CVEX Parameter Tables, Rev 1.06

Alan R. Whitney

29 May 2001

Introduction

The CVEX file specifies the detailed correlator configuration for each task according to a specified 'mode' key associated with each task.

Rules of Correlator Resource Allocation

As outlined in Mark IV Memos IM081 and IM100, the minimum unit of correlator resource allocated to a single task is one 'slice', consisting of four correlator boards, one in each of four segments. Other possible allocations to a single task are two or four slices. Not all boards within a slice are necessarily active; however, those that are not utilized are not available to a concurrent task and will lie idle. Due to the signal distribution scheme of the correlator (see referenced memos), each correlator 'section' has access to four (and only four) SU outputs from each of up to 16 SU's. Therefore, processing of 16 (or 14, perhaps) channels, as will be the common case, requires the use of at least one board in each section.

The CVEX Block Structure

The CVEX block structure is indicated diagrammatically as follows, where each indentation implies another level of referencing:

$CVEX_REV	specifies revision level of CVEX file
$CORR_CONFIG	highest-level keys for correlator configuration
$CORR_MODE	specifies corr config by board (as segment and slice)
$CORR_BD_PARMS	specifies several board-level parameters
$CORR_SECTION_MODE	specifies section config by chip
$CORR_CHIP_MODE	specifies chip config by block
$CORR_BLOCK_MODE	specifies detailed block config

The cvex organization has been tailored to easily and compactly define the modes in the style of those defined in Mark IV Memos IM081 and IM100, though it is not limited to that. In fact, virtually any configuration can be created, at the expense of the cvex complexity.

Note that the cvex structure is heavily nested. Since 'normal' correlator configurations are quite redundant from board, this nested structure allows such configurations to be specified with considerable compactness. On the other hand, the flexibility exists to configure every chip in the system differently, again at the expense of cvex complexity.

Virtual Parameters

Flexibility and ease of re-use of cvex specifications is enhanced by the use of 'virtual parameters'. Virtual parameters are specified in vex statements as named parameters prefaced by the tilde character ('~'). Most of the virtual parameters have to do with signal routing within the correlator and are assigned values by the correlator manager software at runtime. Because these virtual parameters are assigned values by a 'smart' agent at runtime, a single cvex correlator-configuration-specification can easily accommodate the dynamic requirements of specific SU and channel assignments. For special setups, the virtual parameters may be replaced by actual parameters, in which case the specified parameter values will be used.

The use of virtual parameters is particularly valuable in the assignment of mirror snakes. After the correlator manager has assigned the set of primary snakes to process a scan, any unallocated snakes are easily assigned as mirrors simply by setting 'leftover' virtual parameters to the appropriate values.

Virtual parameters are designated '~vp' in the parameter tables below.

Cross-Correlation Channel Matching

In normal 'one-to-one' processing, corr_man will assign cross-correlation snakes only to channels from different stations with the same channel names (as defined in the ovex $FREQ block). The only exception to this rule is in the case of cross-polarization processing, where corr_man assumes that the channel names for 'matching' cross-polarized channels differ only in the last character (for example, 'X1R' and 'X1L').

The above rules can work only if each channel-pair to be cross-correlated are both available to a correlator board. In the normal wiring configuration between the SU's and the correlator, each correlator section receives only four channels from each SU; if the two channels to be cross-correlated are not available in the same correlator section, they cannot be correlated and corr_man will signal and error.

The 'name-matching' rules stated above may be overridden by an 'ignore_chan_names=on' statement in the $CORR_CONFIG block. In this case, cross-correlation pairs are assigned only on the basis of their physical SU output channel number. For example, each SU chan 0 output would be cross-correlated only with other SU chan 0 outputs, etc. This capability is primarily useful in testing and diagnostic situations.

Mirroring

An important aspect of correlator quality control and system monitoring is correlator 'mirroring'. Mirroring allocates otherwise-unused resources to duplicate the functionality of primary correlator 'snakes', which allows continual cross-checking of the raw correlation results for functional verification of all correlator elements. Mirror cross-checking normally is done by the DSP's on the correlator board and engages no resources outside of the correlator board itself. Normally, mirroring correlators elements are allocated and operated completely transparently to the operators, making themselves visible only when mirroring cross-checks fail.

CVEX Rules

Each cvex $block is allowed to specify only parameters which are defined for that type of $block. The CVEX Parameter Tables define in detail these parameters for each type of $block.

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.

$CVEX_REV Block

The $CVEX_REV block specifies the revision level of the parameters structure of the CVEX file

Parameter	Field	Description	Type	Allowed values	Units	Comments
rev	1	cvex revision number	real	1.0		Will be revised as necessary

$CORR_CONFIG Block

The $CORR_CONFIG block specifies the high-level task-specific correlator configuration. A 'def' is specified for each possible correlator configuration, one of which must be chosen for each task. Each such 'def' contains a 'ref' and an optional parameter:

Parameter	Field	Description	Type	Allowed values	Units	Comments
ref $CORR_MODE	-	correlator config as reference to $CORR_MODE keyword	ref			Required
ignore_chan_names	1	Causes corr_man to ignore channel names when matching channel for correlation		on|off		Optional. Default is 'off'. See 'Cross-Correlation Channel Matching' discussion above.
auto-corr	1	Force auto-corr off/on		on|off		Optional. Default is 'on'. 'off' causes all auto-corr setup to be ignored.

Notes:

1. Note: By convention, the keywords for correlator modes are of the form 'XYY-ZZ' where 'X' specifies the number of slices in the mode ('A'=4,'B'=2,'C'=1), 'YY' the number of lags, and 'ZZ' specifies the number of channels/stn (for example, 'B32-16').

$CORR_MODE Block

Each $CORR_MODE 'def' specifies a correlator configuration for a particular task with a 'board=' statement for each correlator board in the configuration. Each 'board=' statement specifies the board configuration by keyword reference for each board section, as well as the channel-pair to be correlated within each board section.

Parameter	Field	Description	Type	Allowed values	Units	Comments
board	1	correlator segment number	~vp| int	0-3		Combination of segment and slice numbers specify a particular board. Normally assigned by corr_man
	2	correlator slice number	~vp| int	0-3		Normally assigned by corr_man
	3	reference to keyword in $CORR_BD_PARMS block	keyword	-		Specifies several board-wide processing parameters
	4a	section config as reference to $CORR_SECT_MODE keyword	keyword			Specifies board-section configuration
	4b	fist corr chip# in section	int	0-31		Section will be anchored to this chip number
	4c	'reference' chan name for cross-correlation channel-pair	~vp| char	Chan name		Normally assigned by corr_man according to rules discussed above.
	4c	'remote' chan name for cross-correlation channel-pair	~vp| char	Chan name		Normally assigned by corr_man to be same as 'reference' chan
	(5a,b,c,d)	parameters for 2nd section, if any				
	(6a,b,c,d)	parameters for 3rd section, if any				
	(7a,b,c,d)	parameter for 4th section, if any				Typically, max of 4 sections/corr bd (see notes)
cross_pol_chan_pair	1	Chan name of first chan of cross-polarized chan-pair	~vp| char	Chan name		One 'cross_pol_chan_pair=' statement for each cross-polarized channel pair. Relevant only for cross-polarization processing
	2	Chan name of second (matching)	~vp|	Chan		

		chan name of cross-polarized chan-pair	char	name		

Notes:

1. Following the correlator configuration guidelines of Mark IV memos IM081 and IM100, each correlator board can be divided into 1, 2 or 4 sections where each section processes all baselines for a single channel.
2. The virtual 'channel names' are dynamically replaced with real channel names at runtime, courtesy of corr_man.
3. The 'cross_pol_chan_pair=' statement allows corr_man to identify which channels specified in the 'board=' statements are matching cross-polarized pairs so that proper assignments can be made.
4. The physical chips on each correlator board are number 0-31, with each connected to the next highest-numbered chip so that cascading may be used.
5. Each correlator-board section normally consists of an adjacent set of chips; the particular set of chips in a section is defined in the $CORR_SECT_MODE block. Obviously, no chip may be defined to be in more than one section on a given board.

$CORR_BD_PARMS Block

Each $CORR_BD_PARMS 'def' specifies several high-level correlator-board wide parameters.

Parameter	Field	Description	Type	Allowed values	Units	Comments
accum_divide_ratio	1	decimation ratio applied to chip accumulators	int	1,2,4,8,16		Causes accumulators to be enabled only on every n^{th} data sample (accumulators are active only outside of BOCF).
shsmp_divide_ratio	1	decimation ratio applied during BOCF	int	1,2,4,8,16		During BOCF, allows only every n^{th} data sample to shift data into header buffer-capture or correlator shift registers
	2	decimation ratio applied outside of BOCF		1,2,4,8,16		Outside of BOCF, allows only every n^{th} data sample to shift data into correlator shift registers.
sample_count_per_lag_enable	1	Enable valid-sample count on every lag.	char	on|off		If 'on', reduces number of lags by half.

Notes:

1. Note that 'accum_divide_ratio' and 'shsmp_divide_ratio' are *with respect to the SU output data rate,* and not the SU output clock rate, which is fixed at ~32 MHz). For example, to process every second data sample, set 'accum_divide_ratio=2' (irrespective of record or playback sample-data rate).
2. 'accum_divide_ratio' and 'shsmp_divide_ratio' can be used together, for example, to reduce the effective sample rate for oversampled data.
3. The phasing of the decimator is always such that the first sample of BOCF or the first sample after BOCF are accepted, followed by every n^{th} following sample.

$CORR_SECT_MODE Block

Each $CORR_SECT_MODE 'def' specifies the configuration of a correlator-board section with a 'chip=' statement for each chip within the section. Each 'chip=' statement specifies the chip configuration (as a keyword reference), the chip number within a section, and the signal source for each chip input (usually specified as a virtual station).

Parameter	Field	Description	Type	Allowed values	Units	Comments
chip	1	chip config by reference to $CORR_CHIP_MODE keyword	keyword			Specifies chip configuration; require one 'chip=' statement for each chip in section.
	2	relative chip number in section	int	0-31		
	3	station to which chip input X0 is to be connected	~vp| stn name| null			Normally a virtual parameter of the form 'stn*x*' (x=integer) assigned by corr_man at runtime. If 'stn name', must be single-char name. 'Null' if input is not used or is cascaded to/from an adjacent chip.
	4	station to which chip input X1 is to be connected	same			
	5	station to which chip input X2 is to be connected	same			
	6	station to which chip input X3 is to be connected	same			

Notes:

1. Normally, the chips within a each section are adjacent, though the statement format allows non-adjacent chips for special purposes.
2. By convention, virtual station names are of the form '~stn*x*', where *x* is a hex integer. Also by convention, the ordering of the 'stn*x*'s' is done such that lower-numbered ~stn's are always the 'reference' station in a correlation with a higher-numbered '~stn'.

$CORR_CHIP_MODE Block

Each $CORR_CHIP_MODE 'def' specifies the configuration of a correlator chip with a 'block=' statement for each of the eight blocks within a correlator chip. Each 'block=' statement specifies the block parameters as a keyword reference to the block mode and explicit signal-muliplexer settings. In addition, each snake or snake segment within the chip is defined with a 'snake=' statement.

Parameter	Field	Description	Type	Allowed values	Units	Comments
block	1	corr-chip block config as reference to $CORR_BLOCK_MODE keyword	keyword			Specifies block mode (see Mark IV memos 237, IM100).
	2	block name	char	A0-A3 B0-B3		Block name (see Mark IV memos 237, IM100)
	3	multiplexer M0 setting	int	0-7		
	4	multiplexer M1 setting	int	0-7		
	5	multiplexer M2 setting	int	0-7		
	6	multiplexer M3 setting	int	0-7		
snake	1	snake type	char	complex| real|		Specifies whether data is to be treated as complex, real, or auto-correlation. Determines how data is

				auto		treated and written to output files
	2	chip input for ‘reference’ stn	X0|X1| X2|X3| null			Relevant (and required) if ‘head block’ of snake is on this chip. Null if head block not on this chip.
	3	chip input for ‘remote’ stn	X0|X1| X2|X3| null			Relevant (and required) if ‘tail block’ of snake is on this chip. Null if ‘auto’ mode or tail block not on this chip.
	4	First (head) cell in snake or snake segment or, if cascade to/from adjacent chip, ‘pointer’ to cell in adjacent chip (see Notes)	char	A0l-A3l A0r-A3r B0l-B3l B0r-B3r		If ‘pointer’ to cell on previous or next-chip (cascade case), preface with ‘-’ or ‘+‘, respectively. If block lag-read-order is in order of decreasing delay, add a ‘-‘ suffix.
	5	Next cell in snake, in order of increasing delay	char	same		
	….	Next cell in snake, in order of increasing celay	char	same		
	….	Last (tail) cell in snake or snake segment or, if cascade to/from adjacent chip, ‘pointer’ to cell in adjacent chip	char	same		If ‘pointer’ to cell on previous or next-chip (cascade case), preface with ‘-’ or ‘+‘, respectively.

Notes:

1. One ‘block=’ statement must exist for each block with the chip.
2. Each block must be referenced once, and once only, in a ‘snake=’ statement.
3. Cross-correlation snakes are specified as ‘cell chains’ in order from the head to tail of the snake. In the case of ‘complex’ snakes, only the ‘real’ part of the snake is specified; the ‘imag’ part is assumed to be composed of the complement cells in the corresponding blocks.
4. Each snake segment (i.e. each ‘snake=’ statement) can contain up to 8 cells (within chip) plus optional ‘pointers’ (at snake-segment ends) to cells in adjacent chips (multichip snakes), for a total of up to 10 entries.
5. For multichip snakes, a cell ‘pointer’ at the beginning or end of a snake segment and identified by a ‘+’ or ‘-‘ prefix, points to the next cell in the snake in the preceding or following chip, respectively, in chip-number order. The first snake segment, for example, might end with a ‘-‘ cell pointer pointing to a matching snake segment in the previous chip, and the last snake segment might start with a ‘+’ cell pointer pointing to a matching snake segment in the following chip. [The external X outputs of each chip can be connected to the X inputs of the *preceding* chip. The Y inputs are opposite.] For long complex snakes, the head and tail are typically in the same chip (two snake segments), but the snake loops out through other chips and back.
6. Note that autocorrelation snakes are listed beginning from the signal-input end, so maximum lag is at the beginning and zero lag is at the end.
7. Multichip snakes cannot cross board or section boundaries.
8. Snakes cascaded from chip-to-chip proceed from higher to lower chip numbers (i.e. a 256-complex-lag snake would have inputs on chip #31 and cascade to chip #30, for example).

$CORR_BLOCK_MODE Block

Each $CORR_BLOCK_MODE 'def' specifies the configuration of a correlator-chip block.

Parameter	Field	Description	Type	Allowed values	Units	Comments
rotator_mode	1	left (real) cell rotator mode	int	0-3		0 – pass data samples unchanged 1 – logical 1 multiplied by cosine from phase gen 2 – not used 3 – data sample multipled by cosine from phase gen
	2	right (imag) cell rotator mode	int	0-3		same, except sine
xdelay	1	left cell xdelay	int	0|1		1 – add one-sample delay to x-data path (see Figure 2 of Mark IV memo IM081)
	2	right cell xdelay	int	0|1		same
ydelay	1	left cell ydelay	int	0|1		1 – add one-sample delay to y-data path
	2	right cell ydelay	int	0|1		same
tap_motion_enable	1	enable tap motion	int	0|1		0 – force vernier delay into zero-delay state (both left and right cells) 1 – enable tap control by delay generator
header_mode	1	set disposition of data during BOCF header	int	0-3		Controls the validity of data samples during BOCF: 0 – pass validity data unchanged 1 – force sample data invalid 2 – force sample data valid 3 – not used
invalid_on_tap_motion	1	set data in block invalid on tap motion	int	0|1		Invalidate data in block when tap motion occurs; relevant only if 'sample-count per lag' is enabled in $CORR_BD_PARMS block

Notes:

1. Typical configurations are for complex head, mid and tail blocks, as well as auto-correlation blocks.

Update History

7 Sept 1999: First issue

28 February 2000: Add 'auto-corr=' statement to $CORR_CONFIG section.

16 July 2000: Update $CORR_CHIP_MODE section for multi-chip snakes.