



# Planning Tool Suite -- getCal

Andy Boden

ISC/Caltech

ISC Shared-Risk Users Workshop – April 15, 2002

# Outline



- Introduction to the getCal Suite
- Key Features
- Design Overview
- Illustrative Use Cases
- Installation/Dependencies
- Wrap-Up



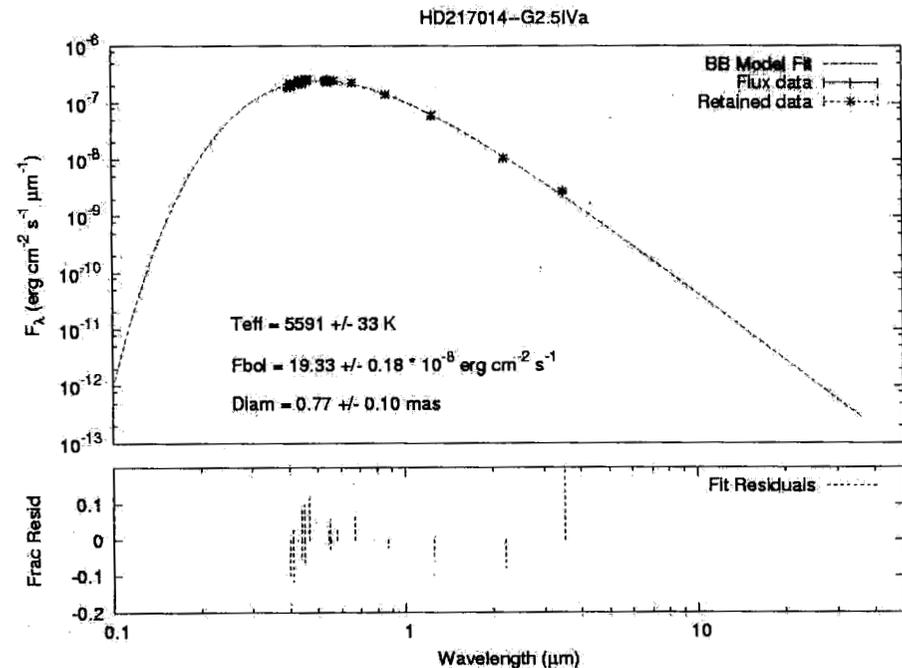
## Introduction to the getCal Suite

- getCal is a PTI-heritage Interferometry Experiment Planning Tool That Attempts to Assist User in Composing an *Observation Plan*
- Designed With Wide (Unix) Portability in Mind
- Implemented as Numerous Small Components; Interface Either Through Top-Level Glue-Scripts and GUIs or Directly With Components



# Attributes of a Good Calibrator?

- Attributes of a “Good” Visibility Calibrator
  - Unresolved (minimally resolved)
  - “Apparently” Single
  - Bright (or similar to target)
  - Similar observing geometry to target (near in sky)
- Identifying Good Calibrators
  - Geometric search
  - Astrophysical constraints
  - Angular diameter estimation
  - Spectral energy distribution analysis
  - Ancillary information (e.g. Simbad classification & measurements)



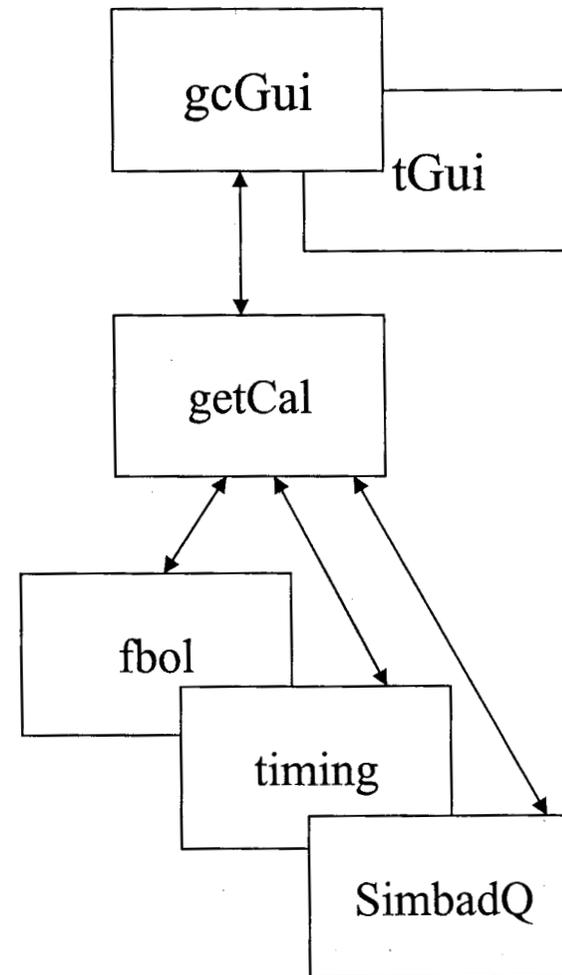


# Key Features of getCal

- getCal is an Experiment/Observation Planning Tool That:
  - Resolves astronomical designations into standardized catalog entries and astrometry (via Simbad)
  - Identifies potential visibility calibration sources according to various observational and/or astrophysical criteria
  - Retrieve broad-band photometry from archival (Simbad, Catalog of Infrared Observations) sources and model spectral energy distribution (SED) with effective temperature/bolometric flux/angular diameter parameters
  - Computes observing accessibility and geometry according to various constraints
  - Various GUIs that facilitate access to components
  - Interfaces to KI Control Components
    - ❖ Composes KI “Astronomical Observing Template” (AOT)
    - ❖ Keck “sky” planning application

# getCal Design Overview

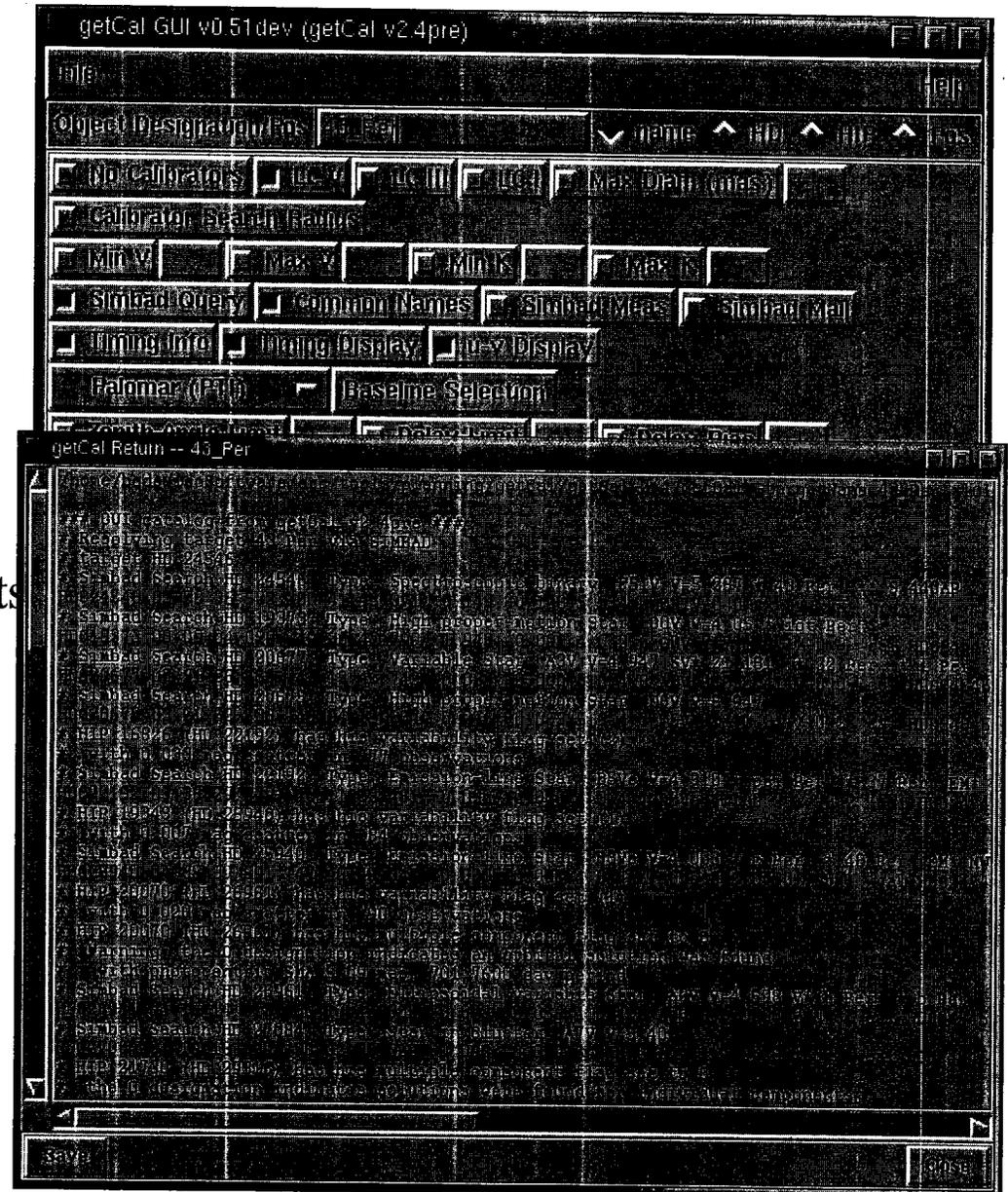
- getCal is designed as multi-layer toolset
  - GUI level – GUIs that interface with command-line tools the facilitate interface or present results (e.g. gcGui, tGui)
  - Wrapper level – top-level scripts that provide consolidated functionality with command-line interface (e.g. getCal, gcList)
  - Component level – individual components that implement individual functions (e.g. Hipparcos catalog “cone search”, Simbad name resolution & information retrieval, accessibility calculations)
- Script (perl) implementation to enhance portability





# Illustrative Use Cases (1)

- Identify Candidate Calibrators for Given Source
  - Geometric search
  - Magnitude constraints
  - Astrophysical constraints (e.g. luminosity class, apparent diameter)
  - Multiplicity vetting





# Illustrative Use Cases (2)

- Accessibility calculations
  - Annual accessibility
  - Diurnal accessibility
  - u-v tracks

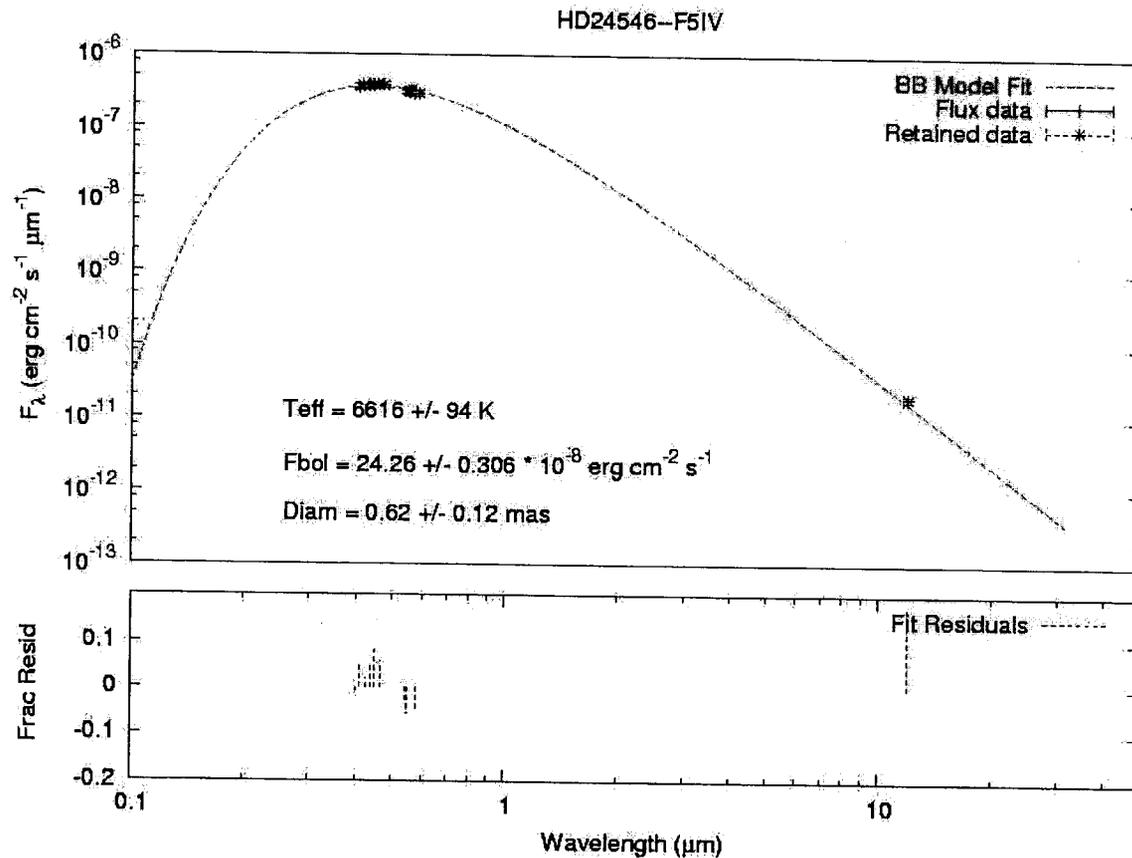
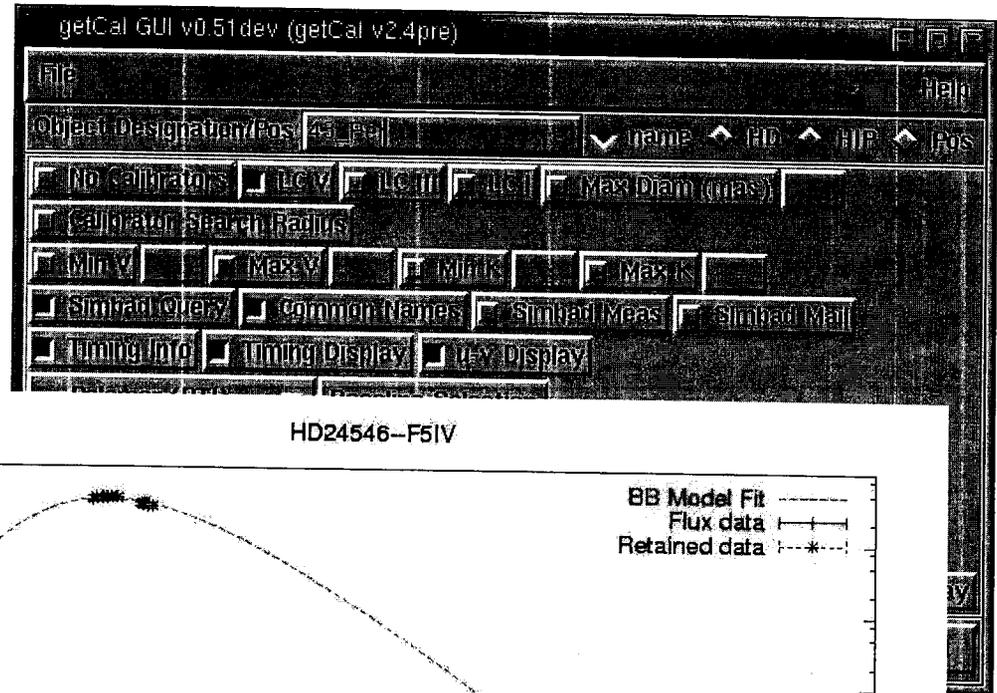
The image displays three overlapping screenshots of astronomical software GUIs:

- obsCalendar GUI v0.1dev (getCal v2.4pre4):** Shows a calendar interface with a grid for months (Jan, Feb, Mar, Apr, May, Jun) and years (2000, 2001, 2002). It includes a 'Target' field and a 'Date' field.
- getCal GUI v0.51dev (getCal v2.4pre):** Features a 'File' menu, 'Object Designation/Pos' field (set to '43 Fall'), and various checkboxes for 'No Calibrators', 'UCV', 'UCM', 'UCI', and 'Max Diam (mas)'. It also has a 'Calibrator Search Radius' field.
- timing GUI v0.83dev (getCal v2.4pre):** Contains a 'File' menu, 'Target' field, and a 'Baseline' field. It displays a large table of data with columns for 'ID', 'U', 'V', 'B', 'S', 'D', 'E', 'N', 'S', 'E', 'N', 'S', 'E', 'N'. Below the table are buttons for 'Inverted track', 'u-track', 'Diary track', 'Observed track', and 'Baseline'. At the bottom, there are buttons for 'Current GWLW', 'Source/Source', 'Time label', 'Clusters', 'Calibrators', 'Moon', 'Sun', and 'Cross'.

# Illustrative Use Cases (3)



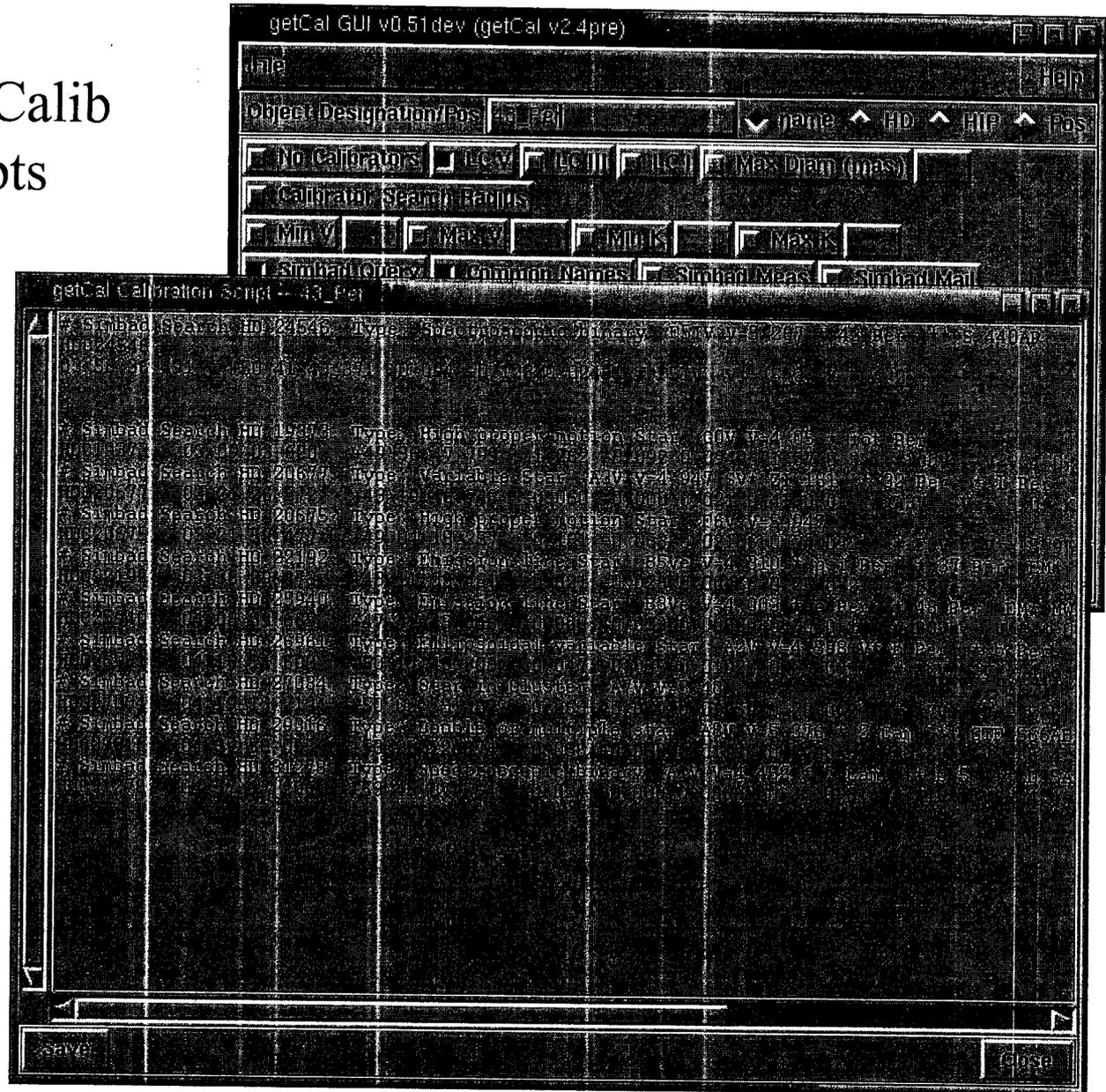
- Spectral Energy Distribution/Bolometric flux – effective temperature modeling



# Illustrative Use Cases (4)



- Composing wbCalib calibration scripts



# getCal Documentation



- getCal documentation is on-line at [isc.caltech.edu](http://isc.caltech.edu)

**getCal v2.3 -- PTI/KI Experiment Planning Software**

getCal is interferometric experiment planning software. It can be used to resolve common star names into standard catalog designations and astrometry; extract visibility calibrators from the Hipparcos catalog according to a variety of geometric, brightness, and astrophysical criteria; compute target zenith and delay accessibility; interface with the Simbad astronomical database, the PTI GUI, visibility calibration codes, and sky visualization software. getCal also has other uses, is extremely portable, can be autonomously driven over a list of sources, and can be accessed either from either the Unix command line or a (Perl/Tk) GUI.

**getCal Highlights**

To the user, getCal appears as an integrated application available from both command-line and GUI that among other things:

1. Resolves standard astronomical names (e.g. Iota Pegasus) or catalog designations (e.g. HR 8430) into standard catalog (e.g. Henry-Draper catalog) designation, rough classification (e.g. Spectroscopic Binary), and astrometry via Simbad.
2. Optionally extracts ancillary/quantitative information from the Simbad database (e.g. broadband photometry, v sin i).
3. Extracts potential visibility calibrators from standard astronomical catalogs (e.g. Hipparcos) according to geometric and astrophysical criteria (e.g. brightness, luminosity class, estimated angular diameter), and will also query Simbad on the potential calibration objects (1 and 2).
4. Estimates calibrator angular diameters by a variety of methods, including bolometric flux/effective temperature methods based on spectrophotometry retrieved from Simbad.
5. Computes target and calibrator accessibility (zenith angle and delay) restrictions for user-defined observation location/baselines.
6. Value-added timing GUI tool generates displays of target (list) timings, and runs in real-time against a nightly observing list, displaying LST and sunset/sunrise indicators.
7. u-v GUI tool displays u-v tracks limited by delay and zenith-angle restrictions on user-selectable target and baselines. Also runs in real time to provide current u-v and relative geometry information.
8. Composes provisional calibration scripts used to interface with/drive standard KI (PTI) visibility/V2 calibration codes.
9. Outputs target astrometry in either PTI-standard or Keck-standard (sky) input formats.
10. Drives xephem sky plotting software for sky visualization.

**getCal Model**

getCal is really a multi-layered tool, bound together by a top level wrapper script. As originally developed it is designed to run purely as a command-line tool; essentially all the functionality in the getCal suite is accessible via command-line invocation. The wrapper script (getCal itself) uses individual programs in the getCal suite (many of which have utility outside of getCal proper) to provide the getCal functionality. The command-line interface allows getCal to be run in text-only mode and driven automatically from even higher-level scripts. getCal can even profitably without network connection (a must when you're trying to put those finishing touches on your observing plan in route...).

The canonical getCal command-line invocation looks something like:

```
getCal -targetName Iota_Peg
at which point your computer should respond with something like:
### GUI catalog from getCal v2.2.1dev ###
# Resolving target Iota_Peg via SIMBAD
# target HD 210027
HDC210027 22 07 00.666 +25 20 42.402 0.328 0.027 3.8 2.7 F5V 0.0 xxxx xxx trg
```

<http://isc.caltech.edu/software/getCal>

# getCal Installation & Dependencies



- getCal build/installation recipe:
  - install.getCal – shell script that prepares components
  - install.extern – shell script that makes external symLinks into convenient path location (e.g. /proj/isc/iscSoftware/bin, /usr/local/bin)
  
- getCal dependencies
  - Perl 5 (e.g. 5.005, 5.6.0)
  - Perl/Tk (800 series)
  - Hipparcos catalog and annexes
  - Simbad username/password (& network connectivity)

## getCal To Do List



- Add KI long delay line (LDL) position optimization tool
  - Users can see and optimize LDL position for individual objects or experiment clusters
- Rework build & install procedures in a more standard (i.e. GNU Autotools) methodology
- Add better timing visualization support for full-array operations