Proposed Instrument Software Scheme for CASIMIR and GREAT on SOFIA

Version 1

A. Harris, N.S. Amarnath, M. Pound, K. Rauch, P. Teuben

10 September 2002

1 Introduction

CASIMIR and GREAT are the two first-light heterodyne instruments for SOFIA. At a fundamental level they contain similar subsystems with simple interfaces, so it makes sense to develop a common software scheme. In this note we propose a conceptual plan that to a large extent builds on software that has already been developed and tested. This plan partitions the software in a small number of high-level tasks, describes each task, and specifies the communication between tasks. This modularity with clean and simple interfaces is important for a distributed software effort. It also allows efficient testing. For example, the task for driving the SOFIA telescope could be replaced by one that drives the KOSMA or CSO telescopes with no change in interfaces to the rest of the software to allow receiver and spectrometer testing on a telescope before SOFIA flys. We also describe a possible breakdown of the backend task that controls the backend spectrometers, archives system data, and generates quick-look spectra for display.

2 Software conceptual design

Figure 1 is a block diagram containing the software conceptual design we propose. It contains six tasks shown in solid boxes: Observing, Telescope, Receiver, Calibration system, Backend, and Database. Communication between these tasks is through the KOSMA `file_io` formalism, shown in boxes with dashed outlines, a set of ASCII files that unidirectionally pass parameters from one task to another. The `file_io` files contain timestamp information that permits temporal synchronization of actions by different tasks. In addition to this network of tasks and communications, Figure 1 shows the quick-look spectral display, currently specified as the stand-alone CLASS spectral line package with a CLASS-FITS intermediate file.

2.1 The tasks

2.1.1 Observing task

The observing task is the interface to the human observer, who specifies source coordinates, local oscillator frequencies, observing modes (chopping, OTF mapping…) backend spectrometer configurations, integration times, and so on. This task could contain source catalogs and line lists to assist the observer. It reports error and system status data to the observer and might include routines for helping the observer with pointing and other tasks.

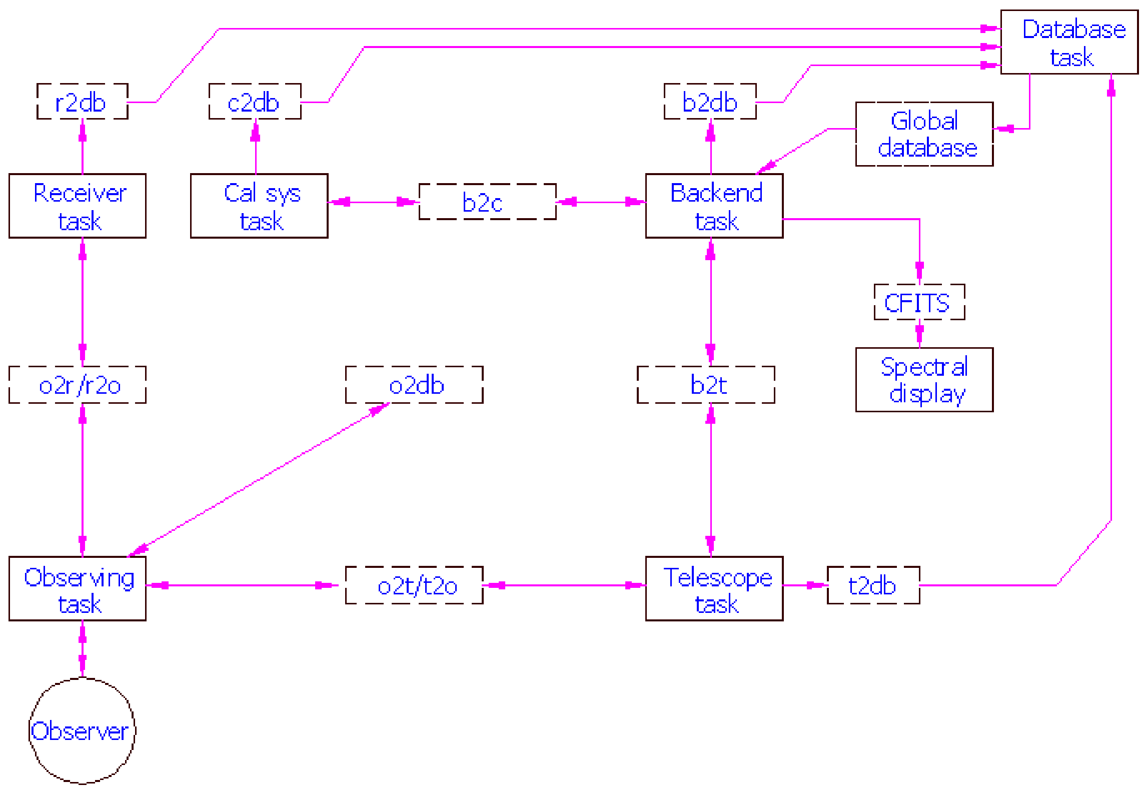

Figure 1: Block diagram of tasks and communication files.

2.1.2 Telescope task

The telescope task is the interface between the observing task and the MCCS for telescope related information such as source coordinates, position offsets, chopper setups, tracking error status, and so on. In its simplest form it is a filter that converts information from the `file_io` format to statements that the MCCS can follow. It may also incorporate telescope limit information as an error check.

2.1.3 Receiver task

The receiver task sets the local oscillator frequency, monitors temperature and mixer bias conditions, verifies phase lock, and so on.

2.1.4 Calibration system task

The calibration system task controls the amplitude calibration black body load positions and temperatures and any other calibration parameters that are common to all backends. Backend spectrometers with specific calibration routines (e.g. comb spectra for AOS or phase measurements for analog correlators) are part of individual backend tasks.

2.1.5 Backend task

The backend task controls the backend spectrometers, produces spectra calibrated in amplitude and velocity, and archives spectral and system status information. Since different spectrometers and different observing modes need different calibration data, the backend task also includes all of the calibration logic and interfaces to calibration systems. Section 3 outlines this task is described in more detail. The backend task incorporates some of the data from the global database maintained by the database task into headers and may record other data in bulk form within the archive. All of the backend task inputs are through the global database with the exception of a synchronizing `file_io` connection from the database task.

2.1.6 Database task

The database task gathers data from the `file_io` communication channel files to record the state of the entire system for each integration in a global database. The database task is also responsible for generating and recording observatory-specific data that is commonly available at most observatories (such as LSR velocity correction along the line of sight, conversion of boresight positions from telescope coordinates to astronomical reference frame, etc.) from its inputs.

3 Backend task

Figure 2 is a schematic view of the backend task to show its modularity and interfaces to external tasks.

3.1 Global database and file writer

The right hand side of the diagram shows a database that records the global system state and a common file writer program that records data from all spectrometers. Setting keywords in the internal global data base selects the spectrometers that will be active for an observation and sets the observing mode common to all spectrometers (e.g. chopped, on the fly map, etc.). Other keywords set integration (exposure) time, specify chop and nod timing, and so on.

The file writer reads raw and calibrated data from an arbitrary number of spectrometer data reduction pipelines connected to spectrometers with an arbitrary number of spectral channels including single-channel data for pointing. It produces several possible outputs:

3.1.1 SDFITS

The backend task will always produce an archive file in SDFITS format that contains all raw and first-cut calibrated data from all spectrometers. This file could also contain observatory and system status information (e.g. temperatures, bias currents, aircraft heading, boresight water vapor, etc.) that are not directly related to data reduction and that would not normally appear in spectral header data. A single entry in the file contains raw and reduced data, as well as header references to calibration files, for all active spectrometers in each integration cycle. SDFITS is compatible with the DISH package and is theoretically compatible with CLASS.

3.1.2 CFITS

The task can generate a scratch file containing a calibrated (amplitude and file) in CLASS-specific FITS format for each active spectrometer output. The files use the CLASS-FITS format to avoid version problems; it is then straightforward to read them into the CLASS database for further processing.

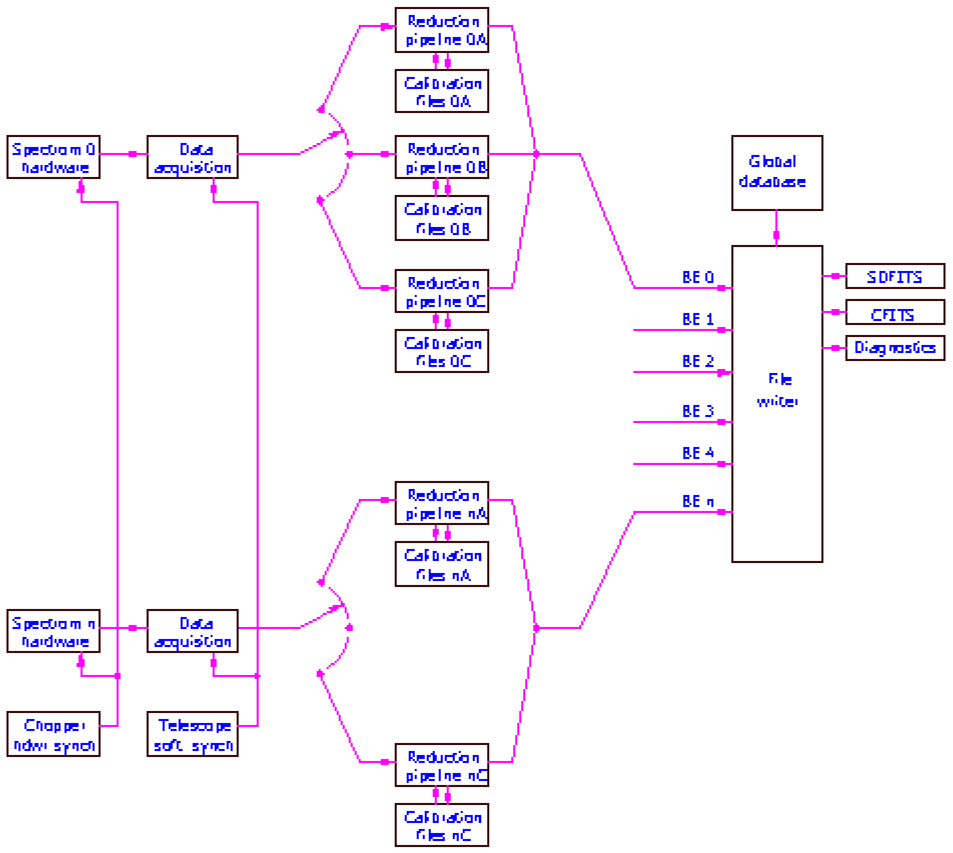

Figure 2: Schematic diagram of backend task

3.1.3 Diagnostics

The task can also generate summary files in ASCII format for diagnostic use.

3.1.4 Other formats

Generating files formatted for other purposes or reduction packages is possible by adding a keyword to the global data base and by writing the appropriate filter from the internal data format to the necessary output format.

3.2 Multiple spectrometers, reduction pipelines, and observing modes

Each set of spectrometer hardware has a single data acquisition system but will have different data reduction pipelines depending on the spectrometer and observing mode. The left-hand side of Figure 2 shows this, with a schematic switch corresponding to the observing mode keyword that selects the same pipeline for all active spectrometers for a given mode. Mode "B" is missing for spectrometer n, indicating that not all spectrometers will need all observing modes. Each reduction pipeline requires a specific set of calibration files, and the overall pipeline includes software to generate these files. Individual calibration routines will write to the amplitude calibration and frequency calibration systems as needed. Calibration data will be archived by the file writer, but it may be easier to keep working calibration data in scratch files for easier access.

3.3 Synchronization

One backend, which may be a spectrometer, a specialized piece of equipment, or some other external set of signals, is the master for synchronization. Fast synchronization, for example to the chopping secondary, is through hardware signals to all spectrometers. Slower signals, such as nodding, may be either hardware or software through a `file_io` channel depending on the data acquisition hardware and data reduction pipelines for different observing modes. Synchronization for multiple processes will be by absolute time. For example, for an on-the-fly (OTF) map the observing task will specify an absolute time for the telescope to be at a given position, moving at a given speed and direction for some duration, and will also specify an absolute beginning time, time increment, and number of cycles for spectrometer readouts.