Requirements for Multifrequency Observing with the GBT

Ronald J. Maddalena
October 23, 2002

DRAFT

I. Introduction

Up to now, the use of the GBT has mostly been concerned with observing at a single frequency with a single bandpass or spectral window. The observers who have tried to observe multiple spectral-line transitions have had to devise their own methods of configuring the hardware and processing the data.

The purpose of this document is it describe step-by-step how one can set up multifrequency observing. I hope to provide requirements to those implementing the telescope control systems so that observers will have sufficient tools to make the task of multifrequency observing easier than it has in the past.

A few general comments:

- Astronomers will ask for multifrequency continuum, pulsar, radar, VLB and spectral line observing. I have tried to be complete for only continuum and spectral-line observing.
- A few of the capabilities I describe (offset frequencies and multifrequency Doppler tracking) can be phased in over time.
- I have assumed all observing will be dual polarization (linear or circular), single-beam observing but what is discussed here eventually needs to be expanded to cover single and cross polarization observing as well as multi-beam observing.
- Throughout I use notations, variable names, and equations that are only meant to facilitate this discussion and are not meant to influence implementation. The code presented in the appendix is provided as a model for the developers.
- The reduction of multifrequency data will be discussed in a separate document.
- Many of the ideas presented come from the memos, e-mails, or conversations with F. Ghigo, D. Balser, A. Roshi, B. Garwood, J. McMullin, T. Minter and others.

The document starts off with a few observing scenarios that cover the kinds of multifrequency observing astronomers will expect the GBT to perform (§ II). I next describe what an observer needs to specify for multifrequency observing (§ III). Once that information is known, the control system can configure the hardware for a suite of observations (§ IV) and for a single observation (§ V). Section VI provides the information that needs to be passed from the control system to the analysis software for proper data reduction. I conclude (§ VII) with those items I believe still need some discussion before detailed design and implementation can begin.

II. Observing Scenarios

Almost all multifrequency observing can be covered by just a handful of scenarios. The reader should make note of the abbreviations I suggest for each mode since I will use them throughout this memo. The scenarios are:

- Single frequency observing in a single spectral window (SFSW): The observer will want to observe a single spectral line transition of a molecule or atom or a band centered at a single frequency for continuum observing. GBT observing up to now has been predominately of this type. An example of this would be observing galactic atomic hydrogen with a bandpass sufficiently wide to encompass all of the emission. Or, an 8.4 Ghz continuum map of M17.
- Multiple frequencies in a single spectral window (MFSW): The observer will want a single spectral-line bandpass to encompass two or more transitions from different atoms or molecules or different transitions from the same molecule. This is a spectral-line observing mode only. Two examples are: 1) Observing the 1665 and 1667 MHz lines of OH in a 2.5 MHz or wider bandpass; 2) Observing the 4.8 Ghz H_2CO and 5.3 Ghz CH_2NH with an 800 MHz bandpass.
- Multiple frequencies in multiple spectral windows. Three scenarios come to mind, each of which is just an extension or hybrid of the above modes:
    - Multiple frequency spectral-line observing with each spectral window covering a single transition or continuum observing at multiple frequency (MFMW-1). This is essentially an extension of SFSW observing and an example of which is observing the 1612 and 1720 MHz lines of OH, each in a 5 MHz bandpass. Or, continuum observing at 8.4 and 10 Ghz with a bandwidth of 200 MHz..
    - Multiple frequency observing with each spectral window covering multiple transitions (MFMW-2). Essentially an extension of MFSW observing and a spectral-line observing mode.
    - Multiple frequency observing with some spectral windows covering a single transition while others cover multiple transitions (MFMW-3). This is a spectral-line observing mode and is a combination of SFSW and MFSW observing..
- Single frequency in multiple spectral windows (SFMW). Observers will pick this kind of spectral-line observing for two common reasons.
    - when they know the rest frequency of the atom or molecule but don't know the velocity of the object along the line of sight. An example of this kind of observing would be an attempt to discover galaxies in the zone of avoidance from their HI emission.
    - when the expected line width is broader than the maximum bandwidth of a backend. Here, the observer is trying to synthesize a broad bandpass from narrow ones.

    In almost all cases, the spectral windows are placed in series with maybe some overlap in frequency coverage.

Other types of observing can be described by one of these scenarios. For example, a spectral-line survey of TMC-1 could be set up using either MFSW or MFMW-2.

Summary: Multi-frequency observing must be able to handle observing in the following modes:

- SFSW - Single frequency, single spectral window
- MFSW - Multiple frequencies, single spectral window
- MFMW - Multiple frequencies, multiple spectral windows of which there are three subtypes.
  - MFMW-1 - multiple SFSW
  - MFMW-2 - multiple MFMW
  - MFMW-3 - a hybrid of SFSW and MFSW.
- SFMW - Single frequency, multiple spectral window

Continuum, pulsar, radar, and VLB observing can only be accomplished with the SFSW and MFMW-1 scenarios.

III. User Inputs for Configuring the System and Observing

For decades other telescopes have provided the ability to observe in all of the above scenarios. Thus, we don't need to look far for models of how we should present to the observer the various options for configuring the hardware for multifrequency observing. For example, I have tried to base my suggestions on the philosophies behind the 12-m and 140-ft control system.

From the above scenarios, it is almost obvious that the observer needs to be asked for the number of required spectral windows (*numWindows*).

For each spectral window, most telescope systems require that the observer supplies a single rest frequency. Initially, I thought that we would have to ask the user for multiple rest frequencies in order to cover the MFSW and most of the MFMW scenarios. But, I believe that Aips++ can handle the requirements by allowing the user the ability to change the assigned rest frequwncy to each spectral window. Thus, I think we should pick the simplest of options and have the observer supply a single rest frequency for each spectral window. Thus, the observer needs to supply a vector of rest frequencies whose length is numWindows [e.g., *restFrequencies(numWindows)*] and whose units are MHz. It would be nice if the configuration tool could supply a list of common rest frequencies to reduce the number of times observers would have to type in frequencies.

Since Doppler tracking with the GBT's L.O. 1 can only be done correctly for a single rest frequency, one of the values in the vector has to be designated as the rest frequency used in the Doppler calculations. I suggest that this 'master' rest frequency be the first in the vector and that the user interfaces make it clear that this rest frequency is given special significance by the control software. Note that the system could Doppler track multiple spectral windows simultaneously by using L.O. 2 (see § IV and V) but I consider this a feature that we will want to add at some future time.

For scenario SFMW, the rest frequencies for all of the spectral windows should have the same value. But, how does one specify the offsets between the different spectral windows? For scenario MFSW, how does one specify how much a spectral line should be offset from the center of the spectral window? In previous control systems, the user typically fudges either the rest frequencies or the intermediate frequencies.

A. Rosh suggested a more elegant solution where the observer can specify an offset frequency for every spectral window. The offset is defined as the amount by which the frequency of the center of a bandpass is shifted in MHz and would have a default value of zero and units of MHz. There are three possibilities for a definition of offset frequency, with each definition having certain benefits:

- Offset in rest frequency. For example, in scenario MFSW, if the astronomer was observing the 4.8 GHz H_2CO and 5.3 Ghz CH_2NH with an 800 MHz , he or she could specify a rest frequency of 4.8 GHz and an offset of 250 MHz to center the bandpass between the two lines. However, SFMW observing doesn't fit well with this definition of offset frequency.
- Offset in intermediate frequency. In scenario SFMW, an HI observer could specify a rest frequency of 1420 MHz, a bandwidth of 200 MHz and offsets of 180, 360, 540, ... MHz for the various spectral windows. Unfortunately, observers may not know the sign of to use for the offset since we have a mixture of upper and lower-sideband mixers and multiple mixes in our receiver and I.F. systems.
- Offset in sky frequency (i.e., the Doppler-shifted rest frequency). Here, the SFMW observer could specify a proper placement of the various spectral windows without worrying about the sign of the offsets but the MFSW observer would have difficulty calculating a proper offset if the source has a high velocity since the separation between two lines will change as a function of source velocity for some definitions of Doppler shifts.

Since there is no clear choice here, I will assume for the rest of this discussion the last definition (offset in sky frequency) but what follows can be easily modified if we choose a different option. Since this capability can be fudged in other ways (e.g., specifying a rest frequency that is offset from the actual rest frequency), the capability of specifying an offset can be a later addition to the control software.

As will become obvious in the next section, in order for the configuration tool to configure the hardware for multifrequency observing the observers will need to specify a Doppler definition (*velDefinition*= radio, optical, or relativistic) and a range of velocities (*velMin* and *velMax* in km/s) that will cover all of the objects in the observer's source list. The observer will specify the *receiver* and *backend*. I will make the simplifying assumption the user will specify a single *bandwidth* (in MHz) for all spectral windows, though it would be easy to extend the following sections to handle different bandwidths in different windows. For spectral line work, the user will need to specify the number of channels (*numChannels*, assumed to be the same for all spectral windows) and, for the Spectrometer, the number of levels (*numLevels*) at which the samplers are to be run.

For observing with GO, the astronomer must specify *velDefinition*, a rest frame (LSR, solar system barycenter, galactocentric, etc.), *receiver*, *backend*, *bandwidth* (in MHz), source velocity (*velocity* in km/s), coordinate frame (J2000, B1950, Galactic, etc.), source position (in the natural units for the coordinate frame), epoch (if coordinate frame requires it) and time of observation (start time and date) or ASAP if the start time doesn't matter. Since the configuration tool knows many of the values that are needed for observing, it would be beneficial if GO could inherit these values from the configuration tool somehow.

Summary: To configure the hardware the user must be able to specify:

- *numWindows*
- *restFrequencies(numWindows)* of which the first is to be considered the master for L.O. 1 Doppler tracking
- *offsets(numWindows)* – possibly a later addition
- *velDefinition*
- *velMin* and *velMax*
- *receiver* and *backend*
- *bandwidth*
- *numChannels* and *numLevels* when appropriate.

For observing, the user must specify:

- *numWindows*
- *restFrequencies(numWindows)* of which the first is to be considered the master for L.O. 1 Doppler tracking
- *offsets(numWindows)* – possibly a later addition
- *velDefinition* and *restFrame*
- *velocity*
- *receiver* and *backend*
- *bandwidth*
- *xpos, ypos, coordMode, epoch*
- *startTime* or *ASAP*

In our current plans, the items in the first set of bullet will be assigned values as part of the configuration tool. Some of the items in the second list will be inherited from the configuration tool. All of the items in the second set would be assigned values as part of the observing tool (GO).

IV. Hardware Configuration

Once the control software has the information need to configure the hardware, how does one determine the filters needed to use in a receiver or piece of I.F. equipment, what are the values of synthesizers, etc? In particular, how does one assign values to parameters in the various hardware managers for the GBT? This section addresses how one can set up a legitimate signal

path through the GBT for a suite of observations of multiple sources (i.e., what should the configuration tool do). Section V will deal with what needs to be set before each observation of a source (i.e., by GO).

Many of the steps in configuring the hardware are documented elsewhere and won't be covered here. In particular, I will not cover how one uses the GBT cabling file to determine the routing between devices. Instead I will concentrate on those aspects of configuring the hardware that have not been documented elsewhere and are relevant for multifrequency observing.

A. Receiver R.F. Filters

Many of the GBT receivers have R.F. bandpass filters before the receiver's mixer. The filter for a suite of observations should not be changed between sources. Instead, the configuration software must calculate the minimum and maximum sky frequency that will cover the range of rest frequencies and source velocities. The R.F. filter that should be chosen must cover the range of sky frequencies augmented by the desired bandwidth.

As an aid to implementation, the appendix provides Tcl code for calculating sky frequency from rest frequency, velocity definition, and source velocity (**calculateSkyFreq**) as well as the minimum and maximum sky frequency from *restFrequencies, velMin, velMax, bandwidth, velDefinition,* and *offsets* (**skyFrequencyRange**). Note how the latter routines uses bandwidth to add extra coverage in sky frequency. Using the output from a similar routine, the configuration tool should look at the range of frequencies covered by the R.F. filters in a receiver and pick the filter with the narrowest width that covers the range of sky frequencies.

Note that the calculation for sky frequency range does not involve the rest frame since the configuration tool does not know the date or time of the observing or the source position. Thus it cannot correct the Doppler calculation for the Earth's motion in a chosen rest frame. The resulting range of frequencies could be off from its optimum values by the sky-frequency equivalent of a few hundred km/s. But, having non optimum values is probably not sufficient to produce a bad configuration since receiver filters do not have sharp edges.

B. L.O. 1 Frequency and Center I.F.

An accurate L.O. 1 frequency can only be determined at the time of observation since it depends upon source position and time of observation. Nevertheless, we need to calculate guesses for the L.O. 1 frequency in order to calculate an optimum center I.F. frequency for a range of observations.

Before we can calculate these quantities, the configuration tool must know the center frequency formula (CFF) for the chosen receiver. The CFF describes how one can calculate sky frequency from the L.O. 1 frequency and I.F. Center. By rearranging the CFF, one can calculate L.O. from I.F. and sky frequency or I.F. from sky and L.O. frequencies. The CFF must take into consideration any multipliers or offset mixers in the L.O. chain. Everyone writes CFF

differently but all representations are just algebraic permutations of the same equations. For this work I'll write the CFF in the form:

$$Sky\,Frequency = A \pm N * LO_1 \pm IF_{center}$$
$$IF_{center} = Sky\,Frequency - A \mp N * LO_1$$
$$LO_1 = \left(Sky\,Frequency \mp IF_{Center} - A\right) / N$$

As an example, Rcvr1_2 has a CFF of SkyFrequqncy = LO_1 - IF_{center}.

Each receiver has an preferred center I.F. (e.g., 3000 MHz for Rcvr1_2, 1080 MHz for PF1, etc.). The first step in calculating an optimum center I.F. is to chose an L.O. 1 such that the average of the maximum and minimum sky frequency is converted by the mixer to the preferred center I.F. Next, the configuration tool should calculate an I.F. from the sky frequency of the "master" rest frequency and the just-calculate L.O. 1 frequency. Since we are expecting a range of source velocities, the above calculations are done for velMin and velMax and the resulting center I.F. averaged together. The average center I.F. is then the optimum value for configuring the hardware. I have provided Tcl code to perform these calculations for Rcvr1-2 in the appendix (**optimumIFCenter**).

The configuration tool will need to know for later calculations a good guess of the LO1 frequency. To do so requires using the optimum center I.F., the master rest frequency, and probably the average of velMin and velMax (see **setPreliminaryLO1** in the appendix).

C. Receiver I.F. Filters and I.F. Rack Filters

Once the optimum center I.F is known, we can now determine which I.F. filters to use in the receiver (if any exist there) and in the I.F. rack. The filter to pick must be the one with the narrowest width that covers the range in observing I.F. The range in observing I.F. is calculated from the CFF, the minimum and maximum sky frequencies, augmented by the backend bandwidth, and the L.O. 1 frequency. The calculation is repeated for velMin and velMax so as to find the minimum and maximum I.F. that will cover all of the proposed observations. The **ifRange** procedure in the appendix is an example of this calculation.

E. L.O. 2 Frequency

For some very limited continuum SFSW observations, the setting of the filters in the I.F. rack will be the last stage in configuring observing. In most cases, multifrequency observing will require configuring the L.O. 2 synthesizers in the Converter Rack. Although one needs to examine the cabling file to determine which Converter Rack modules will be needed for an experiment, the number of modules will always be equal to *2 x numWindows* and the number of L.O. synthesizers will be equal to *numWindows*. Polarization pairs will share a common L.O. 2 synthesizer (e.g., Synthesizer 2 feeds modules 2 and 6 which should be the two polarizations for

a single spectral window). The cabling file will also determine the routing of signals from the converter rack to the analog filter rack or various backends.

In addition to signal routing, multifrequency observing requires the setting of the synthesizers for L.O. 2. The second I.F. frequency after the mixer in the converter rack depends upon the backend, or in the case of the Spectrometer, the chosen bandwidth. Once the second I.F. is known, for each rest frequency, the configuration tool must calculate the sky frequency using the average of velMin and velMax, and then calculate the first I.F. The L.O. 2 frequency is a rather simple calculation that uses the first I.F. and knowledge of the backend and bandwidth (see **calcLO2** in the appendix.)

F. Other Devices

Essentially, the only thing that remains to be configured is backend specific. For example:

- DCR: Must use the cabling file to determine which samplers are going to be used. The number of samplers will be 2 x *numWindows*. For most multifrequency work, the Analog Filter Rack (AFR) will be used and the number of AFR modules will be 2 x *numWindow*. The modules that will be used in the AFR depend upon the bandwidth and which Converter Rack Modules are used. The filters in the AFR are also determined by the desired bandwidth.

- Spectral Processor: The samplers to use must be in sequence and balanced between the two racks. These restrictions essentially dictates the Converter Rack modules that must be employed. The number of samplers will always be 2 x *numWindows*

- Spectrometer: 2 x *numWindows* modules in the Analog Filter Rack (AFR) will be used. The modules that will be used depend upon the bandwidth and which Converter Rack Modules are used. The filters settings in the AFR are determined by the desired bandwidth. The samplers that will be used in the Spectrometer depend upon the cabling file but the number of samplers will always be 2 x *numWindows*.

  The Spectrometer also has the characteristic that the same scientific goals cane sometime be accomplished with one of multiple configurations. The *findSpectrometerConfig* routine in the appendix will determine the Spectrometer mode strings and control parameters for a user's choice of Spectrometer bandwidth, number of levels, and number of samplers (= 2 * *numWindows*). If more than one mode is possible, then it might be the cabling to the Spectrometer that will dictate which mode should be used. If, after eliminating modes which are illegal because of cabling, there still exists a choice of configurations then it is probably best if the configuration tool were to select the configuration that uses the fewest Spectrometer banks.

Example: Let us consider an example where the astronomer wants to observe rest frequencies of 1420, 1612, 1665, 1667, and 1720 MHz with a bandwidth of 12.5 MHz (i.e., Rcvr1_2 and the Spectrometer will be used). Since the 16655 and 1667 lines fall into the same bandwidth, the observer will require four spectral windows with the third window offset by 1 MHz. The list of objects will have velocities that extend from +200 to -1000 km/s with velDefinition = radio. The user selects 1420 MHz as the "master" for Doppler tracking, 9-level sampling with 8192 channels per spectrum.

% skyFrequencyRange {1420 1612 1665 1720} -1000 200 12.5 radio {0 0 1 0}
1412.8026781 1731.98730163

[This indicates we will need to use the 1.1 - 1.8 GHz filter in the receiver]

% *optimumIFCenter {1420 1612 1665 1720} Rcvr1_2 -1000 200 radio {0 0 1 0}*
3150.20013843
% *setPreliminaryLO1 {1420 1612 1665 1720} Rcvr1_2 -1000 200 radio {0 0 1 0} 3150.20013843*
4572.09478223

[We now can specify the center I.F. and have an estimate of a L.O. frequency that will be used when observing commences.]

% *ifRange {1420 1612 1665 1720} Rcvr1_2 -1000 200 12.5 radio {0 0 1 0} 4572.09478223*
2840.1074806 3159.29210413

[The appropriate filter to use in the I.F. rack is the one that covers 2840 to 3160 MHz.]

% *calcLO2 {1420 1612 1665 1720} Rcvr1_2 -1000 200 12.5 Spectrometer radio {0 0 1 0} 4572.09478223*
13182.7001384 12990.4439612 12936.3732457 12882.2998616

[We now have the three values for the L.O. 2 synthesizers in the converter rack.]

% *findSpectrometerConfig 12.5 9 4 8192*
2 Bank(s) -> 1N2----12-9
(A1 B1 - 2 sampler(s) per bank)

1 Bank(s) -> 2N4----12-9
(A2 - 4 sampler(s) per bank)

Two Spectrometer configurations are possible, and, if the cabling is appropriate, the second, single-bank configuration is preferred.

V. Observing

The previous configuration step sets up the system for observing a suite of objects. Both GO and the control system need to be modified so as to properly change the set up of the system to accommodate the observation of individual sources. Fortunately, most of what we already have gone through will be directly applicable to what the control system must do on a source by source, scan by scan basis.

- The control system will not need to change filters in either the receivers or I.F. rack as the observer moves from one source to the next. The configuration step insures that the chosen filters will handle all sources and, therefore, there is no equivalent of **skyFrequencyRange** or **ifRange** for observing.
- The **calculateSkyFreq** algorithm presented in the appendix is inadequate. For observing, we need this calculations to correctly calculate the systemic velocity in the user's chosen velocity frame. The algorithm must calculate the systemic velocity from the time of the observation, coordinate system, and source position. The full algorithm is well described in memos by Joe Brandt and Rick Fisher and is already in use for calculating L.O. 1 frequencies.
- By using the modified **calculateSkyFreq**, the control system should calculate the optimum I.F. center frequency. This step should probably be done by GO at the start of every observation. Note that the version of **optimumIFCenter** in the appendix requires a *velMin* and *velMax*. For observing, both of these values will be the same and will be the user-specified velocity of the source that is about to be observed.
- The **setPreliminaryLO1** routine in the appendix is an exact functional copy of the code Joe Brandt must be using in the control system for setting L.O. 1 frequencies. Thus, this step is already being handled.
- The **calcLO2** routine should be called at least at the start of an observation for a source. Again, this routine asks for a *velMin* and *velMax* but the value inserted for observing for both these arguments will be the source velocity.

It is interesting to note that if we wanted to perform correct Doppler tracking for all spectral windows, not just for the master, the **calcLO2** routine can be used every time the value of L.O. 1 changes in order to set the L.O. 2 synthesizers properly. Thus, we now have a conceptual and computational outline of how we can accomplish proper multifrequency Doppler tracking.

It is also important to note that the configuration tool must pass to GO the number of spectral windows, the list of rest frequencies and offset frequencies, the receiver, backend, etc. Many of these values GO can currently pick up directly – the configuration tool sets the values for most items into manager parameters and GO then can obtain the values from the manager. Unfortunately, we currently don't have any parameters that correspond to *numWindows*, *restFrequencies*, or *offsets*. It is unclear whether we need new parameters to pass this information to GO or whether there are other mechanisms we can use to pass the information. If we decide on parameters, then it is then unclear to which manager these parameters belong.

Summary: For observing, the control and GO software must be modified to handle the following:

- An optimum center I.F. should be calculated at the start of every scan and placed into the L.O. 1 manager.
- L.O. 2 frequencies should be recalculated at the start of every scan and placed into the Converter Rack manager.
- Proper Doppler tracking for all spectral windows can be accomplished by updating L.O. 2 frequencies whenever L.O. 1 is updated. We may want to hold off implementing this feature.
- *numWindows*, *restFrequencies*, and *offsets* need to be passed from the configuration tool to either GO or an appropriate manager.

VI. FITS Files

Almost all of the information needed to describe how the system has been set up for multifrequency observing already exists in our current set of FITS files. The information needed by a data analysis system that doesn't already exist in the FITS files is the *restFrequencies* vector. B. Garwood's opinion is that if GO knows the values of this vector, then they best belong in the GO FITS files. If we decide to store these values in a manager's parameter, then that manager must write that information to its FITS files. There's no need for the *offset* vector to be included in the FITS files since the information already in the I.F. Manager's FITS files contain sufficient information for any analysis system.

In any case, B. Garwood suggests that we retain the current RESTFREQ keyword in our FITS files and add a RESTFRQS vector of keyword to the appropriate FITS file.

Summary: A vector keyword, called RESTFQS, must be added to a FITS file, preferably the file created by GO.

VII. Items for Discussion

There are a few outstanding items in the above discussion.

- How does GO inherit the values from the configuration tool?
- Is offset in sky frequency what observers will find most useful?
- Is the averaging of *velMin* and *velMax* in some of the calculations correct?
- Does GO or a manager write the new RESTFQS keyword to its FITS file?
- Who knows enough to modify this discussion to cover pulsar, VLB, and radar observing?
- Should we extend this document to cover multi-beam and single/cross-polarization observing?
- When should we think about phasing in offset frequencies and L.O. 2 Doppler tracking?

- Should *offset* vector be included in the FITS files?
- Does GO or a manager take care of setting center I.F. and L.O. 2 frequencies at the start of an observation?

Appendix

The following TCL code illustrates how one can calculate the various quantities described in the above text. I have tried the code on a only few cases, not enough to convince me that it is free of errors. Only minimum error checking is currently being done.

```
#-------------------------------------------------
# calculateSkyFreq
#
# Calculates sky frequency
#
# Arguments:
#
#          restFrequency – rest frequency in MHz
#   VelDefinition – velocity definition.  Either radio, optical, or relativistic
#   Velocity – source velocity
#
# Returns:
#          Sky frequency in MHz
#
proc calculateSkyFreq {restfrequency velDefinition velocity} {

   set z [expr $velocity / 299792.5]

   switch $velDefinition {
      optical {
         return [expr $restfrequency/($z + 1.)]
      }
      radio {
         return [expr $restfrequency*(1. - $z)]
      }
      relativistic {
         return [expr $restfrequency * sqrt( (1. - $z) / (1. + $z) )]
      }
      default {
         return -code error "Bad velocity definition: $velDefinition"
      }
   }
}

#-------------------------------------------------
# skyFrequencies
#
# Returns list of sky frequwencies for a list of rest frequencies and a velocity
#
# Arguments:
#
```

```
#           restFrequencies – list of rest frequency in MHz
#           velocity - Velocity of source
#   VelDefinition – velocity definition.  Either radio, optical, or relativistic
#           offsets – Offset frequencies in sky-frequency space
#
# Returns:
#           list of min and max sky frequencies in MHz
#
proc skyFrequencies { restFrequencies velocity velDefinition offsets} {

   set numWindows [llength $restFrequencies]
   if {$numWindows != [llength $offsets]} {
      return -code error "Number of offset frequencies and rest frequencies don't match"
   }

   # For each rest frequency, calculate the sky frequency and add to list of sky frequencies
   for {set idx 0} {$idx < $numWindows} {incr idx} {
      set restf [lindex $restFrequencies $idx]
      set off [lindex $offsets $idx]
      lappend skyFrequencies [expr [calculateSkyFreq $restf $velDefinition $velocity] + $off]
   }

   # Sort the list of sky frequencies.
   return $skyFrequencies
}

#-------------------------------------------------
# skyFrequencyRange
#
# Calculates the range sky frequency for a suite of observations
#
# Arguments:
#
#           restFrequencies – list of rest frequency in MHz
#           velMin/max - range of source velocities
#           bandwidth - backend bandwidth
#   VelDefinition – velocity definition.  Either radio, optical, or relativistic
#           offsets – Offset frequencies in sky-frequency space
#
# Returns:
#           list of min and max sky frequencies in MHz
#
proc skyFrequencyRange { restFrequencies velMin velMax bandwidth velDefinition offsets} {

   # Create a list of sky frequencies at both velMin and Max.
   set  skyFrequencyList [skyFrequencies $restFrequencies $velMin $velDefinition $offsets]
   eval lappend skyFrequencyList [skyFrequencies $restFrequencies $velMax $velDefinition $offsets]

   # Sort the list of sky frequencies.  The first and last elemnnts in the list should be the
   # min and max sky velocities
   set skyFrequencyList [lsort $skyFrequencyList]
   set maxSkyFreq [lindex $skyFrequencyList end]
   set minSkyFreq [lindex $skyFrequencyList 0]

   # Adjust range of frequencies by half the bandwidth
```

```
   return "[expr $minSkyFreq - $bandwidth/2.] [expr $maxSkyFreq + $bandwidth/2.]"

}

#---------------------------------------------
# skyFreqFromLO1IF,  lo1FreqFromSkyIF, ifFreqFromSkyLO1
#
# Routines that return either sky frequency, I.F, or L.O. frequencies from
# two of these three values
#
# Arguments:
#
#          receiver - Name of the receiver
#   Two of the following:  sky - sky fequency
#                          lo1 - lo1 frequency
#                          ifCenter - I.F. frequency
#
# Returns:
#          The one tem of the three not passed as an argument
#
proc skyFreqFromLO1IF { receiver lo1 ifCenter } {

   switch $receiver {
      Rcvr1_2 {
         return [expr $lo1 - $ifCenter]
      }
   }
}

proc lo1FreqFromSkyIF { receiver sky ifCenter } {

   switch $receiver {
      Rcvr1_2 {
         return [expr $sky + $ifCenter]
      }
   }
}

proc ifFreqFromSkyLO1 { receiver sky lo1 } {

   switch $receiver {
      Rcvr1_2 {
         return [expr $lo1 - $sky]
      }
   }
}

#---------------------------------------------
# preferredIFCenter
#
# Returns a receiver's preffered I.F.
#
# Arguments:
#
#          receiver - Name of the receiver
```

```
#
# Returns:
#           Value of the preferred I.F. in MHz
#
proc preferredIFCenter { receiver } {
   switch $receiver {
      Rcvr1_2 {
         return 3000
      }
   }
}


#-----------------------------------------------
# optimumIFCenter
#
# Returns the optimum center IF.  Assumes the same bandwidth for all spectral windows.
#
# Arguments:
#
#           restFrequencies – list of rest frequency in MHz
#   receiver - Name of the receiver
#           velMin/max - range of source velocities
#   velDefinition – velocity definition.  Either radio, optical, or relativistic
#           offsets – Offset frequencies in sky-frequency space
#
# Returns:
#           list of min and max sky frequencies in MHz
#
proc optimumIFCenter { restFrequencies receiver velMin velMax velDefinition offsets} {

   # Calculate the sky frequwncies for the master rest frequwncy at velMin
   set masterRestFrequency [lindex $restFrequencies 0]
   set masterOffset [lindex $offsets 0]
   set skyFreqForRestFreqVelMin [expr [calculateSkyFreq $masterRestFrequency $velDefinition $velMin] + $masterOffset]

   # Find the average sky frequency for velMin
   set skyFrequencyListVelMin [lsort [skyFrequencies $restFrequencies $velMin $velDefinition $offsets]]
   set avrgSkyFreqVelMin [expr ([lindex $skyFrequencyListVelMin 0] + [lindex $skyFrequencyListVelMin end])/2.]

   # Calculate LO1 for the receiver's preferred center IF and the average sky frequency at velMin
   set loCenterVelMin [lo1FreqFromSkyIF $receiver $avrgSkyFreqVelMin [preferredIFCenter $receiver]]
   set ifSkyVelMin [ifFreqFromSkyLO1 $receiver $skyFreqForRestFreqVelMin $loCenterVelMin]

   # repeat for velMax
   set skyFreqForRestFreqVelMax [expr [calculateSkyFreq $masterRestFrequency $velDefinition $velMax] + $masterOffset]
   set skyFrequencyListVelMax [lsort [skyFrequencies $restFrequencies $velMax $velDefinition $offsets]]
   set avrgSkyFreqVelMax [expr ([lindex $skyFrequencyListVelMax 0] + [lindex $skyFrequencyListVelMax end])/2.]
   set loCenterVelMax [lo1FreqFromSkyIF $receiver $avrgSkyFreqVelMax [preferredIFCenter $receiver]]
   set ifSkyVelMax [ifFreqFromSkyLO1 $receiver $skyFreqForRestFreqVelMax $loCenterVelMax]

   # Average the two center If's
   return [expr ($ifSkyVelMin+$ifSkyVelMax)/2.]
}
```

```
#-----------------------------------------------
# setPreliminaryLO1
#
# Returns an adequate guess for a typical LO1.
#
# Arguments:
#
#          restFrequencies – list of rest frequency in MHz
#   receiver - Name of the receiver
#          velMin/max - range of source velocities
#   velDefinition – velocity definition.  Either radio, optical, or relativistic
#          offsets – Offset frequencies in sky-frequency space
#   ifCenter - Center I.F.
#
# Returns:
#          list of min and max sky frequencies in MHz
#
proc setPreliminaryLO1 { restFrequencies receiver velMin velMax velDefinition offsets ifCenter} {

   set avrgVel [expr ($velMin + $velMax)/2.]

   # Calculate the sky frequwncies for the master rest frequwncy at velMin
   set masterRestFrequency [lindex $restFrequencies 0]
   set masterOffset [lindex $offsets 0]
   set skyFreqForMaster [expr [calculateSkyFreq $masterRestFrequency $velDefinition $avrgVel] + $masterOffset]

   return [lo1FreqFromSkyIF $receiver $skyFreqForMaster $ifCenter]
}

#-----------------------------------------------
# ifRange
#
# Returns the range in I.F. that will cover an observation.
#
# Arguments:
#
#          restFrequencies – list of rest frequency in MHz
#   receiver - Name of the receiver
#          velMin/max - range of source velocities
#   bandwidth - backend bandwidth
#   velDefinition – velocity definition.  Either radio, optical, or relativistic
#          offsets – Offset frequencies in sky-frequency space
#   lo1 - preliminary L.O. 1
#
# Returns:
#          Min and max I.F in MHz
#
proc ifRange { restFrequencies receiver velMin velMax bandwidth velDefinition offsets lo1} {

   set  skyFrequencyList [lsort [skyFrequencies $restFrequencies $velMin $velDefinition $offsets]]
   set if1 [ifFreqFromSkyLO1 $receiver [expr [lindex $skyFrequencyList end]  + $bandwidth/2.] $lo1]
   set if2 [ifFreqFromSkyLO1 $receiver [expr [lindex $skyFrequencyList 0]  - $bandwidth/2.] $lo1]

   set  skyFrequencyList [lsort [skyFrequencies $restFrequencies $velMax $velDefinition $offsets]]
   set if3 [ifFreqFromSkyLO1 $receiver [expr [lindex $skyFrequencyList end] + $bandwidth/2.] $lo1]
```

```
    set if4 [ifFreqFromSkyLO1 $receiver [expr [lindex $skyFrequencyList 0] - $bandwidth/2.] $lo1]

    if {$if1 > $if2} {
        set ifmax [expr $if1>$if3?$if1:$if3]
        set ifmin [expr $if2<$if4?$if2:$if4]
    } else {
        set ifmax [expr $if2>$if4?$if2:$if4]
        set ifmin [expr $if1<$if3?$if1:$if3]
    }

    # Adjust range of frequencies by half the bandwidth
    return "$ifmin $ifmax"
}

#-----------------------------------------------
# calcLO2
#
# Returns a list of L.O. 2 frequencies
#
# Arguments:
#
#           restFrequencies – list of rest frequency in MHz
#   receiver - Name of the receiver
#           velMin/max - range of source velocities
#   bandwidth - backend bandwidth
#   backend - name of backend
#   velDefinition – velocity definition.  Either radio, optical, or relativistic
#           offsets – Offset frequencies in sky-frequency space
#   lo1 - preliminary L.O. 1
#
# Returns:
#           List of L.O. 2 synthesizers settings
#
proc calcLO2 { restFrequencies receiver velMin velMax bandwidth backend velDefinition offsets lo1} {

    # Calculate list of sky frequqncies for the average of velMin and velMax and step through each.
    foreach skyFrequency [skyFrequencies $restFrequencies [expr ($velMin+$velMax)/2.] $velDefinition $offsets] {

        # First I.F. frequqency
        set ifForLO2 [ifFreqFromSkyLO1 $receiver $skyFrequency $lo1]

        switch $backend {
            DCR -
            Spectrometer {
                switch $bandwidth {
                    12.5 {
                        lappend rtrn [expr 10500 + $ifForLO2 + 32.5 - 500]
                    }
                    50 {
                        lappend rtrn  [expr 10500 + $ifForLO2 + 75 - 500]
                    }
                    200 {
                        lappend rtrn  [expr 10500 + $ifForLO2 + 900]
                    }
                    800 {
```

```
                lappend rtrn  [expr 10500 + $ifForLO2 + 1200]
              }
            }
          }
          SpectralProcessor {
            lappend rtrn  [expr 10500 + $ifForLO2 - 250]
          }
          VLBI {
            # add appropriatre code here
          }
          BCPM {
            # add appropriatre code here
          }
        }
      }
      return $rtrn
}

#-----------------------------------------------
# findSpectrometerConfig
#
# Determines the Spectrometer configurations that will work for a specific
# set of observing parameters
#
# Arguments:
#
#          bandwidth -- Bandwidth {800, 200, 50, or 12.5)
#   numLevels -- 3 or 9, ignored in bandwidth is 800 or 200
#   numSamps -- number of samplers
#   numChannels -- number of channels per spectra
#
# Returns:
#          List of mode strings and M&C paaramter values that can be used to set up the
#   Spectrometer.  For some combinations, no legitimate mode exists.  For others,
#   more than one will be returned.
#
proc findSpectrometerConfig { bandwidth numLevels numSamps numChannels } {

   # Need to calculate Number of quads * number of banks.  This depends upon
   # bandwidth, sampling level, and number of samplers
   set p 1
   set s 1
   switch $bandwidth {
      800 {
         set s 16
      }
      200 {
         set s 4
      }
      12.5 -
      50 {
         switch $numLevels {
            3 {
               set p 1
            }
```

```
            9 {
               set p 4
            }
         }
      }
   }
}
set quadsXBanks [expr 4 * $numChannels * $numSamps * $s * $p / 262144]
if {$quadsXBanks <= 0 || $quadsXBanks > 4} {
   return -code error "Configuration not possible"
}

# Determine whether this is a W(ide) or N(arrow) mode.  Determine the trailing part of the
# mode string
switch $bandwidth {
   200 -
   800 {
      set char W
      set bw_lvl "----$bandwidth"
   }
   12.5 -
   50 {
      set char N
      set bw [string range $bandwidth 0 1]
      switch $numLevels {
         3 {
            set bw_lvl "----$bw-3"
         }
         9 {
            set bw_lvl "----$bw-9"
         }
      }
   }
}

# Determine the first and third character for the current mode.  Sometimes
# multiple modes provide the same configuration.
set modes ""
foreach numQuadsPerBank {1 2 4} {
   set numBanks [expr $quadsXBanks / $numQuadsPerBank]
   if {$numBanks > 0} {
      set numSampsPerBank [expr $numSamps / $numBanks]
      if {$numSampsPerBank < 16 && $numSampsPerBank > 0} {

         # Find illegal modes
         switch $bandwidth {
            50 -
            12.5 {
               if {$numLevels == 9 && $numSampsPerBank == 8 && $numQuadsPerBank > 1} {
                  break
               }
            }
            200 {
               if {$numSampsPerBank == 8 || ($numSampsPerBank == 4 && $numQuadsPerBank > 1) } {
                  break
               }
```

```
            }
            800 {
               if {$numSampsPerBank > 2 || ($numSampsPerBank == 2 && $numQuadsPerBank > 1) } {
                  break
               }
            }
         }

         # All has past muster so we can create the configuration mode string.
         append modes "$numBanks Bank(s) -> $numQuadsPerBank$char$numSampsPerBank$bw_lvl\n "

         # Create the M&C parameter configuration designation
         set configParms {A B C D}
         append modes "      ("
         for {set ibank 0} {$ibank < $numBanks} {incr ibank} {
            append modes "[lindex $configParms $ibank]$numQuadsPerBank "
         }
         append modes "- $numSampsPerBank sampler(s) per bank)\n\n"
      }
   }
}

return $modes
}
```