# InstrumentModels for Single Dish Calibration

A Taxonomy of Astronomical Instrumentation

1992 August 13

**R. O. Redman and R. Payne**

# 1 Classifying Astronomical Instrumentation for Calibration

An enormous variety of devices have been built to collect astronomical data. To calibrate the data from one of these devices AIPS++ must associate with it at least one InstrumentModel describing how the data was converted from radiation into numbers. Since the only common themes linking all of these machines are the radiation entering from the sky and the numbers exitting to the disk, it is with these that we should start when designing classes of InstrumentModels.

The electromagnetic radiation field from the sky may be characterized for most purposes by the intensity

$$I(\nu, \vec{n}, t, P) \tag{1}$$

where n^ is a unit vector pointing towards a direction on the celestial sphere (cap Omega) and $P$ is a discrete index to the four Stokes polarization states. Note that n^ is actually a two-dimensional quantity, which is most often designated by right ascension and declination. The independent variables in expression (1) will be refered to as the radiative coordinates. This document will attempt to classify instruments which measure the intensity at a set of points in radiative coordinate space. It is believed that the same collection of InstrumentModels should be applicable to an even wider selection of instruments; this will be discussed in a later memo.

For virtually all devices, the output of each channel will be separable in the independent radiative variables frequency, time, direction, and polarization to a good approximation. Thus the signal output from each channel N will be a (probably nonlinear) function of the flux into the channel which in turn may be written as an integral of the detector response over I:

$$\begin{aligned} S_N &= G(F_N) \\ F_N &= \sum_P W_{NP} \int_{t_0}^{t_1} dt \iint_\Omega d\omega\, b(\vec{n}) \int_0^\infty d\nu\, p_N(\nu)\, I(\nu, \vec{n}, t, P) \end{aligned} \tag{2}$$

where p(nu) is the sensitivity profile of the channel in frequency, b(n^) is the beamshape, and W_NP is the weighting of each polarization state in the final signal. For many instruments, the signal S_N is directly digitized. We may classify these devices as "first order" systems, because each output number will come from a single detector element. First order systems may be distinguished from second order systems which combine the signals from two detectors to produce their numerical output. Examples of second order systems are interferometers and some forms of polarimeters. In principle, third order (closure phase) and fourth order (closure amplitude) devices could exist as well, but in practice the higher order quantities are usually generated in software from the second order numbers.

For second order detectors it will often be useful to describe the correlator quite seperately from the individual detectors, giving rise to distinct ReceptorModel and CorrelatorModel objects. This distinction is rarely useful for first order systems where the detector often digitizes its own signal or immediately passes its signal to a backend which will do the job for it. Since interferometers and polarimeters lie outside the authors' experience, they will not be discussed further in this memo which will concentrate on first order systems.

An instrument will normally output its data as a set of arrays of samples spaced along one or more of the continuous coordinate axes, with the other radiation properties being held constant for all of the samples in each array. The entire set of arrays constitutes one "Yeg" from the device, but for our immediate purposes we will concentrate on a TelescopeComponent describing a single array within the Yeg, building more complex devices from aggregations of simple components. The cleanest distinction between devices is the dimensionality of this array. Photometric time series and spectra are examples of one dimensional arrays. Electronic cameras such as the SCUBA submillimeter bolometer array or a 2048x2048 CCD are examples of two dimensional arrays. Each of these devices needs TelescopeComponents to describe how to calibrate the intensity and the position of each array element along its coordinate axes. The non-varying properties for each array should be labelled in the final calibrated Yeg; it is the responsibility of the coordinating TelescopeModel to supply these values if they are not already part of the raw Yeg.

Quite largely, the calibration of a Yeg using a TelescopeModel can be described without any detailed knowledge of the contents of the Yeg. This clean seperation of Yeg from TelescopeModel is broken in one critical aspect. The model chosen to represent the data, an array of numbers representing the flux in a set of channels, does not carry any information about the radiation properties of each channel. Calibration involves two distinct processes: conversion of the raw signal into true estimates of the intensity, and the addition of information (i.e. a TelescopeModel) describing the radiation properties of each channel. In principle, this TelescopeModel can be added nondestructively to the Yeg. It will be used in the presentation of the contents of the Yeg, but will act only indirectly on the data itself. By contrast, the whole point of the flux calibration is to modify the data itself so that it represents the intensity of radiation as accurately as possible. Rather than overwrite the raw data (**VERY** dangerous!) a new Yeg should be generated to hold the calibrated data plus the TelescopeModel. Since not all observations require an intensity calibration, the new Yeg should be generated by the system before the intensity calibration is attempted.

With these considerations in mind, a generic InstrumentModel is presented in Figure 1. Each of the qualifiers attached to the GenericInstrument corresponds to one of the radiation coordinates and selects a coordinate subcomponent which describes the behaviour of the instrument in that coordinate. For instance, the frequency qualifier should select a suitably parameterized sensitivity profile for each channel; suitable parameters would probably be the center frequency and channel width in this case. Calibration of the instrument would consist of setting values for all of these parameters. It is useful to note that the GenericInstrument, although it was motivated by Equation (2), is more general than the equation, and with suitable definitions for the coordinate subcomponents can handle a very wide variety of instrumentation. Understanding the structure of a generic instrument also goes a long ways towards defining the contents of a Yeg for single dish astronomy; this is a large topic which will be dealt with in a later memo.

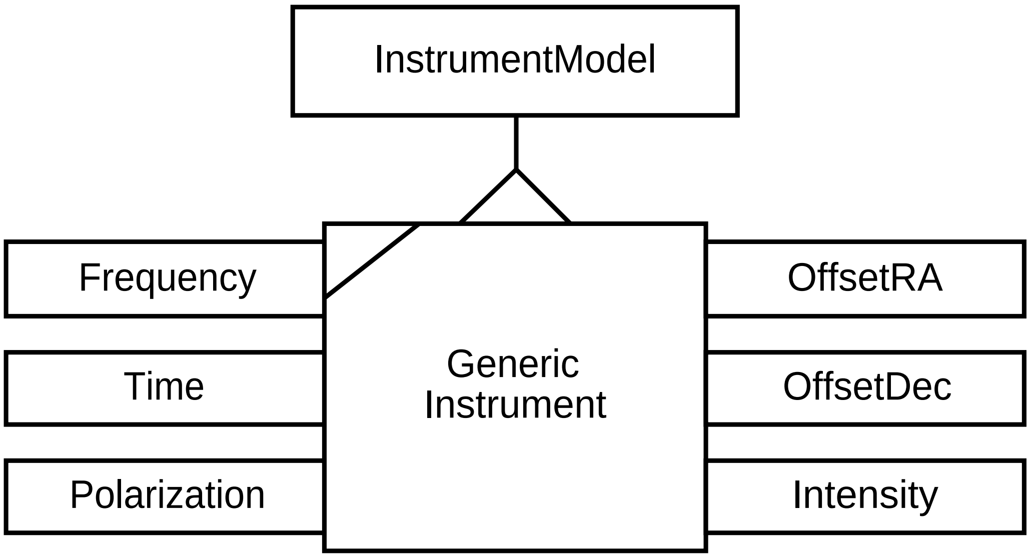


**Figure 1** — The Structure of a Generic Instrument

The following sections will consider the instrument components needed to build a detailed InstrumentModel. Rather than attempt a comprehensive description of all possible models, each section will offer a representative collection of components which should handle most of the important cases and which can be easily supplemented as new cases arise. Also, the focus will be on the parameterization of the coordinate subcomponents, rather than on their functional representation in radiation coordinates. The functions are likely to be very simple in the general cases and only develop an interesting complexity for particular instances, which may safely be left to later developers. Never-the-less, it should always be born in mind that a "center frequency" or a "coordinate offset" actually refer to parameters in a sensitivity profile and a beam shape model respectively.

# 2 Mathematical Classes to Describe Instruments

To describe a coordinate axis for one of the arrays, it is necessary to provide a function which accepts a channel number and returns an axis value. The most direct means of doing this in a TelescopeComponent is to provide the component with an operation() which takes a channel number (a two-dimensional channel number if the array is two-dimensional) and returns the value of the axis at that channel. Figure 2 describes a family of TelescopeComponents called FuncOfChan which would provide the commonly needed functional forms and can be extended easily to include new functions. Only the one-dimensional family is shown in detail in the figure. The two-dimensional analogues are easy to imagine, and it would be extremely easy to add a third dimension to the scheme if it should prove necessary.

An important facility not indicated in Figure 2 is an inverse function mapping the function back into channel number. It is easy to see how this might be done for the simpler FuncOfChan classes, although issues of existance and uniqueness are nontrivial even here. It is not at all easy to see how to invert a ComplexFuncOfChan, and it would probably be necessary to invert two FuncOfChan2D's simultaneously to find a unique solution. This issue is too important to ignore, but too complex to address properly in a concept paper such as this. Further work will be needed to decide when inverse functions are necessary and how to provide them in the more difficult cases.

The most common of the FuncOfChan will surely be Linear. The IF of most radio astronomical spectrometers, the sample times in most time sequences, and the pixel positions in Reticon and CCD arrays are all linear functions of channel number. Corresponding to a Linear generalized position is a Constant generalized spacing. For the three examples given above, the Constant spacing would be the channel spacing, the sample time, and the pixel size respectively.

Imperfections in optical devices such as acousto-optic spectrometers and CCD cameras often introduce nonlinearities in the coordinate axes. The class Polynomial is one possible representation of a nonlinear axis. Other possible classes, not illustrated, might include spline functions and trigonometric polynomials.

For some devices, the sampling of the coordinate axis may be too scattered to be represented by any simple, functional form. In this case, a LookUp table may be the appropriate description. For example, a spectrometer used to probe atmospheric lines at millimeter wavelengths provided 128 channels ranging in width from 10 KHz up to 32 MHz. A custom data reduction program was written for this device, when a simple LookUp table could have allowed the data to be reduced with a standard package.

Some properties, such as the weight assigned to each channel, can assume a finite (usually small) number of real values. For these quantities, a PiecewiseConstant may be an appropriate representation, being significantly smaller and faster to process than a LookUp table.

In some cases an expression may be known for the function which is most conveniently written as a `double function(int)`, with or without internal parameters. In order that these cases may be handled in the same way as any other FuncOfChan, the class FuncOfInt provides the necessary conversion. C++ functions which want a FuncOfChan may be overloaded using FuncOfInt to extend their utility to ordinary functions:

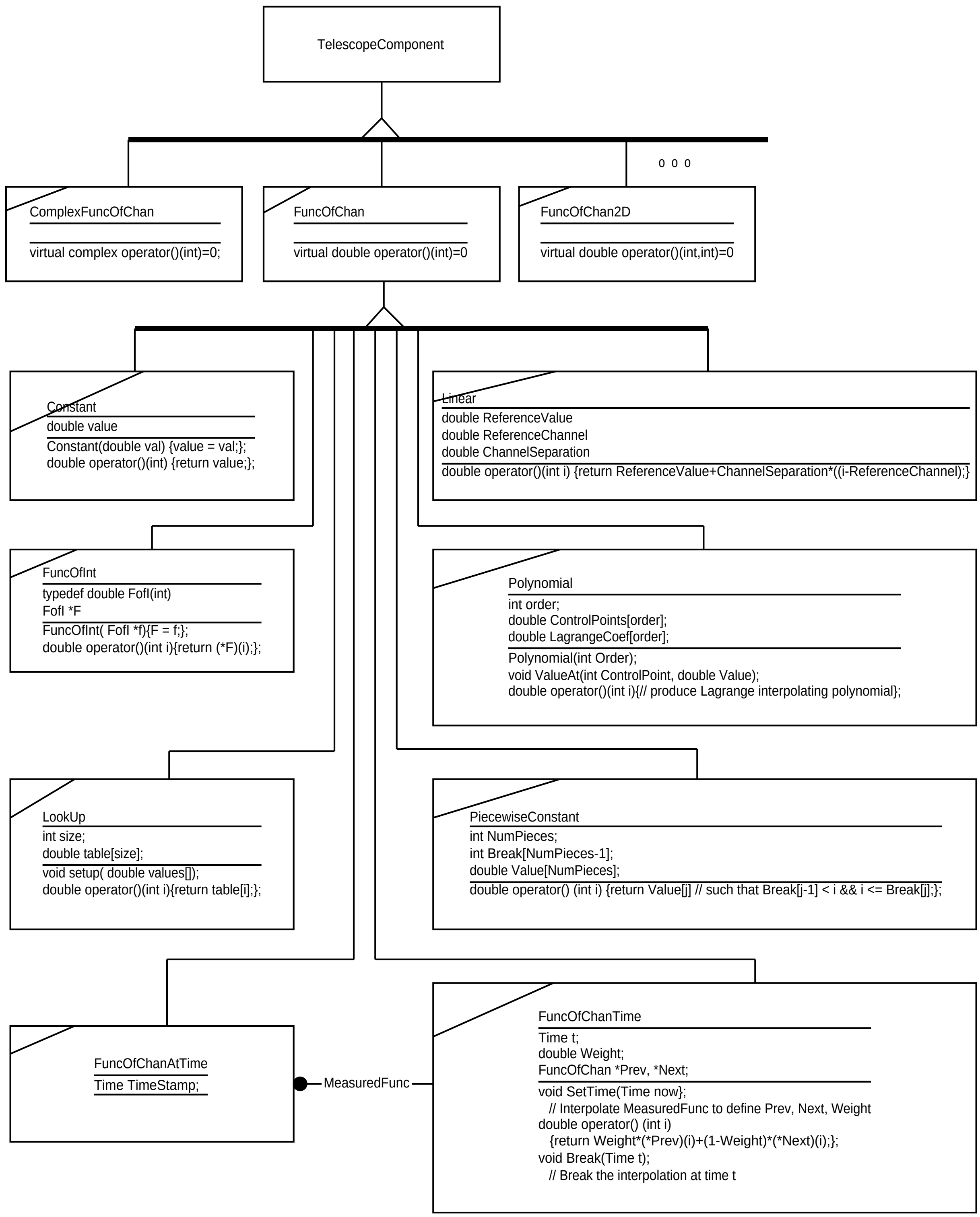


**Figure 2** — Mathematical Representations for Coordinate Axes

```
double dosomething( FuncOfChan &f);
double dosomething( double f(int)) {return dosomething(FuncOfInt(f));};
```

A very useful class not shown in Figure 2 would be a dynamically defined function, whose definition could be provided as an ASCII string either from the keyboard or perhaps as an entry in the header of the YegSet. This string would be parsed into tokens and semi-compiled by the class constructor so that it could be executed rapidly. The necessary tools to define such a function should be available almost for free from the user interface. This kind of function could replace all of the simpler classes in Figure 2, and would allow the YegSet to be nearly self-documenting, since the functions defining the behaviour of the coordinates could all be written explicitly in the header of the YegSet.

Many properties of a TelescopeModel are slowly varying functions of time which may affect the signal in each channel in arbitrary ways. The most common calibration process will be to measure these properties at particular times and to estimate the corrections for the remaining data by interpolating in time between the measured values. For times before the first calibration measurement or after the last, the value at the nearest calibration should be used, i.e. constant extrapolation rather than linear extrapolation. As shown in Figure 2, the class FuncOfChanTime provides this functionality. It is intended that FuncOfChanTime should store the measured FuncOfChans in a time-ordered list, and that the pointers Prev and Next should be used to store the most recently accessed place in the list. This should allow rapid access to the required values for most purposes. Note that FuncOfChanTime is intended to be accessed potentially many times by channel for each change in time. If no measurements have been entered into a FuncOfChanTime, it should return a programmer specified value entered through the constructor; this should be a 0 for additive functions or 1 for multiplicative functions. Finally, it should be possible to insert a "break" into the interpolation, i.e. at a specified time the interpolation makes a discontinuous jump and on either side of the break is constant and equal to the next measured value.

# 3 Flux Density Calibration

Virtually all astronomical observations are intended to measure the flux of radiation into the detectors on the telescope. This is one of the few universal properties of all astronomical detectors; every top level TelescopeModel will have an InstrumentModel defining the flux calibration, even if subsequent processing reduces the data to a nonlinear property of the flux, such as the closure phase or a polarization position angle.

Considerable care is usually taken to ensure that the signal output is linear in the flux density at the expected signal strengths, although the current discussion will not be restricted to linear devices. (CCD data reduction, for instance, often corrects for the nonlinearity of the A/D converter before applying any further processing.) Regardless of whether the instrument response is linear or not, the slope of the response for a weak signal defines the gain of the system and is a most important, measured property of the instrument.

Most instruments will have a zero level as well as a gain, i.e. the signal output with no input. For optical detectors such as CCD's this is also known as the "dark current". How the zero level is handled depends critically on the timescale on which it varies. If this timescale is extremely short, the original offset might be removed in hardware, perhaps by a lock-in amplifier; the output of the lock-in amplifier will have its own offset which will probably vary quite slowly. Alternately, the rapid variations may be removed by accumulating a chopped signal in two related Yegs, which are conventionally labelled “On” and “Off”. In principle the difference signal ON-OFF should be free of the offset. In practice, removing the remaining “baseline”, the unmodelled part of the zero level, is often a major chore in spectroscopic data reduction.

The gain and the offset either will be independent of the channel number, or will vary from channel to channel so unpredictably that lookup tables must be used to represent them. As an example of the former behaviour, a time series of power entering a continuum detector should use the same gain for every channel. Generally speaking, we can expect that a gain which is independent of channel will vary slowly with time as well. As an example of the latter behaviour, the gain of a spectrometer fed by a heterodyne receiver will vary dramatically from channel to channel, and must be represented as an array. Most often in these systems the gain and offset will vary rapidly with time and must be treated as ephemeral quantities. Sometimes it is necessary to apply both an ephemeral correction plus a slowly-varying, channel-by-channel correction.

Calibration of the flux density actually involves calibration of two distinct properties of the system: the gain of the lumped receiver, IF and backend, and the system temperature. The gain is an ephemeral quantity for many systems and is typically calibrated using On, Off and Dark spectra from the ratio (On-Off)/(Off-Dark). The flux density is calculated by scaling this ratio by the system temperature Tsys. The system temperature is often treated as a constant, independent of channel (i.e. a Constant FuncOfChan) although it is usually better to use a LookUp FuncOfChan whenever possible. It can usually be treated as a slowly varying quantity. In some systems the quantity Tsys/(Off-Dark) is stored as the overall gain. In all cases, it is appropriate to store this quantity in the `Gain` FuncOfChanTime.

To handle the numerous possibilities, the class FluxCalibration provides two FuncOfChanTime for the `Gain` and `ZeroLevel` (neither of which needs to be used) to handle the slowly-varying components in the flux calibration, as well as `solve()` and `apply()` methods suitable for the most common ephemeral calibrations. Providing default pointers to 0 for `Off` and `Dark` allows these Yegs to be omitted when they are not appropriate.

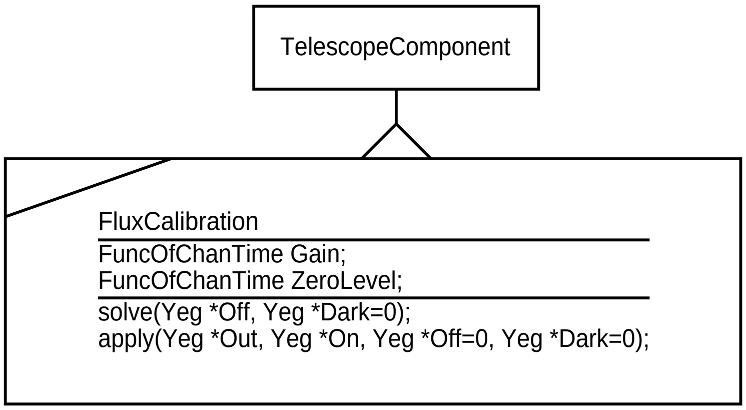


**Figure 3** — The One Dimensional FluxCalibration for Real Numbers

One significant distinction between second-order systems, such as interferometers, and first order systems, such as spectrometers, is that the former are sensitive to the phase of the incoming signal, whereas the latter are not. This is often represented by writing the data as complex numbers. Correspondingly, the gain and offsets must be complex quantities. The class `FluxCalibration` will have a twin `ComplexFluxCalibration` which is identical except that all instances of `FuncOfChan` are replaced by `ComplexFuncOfChan`. Similarly, two (and higher) dimensional detectors will replace FuncOfChan with the corresponding FuncOfChan2D (and so forth).

Although an instrument actually records the integrated power incident on the detector elements, the signal is most often interpreted as a flux density or some quantity proportional to it:

$$F_\nu, \quad F_\lambda, \quad \lambda F_\lambda \quad \text{or} \quad \mathrm{T}_\mathrm{A}^*.$$

These quantities depend both on the detected power and the channel width. If the dispersion of a spectrometer is not linear, or if the optics suffer from aberations, the channel width may vary across the spectrum and from Yeg to Yeg. It is therefore important to calibrate the frequency (wavelength) before applying the FluxCalibration.

An important part of calibrating the intensity in each channel is to estimate its associated noise. Interestingly, the noise in the measurement has exactly the same relation to the measured intensity as the channel width has to the center frequency or the beam width has to the beam center. It is simply the width of the probability distribution of the intensity resulting from the measurement. For data reduction purposes it is often more convenient to represent the noise by a weight given by the inverse of the variance, so that the weight of the sum of two numbers is simply the sum of the weights. For a radio astronomical spectrum in temperature units this weight is normally calculated

$$w = 1/\sigma^2 = kBt/T_{SYS}^2$$

where B is the channel bandwidth, t is the integration time, Tsys is the system temperature at that channel and k is a constant depending only on the switching mode used for the observations. Ideally this weight should be calculated seperately for each channel and stored as a LookUp FuncOfChan. To save space, however, it is often desireable to compact the weight into a PiecewiseConstant, or even a Constant, for each Yeg. The weight should be stored in a calibrated Yeg along with the measured intensity, with the FluxCalibration in the InstrumentModel providing (implicitly, at least) the probability density function for its interpretation.

# 4 Frequency Calibration

In a spectrum the independent variable may be interpreted as frequency or wavelength, and for data reduction purposes is distinguished from other independent variables by its sensitivity to the velocity reference frame. The Doppler shift will be supplied by an associated PlatformModel through a spectrometer TelescopeModel, and only its application will be considered here. For the present purposes the primary difference between a frequency and a wavelength scale is its reaction to a Doppler shift.

Most radio astronomical spectrometers are heterodyne systems. Calibration of the frequency scales on these systems will normally involve reading the LO from the Yeg header and the IF parameters of the backend from an observatory database. Most optical spectrometers disperse the light with optical components such as gratings and prisms which for which the natural axis is the wavelength of the light in air. Calibration is often accomplished by injecting a known signal such as an iron arc, identifying features in the spectrum, and fitting a polynomial to their apparent positions.

Figure 4 illustrates the frequency and wavelength classes. Note that although a general Heterodyne class and CalibratedFrequency class are defined, redundant classes CalibratedLinearHeterodyne and CalibratedNonlinearHeterodyne are also defined, since these classes define the majority of all normal radio astronomical spectra.

The final object in Figure 4 is a velocity calibrator called RadialVelocity for a particular velocity reference frame. This object is quite different from the other objects in the diagram, since it requires a direct link between a PlatformModel defining the transformation from frequency into radial velocity, and an InstrumentModel defining the frequency scale. It is in fact a TelescopeModel, not a TelescopeComponent, and is included in this diagram only because of the tight link in the minds of most astronomers between velocity and frequency. Since the structure of PlatformModels lies outside the scope of this document, the operation of RadialVelocity cannot be specified further.

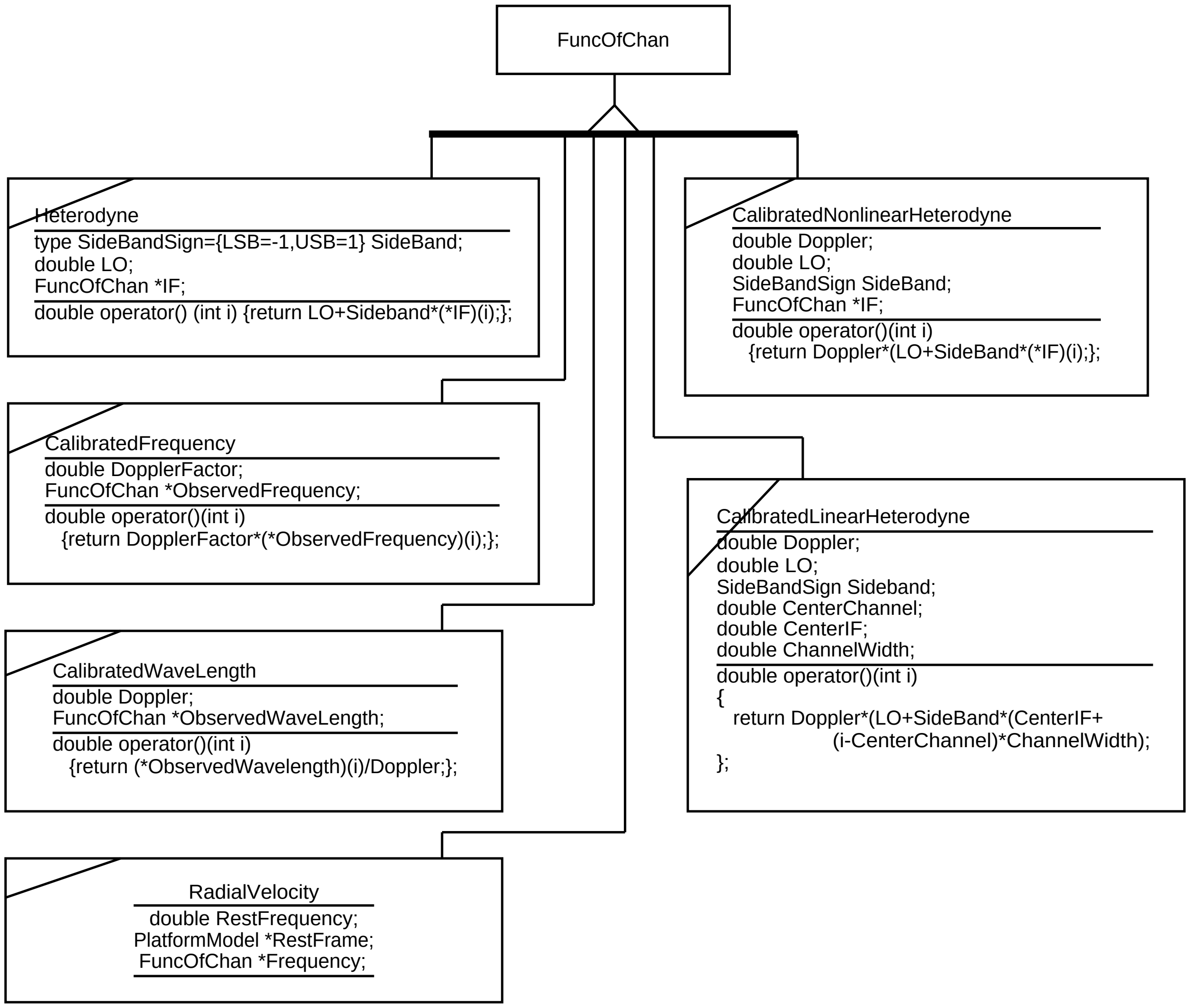


**Figure 4** — Frequency and Wavelength Calibration Classes

# 5 Right Ascension and Declination

In the astronomical context, a map represents a function of position on the celestial sphere. There are several different approaches to mapping an intensity distribution on the sky, such as using arrays of detectors, or by scanning the telescope, or even by bodily moving whole sets of telescopes around as is done in interferometery (often with the aid of the rotating Earth). Maps are most often made by offsetting from a nominal map center, sampling on a grid in offset coordinates. Although the center of the map is a property of the TelescopeElementModel, the offset coordinate system is usually closely tied to the detectors, and is properly part of the InstrumentModel. Combining the instrumental offsets with the center position provided by the TelescopeElementModel is the responsibility of the coordinating TelescopeModel.

The primary axis of the offset coordinate system will be labelled the X coordinate. If the telescope scans to take data this will be the direction in which it moves most rapidly while sampling. In a fixed array of detectors, the X axis will correspond to the first index into the array. The perpendicular direction in the sense of increasing position angle on the sky will be called the Y coordinate. The orientation on the sky is defined by the position angle PosAng of the X axis. The second sampling coordinate is normally Y, giving a rectangular grid. If the samples are uniformly spaced at intervals of dX and dY, the pair of numbers (dX,dY) defines the unit cell for the grid and is refered to as the cell size. Another popular strategy is to sample at fixed intervals in position angle, defining a polar offset coordinate system, in which case dPosAng will refer to the increment in position angle. In more complicated systems there may be several sets of offset coordinates in use at once. The class OffsetCoordinates, shown in Figure 5 provides several tools useful for working in offset coordinate systems with modest requirements for accuracy.

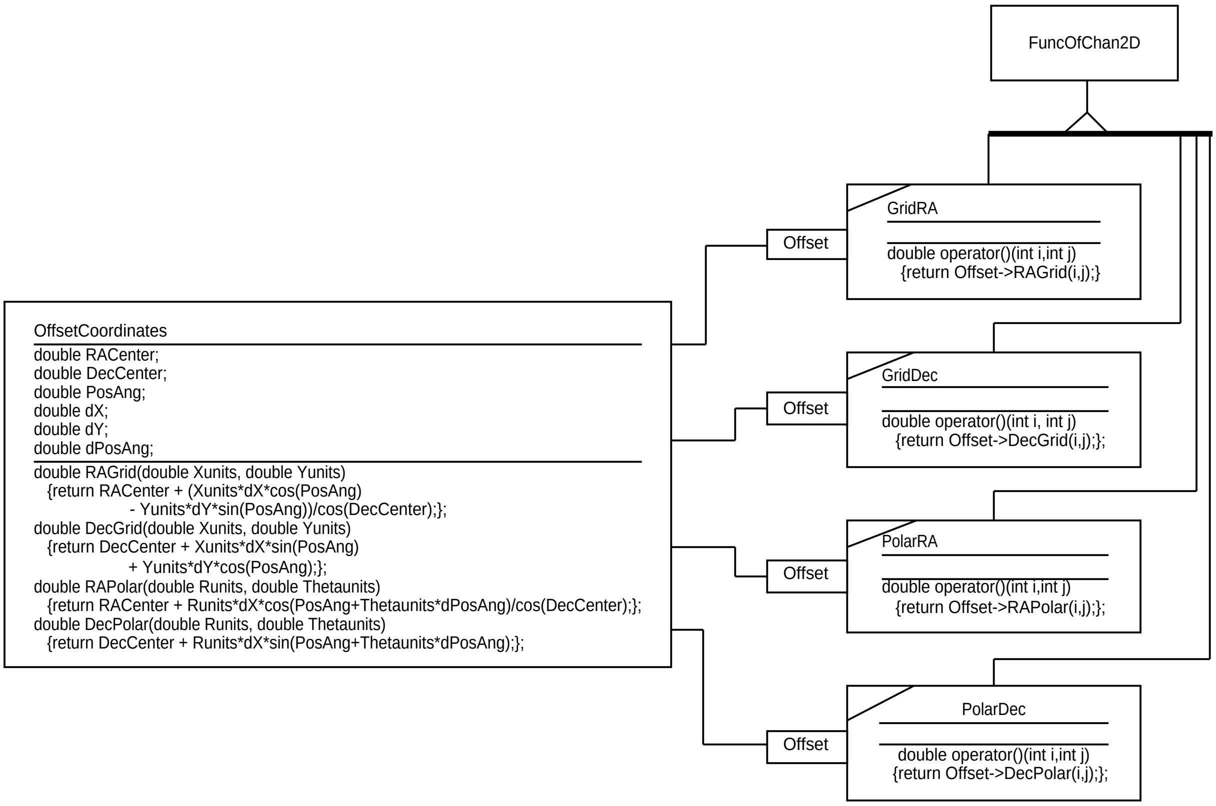


**Figure 5** — Offset coordinates for mapping

Any map must provide two FuncOfChan2Ds giving the right ascension and declination of channel (i,j). Classes GridRA, GridDec, PolarRA and PolarDec are examples which will be suitable for most situations requiring modest accuracy. Note that by associating the FuncOfChan2D with a projection class, rather than embedding the projection inside the FuncOfChan2D, we save duplication of the projection and avoid a possible misalignment of the coordinate axes should the programmer neglect to update both axis projections.

# 6 Time Series

Time series represent one of the simplest coordinate axes, generally just a Linear FuncOfChan. The various options for the fiducial time, such as the start of the observation, the midpoint of the series, or the midpoint of the first sample in the series, may all be handled by specifying the Linear.CenterChannel. Very long or very precise time series may need a Doppler correction, for which a class identical to a calibrated linear wavelength scale could be used.

# 7 Polarization

?????

# 8 Examples of TelescopeModels for Instrumentation

Several examples will illustrate how complex devices can be assembled from the components described in the previous section, following the template of a GenericInstrument as shown in Figure 1.

## 8.1 An Autocorrelator Spectrometer on a Single Dish

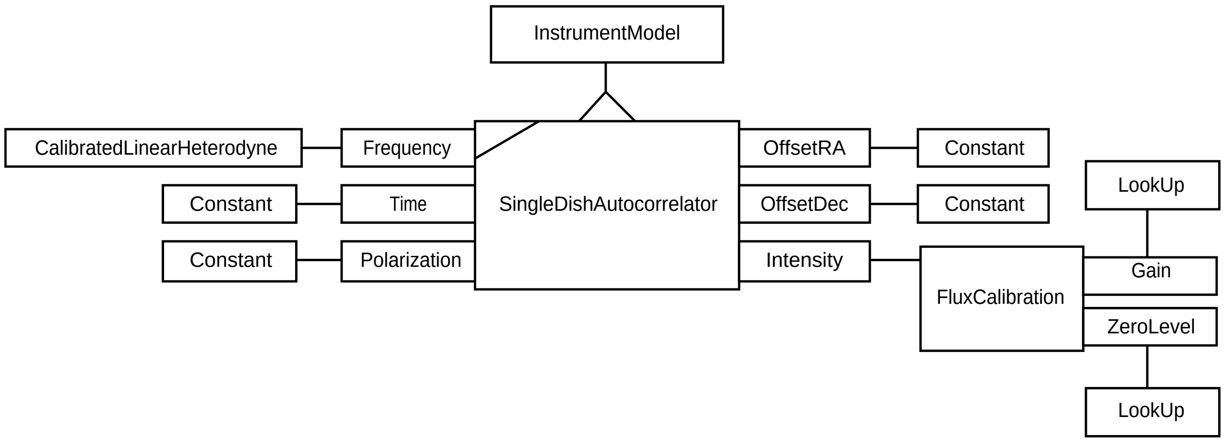


**Figure 6** — An Autocorrelator Fed from a Single Receiver

An autocorrelator is intrinsically linear in frequency so that a CalibratedLinearHeterodyne should completely and accurately describe the frequency dependence. The intensity calibration is handled using LookUp tables for the Gain and Zerolevel. For most switching modes the Zerolevel is actually superfluous. This distinction will have to be handled in more detailed models of the FluxCalibration object. All the other aspects of the system are effectively Constants (**not** `const`), although some, such as the polarization, may vary in a significant way from Yeg to Yeg.

## 8.2 A Multi-Horn Spectrometer Array

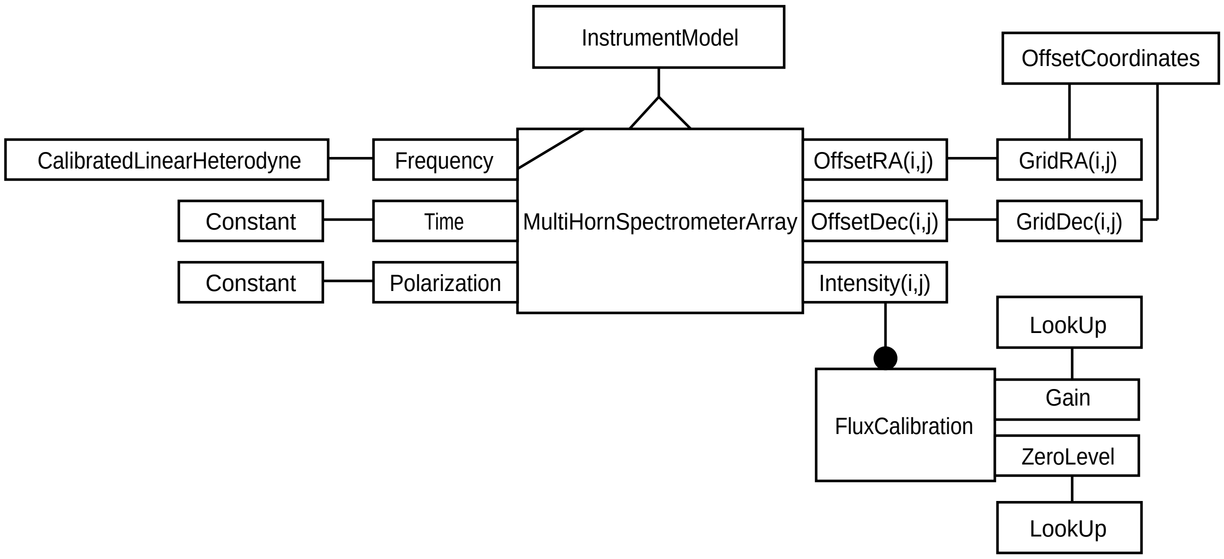


**Figure 7** — Array of Receivers with Matching Autocorrelators

Most receiver arrays will use a common LO, perhaps generated by different oscillators but slaved to a common frequency. They will usually feed a matching array of identical backends, which in this example are taken to be autocorrelators. If AOSes are used, replace CalibratedLinearHeterodyne with CalibratedNonlinearHeterodyne. The receiver array is assumed to be rectangular so that the offsets may be calculated directly using the GridRA and GridDec functions provided by OffsetCoordinates. Note the implicit type conversions `int` to `double` in the calls to GridRA and GridDec which actually define the rectangular grid. Other geometries for the receiver array may use a different mapping into the Offset-Coordinate space. In extreme cases, it may be necessary to use a LookUp(i,j) to provide the locations of the individual horns in OffsetCoordinates. The Intensity is treated in this case as a function returning a pointer to a FluxCalibration for each reciver. This could be handled equally well using a LookUp3D, which would eliminate the multiplicity of the association of Intensity to FluxCalibration.

## 8.3 A Long-Slit Spectrograph

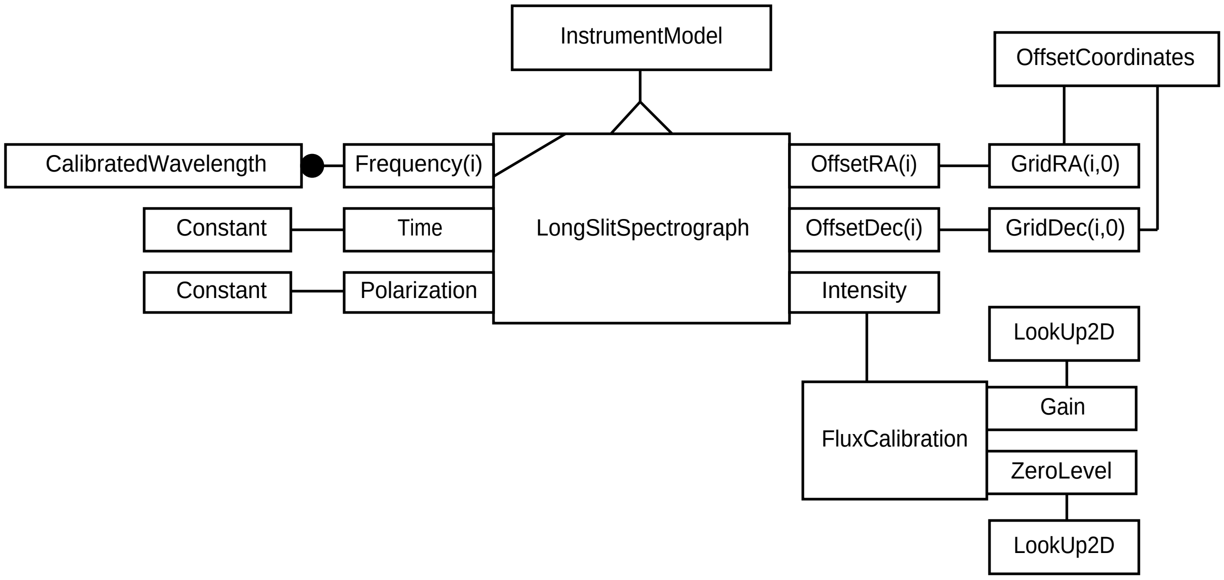


**Figure 8** — A Multi-Spectrometer Array Modelled on a Long-Slit Spectrograph

In this kind of instrument a two-dimensional camera such as a CCD is aligned with the slit so that one axis of the array represents frequency and the other is an offset coordinate. It will be recognized that the very similar structures are widely useful. A receiver array with all of the horns in a line has the same structure, with an appropriate substitution for the frequency component. Similarly the fiber-optic spectrometer feeds which pick the light from many galaxies in the field of the telescope and feed them to a common spectrometer has the same structure except that a LookUp is needed into the OffsetCoordinates. Because of distortions in the optics, it may be necessary to use a Polynomial in the CalibratedWavelength structure which may differ from one spectrum to the next.

## 8.4 A Direct-Imaging CCD Camera

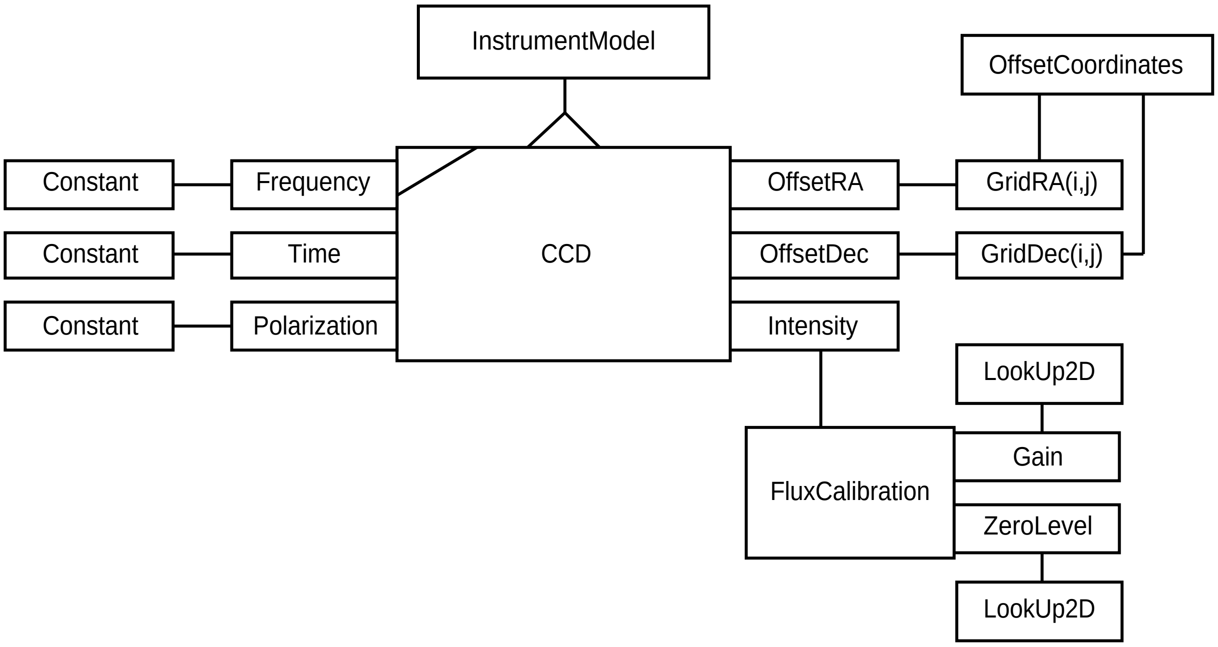


**Figure 9** — A CCD Camera Used for Direct Imaging

For direct imaging, the frequency will be effectively constant, and should be a label in most cases rather than a number. It should be noted that a continuum bolometer such as SCUBA will look nearly identical st this level of abstraction. Mapping a region by scanning with a single detector on a rectangular grid would change the Gain and ZeroLevel in FluxCalibration into FuncOfChanTimes supplying a Constant as a slowly-varying function of time.