UK Radio Astronomy Association

Starbase

User Guide

```
VlfReceiver.Core.reset()
VlfReceiver.DataCapture.getSpace()
VlfReceiver.DataCapture.setRate(1)
VlfReceiver.DataCapture.capture(true)
VlfReceiver.DataCapture.getData(Staribus, PassThrough)
VlfReceiver.Exporter.exportRawData(filename, timestamp, dataformat)
VlfReceiver.Exporter.exportChart(filename, timestamp, format, width, height)
```

Acknowledgements

Contributors

The following authors have contributed to the
Starbase User Guide
Andrew Lutley, Alan Melia, Laurence Newell.

Table of Contents

1 Introduction

Starbase is a general-purpose Java software package developed by the UK Radio Astronomy Association (UKRAA), intended for the collection, analysis and storage of data from multiple data sources, specifically for astronomical applications. The current release of the software contains an Observatory, with facilities to collect data from local instruments or remotely over the Internet. Planned future developments include a data repository, and an analysis package. Starbase is platform independent and will run under MS Windows, Linux, or MacOS.

The Starbase Observatory connects to a variety of instruments, presenting their data in a consistent way, and allowing storage and retrieval from the local file system. The user interface 'metaphor' is that of a programmable oscilloscope or signal generator, and so should be familiar to most users of radio equipment. The command interface is specifically intended to be scriptable, to facilitate building macros (“batch files”), or running unattended: the macro facility is under development. The Observatory communicates with Instruments either via the serial port with RS-232 or RS-485, or via an Ethernet connection (*e.g.* FTP, HTTP, UDP). Any new instrument may be added to the Observatory but it must first be described in an XML file. Some additional Java software may also be necessary, depending on the functionality required.

Please note that the Starbase software is **not** intended to replace the hardware of a realtime data logger, since the actions of repetitively retrieving data samples and displaying the results, perhaps from several instruments simultaneously, can be very processor intensive. If possible, it is usually better to log offline with external hardware, and download the data in one operation.

Starbase has a unique design, with an extensible *fractal* architecture, which is fully configurable by editing XML files that follow a strict structure (a 'schema'). Some customisation will be required for your particular needs; this is readily achieved by some simple editing of text files as will be explained in this document.

Starbase is provided free of charge with a licence to use, not to own. The Java source code is provided under the General Public Licence (GPL). The architectural schemas are provided to users to allow full use of the XML configuration. Starbase also contains many third-party libraries, the individual licence terms of which are included in the distribution. If you would like to be involved in the Starbase development, then please contact `starbase@ukraa.com`

2 Software Installation

2.1 Java Runtime Environment

`jre7-windows-i586.exe`

Starbase is developed using the Java language, from Sun Microsystems, now distributed by Oracle. In order to run Starbase you need the 'Java Runtime Environment' (or JRE) installed on your PC. This *must* be Java v1.6 or later (also known as 'Java 6'). If you use an earlier version you are almost certain to see runtime errors caused by a mismatch between the code in Starbase and that in the Java libraries. Java is available for Windows, Linux and the Apple Mac: the JRE is available directly from Sun, or you may have been provided with the latest version on your distribution CD.

http://java.sun.com/javase/downloads/index.jsp

If you are unsure of the version of Java installed on your PC, then open a command window and type the following command:

```
java -version
```

In Windows, open a DOS window (run cmd). In Linux, Ubuntu for example, select "terminal" from the Accessories drop-down menu. There may be several version of java installed and you may need to ensure that the version 6 is at the top of the list, or set as the preferred version.

2.2 Installation of Starbase

`Starbase-Windows-Installer.exe`

- Download the Starbase installer appropriate to your operating system from the UKRAA website, or from the distribution CD.
  The latest build is available from: www.ukraa.com/www/downloads.html

- Run the installer and follow the instructions carefully

- **Read** the ReadMe file appropriate to your platform's operating system:

  - `dist/ReadMe-Windows.txt`
  - `dist/ReadMe-Mac.txt`
  - `dist/ReadMe-Linux.txt`

- **Read** the release notes for this build of the software:

  - `dist/ReleaseNotes.txt`

- You will find a lot of other useful documents in `dist/doc`

- If you have problems with installing Starbase,
  please contact `starbase@ukraa.com`

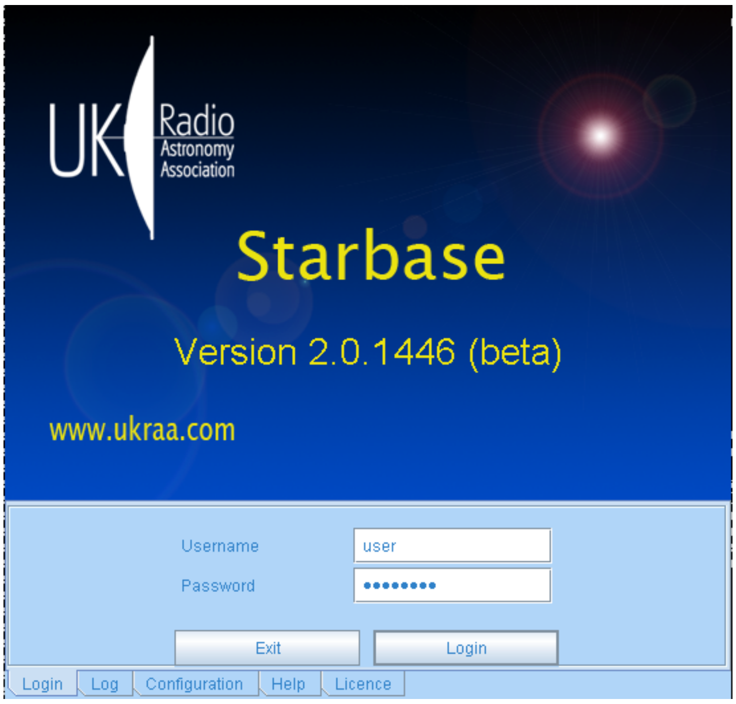
Please remember to include the Starbase version and build number (found on the login panel, or on any menu), and any `EventLog` information available.

If you decide to go ahead and do your own configuration outside of the installer, please note that when editing the configuration files you must use a plain text editor such as Windows Notepad,
***not** a word processor!*
*In XML files, remember to use forward-slash (/) in all pathnames, regardless of the host platform! This is because the backslash will be treated differently by the XML file. Edit only those values between ">" and "</", e.g. <Name>**EditThisValue**</Name>*

3 Running Starbase Observatory for the First Time

3.1 Logging In

In order to run Starbase, you should click the shortcut on the desktop, or execute the command file appropriate to your operating system (on Windows, `starbase.bat`, on Linux `starbase.sh`). You should see the following panel, and possibly some activity in a DOS or console window, if your configuration uses one. These messages will be helpful when solving installation problems (*e.g.* on Linux, run `starbase.sh` from the terminal window).



The Starbase Login panel

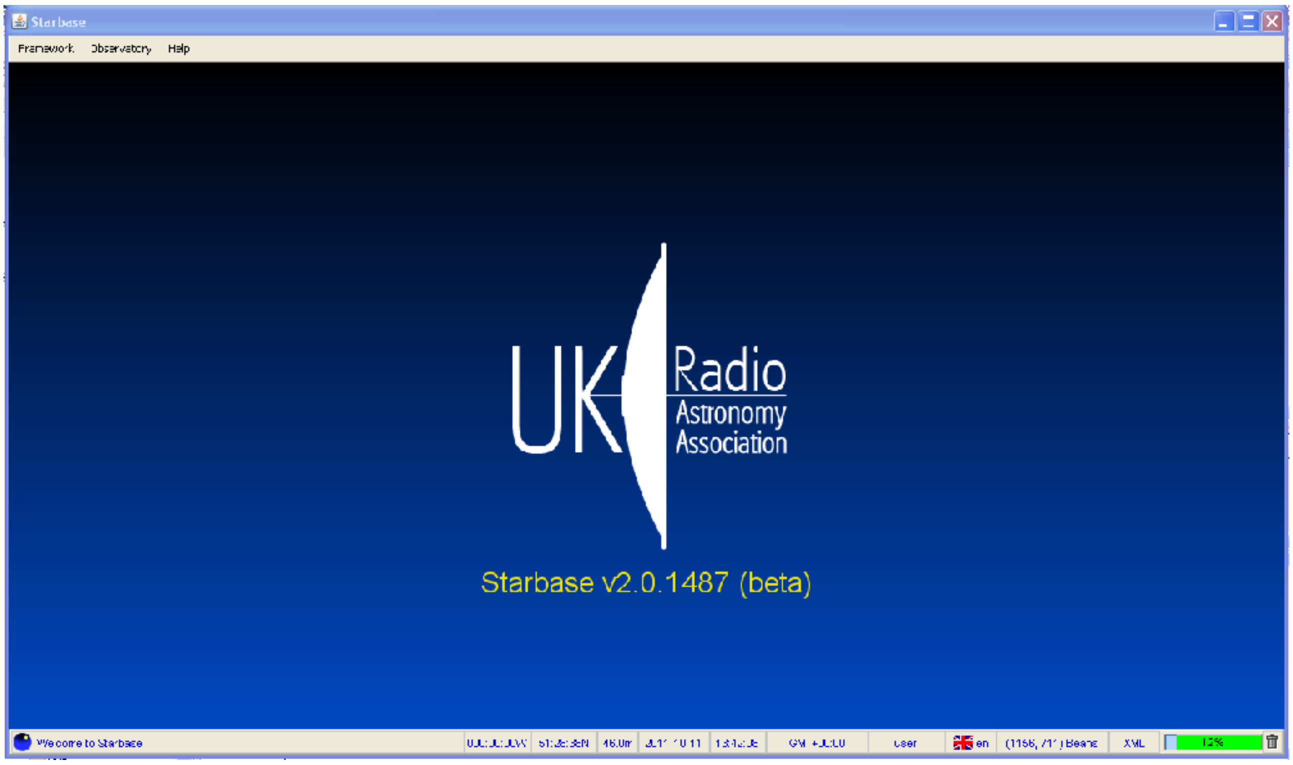
Log In using the username and password you have been given (or use the defaults `user` and `starbase`). Make a note of the Version Number, and quote this if reporting any software problems to starbase@ukraa.com. The version number shown above is for example only, your installation will be more recent.

The **Log** tab on this panel will show messages concerning the various stages of loading the program. You may copy text from this tab by selecting using the mouse, and using `control-C`. This feature may be useful if you are reporting a fault. The **Configuration** tab shows the contents of the `loader.properties` file referred to later in the Starbase Software Configuration section. The **Licence** tab shows a copy of the terms of the GPL; running the software implies acceptance of these terms.

3.2 The Starbase User Interface

Following login, you should see the screen below, which may be resized to suit your display. This is the *Framework*, in which *plugins* such as the Starbase *Observatory* can be displayed. The Framework is a general-purpose software 'container' which makes no assumptions about the nature of its use, and is configured via several XML files.

The status bar shows information such as the current language, country, location, and the time of the host platform's clock (NOT the Observatory Clock, since the Observatory plugin may not yet be running). You may know the host's time as 'Wall Time', *i.e.* that displayed by a wall clock. There is also a display of the amount of memory currently in use.



The Framework screen

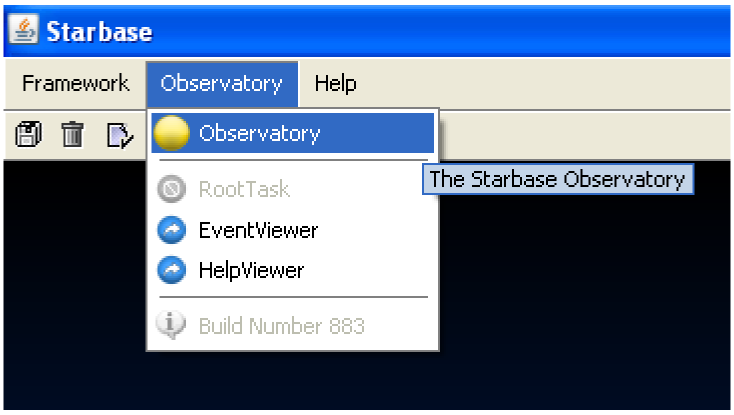
The initial Framework location (latitude, longitude, height above sea level) is encoded in the `frameworks.xml` file, and can be updated automatically if a GPS receiver is available in the Observatory. Clicking on the 'waste bin' in the lower right corner will begin a memory management process, which may improve performance if the memory usage goes above about 70%. See a later section for details on how to optimise the use of memory.

3.3 Logging Out

The Framework menu also contains an option to Log Out of the Framework. This will return you to the login panel, from where you could log in again as a different user. Different users are associated with different *roles*, and each role may have access to different instruments in the Observatory. This feature may be useful for instance in an educational environment, where the potential complexity of the Observatory can be released in a controlled manner.

3.4 Starting the Observatory

The Starbase Observatory plugin is started from the main menu bar, by clicking on the Observatory menu as shown below:



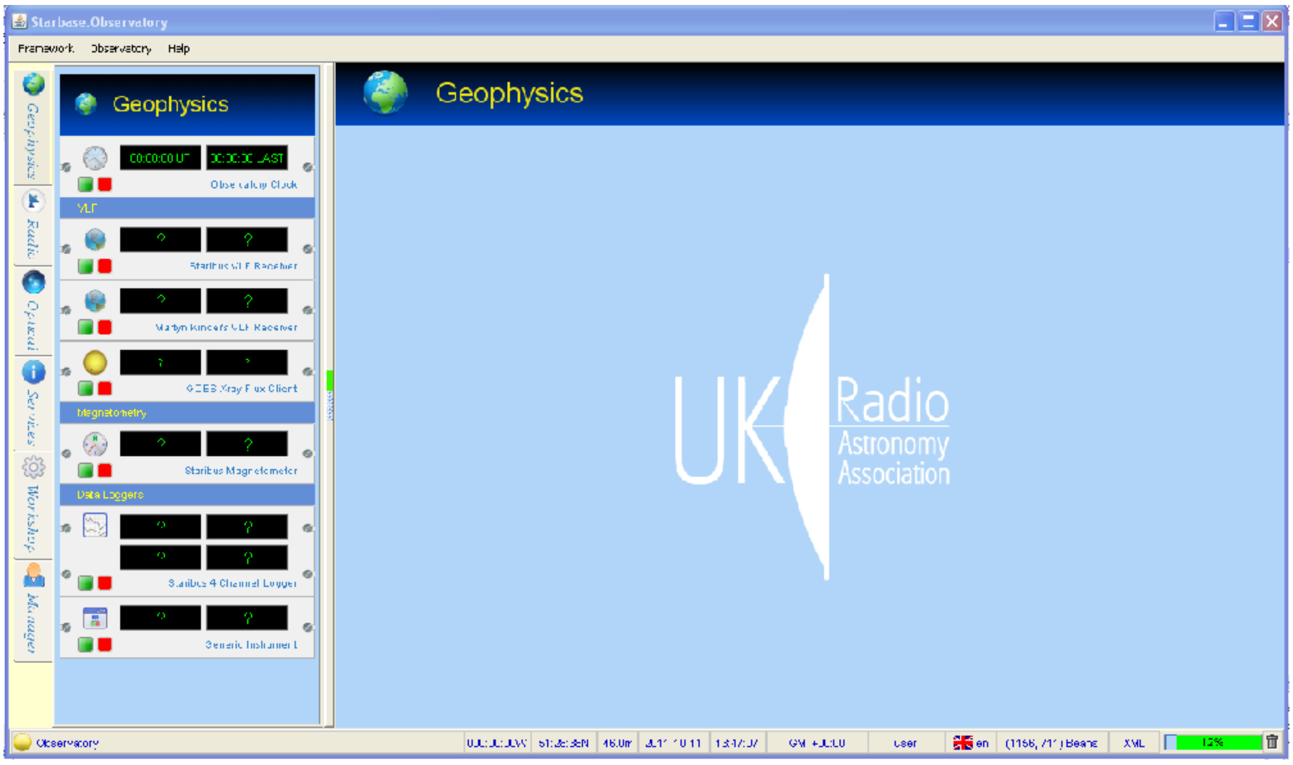
Starting the Observatory

The Observatory will take several seconds to start up, since the instrument XML descriptions have to be read and all of the instruments initialised. In addition, the startup software attempts to find an `ObservatoryClock`, for use by any instrument in the Observatory, and an `ObservatoryLogger`, to which all logging messages are sent. If these instruments are not currently defined, then simple defaults are installed instead.

3.5 Stopping the Observatory

The Observatory may also be **stopped** by `control-click` on the Observatory menu option. This will stop all running commands, stop all instruments, and remove all data and log entries. This feature may be useful to recover from out-of-memory problems or other serious faults. Alternatively, you may wish just to conserve memory while performing another task, but retain the Framework.

3.6 The Initial View of the Observatory



The Observatory initial view

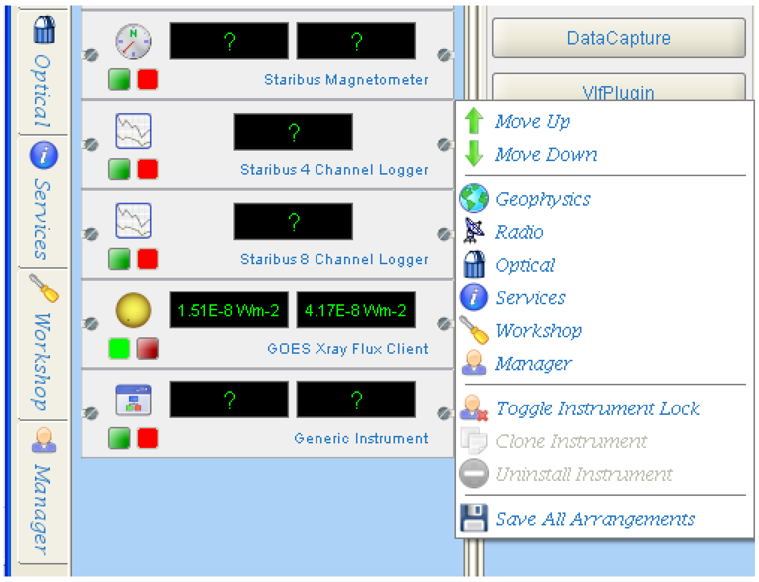
The Observatory is designed to mimic a rack of electronic equipment (on the left), where each panel represents a different instrument. Each instrument has an On (green) and Off (red) button, and a simple display, as would be the case with real equipment. You must select an instrument and switch it on in order use it, after which you will be presented with a a set of buttons which allow you to send commands to the instrument.

The initial **Group** view (left hand side) of the Observatory shows the `Geophysics` group, which contains VLF receivers and data loggers, and a magnetometer.

*Note that although an instrument may appear in several Groups, it is the **same** instrument! So for instance, if you stop the clock in one Group, it will stop everywhere. This applies to all instruments which appear in several groups, they will all show the same data, regardless of which group is used to control them.*

3.7 Customisation of Groups and Instruments

You may rearrange the Instruments in an Observatory Group, or move them to another group, by `right-click` on the Instrument in the rack and selecting an option from a menu. The menu appears as shown below, but not all functions are fully implemented at the time of writing:



Instrument Rearrangement Menu (right click)

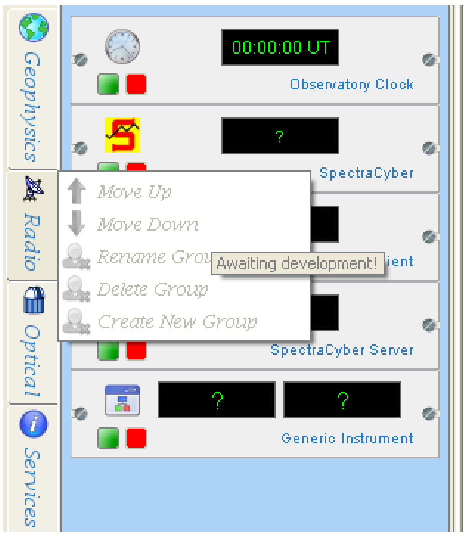
The *Instrument Lock* may be used to protect an Instrument from being accidentally changed, by turning off the `right-click` menu. If you wish to move the Instrument at a later date, then `control-right-click` will make the menu appear regardless of the Lock status of the Instrument.

It is possible to save the new arrangements with the last option on the menu as shown above.

Note that if you install a later distribution of Starbase, you may want to keep a copy of the file containing your new instrument arrangements, and replace the one in the new distribution.
Transfer this file between builds:

`dist/plugins/observatory/imports/instruments-attributes.xml`

It will eventually be possible to manage the Observatory Groups directly from a `right-click` menu, as the screenshot below shows. Currently no operations are implemented, but the menu appears (with `right-click` on a tab) to indicate how the management functions will be achieved.



Observatory Group Management Menu (right click)

3.8 User Feedback

UKRAA develop hardware and software with a policy of 'continuous improvement'. If you have any feedback about Starbase, its user interface, the support for specific data sources, the configuration process, and so on, then please contact:

starbase@ukraa.com

It would be very helpful if you reported any problems together with the *version number* of the software you are running. The version number is readily available on the menus and in the logs. The contents of `EventLogs`, or copies of the command (DOS) window may also be useful.

`EventLogs` for instance can be saved as a timestamped XML file in the `/archive` folder with this *Starscript* command:

```
MyInstrument.Exporter.exportEventLog(archive/EventLog,
                                     true,
                                     FormattedStardata)
```

The next section describes how to execute commands.

4 Making an Observation

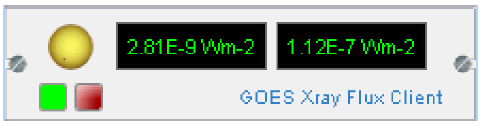
This section describes how to use the Starbase Observatory to make a simple observation, *i.e.* obtain some data from an instrument. This observation may be your own, in that you are responsible for the hardware and software for collecting the data, or it may have been made by another observer, for instance at a remote location. Please make sure that when you publish your work, that you make it clear if you have used data from other sources!

One design goal for Starbase is to allow individual observers to contribute their observations to a central database. Other users of the system can then view the complete set of observations, and perhaps download a selection in order to analyse them in more detail. The functions required to access the database are currently in development.

4.1 Selecting an Instrument

The first step in making an observation is to select an instrument to use. This example will use the GOES Xray Flux Client, a simple instrument which downloads data from the NOAA Space Weather Prediction Centre FTP server at ftp://ftp.swpc.noaa.gov/pub/lists/xray.

The Instrument 'control panel' appears in the Geophysics group as shown below:



The GOES Xray Flux Client Instrument

Note the start (green) and stop (red) buttons. The indicators show the last data values read.

First **select** this Instrument by clicking anywhere on this panel. The right hand side of the screen will change to show the Instrument's commands. **Start** the instrument by clicking the small green button in the lower left corner of the control panel (it will glow brighter). The instrument is now ready for use.

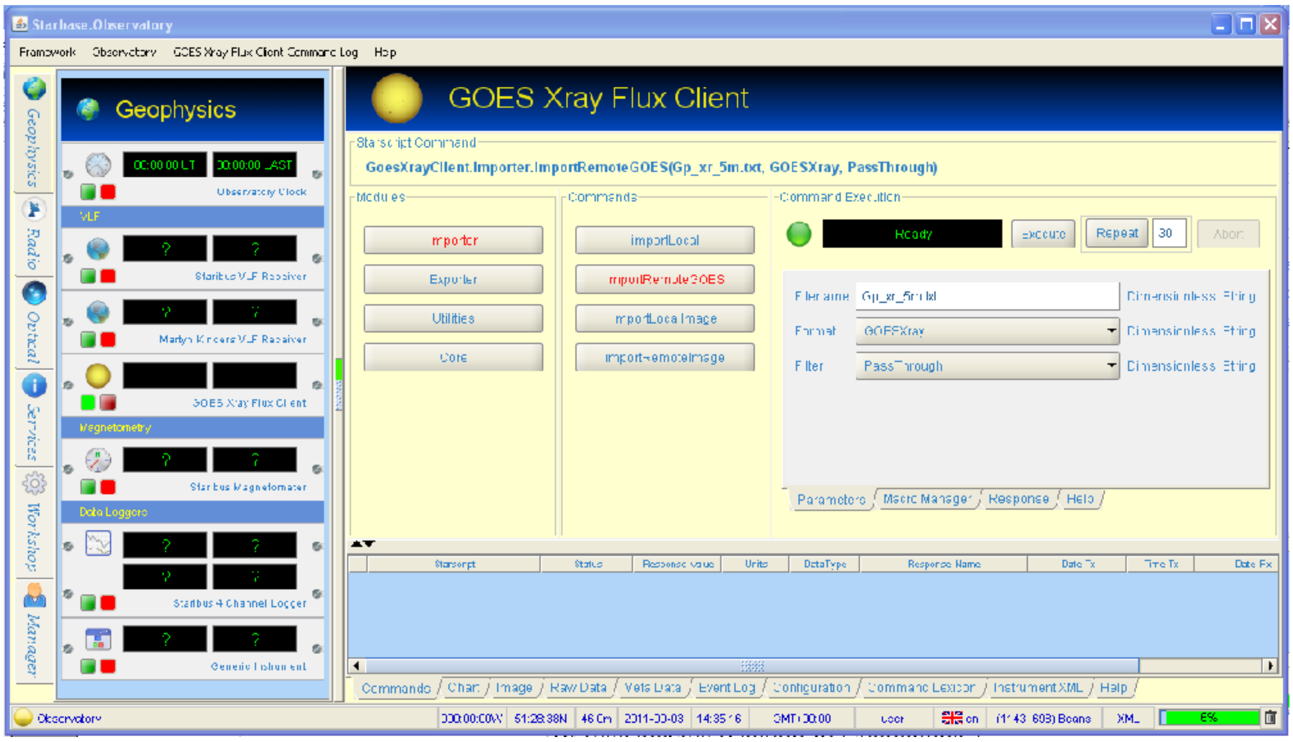
All instruments are stopped automatically if you shut down the Observatory (or log out). The system will attempt to do this in a controlled, safe way, but you may find that you lose data!

4.2 Acquiring Data

Starbase may acquire data in one of two ways:

- ***Data Capture***
  the data are obtained from hardware connected to your computer, for instance a Staribus Logger or VLF Receiver
- ***Importing***
  the data are obtained from an existing file on your computer or on a remote server (e.g. via FTP or HTTP)

The image below shows the selection of the GOES Xray Flux Client. Starbase Instruments (with a very few exceptions) have a *Commands* tab as shown below, which is used to control the Instrument:



GoesXrayClient.Importer.importRemoteGOES(Gp_xr_5m.txt, GOESXray, PassThrough)

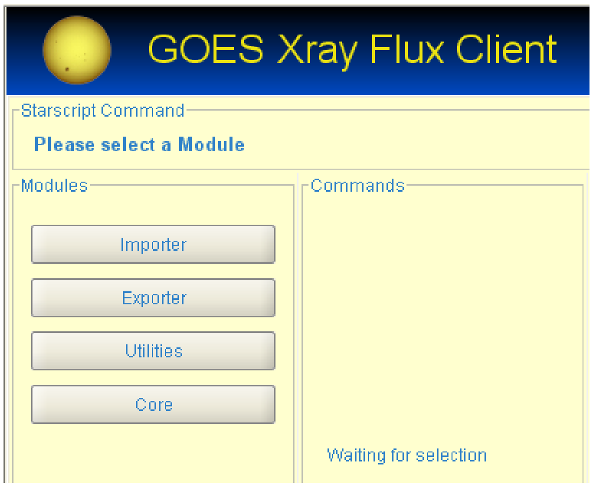
Filename	Gp_xr_5m.txt	Dimensionless String
Format	GOESXray	Dimensionless String
Filter	PassThrough	Dimensionless String

Starscript	Status	Response value	Units	DataType	Response Name	Date Tx	Time Tx	Date Rx

All Instruments respond to Commands

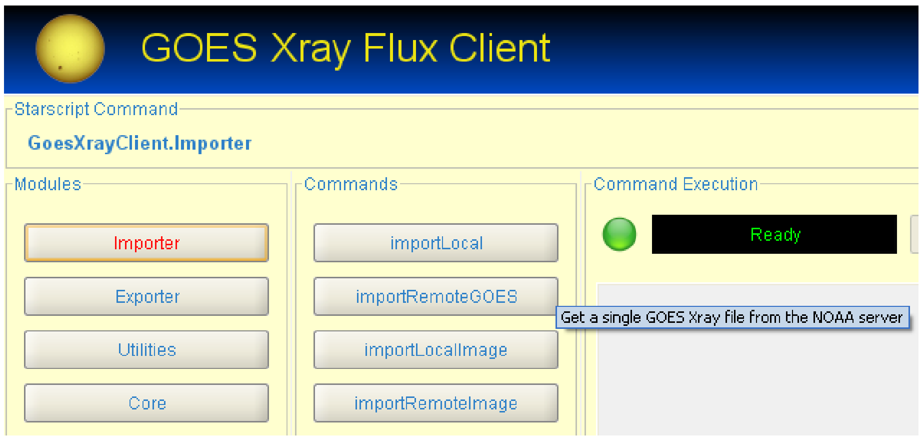
This example will show you how to download some Geostationary Operational Environmental Satellite (GOES) data from the NOAA Space Weather Prediction Centre (SWPC) server.

To set up and execute the command, firstly, select the `Importer` module from the left-hand set of buttons. Notice that this instrument does not have a `DataCapture` module, since your observatory does not contain a satellite!



Select the Importer module

This action will reveal those commands (in the centre set) which may be used to read (*i.e.* `import`) data from several sources. In this case, we will read the data file from the NOAA server, so select the `importRemoteGOES` command:



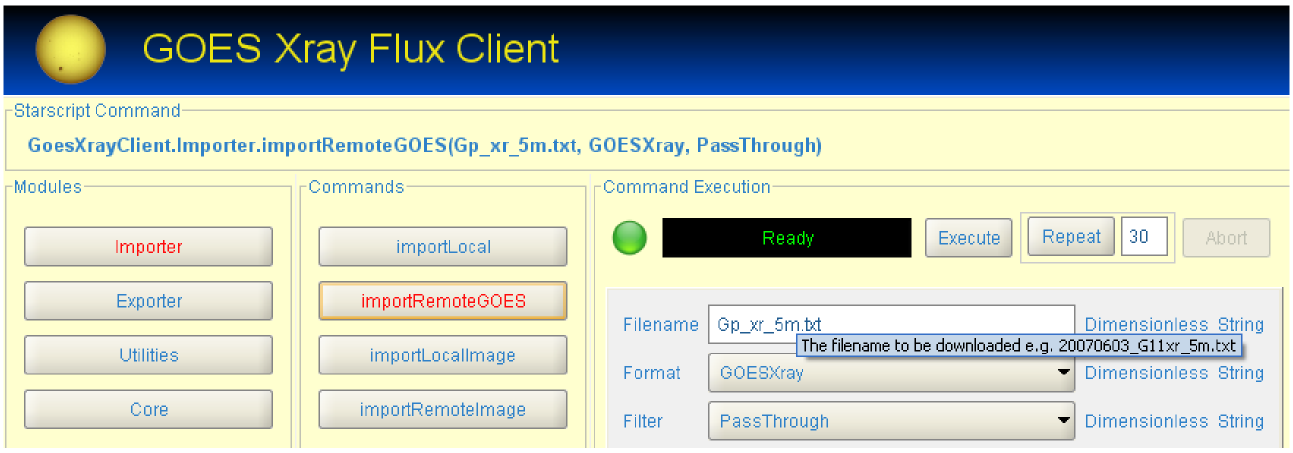
Select the `importRemoteGOES` *Command*

Notice how the command is being built up step by step, as shown in the window above the buttons. The command so far is:

```
GoesXrayClient.Importer.importRemoteGOES
```

This 'command language' is called *Starscript*, and will eventually be extended to allow composite commands to be executed as macros, thus simplifying multi-stage operations, and allowing users to exchange programs.

The next step is to specify the required dataset to download. The instrument Help tab will give further information on the options here. The easiest option is to take the data from the primary (or default) satellite, as shown below.



GoesXrayClient.Importer.importRemoteGOES(Gp_xr_5m.txt, GOESXray, PassThrough)

Filename Gp_xr_5m.txt Dimensionless String
The filename to be downloaded e.g. 20070603_G11xr_5m.txt
Format GOESXray Dimensionless String
Filter PassThrough Dimensionless String

Specify the dataset to download as the `Filename` *parameter*

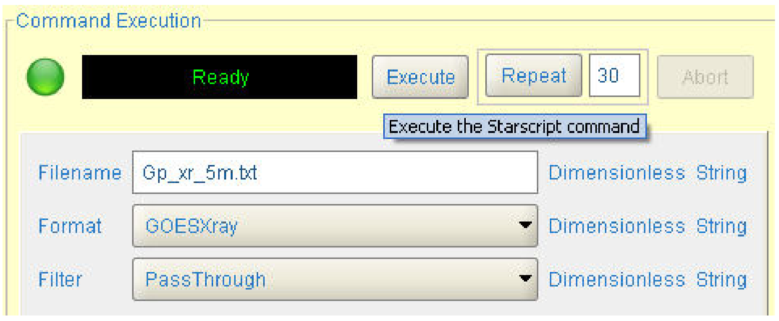
There is only one data format available here, so the `Format` parameter must be `GOESXray`. The `Filter` will either take the data unmodified, or filter with a `SimpleIntegrator`. In this application, it is best to use the `PassThrough` filter, which will not process the data in any way.

The full *Starscript* text of the command is therefore:

```
GoesXrayClient.Importer.importRemoteGOES(Gp_xr_5m.txt,
                                         GOESXray,
                                         PassThrough)
```

The default parameters have been set to obtain the data from the (primary, or default) satellite dataset, as explained in the Help tab.

Finally, execute the command by clicking the **Execute** button.


Command Execution

Ready | Execute | Repeat | 30 | Abort

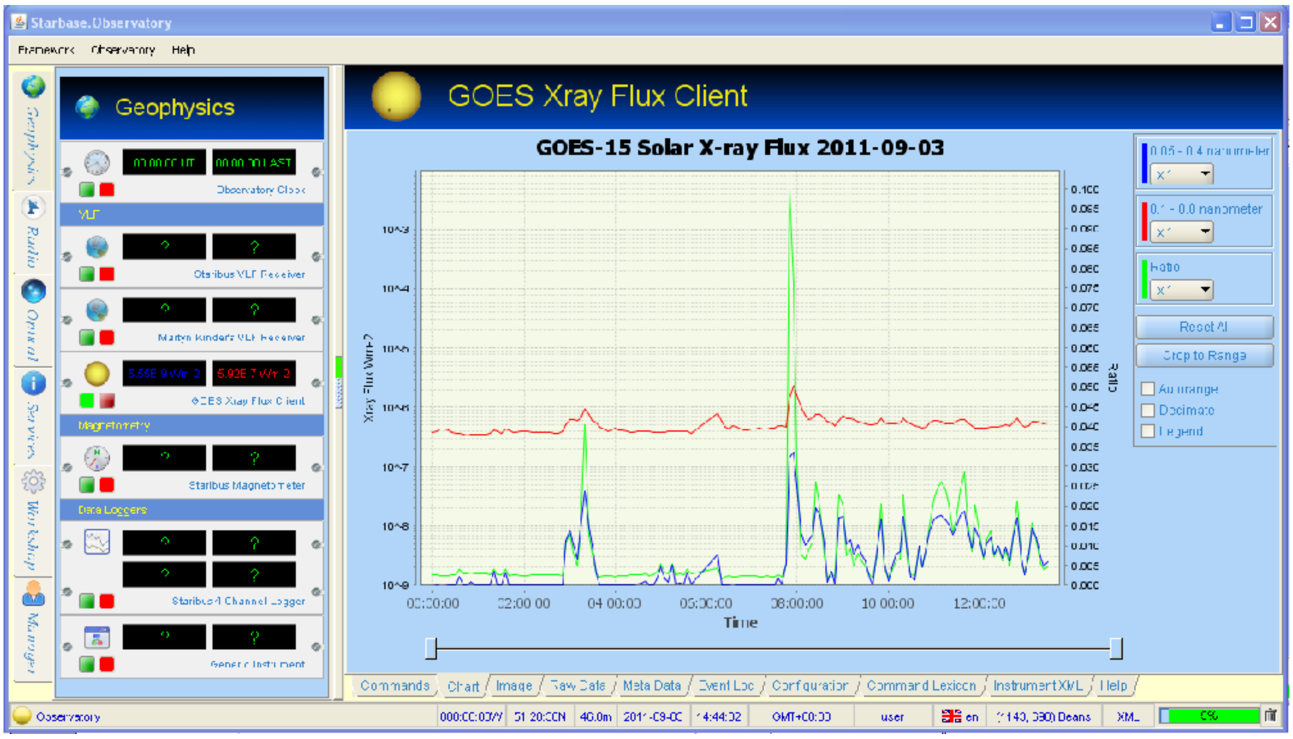
Execute the Starscript command

Filename	Gp_xr_5m.txt	Dimensionless String
Format	GOESXray	Dimensionless String
Filter	PassThrough	Dimensionless String

Execute the Command

Provided that the server is available, you should see a `SUCCESS` Status near the right hand side of the Command Log, and the data will be available on the Chart and RawData tabs.

If all of the previous steps have been followed correctly, then you should see the chart below, giving the X-ray flux from the Sun at two wavelengths, and the ratio of the fluxes in each channel.



Solar X-ray Flux from GOES 14

If you are making solar observations using other instruments, then these charts (and the associated data) are a very useful resource against which to compare your results. It can be very rewarding (and reassuring) to find that you have captured an event which was also seen by the satellite!

4.3 Data Flow, or 'Where does my data go?'

Starbase can read data from many data sources, using commands in the DataCapture or Importer modules. These data are known as RawData, since they are exactly as read from the instrument or file. Starbase can apply *filters* to these data during the capture or import to produce a second dataset, called ProcessedData. The user can select the filter as one of the parameters to the capture or import command. The ProcessedData are used to generate a Chart. Finally, the RawData, ProcessedData and the Chart are all displayed on tabs. These datasets and their relationships are shown in the diagram on the next page

All data sources can have MetaData associated with the RawData, and these data are carried through as shown. MetaData is used to record all aspects of the observation, together with the original data, so that the complete context of the observation may be accurately reconstructed at any time in the future. The MetaData items are chosen from a *MetadataDictionary*, thus ensuring that all observers use the same terminology. The MetaData may be edited in the *MetadataExplorer* tab. Metadata are discussed further in a later section. The edited data are saved with the datasets during any export operation.

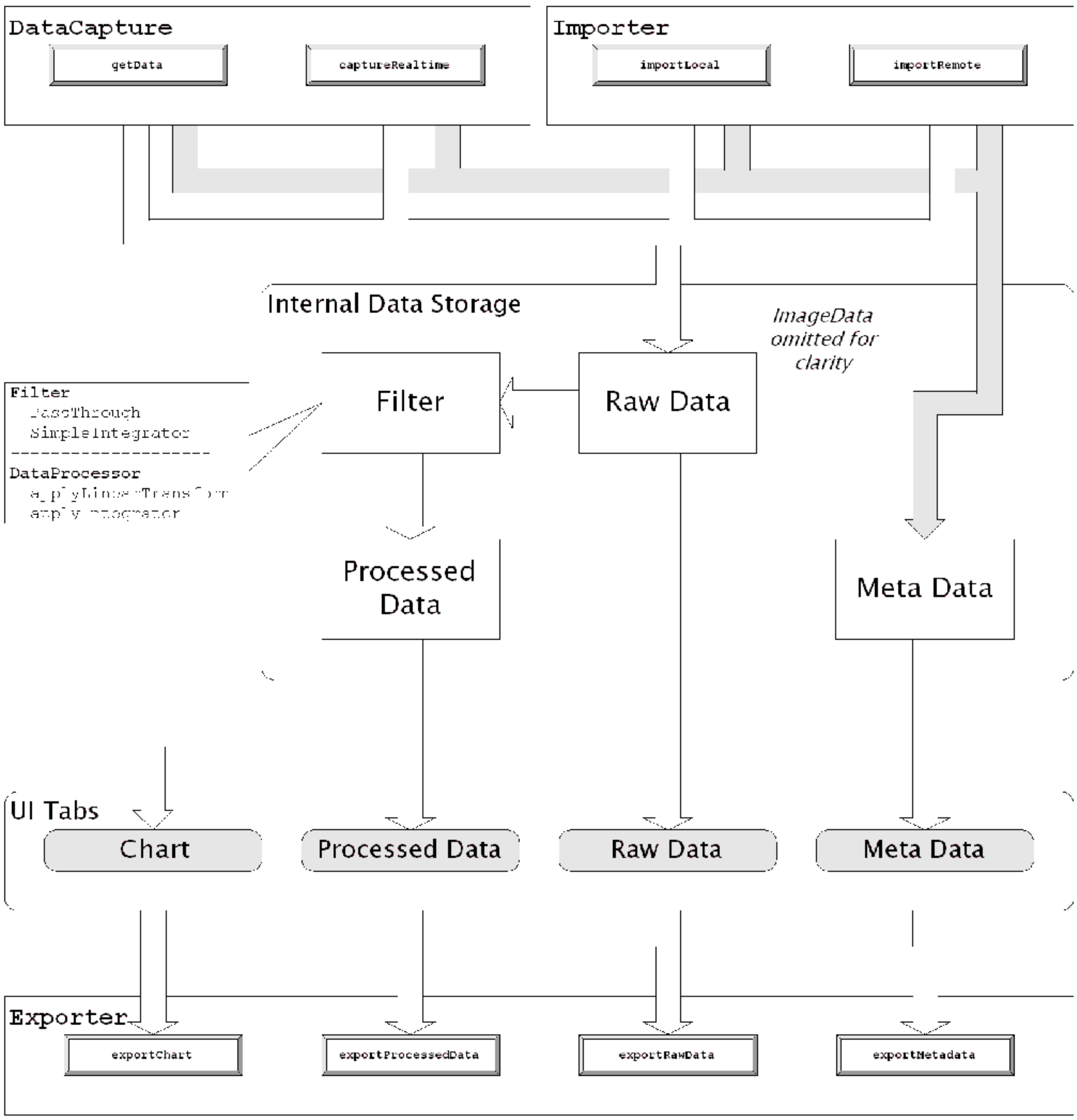
A DataProcessor module can apply other functions to the different datasets, allowing detailed analysis. For instance, data may be scaled and offset, or filtered.

A typical Starbase session may appear as below (parameters omitted for clarity):

```
Magnetometer.Core.ping()
Magnetometer.MagnetometerPlugin.setGain()
Magnetometer.DataCapture.captureRealtime()
Magnetometer.Exporter.exportRawData()
Magnetometer.DataProcessor.applyLinearTransform()
Magnetometer.DataProcessor.applyFilter()
Magnetometer.Exporter.exportProcessedData()
Magnetometer.Exporter.exportChart()
```

UKRAA intend to provide a 'macro' facility for Starbase, so that simple programs such as the example above can be executed on a single button. The user will be able to prepare a complex series of operations for later use, or perhaps share them with other users. The macros will become part of the ***Starscript*** *command language, and may themselves be part of other macros.*

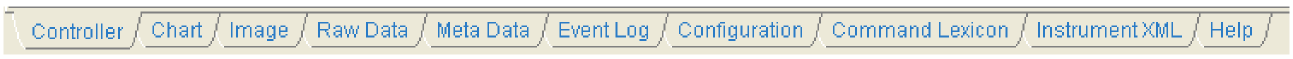
This diagram shows the relationships between `RawData`, `ProcessedData`, the `Chart` and `MetaData`.



Starbase Internal Data Flow

4.4 Data Viewers, or 'What datasets are available?'

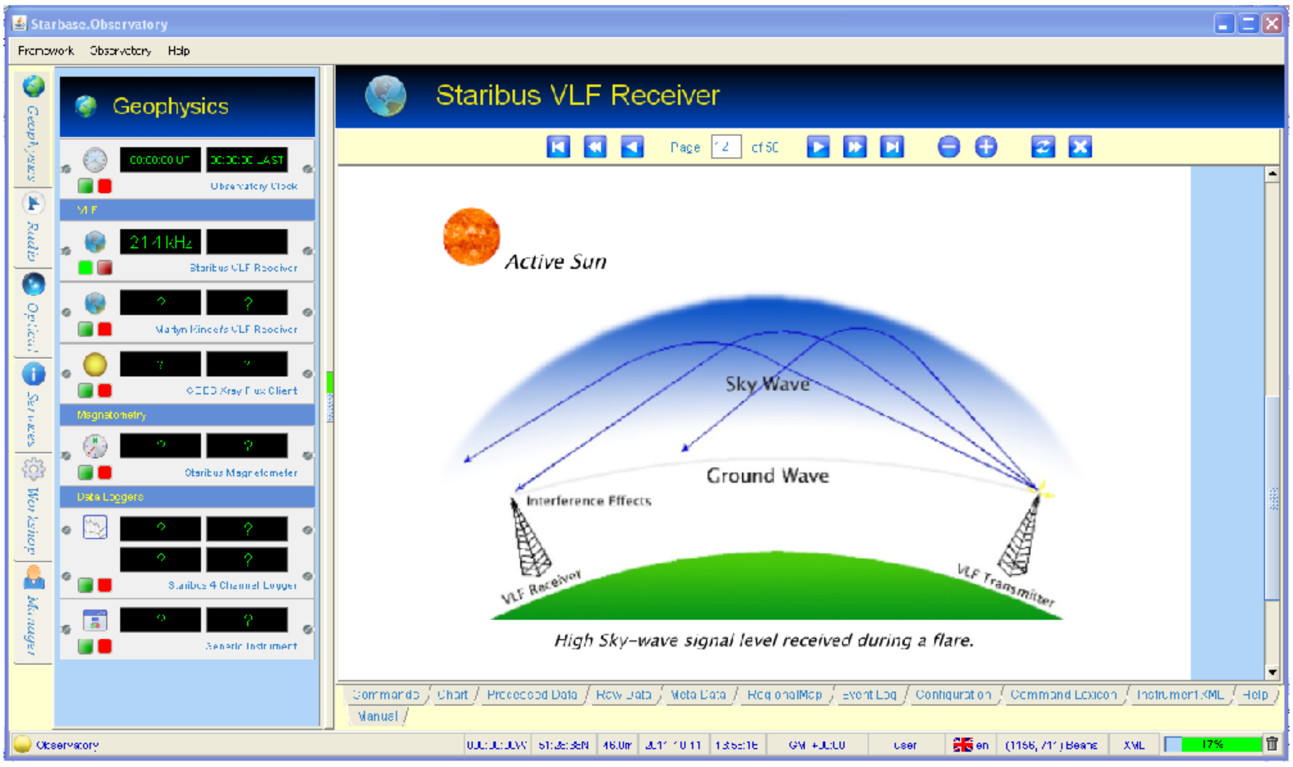
Each Instrument has several *tabs* which present data relevant to that Instrument, such as observations in chart or numerical form, event logs, help screens,and so on. You are encouraged to experiment by investigating all of the tabs, there's a lot of information in there! Here is a summary of the most common tabs available:



Commands	The main control panel, to execute *Starscript* commands
Chart	A graph of the Processed Data (below)
Raw Data	The Raw Data as collected by the instrument
Processed Data	The Processed Data from the chosen filter (*e.g.* SimpleIntegrator)
Meta Data	Metadata associated with the observation (*e.g.* Observer.Name)
Regional Map	A map of the area around your Observatory location
Instrument Log	Events specific to the instrument data format (*e.g.* GPS locations)
Event Log	Events produced by the instrument (*e.g.* FTP logging)
Configuration	Parameters not associated with data (*e.g.* serial port settings)
Command Lexicon	A list (dictionary) of all commands available in this instrument
Instrument XML	The XML which defines the operation of the instrument
Help	The instrument Help file as HTML or PDF
Manual	The instrument User Manual as PDF (also available in dist/doc)

4.5 Help Tabs

Please note that some help (or manual) tabs contain information in HTML form, and some as PDF files of the User Manuals, which can take several seconds to load – please be patient! The example below shows a page from the VLF User Manual:


Starbase.Observatory
Framework Observatory Help
Geophysics
Staribus VLF Receiver
Active Sun
Sky Wave
Ground Wave
Interference Effects
VLF Receiver
VLF Transmitter
High Sky-wave signal level received during a flare.
Commands Chart Processed Data Raw Data Meta Data RegionalMap Event Log Configuration Command Lexicon Instrument XML Help Manual

Instrument Help or Manual Tabs sometimes contain the User Manual PDF

Note that the toolbar allows you to 'unload' a document if it is no longer required, this could save a considerable amount of memory in the case of a large user manual.

All manuals and other documents are available as separate files in `dist/doc`. All users are welcome to contribute to the content of the help files, please send us your suggestions! The User Guide distributed with Starbase is the version current when the software was built; a later version may exist on the UKRAA webserver, see www.ukraa.com/www/downloads.html

4.6 Metadata Support

There were several design goals during the development of Starbase, one of the more important was the following:

All data passing through the system must be accompanied by structured and consistent metadata.

Metadata – from the Greek: μετά = "after", "beyond", "with", "adjacent", "self", in this context, *data describing the data themselves*.

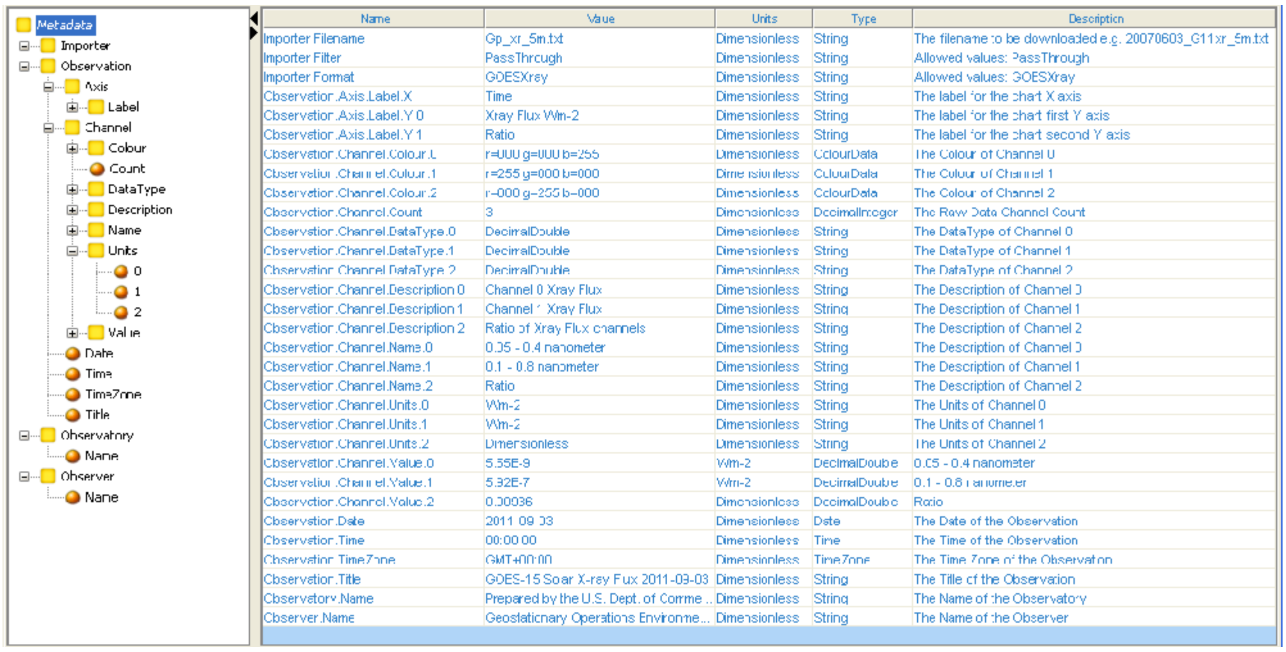
This goal is definitely one that evolved over the years, rather than being clear from the start, but it has become one of the unique selling points of Starbase. A quick anecdote to show why this is important, and what I mean in more detail: I was given the entire life's work of an amateur radio astronomer, on dozens of paper chart rolls, with the hope that I “could do something with them”. It turned out that the charts were literally only recordings showing varying signal levels as a function of time – were they radio signals, or the temperature of the lab, or did someone just jolt the recorder? There were absolutely no indications of the context of the recordings, or even the subject matter. So, sadly the complete collection is effectively useless. Had the charts been annotated with *e.g.* the source under observation, a calibrated scale, the date, the name of the observer, and so on, then something useful or interesting may have been extracted. They were missing their *metadata*.

This experience reinforced my belief that this is one area which is often overlooked by the amateur observer (and perhaps even some professionals), and yet is relatively easy to implement in a useful way. The simple reason behind this design goal is to make sure that those observations I have carefully made are still understandable in the years to come, by others who have no knowledge about how the observations were made, or by whom. I needed to come up with a set of metadata which would completely describe every aspect of an observation, and implement it in a rigorous way which did not allow incorrect or invalid data to creep in to the record.

Starbase defines a simple (extensible) *Metadata Dictionary*, which is a set of predefined metadata items, which can be attached to the observational data. You may think that “surely this has been done before, why reinvent the wheel?”. Well yes, there are several wheels out there, but in my opinion (please prove me wrong) they are all not quite round enough or robust enough to continue rolling in reliable ways, as I hope to illustrate.

Another related factor in this problem is the choice of file format in which to store the data. The format must of course also be able to store the metadata, otherwise each record would have at least two files, which is inconvenient and error-prone. Some formats do support metadata, *e.g.* FITS and RSP. For instance, FITS does have standard dictionaries for keywords, although the names are simple, non-hierarchical, and do not appear to be derived from a consistent data model. One common problem is that different users often have different names for the same items, and perhaps different units and data types. For instance, a TimeZone may be described textually 'GMT+01:30' or as the number of hours offset from GMT, as '1.500', or in a culturally-specific form, such as 'CST' – is that Central Standard Time, or China Standard Time?. If different styles are used, how does the metadata *consumer* know what was intended by the metadata *producer*? Some file formats are not obvious candidates, but can be modified, *e.g.* CSV and TSV. For simplicity during development and testing, I also designed a new XML file format which fully captured the metadata (and the data), and translated easily into the Java data structures in the software. Starbase metadata relate to a comprehensive (but extensible) data model based on *e.g.* Observatory, Observer, Observation, Instrument, and so on.

The screenshot below of the Metadata Explorer tab in Starbase should explain how the metadata are structured, and the ease with which they can be navigated, examined and edited.



Name	Value	Units	Type	Description
Importer Filename	Gp_xr_5m.txt	Dimensionless	String	The filename to be downloaded e.g. 20070603_G11xr_5m.txt
Importer Filter	PassThrough	Dimensionless	String	Allowed values: PassThrough
Importer Format	GOESXray	Dimensionless	String	Allowed values: GOESXray
Observation.Axis.Label.X	Time	Dimensionless	String	The label for the chart X axis
Observation.Axis.Label.Y.0	Xray Flux Wm-2	Dimensionless	String	The label for the chart first Y axis
Observation.Axis.Label.Y.1	Ratio	Dimensionless	String	The label for the chart second Y axis
Observation.Channel.Colour.0	r=000 g=000 b=255	Dimensionless	ColourData	The Colour of Channel 0
Observation.Channel.Colour.1	r=255 g=000 b=000	Dimensionless	ColourData	The Colour of Channel 1
Observation.Channel.Colour.2	r=000 g=255 b=000	Dimensionless	ColourData	The Colour of Channel 2
Observation.Channel.Count	3	Dimensionless	DecimalInteger	The Raw Data Channel Count
Observation.Channel.DataType.0	DecimalDouble	Dimensionless	String	The DataType of Channel 0
Observation.Channel.DataType.1	DecimalDouble	Dimensionless	String	The DataType of Channel 1
Observation.Channel.DataType.2	DecimalDouble	Dimensionless	String	The DataType of Channel 2
Observation.Channel.Description.0	Channel 0 Xray Flux	Dimensionless	String	The Description of Channel 0
Observation.Channel.Description.1	Channel 1 Xray Flux	Dimensionless	String	The Description of Channel 1
Observation.Channel.Description.2	Ratio of Xray Flux channels	Dimensionless	String	The Description of Channel 2
Observation.Channel.Name.0	0.05 - 0.4 nanometer	Dimensionless	String	The Description of Channel 0
Observation.Channel.Name.1	0.1 - 0.8 nanometer	Dimensionless	String	The Description of Channel 1
Observation.Channel.Name.2	Ratio	Dimensionless	String	The Description of Channel 2
Observation.Channel.Units.0	Wm-2	Dimensionless	String	The Units of Channel 0
Observation.Channel.Units.1	Wm-2	Dimensionless	String	The Units of Channel 1
Observation.Channel.Units.2	Dimensionless	Dimensionless	String	The Units of Channel 2
Observation.Channel.Value.0	5.55E-9	Wm-2	DecimalDouble	0.05 - 0.4 nanometer
Observation.Channel.Value.1	5.32E-7	Wm-2	DecimalDouble	0.1 - 0.8 nanometer
Observation.Channel.Value.2	0.00036	Dimensionless	DecimalDouble	Ratio
Observation.Date	2011 09 03	Dimensionless	Date	The Date of the Observation
Observation.Time	00:00:00	Dimensionless	Time	The Time of the Observation
Observation.TimeZone	GMT+00:00	Dimensionless	TimeZone	The Time Zone of the Observation
Observation.Title	GOES-15 Solar X-ray Flux 2011-09-03	Dimensionless	String	The Title of the Observation
Observatory.Name	Prepared by the U.S. Dept. of Comme...	Dimensionless	String	The Name of the Observatory
Observer.Name	Geostationary Operations Environme...	Dimensionless	String	The Name of the Observer

Starbase Metadata Explorer tab (data from GOES satellite)

Getting to know the Metadata system and making the best use of it will take some time, but you are encouraged to try it, and see if it helps you. You may have suggestions for improvements, or additions to the Metadata dictionary, if so, we'd be pleased to hear from you.

Wizards only beyond this point...

If you have read this far, you should have all you need to be able to use Starbase for real observations, and for manipulating and saving the results. The following sections are intended for those who cannot resist looking 'under the covers' to find out how it all works. Beyond here there be dragons!

5 Starbase Software Configuration

5.1 Configuration of the Starbase software environment

dist/starbase.bat

Modify the command file (*e.g.* on Windows `starbase.bat`) as required for your environment, but this should not be necessary unless you have had a custom installation of Java. Check that the first few lines of the file are correct:

```
REM Modify this line to point to the root of your own Java installation
set JAVA_HOME="C:\Program Files\Java\jre6"

REM Modify this with the location of the extracted Starbase distribution folder
rem SET SB_HOME="D:\Java\Starbase\dist"
```

5.2 Customising Starbase for a specific User and Location

dist/imports/frameworks.xml

You may edit `frameworks.xml` to set up various parameters for your location. Starbase will however still function correctly if you do not make these changes.

```
<?xml version="1.0" standalone="yes"?>

<Frameworks xmlns="frameworks.xmlbeans.model.fc.lmn.org"
            xmlns:poi="poi.xmlbeans.model.fc.lmn.org">

    <Framework>
        <Name>Starbase</Name>
        <Description>The ultimate data-driven Framework!</Description>

        <!-- You could change these to suit your Locale -->
        <!-- You must use the ISO standard names, which are case-sensitive -->
        <!-- Language Codes: http://www.loc.gov/standards/iso639-2/php/English_list.php -->
        <!-- Country Codes: http://www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html -->
        <Language>en</Language>
        <Country>GB</Country>

        <!-- Timezones use the Java convention for names -->
        <!-- e.g. GMT-08:00 is equivalent to 8 hours behind Greenwich time -->
        <!-- See for example: http://mindprod.com/jgloss/timezone.html -->
        <!-- or: http://www.twinsun.com/tz/tz-link.htm -->
        <!-- Please report any anomalies in Timezone operation - it is a complex subject! -->
        <Timezone>GMT+00:00</Timezone>
        <!-- Framework Location: These coordinates are of the Royal Greenwich Observatory -->
        <!-- These will be updated automatically if a GPS receiver is connected -->
        <!-- and shown on the Framework status bar -->
        <!-- WEST is POSITIVE (astronomical convention) -->

        <Longitude>+000:00:00</Longitude>

        <Latitude>+51:28:38</Latitude>

        <!-- Height above sea level, in metres -->
        <!-- Default HASL is of the Royal Greenwich Observatory-->
        <!-- This will be updated automatically if a GPS receiver is connected -->

        <HASL>46.0</HASL>

        <!-- PointOfInterest information relevant to the Framework -->
        <!-- The Framework is its own PointOfInterest! -->
        <!-- You may add others below -->
        <!-- Icon pathnames are relative to the 'dist' root -->
        <!-- The filename of the icon to use to show the Framework on the maps -->
        <!--<PointOfInterestFilename>poi-framework.png</PointOfInterestFilename>-->
        <!-- POIs appear on the RegionalMap tabs in the Observatory, for e.g. the VLF Receiver -->
        <!-- This name of this POI *must* be the *same* as the Framework name -->
        <!-- and may not contain spaces -->
        <!-- It is probably best not to experiment with this just yet! -->
        <PointOfInterest>
            <poi:Name>Starbase</poi:Name>
            <poi:Description>The location of this Starbase</poi:Description>
            <!-- WEST is POSITIVE (astronomical convention) -->
            <poi:Longitude>-001:16:20</poi:Longitude>
            <poi:Latitude>+52:03:13</poi:Latitude>
            <poi:HASL>0</poi:HASL>
            <!-- See: http://www.arrl.org/locate/grid.html -->
            <poi:MaidenheadLocator>JO02pb</poi:MaidenheadLocator>
            <poi:IconFilename>poi-framework.png</poi:IconFilename>
        </PointOfInterest>

        <PointOfInterest>
            <poi:Name>London</poi:Name>
            <poi:Description>UK Capital</poi:Description>
            <poi:Longitude>-000:06:22</poi:Longitude>
            <poi:Latitude>+51:31:02</poi:Latitude>
            <poi:HASL>0</poi:HASL>
            <poi:MaidenheadLocator>JO01bq</poi:MaidenheadLocator>
            <poi:IconFilename>poi-london.png</poi:IconFilename>
        </PointOfInterest>

        <!-- A map of your locale, calibrated appropriately -->
        <!-- Note that the map must have a projection which gives **linear** longitude and latitude -->
        <!-- You can make your own map by following the instructions in these links -->
        <!-- http://rimmer.ngdc.noaa.gov/ -->
        <!-- http://www.ruf.rice.edu/~ben/gmt.html -->
        <!-- http://www.ai.sri.com/geotransform/samples.html -->
        <!-- The Starbase distribution contains detailed instruction in dist/maps/ReadMe.txt -->
        <!-- Send us your maps so we can add to those available! -->
        <MapFilename>europe.png</MapFilename>

        <!-- BEWARE! Longitude is POSITIVE to the WEST -->
        <!-- BEWARE! Latitude is POSITIVE to the NORTH -->
        <MapTopLeftLongitude>+010:37:29</MapTopLeftLongitude>
        <MapTopLeftLatitude>+65:00:00</MapTopLeftLatitude>
        <MapBottomRightLongitude>-025:00:00</MapBottomRightLongitude>
        <MapBottomRightLatitude>+34:30:00</MapBottomRightLatitude>

        <!-- A flag to control the automatic updating of the Framework location on the status bar if
GPS is available -->
        <!-- Use the Observatory GpsReceiver Instrument -->
        <AutoUpdate>1</AutoUpdate>
              further content ommitted for clarity
```

5.3 User Roles

XML **`dist/imports/users.xml`**

You may edit `users.xml` if you wish to change the default username (*user*) and password (*starbase*), and so on, as shown in the XML fragment below.

```
<User>
    <UserName>user</UserName>
    <Password>starbase</Password>
    <Active>1</Active>
    <SortOrder>0</SortOrder>
    <Editable>1</Editable>
    <RoleName>Simplified</RoleName>
    <CountryCode>GB</CountryCode>
    <LanguageCode>en</LanguageCode>
    <Email>starbase@ukraa.com</Email>
    <EditorClassname>org.lmn.fc.ui.editors.UserEditor</EditorClassname>
    <Description>A User with simplified functionality during the launch of Starbase</Description>
    <IconFilename>user.png</IconFilename>
</User>
```

XML **`dist/imports/roles.xml`**

You may also add other users with different user roles. The roles are not intended as a security measure, but to allow functionality to be filtered to suit the type of user, for example beginner or expert roles. It may be best to write to Starbase support on starbase@ukraa.com if you need to define new roles in `roles.xml`.

```
<Role>
    <RoleName>Simplified</RoleName>
    <FrameworkViewer>0</FrameworkViewer>
    <FrameworkEditor>0</FrameworkEditor>
    <AtomViewer>1</AtomViewer>
    <AtomEditor>1</AtomEditor>
    <TaskViewer>1</TaskViewer>
    <TaskEditor>1</TaskEditor>
    <ResourceViewer>1</ResourceViewer>
    <ResourceEditor>1</ResourceEditor>
    <UserViewer>1</UserViewer>
    <UserEditor>0</UserEditor>
    <UserCreator>0</UserCreator>
    <UserDeletor>0</UserDeletor>
    <FullScreen>1</FullScreen>
    <Description>A Role with simplified functionality during the launch of Starbase</Description>
</Role>
```

We welcome suggestions for new Roles, and ways in which this feature could be used.

5.4 Runtime Debugging Options

`dist/loader.properties`

The `loader.properties` file contains some simple switches to control debugging messages, such as for the Staribus communications protocol. These messages may be useful during testing, but are best turned off in normal use, since they may slow the system down.

Enable.Debug	'true' or 'false' to control debugging messages
Enable.Debug.Staribus	'true' or 'false' to control debugging Staribus comms protocols
Enable.Debug.Timing	'true' or 'false' to control timing messages for debugging
Enable.Debug.State	'true' or 'false' to control InstrumentState messages for debugging
Enable.Toolbar	'true' or 'false' to control use of the toolbar on low resolution screens
Enable.CommandVariant	'true' or 'false' to control use of CommandVariant in Staribus protocol

The other switches in the file are unlikely to be of use in normal operation.

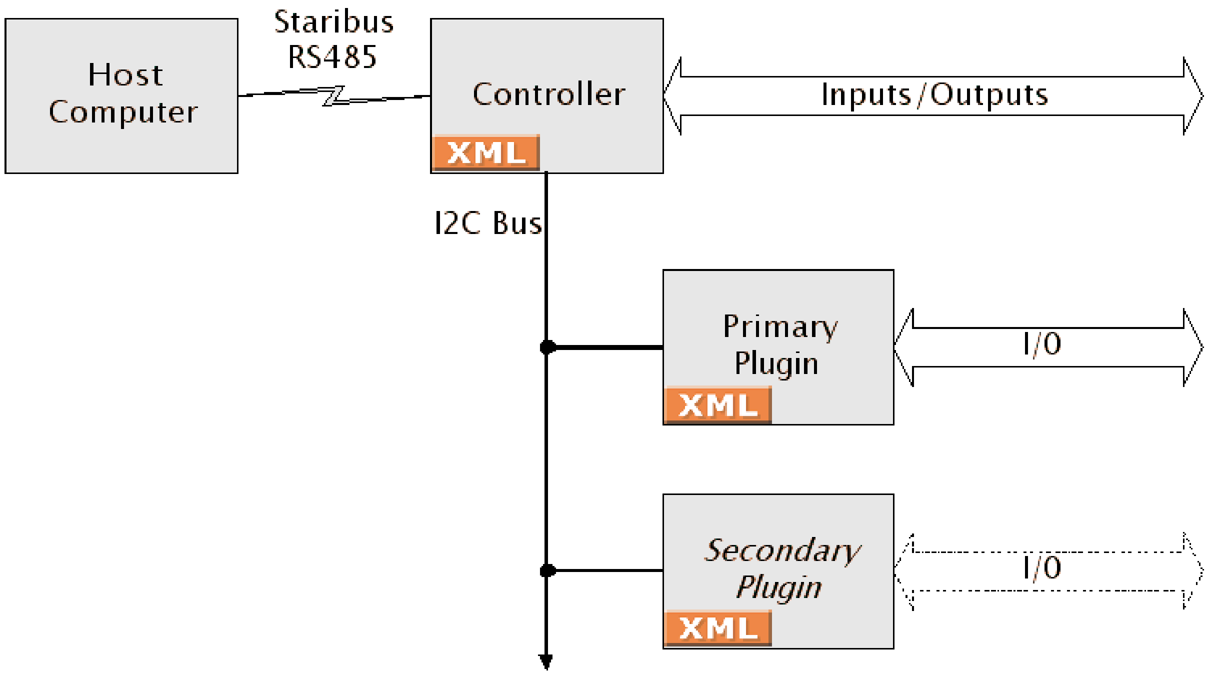
6 Staribus Hardware Configuration

Starbase is capable of connecting to many different kinds of data source, or instrument. Some instruments are implemented entirely in software on the computer (*virtual* instruments), some have real hardware, which may be *local* to the computer or connected via the Internet (*remote*). This section deals with the configuration of those instruments connected to the computer via a local serial bus (RS485), which is the UKRAA **Staribus** (**Star**base **I**nstrumentation **Bus**).

6.1 System Architecture

Staribus instruments consist of a UKRAA Controller and zero or more 'plugins' which provide the specific functionality. The UKRAA VLF Receiver Instrument for instance is made up of a Controller and a Receiver plugin. More complex instruments may require several plugins, each of which must be configured correctly to ensure that the Observatory can recognise what is required.

The diagram below shows how a single Controller may have one *Primary Plugin*, which defines the nature of the Instrument (for instance as a Receiver or Logger), and up to six *Secondary Plugins* (providing extra functionality such as additional amplifiers, oscillators *etc.*). The configuration XML in each device must be correct and consistent for the composite Instrument to function properly.



Staribus Controller, Primary Plugin and Secondary Plugins

6.2 Configuring Staribus Controller Addresses

dist/plugins/observatory/imports/Logger-Staribus-4ch-instrument.xml

A *single* UKRAA Staribus Controller connected via **RS232** or each of *several* controllers connected via **RS485** must have an address which is used to identify the device to the Observatory. This address is a decimal number in the range 001 to 253 (address 254 is reserved for broadcasts to all devices). It is recommended that you use the *lowest* addresses possible in your system, since this may improve performance during automatic 'Discovery' of the instruments (this feature is still under development at the time of writing).

The default `StaribusAddresses` of the PrimaryPlugins of a standard Starbase distribution are:

- **001** `Four Channel Logger`
- **002** `Eight Channel Logger`
- **003** `VLF Receiver`
- **004** `Magnetometer`

You are free to choose whichever addresses you find convenient. Edit the XML file appropriate to each Instrument as shown below, where the example is for the `Staribus Four Channel Logger`.

```
<xml-fragment xmlns:md="metadata.xmlbeans.model.fc.lmn.org"
xmlns:dt="datatypes.xmlbeans.model.fc.lmn.org"
xmlns:gp="groups.xmlbeans.model.fc.lmn.org"
xmlns:ins="instruments.xmlbeans.model.fc.lmn.org">
  <ins:Identifier>Staribus4ChannelLogger</ins:Identifier>
  <ins:Name>Staribus 4 Channel Logger</ins:Name>
  <ins:Description>A Staribus 4 Channel Data Logger</ins:Description>
  <ins:InstrumentClassname>org.lmn.fc.frameworks.starbase.plugins.observatory.ui.instrume
nts.impl.staribuslogger.StaribusLogger</ins:InstrumentClassname>
  <ins:Controllable>true</ins:Controllable>
  <ins:SelectorPanelHeight>1</ins:SelectorPanelHeight>
  <ins:ResourceKey>StaribusLogger</ins:ResourceKey>
  <!--The Data Access Object is the Java interface from the Host PC to the Controller-->
  <!--In this case via the RS485 on a COM port as defined in properties-Staribus.xml-->
  <ins:DAO>
       DAO contents ommitted for clarity
  <ins:/DAO>
  <!--The Staribus Controller-->
  <ins:Controller>
    <!--A placeholder for the discovered address on Staribus, or an address already assigned-->
    <!--Three DECIMAL digits, range {001...254}-->
    <!--Default address is 001-->
    <ins:StaribusAddress>001</ins:StaribusAddress>
    <ins:Identifier>Core</ins:Identifier>
```

All Staribus instrument addresses must be unique.
Starbase will check this configuration as it starts up,
and warn the user if changes are required.

6.3 Configuring the Staribus Communications Port

XML **dist/plugins/observatory/imports/StaribusPort-properties.xml**

In order to use the single common Staribus port for all connected controllers, the port must be assigned to a physical port on your computer. On a Windows system, these will be called `COM1`, `COM2`, and so on. Enter the name you wish to use as shown below, in `StaribusPort-properties.xml`. This name must be in the format as recognised by your operating system. This configuration is normally performed by the installer.

Staribus.Port.Name

```
    <!-- Set up the COM port to use for Staribus -->
     <PropertyResource>
         <ResourceKey>
             <Key>Staribus</Key>
             <Key>Port</Key>
             <Key>Name</Key>
         </ResourceKey>
         <Editable>1</Editable>
         <DataTypeClassName>java.lang.String</DataTypeClassName>
         <EditorClassname>org.lmn.fc.ui.editors.PropertyEditor</EditorClassname>
         <Value>COM1</Value>
         <Description>The COM port used for the Staribus serial port</Description>
         <CreatedDate>2000-01-01Z</CreatedDate>
         <CreatedTime>00:00:00.000Z</CreatedTime>
         <ModifiedDate>2000-01-01Z</ModifiedDate>
         <ModifiedTime>00:00:00.000Z</ModifiedTime>
     </PropertyResource>
```

*The Baudrate of the serial communications port must be set correctly, to **57600** baud, as shown below. However, note that during testing of a controller with **no configuration EEPROM**, the rate automatically defaults to **9600** baud, and must therefore be changed in this file if that mode of operation is required.*

Staribus.Port.Baudrate

```
     <PropertyResource>
         <ResourceKey>
             <Key>Staribus</Key>
             <Key>Port</Key>
             <Key>Baudrate</Key>
         </ResourceKey>
         <Editable>1</Editable>
         <DataTypeClassName>java.lang.Integer</DataTypeClassName>
         <EditorClassname>org.lmn.fc.ui.editors.PropertyEditor</EditorClassname>
         <!-- Allowed values: 2400, 4800, 9600, 19200, 38400, 57600, 115200 depending on the
controller configuration -->
         <!-- The default value is 57600 Baud-->
         <Value>57600</Value>
         <Description>The Baud Rate setting for the Staribus serial port</Description>
         <CreatedDate>2000-01-01Z</CreatedDate>
         <CreatedTime>00:00:00.000Z</CreatedTime>
         <ModifiedDate>2000-01-01Z</ModifiedDate>
         <ModifiedTime>00:00:00.000Z</ModifiedTime>
     </PropertyResource>
```

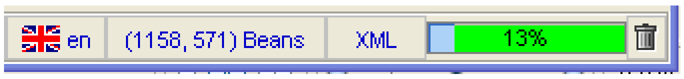
7 Advanced Configuration

This section shows how to configure some details of Starbase in order to improve performance, or to connect different data sources. Most users are unlikely to need to venture into these areas.

7.1 Starbase Memory Allocation

dist/starbase.bat

Starbase could under some circumstances require more memory than its current allocation, particularly if running several instruments, each of which is capturing or displaying large datasets. It is possible to change the memory allocation as described below.



Starbase Memory Usage Indicator

The memory indicator in the lower right of the Framework screen will show how much of the allocation is in use. If the display consistently shows above about 70%, and certainly if the display turns red, then it may be beneficial to increase the memory allocation. Modify the command file (*e.g.* on Windows `starbase.bat`, on Linux `starbase.sh`) on this line:

```
java -Xms512m -Xmx512m -XX:+UseConcMarkSweepGC  …........
```

Change the *two* numbers (in this case 512) to indicate how many megabytes of RAM should be used. Changing to 1024Mb should be more than adequate for most users. Be careful not to change any other parameters to the Java command!

Starbase will function satisfactorily with only 256Mb of memory, and a Windows batch file is provided as an example, with the command line already set up. The file is `dist/starbase256.bat`. The program is likely to be less responsive than with a larger amount of memory, but it should remain stable and usable. Memory allocation lower than 256Mb is not recommended!

7.2 Customising the Observatory Groups

dist/plugins/observatory/imports/instruments-attributes.xml

You may map each Instrument to any UserRole or to any ObservatoryGroup, as shown in the XML fragment below (example for the Logger), by editing instruments-attributes.xml.

```
<Configuration>
    <Identifier>Staribus4ChannelLogger</Identifier>

    <UserRoles>
        <RoleName>SuperUser</RoleName>
        <RoleName>Administrator</RoleName>
        <RoleName>Simplified</RoleName>
        <RoleName>Educational</RoleName>
        <RoleName>Researcher</RoleName>
        <RoleName>Developer</RoleName>
    </UserRoles>

    <ObservatoryGroups>
        <gp:Group>
            <gp:GroupID>0</gp:GroupID>
            <gp:SortIndex>4</gp:SortIndex>
        </gp:Group>
    </ObservatoryGroups>
</Configuration>
```

dist/plugins/observatory/imports/Observatory-groups.xml

You may also edit the definitions of ObservatoryGroups, or even create new groups of your own, by editing Observatory-groups.xml as shown in the XML fragment example below:

```
<Definition>
    <GroupID>2</GroupID>
    <SortIndex>2</SortIndex>
    <Name>Optical</Name>
    <IconFilename>group-optical.png</IconFilename>
    <Tooltip>Optical Instruments</Tooltip>
</Definition>
```

The SortIndex controls the order in which the group tabs appear. The icon filenames are relative to the dist folder. The GroupID must be unique.

You will need to be very methodical in order to keep the group definitions all consistent to ensure that the Instruments appear as you intend! Make backup copies of your original files.

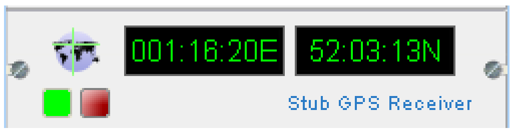
7.3 Configuring a Starbase RS232 Instrument

XML **dist/plugins/observatory/imports/*InstrumentName*-properties.xml**

If you use an Instrument connected via its own dedicated RS232 link, *i.e.* not part of Staribus (RS485), then you must make sure that the serial communication parameters are set correctly. This procedure is identical to setting up the Staribus communications port, but here you must edit the XML **properties** file appropriate for your instrument – an example of the file naming convention is given below:

```
dist/plugins/observatory/imports/myinstrument-properties.xml
```

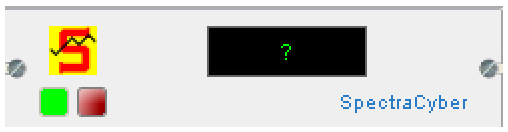
This *optional* configuration currently applies only to the following Instruments:



GPS Receiver

The GPS Receiver can use any RS232-based GPS unit producing NMEA-0183 'sentences'. The Instrument is used to determine the location of the Observatory, and for educational purposes relating to tracking GPS satellites. There is also a 'stub' version of the receiver, which does not require a real GPS device, in order to show how such a receiver would work, using dummy data.

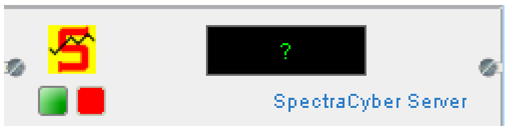
http://en.wikipedia.org/wiki/NMEA_0183



SpectraCyber Hydrogen Line Receiver

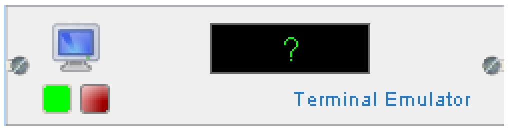
The SpectraCyber Hydrogen Line Receiver produced by Radio Astronomy Supplies is used by BAA RAG as part of a project at the UK National Space Centre in Leicester. The aim of the project is to be able to control the receiver remotely using Starbase, which may then be used to disseminate the observations of the hydrogen line emissions in our Galaxy.

http://www.nitehawk.com/rasmit/ras.html



SpectraCyber Server

The SpectraCyber Server is a variant of the SpectraCyber, which is used to allow remote operation. The Server is directly connected to the receiver hardware, via the serial port in the usual way. However, it listens for incoming commands over the internet, rather than from the local control panel. Another Starbase Observatory running elsewhere runs a SpectraCyberClient, which sends its commands via the internet to the Server. The user of the Client is able to control the remote instrument from anywhere in the world. The protocol used is UDP, which is not as robust as HTTP, but has been found to be satisfactory in this application.



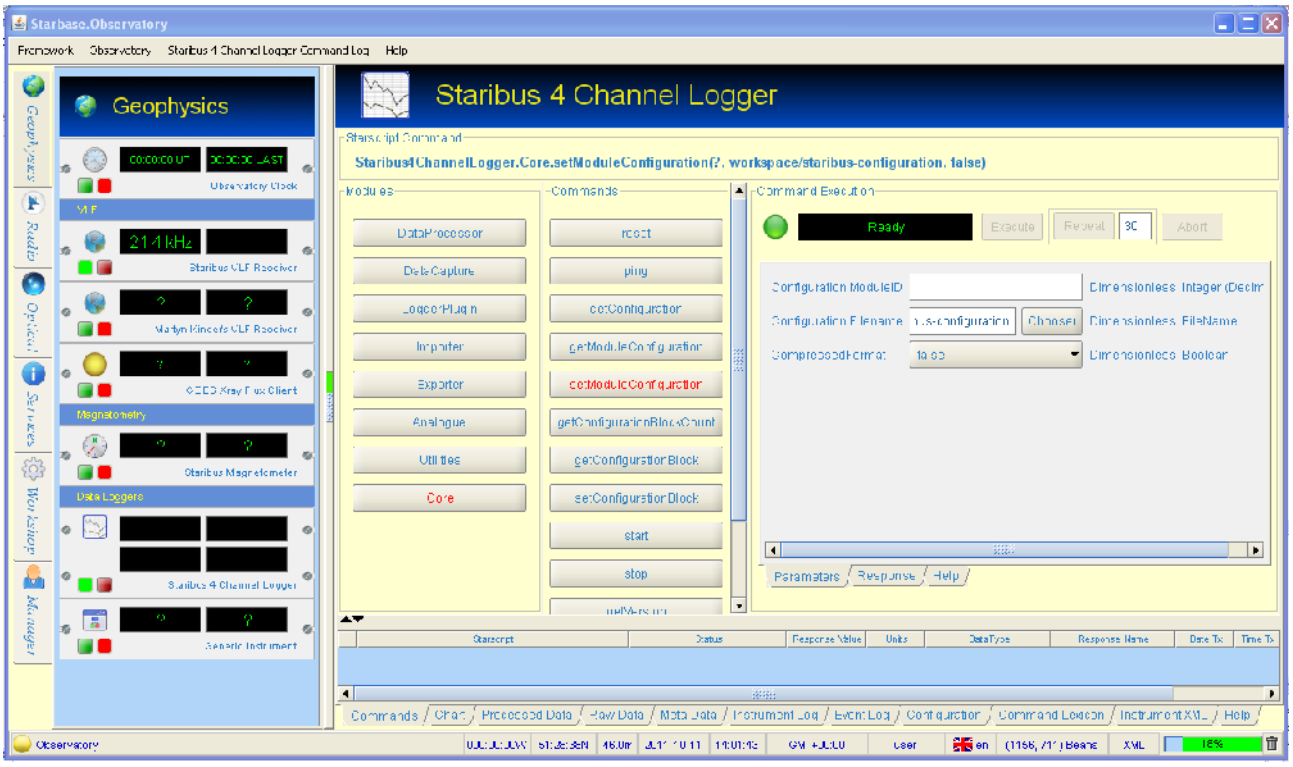
Terminal Emulator

The Starbase Terminal Emulator is a simple instrument which behaves like an RS232 Visual Display Unit. Commands typed on the keyboard are sent to the serial port, and the emulator screen shows any responses received via the same port. This may be used to communicate with devices requiring manual control (such as a communications receiver), or perhaps during testing before connecting to Staribus. The terminal session may be captured and exported to a file. This instrument is very basic; it is not intended to be a full-featured emulator of a specific protocol.

7.4 Programming an XML Configuration EEPROM using Starbase

When you become more familiar with the operation of Starbase, and the way it interacts with the Staribus plugins, you may find it useful to be able to change the XML configuration of an instrument. To do this requires reprogramming the EEPROM in the plugin with a new XML file. The design of the content of the file is outside the scope of this manual, but this section explains how to program the device itself given an XML file on the host computer.

Any Staribus compatible instrument may be used as an EEPROM programmer, since the commands required are present in all `Core` modules. The Appendix shows a circuit diagram of a simple EEPROM programmer plugin. The example below shows the `Staribus Four Channel Logger`, which is a convenient base instrument for this task (although any Staribus instrument can be used).



Starscript Command

```
Staribus4ChannelLogger.Core.setModuleConfiguration(?, workspace/staribus-configuration, false)
```

Configuration ModuleID		Dimensionless Integer (Decim
Configuration Filename	...s-configuration	Dimensionless FileName
CompressedFormat	false	Dimensionless Boolean

The Staribus Four Channel Logger Command Panel

Several `Core` commands need to refer to the specific module in use, via the `ModuleID` parameter. The Controller is always `ModuleID 0`, the Primary Plugin is always `ModuleID 1`. Secondary Plugins have ModuleIDs from `2` to `7`.

Proceed as follows:

- Connect a PluginAdaptor or an EEPROM Programmer Plugin (see Appendix) to the UKRAA Controller
- Insert a blank EEPROM into the programmer socket
- Connect the UKRAA Controller to the computer and apply power;
  it will respond to the `StaribusAddress` in the **Controller** EEPROM.
- Ensure that you know the `StaribusAddress` in the Controller,
  and that a Starbase instrument is correctly configured for this address
- Start Starbase and the Observatory plugin
- Start the Staribus instrument you are using as a programmer *(e.g. a Logger)*
- Execute `Core.ping()`
  This should give a `SUCCESS` message in the Status in the Command Log
- Select `Core.setModuleConfiguration`
  Set parameter `Configuration.ModuleID` to `1`
  Set parameter `Configuration.Filename` to the path to your XML file
  e.g. `C:/Temp/staribus-controller.xml`
  or `workspace/staribus-configuration/staribus-controller.xml`
  Set parameter `CompressedFormat` to `true`
- Execute `Core.setModuleConfiguration(id, filename, true)`
  If `Enable.Debug.Staribus` is set in `dist/loader.properties`
  then you should see SUCCESS messages as each data block is written
- Execute `Core.getModuleConfiguration(id)`
  This will read the XML back and validate it to ensure it was written correctly

If programming a succession of EEPROMs, turn off the UKRAA Controller, and remove the programmed EEPROM. **Label it!** Plug in a new, blank EEPROM, apply power to the Controller and proceed as above. Remember to allow time for the Controller to initialise fully before beginning.

The use of the `CompressedFormat` option will save a considerable amount of space in the EEPROM, but will remove all formatting from the XML. This will normally not be a problem, unless you want to preserve the readability of the XML.

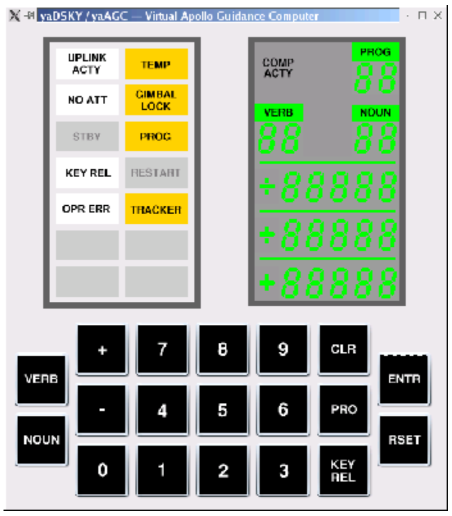
8 Appendix

8.1 User Interface

The command interface of Starbase was inspired by that used on test equipment from the likes of Hewlett Packard and Marconi. These pieces of equipment gained a reputation for being very easy to learn to use, and seemed a good model for the kind of interface required for Starbase.

The design goal was to build an interface which was exactly the same regardless of the data source, so all 'command sets' of any kind of instrument are mapped to the same user interface. This would mean that once the users have learnt how to operate one instrument, then they have essentially learnt them all. This type of interface is known as 'noun-verb', where the *noun* is the target of the command, and the *verb* is the action - the user selects the item on which to operate, and then chooses the action to perform.

An interesting example of this interface was used on the display unit (DSKY) for the Apollo Guidance Computer, shown below:



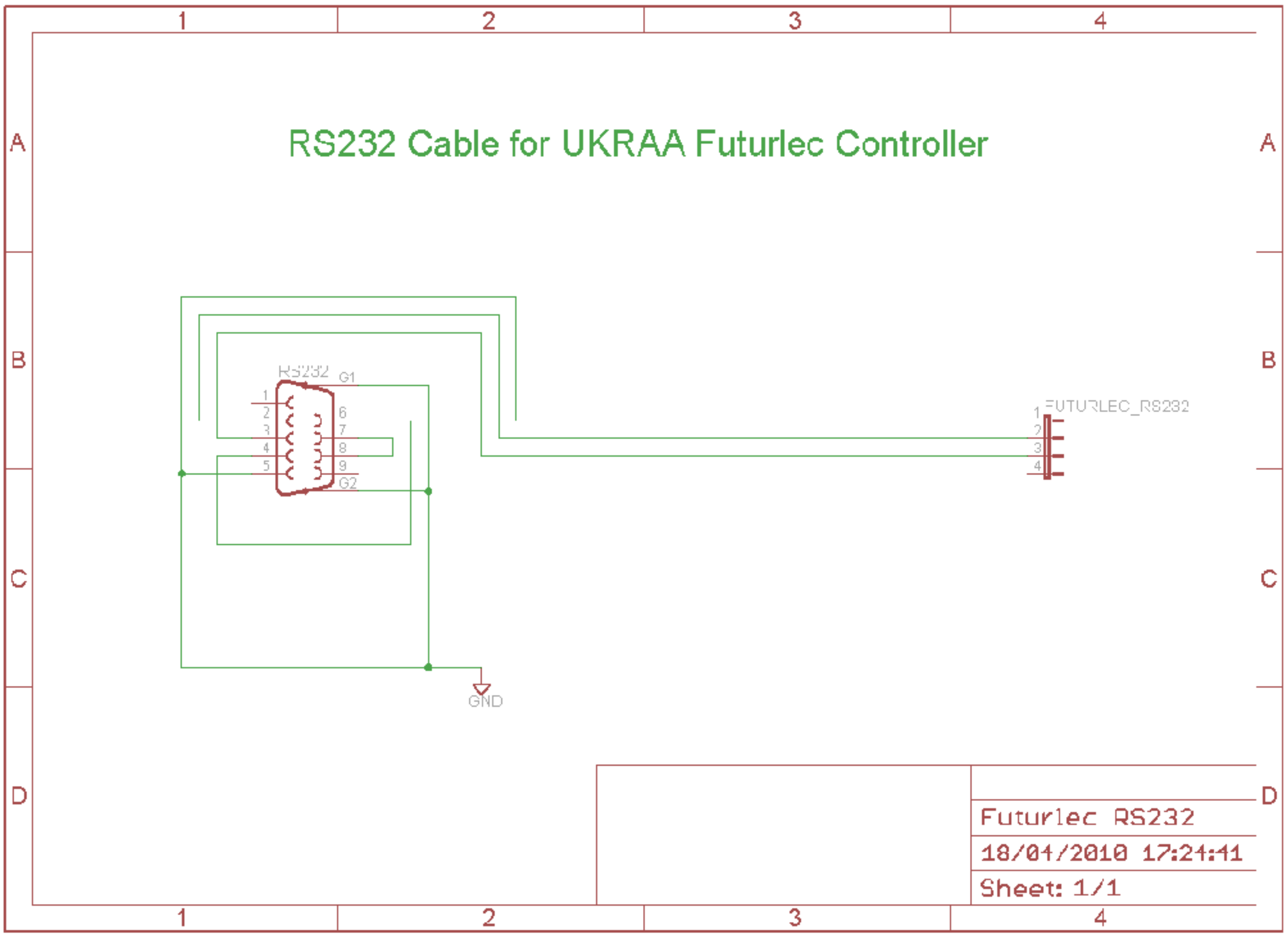
The Apollo Guidance Computer DSKY

Here the nouns may be rocket engines or gyros, and the verbs may be life-changing, but the interface 'metaphor' is essentially the same as an HP oscilloscope.
See these references for information on this user interface:

http://en.wikipedia.org/wiki/Apollo_Guidance_Computer

http://www.ibiblio.org/apollo/

8.2 UKRAA Futurlec Controller RS232 Cabling

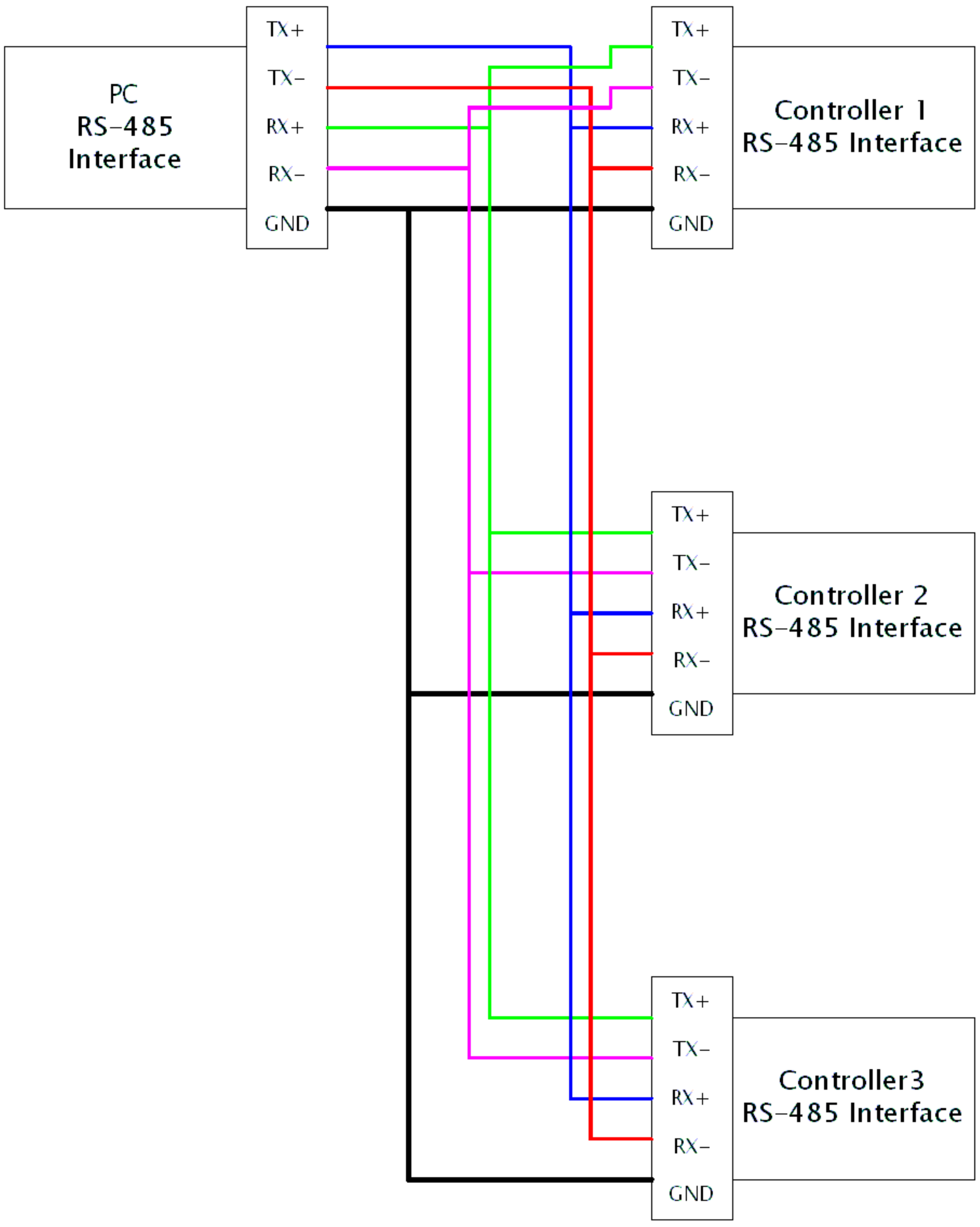


RS232 Cable for UKRAA Futurlec Controller

This cable connects the Futurlec controller **RS232** 4-way header to a 9-way PC serial connector.

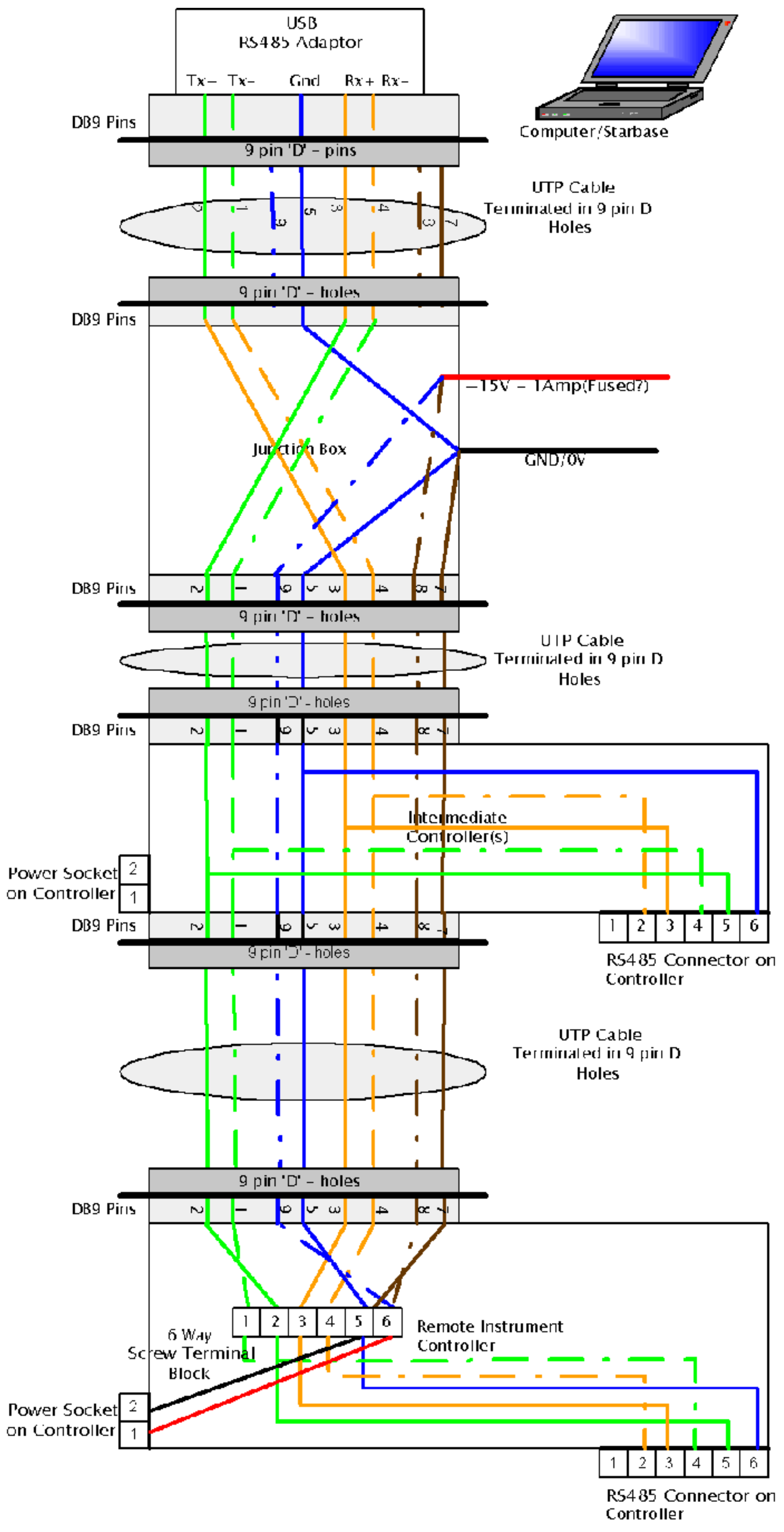
*Note that different cabling and driver chips are required if using the **RS485** bus connection. The choice of pin connections to be used as the UKRAA RS485 standard is currently under investigation. The next section describes the story so far.*

8.3 RS485 Controller Cabling



RS485 Controller Cabling

The above diagram shows the connections required for the RS485 bus. There are two wires for Transmit, and two wires for Receive. The cable should consist of twisted-pairs (*e.g.* CAT5), where Tx and Rx are separate pairs. The next diagram shows one *suggestion* for wiring multiple controllers, and supplying their power.



Notes

USB RS485 Adaptor terminates in DB9 pin male (pins)

Interface Box:
Power is injected (15V @ 1 Amp) using a 2.5 mm concentric adaptor. Power is then carried using pins 8,9 for +15V and 5,7 for Ground. Ground pins are carried to DB9 screen if screened cable is used.

15V/Ground are carried down 2 x twisted pair - solid wire carrying Ground, Stripes carrying +15V

The crossover for the master to first slave is carried out here.
Box has DB9 pin male (pins) input and output.

Each controller/ instrument has 2 x DB9 male (pins) (in/out).
Power is taken from pins 7,8 and 5,9.

All interlinking cables are common, manufactured from UTP or STP.

Tx pair =orange pair
Rx pair = green pair
+15V = Brown pair
GND = blue pair

Remote controllers in hostile environments may terminate the cables into screw terminal blocks. However, using standard colours all the way through reduces the risk of errors.

Pin 6 in unused.
This pin is used to provide remote power to USB converters where required. This can vary from converter to converter so better left unused.

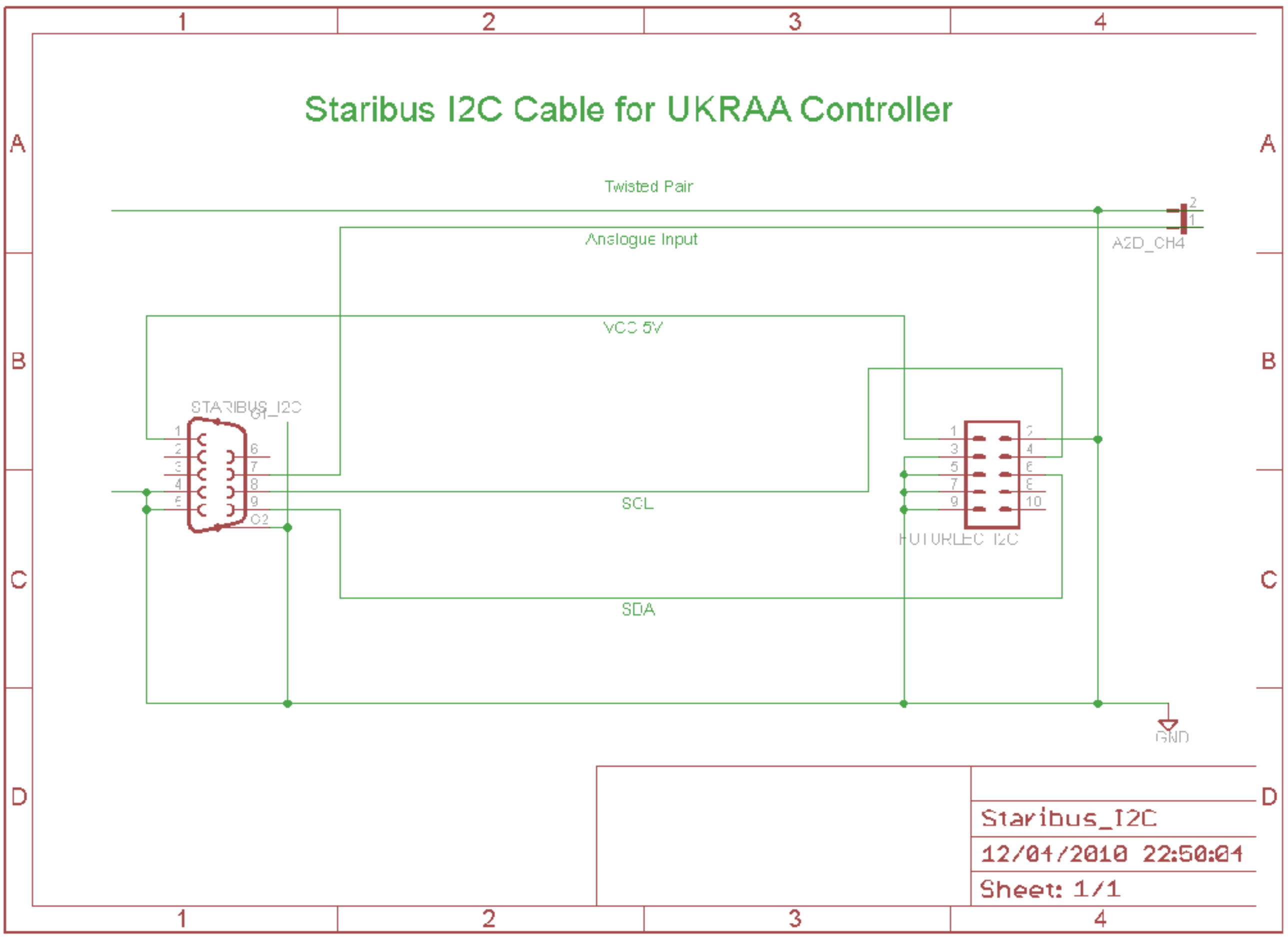
8.4 Staribus I^{2}C Address Map

All current Staribus Instruments store their XML configuration data in a serial EEPROM accessed via the I^{2}C bus. The temperature of the circuit board is monitored with a temperature sensor device, also connected via I^{2}C. All types of I^{2}C device have restricted bus address ranges, depending on their specific function, which are allocated during manufacture. The temperature sensor and memory devices on the Staribus Instruments can respond only to the I^{2}C bus addresses shown in the table below.

Address *(binary)*	Address *(Hex)*	Module	Function
1001 1000	98	_	**Primary Plugin**: Write to LM73 Temperature Sensor Pointer Register
1001 1001	99	_	**Primary Plugin**: Read from LM73 Temperature Sensor Pointer Register
1001 1010	9A	_	**Reserved** Write to LM73 Temperature Sensor Pointer Register
1001 1011	9B	_	**Reserved** Read from LM73 Temperature Sensor Pointer Register
1001 1100	9C	_	**Reserved** Write to LM73 Temperature Sensor Pointer Register
1001 1101	9D	_	**Reserved** Read from LM73 Temperature Sensor Pointer Register
1010 0000	A0	0	**Controller**: Write to EEPROM
1010 0001	A1	0	**Controller**: Read from EEPROM
1010 0010	A2	1	**Primary Plugin**: Write to EEPROM
1010 0011	A3	1	**Primary Plugin**: Read from EEPROM
1010 0100	A4	2	**Secondary Plugin**: Write to EEPROM
1010 0101	A5	2	**Secondary Plugin**: Read from EEPROM
1010 0110	A6	3	**Secondary Plugin**: Write to EEPROM
1010 0111	A7	3	**Secondary Plugin**: Read from EEPROM
1010 1000	A8	4	**Secondary Plugin**: Write to EEPROM
1010 1001	A9	4	**Secondary Plugin**: Read from EEPROM
1010 1010	AA	5	**Secondary Plugin**: Write to EEPROM
1010 1011	AB	5	**Secondary Plugin**: Read from EEPROM
1010 1100	AC	6	**Secondary Plugin**: Write to EEPROM
1010 1101	AD	6	**Secondary Plugin**: Read from EEPROM
1010 1110	AE	7	**Secondary Plugin**: Write to EEPROM
1010 1111	AF	7	**Secondary Plugin**: Read from EEPROM

The default settings are highlighted. The addresses reserved for the temperature sensor are intended for instruments requiring multiple sensors.

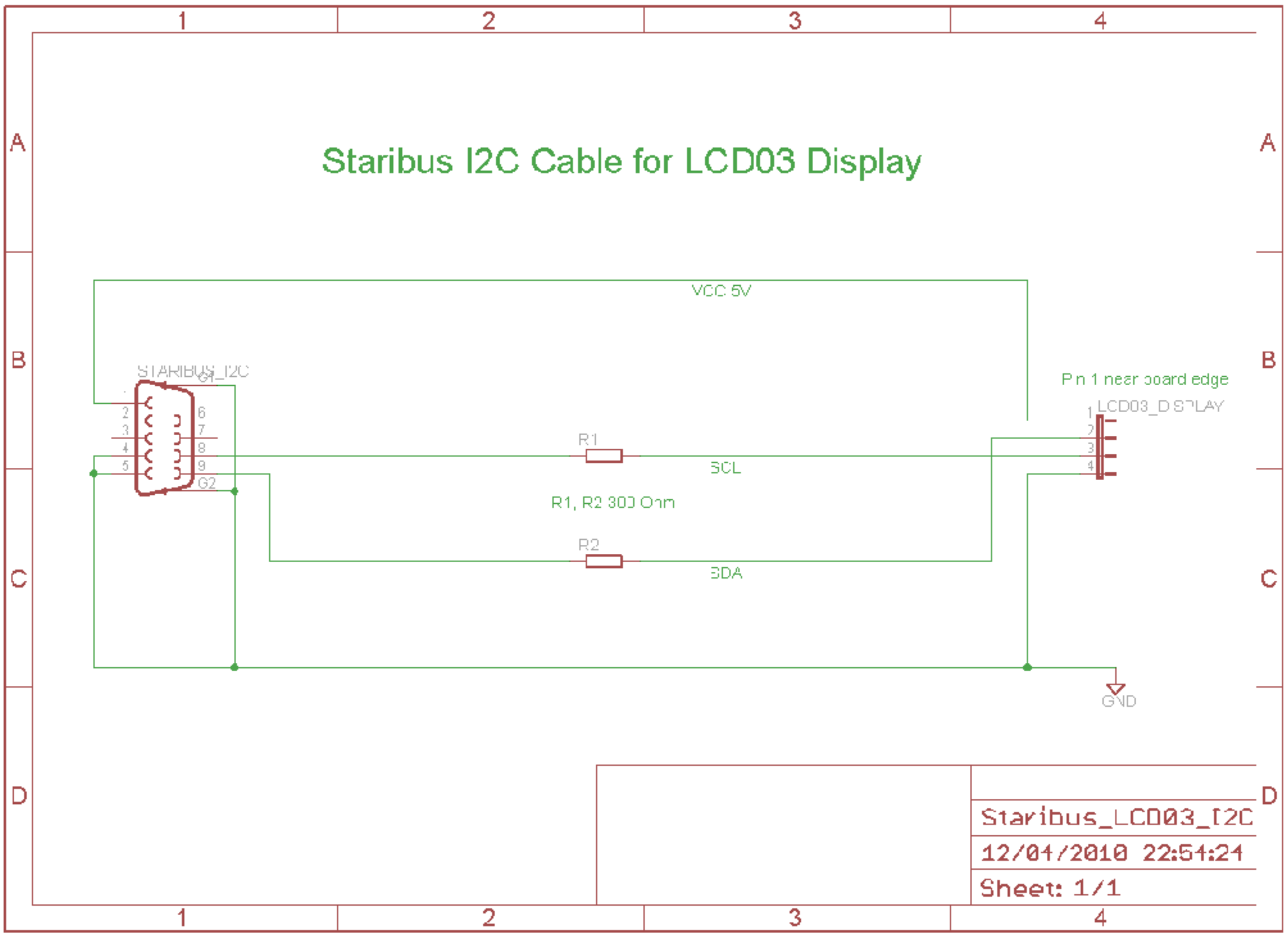
8.5 Staribus I^{2}C Bus Cabling



Staribus I^{2}C Cable for UKRAA Controller

This cable connects the Futurlec I^{2}C 10-way header to the 9-way D-type connector as used for all Staribus I^{2}C plugins. Note that there is also one analogue input channel taken from the 9-way connector **pin 7** to channel **4** of the Futurlec Analogue to digital Converter (ADC) inputs. In the first instance, this is specifically to carry the output from the UKRAA VLF Receiver, which connects to the controller using a standard 25-way to 9-way cable. Most instrument plugins are expected to obtain analogue data via the I^{2}C connection, direct from an Analogue to Digital converter in the plugin.

8.6 Staribus I^{2}C LCD Display Cabling



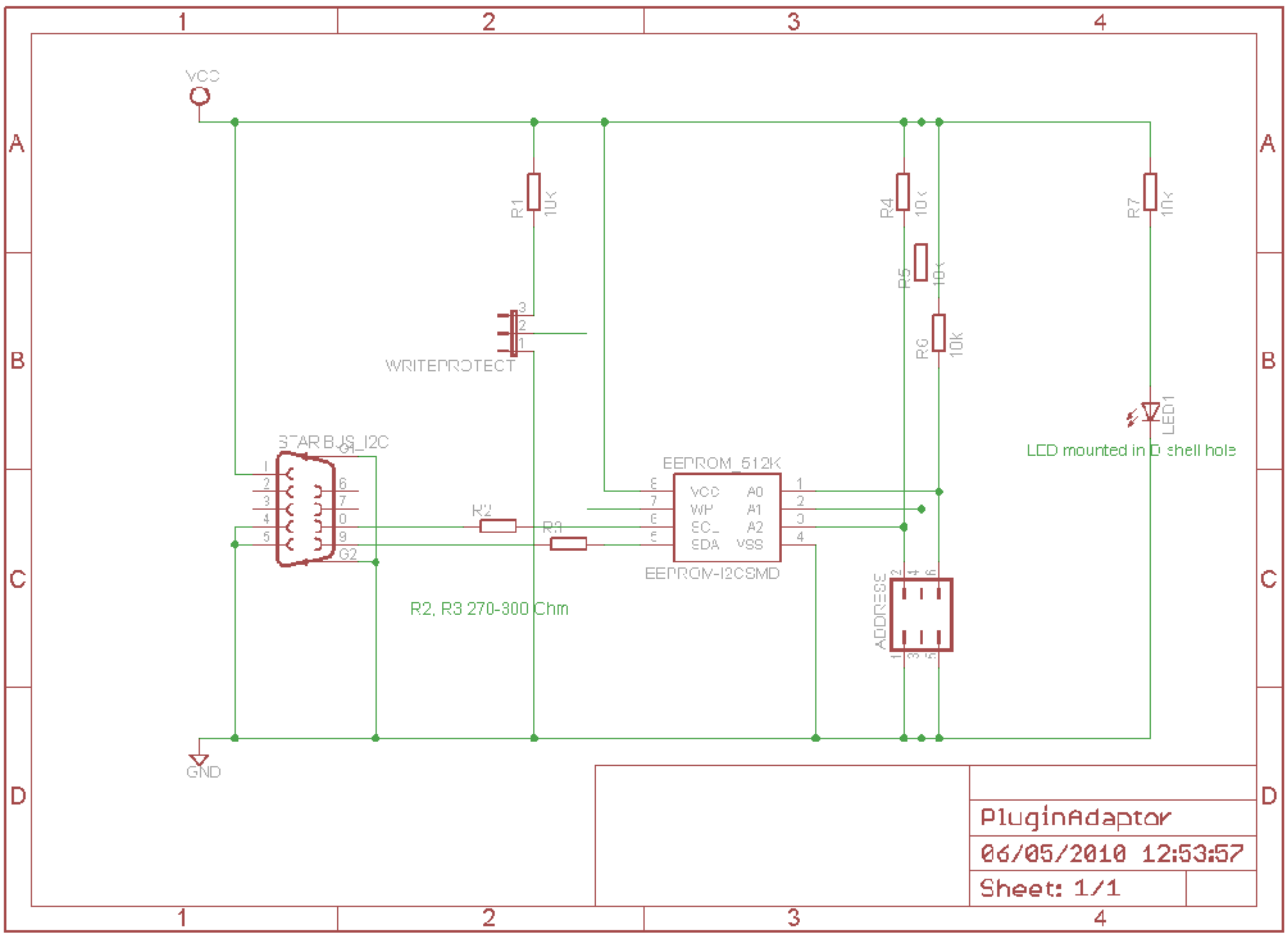
Staribus I2C Cable for LCD03 Display

This cable connects the recommended LCD display to the Staribus I^{2}C 9way connector described in the previous section.

The LCD03 display is obtainable from:
http://www.robot-electronics.co.uk

Note the two series resistors in the I^{2}C control lines; these are 'damping' resistors, which help to improve noise immunity on long bus lines. Such resistors should be fitted to all I^{2}C plugins; their value is not critical (say 150R to 330R). The I^{2}C bus should ideally be much less than 1m in length for reliable operation, but the maximum length will depend on local noise conditions, and the number of devices connected to the bus.

8.7 Plugin Adaptor and EEPROM Programmer



PluginAdaptor and EEPROM Programmer

The standard Controller offers limited functionality until it is configured as a specific Instrument. The plain Controller will respond to `Core` commands, and `Utilities` such as the Clock and the analogue converters. For operation as a real Instrument, the I^{2}C plugin must provide additional configuration via one or more EEPROM devices. The circuit above is a simple plugin with a single EEPROM, which may be used to configure the Controller as a four or eight-channel Logger. Such a plugin module could also be used as an EEPROM programmer, since the `Core` commands contain utilities for managing the contents of all EEPROMs on the I^{2}C bus. This circuit could be built inside a 9-way D connector shell, for example. The write protection jumper is optional, and the circuit could be permanently wired to allow programming. See an earlier section for EEPROM addressing information. Ready-built plugin adaptors are available from the UKRAA online shop.

8.8 UKRAA VLF Receiver to Staribus Controller Cable

The connection of a UKRAA VLF Receiver requires special consideration, because the socket provided can be used for several purposes. A standard 25 way to 9 way serial cable may be used to connect the VLF Receiver to a UKRAA Controller's I^2C port. No other connections are necessary.

Note that the I^2C port also provides a 5V output, which may be used to power low-current devices. The cable connections are given below for convenience.

Receiver 25way Pin	**Controller 9 way Pin**	**Function**
20	4,5	Ground
5	8	SCL I^2C clock signal
22	9	SDA I^2C data signal
4	7	Receiver 2.5V output to Controller A2D input
n/a	1	+5V Power Supply Output from Controller

9 Contacts

The UK Radio Astronomy Association
Springfield
Rookery Hill
Ashtead Park
Ashtead
Surrey
KT21 1HY

Limited by Guarantee Reg. No. 6481611
Registered UK Charity No. 1123866

E-mail: info@ukraa.com

Website: www.ukraa.com

Fax: 0870 132 3728

Starbase Information
E-mail: starbase@ukraa.com

Website: www.ukraa.com/ukraa/starbase

Download: www.ukraa.com/www/downloads.html

BAA Radio Astronomy Group
E-mail: radiogroup@britastro.org

Website: www.britastro.org/radio

10 Acknowledgment

I would like to take this opportunity to thank the Council of the BAA for awarding me the Horace Dall Medal in 2009, "*... to a person who has shown marked ability in the making of astronomical instruments*", which was a gratefully-received vote of confidence in the project.

11 Revision History

Revision	Date	Author	Status
Draft A	2010-04-12	L M Newell	Internal draft for peer review
Draft B	2010-04-13	L M Newell	Incorporated reviewer's comments
Draft C	2010-04-25	L M Newell	Incorporated reviewer's comments
Draft D	2010-05-07	L M Newell	Incorporated reviewer's comments
Draft E	2010-05-15	L M Newell	Incorporated reviewer's comments
Issue 1	2010-05-20	L M Newell	Issued document
Issue 2	2011-10-10	L M Newell	Changes for Starbase version 2.1.0000

UK Radio Astronomy Association