

# A User's Guide for the GBT Mapping Pipeline

---

*Jim Braatz (NRAO)  
June 19, 2012*

## Introduction

The GBT Data Processing Pipeline facilitates reducing spectroscopic data taken with the GBT. There are two primary software tools that constitute the pipeline: the first is called the “mapping pipeline” and it helps users create FITS image cubes from GBT mapping data. The second is the “spectral pipeline” and it helps users process single-pointing GBT spectra. At this time, the spectral pipeline is available only as a beta-release product. This document describes the mapping pipeline.

The mapping pipeline was developed initially to support mapping with the KFPA, but it can be applied to data taken with any of the receivers that use standard position-switched or frequency-switched observing modes. Pipeline tools can also help with reducing other types of data, such as W-band mapping observations.

To use the pipeline effectively, you should follow certain guidelines and observing procedures when taking your data. The observing steps are described here, are relatively simple, and are implemented in Astrid.

While this document addresses observing strategies that use the GBT spectrometer, the methods will be nearly identical for the new VEGAS spectrometer when it is introduced for science observations late in 2012.

## Getting Started

**Where to Run the Pipeline:** Currently, the GBT pipeline has certain dependencies and it runs only on 64-bit computers in Green Bank. Good choices for public computers in Green Bank from which to run the pipeline include *arcturus* and *euclid*.

**Disk Space:** The pipeline typically requires a significant allocation of disk space. Even small data sets use several gigabytes. So generally, you must make use of the Green Bank scratch areas to use the pipeline.

On arcturus: `/export/home/arcturus/scratch/user_id`  
On any GB computer: `/home/scratch/user_id`

In the above directories, replace “user\_id” with your linux user ID, e.g. “jbraatz”.

If you do not already have a directory in the GB scratch area, contact the GB helpdesk to have GB computing create a directory for you. The NRAO helpdesk is accessible at: <http://help.nrao.edu>

## A First Look at the Mapping Pipeline

You can run the pipeline using the command:

```
% gbtpipeline
```

For quick help, use the online help option:

```
% gbtpipeline -h
```

Here is a sample pipeline execution using KFPA position-switched test data that observed NH<sub>3</sub> toward W51:

```
% gbtpipeline -i /home/gbtpipeline/reference-data/TKFPA_29.raw.acs.fits  
-m 14:24 --refscan1=13 --units=Jy
```

This command maps data from scans 14-24, using scan 13 as the reference observation. The data are calibrated to units of Jy. The program creates three output files, plus a number of intermediary files. The important output files are:

W51\_23740\_MHz\_cont.fits – a single-channel “continuum” map

W51\_23740\_MHz\_cube.fits – a multi-channel data cube

W51\_23740\_MHz\_line.fits – a continuum-subtracted data cube

You can examine these data cubes using the casa viewer, e.g. :

```
% casaviewer W51_23740_MHz_line.fits
```

Select the menu option “Tools -> Spectral Profile”, then left-click on the crosshair icon, and left-click on the map. This will display a spectral cut through the cube.

## Basic Operation of the Pipeline

The input to the pipeline is an SDFITS data file generated from a GBT mapping observation, and from that file the pipeline generates an image cube. The pipeline operates in three basic stages.

1. The first stage calibrates the spectral line data and produces FITS files compatible with the GBTIDL “keep file” format of SDFITS.
2. The second stage uses the utility program *idlToSdfits* to convert these keep files into a FITS format that can be read by AIPS.
3. The third stage uses AIPS to grid these calibrated data and produce the output data cubes.

In typical use, a single call to the *gbtpipeline* command executes all three stages of operation. You can specify the range of scans to be mapped and identify the reference scans used for calibration (if the data are “position switched”), as in the NH<sub>3</sub> mapping example, above. A standard position-switched mapping observation begins with a reference scan, executes the mapping scans, and then finishes with another reference scan, all using a common spectrometer configuration. In the *gbtpipeline* command, you can specify a second reference scan in the parameter list, and in this case the two reference scans will be interpolated for each integration in the map, as in this example:

```
% gbtpipeline -i /home/gbtpipeline/reference-data/TKFPA_29.raw.acs.fits
-m 14:24 --refscan1=13 -refscan2=26
```

Here are the standard options available from the *gbtpipeline* command line:

## Pipeline Options

### Input file or project directory:

- i FILE, --infile FILE  
SDFITS file name containing map scans  
Either -i or -d must be specified
- d DIR, --sdfits-dir DIR  
project directory; used if -i option is not used  
If this option is set, the pipeline will first run the sdfits  
filler on the specified project and process the newly  
filled data

### data selection:

- m N1[N2], --map-scans N1[N2]  
range of scan numbers to map  
Either -m or --allmaps must be specified
- allmaps  
If set, attempt to process all maps in input file.
- refscan1 SCAN  
first reference scan. Must be specified if using -m
- refscan2 SCAN  
second reference scan (optional)
- c CHANNELS, --channels CHANNELS  
channel selection, i.e. 100:200

(default: process all channels)  
-f F1[,F2], --feed F1[,F2]  
    comma-separated feed(s) to process (default: all feeds)  
-p P1[,P2], --pol P1[,P2]  
    comma-separated polarization(s) to process (default: both)  
    Valid options: LL, RR, XX, YY

**control:**

--imaging-off  
    If set, the pipeline will calibrate the data and generate data files for input into AIPS, but will not create images.  
    (default: imaging is on)  
-a N, --average N  
    average the spectra over N channels (default: no averaging)  
-n N, --idlToSdfits-rms-flag N  
    flag integrations with noise in excess of the specified rms  
    See idlToSdfits for further details  
-w N, --idlToSdfits-baseline-subtract N  
    subtract median-filtered baseline  
    See idlToSdfits for further details  
--display-idlToSdfits  
    display idlToSdfits summary plots

**calibration:**

-u UNITS, --units UNITS  
    calibration units. Valid options are Ta, Ta\*, Tmb, Tb, Jy (default: Ta\*)  
--spillover-factor N rear spillover factor ( $\eta$ -l) (default: 0.99)  
--aperture-efficiency N  
    aperture efficiency (default: 0.71)  
--main-beam-efficiency N  
    main beam efficiency (default: 0.91)  
--gain-coefficients N1, N2, N3  
    comma-separated gain coefficients (default: .91,.00434,-5.22e-5)  
--map-scans-for-scale  
    use the mapping scans to calibrate Tsys  
    (default: use the reference scan to calibrate Tsys)  
--gain-factors-left G1[,G2, ...]  
    comma-separated gain factors for each left-polarized feed,  
    relative to feed 1. First entry must be 1.0.  
--gain-factors-right G[,G]  
    comma-separated gain factors for each right-polarized feed,  
    relative to feed 1. First entry must be 1.0.  
-t N, --zenith-opacity N  
    zenith opacity ( $\tau$ -z) (default: calculated from weather)  
--fs-as-ps      optionally process a FS map as PS

**output:**

-v N, --verbose N set the verbosity level-- 0-1:none, 2:errors only,  
3:+warnings, 4:+user info, 5:+debug (default: 0)  
--clobber Overwrite existing output files.

**other options:**

--max-processors N optional max number of processors, to reduce resource usage  
(default is selected based on available processors)

You can store input parameters in a file and read the command line parameters from that file, like this:

```
gbtpipeline @filename.par
```

Any options set on the command line will override whatever is read from the file.

## Advanced Operation of the Pipeline

Basic usage of the *gbtpipeline* command, as described above, should meet the data reduction needs of many projects. Sometimes, however, you may need to customize the mapping results or take advantage of certain features that are not available from the standard *gbtpipeline* parameters. In this case, you can run any of the three stages of the pipeline separately, customizing the results at each stage.

### Stage 1: Calibration

The first stage of the pipeline calibrates the data and creates an SDFITS file for each spectrometer sampler. As an alternative, you may use GBTIDL to produce the SDFITS files. GBTIDL gives you detailed control of the calibration and flagging, on an integration-by-integration basis. The SDFITS file must contain one record for each integration in the map, and each record should contain a calibrated spectrum. You can either create one SDFITS file with all the data, or multiple SDFITS files containing partial data sets (for example, separating feeds into separate SDFITS files for individual processing). Separate SDFITS files are easily combined for mapping in a later step.

### Stage 2: File Conversion

With an SDFITS file in hand, you can convert the file to an AIPS-compatible format using the utility program *idlToSdfits*. A typical execution will look like:

```
% idlToSdfits -l -o myfile.sdf myfile.fits
```

The “-l” option suppresses the creation of a plot file. The *idlToSdfits* program has a host of additional options that can be examined by typing “idlToSdfits” at the linux prompt, with no parameters.

The product of Stage 2 is one “sdf” file for each SDFITS file from Stage 1.

### Stage 3: Combining Data and Imaging

Prior to running Stage 3, you should have a set of “sdf” files. These can either be created as described in the above steps, or by running the *gbtpipeline* command, itself. You can specify the “--imaging-off” option of *gbtpipeline* to halt the process after creating the sdf files (the second stage). You can then run the third stage, with more control of the mapping procedure, manually. Here we describe that process.

**doImage Scripts:** The pipeline includes a utility program called *doImage* that runs ParselTongue scripts in AIPS. To use ParselTongue scripts, you need to know the AIPS ID used on your behalf by the pipeline. Get the ID from the linux prompt using:

```
% id -u
```

There are several *doImage* scripts in the standard *gbtpipeline* distribution, contributed mainly by Glen Langston. Two important scripts are:

1. *dbcon.py* : Combine data from multiple mapping procedures
2. *mapDefault.py* : Map data

So, you can execute the third stage of the pipeline, in its basic form, by executing the following two linux commands:

```
% doImage /home/gbtpipeline/release/contrib/dbcon.py <aips_id> <file1.sdf> [<file2.sdf>]
% doImage /home/gbtpipeline/release/contrib/mapDefault.py <aips_id>
```

The result of these commands is three FITS files, one with the full image data cube, one with a continuum map, and one with a line map.

**dbcon.py** The parameters to *dbcon.py* are:

```
dbcon.py <aipsNumber> <file1.sdf> [<file2.sdf>] [...]
```

where the first parameter is the AIPS ID and subsequent parameters are the sdf files to be combined in the final image. You can use one, two, or more than two files in the list. The “dbcon” step must be executed prior to the “mapDefault” step, even if only one sdf file is used.

**mapDefault.py** The parameters to *mapDefault.py* are:

```
mapDefault.py <aipsNumber> [<nAverage>] [<mapRaDeg>] [<mapDecDeg>] [<imageXPixels>]
[<imageYPixels>] [<refFreqMHz>]
```

- (1) [<nAverage>] spectral channels to average
- (2) [<mapRaDeg>] RA of map center in degrees
- (3) [<mapDecDeg>] Dec of map center in degrees

- (4) [`<imageXPixels>`] pixel size of the map in the X direction
- (5) [`<imageYPixels>`] pixel size of the map in the Y direction
- (6) [`<refFreqMHz>`] rest frequency in MHz (used to calculate the velocity scale)

If any one of the optional parameters in the list is specified, all the preceding parameters must also be specified.

## Examples Using *doImage*

Here we give examples of several common ways to use the *doImage* pipeline tool.

### Example: Combining Data from Multiple Maps

Suppose you have several data sets from independent executions of your GBT mapping observation. You can combine and map the data using the following procedure.

First run the pipeline on each of the maps separately, specifying the “`--imaging-off`” option. Each execution will generate a set of SDFITS files and sdf files.

```
gbtpipeline -i my_sample_data.fits -m 10:19 --referenceScan1=9 --imaging-off
gbtpipeline -i my_sample_data.fits -m 21:30 --referenceScan1=20 --imaging-off
```

Now combine the data using the *doImage* command with *dbcon.py*:

```
doImage /home/gbtpipeline/release/contrib/dbcon.py <aips_id> <file1.sdf> <file2.sdf>
```

Finally you can map the data using the *doImage* command with *mapDefault.py*:

```
doImage /home/gbtpipeline/release/contrib/mapDefault.py <aips_id>
```

### Example: Mapping Pre-calibrated Data

Under certain circumstances, *gbtpipeline* will be unable to calibrate GBT data. For example, the pipeline cannot currently calibrate W-band data, since observations with that receiver use a unique observing sequence for calibration. To map data such as this, you must first calibrate the data in GBTIDL and produce a “keep file” that has one record per beam, per integration in the map.

You can then grid the data with the following steps. First, convert the keep file to a FITS format that can be read by the AIPS FITS reader:

```
% idlToSdfits -o my_sdf_file.sdf my_keep_file.fits
```

Next, load the data into AIPS with the *dbcon.py* script:

```
% doImage /home/gbtpipeline/release/contrib/dbcon.py <aipsid> my_sdf_file.sdf
```

And finally map the data with the *mapDefault.py* script:

```
% doImage /home/gbtpipeline/release/contrib/mapDefault.py <aipsid>
```

### Example: Line Maps and Temperature from NH<sub>3</sub>(1,1) and NH<sub>3</sub>(2,2)

Here we give an example of using *doImage* scripts to generate line maps and a temperature map from the NH<sub>3</sub> data. We make line maps by restricting the spectral channels used in the map to only those near the NH<sub>3</sub> line centers. Glen Langston contributed the scripts.

To generate the maps, you can run these exact commands using the pipeline in Green Bank:

```
gbtpipeline -i /home/gbtpipeline/reference-data/TKFPA_29.raw.acs.fits -m 14:24
--refscan1=13 -a 3 -v 4

doImage /home/gbtpipeline/release/contrib/sumLine.py 13482 W51-NH3-11 23694.506 60. 20.
doImage /home/gbtpipeline/release/contrib/sumLine.py 13482 W51-NH3-22 23722.6336 60. 20.
doImage /home/gbtpipeline/release/contrib/tempNH3_1122.py 13482 W51 60. 10. .2
```

The first command is the pipeline mapping command we encountered already in our initial example. The second and third commands use the *sumLine.py* script to generate single-channel maps averaged through the line centers from the NH<sub>3</sub>(1,1) and NH<sub>3</sub>(2,2) lines. The parameters of the *sumLine.py* procedure are:

*sumLine.py* <aipsNumber> <fileName> <restFreq> <lineVel> <lineWid>

- (1) <aipsNumber> : the AIPS ID
- (2) <fileName> : a name for the output file
- (3) <restFreq> : the rest frequency of the line in MHz
- (4) <lineVel> : the line center velocity in km/s, and
- (5) <lineWid> : the line FWHM in km/s.

The fourth command in the sequence generates the temperature map from the original line cube using the *temp\_NH3\_1122.py* script. The parameters for this command are:

*tempNH3\_1122.py* <aipsNumber> <fileName> <restFreq> <lineVel> <cutOff>

- (1) <aipsNumber> : the AIPS ID
- (2) <filename> : a name for the output file
- (3) <lineVel> : the line center velocity in km/s
- (4) <lineWid> : the line width (FWHM) to average, and
- (5) <cutOff> : the minimum line intensity to use in the line ratio (smaller values result in blanked pixels in the temperature map)



These scripts are available in the *contrib* directory. Besides being useful in their own right, they also serve as good examples to help you develop your own scripts.

## Configuring Your GBT Observation to Use the *allmaps* Option

Previous examples demonstrated how you could identify specific mapping scans and reference scans for the pipeline's calibration sequence. By specifying mapping and reference scans explicitly in this manner, you have direct control over the calibration process. An alternative is to allow the pipeline to determine mapping and reference scans automatically. If you do not specify scan ranges at the command line, the pipeline will examine the input data file and attempt to process each mapping observation it encounters there. To identify mapping scans and reference scans, the pipeline uses header information that must be set explicitly during the observation. The *KFPA Data Reduction Guide*, by Langston et al., describes this process. (<https://safe.nrao.edu/wiki/bin/view/Kbandfpa/KfpaReduction>).

The following sample astrid script demonstrates a KFPA observing sequence with keywords set so that the pipeline can identify reference scans properly.

```
# Example Astrid script demonstrating KFPA Mapping for use with pipeline "allmaps"
# We assume that the telescope has been configured prior to running this script

# Identify the source catalog
Catalog("/home/userid/mycatalog.cat")

# Define procedures with scan annotations and header values
execfile("/home/astro-util/projects/TKFPA/kfpaMapInit")

target = "W51" # define mapping target
off = "W51-Off" # define a map reference location, with no emission

Slew(target)

Balance()
# Check the levels are correct
Break("Check Balance")

#perform a total power observation at the reference location
OffTrack( off, None, 30.0, "1")

#Tell Pipeline that mapping is starting
SetValues("ScanCoordinator",{"scanId":"Map"})
RALongMap( target,
    Offset("Galactic", 0.33, 0.0), # This is a galactic coordinate map
    Offset("Galactic", 0.0, 0.10),
    Offset("Galactic", 0.0, 0.008),
    140.0, "1")

#perform the final total power reference obs
OffTrack( off, None, 30.0, "1")
```

The pipeline can then process the resulting data, correctly associating the reference scans with the mapping data. The pipeline command is:

```
gbtpipeline -i my_sample_data.fits --allmaps
```

## Subtracting Spectral Baselines

A typical GBT mapping observation uses short integration times per pixel (a few seconds), and the resulting spectral baselines are flat compared to the thermal noise. The pipeline therefore uses a very basic spectral baseline subtraction, implemented with the AIPS task *IMLIN* in the *mapDefault.py* script. The default behavior is to fit a DC baseline (polynomial of order 0) determined from the edges of the spectrum. Specifically, the baseline is fit to channels  $[0.04, 0.12] * nchan$  and  $[0.81, 0.89] * nchan$ . You can modify this behavior by editing the *mapDefault.py* script directly. Follow these steps:

1. cd to the directory with your mapping data
2. copy `/home/gbtpipeline/release/contrib/mapDefault.py` to the current directory
3. edit `mapDefault.py` appropriately (search for the “inline” command)
4. re-run the pipeline with the “--imaging-off” option
5. run “doImage dbcon <aipsid> <file.sdf>”
6. run “doImage mapDefault.py <aipsid>” (note we are using the local copy of `mapDefault.py`)

A future upgrade to the pipeline will incorporate baseline-fitting parameters directly in the *gbtpipeline* command line.

For more detailed control over the baseline fitting and other aspects of the calibration and flagging, you can use GBIDL to calibrate the data and use pipeline tools for “Stages” 2 and 3.

## Cleaning Up Disk Space

The pipeline creates large data files, and may completely fill the available disk space if care is not taken to clean up unused files. You should remove all intermediate files that are not needed for additional processing. This includes “sdf” files generated during the calibration of data, and AIPS files used in intermediate stages of gridding the data. A utility is available to clean up old AIPS files, and this should be run after each use of the pipeline.

```
% doImage /home/gbtpipeline/release/contrib/clear_AIPS_catalog.py <aips_id>
```

Recall that you can find your AIPS ID by the linux command “`id -u`”.