AIPS++ User Reference Manual - ATNF Package

AIPS++ Group, eds

August 1, 2006

Copyright ©1995-1999 AIPS++ Consortium
AIPS++ User Reference Manual - ATNF Package.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the AIPS++ Consortium.

The AIPS++ consortium may be reached by email at aips2-request@nrao.edu. The postal address is: AIPS++ Consortium, c/o NRAO, 520 Edgemont Rd., Charlottesville, Va. 22903-2475 USA.

Contents

1 ATNF **5**
1.1 atca - Module 5
1.1.1 atcafiller - Tool 5
1.2 multibeam - Module 11
1.3 rfi - Module 14

Chapter 1

ATNF

ATNF-related modules and tools

Description
Package for ATNF-related functionality.

1.1 atca - Module

Module for Australia Telescope Compact Array data processing

Description

This module contains `tools` that are targeted at processing data from the Australia Telescope Compact Array (ATCA). AIPS++ includes many other generic `tools` for processing ATCA data. The tools described below are specific to the ATCA.

Tools

atcafiller	A tool for converting ATNF/ATCA RPFITS files to a MeasurementSet

1.1.1 atcafiller - Tool

Package atnf
Module atca

A tool for converting ATNF/ATCA RPFITS files to a MeasurementSet
include "atcafiller.g"

Description
The `atcafiller` tool is used to convert ATNF/ATCA RPFITS files to AIPS++ MeasurementSets.

Constructors

atcafiller	Construct the ATCA filler tool

Functions

fill	Fill the data
select	Select the data to fill

atcafiller - Constructor

Package	**atnf**
Module	**atca**
Tool	**atcafiller**

Construct the ATCA filler tool

Synopsis
atcafiller(msname, filenames, options, shadow)

Arguments

msname	Filename for MeasurementSet to create Allowed:String Default: ”
filenames	Existing RPFITS files to read; wild cards accepted, don’t use commas Allowed:String or Vector of Strings Default: ”
options	birdie, reweight, noxycorr,compress,fastmosaic,hires Allowed:Vector of Strings Default: ”
shadow	dish size for flagging shadowed data Allowed:Float Default: 22.0

Returns
atcafiller tool

Description
This constructor just creates the filler tool, it does not do any actual filling yet. In the filenames argument commas are not allowed because of the wildcard capability.
The options argument allows for special processing options during the filling process.

birdie remove self generated birdies in 128 MHz data by deleting every other channel and the edge channels

reweight suppress ringing of interference spikes by reweighting the lag spectrum for minimum spectral sidelobe response

noxycorr do not apply the xyphase correction using the noise cal xy phase measurements (use if you know there was a problem with the measurements)

compress Compress the data, model, corrected data and imaging weight columns (may reduce dynamic range)

fastmosaic Produce a MeasurementSet with small tiles to optimize I/O for observations with a large number of pointings and a short time (a few integrations) on each pointing. Failing to use this options for such data may cause very inefficient access, especially for continuum data (10-100 times slower)

hires For data taken in binning mode: set the TIME_CENTROID column to the center of each bin (normally center of integration interval). The TIME column is unaffected. TIME_CENTROID will no longer be in time order for dual frequency mode.

Set the shadow parameter to zero to keep shadowed data. Set it slightly larger than the dish size (22m) if you don't trust data that is almost (geometrically) shadowed.

Example

```
- rpf := atcafiller('myms',
  "/data/RPFITS/1999-01-01_0101.C999 /data/RPFITS/1999-01-01_0202.C999",
  "birdie reweight", 22.5, F);
```

---

atcafiller.fill - Function

Package	**atnf**
Module	**atca**
Tool	**atcafiller**

Fill the data

Synopsis
fill()

Description

This function does the actual filling of the data from rpfits format files to a MeasurementSet. After the completion of the fill operation you should call the done method to avoid problems with incomplete flushing of the data to disk. You may want to run select before fill, to reduce the amount of data you read in.

Returns
Bool

Example
Fill the data from files 1999-01-01_0101.C999 and 1999-01-01_0202.C999 in the directory /data/RPFITS into the MeasurementSet myms.

```
- rpf := atcafiller('myms',
  "/data/RPFITS/1999-01-01_0101.C999 /data/RPFITS/1999-01-01_0202.C999");
- rpf.fill();
- rpf.done();
- myms:=ms('myms');
- myms.nrow();
1204
- myms.close();
```

---

atcafiller.select - Function

Package	**atnf**
Module	**atca**
Tool	**atcafiller**

Select the data to fill

Synopsis
select(firstscan, lastscan, lowfreq, highfreq, freqchain, fields, bandwidth1,

numchan1)

Arguments

firstscan	first scan to read Allowed:Positive Int Default: 1
lastscan	last scan to read Allowed:Positive Int Default: 9999
lowfreq	lowest reference freq to select in GHz Allowed:Double Default: 0.1
highfreq	highest reference freq to select in GHz Allowed:Double Default: 1000.0
freqchain	select one of the simultaneous frequencies, 0=both Allowed:Int Default: 0
fields	list of field names (sources) to select Allowed:Vector of Strings Default: ”
bandwidth1	select on bandwidth (MHz) of the first frequency chain, 0=all Allowed:Int Default: 0
numchan1	select on number of channels in first frequency chain, 0=all Allowed:Int Default: 0

Description

Select on various items, to load only a subset of the data present in the input file. The default is to load everything. Note that the frequency selection using lowfreq and highfreq acts purely on center (reference) frequencies, channel selection is not available at this stage. Use freqchain to select the frequency chain (or IF) to load, specify either 1 or 2, the default of zero will load both. Use the fields parameter to specify a list of field names (sources) you want to load. For detailed selection of complex observations the last two parameters let you specify the required bandwidth (in whole MHz) and number of channels of the first frequency chain.

Returns

Bool

Example
Select scans 4 to 10 inclusive, frequency chain 1, all data with a reference frequency of more than 1.4 GHz.

```
- rpf := atcafiller('myms', '/data/RPFITS/1999-01-01_0101.C999');
- rpf.select(firstscan=4, lastscan=10, freqchain=1, lowfreq=1.4)
- rpf.fill();
- rpf.done();
```

1.2 multibeam - Module

Module for Multibeam receiver data processing

Description

This module pertains to the processing of Multibeam data.
`Livedata`, the Parkes Multibeam data reduction task, directs and regulates the flow of Multibeam data between the following six independent clients:

```
Reader ->
          Bandpass calibrator
                               -> Data monitor
                               -> Writer
                                              -> Statistics
                                              -> Gridder
```

Data flow is from left to right. All clients other than the reader may be disabled or short-circuited, i.e. any line(s) of the above may be deleted other than the first except that the statistics calculator and gridder are dependent on the writer since they process its output file. However, once started, the gridder may remain active even after the writer has been disabled since it is fed via a special purpose queue.
`Livedata` is started by issuing the command

```
% glish -l 'livedata.g'
```

This starts up the **Livedata** GUI from which operations are controlled. The GUI has popuphelp attached. Consult with your local Multibeam expert for details on the reduction process as required.

- Reader: This client reads data from either an MBFITS format file (the single-dish variant of RPFITS), or an AIPS++ Measurement Set. The input data, which may be selected on a beam-by-beam basis, is packaged as a *Glish* record for the next client in the chain.
- Bandpass calibrator: This client does most of the work in calibrating Multibeam data. It operates in several modes depending on the method of observation:
  - Scanning modes as used in HIPASS and ZOA surveys.
  - Frequency switching modes.
  - Position switching modes where each element of the Multibeam system is pointed at the source in turn.

  - Extended source modes such as used for High Velocity Clouds.

    Each of these modes uses a separate bandpass calibration strategy. These are based on robust statistical estimators, particularly median estimators, which allow rejection of RFI without human intervention.

- Data monitor: The data monitor interfaces to **MultibeamView**, a specially modified version of the Karma Karma package **kview** application. In fact, it invokes **MultibeamView** twice, once for each polarization, to provide two panels displaying frequency versus time with various image enhancement options. This provides visual inspection for each pair of polarizations of the 13 beams, one pair at a time.

- Writer: Data is written to an AIPS++ Measurement Set by the writer client. These measurementsets are converted to SDFITS format for data archiving by a separate offline task.

- Statistics: The statistics client reads through an AIPS++ Measurement Set and computes basic statistical measures for each beam and polarization.

- Gridder: Each Multibeam integration consists of a spectrum taken at a particular point on the sky. The gridder takes the spectra from a collection of bandpass-calibrated measurementsets and interpolates them onto a regular map grid, writing the result to a FITS data cube as it goes. Typically each point in the sky is sampled many times in separate observations and the gridder combines these using robust (median) statistics to automatically eliminate Radio Frequency Interference.

  Being the most computationally intensive client, the gridder is usually used separately for offline data reduction. However, it can be linked into the Livedata reduction chain, though, unlike the other clients, it may run remotely on another machine in a quasi-batch mode.

`LiveData` operates in near-realtime at the Parkes telescope to allow immediate assessment of data quality. However, it may also be used for offline data processing.
Each client is implemented as a C++ program with a separate controlling *Glish* script client (subsequence) which provides an optional GUI. The six script clients are coordinated by another script client which directs and regulates the flow of data between them by means of *Glish* events. This "reducer", the `LiveData` control hub, also maintains the input queue for the gridder.
An additional "scheduler" client is added for interactive use. It provides a queueing mechanism which supplies input to the data reduction pipeline. It provides LiveData's near-realtime capability by automatically discovering MBFITS files as they are written by the Multibeam correlator.

LiveData may be run in batch mode by invoking the reducer directly (usually from a *Glish* script) thus bypassing the scheduler (discuss this with your local Multibeam data processing expert).

1.3 rfi - Module

Radio Frequency Interference

Description
This module will contain functionality to handle RFI. It is as yet empty.

Index

atca, 5
atca.atcafiller, 5, 6
atca.atcafiller.fill, 7
atca.atcafiller.select, 8
atcafiller, 5, 6

fill, 7

multibeam, 11

rfi, 14

select, 8