MIDASsort HOWTO

Duncan Appelbe
CCLRC Daresbury Laboratory
Nuclear Physics Group

Table of Contents

Introduction

MIDASsort forms part of the MIDAS[1] distribution and forms an extension to both the *sunsort* and *Sort shell* packages. The GUI's implemented by *sunsort* and *Sort shell* were based on the openlook libraries which are nolonger supported by Sun. A new interface is provided based upon MIDAS tcl, with routines that generate spectra in the MIDAS format.

As a general note for new users to MIDAS, you do not have to press the return/enter key when you have inputted data into any of the MIDASsort GUI's as the act of moving your mouse away from the active area has the same effect.

Scope

This document is split into two parts; the first is a general users manual, which explains how to use this package and the different data handling routines that are available to the user. The second part is for the developer, or any interested party, and explains the structure of the different routines that make up this package, and how they can be extended/ added to.

For USERS

Starting a new sort session

At this point I am assuming that the MIDAS package has been installed on your computer and that the appropriate enviroment variables have been set.

In an xterm/terminal launch a midas-session:

```
kirk@enterprise: midas-session
```

Once the midas session has completed loading you need to check that you have the correct resources assigned. In principle, these resources will be automagically setup for you. It is advisable that you check that the correct resources are set for you, if not you should change them manually. For help consult your local *MIDASGURU*. Once that all is set up you can safely launch MIDASsort, this is done by clicking on the *Sort Tool* button on the MIDAS base frame (Fig 1). This should launch a window similar to the one shown in Fig. 2.

Figure 1. The MIDAS Base frame

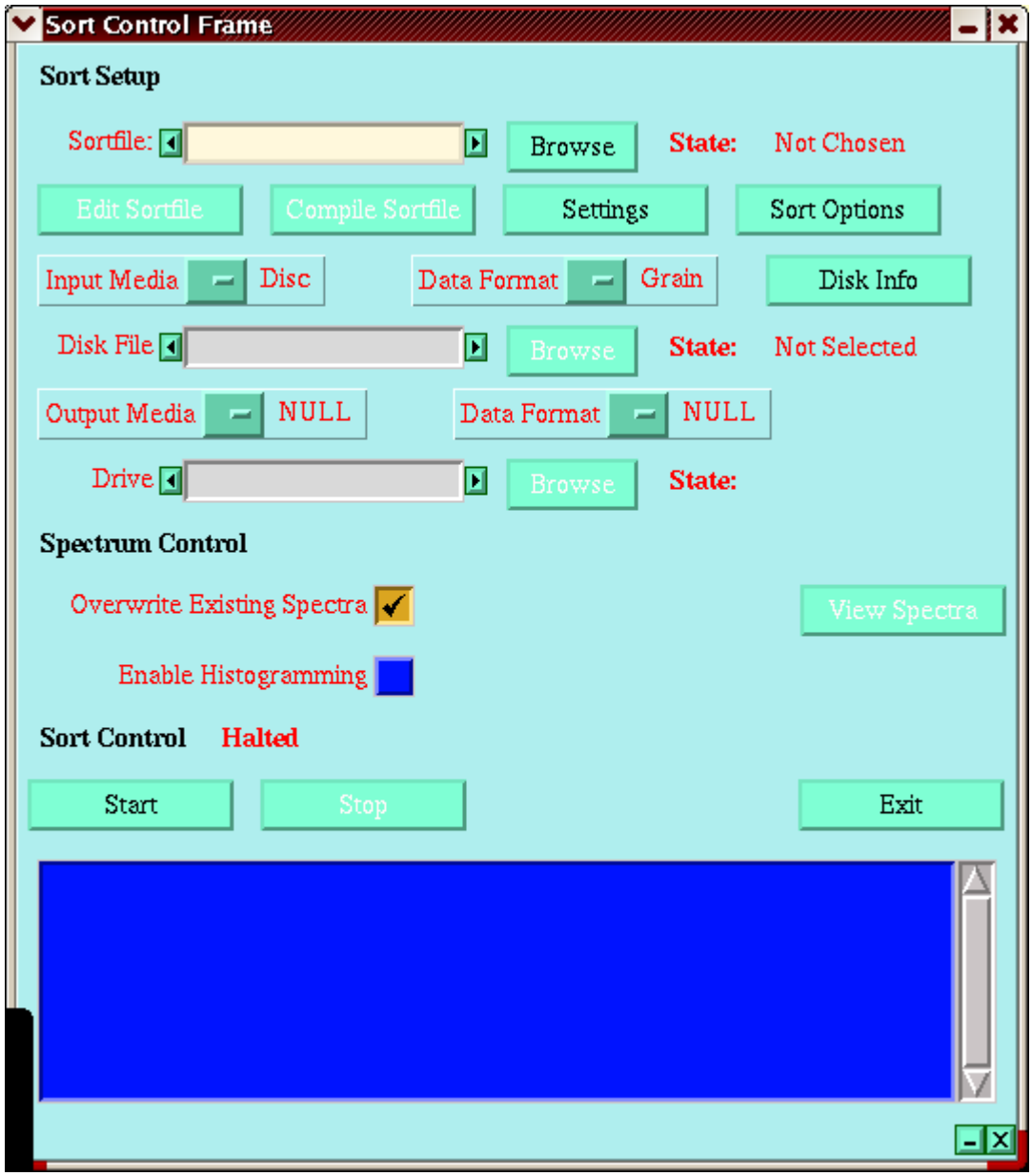

Figure 2. The Sort Control Frame

From this window it is possible to control every feature of the sort. If this is the *first* time that you have used MIDASsort you should check the default settings for your sort envivroment. To do this click on the *Settings* button, this should launch the window shown in Fig. 3. The first time that the sort is run, these settings will all be defaults (in general the paths will be to your home directory), if you modify them they will be saved and will reloaded each time you run this sort package. With the excption of the *MIDASBASE* parameter, which is obtained from your system settings, you should set them up to suit.

Sort directory : is the directory in which all of your sort program source files are kept.

Sort Data directory : this is the directory where your data files (if you are reading data from disc) can be found.

Gainfile directory : this is the directory where your gainfile(s) can be found. In the current implemntation of MIDASsort this option is not enabled.

Gatefile directory : this is the directory where your gatefile(s) can be found. In the current implemntation of MIDASsort this option is not enabled.

Users Makefile : This is the file (full path) of the users makefile which can be included to supply additional options for when the sort is compiled (see later section for the formatting of this file).

Config Table : This is the path to the config table that was used for your experiment. This table is only used if you are going to histogram your data.

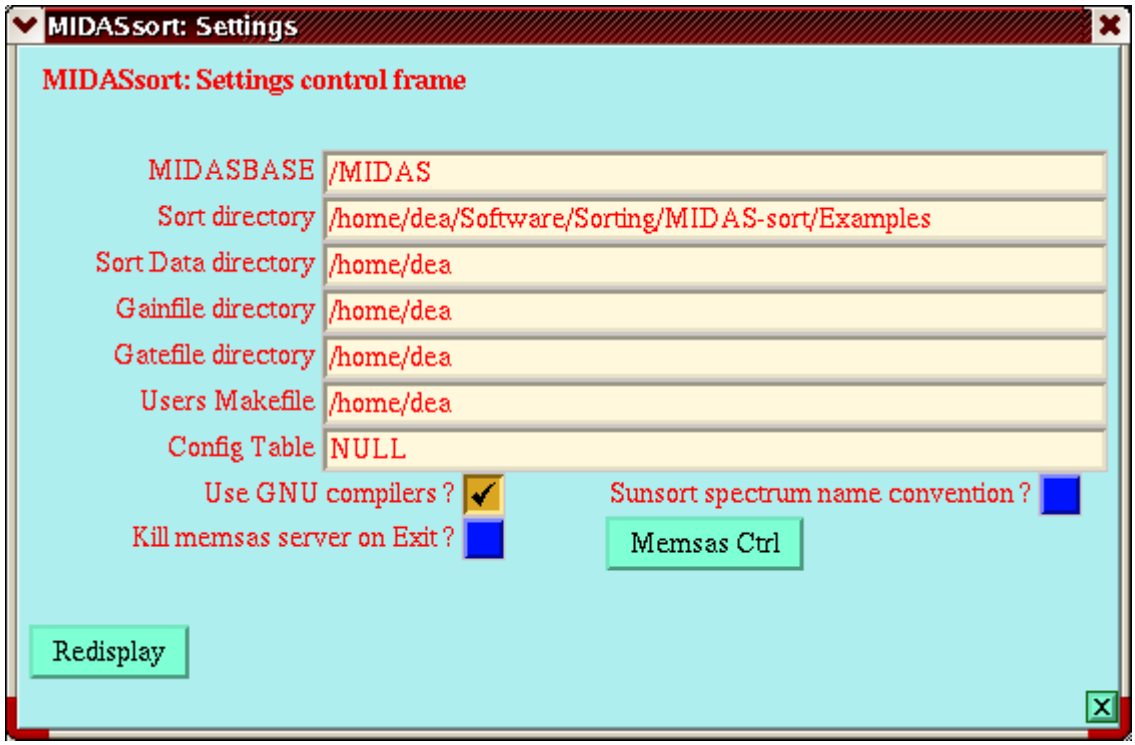

Figure 3. The Sort Settings Frame

In addition to the directory/file paths that should be checked there also exist several check-boxes and additional control buttons. The check-boxes toggle flags and should all be self explanatory. The *Memsas Ctrl* button will launch a frame that will display the properties of any memsas servers that are running on the *host* machine. It is posible to kill off running servers, but beware of this as you have to save your spectra manually at present.

Once you are happy with these settings close the window so as to avoid clutter on your desktop.

A Sort Program.

In order to sort the data the package requires the user to supply one or more of the following routines:

i. 1D spectrum definitions.
ii. 2D spectrum definitions.
iii. Initialisation routine.
iv. Sort routine.
v. Finalisation routine.

These definitions and routines are seperated by the use of "starwords", which, if ommitted will cause compilation errors. The startwords that are currently supported are:

```
*oned

*twod

*sort
```

**oned* is used to specify the 1D spectra that are to be incremented. **twod* is used to specify the 2D spectra that are to be incremented. **sort* after this starword come the user written sort routines.

Spectra are specified in either of the following two formats:

```
id name length

idlow..idhigh name length
```

in the former case "id" represents a number used to identify the spectrum to the different spectrum manipulation routines, "name" *must* begin with a letter, and is the name of the spectrum, finally "length" represents the number of channels in the spectrum (or one side of a matrix). In the latter case idlow and idhigh represent the low and high numbers of a series of spectra with the same name (and a number appended to the name for identification). Examples are shown below:

```
*oned
1    tp  16384
2    id  256
3    fb  8192
4    cl  8192
5..9 ring 8192
*twod
10  gfb  2048
11  gcl  2048
12  gfbcl 2048
```

To make life easier for you when calling the spectrum manipulation routines (from "C") a series of variables are identified by the spectrum name (in upper case).

Prior to the "*sort" word you can comment your listing using #. After this word the program parser assumes that everything is related to the physical sort so be careful. The sort can be written in either C or Fortran, depending upon the language that you choose *please* append either ".c" of ".f" to the filename of your sortfile, as it is the characters after the *last* "." that the package uses to determine which language you are using.

There is one routine that the user *must* specify, the "sortin" routine, other routines include "init", "final" and any user specific functions/subroutines. This is because the "sortin" routine is used to increment spectra and manipulate data. This routine is called for every event. A simplified diagram of the program flow is shown in Fig. 4.

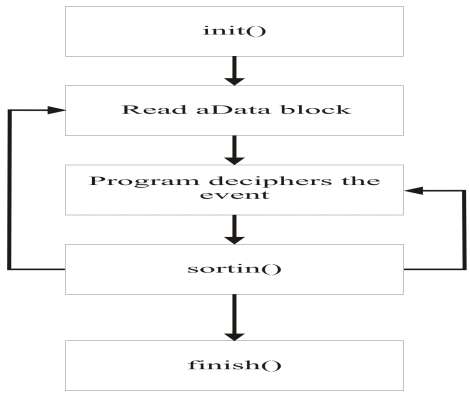

Figure 4. A simplified datagram showing when the different routines specified after the "*sort" word are used.

For Developers/Maintainers

> **Warning**
>
> This part of the documentation is for Developers/Maintainers *only*.

This document has been written using the "DocBook standard".

A. User Functions.

Unless specified all arguments are integers.

Function	Description
gate1d(list,val)	Return true/false (1/0) if the value "val" passes any pair of limits in the 1D array "list". This array has entries (x) = lo, (x+1)=hi, with the last pair of elements being set to zero. In this version, there is no maximum size to "list"

Function	Description
gate2d(lim,x,y)	Return true/false (1/0) if the vertex specifed by "x" and "y" is constrained by the polygon vertices contained in the 1D array "lim". The values of x and y are integer. The array "lim" is an integer list containing the vertices used to specify the bounds of the vertex: this array *must* end with two zeros. The maximum number of elements in "lim" (including the terminating zeros) is 512.
inc1d(id,x)	Increment channel "x" in spectrum number "id". It is suggested that you check the bounds before you call thsi routine, however if you are too lazy the routine also performs the check. This routine is specified as a void.
inc2d(id,x,y)	As "inc1d" but for vertex (x,y).
incv1d(id,x,val)	This routine is similar to "inc1d" however it has an additional parameter, "val". This routine will increment channel "x" in spectrum "id" by "val" counts.
incv2d(id,x,y,v)	As "inc1d" but for a 2D matrix.
set1d(id,x,v)	Routine to set the number of counts in channel "x" of spectrum "id" to "v". This routine returns a void.
read2d(file,set)	This routine can be used to extract the vertices of a 2D polygon from a file generated by MIDAS. The filename is specified (complete path) by "file". You also need to specify the set number using the variable "set", which varies from 0 - n
set2d(id,x,y,v)	Routine to set the number of counts in a 2D matrix to "v".
val1d(id,x)	This routine returns the number of counts in channel "x" of spectrum "id". The return value is an integer.
val2d(id,x,y,v)	This routine returns the number of counts at "x,y" of spectrum "id". The return value is an integer.

B. ToDo List

1. Find a proper maintainer for this package.
2. Add in the ability of the user to modify variables on the fly.

Notes

1. http://npg.dl.ac.uk/MIDAS/base.html