from which the flag table was derived. In practice this means the the alternate measurement set should be derived in some way from the original one. Normally the flags in the "flag" table will replace those in the measurement set. It is possible to merge the flags so that the resultant flags are the logical OR of those already in the measurement set and the flags in the "flag" table. This is done by setting the merge argument to T.

Arguments

flagfile	Name of flag file Allowed:String Default: no default
msfile	Name of measurement set Allowed:String Default: unset
merge	Merge the flags from both tables Allowed:Boolean Default: F

Returns

T or fail

Example

```
include 'msplot.g';
msplotapplyflags('3C273XC1.ms.flags.12')
```

This example applies the flags in the table called `3C273XC1.flags.12` to a measurements set called `3C273XC1.ms` in the same directory.

1.3.12 msplot - Tool

Package general
Module ms

tool for MeasurementSet plotting, display, listing and editing
include "msplot.g"

Description

`msplot` provides a GUI-based tool for displaying the data (as line plots or gridded onto a raster) of various parts of a measurement set. It is built on top of the ms tool, allowing plotting of many of the variables accessible via the ms:ms.getdata function. `msplot` also supports interactive flagging of data in a measurement set. Non-interactive flagging is possible using the flagger tool or the autoflag tool.

Next to the **Show** label at the top, you will find a button that allows selection of one of possible plots, displays, and listings. In this context, a plot means a graph of plotted points (made using the pgplotter), a display means an image-like display of gridded data (made using the viewer), and a listing send various information to the logger. The most straightforward plot is X versus Y, where, for example, X could be selected to be Time, and Y could be selected to be field_id. Generally plotting visibility against something else will be more useful: for example, on the Y axis might be observed amplitude, and X might be uvdist. Use the plotflagged switch under the Options menu to control whether flagged points are visible. Use the multi-panel switch under Options to control whether multiple plots are displayed on the same page (as would be useful for *e.g* ATCA baseline plots. Summary and row-by-row listings are available, though the latter is quite underdeveloped currently.

The types of plots allowed are adapted to the type of data: for synthesis data, one can plot uv coverage and the data for a slice in the uv plane. For single dish data, one can plot the pointing coverage and the data for a slice in the sky plane.

For the Plot of X versus Y, the axes to be plotted can be chosen by selecting under the various buttons on the second row labelled X Axis and Y Axis. Iteration over various axes may be enabled by selecting buttons under the Iterate button. To stop an iteration or other operation, press the red Stop button. By suitable combination of these controls, you should be able to able to get a very wide range of useful X, Y plots.

For the Image display, visibility data are displayed as an image, gridded on axes: Interferometer, Row (Time), Channel, and Correlation. The choice of what form of visibility to show as brightness is controlled by the Image button. Selection of the Measurement Set data to be shown is controlled by a number of buttons on the third row. Selecting one of these causes a frame to be shown directly underneath.

X, Y plot limits The scaling for the plot axes may be specified using the entry boxes. The range of values in the plot may be filled into the box using the Ranges button at the right hand side.

Data selection Ranges of quantities to be selected may be input via entry boxes. The range of values in the measurement set may be filled into the box using the Ranges button at the right hand side. A selection may then be made and the Ranges recalculated.

Spectral selection Channels to be plotted may be selected with or without averaging. One specifies the number of desired *output* channels (NCHAN), the *input* channel to start with (START), the number of *input* channels to step between output channels (STEP), and the number of *input* channels to average per output channel (WIDTH). For example, to plot each of 32 channels, use NCHAN=32, START=1, STEP=1, WIDTH=1. To plot the average of all the channels use NCHAN=1, START=1, STEP=1, WIDTH=32. NCHAN=3, START=2, STEP=5, WIDTH=3 will plot 3 output channels, each the average of channels [2,3,4,5,6], [5,6,7,8,9], and [8,9,10,11,12].

Polarization selection The names of the requested polarizations may be input: *e.g.* I Q U V or RR RL LR LL, *etc.*.

The control arguments for `msplot` are saved and restored between executions, and in addition the arguments may be reset to the default values. All of this is possible from the Arguments button at the bottom of the GUI.
When plotting, msplot works by issuing commands to the pgplotter. The plot may be saved in AIPS++ plot file format (actually an AIPS++ table), or written as a postscript file.
When displaying, msplot uses the viewer. It first constructs an image with the relevant axes (by gridding the data onto a cube), and then sends it to the viewer for display. All the capabilities of the viewer may be used to control the display.

1.3.13 Editing via the viewer

Msplot also allows editing of data. This is enabled by constructing msplot with an `edit=T` argument. Editing is possible via the X,Y plots, and the image-like displays. If editing is possible then the top window of the relevant frame will have the line 'Editing Commands'.
In the image-like display, the visibility amplitude (or whatever is selected using the Image menu) is displayed on axes of interferometer, row, correlator, and channel. To optimize the screen use, the view is adapted to the shape of the data. X is the axis with the largest number of pixels, Y the next largest, and so on. Note that the displayed axes may be changed using the Adjust button on the display panel. The world coordinates for each axis (X, Y, Z) will be displayed.
To edit with the image display:

- Use the Image button to control what is displayed as brightness in the plot. This can be Amplitude, Phase, Real, or Imaginary of the Observed, Corrected, Model, or Residual Visibility.

- Use the Polarization Selection button to select the polarizations to be plotted *e.g.* RR, RL, LR, or LL. You can usually leave this as the default, and all full polarizations will be displayed (as the hidden fourth axis).

- Use the Spectral selection button to select the spectral window(s) and the range of channels.
- Optionally use the Field Id range selection further select the data.
- Press Go to display the data.
- Optionally adjust the display using the Adjust button.
- Select the type of flagging using the left hand button on the row of buttons. This has various possibilities: by Antenna, by Time, by Interferometer, by Channel, and by Correlator. For example, to flag all data for a given Time, select just the Time button. To flag all data for a given Antenna and a given Time, select both Time and Antenna.
- The flags may be applied to all Spectral Window Ids and Polarization Ids by selecting the appropriate button.
- To start flagging, move to the display window. To select a **single value**, press the cross-hair button on the left hand size of the display, and then click on the display to set the cross hair, and then double-click to accept the value. To remove the displayed cross-hair, press escape. To select **a range of values**, press the square region button and then click and drag on the display to set the (square) region. To accept the region, double click. To remove the displayed region, press escape.
- To see the currently defined edits, press the List button.
- If you make a mistake, press the Cancel button to forget the current edit or Clear to forget all the stored edits.
- Once you are finished defining edits, they may be applied and the display reloaded using the Flag button. You may now continue defining more edits.
- If you really mess up, then you may revert to the original flagging status on startup using the Revert button.

1.3.14 Editing via X,Y plots

msplot will plot any of a range of quantities in X,Y scatter plots. This may then be edited by drawing a box on the display using the mouse.

- Use the X axis, and Y axis buttons to control what is plotted along the axes.
- Optionally use the Polarization Selection button to select the polarizations to be plotted *e.g.* RR, RL, LR, or LL. You can usually leave this as the default, and all full polarizations will be displayed (as different colors).

- Optionally use the Spectral selection button to select the Spectral Windows and the range of channels.
- Optionally use the Fields button to further select the data.
- To start flagging, move to the plot window. You will select regions by pressing the mouse button 1 (usually the left button), and then dragging to the top right corner. The selected region will be drawn in a green cross-hatched pattern, and the coordinates of the region will be printed out. You may make as many of these regions as you like.
- To control whether the edits are applied to all channels and all correlations uniformly, you may select the corresponding button on the edit window.
- To see the currently defined edits, press the List button.
- If you make a mistake, press the Cancel button to forget the current edit or Clear to forget all the stored edits. Note that the cross-hatched region will remain but will turn light grey.
- Once you are finished defining edits, they may be applied and the display reloaded using the Apply button. You may now continue defining more edits.
- If you really mess up, then you may revert to the original flagging status on startup using the Revert button.

The same set of steps may be used with the Locate button instead of the Flag button. This lists the selected points to the logger. No flagging is done. This is useful for finding outlier points.
The flags are applied with each press of the Flag button. On exit from msplot, you will be asked whether to keep these flags permanently or to restore the old flag columns. If you choose to keep these flags, then the flag column table will contain the flag columns as they were prior to this run of msplot. If you choose to restore the old flagg columns then the flag column table will contain the flag columns after applying these edits.

1.3.15 Useful tips

- For synthesis data, set Data selection/Show Zero Spacing to False to avoid the scale being dominated by the zero spacing points.
- To find the position angle of a feature, use the Plot of Y (data) versus reprojected U axis. Set the number of plots to be 18, and you will see the visibility function plotted for increments of ten degrees.
- Some of the entry fields have graphical tools to help in setting: Data selection/Antennas, Data selection/Interferometer Number, Data

selection/Field Id. As is standard throughout AIPS++, these are accessed via the wrench menu on the right end of each entry widget.

- Data selection/Antennas and Data selection/Interferometer Number have different purposes and usually should not be used together. The former allows all data from a set of antennas to be selected, whereas the latter allows more detailed control over which antenna pairs are to be selected.
- The ranges in the Data selection frame are evaluated by selecting with the current selections, and then finding the ranges. The initial values are calculated with no selection. Use the Update button on the Data selection frame to update the Ranges for the current selection. For example, suppose that you wish to find the range of observed times for a given field id. Set the field id as desired and then press the Update button. The next press of the Range button for the Time will fill in the time range for that field id.
- In both the Spectral selection and Polarization selection frame, the initial values for some of the parameters are filled in when a value is entered in the List entry box.
- To find outlier points with the plotter, use the same steps as you would for flagging but press Locate instead of Flag. The selected points will be identified in the logger.

1.3.16 Background on editing

The Measurement Set contains two columns, FLAG_ROW and FLAG that are used to denote flagging of data. On starting msplot with the argument edit=T, these columns are saved to another table, the flag column table, from which they may be restored as required. As msplot proceeds, the flag columns in the measurement set are modified whenever the Apply button is pressed. Thus on exit from msplot, if one wishes to discard the edits, these columns must be copied back from the temporary flag column table. The user is asked whether this is to be done whenever msplot exits (or when AIPS++ is exited). If the edits are to be discarded, the new flag columns are stored in the flag column table, from where they may be applied using the function msplotapplyflags.
Note that non-interactive flagging is possible using the flagger tool.

Example

```
include 'msplot.g'
include 'imager.g'
imagermaketestms() # To make the standard MS 3C273XC1.ms
mymsplot:=msplot('3C273XC1.ms') # To create a msplot
```

msplot - Constructor

Package	**general**
Module	**ms**
Tool	**msplot**

Construct a visibility plotter

Synopsis
msplot(msfile, nrows, edit, flagfile, displaywidth, displayheight)

Description
Construct a visibility plotter from a MeasurementSet. In processing, only `nrows` rows are read at any one time so this should be used to tune the amount of memory used in processing.
The flagfile is used to stop a copy of the FLAG and FLAG_ROW columns prior to editing. This allows any changes in the flagging to be reversed on exit. The flagfile may then applied afterwards using the function msplotapplyflags

Arguments

msfile	Name of MeasurementSet Allowed:String Default: ''
nrows	Number of rows to buffer at once Allowed:Integer Default: 16384
edit	Allow editing? Allowed:Boolean Default: F
flagfile	Name of flag file Allowed:File name Default: unset
displaywidth	Width of display in pixels Allowed:Integer Default: 600
displayheight	Height of display in pixels Allowed:Integer Default: 300

1.3.17 msplotapplyflags - Function

Package **general**
Module **ms**

Apply flag columns

Synopsis
msplotapplyflags(flagfile, msfile, merge)

Description
msplot can, during a flagging session, create a table that contains the contents of the FLAG & FLAG_ROW columns of the measurement set. This function can copy the flags from this "flag" table back into the corresponding columns of the measurement set.
The name of the measurement set that was used to create the "flag" table is stored in the flag table and it is not necessary to specify which measurement set the "flag" table is associated with. You can apply the "flag" table to a different measurement set by specifying its name using the msfile argument. It is **critical** that the alternate measurement set have the same number of rows, and that the data in each row of be the same shape as the measurement set from which the flag table was derived. In practice this means the the alternate measurement set should be derived in some way from the original one.
Normally the flags in the "flag" table will replace those in the measurement set. It is possible to merge the flags so that the resultant flags are the logical OR of those already in the measurement set and the flags in the "flag" table. This is done by setting the merge argument to T.

Arguments

flagfile	Name of flag file Allowed:String Default: no default
msfile	Name of measurement set Allowed:String Default: unset
merge	Merge the flags from both tables Allowed:Boolean Default: F

Returns
T or fail

Example

```
include 'msplot.g';
msplotapplyflags('3C273XC1.ms.flags.12')
```

This example applies the flags in the table called `3C273XC1.flags.12` to a measurements set called `3C273XC1.ms` in the same directory.

1.4 measures - Module

Measures handling
include "measures.g"

Description

A measure is a quantity with a specified reference frame (e.g. UTC, J2000, mars). The measures module provides an interface to the handling of measures. The basic functionality provided is:

- *Conversion* Conversion of measures, especially between different frames (e.g. UTC to LAST)
- *Calculation* Calculation of e.g. a rest frequency from a velocity and a frequency.

This functionality is provided in both a command line interface and a *GUI* interface.
To access the `measures` module, include the *measures.g Glish* script. This will create a default Measures `tool` called **dm**.
Altough a Measures `tool` has state (it remembers the frame you are working in), there is only in very special cases the need to create your own `tool` . Just use the default **dm** one. However, if you do require one, perhaps to distinguish working in two different environment frames for an input observation and and output display, then use the `measures` constructor to make one:

```
- include 'measures.g';
- mydm := measures(); # measures tool created for you
- mydm.type();
measures
```

In normal use, the **measures** `tool` will be operated using a *GUI.* All operations can be done from either the command line or the *GUI.* The aipsrc mechanism is used by the measures `tool` to specify certain operational details (like using DE405 rather than the default DE200 for planetary positions).

Measures

Measures are e.g. an epoch, or coordinates, which have, in addition to values (as quantities), also a reference specification, and possibly an offset. They are represented as records with fields describing the various entities embodied in the measure. These entities can be obtained by the access methods *gettype*, *getref*, *getoffset*. *getvalue*.

Each measure has its own list of reference codes (see the individual methods for creating them, like *direction*). If an empty or no code reference code is given, the default code for that type of measure will be used (e.g. it is *J2000* for a direction). If an unknown code is given, this default is also returned, but with a warning message.

The values of a measure (like the right-ascension for a direction) are given as quantities. Each of them can be either a scalar quantity with a scalar or vector for its actual value (see the following example). The values can also be given as an r_array, in which case each element of that array will be used as a value. E.g a vector of length 2 of quanta will be seen in a direction constructor as a longitude and a latitude.

```
- dm.epoch('utc','today') # note that your value will be different
[type=epoch, refer=UTC, m0=[value=51350.0778, unit=d]]
- dm.direction('j2000','5h20m','-30.2deg')
[type=direction, refer=J2000, m1=[value=-0.527089434, unit=rad],
  m0=[unit=rad, value=1.3962634]]
- a := dm.direction('j2000','5h20m','-30.2deg')
- dm.gettype(a)
direction
- dm.getoffset(a)
F
- dm.getref(a)
J2000
- dm.getvalue(a)
[*53=[unit=rad, value=1.3962634], *54=[value=-0.527089434, unit=rad]]
- dm.getvalue(a)[1]
[unit=rad, value=1.3962634]
- dm.getvalue(a)[2]
[value=-0.527089434, unit=rad]
- # try as a scalar quantity with multiple values
- a := dm.direction('j2000', dq.quantity([10,20],'deg'),
+ dq.quantity([30,40], 'deg'))
- dm.getvalue(a)[1]
[unit=rad, value=[0.174532925 0.34906585] ]
- dq.getvalue(dm.getvalue(a)[1])[2]
0.34906585
- a
[type=direction, refer=J2000, m1=[value=[0.523598776 0.698131701] , unit=rad],
 m0=[unit=rad, value=[0.174532925 0.34906585] ]]
```

```
- # try as an r_array
- b := r_array(,2)
- r_fill(dq.quantity([10,20],'deg'), b, 1)
T
- r_fill(dq.quantity([30,40],'deg'), b, 2)
T
- a := dm.direction('j2000', b)
- dm.getvalue(a)
[*61=[unit=rad, value=[0.174532925 0.34906585] ],
 *62=[value=[0.523598776 0.698131701] , unit=rad]]
```

Known measures are:

- epoch: an instance in time (internally expressed as MJD or MGSD)
- direction: a direction towards an astronomical object (including planets, sun, moon)
- position: a position on Earth
- frequency: electromagnetic wave energy
- radialvelocity: radial velocity of astronomical object
- doppler: doppler shift (i.e. radial velocity in non-velocity units like 'Optical', 'Radio'. The radialvelocity GUI caters for Doppler coding as well)
- baseline: interferometer baseline
- uvw: UVW coordinates
- earthmagnetic: Earth' magnetic field

In addition to the reference code (like J2000), a measure needs sometimes more information to be convertable to another reference code (e.g. a time and position to convert it to an azimuth/elevation). This additional information is called the reference *frame*, and can specify one or more of 'where am i', 'when is it', 'what direction", 'how fast'.
The frame values can be set by the doframe tool function, or by a menu on the measure guis.
Since you would normally work from a fixed position, the position frame element ('where you are'), can be specified in your .aipsrc if its name is in the Observatory list (obslist) tool function. You can set your preferred position by adding to your *.aipsrc* file:

```
measures.default.observatory: atca
```

More information on the individual measures can be found in the GUI descriptions for epoch; direction; position; frequency; radialvelocity; doppler.

Example
Suppose we have an object with a J2000 right ascension and declination, and want to know the time of rising above the horizon of a certain astronomical direction. In practice you would do this using the special application in the gui, but the following shows how to do it the hard and long way.

```
- include 'measures.g'# 1
T
- tim := dm.epoch('utc','today') # 2
- tim
[type=epoch, refer=UTC, m0=[value=51350.0934, unit=d]]
- pos := dm.observatory('ATCA') # 3
- dm.doframe(tim)  # 4
T
- dm.doframe(pos)                # 5
T
- coord := dm.direction('J2000', '5h20m30.2', '-30d15m12.5')  # 6
- hadec := dm.measure(coord,'hadec') # 7
- last := dm.measure(tim,'last') # 8
- sdec := dq.sin(dm.getvalue(hadec)[2]) # 9
- cdec := dq.cos(dm.getvalue(hadec)[2])
- clat := dq.cos(dm.getvalue(pos)[2])
- slat := dq.sin(dm.getvalue(pos)[2])
- ha := dq.acos(dq.neg(dq.div(dq.mul(sdec, slat),
 dq.mul(cdec, clat))))  #10
- dq.norm(dq.add(ha, dm.getvalue(coord)[1]), 0) #11
[value=189.926855, unit=deg]
- dq.time(dq.norm(dq.add(ha, dm.getvalue(coord)[1]), 0)) #12
12:39:42.445
- rt := dq.totime(dq.norm(dq.add(ha, dm.getvalue(coord)[1]))) #13
- rt
[value=-0.472425402, unit=d]
- rtoff := dm.epoch('r_utc', dm.getvalue(tim)[1]) #14
- rtoff
[type=epoch, refer=R_UTC, m0=[value=51350, unit=d]]
- rte := dm.epoch('last', rt, off=rtoff) #15
- rte
[type=epoch, refer=LAST, m0=[value=58062.5276, unit=d]]
- dm.measure(rte, 'utc') #16
[type=epoch, refer=UTC, m0=[value=51349.3674, unit=d]]
- dq.time(dm.getvalue(dm.measure(rte, 'utc'))[1], form="ymd time")
1999/06/20/08:48:59.738
- dq.time(dm.getvalue(dm.measure(dm.measure(rte, 'utc'), 'last'))[1]) #17
12:39:42.445
- # try it another way
- tim := dm.epoch('utc', 'today') #18
```

```
- tim := dm.measure(tim,'tai') #19
- dm.doframe(tim) #20
T
- dm.showframe(F) #21
position: 09:58:12.033 -030.08.43.736 6372960.26 m ITRF
epoch: 2002/10/18/00:15:49 UTC
T
- is_measure(tim) #22
T
- sun:=dm.direction('sun') #23
- dm.rise(sun) #24
[rise=[value=20.4413559, unit=deg], set=[value=158.042775, unit=deg]]
- dm.rise(sun,'10deg')  #25
[rise=[value=27.3170864, unit=deg], set=[value=151.167045, unit=deg]]
- dq.form.long(dm.rise(sun).rise) #26
01:21:45.925
- dq.setformat('long','dms') #27
T
- dq.form.long(dm.rise(sun).rise)
+020.26.28.881
```

1. Gives access to the measures, with a default server dm
2. Get the time now as an epoch measure
3. Get the position of an observatory
4. Set the time in reference frame. Note that if you are working in a GUI environment, the frame set will be displayed. If working in a pure CLI environment, the current frame can be shown with the showframe tool function.
5. Set the position in reference frame
6. Get the coordinates as a measure
7. Convert the coordinates to hour-angle, declination
8. Get the sidereal time for now (just for the fun)
9. Calculate the sines/cosines of declination and latitudes
10. Get the hour-angle for zero elevation
11. Get the sidereal time of setting (rising?), normalised between 0 and 360 degrees
12. Show it in a time format
13. Save it as a time (remember, we calculated it as an angle)

14. We would like to convert this sidereal time (rte) back to UTC. To be able to do that, we most now when (for which date) the sidereal time is valid. In general we do not know the Sidereal date, else it would be easy. We can however, specify an offset with a measure. Special reference codes are available (called 'raze'), which will after conversion only retain the integral part. This line says that we specify tim as an epoch in UTC, to be razed after conversion. If we now use this as an offset for a sidereal time, the offset will be automatically converted to sidereal time (since it has to be added to a sidereal time), razed, i.e. giving the Greenwich sidereal date, and hence the complete epoch is known. (Note that the indecision over the 4 minutes per day has to be handled in special cases).

15. We can now define our sidereal time rt as a real epoch

16. We now convert the sidereal time rte, to an UTC, and show it as a time as well

17. We do not trust this funny razing, and convert the UTC obtained straight back to a sidereal time. And, indeed it worked (compare line 12)

18. Another time

19. Convert it to TAI (just to show how to do it)

20. Frame it

21. Show the current frame (the F argument indicates to not use GUI if one present) (note that the position is set to ATCA, as defined by the .aipsrc variable). If an error happens, the command

    ```
    dm.doframe(dm.observatory('atca'))
    ```

    will solve it.

22. to check if we really got a measure, or an error occurred.

23. define the position of the sun (the actual position will be at the frame time)

24. get the sidereal time of rise/set of the sun (at the default elevation of 5 deg)

25. try again for an elevation of 10 deg

26. display the rise sidereal time as a 'longitude'. A variety of dq.form.x routines exist, using a global format setting mechanism (in general you have your preferred way to display something)

27. change the format, and show again

The above could be done using the GUI (start it with dm.gui()), setting your position in the Frame menu option, and using the rise/set application.
Another option would be to use the direction GUI (from the Tool menu in the general GUI or starting dm.directiongui() from the command line); selecting the Sun as planet; setting the position in the frame menu and the time; Convert to J2000, and select rise/set from the Info menu.
If at one stage you want to use a GUI result on the command line, the 'copy' button will transfer the last result to the clipboard (and read it with the normal clipboard operation

```
- dcb.paste()
[type=direction, refer=J2000, m1=[value=0.669116295, unit=rad], m0=[unit=rad, value=0.0259745717]]
```

Tools

measures	measures tool
measuresgui	gui code for measures.g

Functions

is_measure	Check if measure

1.4.1 is_measure - Function

Package general
Module measures

Check if measure

Synopsis
is_measure(v)

Arguments

v	in	value to be tested Allowed:any

Returns
Bool

Description
Checks if the operand is a correct measure