NOTE 207 – Guide to gbtlogview

Joe McMullin

April 11, 1997

Contents

1 Purpose

gbtlogview is a valuable resource to the engineering and technical staff at Green Bank. This note is indended to introduce and demonstrate the utility of the program along with various scripts available for use.

2 Introduction

The program, gbtlogview, is a tool for listing and plotting AIPS++ table format data.

3 Running gbtlogview

gbtlogview is currently in a state of flux, as AIPS++ versions are evolving. Most of these changes will be transparent to the user in terms of invoking the program. Starting gbtlogview, including setting the various paths, etc, is done by:

```
scylla<27>% gbtlogview2                              (new version of AIPS++)
```

or

```
scylla<27>% gbtlogview2                              (test version of AIPS++)
```

For those interested in the details, the gbtlogview2 script is the following:

```
/opt/local/bin/gbtlogview2

#!/opt/local/bin/bash

# Starts test version of aips++ GUI for engr logs.  RDN 6/20/96

source /aips++/new/aipsinit.sh
# aipsinit _
gbtlogview
```

The gbtlogview script called by gbtlogview2 is:

```
gbtlogview : /aips++/new/sun4sol\_gnu/bin/gbtlogview

#!/bin/sh
#
.
.    Copyright comment text has been cut out.
.
#
#   $Id: gbtlogview.tex,v 19.0 2003/07/16 03:48:03 aips2adm Exp $
#------------------------------------------------------------------------------

# in cv
#INSTALL_PATH=./
# in gb
```

```
INSTALL_PATH=/aips++/local/bin

xrdb -merge $INSTALL_PATH/xres
glish -l aips++.g -l gbtlogview.g
```

This sets up an addition to the X-server resource database and then executes the glish scripts aips++.g and gbtlogview.g (in /aips++/version/sun4sol_gnu/libexec).

After executing 'gbtlogview2', the user is then in the glish environment in the executing window, while a "GBT Log Data" GUI appears (Figure **??**). The GUI is intended as the main interface. The frame labeled 'Data Source(s)' has a list of data *clients* for examination, including various receivers and the local weather station. Each or all of these sources may be selected (by clicking on them with the mouse). The next frame over features several toggle buttons which control the range of selected data. When one of these buttons is selected, for example the "Last Hour" button, information on the start and end time and output table name are automatically obtained. To actually construct a table of values from the selected quantities, the 'Fill' button takes the data from the selected *clients* and produces a flat table of data records with all of the relevant information. Figure **??** displays the window that opens to show the filler activity.

Two Data Sources of special note are "Specify Device" and "Specify ASCII table". "Specify Device" allows the user to specify the directory where a non-standard device's data files are stored. Pressing the "Fill" button in this case brings up a frame where you can type the name of the directory (e.g. one can access a normal Rx this way as well, Directory: /GBTlogs/Receivers/RC04_6). It will then grab standard data files according to the time range specified.

"Specify ASCII table" allows the user to specify an ascii table on disk. Pressing the "Fill" button, brings up a frame where you specify the directory and table name (e.g. Directory: /scylla/jmcmulli/ [note the necessary trailing slash], Filename: test.inp). The format of the table must be in the form:

```
Col1 Col2 Col3 Col4 Col5 Col6 # Column titles
I       I       I       R       R       R       # The format of the column:
                                                #I=int,R=real,A=string
                                                #D=double,X=complex(R,I)
                                                #Z=complex(Amp,phase)
1       4       9       3.45    5.46    7.89    # These are the actual data
2       2       7       2.1     0.1     4       # values.
```

These two special devices are mutually exclusive.

The 'Browse' button pops up a window with all of the selected data sources' information in table format. The table may be scrolled through with the scroll bars but plotting isn't currently enabled from this window.

The 'Plot Columns' window (Figure **??**) has all of the data quantity names in three columns (X, Y, and Y2). Clicking on a quantity selects it for that axis. The 'Plot' button then displays it on the plotter window (Figure **??**). Several Y-axis items may be selected for comparison, along with an additional Y2-axis item which will have its axis labeled on the right hand side of the plot.

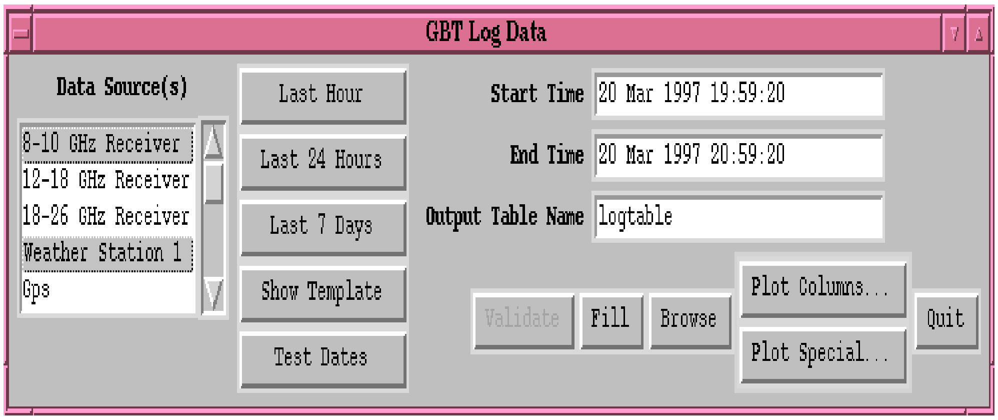
4 Tables

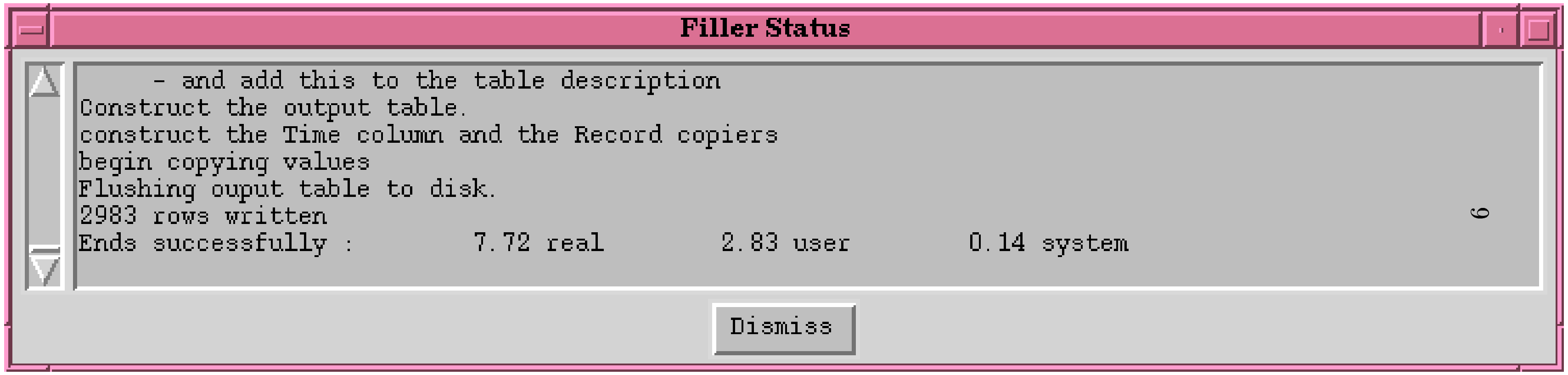
AIPS++ manipulates and stores all data in the form of tables. These tables consist of an unlimited number of columns of data with optional column keywords and table keywords. Manipulating these tables is the key to fully utilizing gbtlogview. Within AIPS++, the table module is equiped with various methods and functions for this purpose. Please refer to the table - Module in the Reference Manual.

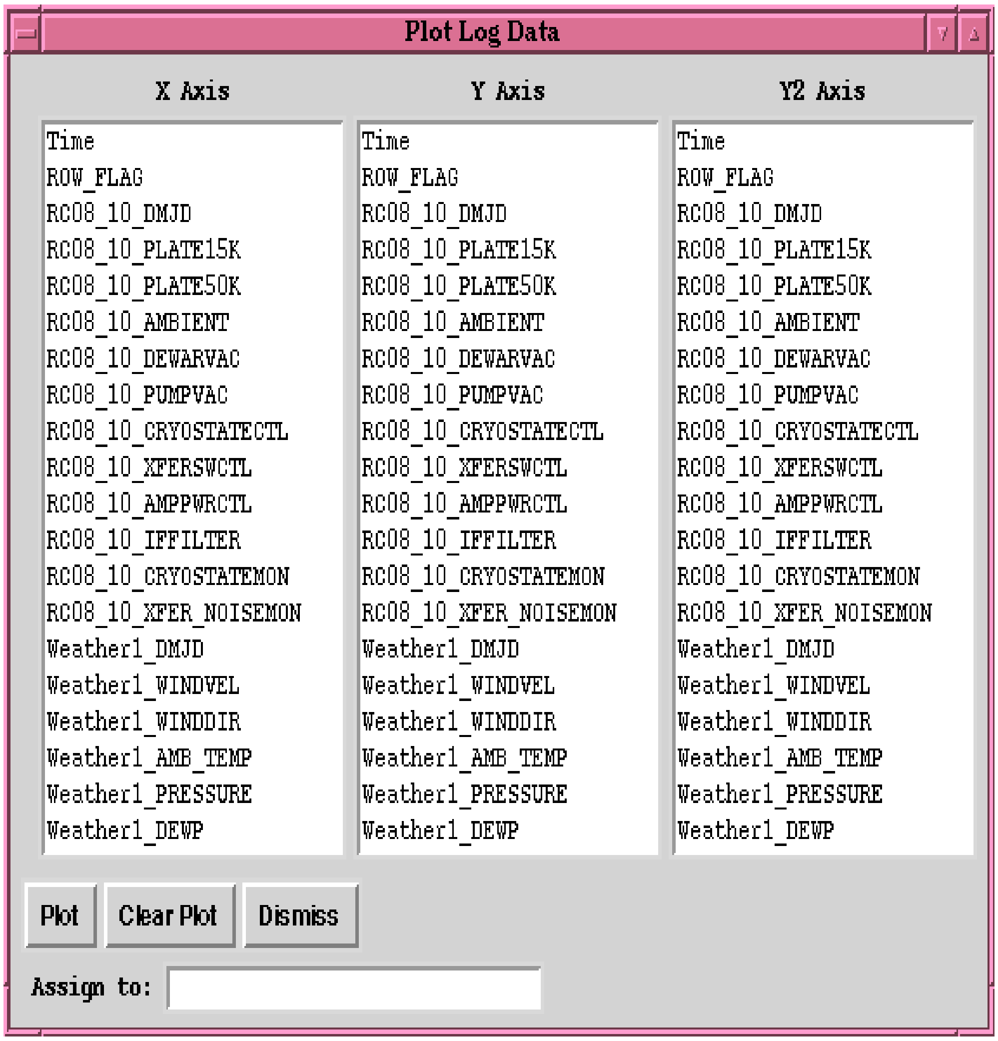
Initially, after hitting the 'Fill' button, the table "logdata" will be created on disk in the current directory. To verify this type:

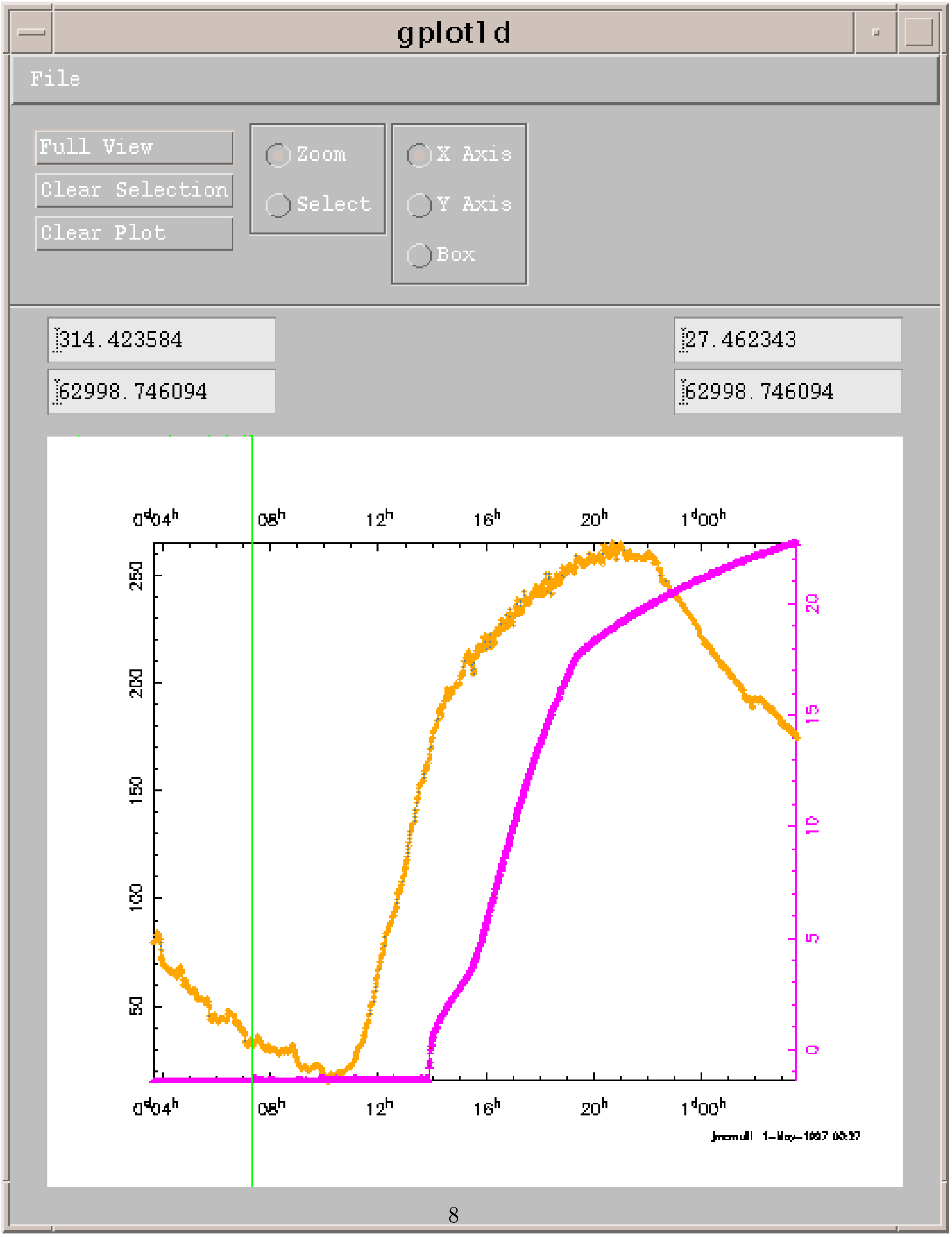
```
- tableopentables()
logtable
- table("logtable").colnames()
Time ROW_FLAG RC08_10_DMJD RC08_10_PLATE15K RC08_10_PLATE50K RC08_10_AMBIENT RC0
8_10_DEWARVAC RC08_10_PUMPVAC RC08_10_CRYOSTATECTL RC08_10_XFERSWCTL RC08_10_AMP
PWRCTL RC08_10_IFFILTER RC08_10_CRYOSTATEMON RC08_10_XFER_NOISEMON RC12_18_DMJD
RC12_18_PLATE15K RC12_18_PLATE50K RC12_18_AMBIENT RC12_18_DEWARVAC RC12_18_PUMPV
AC RC12_18_CRYOSTATECTL RC12_18_XFERSWCTL RC12_18_AMPPWRCTL RC12_18_IFFILTER RC1
2_18_CRYOSTATEMON RC12_18_XFER_NOISEMON RC18_26_DMJD RC18_26_PLATE15K RC18_26_PL
ATE50K RC18_26_AMBIENT RC18_26_DEWARVAC RC18_26_PUMPVAC RC18_26_CRYOSTATECTL RC1
8_26_XFERSWCTL RC18_26_AMPPWRCTL RC18_26_IFFILTER RC18_26_CRYOSTATEMON RC18_26_X
FER_NOISEMON Weather1_DMJD Weather1_WINDVEL Weather1_WINDDIR Weather1_AMB_TEMP W
eather1_PRESSURE Weather1_DEWP Gps_DMJD Gps_ID Gps_DAY Gps_DOY Gps_DATE Gps_TIME
 Gps_SV Gps_EL Gps_AZM Gps_SN Gps_AGE Gps_ION Gps_INTRNL Gps_DF_F Gps_AVG_DF_F G
ps_NO1 Gps_TI Gps_TI_FIT Gps_TI_RATE Gps_NO2
```

The second command, 'table("logtable").colnames()' lists all of the column names within the table. The list displayed is if all of the receiver,

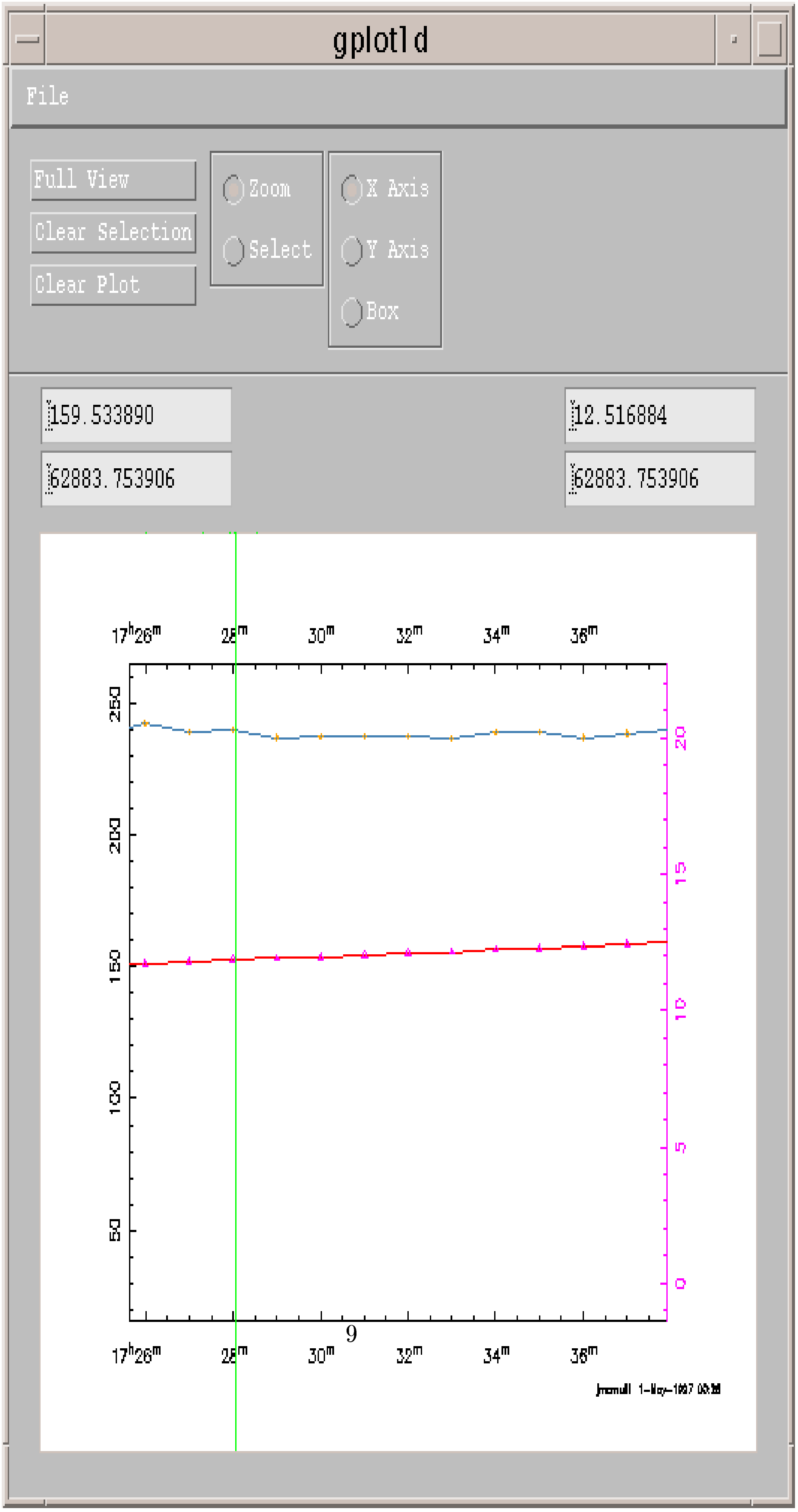

GBT Log Data
Data Source(s)
8-10 GHz Receiver
12-18 GHz Receiver
18-26 GHz Receiver
Weather Station 1
Gps
Last Hour
Last 24 Hours
Last 7 Days
Show Template
Test Dates
Start Time
20 Mar 1997 19:59:20
End Time
20 Mar 1997 20:59:20
Output Table Name
logtable
Validate
Fill
Browse
Plot Columns...
Plot Special...
Quit


Filler Status
- and add this to the table description
Construct the output table.
construct the Time column and the Record copiers
begin copying values
Flushing ouput table to disk.
2983 rows written
Ends successfully : 7.72 real 2.83 user 0.14 system
Dismiss


Plot Log Data
X Axis
Time
ROW_FLAG
RC08_10_DMJD
RC08_10_PLATE15K
RC08_10_PLATE50K
RC08_10_AMBIENT
RC08_10_DEWARVAC
RC08_10_PUMPVAC
RC08_10_CRYOSTATECTL
RC08_10_XFERSWCTL
RC08_10_AMPPWRCTL
RC08_10_IFFILTER
RC08_10_CRYOSTATEMON
RC08_10_XFER_NOISEMON
Weather1_DMJD
Weather1_WINDVEL
Weather1_WINDDIR
Weather1_AMB_TEMP
Weather1_PRESSURE
Weather1_DEWP
Y Axis
Time
ROW_FLAG
RC08_10_DMJD
RC08_10_PLATE15K
RC08_10_PLATE50K
RC08_10_AMBIENT
RC08_10_DEWARVAC
RC08_10_PUMPVAC
RC08_10_CRYOSTATECTL
RC08_10_XFERSWCTL
RC08_10_AMPPWRCTL
RC08_10_IFFILTER
RC08_10_CRYOSTATEMON
RC08_10_XFER_NOISEMON
Weather1_DMJD
Weather1_WINDVEL
Weather1_WINDDIR
Weather1_AMB_TEMP
Weather1_PRESSURE
Weather1_DEWP
Y2 Axis
Time
ROW_FLAG
RC08_10_DMJD
RC08_10_PLATE15K
RC08_10_PLATE50K
RC08_10_AMBIENT
RC08_10_DEWARVAC
RC08_10_PUMPVAC
RC08_10_CRYOSTATECTL
RC08_10_XFERSWCTL
RC08_10_AMPPWRCTL
RC08_10_IFFILTER
RC08_10_CRYOSTATEMON
RC08_10_XFER_NOISEMON
Weather1_DMJD
Weather1_WINDVEL
Weather1_WINDDIR
Weather1_AMB_TEMP
Weather1_PRESSURE
Weather1_DEWP
Plot
Clear Plot
Dismiss
Assign to:



gplot1d
File
Full View
Clear Selection
Clear Plot
Zoom
Select
X Axis
Y Axis
Box
314.423584
62998.746094
27.462343
62998.746094
0d04h
08h
12h
16h
20h
1d00h
250
200
150
100
50
20
15
10
5
0


gplot1d
File
Full View
Clear Selection
Clear Plot
Zoom
Select
X Axis
Y Axis
Box
159.533890
62883.753906
12.516884
62883.753906
17h26m
28m
30m
32m
34m
36m
250
200
150
100
50
20
15
10
5
0

weather and gps data had been selected; typically, this list will be shorter.

The structure of the table as constructed by the filler is as follows:

Time	Device1_DMJD	Device1_Quantity_X...	Device2_DMJD	Device2_quantity_Y...
t1	t1	x @ t1	t4	y @ t4
t2	t2	x @ t2	t4	y @ t4
t3	t3	x @ t3	t4	y @ t4
t4	t3	x @ t3	t4	y @ t4
t5	t5	x @ t5	t4	y @ t4
t6	t6	x @ t6	t4	y @ t4
t7	t6	x @ t6	t7	y @ t7
t8	t8	x @ t8	t7	y @ t7

Where the Time column is the running time stamp which will be equal to the most rapidly sampled variable and t1¡t2¡tn.

The filler essentially pads those columns which are more coarsely sampled with duplicate records until the monitoring program updates that variable. At that time, the device's time column "Devicen_DMJD" will be equal to the "Time" column.

At this point, you can construct a table object within glish (data) and then manipulate the table according to your needs.

```
- data:=table("logtable")
```

4.1 Examples of using the table-module

4.1.1 A simple regridding routine

Regrid a rapidly sampled variable to the same sampling rate of a more coarsely sampled variable. Use monitoring of 15 K stage on 8-10 GHz receiver versus ambient temperatures from the weather station.

```
- data:=table("logtable")                 # Build a table object within glish.
- sub:=data.query("Time == RC08_10_DMJD") # Select a subtable of that data
                                          # based on the condition that the
  # 8-10 GHz time stamp is equal to
  # the current time (that is, a
  # sample has just been taken of
  # the 8-10 GHz receiver).
- sub.nrows()                             # The number of unique samples of
1432      # 8-10 GHz receiver data.
- data.nrows()                            # The original number of samples
```

```
86118                                     # based on the most finely sampled
                                          # variable.
- xaxis:=sub.getcol("RC08_10_DMJD")       # Get the time field for the 8-10
                                          # GHz data.
- xtimeinseconds:=(xaxis-as_integer(xaxis[1]))*86400. # convert from days->sec
- y1axis:=sub.getcol("RC08_10_PLATE15K")  # Get 15 K Plate Temperature.
- y2axis:=sub.getcol("Weather1_AMB_TEMP") # Get Ambient temp from Weather Stat.
- clear()    # Clear plot
T
- timeY(xtimeinseconds,y1axis,"")         # Plot time vs. 15 K plate temp on Y1
0
- timeY2(xtimeinseconds,y2axis,"")        # Plot time vs. Amb temp on Y2
1
```

Clicking on the far right button will blow up the plot screen to full size. Using the cursor, one can select subregions and see that the ambient temperature data (which is sampled at approximately a 1.5 second rate) is now on the 1 minute grid of the 15 K Plate data.

4.1.2 Count cases greater than a certain value

If one wanted to know, how many times the 8-10 GHz receiver's 15 K plate had gone above 17 K.

```
- bigtemp:=data.query("RC08_10_PLATE15K > 17.0")
- bigtemp.nrows()
49224
```

4.1.3 Maximum and minimum within a certain time range

Continuing with the example above, find the maximum and minimum temperature in the last 3 hours of the data.

```
- xstart:=dm.quantity(xtimeinseconds[1],'s') # get the beginning time and
     # convert it into a quantity with
     # units of seconds.
- xstart
[unit=s, value=13098.2014]
- xstop:=dm.add(xstart,'3.h')      # set the end time to the start
     # time plus 3 hours.
- xstop
```

```
[unit=s, value=23898.2014]
- platetemp:=data.getcol("RC08_10_PLATE15K") # make an array of all the 15 K
     # plate temperatures.
- mask:=xtimeinseconds<xstop[value]          # define a boolean mask that is
     # true only within the time range.
- maxinfirst3:=max(platetemp[mask])          # get the max in this range
- mininfirst3:=min(platetemp[mask])          # get the min in this range
- print mininfirst3, maxinfirst3
15.1367188 16.1132812
```

5 rtools4.g

A number of utility plotting functions are available through the glish script, rtools4.g. The functions allow easy plotting and comparison of variables of interest.

The utilities currently available are:

- plotXdewvac(flg): Plots the X-band dewar vaccum reading.
- plotKudewvac(flg): Plots the Ku-band dewar vacuum reading.
- plotKdewvac(flg): Plots the K-band dewar vacuum reading.
- plotXpumpvac(flg): Plots the X-band pump vacuum reading.
- plotKupumpvac(flg): Plots the Ku-band pump vacuum reading.
- plotX15K(flg): Plots the X-band 15K temperature.
- plotX50K(flg): Plots the X-band 50K temperature.
- plotXamb(flg): Plots the X-band ambient temperature.
- plotKu15K(flg): Plots the Ku-band 15K temperature.
- plotKuamb(flg): Plots the Ku-band ambient temperature.
- plotK15K(flg): Plots the K-band 15K temperature.
- plotK50K(flg): Plots the K-band 50K temperature.
- plotKamb(flg): Plots the K-band ambient temperature.
- plotTemp(flg,scale): Plots the Air Temperature from WT1.

•plotDewpt(flg,scale): Plots the Dew Point from WT1.

•plotRH(flg): Plots the relative humidity calculated from temperature.

•plotPressure(flg,scale): Plots the pressure data from WT1.

•plotWindvel(flg,scale): Plots the wind velocity from WT1.

•plotWinddir(flg): Plots the wind direction from WT1.

•plotPpsRackTemp(flg,scale): Plots the rack temperature from OnePpsStatus.

•plotRtpm140Delay(flg):Plots the delay column from Rtpm140.

•plotRtpmOvlbiDelay(flg): Plots the delay column from RtpmOvlbi.

•plotDeltas(ticka,tickb): Plots the OnePps Deltas.

In the new version of AIPS++, any of these functions can be accessed after data has been 'filled', as long as the command 'include rtools4.g' has been executed either interactively after the glish prompt in gbtlogview, or by setting it up in one's .glishrc file.

The syntax for their use is simply:

```
- function(flg)
```

or

```
- function(flg,scale)
```

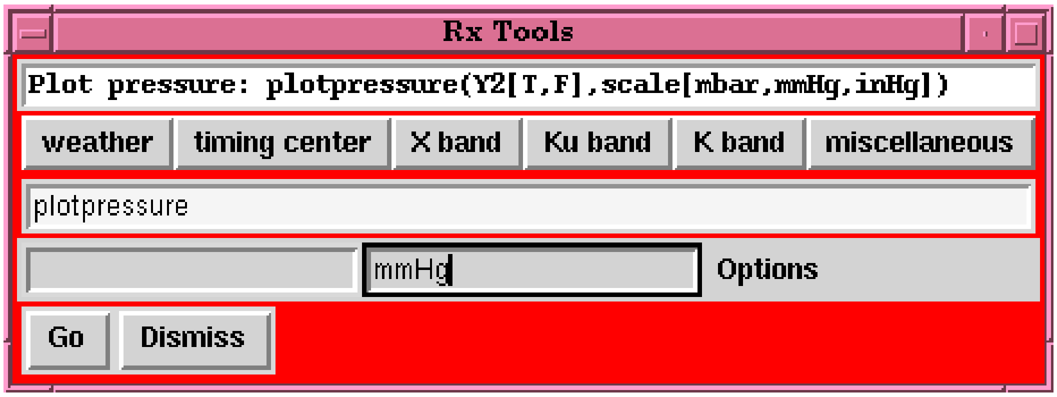
where flg is either blank, (), in which case the plot goes to the Y1 axis, or flg is (y2), in which case the plot uses the Y2 axis; currently, one must always plot something to the Y1 axis before the Y2 axis can be used. If the function takes a 'scale' parameter, this can be either 'f' for farenheit (functions: plotTemp, plotDewpt, plotPpsRackTemp, and plotPpsRoomTemp), 'mph' for miles/hour (function:plotWindvel), or 'mmHg' and 'inHg' for millimeters/inches of mercury (function: plotPressure); the defaults (either left blank or any other combination of charactes) are degrees celsius, meters/sec, and mbars, respectively.

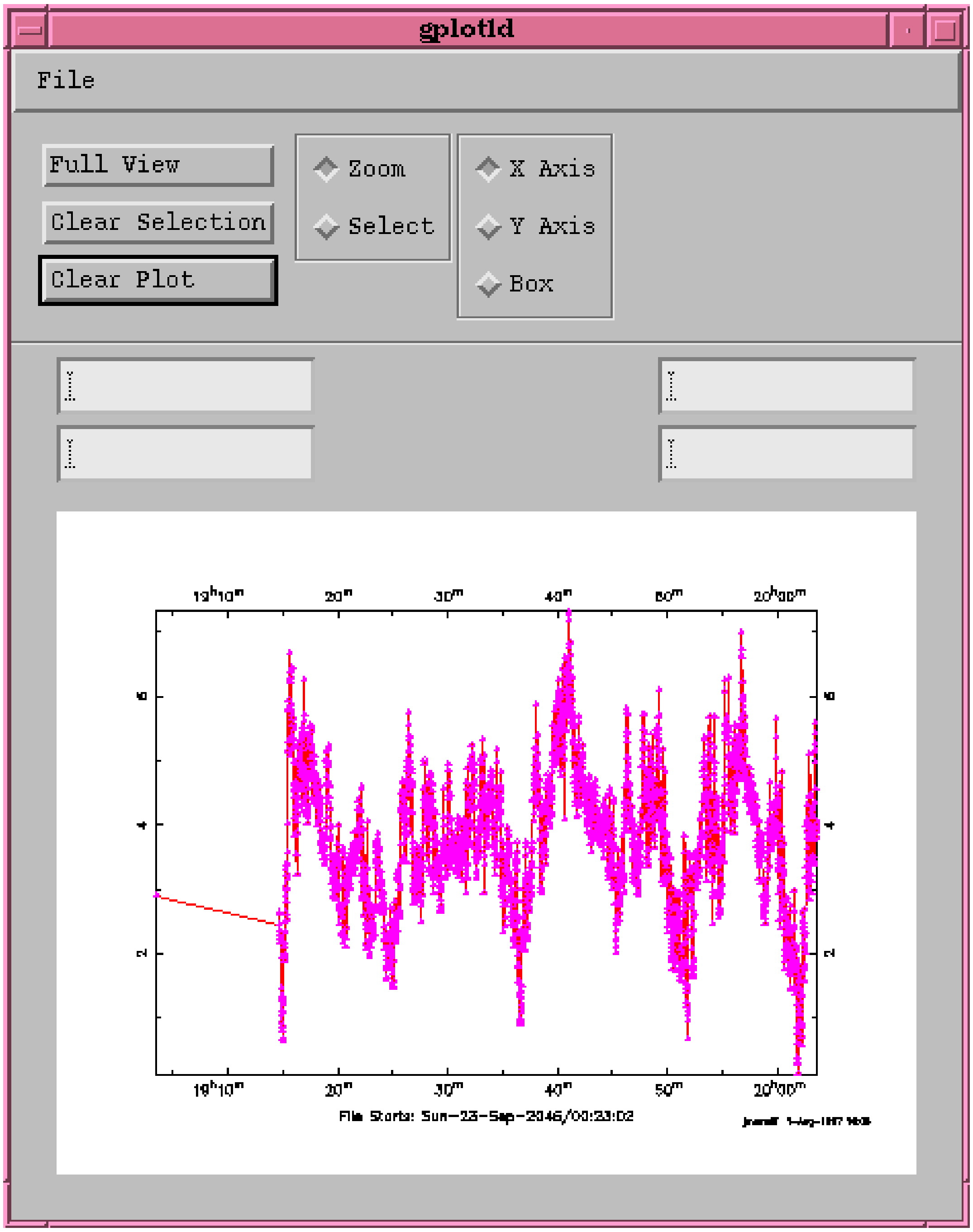
The plot will be similar to that provided normally by gbtlogview.

In the 'test' version of AIPS++, these utilities have been integrated into a GUI. It can be invoked after data has been filled by:

```
- include 'rxtools.g'
X-band data found
Weather1 data found
OnePpsDeltas data found
T
- call_rx_gui()
F
```

Each of the buttons shown in the figure are menus for calling up different functions, while the entry boxes (Options) allow specification of the function inputs. For example, plottemp has two optional specifications (as shown on the status line), Y2[T,F] and scale(C,F); these indicate that you can specify whether the temps should be plotted on the Y2 axis (true or false), and which temperature scale (celsius or fahrenheit) to use for the plots.


Rx Tools
Plot pressure: plotpressure(Y2[T,F],scale[mbar,mmHg,inHg])
weather
timing center
X band
Ku band
K band
miscellaneous
plotpressure
mmHg
Options
Go
Dismiss


gplot1d
File
Full View
Clear Selection
Clear Plot
Zoom
Select
X Axis
Y Axis
Box
19h10m
20m
30m
40m
50m
20h00m
6
4
2
File Starts: Sun-23-Sep-2046/00:23:02