AIPS++ User Reference Manual - ALMA Package

AIPS++ Group, eds

August 1, 2006

Copyright ©1995-1999 AIPS++ Consortium
AIPS++ User Reference Manual - ALMA 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 ALMA **5**
1.1 alma - Module 5
1.1.1 iramcalibrater - Tool 6
1.1.2 resample - Function 17
1.1.3 shadow - Function 18
1.1.4 almati2ms - Tool 20
1.1.5 almatifiller - Function 24

Chapter 1

ALMA

ALMA-related modules and tools

Description
Package for ALMA-related functionality.

1.1 alma - Module

Module for clic emulation functionalities
include "iramcalibrater.g"

Description

`iramcalibrater` provides the functionalities and algorithm implementation to calibrate millimetric interferometer data as done by GILDAS.

iramcalibrater tool

This tool provides some of the functionalities of IRAM's calibration package (CLIC) for data from Plateau de Bure observatory. Some of the algorithms as used by IRAM's CLIC have been re-implented here.

Tools

almati2ms	A tool for converting ALMA-TI format data to a measurement set
iramcalibrater	tool to perform calibration on data (a la GILDAS)

Functions

almatifiller	Convert ALMA-TI fits files to an MS

resample	Does a simple 3 point Hanning on spectral MS
shadow	Check and flag for antenna shadowing

1.1.1 iramcalibrater - Tool
Package alma
Module alma

tool to perform calibration on data (a la GILDAS)

Description

Overview of `iramcalibrater` tool functions:

Functions that have been implemented in the `iramcalibrater` tool are:

Data selection phcor

Bandpass calibration rf

Phase calibration phase

Flux calibration flux

Amplitude calibration amp

UV data extraction/combination uvt

Example
The following example illustrates a typical iramcalibrater tool session

```
# Include the iramcalibrater functionality
include 'iramcalibrater.g'

# Create a iramcalibrater tool
myic:=iramcalibrater(msname='07feb97-g067.ms');

# Do a selection of the better data between the online radiometric
# corrected data and the non-corrected data
myic.phcor(trial=F);

# Perform the  bandpass calibration
myic.rf(fieldname='0528+134', freqgrp='3mm-LSB', visnorm=F, bpnorm=T,
```

```
          gibb= 2, drop=0);

# Perform the phase calibration
myic.phase(fieldnames=['0528+134', '0415+379'], freqgrp='3mm-LSB');

# Perform the flux calibration
myic.flux(fieldnames=['0528+134', '0415+379', 'MWC349'], freqgrp='3mm-LSB', plot=T);

# Perform the amplitude calibration
myic.amp(fieldnames=['0528+134', '0415+379'], freqgrp='3mm-LSB')

# Extract the calibrated target source and append it to another ms
myic.uvt(fieldname='GG_TAU', spwid=7, filename='ggtau3mm.ms', option='append');
```

The `iramcalibrater` tool needs to be constructed with an associated ms.

Constructors

iramcalibrater	Construct an iramcalibrater tool

Functions

amp	perform amplitude calibration on the data
done	Close the iramcalibrater tool
flux	perform flux calibration on the data
phase	perform phase calibration on the data
phcor	Select data atmospheric corrected data or not
rf	perform phase and amplitude bandpass calibration
uvt	to extract calibrated and combine it into another ms

iramcalibrater - Constructor

Package	**alma**
Module	**alma**
Tool	**iramcalibrater**

Construct an iramcalibrater tool

Synopsis
iramcalibrater(ms, initcal)

Description
This is used to construct a `iramcalibrater` tool. This tool can then be used

to perform the different calibration routines needed and/or extract a given source and spectral window uv-data for combination with other days of previously calibrated data.
If restarting a calibration of existing data, set `initcal=T` to initialize the scratch columns of the dataset.
It can be used as `iramcalibrater('mydata.ms')` to construct the iramcalibrater tool associated with the given data
This returns a Glish variable containing the tool functions of `iramcalibrater`.

Arguments

ms	Measurementset to calibrate Allowed:String Default: ”
initcal	Initialize calibration? Allowed:Boolean Default: 'T'

Returns
iramcalibrater tool

Example

```
myic:=iramcalibrater(ms='10feb97-g067.ms',initcal=T) ;
```

---

iramcalibrater.phcor - Function

Package **alma**
Module **alma**
Tool **iramcalibrater**

Select data atmospheric corrected data or not

Synopsis
phcor(trial)

Description

This function examines the calibrators and compares the amplitude of the atmospheric/radiometric-corrected data with that of the non-corrected data and then performs a per scan and per antenna selection on all the data which is going to be used subsequently. This needs to be run once, and is the first thing that is needed on a new measurement set.

Arguments

trial	Execute in trial mode or not Allowed:Bool Default: T

Returns

Example

```
myic:=iramcalibrater('10feb97-g067.ms')
myic.phcor(F)
```

This function prints out the scans for which the atmospheric correction will be rejected

---

iramcalibrater.rf - Function

Package **alma**
Module **alma**
Tool **iramcalibrater**

perform phase and amplitude bandpass calibration

Synopsis
rf(fieldname, freqgrp, visnorm, bpnorm, refant, gibb, drop, degamp, degphase)

Description

This function will perform a bandpass calibration on the selected data. The solutions are antenna based polynomials. The desired degree for the amplitude and phase polynomials may be specified in the `degamp` and `degphase`

parameters. The defaults are usually adequate here, but observations with unusually wide bandwidths might benefit from user input here.
Plots of the solutions are returned to the disk as a postscript file with a name consistent with the ms file name and the bands/sidebands solved for.

Arguments

fieldname	Field name of (single) calibrator to be used Allowed:String
freqgrp	spectral windows to be used in the bandpass calibration Allowed:'3mm-LSB' —— '3mm-USB' —— '1mm' —— '1mm-LSB' —— '1mm-USB' Default: '3mm-LSB'
visnorm	Do an auto calibration in time before integrating Allowed:Bool Default: F
bpnorm	normalize bandpass solution Allowed:Bool Default: T
refant	Reference antenna Allowed:Int Default: 1
gibb	number of central channels to reject in solve Allowed:Int Default: 2
drop	percentage of channels to ignore on edges of band Allowed:Int Default: 5
degamp	Degree of amplitude polynomial in fit Allowed:Int Default: 6
degphase	Degree of phase polynomial in fit Allowed:Int Default: 12

Returns
Bool

Example

```
myic:=iramcalibrater('10feb97-g067.ms')
myic.phcor()
```

```
myic.rf(fieldname='0528+134', refant=1, freqgrp='3mm-LSB')
myic.done()
```

---

iramcalibrater.phase - Function

Package	**alma**
Module	**alma**
Tool	**iramcalibrater**

perform phase calibration on the data

Synopsis
phase(fieldnames, freqgrp, refant, phasetransfer, rawspw, npointaver, phasewrap)

Description
This function performs a phase calibration using the selected calibration sources and spectral window ids. Previously bandpassed calibrated data, averaged in the spectral domain, are used to find the phase solution. The solutions are antenna based cubic splines.

A plot of the solution is returned to the disk as a postscript file with a name consistent with the ms file name and bands/sidebands solved for.

For less-than-well-behaved data with phasewraps, the `npointaver` and `phasewrap` parameters may be used to tune the phase tracking algorithm. `npointaver` sets the number of consecutive time points that are averaged for comparison with the next point to see if it has slipped a cycle. `phasewrap` is the phase difference required in the comparison to indicate a cycle slip. When a cycle slip is detected, the current point and all future points will be shifted by $+/-360$ before the solution. Such slip corrections will accumulate for every detected slip. For well-behaved data, the defaults should be adequate.

Returns
Bool

Arguments

fieldnames	Field names of calibrators to be used Allowed: Vector of Strings
freqgrp	frequency group to solve for phase Allowed: '3mm-LSB' —— '3mm-USB' —— '1mm' —— '1mm-LSB' —— '1mm-USB' Default: '3mm-LSB'
refant	Reference antenna Allowed: Int Default: 1
phasetransfer	type of transfer for 1mm solution Allowed: 'raw' —— 'curve' —— 'none' Default: 'raw'
rawspw	spectral window used in case of raw phase transfer Allowed: Int Default: -1
npointaver	Number of consecutive time points averaged for phase tracking Allowed: Int Default: 10
phasewrap	Phase difference that indicates a cycle slip in phase tracking Allowed: Quantity Default: '250deg'

Example

```
myic:=iramcalibrater('10feb97-g067.ms')
myic.phcor(F)
myic.rf(fieldname='0528+134', freqgrp='3mm-LSB')
myic.phase(fieldnames=['0528+134', '0415+379'], freqgrp='3mm-LSB')
myic.rf(fieldname='0528+134', freqgrp='1mm')
myic.phase(fieldnames=['0528+134', '0415+379'], freqgrp='1mm',
phasetransfer='raw', rawspw=3)
myic.done()
```

In the example above we used 0528+134 to do the bandpass calibration and 0528+134 and 0415+379 to do the phase calibration at 3mm. Then the bandpass is done on the 1mm (both side bands) with a phasetranfer from the spw=3 as raw phase reference at 3mm.

---

iramcalibrater.flux - Function

Package **alma**
Module **alma**
Tool **iramcalibrater**

perform flux calibration on the data

Synopsis
flux(fieldnames, fixed, freqgrp, timerange, plot, gibb, drop, numchancont)

Description

This function performs a flux density scale calibration using the selected calibration sources and spectral windows. Previously bandpassed and phase calibrated data are used to find the flux calibration needed. If `fixed` is unspecified, the function will execute in interactive mode, and the user will be queried for names and flux densities of known calibrators (at least one of the fields specified in `fieldnames` must be specified). The flux densities of the remaining fields and the efficiencies of the antennas will be derived and reported. In this mode, the user can iteratively change the flux density(ies) of the fixed-flux density calibrators to achieve the efficiencies expected.
If `fixed` is specified with any flux densities, this function will run in NON-interactive mode, and the user will have no opportunity to iterate as described above. This mode is useful if flux densities are confidently known a priori. In `fixed`, specify flux densities as a vector of string quantities (i.e., with units, e.g., '5Jy', or "5Jy 3.4Jy"). The values will be matched one-for-one with the field names in `fieldnames`. If only a subset of fields have known flux densities which the user desires fixed, list these fields first in `fieldnames`, or specify '-1Jy' for the flux densities of fields that should be derived. This mode assumes that the user is confident in the specification of known flux densities, and the fixed and derived flux densities/efficiencies will be automatically applied.
Timerange(s) may be specified to limit the range of data used in the calculation by using the `timerange` parameter. Specify these as pairs of times in a string. The format for the times can be: YYYY/MM/DD/hh:mm:ss, hh:mm:ss, hh:mm, or HH.xxh. In the latter two cases, the date will be assumed from the dataset. Note that the trailing 'h' in the third option specifies units of hours (e.g., 19:30:00 is the same as 19.5h ; in fact any appropriate time quantity (as string) may be specified). The number of time values specified must be even, with each pair representing a different timerange.
Use the `numchancont` parameter to specify (if necessary) the number of channels in the spectral window(s) that should be used for determining the flux scale.

Returns
A list of flux densities in Jy.

Arguments

fieldnames	Field names of calibrators to be used Allowed: Vector of Strings
fixed	Fixed flux densities for known calibrators (as string quantities) Allowed: Vector of Strings
freqgrp	spectral windows to be used in the bandpass calibration Allowed: '3mm-LSB' —— '3mm-USB' —— '1mm' —— '1mm-LSB' —— '1mm-USB' Default: '3mm-LSB'
timerange	Timerange(s) specified in pairs Allowed: String Default: ' '
plot	Display flux renormalized amplitudes Allowed: Bool Default: F
gibb	number of central channels to reject in solution Allowed: Int Default: 2
drop	percentage of channels to ignore on edges of band Allowed: Int Default: 5
numchancont	Number of channels in continuum spectral window Allowed: Int Default: 64

Example

```
myic:=iramcalibrater('10feb97-g067.ms')
myic.phcor(F)
myic.rf(fieldname='0528+134', freqgrp='3mm-LSB')
myic.phase(fieldnames=['0528+134', '0415+379'], freqgrp='3mm-LSB')
myic.flux(fieldnames=['0528+134', '0415+379', 'MWC349'],
          fixed=['2.92Jy','6.25Jy'],
          freqgrp='3mm-LSB',
          timerange='18:30 19:00 21.25h 21.5h',
          plot=T)
myic.done()
```

In the example, data for three fields in the 3mm lower sideband during the time intervals 18:30-19:00 and and 21:15-21:30 are used in the flux density scale calibration, and only the first two fields have fixed flux densities specified (the third will be derived).

iramcalibrater.amp - Function

Package	**alma**
Module	**alma**
Tool	**iramcalibrater**

perform amplitude calibration on the data

Synopsis
amp(fieldnames, freqgrp)

Description
This function performs a phase calibration using the selected calibration sources and spectral window ids. Previously bandpass, phase and flux calibrated data are averaged in the spectral domain, are used to find the amplitude solution. The solutions are antenna based cubic splines. A plot of the solution is returned to the disk as a postscript file with a name consistent with the ms file.

Returns
Bool

Arguments

fieldnames	Field names of calibrators to be used Allowed:Vector of Strings
freqgrp	frequency group to use Allowed:'3mm-LSB' —— '3mm-USB' —— '1mm' —— '1mm-LSB' —— '1mm-USB' Default: '3mm-LSB'

Example

```
myic:=iramcalibrater('mynew.ms')
myic.phcor(F);
myic.rf(fieldname='0528+134', freqgrp='3mm-LSB')
myic.phase(fieldnames=['0528+134', '0415+379'], freqgrp='3mm-LSB')
myic.flux(fieldnames=['0528+134', '0415+379', 'MWC349'], freqgrp='3mm-LSB', plot=T)
myic.amp(fieldnames=['0528+134', '0415+379'], freqgrp='3mm-LSB')
myic.done()
```

In the example above we used 0528+134 to do the bandpass calibration and fields 0528+134 and 0415+379 to do the phase and amplitude calibration, while using '0528+134', '0415+379', 'MWC349' for flux.

---

iramcalibrater.uvt - Function

Package	**alma**
Module	**alma**
Tool	**iramcalibrater**

to extract calibrated and combine it into another ms

Synopsis
uvt(fielname, spwid, filename, option)

Description
Usually once the calibration of several days of data has been performed, the target source is to be extracted and combined into a single ms for imaging. the `uvt` function is used to perform this task.

Returns
Bool

Arguments

fielname	Name of source to be extracted Allowed:String
spwid	spectral window for which data is to be extracted Allowed:Int Default: 1
filename	name of new ms into which data is to written Allowed:String
option	'new' ms or 'append' to an existing one Allowed:String Default: 'new'

Example

```
myic:=iramcalibrater('10feb97-g067.ms')
myic.phcor(F)
myic.rf(fieldname='0528+134', freqgrp='3mm-LSB')
myic.phase(fieldnames=['0528+134', '0415+379'], freqgrp='3mm-LSB')
myic.flux(fieldnames=['0528+134', '0415+379', 'MWC349'], freqgrp='3mm-LSB', plot=T)
myic.amp(fieldnames=['0528+134', '0415+379'], freqgrp='3mm-LSB')
myic.uvt(fieldname='GG_TAU', spwid=7, filename='ggtau3mm.ms', option='append')
myic.done()
```

In the above example after the standard calibration, data from the spectral window 3 for the field GG_TAU is extracted.

---

iramcalibrater.done - Function

Package	**alma**
Module	**alma**
Tool	**iramcalibrater**

Close the iramcalibrater tool

Synopsis
done()

Description
This is used to totally close the `iramcalibrater` tool.

Returns
Bool

Example

```
myic:=iramcalibrater('10feb97-g067.ms')
myic.phcor(F)
myic.done()
```

1.1.2 resample - Function

Package	**alma**
Module	**alma**

Does a simple 3 point Hanning on spectral MS

Synopsis
resample(infile, outfile)

Description

This is a simple function that just does a Hanning smoothing on the channels in the MS. The weighting scheme is 0.125(chan-1)+ 0.75chan + 0.125(chan+1).

Returns
Bool

Arguments

infile	Name of input ms Allowed:String Default: No Default
outfile	Name of output ms, " output in the input Allowed:String Default: "

Example

```
resample('ggtau_1mm_256chan.ms', 'ggtau_1mm_256chan_hann.ms');
```

1.1.3 shadow - Function
Package alma
Module alma

Check and flag for antenna shadowing

Synopsis
shadow(msname, trial, minsep)

Description

This function is run to check and flag data when antennas are shadowed. If run with argument trial=T, then it does not flag but just report the scans that are shadowed. The argument minsep is the minimum projected seperation acceptable. No shadowing occurs if seperation is more than antenna diameter.

Returns

Bool

Arguments

msname	Name of input ms Allowed:String Default: No Default
trial	trial run, if T, no flagging Allowed:Bool Default: T
minsep	Minimum acceptable seperation Allowed:Double Default: 14.0

Example

```
shadow('10feb97-g067.ms', trial=F, minsep=13.0);
```

In the above example, flag all data for which any antenna is shadowed by any other antenna which is 13m or closer in projection.

1.1.4 almati2ms - Tool
Package alma
Module alma

A tool for converting ALMA-TI format data to a measurement set
include "almati2ms.g"

Description
The almati2ms tool contains functions to convert data stored in the ALMA-TI external format into an AIPS++ MeasurementSet (MS). The ALMA-TI format is described in ALMA Computing Memo. No. 15 (Lucas and Glendenning 2001)
The almati2ms tool is most easily run using the global function almatifiller. More fine-grained control can be achieved by creating an individual almati2ms tool and setting the functions on the tool directly. This may be more useful in some cases, especially scipring. The tool functions fall into the following categories:

1. Create an almati2ms tool
2. Set general filler options
3. Select which data are to be filled
4. Start the filler
5. Remove the tool

Creating a almati2ms tool
The only way to create a almati2ms tool is to use the almati2ms constructor. Calling this function will start a precompiled executable, called almati2ms, usually running on your machine. This executable executes all the almati2ms functions. The almati2ms constructor requires specification of the output MS file name, the input ALMA-TI FITS file and a flag to indicate if the new FITS data are to be appended to or overwrite an existing MS.

Set general filler options
The general filler options are specified using the function setoptions. These parameters control whether the output MS data are to be written in compressed or uncompressed mode and whether the sub-band data are to be combined per sideband into a common spectral window.

Select which data are to be filled
The select function is provided to allow selection of a subset of the input ALMA-TI data. At present the options are provided to select an observing

mode (as stored in the ALMA-TI DATAPAR table) and which particular set of channel-zero data are to be filled.

Start the filler
Once an almati2ms tool has been created and the selection and general filler options set, the data can be filled by running the fill function.

Removing the tool
When you have finished with the almati2ms you should shut down the process, using the done function. Once a tool is shut down it can no longer be used. This also closes the output measurement set.

Example
The following example shows the minimum necessary to run the almati2ms tool.
The first line makes all the definitions for this tool available to the user. The second line calls the constructor and specifies the input and output files. The third sets a general option of no compression of the output file. The fourth states that only observing mode CORR and the time-averaged channel-zero continuum data should be filled. The fifth line starts the copying process and the sixth line closes the tool.

```
include 'almati2ms.g'                                                  #1
a := almati2ms(msfile='output.ms', fitsin='ALMATI.fits', append=F) #2
a.setoptions(compress=F)                                               #3
a.select(obsmode='CORR', chanzero='TIME_AVG')                          #4
a.fill()                                                               #5
a.done()                                                               #6
```

This example could also have been achieved direclty using the almatifiller global function. This global function will execute all almati2ms tool functions in the sequence indicated above and, in addition, allows multiple input ALMA-TI files to be specified using a directory name and file modifier pattern. This is illustrated below:

```
include 'almati2ms.g'                                                  #1
almatifiller(msfile='output.ms', fitsdir='.', pattern='ALMATI*",
             append=F, compress=F, obsmode='CORR',
             chanzero='TIME_AVG', dophcor=F)                                   #2
```

Constructors

almati2ms Construct an almati2ms tool.

Functions

select	Input data selection
setoptions	Set general filler options

almati2ms - Constructor

Package	**alma**
Module	**alma**
Tool	**almati2ms**

Construct an almati2ms tool.

Synopsis
almati2ms(msfile, fitsin, append, host, forcenewserver)

Description

This constructor is used to create an almati2ms tool. The host argument specifies on which networked machine the precompiled almati2ms process should be started. The default value of unset starts it on the current system.

Arguments

msfile	Output MS file name Allowed:Table name Default: ”
fitsin	Input ALMA-TI FITS file Allowed:String Default: ”
append	True if appending to an existing MS Allowed:Boolean Default: F
host	The host on which the almati2ms should be run Allowed:string Default: unset
forcenewserver	Always start a new server Allowed:Boolean Default: F

Returns
An almati2ms tool

Example

```
include 'almati2ms.g'
a := almati2ms(msfile='ALMA.ms', fitsin='ALMA-TI.fits', append=F);
a.fill()
a.done()
```

This example creates an almati2ms tool, sets the data source as a disk file called ALMA-TI.fits, and specifies that the data should be written to an output MS called ALMA.ms. The MS will be over-written if it already exists.

---

almati2ms.setoptions - Function

Package	**alma**
Module	**alma**
Tool	**almati2ms**

Set general filler options

Synopsis
setoptions(compress, combinebaseband)

Description

This function specifies the general filler options which are to be used. These concern the compression mode for the output MS and whether the sub-bands should be filled to a common spectral window per side-band.

Arguments

compress	True if compressed MS output required Allowed:Boolean Default: F
combinebaseband	True if sub-bands are to be combined Allowed:Boolean Default: F

Returns
T or F

Example

---

almati2ms.select - Function

Package alma
Module alma
Tool almati2ms

Input data selection

Synopsis
select(obsmode, chanzero)

Description

This function can be used to select a sub-set of the input ALMA-TI data. At present this allows selection on observing mode and channel-zero type.

Arguments

obsmode	Observing mode (from ALMA-TI DATAPAR) Allowed:String Default: ”
chanzero	Channel-zero selection options Allowed:String:'NONE'—'TIME_AVG'—'TIME_SAMPLED' Default: 'NONE'

Returns
T or fail

Example

1.1.5 almatifiller - Function

Package alma
Module alma

Convert ALMA-TI fits files to an MS

Synopsis
almatifiller(msfile, fitsdir, pattern, append, compress, combinebaseband, obsmode, chanzero, dophcor)

Description

This global function packages common operations using the almati2ms tool and, in addition, permits multiple ALMA-TI files to be filled per run, as specified by a FITS directory name and file modifier string.

It also allows you do do a 'phcor' at the end of the filling. Usually the DATA column hold the 'atmospheric phase-corrected' data. But it does happen at times that this process is not right and the non phase-corrected data amplitude is higher than the one that has been corrected. The process of 'phcor' compare the phase-corrected data to the non phase-corrected data on calibrators (at the 3mm bands) and whenever the latter is higher, data $\pm$ 20 minutes around that scan is replaced with the non phase-corrected data.

This function returns T is everything went well and F otherwise.

Arguments

msfile	Output MS file name Allowed:Table name Default: ”
fitsdir	Input ALMA-TI FITS directory Allowed:String Default: ”
pattern	File modifier string e.g. 18-feb* Allowed:String
append	True if appending to an existing MS Allowed:Boolean Default: F
compress	True if compressed MS output required Allowed:Boolean Default: F
combinebaseband	True if sub-bands are to be combined Allowed:Boolean Default: F
obsmode	Observing mode (from ALMA-TI DATAPAR) Allowed:String Default: ”
chanzero	Channel-zero selection options Allowed:String:’NONE’—’TIME_AVG’—’TIME_SAMPLED’ Default: ’NONE’
dophcor	Do the phcor process after filling Allowed:Boolean Default: F

Returns
T or F

Example

```
include 'almati2ms.g'                                          #1
almatifiller(msfile='ALMA.ms', fitsdir='.', pattern='G067*",
             append=F, compress=F, obsmode='CORR',
             chanzero='TIME_AVG', dophcor=T)                   #2
```

This example fills all ALMA-TI fits files starting with the file name G067 from the current working directory, and creates a new MS named ALMA.ms. The output MS will not be compressed. Only data from the observing mode CORR will be filled and all but the time-averaged channel-zero data will be rejected. Also, as the dophcor parameter is set to ’T’, it will do the phcor process at the end of the filling process.

Index

alma, 5
alma.almati2ms, 20, 22
alma.almati2ms.select, 24
alma.almati2ms.setoptions, 23
alma.almatifiller, 24, 25
alma.iramcalibrater, 6, 7
alma.iramcalibrater.amp, 15
alma.iramcalibrater.done, 17
alma.iramcalibrater.flux, 12
alma.iramcalibrater.phase, 11
alma.iramcalibrater.phcor, 8
alma.iramcalibrater.rf, 9
alma.iramcalibrater.uvt, 16
alma.resample, 17
alma.select, 24
alma.setoptions, 23
alma.shadow, 18
almati2ms, 20, 22
almatifiller, 24, 25
amp, 15

done, 17

flux, 12

iramcalibrater, 6, 7

phase, 11
phcor, 8

resample, 17
rf, 9

select, 24
setoptions, 23
shadow, 18

uvt, 16