

**SCUBA-2 DR Pipeline Project Office**

University of British Columbia  
 6224 Agricultural Road  
 Vancouver, British Columbia  
 CANADA  
 V6T 1Z1

Tel: +1-604-822-2211  
 Fax: +1-604-822-5324  
 Email: jmolnar@ubc.ca


WWW: <http://scuba2.jach.hawaii.edu>

**Document Title: SCUBA-2 StripChart Utility**

**Document Number: SC2/SOF/S210/006**

**Issue: 1.8**

**Date: 2005/05/25**

Document Prepared By:	Andy Gibb	Signature and Date:	2005-05-24
Document Approved By:	Douglas Scott	Signature and Date:	2005-05-24
Document Released By:	Janos Molnar	Signature and Date:	 2005-05-24



# Contents

<b>1 Introduction</b>	<b>3</b>
1.1 Terminology . . . . .	3
<b>2 Stripchart overview</b>	<b>3</b>
<b>3 Configuration file format</b>	<b>4</b>
3.1 Headings . . . . .	4
3.2 Global parameters . . . . .	4
3.3 Chart parameters . . . . .	5
3.4 Monitor parameters . . . . .	5
3.5 Example chart.ini . . . . .	6
<b>4 User interface</b>	<b>8</b>
4.1 Command-line arguments . . . . .	8
4.2 Interactive control with xstripchart . . . . .	9
4.3 Monitor attributes for each plotting class . . . . .	10
<b>A Stripchart design and structure</b>	<b>11</b>
A.1 Class hierarchy . . . . .	11
<b>B simple file format</b>	<b>13</b>



# 1 Introduction

This document describes the stripchart utility written for the SCUBA-2 Data Reduction Pipeline. The document is structured to provide an introduction to the stripchart program, a detailed description of the configuration options, followed by how a user runs and controls a stripchart and finishing with a high-level overview of the design of the program.

The stripchart has been designed to meet the requirements specified in the Software Requirements document<sup>1</sup> (**DR12** through **DR19**). These requirements will be pointed out via a note in the margin as a matter of course in the description below.

The stripchart plotter plots time-varying data generated by the data reduction (DR) Pipeline. It does this by monitoring specific data files which are written by the Pipeline. These data include, but are not limited to, statistics of bolometer performance (mean, median, standard deviation), flux conversion factors (FCFs), beam-size monitoring and opacity measurements.

**Req. DR12**

The stripchart can read ORAC-DR Index format files and plain ASCII (simple format in the terminology of the stripchart), allowing data from non-Pipeline and/or diagnostic programs to be monitored. Additionally, non-DR data can be plotted along with DR data, for example the WVM-derived optical depth can be plotted against skydip-derived values. The stripchart has a flexible and extensive configuration format which enables the monitoring and plotting of a wide range of data.

**Req. DR19**

## 1.1 Terminology

Conceptually, a *stripchart* is made up of one or more *charts*, upon each of which is plotted data retrieved by one or more *monitors*. The monitors read the data from the specified data files and pass them to *sinks* which plot them on the relevant chart(s).

## 2 Stripchart overview

The stripchart sets up a single plotting window on which all of the charts are drawn. Multiple charts may be plotted by a single stripchart, and multiple monitors may be plotted on a single chart.

**Req. DR13**

**Req. DR14**

The stripchart requires a configuration file (default name `chart.ini` located in the current directory), in which various parameters for the stripchart are specified (see § 3 below). It is possible to specify the number of charts to be drawn, whether the  $y$ -axis is scaled automatically based on the range of  $y$  values or fixed to user-specified values, whether a particular chart displays data within a fixed moving window or simply grows to plot all incoming data, the units for the time axis (hours, MJD etc), as well as colours and styles for points plotted. In addition, the `xstripchart` interface establishes a control window which enables the interactive adjustment of some of the configuration options.

**Req. DR16**

**Req. DR15**

Each chart is configured independently of all the others, allowing different charts to use different scales, windows etc. The same monitor may be plotted on different charts. There is no restriction on the number of stripcharts that can be run at any one time, and multiple stripcharts can employ identical configuration files.

**Req. DR18**



## 3 Configuration file format

The stripchart configuration file format is based on that of .INI configuration files (see e.g. <http://www.cloanto.com/specs/ini.html>). The name is not restricted to ending in .ini. The file is divided into sections through the use of heading keywords (enclosed in square brackets), and in each section parameters are specified through a `parameter = value` syntax. An example configuration file is shown and described in §3.5 below.

### 3.1 Headings

Section headings are enclosed in square brackets. There must be a minimum of three sections defined:

**globals**

for global parameters

**chart*n***

for chart parameters

**monitor\_name**

for monitor parameters

There is one `chart` section for each chart to be plotted. Multiple monitors can be plotted on a given chart. The charts are numbered sequentially, starting from 1.

### 3.2 Global parameters

Global parameters control the overall layout of the plotting window

**nx** – number of plots in the *x*-direction

**ny** – number of plots in the *y*-direction

**output\_class** – plotting package class (device) to use

The maximum number of charts is determined by the `nx` and `ny` parameters. If either `nx` or `ny` is unspecified then it is assumed to be equal to 1. If none are specified, then a single panel is drawn and data from the `chart1` monitors will be plotted. The stripchart has an upper limit of 64 charts, and 16 monitors per chart.

The currently available plotting package classes (devices) are:

**AST:PGPLOT**

**AST:PLplot**

**AST:Tk**

**PLplot**



Variable	Description	Default value
<code>autoscale</code>	Flag to autoscale the $y$ -axis, 1 for autoscaling, 0 for none	1
<code>yscale</code>	Pair of numbers to specify $y_{\min}$ , $y_{\max}$	0,1
<code>yunits</code>	A text string denoting the $y$ -axis units	
<code>growt</code>	Flag to specify whether to always allow the time axis to stretch (grow) as new data are plotted (so that all data are always plotted) or just to have a moving window of fixed width (so old points drop off the left-hand edge of the plot). 1 to 'grow' the time axis, 0 to have moving window	1
<code>window</code>	Width of plotting window in hours, also initial width when <code>growt</code> is true	0
<code>tunits</code>	Units for time axis, allowed values are: <code>unit</code> (MJD), hours, radians (rendered as hh:mm:ss with AST plotting), days	<code>unit</code>
<code>plottitle</code>	Text string with chart title	
<code>data</code>	Comma-separated list of monitor names. These must correspond to a top-level monitor name	
<code>linestyle</code>	Comma-separated list of linestyles (0 or none for no lines)	<code>solid</code>
<code>linecol</code>	Comma-separated list of line colours	<code>yellow</code>
<code>symbol</code>	Comma-separated list of plot symbols (0 or none for no symbols)	<code>open circle</code>
<code>symcol</code>	Comma-separated list of symbol colours	<code>green</code>

Table 1: List of supported chart parameters.

The first three make use of the Starlink AST<sup>2</sup> library and offer greater flexibility than using PLplot alone (for example, PLplot does not allow more than 4 monitors per chart). If no plotting class is specified, then the data are written to `STDOUT`. Note that multiple devices may be specified, but currently the same data are plotted on each device.

### 3.3 Chart parameters

The chart parameters control the configuration of each chart plotted on the stripchart, including scaling and plot colours, symbols and styles. Table 1 lists the chart parameters along with a description and a default value.

Not all values are necessary since sensible defaults are chosen. However, for the parameters which take a list of values there *must* be values for each monitor specified on the `data` line. The supported plot styles and colours are described further in § 4.3 below.

### 3.4 Monitor parameters

Supported monitor parameters are listed in Table 2. Note that there are no default values for any monitor parameters. There are currently four supported monitor classes: `ORACIndex`, `Simple`, `WVM`, and `Random`.

The `ORACIndex` and `Simple` monitors are designed to monitor and read files in the `ORACIndex` and `Simple` formats (see § B for details of the `Simple` file format). They also need to be given



Variable	Description
<code>monitor_class</code>	Either <code>ORACIndex</code> , <code>Simple</code> , <code>WVM</code> , or <code>Random</code>
ORACIndex-specific parameters:	
<code>indexfile</code>	name of ORAC Index file to monitor
<code>column</code>	column to plot in index file
<code>filter_*</code>	additional information to retrieve specific data (e.g. <code>FILTER</code> to select the wavelength, <code>UNITS</code> to specify the units, <code>OBJECT</code> to specify observations of a particular source etc). The filter qualifiers are equivalent to column names in the ORAC Index file.
<code>cvtsub</code>	Name of a subroutine to be called to post-process data returned by the monitor before it is returned to the plot system. Currently only two options are available to convert skydip-derived optical depth at either 850 or 450 $\mu\text{m}$ to the optical depth at 225 GHz.
Simple-specific parameters:	
<code>filename</code>	Simple file to be monitored
<code>tcol</code>	Column in file to treat as time axis
<code>ycol</code>	Column in file to treat as $y$ -axis
<code>tformat</code>	Format for time data. Supported values are described in §B.
Random-specific parameters:	
<code>randmin</code>	Lower limit for random number interval
<code>randmax</code>	Upper limit for random number interval

Table 2: Description of supported monitor parameters.

the names of the files to be monitored along with column information in order for the monitor to retrieve the desired data.

The `Random` monitor generates random numbers uniformly over the specified interval at the polling rate (currently 1 second). The `WVM` monitor reads `WVM` data from the real-time output data file at the `JCMT`.

The `WVM` monitor needs no further parameters.

The input files for the `ORACIndex` and `Simple` monitors are specified either by a path relative to the current directory (e.g. `datafiles/mydata.dat`) or an absolute path (e.g. `/data/mydata.dat`). If no path is given then it is assumed to lie within the current directory or in the directory specified by the environment variable `ORAC_DATA_OUT`.

**Req. DR17**

### 3.5 Example `chart.ini`

The `stripchart` configuration file shown below divides the display window into a  $1 \times 2$  array of panels, and uses the `AST::PGPLOT` plotting class. There are two charts to be plotted: `chart1` and `chart2`. `chart1` plots data from two monitors, one of which is an `ORACIndex` file, the other is a `Simple` file. The  $y$ -scale is set manually and the chart uses a fixed window 30 minutes in length: thus only the most recent 30 minutes' worth of data are displayed. The time axis units are hours since the beginning of the `MJD` containing the first data point.



The `orac1` monitor is to be plotted with cyan crossed joined by a grey dot-dashed line. The `test2` monitor is to be plotted as white filled circles joined by a pink dotted line. The `orac1` monitor selects the values of the GAIN for the 850W filter (in units of per arcsec<sup>2</sup>, as opposed to per beam) in the file `orac1.dat`. The `test2` monitor retrieves data from the file `test3.dat` using column 1 as the time axis and column 3 as the  $y$  axis. The time format is ORACTIME (same as used in ORACIndex files).

`chart2` plots data from a single ORACIndex file called `index.gains` using default values for the plotting attributes (green open circles joined by a solid yellow line). The  $y$ -axis is autoscaled. Note in this case that both `growt` and `window` are specified which means that initially the plot will span a window of 10 hours, but once more than 10 hours' of data have arrived then the time axis will scale to plot the full range of values. The time axis units are days since the beginning of the MJD of the first data point.

```
[globals]
nx=1
ny=2
output_class=AST::PGPLOT

[chart1]
data=orac1,test2
autoscale=0
yscale=0,10
yunits=Jy
growt=0
window=0.5
plottitle=Testing two monitors
linestyle=ddash,dot
linecol=grey,pink
symbol=cross,fcircle
symcol=cyan,white
tunits=hours

[chart2]
autoscale=1
yscale=0,1.5
yunits=Jy
growt=1
window=10
data=fcf850
plottitle=An index plot
tunits=days

[fcf850]
monitor_class=ORACIndex
indexfile=index.gains
column=GAIN
filter_UNITS=ARCSEC
filter_FILTER=850W

[test2]
monitor_class=Simple
filename=test3.dat
tcol=1
```



```
ycol=3
tformat=ORACTIME

[orac1]
monitor_class=ORACIndex
indexfile=orac1.dat
column=GAIN
filter_UNITS=ARCSEC
filter_FILTER=850W
```

## 4 User interface

In order to run a stripchart, a configuration file must exist which contains the parameters for the charts to be plotted along with the names of the data sources to be monitored. A detailed description of the format of the configuration file is given in § 3 above. Template and example configuration files are located in the `configs` directory in the stripchart installation.

The stripchart is started from the command line:

```
% stripchart
```

or

```
% xstripchart
```

An example stripchart is shown in Figure 1.

To terminate the program, type `ctrl-C` on the command line, or click on the ‘Exit’ button (if using `xstripchart`). The configuration file can not be changed once a stripchart is running; however it is possible to adjust some of the parameters interactively (§ 4.2).

### 4.1 Command-line arguments

The stripchart has a number of optional command-line arguments:

**-cfg** <*config\_file*>

specify the name of the configuration file to use to configure the stripchart. By default the stripchart will use `chart.ini`.

**-help**

print help information

**-version**

print the stripchart version number



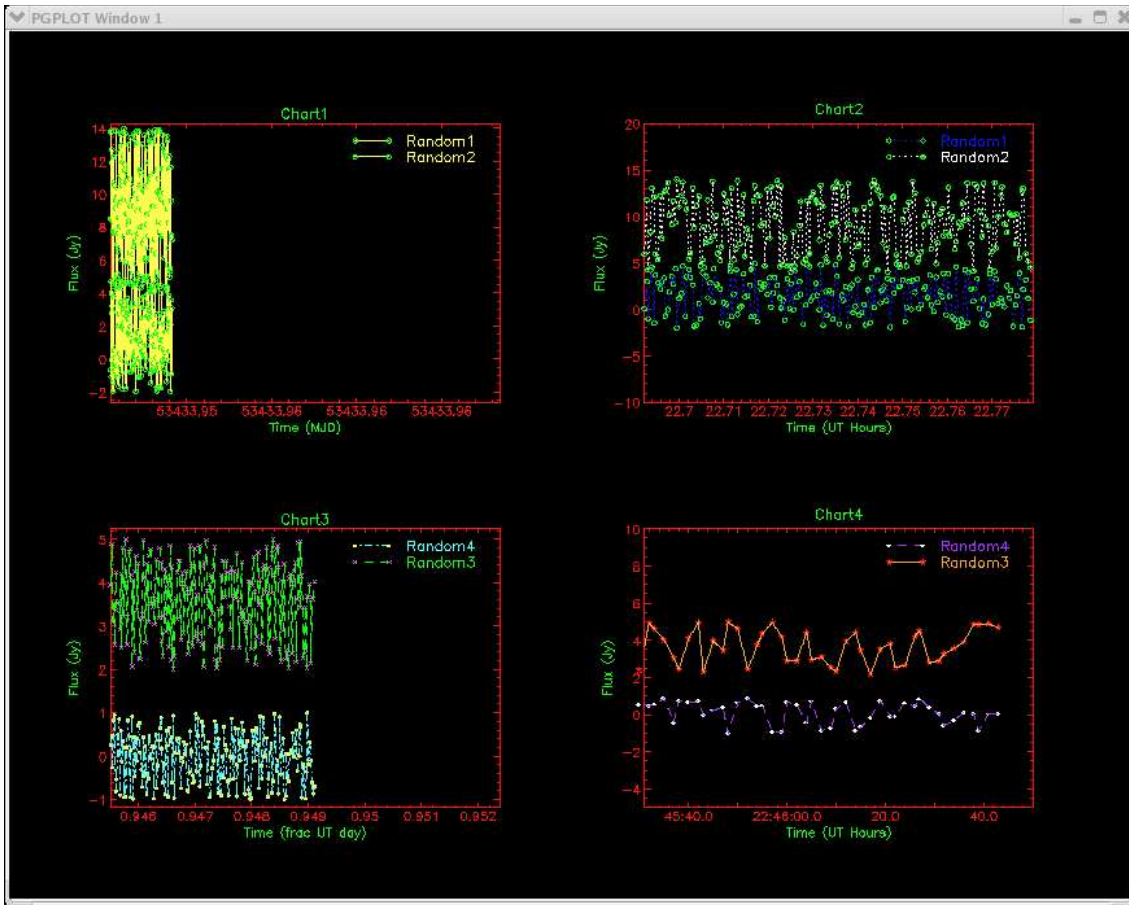


Figure 1: Example stripchart showing four charts, each with 2 monitors.

**-man**

print the documentation to STDOUT

**-dev**

By default, the device class is read from the configuration file. If this option is specified, the configuration file will be overruled and all charts will be created using this device class instead. Only AST-based devices can be selected in this way.

Minimum-matching is supported for each argument (e.g. `-h` is equivalent to `-help`).

## 4.2 Interactive control with xstripchart

xstripchart offers the ability to change some of the chart attributes (such as window or autoscale) interactively without editing the configuration file (see Figure 2). If the `AST::Tk` plotting class is chosen then this menu is embedded in the Tk canvas, otherwise it floats freely as a separate window to the stripchart.

For each chart to be displayed, xstripchart draws a tab which can be clicked on to access the available attributes. The window length may be adjusted, and the plot will update



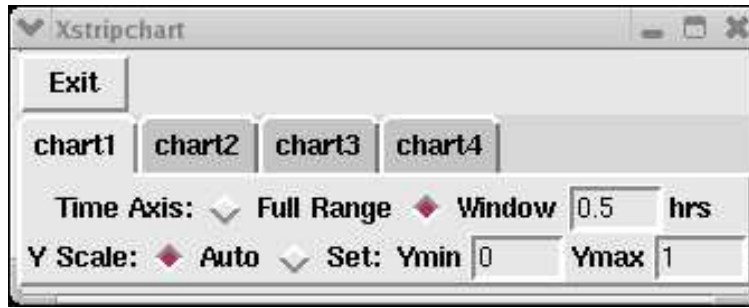


Figure 2: xstripchart control window, showing tabs to control the attributes of four charts.

accordingly. The *y*-axis scale may also be adjusted. Note that the stripchart will update automatically within 1 second of values being entered and does not rely on a carriage return to signify the end of input.

The 'Exit' button halts the stripchart and closes the control and stripchart windows.

### 4.3 Monitor attributes for each plotting class

Each plotting class has different colours, line styles and symbols. The supported keywords for these attributes are listed in Table 3, with some exceptions noted. Note the keywords are merely convenient names used to set the internal integer index number: for PGPLOT and PLplot the internal index number may be substituted instead of the keyword (e.g., the colour red in PGPLOT is predefined with a colour index of 2). Alternative spellings for grey (i.e. gray) are supported. At this stage, it is not possible to set colours as a hex triplet (e.g. #rrggbb). The default stripchart background is black.

Colour	Symbol	Linestyle
red	square <sup>1</sup>	solid
yellow	dot	dash
green	plus	dot
grey	asterisk	dash-dot <sup>2</sup>
blue	circle	dash-dot-dot <sup>2</sup>
cyan	cross (x, times)	long-dash <sup>3</sup>
magenta	triangle	
white	diamond	
pink	star	
springgreen <sup>2</sup>	fcircle	
purple <sup>2</sup>	fsquare <sup>2</sup>	
darkgrey <sup>2</sup>	ftriangle	
chartreuse <sup>2</sup>	fstar	
orange <sup>2</sup>	fdiamond <sup>2</sup>	

Table 3: Supported monitor attributes with exceptions noted. These are the standard colours and symbols to be supported by AST. Symbols beginning with 'f' are filled (solid) versions. Not all are supported for all devices; alternative values are substituted. <sup>1</sup>Square can be specified by name only. <sup>2</sup>Not PLplot. <sup>3</sup>PLplot only.



## A Stripchart design and structure

The stripchart employs an object-oriented design implemented in Perl for ease of integration with the ORAC-DR infrastructure (although the stripchart does not rely on ORAC-DR running). The programming API is documented in the code itself and there is no need to duplicate it here. A diagrammatic representation of the stripchart design and class hierarchy is shown in Figure 3.

The essence of the data flow in the stripchart can be summarized as follows: Monitors retrieve data from the data sources, passing them to the Charts which in turn pass them to the Sinks (which know how to plot them on the Devices). The StripChart is the top-level class which stores Chart, Config and Device objects, and provides the method to tell the Charts to query the Monitors and pass on any data to the Sinks. Most of the communication is done through the Chart objects and associated methods. Errors are handled through the Error class.

Upon startup, the stripchart reads the configuration file and stores basic configuration parameters in the Config object. The names of the relevant Chart and Monitor objects as well as Sink class(es) are determined from the relevant sections of the configuration file (see § 3 above). The stripchart loops through the names of the charts, attaches Sink objects of the requested Device class to the Charts, reads and attaches the plot attributes to the relevant Sink objects (since it is the Sink classes that communicate directly with the Devices). Monitors (and their own attributes) are attached to (stored in) the associated Chart object.

If `xstripchart` is being run, it creates the control panel window at this point (running as a Tk Event object). Next the plotting surface(s) is (are) defined and configured with the appropriate Device object, the Subplots are attached to the Sinks and the stripchart window is drawn with the correct number and placement of subplots.

The stripchart main loop takes the Device objects and their associated Chart objects and gets them to poll their Monitors to check for data to plot. Data will be plotted subject to the time and  $y$ -axis limits determined in the configuration file. The Monitors keep an internal track of the last-read data so that only the new data will be returned and plotted when they arrive.

The data to be plotted are then passed to Sink objects, which in turn relay only the data which need plotting to the relevant Device object. The AST-based Sink objects make use of TimeSeries objects to store the data in the event the chart is rescaled. (The non-AST PLplot version does not use a TimeSeries object because PLplot caches the data itself.) To allow for different choices in displaying the time axis, a TimeMap object is attached to the Sink which carries out the conversion between the internal time format (MJD) in the TimeSeries object and the requested output format as defined in the configuration file.

Once the new data have been plotted, control returns to the main loop, whereupon the polling cycle begins again. The program exits when either the 'Exit' button on the Tk control panel is clicked or when `ctrl-C` is typed at the shell prompt.

### A.1 Class hierarchy

StripChart: the base class contains methods for establishing and updating a stripchart. Stores Config, Chart and Device objects.

- Config: an object used to store stripchart configuration details.



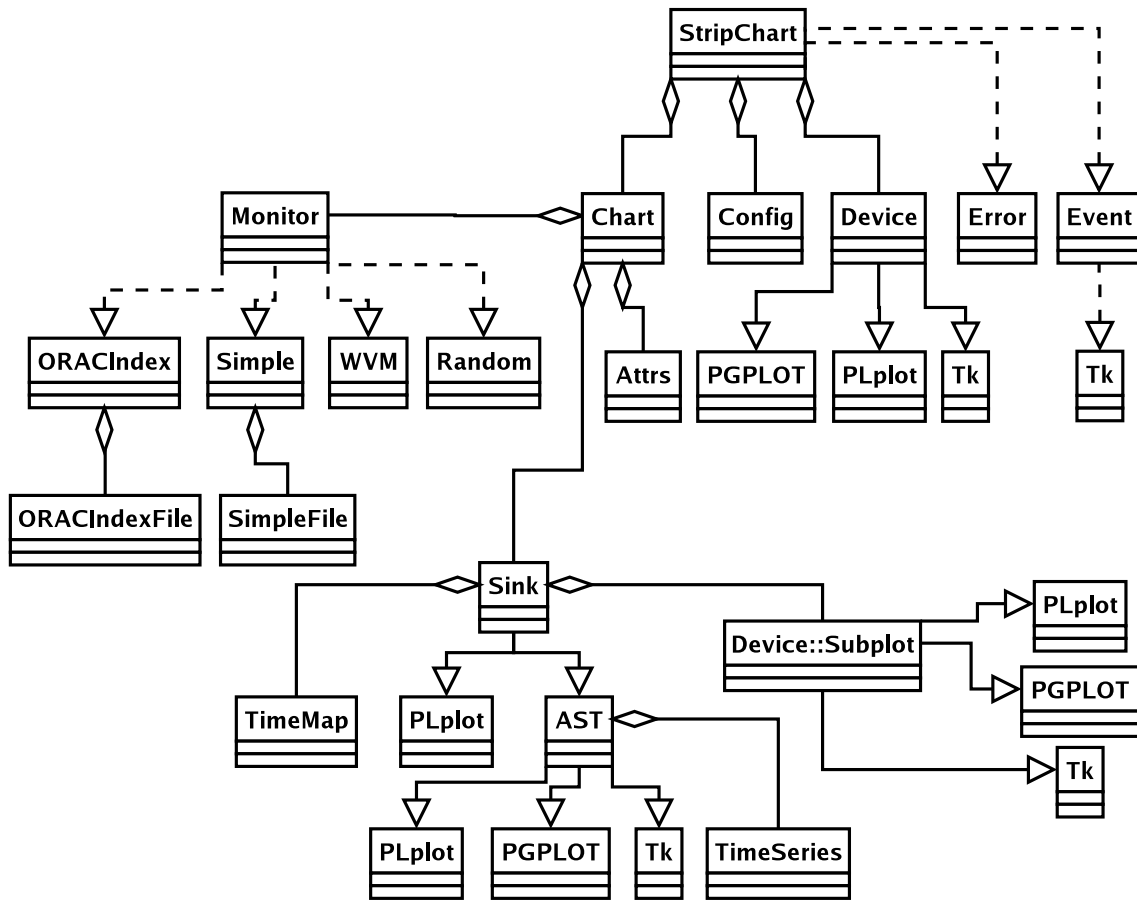


Figure 3: Class hierarchy. The links between classes show the relationship: solid + diamond = contains, solid + arrow = inherits from, dashed + arrow = implements interface.

- Error: contains error handling methods
- TimeSeries: an object which stores the (time, data) pairs for a given monitor. Contains methods for accessing properties of the data (e.g. min, max etc).
- TimeMap: methods for converting MJD into a more human-readable time format for plotting on the time axis of the chart.
- Chart: an object which stores and manages the data sources (monitors) and destinations (sinks).
  - Attrs: an object which stores the monitor attributes specified in the config file
- Device: Base class with methods for handling plot devices
  - PGPLOT: Subclass with PGPLOT-specific routines for initializing and selecting plot devices (inherits from Device)
  - PLplot: Subclass with PLplot-specific routines (inherits from Device)
  - Tk: Subclass with Tk-specific routines (inherits from Device)
- Device::Subplot: Base class with methods for handling subplots on different plot devices
  - PGPLOT: Subclass with PGPLOT-specific routines for dealing with subplots (inherits from Device::Subplot)



- PLplot: Subclass with PLplot-specific routines (inherits from Device::Subplot)
- Tk: Subclass with Tk-specific routines (inherits from Device::Subplot)
- Sink: Base class to create a Sink object
  - AST: Subclass to deal with AST-based sinks (inherits from Sink)
    - \* PGPLOT: Subclass with PGPLOT-specific routines (inherits from AST)
    - \* PLplot: Subclass with PLplot-specific routines (inherits from AST)
    - \* Tk: Subclass with Tk-specific routines (inherits from AST)
  - PLplot: Subclass to deal with the PLplot native stripchart routines (inherits from Sink)
- Monitor: monitor classes
  - ORACIndex: Class for dealing with ORAC Index-format files
  - ORACIndexFile: Provides low-level access for reading Index-format files
  - Simple: Class for dealing with Simple-format files
  - SimpleFile: Provides low-level access for reading Simple-format files
  - WVM: Class for reading WVM data
  - Random: Class for generating a random number stream
- Event
  - Tk: Tk event object to deal with input to the `xstripchart` control window

## B simple file format

The `simple` file format is exactly as its name implies, simple. Its purpose is to allow external programs to be run and generate data which can be monitored by the stripchart. Data from such programs will typically be tracking variables not generated by the Pipeline, allowing additional diagnostic information to be plotted.

The file should contain a minimum of two space-separated columns of numbers, one of which is the time corresponding to the value in the other column(s). Comments may be inserted on lines prefixed with a #.

The permitted time formats for the `Simple` file case are

**ORACTIME** – same time format used in ORAC Index file (yyyymmdd.dddd)

**DMY** – dd/mm/yyyy/hh/mm/ss.sss

**MDY** – mm/dd/yyyy/hh/mm/ss.sss

**YMD** – yyyy/mm/dd/hh/mm/ss.sss

**HMS** – hh:mm:ss.sss

**MJD** – Modified Julian Day number, d.dddd

The separator may *only* be one of three commonly-used types: / (forward slash), : (colon), – (single dash). Only a single separator can be used for each date/time value. White space is not permitted as a separator. If HMS is specified then it is assumed to be on the current UT date.



## References

- [1] Douglas Scott. SCUBA-2 data reduction software requirements. SCUBA-2 Project SC2/SRE/S210/001, 2004.
- [2] Rodney F. Warren-Smith and David S. Berry. AST – a library for handling world coordinate systems in astronomy. Starlink User Note 211, Starlink Project, CCLRC, 2002.

