NATIONAL RADIO ASTRONOMY OBSERVATORY
Green Bank, West Virginia

Spectral Processor Memo No. 33

June 23, 1988

To: Spectral Processor Group
and Mailing List

From: J. Richard Fisher

Subj: **Caesar Keywords Associated with the Spectral Processor**

The subject report follows.

JRF/cjd

Caesar Keywords Associated with the Spectral Processor

This document is a first cut at specifying the keywords needed in the telescope control command language to set up the spectral processor. Some of the words are new and some are old ones modified to handle more than one value, i.e. turned into arrays. The keywords are listed in three different ways. The first associates each keyword with its reference in the data structures in the Caesar code and tells on which SODA screen(s) that keyword should appear. The second gives a brief description of the function of each keyword. The third specifies which keywords are needed to determine the value of each setup parameter in the setup packet data structures sent from the spectral processor MassComp to the rack controller.

The following keywords are used to set up the spectral processor. With each keyword are its reference in shared memory A, the SODA screens on which it should be found, and legal values if there are a finite number of them. String values may be any combination of upper and lower case and may be shortened to a length that resolves all ambiguities but not shorter than 3 characters. A few keywords will appear on screens not associated with the spectral processor, but these are not listed here. New keywords are marked with #, and old keywords that have been changed to arrays or to different array indices are marked with $. The subscript N always has a dimension of 2 and refers to racks A and B. A short description of the keywords follows this table.

	Keyword	Shared Mem. A Ref.	SODA Screens
	clock.source "Internal", "External"	Spectrom.clock_source	Spectrometer
#	if.lo.source.N.M "Internal", External"	LO_tuning.if_lo_source[2][8]	LO Tuning Spectrometer
$	num.spectr.N 1, 2, 4, 8	Spectrom.num_spectr[2]	Spectrometer
$	num.chan.N [2, 4, 8, 16, 32, 64], 128, 256, 512, 1024 See keyword notes.	Spectrom.num_chan[2]	Spectrometer
#	taper.N.M (Available taper names)	Spectrom.taper[2][2]	Spectrometer
#	taper.select.N 1, 2	Spectrom.taper_select[2]	Spectrometer
#	taper.offset "Yes", "No"	Spectrom.taper_offset	Spectrometer
#	max.pwr.N	Spectrom.max_pwr[2]	Spectrometer
#	atod.input.lev.N.M	Spectrom.atod_input_lev[2][8]	Spectrometer
	band.width.N 40, 20, 10, 5, 2.5, 1.25, 0.625, 0.3125, 0.15625, 0.078125 MHz	Setup.band_width[2]	Setup Spectrometer

```
  observ.mode           Setup.observ_mode              Setup
        "Line", "Pulsar", among others for
         other backends

# multiplier.mode       Spectrom.multiplier_mode       Spectrometer
        "Square", "Cross", "Sqr/Cross"

$ ifs.N.M               LO_tuning.ifs[2][8]            LO Tuning

# if.sideband.N.M       LO_tuning.if_sideband[2][8]    LO Tuning
        "Upper", "Lower"                               Spectrometer

# clock.freq            Spectrom.clock_freq            Spectrometer

# clock.freq.deriv      Spectrom.clock_freq_deriv      Spectrometer

$ num.phase.N           Switch_ph.num_phase[2]         Front-end Switching

$ phase.N.M             Switch_ph.phase[2][8]          Front-end Switching
        "Signal", "Reference", "On Cal", "Off Cal"

$ blank.time.N          Switching.blank_time[2]        Front-end Switching

$ phase.time.N.M        Switching.phase_time[2][8]     Front-end Switching

$ sample.time.N         Contin_back.sample_time[2]     Spectrometer

# excise.N.M            Rfi.excise[2][8]               RFI
        "Yes", "No"

# fast.time.const.N.M   Rfi.fast_time_const[2][8]      RFI
        1us, 3us, 10us, 30us, 100us,
        300us, 1ms, 3ms, 10ms

# slow.time.const.N.M   Rfi.slow_time_const[2][8]      RFI
        100us, 300us, 1ms, 3ms, 10ms,
        30ms, 100ms, 300ms, 1sec

# clip.level.N.M        Rfi.clip_level[2][8]           RFI

# threshold.N.M         Rfi.threshold[2][8]            RFI

  utc.start             Data_cntl.utc_start            Data Control

  utc.stop              Data_cntl.utc_stop             Data Control

$ pulsar.period.N       Pulsar.pulsar_period[2]        Pulsar

# pulsar.period.deriv.N #Pulsar.pulsar_period_deriv[2] Pulsar

$ dispersion.measure.N  Pulsar.dispersion_measure[2]   Pulsar

# processor.mode.N      Pulsar.processor_mode[2]       Pulsar
        "DedispTimeSamples", "TimeSyncSpectra",
        "SigAvgDedisp", "SpectraBurst",
        "VoltageTimeSamples", "PowerTimeSamples",
        "PulsePhaseSpectra", "SyncFreqTime"
```

$ num.window.N	Pulsar.num_window[2]	Pulsar
$ begin.window.N.M	Pulsar.begin_window[2][8]	Pulsar
$ end.window.N.M	Pulsar.end_window[2][8]	Pulsar

Below is a rough description of the purpose of each keyword and the units where applicable.

clock.source — Controls whether the spectral processor master clock is driven from its internal 160 MHz oscillator or from the external synthesizer.

if.lo.source.N.M — Controls whether an IF drawer has its frequency set by its internal synthesizer with 10 kHz resolution or by the external synthesizer with 10 Hz resolution.

$ num.spectr.N — Number of IF inputs to each rack (1, 2, 4, or 8).

$ num.chan.N — Number of spectral channels assigned to each IF input. For normal operation this parameter is nearly redundant with "num_spectr", the exception being 2 IF's with either 256 or 512 channels. The num_spectr/num_chan selections are (8/128, 4/256, 2/256, 2/512, and 1/1024). If a smaller number of channels than is permitted by 'num_spectr' is specified, frequency averaging will be done in the accumulator before sending data to the MassComp. A smaller number of channels could be chosen to get spectra more rapidly since the data rate into the MassComp is a limiting factor. The number must be a power of two.

taper.N.M — Specifies the weighting function to be applied to the A/D amplitude-vs-time series before it is transformed. A small number of taper functions are stored in the spectral processor and will be referred to by name in this keyword. The index M may be 1 or 2 since two taper functions may be loaded into the hardware to be selected from with the keyword 'taper.select.N'. In spectral processor hardware memos the taper function is called a window, but this name is used in another keyword for a different purpose.

taper.select.N — Selects one of two taper functions in hardware for each rack. In other words, it tells which value of M in "taper.N.M" is active. Values are either 1 or 2.

taper.offset — Selects whether the data sampling for the time series to be transformed in racks A and B are started together or are offset by half of a series length. If both racks look at the same IF signal, the offset provides inhanced sensitivity when spectra from the same IF are added together. The penalty is that there are half as

many spectral channels to spread around the frequency and polarization dimensions. A value of "Yes" puts the offset into effect.

max.pwr.N — Specifies the largest signal that can be present in the output spectrum without overflowing the 16-bit accumulator input. The data from the square/cross multiplier is a 32-bit word, but only the top 16 bits are presented to the accumulator. This parameter sets left-shifting of the two 16-bit input words to the multiplier to between one and four bits. Greater shift means a smaller maximum power. The unit for this parameter is the fraction of total noise power in the passband. This can be less than or greater than one.

atod.input.lev.N.M — Sets the noise level at the input to the flash A/D convertors. The unit is fraction of noise rms voltage per quantization interval, normally between about 1.0 and 4.0. Lower numbers give more large signal handling room in the A/D but compromize sensitivity and baseline stability because of quantization effects. The IF attenuators will be set to produce an input level within about 0.5 dB of the one specified.

band.width.N — Sets the total bandwidth in each IF passband. All bandwidths within one rack must be the same. Legal selections are 40, 20, 10, 5, 2.5, 1.25, 0.625, 0.3125, 0.15625, and 0.078125 MHz. The total bandwith for one rack is 40 MHZ. In other words, the product of the number of IF's (num.spectr) and the bandwidth of one IF must be 40 MHZ or smaller.

observ.mode — Specifies the general type of observing to be done with the spectral processor or any other back-end for that matter. Only two types refer to the spectral processor: "Line" and "Pulsar." "Line" specifies straight spectrum averaging for fixed integration periods possibly with cal and front-end switching. All other accumulation modes fall under "Pulsar."

multiplier.mode — Sets the multiplier following the real correction to square the output of each complex FFT, cross multiply the outputs from racks A and B, or do both if the bandwidth permits the multiplier to run twice as fast as the FFT. Legal values are "Square", "Cross", and "Sqr/Cross".

$ ifs.N.M — Center frequencies of IF passbands. After taking into account the baseband offset and other conversions in the IF drawer, these parameters set the synthesizer frequencies in the IF drawers or the frequency of the high resolution synthesizer if an external IF LO is selected.

if.sideband.N.M — Selects the active single-sideband convertor sideband. With "Upper" sideband, increasing frequency at the IF corresponds to increasing frequency at baseband. With

"Lower" sideband, increasing IF produces decreasing frequency at baseband.

# clock.freq	Sets the high resolution synthesizer frequency when it is being used to drive the spectral processor master clock. The normal value is 160 MHz. and all legal values are less than or equal to this frequency.
# clock.freq.deriv	Sets the rate of change of clock.freq in typical units of Hz/Sec. When this value is not zero, 'clock.freq' is the frequency at 'utc.start'.
$ num.phase.N	Sets the number of phases in the front-end switch cycle when observ.mode = "Line". The state of each phase is given by 'phase.N.M', the length of each phase is given by 'phase.time.N.M', and the time between phases is given by 'blank.time.N'. The number of phases may be from 1 to 8.
$ phase.N.M	Specifies the state of each front-end switching phase when observ.mode = "Line". The number of active phases is set by 'num.phas.N'. Legal values are "Signal", "Reference", "On Cal", "Off Cal".
$ blank.time.N	Specifies the time between phases when no spectra are accumulated in observ.mode = "Line". The same blanking time is used between all active phases.
$ phase.time.N.M	Specifies the accumulation time on each front-end switch phase in one switch cycle. The number of active phases in a cycle is given by 'num.phase'.
$ sample.time.N	Specifies the total accumulation time of one data dump to the spectral processor MassComp. This time must be an integral number of switch cycles in whatever observing mode is being used.
# excise.N.M	Turns on or off automatic RFI excision based on total power threshold pulse detection for each IF channel. Legal values are "Yes", "No" where "Yes" means that excision is active.
# fast.time.const.N.M	Specifies the response time of the total power pulsed RFI threshold detector. The optimum value depends on the nature of the RFI but is often roughly equal to the characteristic pulse length of the interference. Legal values are 1us, 3us, 10us, 30us, 100us, 300us, 1ms, 3ms, and 10ms.
# slow.time.const.N.M	Specifies the response time of the baseline comparison level for the total power pulsed RFI threshold detector. The optimum value depends on the nature of the RFI and is usually more than 10 times the characteristic time scale of the pulsed interference. RFI detection occurs when the fast-time-constant

response exceeds the slow-time-constant response by the 'threshold.N.M' value. Legal values are 100us, 300us, 1ms, 3ms, 10ms, 30ms, 100ms, 300ms, and 1sec.

clip.level.N.M — Sets the maximum level that can be instantaneously applied to the input of the slow baseline integrator. This clipping prevents severe over-charging and, hence, slow recovery of this integrator. This parameter is specified in fraction of the total IF power and must be greater than 1.0.

threshold.N.M — Specifies the level difference between the outputs of the fast and slow integrators that will flag an RFI pulse. This is specified in fraction of total IF power. It may be considerably less than unity, but it should be greater than about five times the reciprocal of the square root of the product of the IF bandwidth and the fast time constant to reduce the chance of flagging normal noise.

utc.start — Determines the precise time of taking the first data sample in the first time series to be sent through the FFT pipeline at the beginning of a scan. The minimum resolution of this time is 200 ns or the flash A/D sample period for the bandwidth in use, whichever is greater.

utc.stop — Sets the time to stop sending accumulated data to the spectral processor MassComp to the nearest integer second.

$ pulsar.period.N — The period of the pulsar to be observed in UTC seconds at the 'utc.start' time. This is used to set up synchronous accumulation in some pulsar modes.

pulsar.period.deriv.N — The derivative of the pulsar period at the 'utc.start' time in UTC seconds per UTC second. This value may be used to track a pulsar period in a long scan of synchronous averaging by dividing the scan into shorter integration intervals and updating the clock rate between sub-integrations. The sub-integration may need to be as short as 20 seconds depending on the magnitude of the derivative, and the update time between may be on the order of a couple of seconds.

$ dispersion.measure.N — Specifies the pulsar dispersion measure for spectral processor modes associated with dedispersion. The unit is parsecs/cm**3.

processor.mode.N — Specifies the configuration of the spectral processor when observing.mode = "Pulsar" and, hence, sets the context in which all other parameters are intrepreted.
"DedispTimeSamples": dedispersed intensity vs time output
"TimeSyncSpectra": spectra accumulated at selected phases of a pulse period for many pulses

"SigAvgDedisp": dedispersed intensity vs time folded for many pulse periods - synchronous signal averaging
"SpectraBurst": no averaging of spectra - data collected in short bursts whose average rate can be handled by the MassComp input
"VoltageTimeSamples": untransformed time samples - data collected in short bursts whose average rate can be handled by the MassComp input
"PowerTimeSamples": untransformed but squared and averaged time samples - data may be collected in short bursts whose average rate can be handled by the MassComp input
"PulsePhaseSpectra": spectra accumulated at selected phases of a pulse period - really just a hardware variation on "TimeSyncSpectra"
"SyncFreqTime": accumulation of spectra at adjacent and equal width time intervals over a full pulse period for many pulses

$ num.window.N — Sets the number of time windows in a pulse period during which independent spectra are to be accumulated

$ begin.window.N.M — Specifies the beginning of each of the 'num.window' time windows in pulse phase (0.0-1.0, no units). Since the time windows must be in increments of FFT cycles, the nearest available beginning phase will be selected.

$ end.window.N.M — Specifies the end of each of the 'num.window' time time windows in pulse phase (0.0-1.0, no units). Since the time windows must be in increments of FFT cycles, the nearest available ending phase will be selected.

Below is the mapping from Command language keywords to spectral processor setup fields. New keywords are marked with #, old keywords that have been changed to arrays or to different array indices are marked with $, and fields that will probably want to be under interactive control at the spectral processor are marked with an *. Spectral processor mode numbers are called out in S.P. Memo 31, appendix C.

/* Clock source and board setup */

synth_route	Spectrom.clock_source, #Spectrom.if_lo_source[2][8]
clock_source	Spectrom.clock_source
timing_gen	Normal, independent of keywords
ad_interface	Normal, independent of keywords
input_buffer	Normal, independent of keywords
window	Normal, independent of keywords
fft[11]	Setup.observ_mode, #Pulsar.processor_mode[2], usable FFT board configuration set by engineers and remembered by MassComp and rack controller
real_correct	Setup.observ_mode, #Pulsar.processor_mode[2]
square_cross	Setup.observ_mode, #Pulsar.processor_mode[2]
accum	Not used

```
/* Bandpass and intermod filters, sideband, and multiplier config. */

  bandwidth_num Setup.band_width[2], $Spectrom.num_spectr[2],
                $Spectrom.num_chan[2]
  sq_cross_mult #Spectrom.multiplier_mode
  filt_sideband $LO_tuning.ifs[2][8], #LO_tuning.if_sideband[2][8]

/* Attenuators */

  atten_setting[8] #Spectrom.atten or #Spectrom.atod_input_lev

/* Intermediate frequency LO and clock synthesizer settings */

  lo[8][2]       $LO_tuning.ifs[2][8]
  clock[2]       #Spectrom.clock_freq and #Spectrom.clock_freq_deriv or
                 Pulsar.pulsar_period[2] and #Pulsar.pulsar_period_deriv[2]

/* Accumulator, Cal, Sig/Ref.... switching and integration times */

  accum_mode     Setup.observ_mode, #Pulsar.processor_mode[2]
                        modes 1, 3, 4, 5, 6, 8, & 10: The following parameters
                        do not affect the accumulator addressing.  The
                        phase times are needed to make the total power
                        counters operate correctly.
* num_phases     Setup.observ_mode, #Pulsar.processor_mode[2]
                 modes 1, 3, 4, 5, 6, 8, & 10: one t, two T's (cal on and off)
                 modes 2 & 9: $Pulsar.num_window[2],
                              $Pulsar.begin_window[2][32],
                              $Pulsar.end_window[2][32].  Note that the keyword
                              'window' has already been chosen for what we've
                              been calling phase.  What we've called window uses
                              the keyword 'taper'.
                 mode 7: $Switch_ph.num_phase[2]
* t_mult         Setup.observ_mode, #Pulsar.processor_mode[2]
                 modes 1, 3, 4, 5, 6, 8, & 10: Set to 4 to prescale "sm_t_len"
                 modes 2 & 9: $Pulsar.num_window[2],
                              $Pulsar_begin_window[2][32],
                              $Pulsar.end_window[2][32].  Set as required to
                              make longest "t" phase length less than 2**16.
                              All "t's" scale together.
                 mode 7: $Switching.phase_time[2][8].  Set as required to make
                         longest "t" phase length less than 2**16.  All "t's"
                         scale together.
* sm_t_len[32]  Setup.observ_mode, #Pulsar.processor_mode[2],
                 Setup.band_width[2], $Spectrom.num_spectr[2],
                 $Spectrom.num_chan[2], $Contin_back.sample_time[2]
                 modes 1, 3, 4, 5, 6, 8, & 10: Set roughly to the number of
                              spectra in 0.5 seconds / t_mult.
                 modes 2 & 9: $Pulsar_begin_window[2][32],
                              $Pulsar.end_window[2][32].  Use as many phases as
                              required to set time windows in the pulse period.
                 mode 7: $Switching.phase_time[2][8], $Switching.blank_time[2].
                         Set one phase for longest period consistent with
                         required "T's."
```

```
* lg_t_len[32]  Setup.observ_mode, #Pulsar.processor_mode[2],
                Setup.band_width[2], $Spectrom.num_spectr[2],
                $Spectrom.num_chan[2], $Contin_back.sample_time[2]
                modes 1, 3, 4, 5, 6, 8, & 10: Set two phases to one t cycle
                modes 2 & 9: $Pulsar_begin_window[2][32],
                             $Pulsar.end_window[2][32].  If not enough "t"
                             phases, use as many phases as required to set time
                             windows in pulse period.  Use at least two "T"
                             phases to drive cal on/off signal for power
                             counter.
                mode 7: $Switching.phase_time[2][8], $Switching.blank_time[2].
                        Set for Cal on/off, Sig/Ref, and blanking times.
* decode[32]    Setup.observ_mode, #Pulsar.processor_mode[2]
                modes 1, 3, 4, 5, 6, 8, & 10: T(0) to cal off, T(1) to cal on
                   The cal switch drives the total power integrators - need to
                   be sure that cal signal is not actually firing
                modes 2 & 9: $Pulsar.num_window[2],
                             $Pulsar_begin_window[2][32],
                             $Pulsar.end_window[2][32].  Decode blanking t's
                             and T's into bits 7 and 15.  Decode active t
                             windows into address bits 0-4.  Decode active T
                             windows into address bits 8-12, and activate cal
                             bit on at least one T phase.  The address is for
                             the timing generator map on the accumulator board.
                mode 7: $Switch_ph.phase[2][8]  All t's decode to zero.  Decode
                        cal T to address bit 14, sig/ref T to bit 13, and
                        blanking T's to bit 15.
  tp_cal_cycles Setup.observ_mode, #Pulsar.processor_mode[2]
                modes 1, 3, 4, 5, 6, 8, & 10: Set to one
                modes 2 & 9: $Pulsar.num_window[2],
                             $Pulsar_begin_window[2][32],
                             $Pulsar.end_window[2][32].  Set as required to get
                             a total power integration between 1 and 2 seconds.
                mode 7: $Switch_ph.phase[2][8], $Switching.phase_time[2][8],
                        $Switching.blank_time[2], $Switch_ph.num_phase[2].
                        Set as required to get a total power integration
                        between 1 and 2 seconds.
  integ_time    Redundant to accumulator parameters, not used

/* Flash A/D offset control */

  offset[8]     Determined by examination of spectral channel zero or an
                average of the untransformed time samples (mode 6). Can be
                done manually or automatically by the spectral processor
                MassComp.

/* Window and output data shift selection */

  window_sel    #Spectrom.taper_select[2]
  data_shift    #Spectrom.max_pwr[2], #Spectrom.atod_input_lev[2][8]

/* Setup start and stop times */

  start_tick[2]   Data_cntl.utc_start, #Spectrom.taper_offset (allows start
                                       times offset by half of a taper window)
```

```
  start_date          " (inferred)
  start_offset[2]     "
  stop_tick[2]    Data_cntl.utc_stop
  stop_date           " (inferred)

/* Interference excision */

* select         #Rfi.excise[2][8]
* time_const[8] #Rfi.fast_time_const[2][8], #Rfi.slow_time_const[2][8]
* clip_level[8] #Rfi.clip_level[2][8]
* threshold[8]  #Rfi.threshold[2][8]

/* Window coefficients, this is for one rack only. */

  w_upper[256] #Spectrom.taper[4]
  w_lower[256]
  x_upper[256]
  x_lower[256]
  y_upper[256]
  y_lower[256]
  z_lower[256]
  z_upper[256]

/* Accumulator addressing information */

  alu_map_mod     Setup.observ_mode, #Pulsar.processor_mode[2]
                  #Spectrom.multiplier_mode
                  mode 3: Setup.band_width[2], $Contin_back.sample_time[2],
                          Pulsar.pulsar_period[2]
                          #Pulsar.pulsar_period_deriv[2]
                  modes 1, 5, 8, 10: Set independent of keywords
                  modes 2, 4, 6, 7, 9: This parameter not used
  frame_format    Setup.observ_mode, #Pulsar.processor_mode[2]
                  #Spectrom.multiplier_mode
                  modes 1, 3, & 8: Transmit all error data
                  modes 2, 5, 7, 9, & 10: Transmit "A upper" error word every
                                          M cycles
                  modes 4 & 6: Do not transmit any error words
  data_cycle_num  Setup.observ_mode, #Pulsar.processor_mode[2]
                  #Spectrom.multiplier_mode
                  mode 1: Convenient frame length
                  modes 2, 4, 5, 7, & 9: $Spectrom.num_spectr[2],
                                         $Spectrom.num_chan[2]
                  modes 3 & 10: $Spectrom.num_spectr[2], $Spectrom.num_chan[2],
                                $Contin_back.sample_time[2],
                                Pulsar.pulsar_period[2]
                                #Pulsar.pulsar_period_deriv[2]
                  modes 6 & 8: $Contin_back.sample_time[2],
                               $Switching.blank_time[2]
  error_count     Setup.observ_mode, #Pulsar.processor_mode[2]
                  #Spectrom.multiplier_mode
                  modes 1, 3, 4, 6, & 8: This parameter not used
                  modes 2, 5, 7, & 9: $Spectrom.num_spectr[2],
                                      $Spectrom.num_chan[2]
```

```
                    mode 10: $Spectrom.num_spectr[2], $Spectrom.num_chan[2]
                             $Contin_back.sample_time[2], Pulsar.pulsar_period[2]
                             #Pulsar.pulsar_period_deriv[2]
  time_res_count    Setup.observ_mode, #Pulsar.processor_mode[2]
                    #Spectrom.multiplier_mode
                    modes 1, 3, 5, & 10: Setup.band_width[2],
                                 $Spectrom.num_spectr[2], $Spectrom.num_chan[2],
                                 $Contin_back.sample_time[2]
                    modes 2 & 4: This parameter not used
                    modes 6 & 8: $Contin_back.sample_time[2]
                    modes 7 & 9: Set to 0xFFFF independent of any keywords unless
                                 buffer counter needs prescaling
  offset_count      Setup.observ_mode, #Pulsar.processor_mode[2]
                    #Spectrom.multiplier_mode
                    mode 1: Set to zero independent of keywords
                    modes 2 & 4: This parameter not used
                    modes 3 & 10: $Contin_back.sample_time[2],
                                  Pulsar.pulse_period[2]
                    modes 5, 6, 7, 8, & 9: Set to 0xFFFF independent of keywords

/* Timing generator address mapping information for only one rack */

  map[1024]       Setup.observ_mode, #Pulsar.processor_mode[2]
                  #Spectrom.multiplier_mode
                  mode 7: $Switch_ph.num_phase[2], $Switch_ph.phase[2][8]
                  modes 2 & 9: $Pulsar.num_window[2],
                               $Pulsar_begin_window[2][32],
                               $Pulsar.end_window[2][32]
                  modes 1, 3, 4, 5, 6, 8, & 10: This parameter array not used

/* Buffer counter count cycle lengths in the accumulator addressing scheme
   for only one rack */

  length[256][2] Setup.observ_mode, #Pulsar.processor_mode[2]
                  #Spectrom.multiplier_mode
                 modes 2 & 9: $Contin_back.sample_time[2], Setup.band_width[2],
                              $Pulsar.num_window[2],
                              $Pulsar_begin_window[2][32],
                              $Pulsar.end_window[2][32]
                 modes 3 & 7: $Contin_back.sample_time[2], Setup.band_width[2]
                 mode 10: $Contin_back.sample_time[2], Setup.band_width[2],
                          Pulsar.pulsar_period[2],
                          #Pulsar.pulsar_period_deriv[2]
                 modes 1, 4, 5, 6, & 8: This parameter array not used

/* Dedispersion map, for only one rack. */

* map_a[1024]   Setup.observ_mode, #Pulsar.processor_mode[2]
                #Spectrom.multiplier_mode
                modes 1 and 3: $Pulsar.dispersion_measure[2],
                               LO_tuning.observ_freq[2]
                modes 2, 4, 5, 6, 7, 8, 9, and 10: independent of keywords
```