not sure there is any other
configuration available at VLBA
  enddef;

  def CDP/SX_NARROW;
  *           IF        Pol    Total     Net    Phase-cal
  *           ID                 LO        SB   freq spacing
    if_def= &IF_XR  : : R :   8080 MHz :  U  :  1 MHz  ;
    if_def= &IF_SR  : : L :   2020 MHz :  U  :  1 MHz  ;
    switched_power = &IF_XR : Off ; * if supported
    switched_power = &IF_SR : Off ; * if supported
  enddef;

  def VLBA/X_DBE; *DBE sampling-before-channelization system;
  *           IF       Pol   Total     Net Phase-cal   P-cal base  IF sample
  *           ID                 LO       SB  frq spacing  freq         rate
    if_def= &IF_XR1 : : R : 7600 MHz : U  : 1 MHz    :  0 Hz   : 1024 Ms/sec;
    if_def= &IF_XR2 : : R : 9600 MHz : L  : 1 MHz    :  0 Hz   : 1024 Ms/sec;
    if_def= &IF_XL1 : : L : 7600 MHz : U  : 1 MHz    :  0 Hz   : 1024 Ms/sec;
    if_def= &IF_XL2 : : L : 9600 MHz : L  : 1 MHz    :  0 Hz   : 1024 Ms/sec;
  enddef;

  def VLBA/S_DBE;
```

```
*          &IF     Pol    Total     Net  P-cal   P-cal   IF Sample
*          lnk             LO       SB   space   base      rate
  if_def= &IFSX : : X : 2688.4 MHz : U : 5 MHz : 0 Hz : 1024 Ms/sec ;
  if_def= &IFSY : : Y : 2688.4 MHz : U : 5 MHz : 0 Hz : 1024 Ms/sec ;
  if_def= &IFCX : : X : 4736.4 MHz : U : 5 MHz : 0 Hz : 1024 Ms/sec ;
  if_def= &IFCY : : Y : 4736.4 MHz : U : 5 MHz : 0 Hz : 1024 Ms/sec ;
enddef;
```

$PASS_ORDER Block (Not in VEX2)

The $PASS_ORDER block defines pass and group ordering relevant for Mark IIIA, Mark IV, VLBA and S2 systems. For Mark IIIA, Mark IV and VLBA system, each pass is defined by a two-part field composed of a numeric 'headstack-position reference' (defined in the selected $HEAD_POS 'def') followed by an alphabetic 'subpass identifier' (defined in the selected $TRACKS 'def'), e.g., 2A. For S2, the S2_group_order statement defines the order of usage of tape groups. This section is thought to be obsolete as disc-based recorders have no such concept.

Parameter	Field	Description	Type	Allowed values	Units	Comments
pass_order	>1	First pass ID	char	-	-	e.g., 1A. First pass assumed to be in 'forward' direction. For S2, specified group#.
-	2	Second pass ID	char	-	-	e.g., 2A
-	...	etc	-	-	-	-
S2_group_order	1	First group number	int	-	-	Specifies order in which S2 groups are to be recorded.
-	2	Second group number	int	-	-	-
-	...	etc	-	-	-	-

Notes:

1. The number of fields present in the pass_order or S2_group_order statements specifies the number of tape passes or groups. Number of cassettes per group is defined by recording mode specified in $TRACKS section. S2 groups are numbered starting at 0.
2. First pass is assumed to be in the forward tape-motion direction (Mark IIIA, Mark IV, VLBA).

Example $PASS_ORDER def blocks

```
  def VLBA/32;                                  *32-tracks recording simultaneously
  *                  F     R     F     R     F     R     F     R     F     R     F     R     F
R
     pass_order = 1A : 2A : 3A : 4A : 5A : 6A : 7A : 8A : 9A : 10A : 11A : 12A : 13A
: 14A ;
  enddef;

  def MARK4/64;                                 *64-tracks recording simultaneously
  *                  F     R     F     R     F     R
     pass_order = 1A : 2A : 3A : 4A : 5A : 6A ;
  enddef;
—
  def S2/4;                                     *4 groups of 4 cassettes each
  *                     grp grp grp grp
     S2_group_order = 0 : 1 : 2 : 3 ;
  enddef;
```

$PHASE_CAL_DETECT Block

The `$PHASE_CAL_DETECT` block is used to specify the phase-cal tones to be detected at the observing station.

Parameter	Field	Description	Type	Allowed values	Units	Comments
`phase_cal_detect`	1	pcal_ID	&link			Links to a `chan_def` statement in selected `$FREQ` def.
	2	Tone number (from DC edge of BBC output)	int			Tone number of first tone to be detected. See tone-number definition in Notes.
	(3)	Tone number	int			Tone number of second tone to be detected
	...	etc				

Note:

1. The actual phase-cal frequencies are determined by the LO frequencies specified in the `$IF` and `$FREQ` blocks.
2. The phase-cal frequency spacing are specified in the `$IF` block.
3. Tone number 1 is defined as first tone above 0Hz in the BBC output channel, etc. Phase-cal detection will be done on the set of specified 'tone#'s, which are listed in order of preference in case more tones are specified than can be detected by the hardware. The tones are numbered positively from the low (DC) edge of the BBC output band, with tone number `1` being the first tone *above* DC. Tones may also be specified as negative numbers corresponding to their position from the nominal band edge, with tone number `-1` being the first tone below nominal band edge. A tone number of `0` specifies state counting, rather than phase-cal detection, should take place.

Example $PHASE_CAL_DETECT def blocks

```
  def STANDARD;
  *                     pcal_ID  tone#  tone#
    phase_cal_detect = &USB_CAL :  1  :  -1  ;   *detect lowest and highest
frequency tones in band
    phase_cal_detect = &LSB_CAL :  1  ;          *detect lowest-frequency tone in
band
  enddef;
```

$PROCEDURES Block (Deprecated in VEX2)

This block is deprecated for VEX2.

The $PROCEDURES block specifies parameters relevant to various procedures at an observing station. Timing parameters are to be used as constraints to the scheduling program. The set of timing parameters listed is for the NASA 'sked' program. Other scheduling programs may use these and/or other parameters.

Parameter	Field	Description	Type	Allowed values	Units	Comments
tape_change	1	Tape-change time	real		time	Required parameter
headstack_motion	1	Time to complete headstack motion	real		time	Required parameter
new_source_command	1	Time to initiate pointing to new source	real		time	Required parameter
new_tape_setup	1	Time to setup system for new tape	real		time	Required parameter
setup_always	1	Setup system for each observation	char	on | off		Optional procedure
	2	Time to setup system	real		time	
parity_check	1	Do parity check	char	on | off		Optional procedure
	2	Time needed to do parity check	real		time	
tape_prepass	1	Do tape prepass	char	on | off		Optional procedure
	2	Time needed to do tape prepass	real		time	
preob_cal	1	Pre-observation calibration	char	on | off		Optional calibration procedure
	2	Time needed for procedure	real		time	
	3	Procedure name	char			
midob_cal	1	Mod-observation calibration	char	on | off		Optional calibration procedure
	2	Time needed for procedure	real		time	
	3	Procedure name	char			
postob_cal	1	Post-observation calibration	char	on | off		Optional calibration procedure
	2	Time need for procedure	real		time	
	3	Procedure name	char			
procedure_name_prefix	1	Specify standard procedure library	char			Specifies that a 'standard' procedure library is to be used.

Notes:

1. The `procedure_name_prefix` parameter is intended to allow the specification of frequently-used procedure libraries that individual stations may have honed to their particular requirements. For instance, the geodesy community frequently uses the same station setup over and over again (e.g., so-called 'SX2C' setup). The specification of the 'standard procedures' for a station relieves the requirement of creating a new set of procedures for an experiment.

Example $PROCEDURES `def` blocks

```
  def STANDARD1;
  *Required procedures
    tape_change =          420 sec;
    headstack_motion =         6 sec;
    new_source_command =     5 sec;
    new_tape_setup =        20 sec;
  *Optional procedures
    setup_always =   on :   20 sec;
    parity_check =   on :   70 sec;
    tape_prepass =   off:  600 sec;
  enddef;

  def VLBA_CAL;                              * Calibration procedure timings
  *                     time   procedure
  *                     req'd     name
    preob_cal  =  on : 10 sec : preob;
    midob_cal  =  on : 20 sec : midob;
    postob_cal =  on : 20 sec : postob;
  enddef;

  def STANDARD_CAL;
    preob_cal  =  on : 10 sec : preob;
    midob_cal  =  on : 15 sec : midob;
    postob_cal = off : 20 sec : postob;     * Procedure not used; no time will be
allocated
  enddef;

  def SX2C;
    procedure_name_prefix = SX2C;                    *prescribes use of standard
library of procedures SX2C
  enddef;
```

~~$ROLL Block~~ (Not in VEX2)

The $ROLL block defines the barrel-rolling sequence that may be used in VLBA and Mark4 recording systems. It is intentionally defined in a very general way, but for the most part will probably be confined to a few 'canned' modes. The roll sequence is specified with a roll statement for each track participating in the barrel roll, plus statements defining the roll period and re-initialization interval.

Parameter	Field	Description	Type	Allowed values	Units	Comments
roll	1	Roll on/off	char	on|off	-	Optional. See Notes.
roll_def	1	Headstack #	int	1-4	-	-
-	2	Home track	int	track #	-	Track# that would be written in the absence of barrel-roll.
-	3	Step 0 destination track	-	-	-	Track to which home track is written when barrel-roll is initialized (step 0)
-	4	Step 1 destination track	int	track #	-	Track to which home-track is 'switched' on first increment of barrel-roll.
-	5	Step 2 destination track	int	track #	-	Track to which home-track is 'switched' on second increment of barrel-roll.
-	etc	-	-	-	-	-
-	n+2	Last-step destination track in n-step sequence	int	track #	-	Track to which home-track is 'switched' on last step of barrel-roll. Returns to Step 0 as next step.
roll_inc_period	1	Roll increment period in frames	int	-	(frames)	-
roll_reinit_period	1	Roll-sequence re-initialization period in seconds (at recording)	real	any	time	Fixed at 2 sec for VLBA. Mark4 can be specified.

Notes:

1. Barrel-roll is confined to tracks 2-33 within a given headstack. For cases of barrel-roll using multiple headstacks, the roll sequence definition must include all headstacks.
2. Barrel-roll is applied in the formatter as the last step before the data are written to tape.
3. System tracks do not participate in barrel-roll.
4. The number of fields in the roll statements defines number of positions in the roll sequence. All roll statements must specify the same number of positions in the roll sequence.
5. Note that all track# references elsewhere in the VEX file are to the 'home track#'.
6. roll=off; is default. Presence of any roll_def statements implies roll=on; .
7. Barrel rolling is essentially obsolete in the disk-based recording era.

Example $ROLL def blocks

```
  def VLBA/8;                                  *standard 8-track 8-position VLBA barrel-
roll
    roll = on;
    roll_reinit_period = 2 sec;                *barrel-roll sequence reinitialized every 2
sec
    roll_inc_period = 1;                       *barrel-roll increment period (frames)
  *
  *                      -------------output track----------------------
  *          hdstk home  roll  roll  roll  roll  roll  roll  roll  roll
  *                 trk  step0 step1 step2 step3 step4 step5 step6 step7
    roll_def=  1  :  2 :   2 :   4 :   6 :   8 :  10 :  12 :  14 :  16;
    roll_def=  1  :  4 :   4 :   6 :   8 :  10 :  12 :  14 :  16 :   2;
    roll_def=  1  :  6 :   6 :   8 :  10 :  12 :  14 :  16 :   2 :   4;
  * several lines skipped
    roll_def=  1  : 18 :  18 :  20 :  22 :  24 :  26 :  28 :  30 :  32;
    roll_def=  1  : 20 :  20 :  22 :  24 :  26 :  28 :  30 :  32 :  18;
    roll_def=  1  : 22 :  22 :  24 :  26 :  28 :  30 :  32 :  18 :  20;
    roll_def=  1  : 24 :  24 :  26 :  28 :  30 :  32 :  18 :  20 :  22;
  * several more lines skipped
    roll_def=  1  : 29 :  29 :  31 :  33 :  19 :  21 :  23 :  25 :  27;
    roll_def=  1  : 31 :  31 :  33 :  19 :  21 :  23 :  25 :  27 :  29;
    roll_def=  1  : 33 :  33 :  19 :  21 :  23 :  25 :  27 :  29 :  31;
  enddef;
```

$SCHEDULING_PARAMS Block

The $SCHEDULING_PARAMS block specifies various parameters needed for the scheduling program. Since each scheduling program may have its own unique set of parameters, the $SCHEDULING_PARAMS block is specified strictly as a literal block which must be parsed and interpreted by the relevant scheduling program. The parameters listed are some of those for the current version of the NASA 'sked' program. Other scheduling programs may use these and/or other parameters. Literal blocks are often used in this block.

Note: These parameters are examples only!

Parameter	Field	Description	Type	Allowed values	Units	Comments
sched_program	1	Scheduling program name	char			
	2	Revision	char			
default_scan_length	1	Default scan length			time	
lookahead	1	Lookahead for source rise/set			time	
min_scan_length	1	Minimum scan length			time	
minimum_between_scans	1	Minimum time between scans			time	
modular_scan_length	1	Schedule on minute marks			time	
max_display_width_col	1	Display screen width			(columns)	
confirm	1	Confirm new scans		on | off		
mutual_vis	1	Force all stations to see source, or allow subnet		all | subnet		
low_SNR_reject	1	Reject stations if SNR too low		auto | man		Primarily for geodesy
variable_scan_length	1	Use SNR calculation to set scan length		on | off		Primarily for geodesy
min_sun_angle	1	Min angle between source and sun			angle	
tape_usage_sync	1	Synchronize tape usage		on | off		
sked_optimize	1	Type or optimization for auto-scheduling		sky_coverage | covariance		Primarily for geodesy
window	1	Sliding window for optimization			time	Primarily for geodesy
maximize_num_obs	1	Maximum total # of observations		on | off		Primarily for geodesy
minimize_idle	1	Minimize idle time between scans		on | off		Primarily for geodesy
minimize_slew	1	Minimize antenna slew time		on | off		Primarily for geodesy

Example $SCHEDULING_PARAMS def block

```
def SKED1;
  start_literal();
    sched_program = SKED:Rev_950715
    *time control
    default_scan_length = 196 sec
    lookahead = 20 sec
    min_scan_length = 60 sec
    minimum_between_scans = 0 sec
    modular_scan_length = 10 sec
    *
    *user interface
    max_display_width_col = 79
    confirm = on
    mutual_vis = subnet
    low_SNR_reject = auto
   *user control values
    variable_scan_length = on
    min_sun_angle = 15 deg;
    tape_usage_sync = on
    sked_optimize = sky_coverage
    window = 1 hr
    maximize_num_obs = on
    minimize_idle = on
    minimize_slew = on
  end_literal();
enddef;
```

$SEFD Block

The optional $SEFD block allows the sensitivity of each IF to be modeled and used for a crude calculation of expected SNR when used with specified observing modes an d scan-length times. For geodesy, these calculations can be used to automatically adjust scan times for minimum-acceptable SNR in order to densify the schedule as much as possible. The particular SEFD model to be used can be specified, along with the model parameters.

Parameter	Field	Description	Type	Allowed values	Units	Comments
`sefd_model`	1	SEFD model name	char	`Shaffer` ├?		Models may be added
`sefd`	1	IF_ID 'linkword'	&link			Link to 'IF_ID' in selected `$IF` 'def'
	2	Zenith SEFD	real		flux-density	
	3, ...	Model parameters				

Example `$SEFD` `def` blocks

```
def VLBA-FD;
  sefd_model = Shaffer;            *defines model be used
*          IF ID     SEFD      Model parameters
  sefd = &IF_XR1  :  750 Jy  :  1.0  : 0.954  : 0.0464;
  sefd = &IF_XR2  :  760 Jy  :  1.0  : 0.974  : 0.0263;
  sefd = &IF_XL1  :  750 Jy  :  1.0  : 0.573  : 0.0470;
  sefd = &IF_XL2  :  750 Jy  :  1.0  : 0.549  : 0.0470;
enddef;

def HS;
  ref External.File:$SEFD = HS;
enddef;
```

`$SITE` Block (Minor changes and augmented for VEX2)

The `$SITE` block describes the location of an antenna and may be either earth-based or earth-orbiting. Horizon masks for earth-based sites may be specified as an aid in scheduling.

Parameter	Field	Description	Type	Allowed values	Units	Comments
site_type	1	Type of site	char	fixed | earth_orbit		
site_name	1	Full site name	char	<=16chars		
site_ID	1	Standardized 2-char site code	char	2 chars		
	(2)	Ad hoc 1-char site code	char	1 char		
site_position	1	x	real		length	
	2	y	real		length	
	3	z	real		length	
site_position_epoch	1	Epoch of site_position	epoch		epoch	
site_position_ref	1	Reference for site position	char			Origin of the position information, e,g., analysis center/solution.
site_position_frame	1	Reference frame for the site position	char			For example, ITRF2020.
site_velocity	1	x-velocity	real		velocity	
	2	y-velocity	real		velocity	
	3	z-velocity	real		velocity	
horizon_map_az	1..n	List of azimuth values corresponding to values in horizon_map_el	real		angle	Units specification may be omitted after first field.
horizon_map_el	1..n	List of elevation limits at azimuths specified in horizon_map_az	real		angle	Units specification may be omitted after first field.
zen_atmos	1	Zenith atmosphere added delay	real		time	Typically on order of 7 nsec
ocean_load_vert	1	Ocean-loading vertical amplitude	real		length	
	2	Phase	real		phase	
ocean_load_horiz	1	Ocean-loading horiz amplitude	real		length	
	2	Phase	real		phase	

Parameter	Field	Description	Type	Allowed values	Units	Comments
occupation_code	1	4-char occupation code	char			Primarily used for geodetic experiments
inclination	1	Earth-orbit parameter	real		angle	
eccentricity	1	Earth-orbit parameter	real		-	
arg_perigee	1	Earth-orbit parameter	real		angle	
ascending_node	1	Earth-orbit parameter	real		angle	
mean_anomaly	1	Earth-orbit parameter	real		angle	
semi-major_axis	1	Earth-orbit parameter	real		length	
mean_motion	1	Earth-orbit parameter	real		-	
orbit_epoch	1	Earth-orbit epoch	epoch		epoch	

Notes:

1. The horizon_map_el parameter has either same number of fields as horizon_map_az or one less. If there are the same number of fields, the pairs of values define points that are connected by line segments. If there is one less value for elevation, then the (n)th elevation value defines the height of a "step" between the (n)th and (n+1)th azimuth values.

Example $SITE def blocks

```
  def EFLSBERG;
    site_type=fixed;
    site_name=EFLSBERG;       *full station name
    site_ID=EF;               *standard 2-char identifier
    *                  x           y           z
    site_position = 123456.4 m: 5432112.6 m: 563675.2 m;
    site_position_epoch = 1994y1d; site_position_ref = GLB914F1;
    site_velocity = 0.17 cm/yr: 0.025 cm/yr: 0.12 cm/yr;
    horizon_map_az=0
deg:20:55:60:70:85:100:105:115:220:230:245:270:280:300:305:330:345:360;
    horizon_map_el= 5 deg: 3: 6: 7: 5: 6:  5:  4:  3:  4:  3:  5:  4:  3:  4:  5:
6:  5;
    zen_atmos=7 nsec;
    ocean_load_vert = 3 cm: 90 deg;
    ocean_load_horiz =0.5 cm: 52 deg;
    occupation_code=a478;
  enddef;

  def HAYSTACK;
    ref <external_file>:$SITE = HAYSTACK;
  enddef;

def VSOP;                                          *satellite
    site_type = earth_orbit;
    site_name = SAT1;
    site_ID=VS;
    inclination = 1.06 deg;
    eccentricity = 0.000313;
    arg_perigee = 67.3883 deg;
    ascending_node = 43.9513 deg;
    mean_anomaly = 297.168289 deg;
    semi-major_axis = 42166.129 km;
    mean_motion = 0.;                              *rev/sidereal-day
    orbit_epoch = 1995y44d7h;
  enddef;
```

$SOURCE Block (Augmented for VEX2)

The $SOURCE block defines the sources to be observed and specifies their relevant characteristics, particularly position. A single source is defined in each 'def' block. A crude source model may be specified for each observed 'Band_ID' specified in the $FREQ block for purposes of auto-scheduling (primarily geodesy).

FIXME: remove source_type options earth_satellite and ephemeris as these are not well enough defined at this moment.

Parameter	Field	Description	Type	Allowed values	Units	Comments
source_type	1	Generic source type	char	star | earth_satellite | bsp | tle | ephemeris		Specifies coordinate system in which position of object will be specified. Other types may be added. See Note 5
	(2)	Experiment source type	char	target | calibrator | dummy		dummy may be declared if the source if specified for pointing purposes only. Station software may use this information.
	(3)	Source coordinate system sub-type	char	radec | retarded | nonretarded		This field is only for the ephemeris type.
source_name	1	Source name	char	<= 16 char		~~Typically same as 'def' label name (e.g., 3C273B)~~ See Note 6

Notes:

1. The only source type that can safely be assumed to be supported by all stations and correlators is star *without* use of any of ra_rate, dec_rate, and per scan/station offsets (the latter from the $SCHED block). The degree of support for other source types differ among stations and correlation facilities. Certain mechanisms for conveying ephemerides may be intrinsically unsuitable for precision correlation. It is up to the writer of the VEX file to ensure the facilities involved support any modes used. The primary goal for these other source types is to allow distribution of pointing information to stations.
2. Only one source type may be specified per source.
3. Support for types other than star should be considered experimental with the possibility that future versions of VEX will not be compatible with the options presented here. Reasonable effort will be made to maintain backward compatibility to the extent it is possible and useful to do so.
4. For types other than star and ephemeris sub-type radec, references for algorithms and code/programs/libraries to implement them should be included in this document when available.
5. UTC is assumed for source types that require a time coordinate.
6. At least some VEX1 readers used the def *keyword* as the source name when referenced in the $SCHED source parameter. The source name should come from source_name parameter. To reduce the risk of losing data, All VEX writers are encouraged to use the same string for the def *keyword* and in the source_name parameter to the extent possible.

For source type star:

Parameter	Field	Description	Type	Allowed values	Units	Comments
`IAU_name`	1	Standard IAU source ID	char	9-char		Example: `0102-0304`
`source_position_ref`	1	Origin of source position	char	<= 16-char		For traceability of source position
`ra`	1	Right-ascension	RA		RA	Example: `01h02m03.456s`
`dec`	1	Declination	Dec		Dec	Example: `-03d04'05.678"`
`ref_coord_frame`	1	Source-position reference frame	char	`B1950` | `J2000`		
`ra_rate`	1	RA proper motion	real		ang rate	Typically asec/yr
`dec_rate`	1	Declination proper motion	real		ang rate	Typically asec/yr
`source_position_epoch`	1	Epoch of stated position	epoch		epoch	Needed only if non-zero `ra_rate` or `dec_rate`
`source_model`	1	Component number	int			One `source_model` statement for each major source component for each 'Band_ID' link to the selected `$FREQ` 'def'
	2	'Band_ID' linkword to selected $FREQ 'def'	&link			
	3	Component flux-density	real		flux-density	
	4	Component major axis	real		angle	Angle subtended on sky
	5	Component axis ratio	real			
	6	Component position angle	real		angle	
	7	Component RA offset wrt specified source position	real		angle	
	8	Component dec offset wrt	real		angle	

Parameter	Field	Description	Type	Allowed values	Units	Comments
		specified source position				

Notes:

1. Depending on what the station supports, `ra_rate` and `dec_rate` may be used to describe source position evolution for pointing purposes.
2. Depending on what the station supports, per scan/station radec offsets in `$SCHED` block may be used to correct for station diurnal parallax fot pointing purposes.

For source type `earth_satellite`:

Parameter	Field	Description	Type	Allowed values	Units	Comments
`inclination`	1	Orbit inclination	real		angle	
`eccentricity`	1	Orbit eccentricity	real			Unitless
`arg_perigee`	1	Argument of perigee	real		angle	
`ascending_node`	1	Longitude of ascending node	real		angle	
`mean_anomaly`	1	Orbit mean anomaly	real		angle	
`semi-major_axis`	1	Orbit semi-major axis	real		length	
`mean_motion`	1	Orbit mean motion	real			
`orbit_epoch`	1	Epoch of stated orbit	epoch		epoch	

Notes:

1. References: TBD

For source type `bsp`:

Parameter	Field	Description	Type	Allowed values	Units	Comments
`bsp_file_name`	1	filename containing the ephemeris	char			Consumer defined. See Notes 1 & 2
`bsp_object_id`	1	object number within ephemeris file	int			positive or negative

Notes:

1. Since the VEX file will be used both at observe time and correlation time, explicit paths for the ephemeris file will cause trouble. Each consumer of vex+bsp files should establish their own strategy for handling this.
2. The VEX specification does not rule out use of a URL (e.g., `"http://www.bsp.org/moon.bsp"`. However implementation of this is up to each consumer and may be limited by availability of network infrastructure.

For source type `tle`:

Parameter	Field	Description	Type	Allowed values	Units	Comments
`tle0`	1	0th line of TLE	char			quoted verbatim comon source name for TLE, 20 characters
`tle1`	1	1st line of TLE	char			quoted verbatim 1st line of TLE, 69 characters
`tle2`	1	2nd line of TLE	char			quoted verbatim 2nd line of TLE, 69 characters

Notes:

1. The `tle` source type is for earth satellites with orbits specified by NORAD Two Line Elements (optionally including the 0th line with the satellite name). The `tle0`, `tle1`, and `tle2` parameters represent the TLE lines as quoted verbatim strings (there are embedded spaces).
2. References: The elements provided in TLEs can be propogated using the SGP4/SDP4 algorithms. The *predict* prgram, URL: http://www.qsl.net/kd2bd/predict.html, can be used to calculate azimuth and elevation for observing the satellite using the TLE data as provided here.

For source type `ephemeris`, sub-type `radec`:

Parameter	Field	Description	Type	Allowed values	Units	Comments
`datum`	1	Time	epoch			Timestamp for this table entry
	2	RA	RA		RA	
	3	Dec	Dec		Dec	
	(4)	RA rate	real		angular rate	
	(5)	Dec rate	real		angular rate	
`ref_coord_frame`	1	Source-position reference frame	char	`B1950` | `J2000`		

Notes:

1. Coordinates are geocentric. Depending on what the station supports, per scan/station radec offsets in $SCHED block maybe used to correct for station diurnal parallax.

For source type `ephemeris`, sub-types `retarded` and `nonretarded`:

Parameter	Field	Description	Type	Allowed values	Units	Comments
`state_vector`	1	Time	epoch			Timestamp for this table entry
	2	X	real		length	Geocentric X position
	3	Y	real		length	Geocentric Y position
	4	Z	real		length	Geocentric Z position
	(5)	VX	real		velocity	Geocentric X rate
	(6)	VY	real		velocity	Geocentric Y rate
	(7)	VZ	real		velocity	Geocentric Z rate
`ref_coord_frame`	1	Reference frame	char	`GCRF`		For now, only `GCRF` is supported.

Notes:

1. The coordinates provided are the apparent coordinates at the specified time stamp appropriately retarded, or not, for light travel time referred to the center of the Earth for sub-types `retarded`, and `nonretarded`, respectively. Use of `retarded` values should be avoided.
2. Values for a `state_vector` can be supplied for single epoch.
3. References: TBD.

Example $SOURCE def blocks

```
  *
  def HD123456;
    source_type = star : calibrator;                *source type may be declared as
'dummy' if for pointing purposes only
    source_name = HD123456; IAU_name = 0102-0304;
    ra=01h02m03.456s; dec = -03d04'04.567"; ref_coord_frame = J2000;
source_position_ref = GLB923Z;
    ra_rate = 1.e-2 asec/yr; dec_rate = -1.e-3 asec/yr; source_position_epoch =
1995y;
    *             comp# Band   Flux      MajAxis  Ratio    PA     Raofst  DECofst
    *                   ID
    source_model =  1  : &X : 1.10 Jy : .20 asec : 1.0 : 0 deg : 0 asec : 0 asec ;
    source_model =  1  : &S : 2.40 Jy : .22 asec : 1.5 : 5 deg : 0 asec : 0 asec ;
    *Note: Source model is for scheduling purposes only; 'freq band' is 'link' to
the selected $FREQ def.
  enddef;

  def 3C123;
    source_type = star : target;
    ref <external_file>:$SOURCE = 3C123;
  enddef;

  def SAT1;
    source_type = earth_satellite;
    source_name = SAT1;
    inclination = 1.06 deg;
    eccentricity = 0.000313;
    arg_perigee = 67.3883 deg;
    ascending_node = 43.9513 deg;
    mean_anomaly = 297.168289 deg;
    semi-major_axis = 42166.129 km;
    mean_motion = 0.;                                *rev/sidereal-day
    orbit_epoch = 1995y44d7h;
  enddef;

  def DeathAsteroid;                                 *Note: eVLBI needed for this
experiment
      source_type = ephemeris : : retarded ;
      source_name = KILLER1;
      datum = 2010y270d18h00m00s : 0 km : 0 km : 1000 km : 0 km/sec : 0 km/sec :
-100 km/sec ;
      datum = 2010y270d18h00m05s : 0 km : 0 km : 300 km : 0 km/sec : 0 km/sec : -120
km/sec ;
      ref_coord_frame = GCRF;
  enddef;
```

```
def gps-32;
   source_type = tle;
   tle0='GPS BIIA-10 (PRN 32)';
   tle1='1 20959U 90103A   12129.58081476 -.00000017  00000-0  10000-3 0  5553';
   tle2='2 20959  54.5341 240.2762 0119562 325.8431  33.4117  2.00560570157098';
enddef;
```

$TAPELOG_OBS Block (Minor change for VEX2)

The $TAPELOG_OBS block contains information about the media (historically tape, but now exclusively disks) that were used during the experiment. This table is normally appended after the completion of an experiment. Each station can have multiple media listed which can in principle overlap in time if parallel recording occurred.

Parameters in this table are:

Parameter	Field	Description	Type	Allowed values	Units	Comments
VSN	1	Recorder number	int			
	2	Volume Serial Number (VSN)	char			8 character string, conventionally all capital
	3	Start time	epoch		epoch	Epoch when this disk started
	4	Stop time	epoch		epoch	Epoch when this disk stopped
	(5...n)	Data/Bit-stream link	link			Specifies which bit/data-streams are recorded on this VSN, may not be a merged datastream link, see note 1.

Notes:

1. The data/bit-stream links are used to identify which streams are recorded on a particular VSN if the data are not distributed across all VSNs for a given start to stop period.

Example $TAPELOG_OBS def blocks

```
def Ef;
  VSN = 1 : MET-0011 : 2010y064d01h29m59s : 2010y064d14h45m25s;  * shelf = BE86
  VSN = 1 : NTO-0015 : 2010y064d14h45m40s : 2010y064d21h45m25s;  * shelf = BE95
  VSN = 1 : WSRT-043 : 2010y064d21h45m45s : 2010y065d01h30m00s;  * shelf = BE96
enddef;
def Fd;
  VSN = 1 : NRA0+257 : 2010y064d01h30m00s : 2010y065d01h30m00s;  * shelf = BE44
enddef;
```

$TRACKS Block (Substantially changed for VEX2)

The $TRACKS block defines the various multiplex (~~fan-in and~~ fan-out) modes that can be used to record data with Mark 5A recorders. ~~on the Mark3A, Mark4, and VLBA DAS's. In cases where a mode uses fewer than the full number of heads in a single pass, alphabetical 'sub-passes' are defined (tape passes with the headstacks in a fixed position).~~

For purposes of multiplex definitions, the sample data from each channel are separated into a 'sign' bitstream and (for 2-bit sampling) a 'magnitude' bitstream. The fan-out modes (single bitstream to 1, 2 or 4 tracks) are defined with a set of fanout_def statements, one such statement for each bitstream ~~and subpass~~, which defines the destination tracks and bit ordering among the tracks. In this way a complete definition of the multiplex format is specified. The 'ChanID' linkword in each fanout_def statement connects a particular bitstream to the selected 'def' in the $FREQ block.

~~The fan-in modes (1, 2 or 4 bitstreams to a single track) are defined by a set of fanin_def statements. Each such statement defines the bitstreams written to a single track in a specified subpass. The set of 'ChanID' linkwords in each fanin_def statement connects the particular bitstreams to the selected 'def' in the $FREQ block. and specifies their multiplex order on the track.~~

~~Within each fanout_def or fanin_def statement is a field which specifies the 'sub-pass' to which it applies. A 'subpass' is defined as a single tape pass for which the headstack(s) are held at a fixed position. Typically, for example, 16 of 32 tracks may be written in a single tape pass; for this case there are 2 sub-passes with a given headstack position. By convention, the subpasses are labeled A, B, C, ...,etc.~~

~~Note that, except for the VLBA_trnsprt_sys_trk statement, all references to 'track numbers' in the $TRACKS block are more properly labeled as 'home tracks' since barrel-rolling in the formatter (Mark 4 and VLBA) and track switching within the recorder (VLBA only) may lead to modified physical track assignments. Normally, the actual physical track numbers (pre-barrel-rolled) correspond identically to 'home track' numbers.~~

Parameter	Field	Description	Type	Allowed values	Units	Comments
fanout_def	1	~~Sub-pass ID~~	~~char~~	~~single char~~		One fanout_def statement required for each bitstream. This field is always null.~~By convention, subpass_ID uses characters A, B, C, etc. Null for Mark 5A.~~
	2	'Chan_ID' linkword	char	-		Link to 'Chan_ID' in selected $FREQ def
	3	Sign or magnitude bitstream	char	sign | mag		Fields 2 and 3 uniquely define a single bitstream. The samples are always real.
	4	Headstack number	int	1-4		
	5	First multiplex track	int	track#		Track # within headstack
	(6)	Second multiplex track	int	track#		Required for fanout 1-to-2 or 1-to-4
	(7)	Third multiplex track	int	track#		Required for fanout 1-to-4
	(8)	Fourth multiplex track	int	track#		Required for fanout 1-to-4. Number of fields specifies fan-out ratio.
track_frame_format	1	Frame format on tape track	char	Mark3A | Mark4 | VLBA | LBA_AT | LBA_VSOP		Mark3A and Mark4 are slightly different data-replacement formats. VLBA is non-data-replacement format. VLBA can write Mark3A format.

Parameter	Field	Description	Type	Allowed values	Units	Comments
sample_rate	1	Sample frequency	real		sample rate	
~~fanin_def~~	~~1~~	~~Sub-pass number~~	~~int~~	~~single char~~	~~-~~	~~By convention, uses characters A,B,C,...etc.~~
	~~2~~	~~Headstack number~~	~~int~~	~~1-4~~		
	~~3~~	~~Track number~~	~~int~~	~~track#~~		
	~~4~~	~~'Chan_ID' linkword for multiplex bitstream 1~~	~~char~~			~~Link to 'Chan_ID' in selected $FREQ def~~
	~~5~~	~~sign or magnitude bitstream (of 'Chan_ID')~~	~~char~~	~~sign|mag~~		~~Fields 4 and 5 uniquely define the bitstream which occupies bit position 1 on the specified track.~~
	~~(6)~~	~~'ChanID' linkword for multiplex bitstream 2~~	~~char~~			~~Fields 6 and 7 required if fanin is 2-to-1 or 4-to-1~~
	~~(7)~~	~~sign or magnitude bitstream~~	~~char~~	~~sign|mag~~		~~Fields 6 and 7 uniquely define the bitstream which occupies bit position 2 on the specified track.~~
	~~(8)~~	~~'ChanID' linkword for multiplex bitstream 3~~	~~char~~	~~-~~		~~Fields 8 and 9 required if fanin is 4-to-1~~
	~~(9)~~	~~sign or magnitude bitstream~~	~~char~~	~~sign|mag~~		~~Fields 8 and 9 uniquely define a bitstream which occupies bit position 3 on the specified track.~~
	~~(10)~~	~~'ChanID' linkword for multiplex bitstream 4~~	~~char~~			~~Fields 10 and 11 required if fanin is 4-to-1~~
	~~(11)~~	~~sign or magnitude bitstream~~	~~char~~	~~sign|mag~~		~~Fields 10 and 11 uniquely define a~~

Parameter	Field	Description	Type	Allowed values	Units	Comments
						bitstream which occupies bit position 4 on the specified track.
data_modulation	1	Pseudo-random data modulation	char	on|off		Default is 'off'.
VLBA_frmtr_sys_trk	1	Formatter 'system' track# to be written with specified data	int	0|1|34|35		Applicable to VLBA formatter only. Specifies data to be written to a particular 'system' track formatter outputs.
	2	Data type to be written to system track specified in Field 1	char	xtk_parity|duplicate		xtk_parity if cross-track parity to be written; =duplicate is this track is to duplicate one of the normal data tracks.
	3A	If Field 2 is xtk_parity: First track# of contiguous set of tracks covered	int	2|10|18|26		Limited to specified set of 'first track #'s'
	4A	If Field 2 is xtk_parity: #tracks covered by cross-parity	int	8|16		
	3B	If Field 2 is duplicate: 'home track#' of data to be written to specified 'system' track#	int	2-33		Will always be 'home track' data; is not barrel-rolled
VLBA_trnsprt_sys_trk	1	Physical 'system' track (head#) to be written as a duplicate of specified formatter output	int	0|1|34|35		This is a duplication within the transport itself, so includes all barrel-roll

Parameter	Field	Description	Type	Allowed values	Units	Comments
		track (recorder input track)..				
	2	Formatter output track (recorder input track) to be duplicated	int	2-33		
S2_recording_mode	1	Recording mode ID	char	See Notes		Example: '32x4-2'
S2_data_source	1	Define S2 data source	char	Mark4_formatter | VLBA_BBC_1_4 | VLBA_BBC_5_8		Relevant for S2 only
	2	Define BBCx selection from Mark 4 formatter	&link			For Mark IV only: Link to 'BBC_ID' in $FREQ. See Notes.
	3	Define BBCy selection from Mark 4 formatter	&link			For Mark IV only: Link to 'BBC_ID' in $FREQ. See Notes.

Notes:

1. Reference to Mark IV Memo 230 (aka VLBA Acquisition Memo 393) may help to clarify the details of multiplex and signal switching in the Mark IV and VLBA systems.
2. The actual fanin/fanout ratio is implied by the number of subfields in the `fanout_def` or =fanin_def= statements, as indicated in the above table.
3. When recording 2-bit samples, the bits are identified as *sign* and *mag*, as defined in Mark IV memo 205.7. The samples are actually encoded as `00`, `01`, `10`, and `11`, in order, going from most negative to most positive voltage. The first bit (MSB) is truly the sign, while the second bit (LSB) is encoded for statistical uniformity to keep the tape channel happier. The designation *sign* and *mag* is used here, even though not quite correct, so that the 1-bit sample case is designated as `sign`, which is both accurate and more comfortable than 'MSB'. LBA_VSOP is mag/sign encoding, and LBA_AT is offset binary (same convention as VLBA).
4. For a fanin/fanout ratio of 1-to-1 (i.e., one bitstream to one track), either `fanout_def` or `fanin_def` statements may be used.
5. Cross-track parity is computed after barrel-rolling as the last step before writing to tape.
6. The 'S2_recording_mode' parameter specifies the recording mode to which the S2 system is to be set. The available modes are documented in 'S2-RT User's Manual, Version 3.0 (162)', October 1996, ISTS-SGL-TR96-033, available at `ftp://s2.sgl.ists.ca`. The selected mode defines the number of recording 'groups' and recorded inputs. The `S2_group_order` parameter in the `$PASS_ORDER` section specifies the order in which the groups are to be recorded.
7. The 'S2_data_source' parameter specifies the origin of the sampled data recorded by the S2 recording system. Typical data sources are either through the so-called 'phase-cal' outputs of the Mark IV formatter or direct sampler outputs from the VLBA system.

8. Except for the `VLBA_drive_sys_track` statement, all references to track numbers in the `$TRACKS` block are more properly labelled as formatter-output numbers since signal switching within the recorder may lead to re-arranged physical track assignments. The *track number* label has been retained for convenience.

Mark IV formatter track assignments : The sampled data from the USB and LSB outputs of each of two selected BBC's can be directed to the 'phase-cal output'. The data available to the S2 are unformatted 2-bit samples at 32 Msamples/sec, regardless of the sample rate chosen for output to the Mark IV recording system. If we designate the two selected Mark IV BBC's as BBCx and BBCy, the Canadian VIA (VLBI Interface Adapter) implements a fixed mapping to S2 inputs as follows:

S2 Input	**IN0**	**IN1**	**IN2**	**IN3**	**IN4**	**IN5**	**IN6**	**IN7**
Mk4 bit stream	Lx/sign	Lx/mag	Ux/sign	Ux/mag	Ly/sign	Ly/mag	Uy/sign	Uy/mag

VLBA formatter track assignments : The input to the Canadian VIA box may be taken either from BBC's 1-4 or BBC's 5-8, depending on the physical connector to which it is attached. The mapping within the VIA is fixed, as follows:

S2 Input	**IN0**	**IN1**	**IN2**	**IN3**	**IN4**	**IN5**	**IN6**	**IN7**	**IN8**	**IN9**	**IN10**	**IN11**	**IN12**	**IN13**	**IN14**	**IN15**
BBC1-4	U1/s	U1/m	L1/s	L1/m	U3/s	U3/m	L3/s	L3/m	U2/s	U2/m	L2/s	L2/m	U4/s	U4/m	L4/s	L4/m
BBC5-8	U5/s	U5/m	L5/s	L5/m	U7/s	U7/m	L7/s	L7/m	U6/s	U6/m	L6/s	L6/m	U8/s	U8/m	L8/s	L8/m

The information in the above tables is taken from the 'VLBI System Interface Adapter (VIA) User's Manual, Ver. 1.3', ISTS/SGL, October 25, 1996, available at `ftp://s2.sgl.ists.ca`.

Example $TRACKS `def` blocks

```
  def MARK5A/XX-4-2/8;                  *VLBA mode XX-4-2 8-tk mode recorded on
Mark5A
  *
  *                sub- trksID   sign/
  *                pass           mag   hdstk  trk1
    fanout_def   =     :&CH1  :  sign  :  1  :  2         ; *1-to-1 fanout
    fanout_def   =     :&CH1  :   mag  :  1  :  4         ;
    fanout_def   =     :&CH2  :  sign  :  1  :  6         ;
    fanout_def   =     :&CH2  :   mag  :  1  :  8         ;
    fanout_def   =     :&CH3  :  sign  :  1  : 10         ;
    fanout_def   =     :&CH3  :   mag  :  1  : 12         ;
    fanout_def   =     :&CH4  :  sign  :  1  : 14         ;
    fanout_def   =     :&CH4  :   mag  :  1  : 16         ;
  enddef;
```