PROS USERS GUIDE

Instructions for the off-line analysis of x-ray data within IRAF

June 3, 1998

for PROS 2.4

Please send comments and corrections to rsdc@cfa.harvard.edu, with subject line “PUG”.

PROS is developed, distributed, and maintained by the Smithsonian Astrophysical Observatory, under partial support from NASA contract NAS5-30934.

html address: http://hea-www.harvard.edu/PROS/PUG/PUG.html

DON'T PANIC!

Additional HELP is available on-line. Typing

help <packagename>

provides a list of tasks in the package. Typing

help <taskname>

will give details about the task. In addition to the help files associated with each task and package, there are a number of more general descriptions which give information about algorithms or about topics which affect more than one task. To view a list of these help files, type:

help pros

....

```
Users who are new to the IRAF environment may wish to try the ``ASSIST''
which functions ``above'' IRAF:

What's New in this release?
---------------------------
This initial public release supersedes version 1.0.1BETA and includes:

        * optimized support for displaying FITS files in SAOtng
        * elimination of race conditions in XPA
        * improved WWW image-loading support in ASSIST
        * plugging a huge memory leak in SAOtng
        * port to Dec Alpha

Introduction
------------
We are pleased to announce the availability of first public release
(version 1.0.2) of the SAO R&D software tree containing, among other
things, a new version of ASSIST and a first release of SAOtng. We
invite you to retrieve the software and use it for your astronomical
analysis (and other?) needs.
```

```
The ASSIST
----------
The ASSIST is a graphical user interface to IRAF and other analysis
environments under the X Window System. It is based on the Answer
Garden Substrate, a system designed to help groups organize and access
databases of frequently asked questions and other information. The
ASSIST provides a convenient way to:


    * traverse the IRAF package hierarchy, loading selected packages
    * find and access tasks in the IRAF hierarchy, using keyword searches
    * access often-used tasks directly without traversing the package tree
    * inspect and change task parameters
    * run tasks
    * view task and package help files
    * submit comments, questions, bug reports
    * try out recipes and tutorials
    * use a common graphical interface for IRAF and non-IRAF tasks

Release 1.2 of ASSIST adds support for WWW documents and more
generalized support for arbitrary analysis programs and systems. It
uses the X Public Access mechanism to allow external processes to
control its functions.

Availability
------------
The SAO R&D software tree can be retrieved via anonymous ftp from
sao-ftp.harvard.edu in the ~ftp/pub/rd directory.  The source tree is
contained in a compressed tar file called saord.tar.Z.

The unpacked tar file requires approximately 45 Mb of disk space.
The build will require approximately double that amount, depending
on the platform.
```

The ASSIST was developed at SAO by the HEAD Software R&D group in cooperation with the AXAF Science Center; it is not supported by the RSDC. The underlying Answer Garden system was developed by Mark Ackerman at MIT.

CHANGE DOC

95 Nov 08	Major release for PROS 2.4
96 Nov 30	Addition of chapter 13 (recipes for ROSAT HRI) fix various typos

Contents

LIST OF FIGURES

1 General Introduction

This guide is designed to aid the new user in deriving scientific results from reduced x-ray data using the SAO PROS software system which runs under IRAF as package "xray". We give the basic instructions for running the tasks, some scientific justifications and a few details – where relevant to the task at hand – on the IRAF/PROS system. This document also contains a list of useful IRAF documents (see §10).

Each chapter contains examples on how to achieve specific scientific goals. It does not cover all the available options and tasks. Comprehensive, on-line help files are available for all tasks, and users are referred to those for details.

IRAF runs on a variety of computers and under several operating systems. This guide will not cover all the possible ways to get started. If you are using the IRAF/PROS system at SAO, you will find the siteguide , by Belinda Wilkes, a useful guide to site-specific procedures. For other sites, see the analogous documents or your local IRAF experts.

Please note that the examples in this document have been tested 'live' on SUN-Unix systems. Although we have tried to make these examples as 'operating system independent' as possible, there may be residual details that do not apply to other systems.

NOTE: In most cases Examples shown in this guide use the REV0 filenames of the sample data in *xdata$*. RDF sample data is now distributed by PROS and can be used when running PUG examples. They are available in *xdata$* with a slight filename change ("n00" added to sequence number, i.e. rh110267n00.qp).

Throughout this document, statements typed by the computer are in `typewriter` font, and user inputs are in regular roman fonts. *Italics* will be used for IRAF terminology and file names. Your computer name (and system prompt) will be printed as "cfa#". When two single quotes are juxtaposed, they should be interpreted as a (normal) double quote.

There are many scientific objectives that can be accomplished with the IRAF PROS package. These include:

- Image Display (See §5)
  - Within Sunview (Sun computers only) (using IRAF *imtool*)
  - Within Xwindows (using *SAOimage*)
- Spatial analysis (see §6)
  - Extract counts within specified regions
  - Extract a count distribution (radial and azimuthal profiles)
  - Correct images for exposure and vignetting
  - Smooth images
  - Plot isointensity contours on smoothed arrays
  - Keep track of errors
- Spectral analysis (see §7)
  - Extract counts in pulse height bins for a specified region
  - Fit model spectra to the extracted data
  - Plot data and best-fit spectra
  - Plot χ^2 contours showing 90% confidence region, etc.
  - Determine fluxes and luminosities based on the spectral fits
- Timing analysis (see §8)
  - Extract counts in specified time bins
  - Perform periodic and aperiodic timing analysis

In all cases the user has to read in the data (cf. §3.2) and, if necessary, generate IRAF compatible files (cf. §3.3).

2 IRAF/PROS

The Post-Reduction Off-line Software (PROS) is a portable x-ray analysis software system designed to run under NOAO's Image Reduction and Analysis Facility (IRAF).

The PROS system provides IRAF users with tasks that will properly access the x-ray data. In particular, it supplies the tools to take into account the additional parameters peculiar to x-ray imaging data (photon energy and time information may be retained in x-ray images, along with positions), and algorithms for a correct treatment of the errors, based on photon counting statistics. While written specifically for x-ray data, many of the PROS tasks can be used with other data as well. Similarly many IRAF tasks will be useful for x-ray data.

The PROS software includes spatial, spectral, timing, data I/O and conversion routines, plotting applications, and general algorithms for performing arithmetic operations with imaging data. *SAOimage*, an interactive data display system running under the X-window system, is also fully compatible with the PROS/IRAF system.

The IRAF system, previous to the development of the PROS software, provided many of the features needed to analyze generic astronomical data, optical data in particular. As a result of discussions involving the PROS software development group and the IRAF system developers, features have been added to the 'core' IRAF system to support the development of the software for the analysis of x-ray data. Foremost among these new features is the Quick Position-Ordered Event (QPOE) data files, introduced as of the IRAFv2.8 release. These are lists of photons identified by several quantities, including: arrival time, energy and position in the detector. Despite the name, QPOE files can be sorted (cf. qpsort §4.4) on time or any other component of the event description.

While the general syntax of the PROS tasks conforms to the IRAF conventions, new advanced features have been added (cf. §2.6). These include the availability and use of region descriptors integrated with a sky coordinate system. A detailed description of these new features, as well as a full description of the QPOE files, can be obtained from the corresponding on-line help files (*help, regions, qpoe, coords* or see *'help pros'* for a full list of help files). The algorithms used in various tasks are documented and are available on line.

2.1 Summary of Tasks in this Section

Task	Function	Package Needed
?	list contents of current package	
??	list contents of all loaded packages	
bye	delete last package loaded	language*
ehist	edit last command	language*
eparam	edit parameter file	language*
hedit	edit image file headers	images
help	display help file for specific task or package	system*
logout	leave IRAF environment	language*
lparam	list parameters for specific task	language*
package	list packages loaded	language*
phelp	help using screen pager	system
qphedit	edit .qp file headers	xray:ximages
tedit	edit table data	stsdas:ttools
tupar	edit table headers	stsdas:ttools

non-IRAF tasks

cd	change directory (UNIX, also works in IRAF)
mkdir	create a subdirectory (UNIX, also works in IRAF)
mkiraf	install IRAF setup
cl	enter IRAF environment

*User has automatic access to these packages when in IRAF.

2.2 Bringing up IRAF

Before running IRAF, there is a simple set-up procedure which is computer- and operating system-dependent. The necessary steps to obtain the correct set-up for a SUN workstation are listed below. Specific instructions should be obtained by the user from local computer system and IRAF managers.[1]

Tasks to be performed:

1. create a user directory from which IRAF will be run (called 'myiraf' in these examples). This will contain the user set-up specifications and other IRAF-related subdirectories.

2. create an image directory in a disk partition where there is enough space to contain large files (image arrays). Usually this is assigned by the system manager in a scratch area, periodically cleaned by the system. Data files should be kept in this image directory (*imdir*); the IRAF **.pix* files (see §2.5.1) are automatically stored in *imdir*.

3. move into the 'myiraf' subdirectory, and run the procedure *mkiraf*. This creates a file (*login.cl*) which executes the set-up each time a new IRAF session is started. *mkiraf* should be run every time there is a new release of IRAF.

An example execution of these tasks appears below.

To customize a few of the default options (*e.g.*, type of editor, default printer), the user has two options:

- edit the *login.cl* file. This requires editing and reassigning user choices every time *mkiraf* is run since *mkiraf* initializes *login.cl* at each execution of *mkiraf*.

- create a *loginuser.cl* file that contains user customizations. The assignments in this file override those in *login.cl*. The file needs to be changed only to reflect changes in the user choices.

IRAF can be run from any terminal. However, graphics or image display capabilities require the appropriate hardware.

Throughout this document we use 'visitor' as the username, '/pool' as the scratch area, '/pool/visitor' as the image directory (imdir) and 'myiraf' as the sub-directory from which IRAF will be run (home).

[1]For set-up on SAO computers see the *SAO-HEAD IRAF Users Site Guide*.

2.3 Using IRAF

A comprehensive introduction to the IRAF system can be found in A Users Introduction to the IRAF Command Language(cf. §10). A brief description and a few examples are given below.

The command *cl* starts IRAF.

```
    NOAO SUN/IRAF Revision 2.10.4-p1 Wed Aug 30 20:58:43 MST 1995
    This is the EXPORT version of Sun/IRAF V2.10.4 for SunOS-4.1.

********************************************************************************
IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF IRAF
********************************************************************************
                              SunOS VERSION
********************************************************************************

    Welcome to IRAF.  To list the available commands, type ? or ??.  To get
    detailed information about a command, type 'help command'.  To run a
    command or load a package, type its name.  Type 'bye' to exit a
    package, or 'logout' to get out of the CL.  Type 'news' to find out
    what is new in the version of the system you are using.  The following
    commands or packages are currently defined:

      adccdrom.   dipspc      hrc.        obsolete.   spptools.   xray.
      bjw         euv.        images.     plot.       stsdas.
      ctio.       ftools.     language.   proto.      system.
      dataio.     gethri      lists.      rvsao.      tables.
      dbms.       getpspc     mem0.       saotdc.     utilities.
      dihri       how         noao.       softools.   veta.
cl>
```

A few explanations about how to find your way in IRAF:

The IRAF user interface or command language is called the *cl*. The system is composed of 'packages', each itself composed of packages and commands ('tasks'). The packages currently available are listed at the time IRAF is first called, as in the example above. To select a listed package or run a listed task, type its name. Names can be shortened to one or more letters if they uniquely identify the task.

To run a task, first load the package that contains it ("help *taskname*" will display a help file which includes the name of the package in which the task can be found). When loading a package for the first time, the tasks under that package (the package menu) are automatically displayed. Several packages can be loaded at one time, the system will remember them all. However, it is good practice to unload unnecessary packages, to avoid running out of memory. Packages can be unloaded by typing *bye*. However, they can only be unloaded in reverse loading order. So, for example, if *dataio* has been loaded before *noao*, typing *bye* will unload *noao*. A second *bye* will unload *dataio*.

Every time a new package is selected, the prompt will change to reflect this selection.

To leave the IRAF environment type:

```
  cl> logout
  cfa#
```

2.3.1 Help

A *help* command is provided.

```
  cl > help pros  will give an overview of useful help tasks and
files in the  PROS/XRAY packages.

  cl > help package-name will give one line of explanation for each
of its tasks or packages.

  cl > help task-name will give detailed information about that task.
This is also the best way to find the package in which a task resides.
```

The following commands are useful for package management:

```
  cl > ?              # Lists tasks and subpackages of the current package
  cl > ??               # Lists all tasks and subpackages currently loaded
  cl > bye    # Unloads the current package, returning to the previous one
```

2.3.2 Parameters

IRAF supports two kinds of parameters, auto parameters and hidden parameters, which are treated differently.

- *auto parameters* are updated every time a task is run. The user will be prompted for them unless they are explicitly given on the command line.
- *hidden parameters* are those which can be reasonably defaulted, *i.e.* are often the same from one run of a task to the next. They need to be explicitly changed for choices different than the default. They are denoted in parameter listings by parentheses.

Parameters for each task are contained in 'parameter files' in the *uparm* sub-directory in 'myiraf'. A copy of each file with the latest parameter setup is saved after use. A list of parameters for a specific task can be obtained by typing:

```
cl > lparam taskname
```

For example, there are several ways of typing commands to examine the header of an IRAF image *e.g. dev$pix* [a test image within the IRAF system]:

```
cl > images                                                   # load package

im > imhead                       # Will prompt the user for the input file
im > imhead dev$pix                  # will display the dev$pix.imh header
im > imhead dev$pix longheader=yes     # A more verbose display obtained
                                          # by changing the value of the
                                           # hidden parameter longheader
im > imhead dev$pix.imh long+     # Alternative way to set the parameter
                                                         # longheader
```

Hidden parameters can be permanently changed in two ways:

```
im > imhead.longheader=yes   # explicitly setting <task.param>=<value>
im > eparam imhead     # Edit the parameter file to update one or more
                                        # parameters at the same time
                                                  # see next section
```

2.3.3 Editing Text

- Editing text on the screen

  The history of recent commands can be edited using the command *ehist.* This is useful when using long commands, each only slightly different from the previous one. The editor is set in the *login.cl* file or in the *loginuser.cl* file.

```
cl > ehist                                         # Calls the last command
                                               # and leaves you in the editor
                          # <cr> executes the command once you have edited it.

cl > ehist imcnts         # calls for editing (as above) the most recent
                                   # command beginning with ''imcnts''.
```

  Once in the editor you can step backwards/forwards through the history as when editing a text file.

- Moving text on the screen

  Xwindows

  Mouse buttons can be used to copy text. The cursor shows the location of the next command. With the mouse, move the cursor to the desired text. Highlight the text by left click and drag. A middle click will deposit the highlighted text at the current prompt position.

  Sunview

  Text can be copied in sunview by marking the beginning and end with the left and middle mouse buttons respectively. The text is now in the "stuff" buffer and can be "stuffed" to any window at the current prompt position by clicking on the right mouse button inside that window and selecting *stuff*.

2.3.4 Listing and Printing Files, Parameters, and Tables

To display ascii files, use the IRAF command 'page'

```
cl> page filename
```

Type "q" to exit 'page'.

Task parameters can be listed with 'lparam'

```
cl> lparam qpcopy
```

Table files are displayed using tprint

```
ta> tprint rp90_prd.tab
```

With UNIX-like pipes, these can be used in conjunction with hardcopy print commands like lprint:

```
ta> tprint rp90_prd.tab | lprint
```

2.3.5 Editing Files and Parameters

- *eparam* – edit parameter files

  Accesses the parameter file (see Parameters §2.3.2) for the specified task in an editor. Move the cursor to the item to be changed, type changes, and hit a carraige return. When finished, exit with <ctrl> x (emacs) or <ctrl> d (vi). Type "<esc>-?" within a parameter editor for more details on commands.

- *tupar* – Edit table headers

  Table files (**.tab*) consist of a header including a variety of parameters and the table itself. It is possible to edit the header using *tupar* and the table with *tedit* provided you have write-access to the file. A look at the help file is highly recommended for *tupar*.

- *tedit* – edit table data

  Allows editing of *.tab* tables. Works like emacs but uses <ctrl> D to exit.

- *hedit* – edit image file header

- *qphedit* – edit *.qp* file headers

Parameters may also be set, for the current command-line execution only, by indicating the parameters' temporary value on the command line:

```
imhead xdata$rp110590_img  longhead=yes
```

This will not change the value stored in the parameter file.

Boolean parameters may be set with shorthand notation.

```
imhead xdata$rp110590.img longhead+
imhead xdata$rp110590.img longhead-
```

2.4 Using the PROS software

The PROS software is stored under the IRAF package *xray*, and is divided into several subpackages (*e.g.*, spectral, spatial, timing, etc.). Once the *xray* package is loaded all the sub-packages are accessible. To load the PROS software:

```
cl> xray
        []----------------------------------------------------[]
        []            PROS RELEASE VERSION 2.4                []
        []                 October, 1995                      []
        []                                                    []
        []      Before using this version rerun mkiraf        []
        []                                                    []
        []           For answers mail hotseat                 []
        []----------------------------------------------------[]

The following general information is available via the "help" command:
        help topic:     description:

        pros            description of all help options available in pros
        bugs            description of known IRAF/PROS bugs
        coords          description of IRAF coordinates used in PROS
        exposure        description of PROS exposure correction support
        extensions      description of conventions for PROS file extensions
        explain_errors  description of PROS error computations for low
                        count/bin data
        explain_xdata   description of new RDF file formats
        explain_screen  description of PROS support for temporal screening
        extensions      description of conventions for PROS file extensions
        file_compare    comparison of new/RDF formats to OLD formats in PROS
        filter          description of filtering options in PROS
        missions        list of missions and instruments support by PROS
        prf             information on the Point Response Function (or psf)
        qpoe            description of the QPOE data file and user interface
     *  rosat_files     description of ROSAT data files which can be input
                        to PROS
        regions         description of PROS region masks
        release_xray    release notes for this build
        xspec_interface description of how to convert data into XSPEC
                        PHA files

      xapropos    xdemo       xinstall    xobsolete.  xproto.     xspectral.
      xdataio.    ximages.    xlocal.     xplot.      xspatial.   xtiming.
```

```
xr>
```

After the *xray* package is loaded, the following IRAF logical variables are defined.

- *xdata*: is a directory containing *Einstein* and ROSAT sample x-ray data files which are available for testing the PROS tasks and their use.
- *xdemo*: is a directory containing demos that use the PROS software. These are useful in demonstrating the PROS software and may serve as templates for users.

IRAF uses $ as a delimiter between logical variables and physical names *i.e. xdata$snr.imh* addresses the image *snr.imh* in the directory pointed to by the IRAF variable, *xdata*.

2.5 IRAF/PROS Files

2.5.1 IRAF Image Arrays

Image arrays (*e.g. dev$pix*) are the standard IRAF data files. They are composed of 3 separate files which are linked together by IRAF:

- **.imh* file, stored in the current directory, contains only the header and a pointer to the associated *.pix* file.
- **.pix* file, always stored in the image directory *imdir* (which the user specifies in the login.cl file), contains the data. These are large files, whose size depends on the size of the data array (*e.g.*, over 1 Mbyte for a 512×512 field).
- *..*.imh* file, stored in the current directory. A link used by IRAF to protect the first two files.

By default, **.imh* files are stored in the current directory. The associated pixel files are stored in the image directory (imdir) as they are often large. IRAF can only access a **.pix* file via the **.imh* file which points to it. The two are always treated together by IRAF *e.g.* if you write **.imh* to tape in FITS format, the associated **.pix* files are written as well. These files should only be manipulated from inside IRAF to correctly preserve the links, *i.e.* use *imdelete* and *imcopy* and NOT their *unix* equivalents.

Choices about printer, graphic device, editor, user's 'myiraf' directory (identified as *home* by (see §2.2)), and image directory (identified as *imdir* by IRAF), are contained in the *login.cl* file or *loginuser.cl* and can be edited as required.

2.5.2 QPOE Files

The x-ray data are stored in QPOE files (Quick Position-Ordered Events, **.qp*) rather than image arrays. These are lists of photons identified by several quantities, including the position on the detector, pulse height, and arrival time. Note that, unlike IRAF images, QPOE files have no associated header file, and are always stored in the current directory, unless explicitly specified otherwise. Non-PROS IRAF tasks can also access QPOE data files in place of image arrays.

A ROSAT QPOE file contains all the basic information for an observation (*i.e.*, WCS Pointing Information; START/END information, standard processing results, live time, and instrument parameters). In addition, the temporal status (*i.e.*, viewing geometry, background rate, aspect error, SAA...) is given at regular intervals.

The size of a QPOE file is determined by the number of events and the number of photon qualifiers retained. The recognized names (and types) of the elements of the event structure are:

```
name:          type:          description:
x              short          x position in pixels (on the sky)
y              short          y position in pixels (on the sky)
pha            short          pulse height amplitude in channels
pi             short          pulse invariant bin in channels
time           double         photon arrival time in seconds since
                                 the header reference date
detx           short          detector x position (with instrument corrections)
dety           short          detector y position (with instrument corrections)
rawx/dx        short          raw detector x position (RDF/rev0)
rawy/dy        short          raw detector y position (RDF/rev0)
region         short          region number (defined after a qpcopy
                                  with several regions)
```

The following aliases are recognized by PROS tasks that create QPOE files to obtain files of different sizes (qpcopy, qpsort, etc.):

```
image or peewee           x,y
energy or small           x,y,pha,pi
time or medium            x,y,pha,pi,time
detector or large         x,y,pha,pi,time,detx,dety
region                    x,y,pha,pi,time,detx,dety,region
full                      x,y,pha,pi,time,rawx,rawy,detx,dety,status
```

PROS tasks often call for these aliases in the 'EVENT DEFINITION' parameter.

As indicated by the aliases, different scientific objectives require different information, and consequently different size QPOE files.

All IRAF tasks which take IRAF image data as input will also accept **.qp* files. However, when **.qp* and **.imh* files are both present with the same root name, IRAF will assume the **.imh* file as the default, thus the *.qp* extension must be given to access the QPOE file.

Tasks which input and output an image file will by default assume the same format as the input file. Since non-PROS IRAF tasks cannot write **.qp* files, **.imh* extensions must be explicitly added to the output file name when the input is a **.qp* (see §2.5.3).

If the user does not wish to use all of the photon events in a QPOE file, a <u>filter</u> may be applied to the data. The filter is appended to the **.qp* filename in brackets.

Filter syntax is as follows:

The event structure element is named, followed by an "=" sign, followed by either the specific value or the range limits separated by a colon. For example:

rp110590.qp[pi=2:30]

refers to all the events in rp110590.qp with pi energy values between 2 and 30, inclusive.

The following event structure elements may be filtered: x, y, pha, pi, detx, dety, rawx (dx), rawy (dy), region. More than one event structure elements may be specified in a filter, if each is separated by a comma. For example:

rp110590.qp[x=200:300,y=512:612,pi=7:12]

Note that the order is unimportant.

qpcopy and such tasks that create output qpoe files do not allow blocking with such filters. Such an execution will produce the following error:

```
xi> qpcopy "xdata$rp110590.qp[bl=30,pi=5:24]" none /pool/copy large
```

```
QPEX warning: bad attribute name 'bl'
```

```
xi>
```

For more information see filters and qpoe files see §2.5.4 , or type "help qpoe" in IRAF.

xdisplay allows both blocking and filtering with the following syntax, for example:

```
xi> xdisp "xdata$rp110590.qp[bl=30][227:237,307:317]"
```

Note this "filter" is actually a section, where the coordinates are expressed in 'logical' coordinates, those of the image after blocking.

For more information see physical, logical, and world coordinates, see §2.6.1 .

Note, again, that the order doesn't matter.

```
xp> xdisp "xdata$rp110590.qp[227:237,307:317][bl=30]"
```

will also work.

Other filters may be added as well

```
xp> xdisp "xdata$rp110590.qp[227:237,307:317][bl=30,pi=7:24]"
```

Sections are placed in separate square brackets.

A word about blocking:

Blocking factors allow one to compress qpoe files so that application programs "see" an image of smaller dimension (blocking is <u>NOT</u> allowed when inputting QPOE files to a single event access task). The compression is performed by pixel summing so that, for a block=16, for example, 16×16 pixels values are summed to one pixel. This would serve to compress, in our example, a 4096×4096 image to a 256×256 image.

The ROSAT PSPC qp file images are 15360×15360. ROSAT HRI are 8192×8192. As many display in a standard (512×512) SAOimage window, some blocking expedites the process.

A blocking factor of 30 would then compress a PSPC image to 512×512, while HRI's are compressed to 512×512 with bl=16.

Pixel size for both HRI and PSPC ROSAT images is 0.5 arcsecs on a side. If you create new images with blocking factors, the CDELT1 and CDELT2 variables of the image header will be updated to contain the present pixel size (in decimal degrees) of the array.

To view your present image pixel size, you might use a command such as

```
  xr> imhead xdata$rp110590.qp long+ | match CDELT

CDELT1 = -1.3888890000000E-4

CDELT2 = 1.3888890000000E-4

  xr>
```

2.5.3 **.imh* and **.qp* Extensions - when to use them

The general rule in IRAF is that extensions to image names are not needed. However, with the introduction of QPOE files, both **.qp* and **.imh* have become defaults to most tasks. A few things to bear in mind for correct results when the extensions are not explicitly given:

- often it is clear from the prompt which extension is expected, *e.g.*

  ```
  sp> qpspec
  ```

  `source qpoe file:` xdata$rp110590
  needs a QPOE file.

  However, when in doubt, give the full name with extension.

- when **.imh* and **.qp* files are both present, most IRAF tasks assume the **.imh* file as default. *e.g.*
  cl> imhead rp90
  reads *rp90.imh* header, if **.imh* and **.qp* are both present, *rp90.qp* otherwise.

- when mixing **.qp* and **.imh* files, extensions must be explicitly given: *e.g.*
  im> imcopy xdata$rp110590.qp rp90.imh

- when the name of the image contains a '.', then give the extension explicitly: *e.g. xdata$rp110590.err.imh*
- tasks like *imcalc* require full extensions to the file names
- when the extension is different from the default one, full names and extensions must be given

2.5.4 Differences Between QPOE Files and Image Arrays

QPOE files are event lists. They store information relevant to the arrival of each photon such as position, energy, arrival time, etc. Image arrays are arrays of numbers representing total counts at each position - even if that count may be zero. IMAGE ARRAYS contain SPATIAL INFORMATION only and do NOT contain information such as arrival time or energy. Confusion as to filtering, and input and output formats (qp vs. imh) for various tasks can be alleviated keeping these facts in mind.

- **hedit**

  The task *hedit* allows the editing of the header parameter in a image file, but does not work on QPOE files. It looks as though it works properly but it does not. The PROS task *qphedit* updates QPOE headers.

- **Storing of images in the data directory**

  IRAF image data is stored by default in the image directory *imdir*, *i.e.*, **.pix* files are written in *imdir* while **.imh* files are written in the current directory, and indicate where the data is stored. No such structure exists with QPOE files, *i.e. *.qp* files are stored in the current directory. Make sure there is space in your directory before trying to create a QPOE file (for example from *xpr2qp*).

- **Filtering on QPOE files**

  On the command line, filters of the PROS type on QPOE files must be specified within quotes. *e.g.*

```
im> imcopy "xdata$rp110590.qp[pi=2:10]" foo.imh        # the input file
                                                           # is rp90.imh
```

but in interactive mode no quotes are needed

```
im> imcopy
Input images: xdata$rp110590.qp[pi=2:10]                    # No quotes
```

However, standard IRAF region selections do not require quotes. *e.g.*

```
im> imcopy xdata$rp110590.qp[100:200,100:200] foo.imh
```

If in doubt, include quotes. *e.g.*

```
im> imcopy "xdata$rp110590.qp[100:200,100:200]" foo.imh
```

will work correctly.

Filters may be combined within brackets, *e.g.*

```
im> qplist "xdata$rp110590.qp[pi=10:20,x=2000:3000,y=2000:3000]"
```

When blocking and filtering is done simultaneously, the blocking factor is placed in its own pair of brackets, separate from the filter (see §6.7.2).

- **Filtering of the QPOE kind**

  Filtering commands such as 'block=' only work with PROS software on QPOE files. Unfortunately, however, they are accepted by regular IRAF tasks, *i.e.*, no error message or warning will result, so that one might assume they worked properly. *e.g.*

```
im> imcopy ''rp90[block=2]" foo
```

  produces a file *foo.imh* of the same size as *rp90.imh*.

2.5.5 Other Types of Files

Other files used by the PROS software, besides IRAF array images, are the PLIO masks and TABLE files . These files can be used to filter selected regions in the image, to accept only regions in the image that have been uniformly exposed (see§6.5.2) , or to correct the images for vignetting (see §6.5.1).

N.B.: Note that the PLIO masks are appropriate only for the reference image used to create them, *i.e.*, any change in the **dimension** of the reference file requires that a new mask be made. Tabular data output from PROS tasks are written to TABLE files. A set of tools to view, plot, and manipulate tables is available in the TABLES package.

2.6 Features of the PROS software

A few new features and enhancements to the standard IRAF system have been introduced with the PROS software:

- extensions: Many IRAF tasks will automatically supply the correct default extension for the expected input. In addition to these, the PROS software differentiates files of the same kind (*e.g.*, tables), with *compound extensions*, the first part giving the PROS type, with the syntax *_xxx* (*e.g.*, *_fft*), the second the IRAF file type (*e.g.*, *.tab*).

  Default compound extensions are used by the PROS tasks when no extension is given for the output file name. Most PROS tasks allow the output file name to be defaulted, with the convention that the output file name will be composed of the input 'root' name, plus the default extension specific to the task. In general this is obtained by typing a carriage return. For example, if "rp110590.qp" is the input image file to imcnts, the default output file will be "rp110590_cnt.tab"(cf. §6.3 for details of obtaining count distributions.)

- directory path names: Directories in IRAF are delimited by "/" or "$". If a file is specified as 'directory/filename<input ext>', the output filename will be 'directory/filename<output ext>. For example, if the input file to the *imcnts* task is */pool7/rp110590.qp*, the default output table would be */pool7/rp110590_cnt.tab*. For the output table to be written in the current directory, the "." string should be used instead of the null string. If the response to "output filename" is a carriage return or "" (the null string), the directory names will be retained. Notable exceptions are the spectral tasks (cf. §7), in particular the fitting routines, in which the output table is written to the current directory by default.

- regions: Spatial filtering (cf. §6.2) is accomplished by means of region masks, in which each mask value corresponds either to a single geometric shape, or to two or more geometric shapes combined according to the rules of boolean algebra. Regions are used in virtually all PROS analysis tasks that use image data. Geometric shapes allowed are: circle, box, ellipse, polygon, annulus, pie slice, and rotated box (for more information type *help regions*).

- filters: The introduction of QPOE files, with spatial, temporal and energy information for each photon, allows the filtering of photons in time intervals or energy bands, together with spatial filtering. Most PROS tasks that work on QPOE files accept filtering of photons at run time (cf. energy filtering, §6.5.3.2).

- World Coordinate System: RA and Dec.

- Tables: Analysis results are written to tables.

More specific and detailed discussions of the PROS features can be found in the on-line help files, and in the on-line science specifications. In addition, the following help files may be useful:

XRAY — pros, bugs, coords, explain_errors, explain_xdata, explain_screen, exposure, file_compare, filter, extensions, qpoe, missions, release_xray, regions, prf (or psf), rosat_files, xspec_interface

XSPECTRAL — models_spectral, using_spectral, pspc_fitting

XSPATIAL — prf (or psf)

DETECT — explain_detect, using_detect

XPLOT — using_xplot

TIMCOR — using_timcor, utc

To obtain help, type 'help filename', e.g., "help coords".

2.6.1 Coordinate Systems

The IRAF environment provides three coordinates systems:

- physical: image pixel coordinates used in the reference (original) image.
- logical: image pixel coordinates, relative to the current section and size of the image. These are the units usually returned by tasks that read the image cursor (both in Imtool and *SAOimage*). These are the original (physical) image coordinates modified by transformations (such as sectioning, blocking) performed on the image.
- world: these are sky coordinates, typically celestial; however, other coordinate systems are also available. The PROS software supports Celestial, Ecliptic, Galactic and Supergalactic coordinates.

The PROS units for sky coordinates can be: decimal degrees, radians, ′, ″, hours:minutes:seconds (hms), or degrees:arcmin:arcsec (dms). The default epoch is that of the reference image (*i.e.*, B1950 for *Einstein* images, and J2000 for ROSAT images). However, the sky system may be changed by explicitly declaring the new system, as in the examples below.

The following syntax, (for the ROSAT test data target, AR Lac) can be used to specify source positions in the imcontour task (cf. "help imcontour"):

- 22:13:53.7 44:28:53.9 – the usual representation of RA, Dec
- 333.4737d 44.48163d – in decimal degrees
- 5.820213r 0.776351r – in radians
- b1950; 22:11:49.3 44:13:59.3 – in 1950.0 coordinates, Besselian equinox (*i.e.*. FK4 system).
- J2000; 22:13:53.7 44:28:53.9 – in 2000.0 coordinates, Julian equinox (*i.e.*. FK5 system).

Logical, physical and world coordinate systems are all supported in the PROS software, and regions can be specified with any combination of them. Sky coordinates are identified by the input format (*i.e.*, :, d, r, ′, ″). Coordinates in pixels are logical by default (this is also the IRAF default). To use physical pixel coordinates, you need to declare them as physical, as in the examples below (unique abbreviations are accepted).

For example, to specify a circular region (cf. §6.2) at the center of *xdata$rp110590.qp* [block=30] at the peak position :

- logical: numbers (real or integers) : l; c 231 313 8
- logical: numbers (real or integers) : c 231 313 8

- physical: numbers (real or integers) : ph; c 6930 9390 240
- world: any of the units described above: c 22:13:53.7 44:28:53.9 2′
- mixed units can also be used. Any of the following combinations will be accepted and interpreted correctly:
  - ph; c 6930 9390 2′
  - c 231 313 2′
  - c 22:13:53.7 44:28:53.9 8
  - ph; c 22:13:53.7 44:28:53.9 240

N.B.: When a qpoe file is displayed and no blocking or subsectioning has been applied, the physical and logical units are the same. However, if the input file is, for example, *xdata$rp110590.qp*[6675:7187,9135:9647] (selects a region from the central portion of the file only, reducing the file size to 512×512 while preserving full resolution), the values in the "physical" example above should be:

- physical: 6930 9390 240
- logical: 255 255 240
- world: 22:13:53.7 44:28:53.9 2′

In an example using the Einstein data, *xdata$snr.qp[block=2]*, the values would be:

- physical: 503 513 22.5
- logical: 251.5 256.5 11.25
- world: 22:59:04.9 58:36:55.4 3′

N.B.: Input can be in any of the above syntaxes, in physical, logical and world coordinates. If SAOimage is used to make a region file (§6.2) the output is in logical pixels, regardless of the input units. There is a task, *fixsaoreg*, which will convert these logical coordinates to physical coordinates, which free the region definition from the particular section used in SAOimage. Then a PROS task like imcnts can use the converted region file with any image or section derived from the original source image.

For more information see 'help regions', 'help regcoords', and 'help coords'. Also, see fixsaoreg, skypix, and wcscoords for information regarding coordinate conversion.

2.6.2 Treatment of Errors

The PROS system provides the capability to keep track of the uncertainties in image files through error arrays. For convenience, the squares of the errors are actually stored in these arrays.

There is a task, *errcreate*, that will estimate the errors in an image and produce an error array. Two estimates are available, selected by a parameter in the image header. The default estimate assumes that the counts in the image result from a Poisson process; this estimate is the most reliable one when there are few counts per pixel. The other estimate assumes that the counts result from a Gaussian process. See "help explain_errors" for details of the two algorithms and an example of how to change the default error calculation from Poisson to Gaussian.

Some tasks that transform images, such as *imsmooth, imcompress*, and *imreplicate* have the option to produce a resultant error array, derived from the original image's error array. Note, however, that the *imcalc* task does not produce an error array. The user is responsible for making one that reflects the operations that have been performed on the image. Some analysis tasks such as *imcnts* and *imcontour*, can use error arrays as part of their input.

See section on errcreate (§6.5.4) for further discussion, and read the help files "explain_errors" and "errcreate".

2.6.3 Possible Pitfalls

This section is an attempt to keep track of things that might cause confusion, errors, etc., because of the evolution of the PROS software. Most of these items are reported in earlier sections.

- Missing parameter

  An error involving a missing *tasks* parameter indicates that the parameter file may have changed between the last time the task was used and the current version. Deleting the *uparm* directory every time a new build is released is a drastic way to solve the problem. However, the problem can be dealt with on a task by task basis, by typing *unlearn 'task'* to reset the parameter file to the 'current' default. If still in trouble, delete the parameter file in 'myiraf/uparm' corresponding to the task (the names are somewhat contrived, but with a bit of imagination it is not too difficult to identify the right one).

- Exposure and vignetting masks

  Note that the exposure and vignetting masks are appropriate only for the reference image, *i.e.*, any change in the dimension of the reference file requires that a new mask be made. See 'help exposure'.

- Change of size in **.qp* or **.imh* files

  Any change in size, either through blocking factors or sections, will change the logical coordinate system. In other words, the logical center of the 1024×1024 original QPOE file (512, 512), will become (256, 256) when a block=2 is applied.

  Coordinates that you use to define a section of an image or QPOE file must be in the logical coordinate system of the affected file. For example, the "[200:300,200:300]" in "rp90[200:300,200:300]" are in the logical coordinate system of the rp90 image (which may differ from its physical coordinate system).

- The *imcnts* task and region masks

  It is important to understand how the task *imcnts* uses region masks. Regions given on separate lines, and not logically connected (*i.e.*, with a boolean operator), are treated as separate regions, but they are not necessarily independent. A problem may arise when there are overlapping regions in the same region descriptor. Each photon is assigned to only one region: the most recent region containing the photon's position. In other words, successive regions overwrite previous regions that they overlap. A region descriptor of the kind:

  CIRCLE(403, 403, 22.5)
  CIRCLE(403, 403, 11.25)

will produce a circle of radius 11.25 pixels (region 2) and an adjacent annulus of inner and outer radius 11.25 and 22.5 pixels (region 1). A region of the kind:

CIRCLE(403, 403, 11.25)
CIRCLE(403, 403, 22.5)

will produce only a circle with radius 22.5 pixels (region 2).

The first example will give an annulus and a circle; the second example will output only one region, region 2, since region 1 has been entirely written over. (See 'help regions'.)

- <u>Discrepancies in the output table from *imcnts*</u>

  Regions are identified differently in the table header and in the data section. In the header, each region specification is identified by a letter (or other ASCII character), while regions are numbered sequentially in the table. Since some region specifications may define multiple regions (annuli, pie slices), the header may contain fewer region specifications than the resulting number of regions in the table.

  The number of region specifications in the header is limited, by the tables package's treatment of ASCII characters (*e.g.*, conversion to upper case), to 33. When the limit is exceeded, the table will still contain the total number of rows and the correct results, but the header information will be unreliable.

- <u>File names</u>

  File names can contain arithmetic symbols *e.g.* '+', '–'. However, when files are input to tasks that perform calculations (*imcalc* is an obvious example), file names must be included in quotes, to avoid confusion. In particular, since *imcalc* does not <u>require</u> spaces between file names and operands, the quotes are essential.

  Region descriptor file names should not begin with upper or lower case 'B' or 'J' followed by a number whose value is between 1899 and 2101. (The excluded forms are taken as equinox specifications.)

- <u>Format of Real Numbers</u>

  Most IRAF commands will not accept leading dots in real numbers. .5 should always be written as 0.5 to avoid error messages.

- Unrecoverable Errors

  Sometimes a task won't run after an error occurs (often when ctrl-c is used to stop a task). Type 'flprcache' to stope the subprocesses left over from the aborted task, and then rerun the task in question; it's usually successful. Getting out, and then back into IRAF is the alternate solution when really stuck.

3 DATA I/O

In this chapter we discuss the tasks in *xray:xdataio* for the input and output of photon lists which are provided as FITS files (with or without a primary array) and which contain one or more extension tables - either ASCII or binary format. The tasks we describe will also access/convert primary and/or secondary images (arrays). (There is also a selection of FITS tools in the *ftools* package and other input/output tasks in the dataio package of IRAF.)

FITS provides the rules and framework for file structures to TRANSPORT data. Most implementations differ from each other in the selection of optional keywords and in the use of extension tables and images. For this reason, the Einstein Observatory FITS files are different from the US version of ROSAT data, and the MPE ROSAT files come in a third flavor. In order to minimize this disparity, the so called "Rationalized Data Format" (RDF) was constructed in 1993. Many of the optional choices available have been chosen in the hope that most X-ray missions will adhere to them, thus standardizing X-ray data formats to a greater extent. RDF is used for ROSAT data processed in the US and by MPE commencing April 1994.

To a large extent, the input tasks are designed to recognize differences between FITS flavors for a number of ROSAT processing stages, thus the user does not need to know the particulars of the data formatting. With the implementation of RDF, many modifications to the internal file structure of qpoe files were made. Thus if you have qpoe files on disk generated prior to PROS vs 2.3 (Dec 93 release), you should run *upqpoerdf*. For more information on RDF files, read the help file *explain_xdata*. Also, see the Data Products Guide for descriptions of ROSAT files.

3.1 Summary of Tasks in this Section

Task	Function	Package Needed
efits2qp	convert an Einstein event list FITS to QPOE	xray:xdataio
eincdrom	2 tasks to generate a qpoe file from cdrom FITS	xray:xdataio
eincdpar	set parameters for Einstein cdroms	xray:xdataio:eincdrom
fits2qp	convert a FITS file (containing BINTABLE extensions) into an IRAF/PROS qpoe file	xray:xdataio
hkfilter	Generate a QPOE time filter string from an input housekeeping filter	xray:xdataio
mkhkscr	Generate a housekeeping (TSI) filter	xray:xdataio
mperfits	convert MIDAS FITS images to IRAF image files	xray:xdataio
qp2fits	convert an IRAF/PROS QPOE file into a FITS file	xray:xdataio
rarc2pros	convert ROSAT ARCHIVE FITS files into a set of IRAF/PROS files	xray:xdataio
rfits2pros	convert a set of (disk-resident) ROSAT FITS files into a set of IRAF/PROS data files	xray:xdataio
t2d	transfer files from tape to disk	dataio
upimgrdf	convert a list of IRAF/PROS image files into RDF compatible format	xray:xdataio
upqpoerdf	convert a list of IRAF/PROS QPOE file into RDF compatible format	xray:xdataio
xrfits	Convert FITS data to IRAF IMAGE or TABLE files	xray:xdataio
xwfits	Convert IRAF image and table files to FITS files	xray:xdataio

3.2 Reading FITS files from tape

The ROSAT data format has now been carefully defined and unified so that data processed at all sites will be written to the same, well-defined set of FITS files known as RDF (Rational Data Format). All data processed (or re-processed) since June 1994 and archived since June 1995 is in RDF format. However, data in the earlier formats will remain available until re-processed and can still be read by PROS.

Table 3.1 below summaries these formats. Please see Section 3.3 for information on PROS input.

Table 3.1. Summary of ROSAT FITS Format Files

Name	PROS task	Dataset	Date	Comments
RDF	rfits2pros	PI	>June '94	Rationalized Data Format
	rarc2pros	Archive	>June '95	
FUSFITS	rfits2pros	US PI	<June '94	Rev0, GSFC
	rarc2pros	US Archive	<June '95	
RODFITS	rarc2pros	MPE,UK Archive	<June '95	Rev0, MPE

3.2.1 *t2d* – FITS tape to disk for files with binary tables

To read data from a ROSAT tape use the task *t2d* . Make sure you are in a directory with 10 Mbytes of space. If your data are in RDF format there will be 16 PSPC FITS files (8 HRI). If your data are in US Rev0 format, or is from the Rev0 archive there will be more FITS files.

Tape handling is site-dependent. Users at SAO should refer to the *SAO-HEAD IRAF Users Site Guide* for details concerning tape access. Assuming the user is set up so that a tape drive is available and accessible:

```
To mount DAT tapes at SAO:

cfa#: rsh reel                                  # remote logon to reel
cfa250: getdev mtn                            # assign tape drive mtq.
                    # The other choices are posted on the input drives.
cfa250: cd iraf                   # Enter directory with login.cl file.

cfa#: cl
cl> allocate mtn                 # IRAF allocation (this specific to SAO)
device mtn is already allocated            # ignore this message, it is
                                                     # allocated to you


cl> dataio
da> t2d
Input file descriptor: mtn
```

```
List of files:                              # carriage return (for all files)
Output file root name:  rostape
```

To conserve on disk space you can load only selected files from the tape.

US Rev0 FITS files used by PROS analysis are "2-12,14,25" for PSPC (HRI "2-6,8"). See the ROSAT Data Products Guide for file descriptions.

RDF FITS files are packaged in a smaller number of FITS files with multiple extensions in each file, see *explain_xdata* for a description; the tape list follows:

Detailed List of RDF FITS Files:

```
        PSPC RDF tape list

tape001 -> list of files on tape
tape002 -> rp110590n00_bas.fits     Photon event and good times
tape003 -> rp110590n00_im1.fits     Broad band image
tape004 -> rp110590n00_im2.fits     Hard (0.5-2 keV) band image
tape005 -> rp110590n00_im3.fits     Soft (0.1-0.4 keV) band image
tape006 -> rp110590n00_bk1.fits     Broad background image
tape007 -> rp110590n00_bk2.fits     Hard background image
tape008 -> rp110590n00_bk3.fits     Soft background image
tape009 -> rp110590n00_mex.fits     Exposure map
tape010 -> rp110590n00_ime.fits     Energy-coded image
tape011 -> rp110590n00_src.fits     Source analysis results
tape012 -> rp110590n00_anc.fits     Orbit, aspect and housekeeping data
tape013 -> rp110590n00_his.fits     Processing history
tape014 -> rp110590n00_prt.fits     Postscript information
tape015 -> rp110590n00_raw.fits     Uncorrected events list
tape016 -> rp110590n00_ltc.fits     Source light curves

        HRI RDF tape list

tape001 -> list of files on tape
tape002 -> rh110267n00_bas.fits     Photon event and good times
tape003 -> rh110267n00_im1.fits     Image
tape004 -> rh110267n00_bk1.fits     Background image
tape005 -> rh110267n00_src.fits     Source analysis results
tape006 -> rh110267n00_anc.fits     Orbit, aspect and housekeeping data
tape007 -> rh110267n00_his.fits     Processing history
tape008 -> rh110267n00_prt.fits     Postscript information
```

The differences between RDF, US Rev0, MPE Rev0 data are described in *help file_compare*. A detailed list of FITS files in all three formats is gven in Section 12.

3.2.2 *xrfits, mperfits* – FITS Arrays on tape to IRAF (.imh) files

After the 'allocate mtn' command (Sect.3.2.1)

```
cl> xray
xr> xdataio
xd> xrfits mtn 3 visitor_a  # load tape file 3 and name it visitor_a.imh
xd> xrfits mtn 1-10 visitor        # load files 1 to 10 in visitor_0001
                                                     # to visitor_a.imh
frame visitor.0001 average frame [300,500]      # One line like this per
                                                          # file loaded
                      # .headers will be loaded in the working directory
        # *.pix files (the images) will be loaded in the image directory
xd> xrfits mtn 11,13-14 visitor            # A different loading command
```

If you have image arrays written by MIDAS, use *mperfits* :

```
xd> mperfits mtn 1-3 images # Convert the first three files on tape device mtn
```

When you have finished loading your files,

```
xd> rewind mtn
xd> deallocate mtn                             # IRAF release of tape drive
```

```
To dismount the tapes at SAO:

   cfa250: rlsdev mtn      #release the tape drive
   cfa250: logout                  #logout of reel
```

N.B.#1: File numbers in a single *xrfits* call must be in increasing order.

N.B.#2: To list the names of files 1 to 10 without loading the files, use *rfits* in dataio.

```
da> rfits mtn 1_10 make_im=no
```

N.B.#3: Dimensions of FITS Arrays.

There is sometimes trouble with dimensionality of FITS images. IRAF likes images to have 2 dimensions. To use imlintran, etc. if the input image thinks it has dimensions 3,4,..., it is necessary to copy the image and to modify the header. To find out the dimensionality of an image, type

```
im> imhead velalow        # gives short header for the image ''velalow''
velalow [512:512:1][real]:EINSTEIN XRAY DATA   # this response indicates
                                               # 3 dimensions
```

or this response might be,

```
velalow [512:512][real]:EINSTEIN XRAY DATA
```

which indicates 2 dimensions, a satisfactory input for iraf tasks.

If more than 2 dimensions, type

```
im> imcopy velalow[1:512,1:512] velalow  # will make 2 dimensional image
```

The parameter WCSDIM in the header must also be changed

```
im> hedit
     images: velalow
     field: WCSDIM
     value(3): 2
     WCSDIM 3>2: <cr>
     update velalow (yes): <cr>

im>
```

3.3 ROSAT data and PROS

There have been major changes throughout the system (PROS release 2.3) to accomodate the Rationalized Data Format (RDF). For ROSAT this has meant a change in the existing data file formats, and also a standardization between US, UK and MPE data-set formats; hence all Rev 2 ROSAT re-processed data will be in a common format, regardless of where it was processed. Information on the new ROSAT RDF data files can be found in *help explain_xdata*.

Of course, there is still a large quantity of ROSAT data distributed in the US REV0 format. PROS will continue to support input files in either format, but PROS tasks will always write output files in the NEW (RDF compatible) format. The PROS ROSAT FITS readers (*rfits2pros* , *rarc2pros*) will do any required conversions automatically. However, any EXISTING PROS QPOE files, must be converted manually by the user, with the task *upqpoerdf* .

```
xr> xdataio
xd> upqpoerdf *.qp
```

Existing image files can be updated with the task, upimgrdf .

```
xr> xdataio
xd> upimgrdf *.imh
```

Several of the header keywords are changed by this process. Also, in the case of the ROSAT/HRI the de-gapped HRI coordiantes (detx, dety) are calculated by applying the gapmap to the raw telemtry coordinates (rawx,rawy). These changes from existing US Rev0 format to RDF format are described in *help file_compare*.

One of the additional features of the new RDF format for ROSAT data is the inclusion of the 'rejected' events. Both PSPC and HRI do spatial and temporal screening of the event-list before analysis by the pipeline SASS processing system. Previously, only the photon-events that passed this standard screening process were available to users. With the RDF format and Rev 2 processing, an additional extension of 'rejected' events is available. Access to these events is described in section3.3.1.

Two new tasks, *hkfilter* and *mkhkscr* , are provided to allow users to build QPOE filters for non-standard event screening. Descriptions of these tasks and additional information on how to use and manipulate these data are available in *help explain_screens*.

N.B.: Time and Date ROSAT dates, in the headers, follow the FITS standard or European style, DD/MM/YY (not the American style, MM/DD/YY). The times of events are measured in seconds from the time of launch [21:06:50, day 152, yr 1990 (1 June 1990)].

Time is measured with the spacecraft clock. These times must be converted to UT or barycentered times for precise work. See *xtiming.timcor* packages for information.

3.3.1 *rfits2pros* – PI Data

Rfits2pros converts the FITS files for a ROSAT observation into a set of IRAF/PROS files that includes qpoe, image, and table files. *rfits2pros* reads data in either rationalized data format (RDF) or US Rev0 format. The FITS files must first be loaded from the tape distributed to PI's (see).*t2d*

To translate the FITS files to IRAF/PROS use *rfits2pros* as follows:

```
cl> xray
xr> xdataio
xd> rfits2pros rostape pspc
```

See *help explain_xdata* for a description of the RDF data files. See chapter 12 in this document for a list of the US Rev0 files. A description of all FITS files can be found in the ROSAT DATA PRODUCTS GUIDE.

The complete (unscreened) events are available in ROSAT data distributed in the rationalized data format (RDF). Run rfits2pros with the 'unscreened=yes' option to get an 'unscreened' qpoe (standard + rejected events).

3.3.2 Requesting Rosat Data From the Public Archive

To request ROSAT data from the public archive users should send an e-mail message to

NDADSA::ARCHIVES

or

archives@ndadsa.gsfc.nasa.gov

(**N.B.:** These addresses were current as of October 1995.)

The subject line should read

request rosat <data_type>

where <data_type>=FITS, BASIC, DERIVED, ANCILLARY, CALIBRATION or ALL.

You may also get the data via the WWW page of NDADS:

http://ndadsa.gsfc.nasa.gov/ndaads/archives.homepage.html

If you need to search their database, try:

http://hypatia.gsfc.nasa.gov/wisard/wisard.html

Data Files for Rev0 PSPC

NDADS DATA TYPE	File description	Sample file
FITS	photon list	wp110130_events.tfits
	broad-band image	wp110130_image1.ifits
	hard-band image	wp110130_image2.ifits
	soft-band image	wp110130_image3.ifits
	energy-coded image	wp110130_imageec.ifits
	Exposure map	wp110130_exposure.ifits
BASIC	photon list	wp110130_events.tfits
	rejected events	wp110130_difevents.tfits
DERIVED	broad-band image	wp110130_image1.ifits
	hard-band image	wp110130_image2.ifits
	soft-band image	wp110130_image3.ifits
	energy-coded image	wp110130_imageec.ifits
CALIBRATION	response matrix	wp110130_drmpspc.tfits
	instrument map	wp110130_moimp.ifits
	Off-axis area, pspc b	wp110130_effarea_pspcb.tfits
	Off-axis area, pspc c	wp110130_effarea_pspcc.tfits
	exposure map	wp110130_mexmap.ifits
ANCILLARY	Quality file	wp110130_quality.tfits
	attitude file	wp110130_attitude.tfits
	event rates	wp110130_eventrates.tfits
	orbit	wp110130_orbit.tfits
ALL	all the above	all the above

Data Files for Rev0 HRI

NDADS DATA TYPE	File description	Sample file
FITS	photon event list	wh110131_events.tfits
	SIMBAD sky list	wh110131_simbad.tfits
BASIC	photon event list	wh110131_events.tfits
	rejected events	wh110131_difevents.tfits
DERIVED	SIMBAD sky list	wh110131_simbad.tfits
CALIBRATION	effective area	wh110131_effarea_hri.tfits
	hot spot map	wh110131_hotmat.tfits
	quantum efficiency map	wh110131_qefficiency.tfits
ANCILLARY	aspect	wh110131_attitude.tfits
	event rates	wh110131_eventrates.tfits
	orbit	wh110131_orbit.tfits
	data quality	wh110131_qualitylimits.tfits
	data quality limits	wh110131_quality.tfits
ALL	all the above	all the above

The body of the letter should contain the dataset_id(s) in the form

w<i><ror>

where <i>= p for unfiltered PSPC, f for filtered PSPC, or h for HRI and
<ror> = the six digit ROR number.

You may request more than one dataset at a time

For example, to request FITS-type data from the datasets 200423 (filtered) and 200194 (unfiltered) a DECNet user would send the following e-mail:

To: ndadsa::archives
Subj: request rosat fits

wf200423

wp200194

<eof>

See ROSAT Status Report #56, available via anonymous ftp from legacy.gsfc.nasa.gov

N.B.: When RDF files become available in the archive, the details will change.

3.3.3 *rarc2pros* – Converting Archived Data to PROS files

Rarc2pros converts the FITS files for a ROSAT observation into a set of IRAF/PROS files that includes qpoe, image, and tab files. The FITS files must be on disk (see3.2.1) and the task works on the full set or a subset of the files from the archive in either rationalized data format (RDF) or Rev0 format (US or GERMAN).

```
xd> rarc2pros wp110590 pspc
```

3.4 EINSTEIN data

This section deals with tasks that are required to read *Einstein* data into PROS. The first section deals with reading FITS files from *Einstein* CD ROMs, the second deals with MPC data. Please note that several tasks described in previous versions of this document that convert original *Einstein* format files (*.xpr, .img, .gzm* and *.exp*) are now obsolete. The task *sdfspec* is also obsolete, as is the non-IRAF service *getedata*.

3.4.1 Reading *Einstein* FITS files

Einstein FITS files can be read into PROS using the standard IRAF and PROS routines, **dataio.rfits** for images and **xray.xdataio.efits2qp** for Event Lists. The user is prompted for the name of the FITS file and the dataset is drawn from (IPC Unscreened Data, HRI Event lists etc.) For example;

```
xl> efits2qp
input FITS file name: i2029n40.xpc
dataset (hrievt|ipcevt|slew|ipcu): ipcevt
output IRAF/PROS QPOE file name (.): test
```

```
Writing output QPOE file: test.qp

aux=.convtgr, type=convtgr -> test.qp
```

3.4.2 Access to CD-ROM FITS Files

3.4.2.1 eincdrom Much of the *Einstein* data archive has now been released on CD-ROM in FITS BINTABLE format (IPC Events, HRI Events, the Slew Survey IPC Unscreened Data) and in FITS image arrays (EOSCAT and HRI images). The routines in eincdrom are macros that use standard IRAF/PROS tasks to read the FITS files directly from the CD-ROM. The tasks in this section are

- **eincdpar** . This routine sets the package wide parameters, including the location of the CD-ROMs and the indexes on your local system. Listing the parameters gives the following:

```
ei> lpar eincdpar
     (ipcevt1 = "ipcevt1$")     Einstein ipc event cd 1 directory
     (ipcevt2 = "ipcevt2$")     Einstein ipc event cd 2 directory
     (ipcevt3 = "ipcevt3$")     Einstein ipc event cd 3 directory
     (ipcevt4 = "ipcevt4$")     Einstein ipc event cd 4 directory
     (hrievt1 = "hrievt1$")     Einstein hri event cd 1 directory
     (hrievt2 = "hrievt2$")     Einstein hri event cd 2 directory
     (eoscat1 = "eoscat1$")     EOSCAT cd 1 directory
     (eoscat2 = "eoscat2$")     EOSCAT cd 2 directory
     (eoscat3 = "eoscat3$")     EOSCAT cd 3 directory
     (hriimg1 = "hriimg1$")     Einstein hri image cd 1 directory
     (hriimg2 = "hriimg2$")     Einstein hri image cd 2 directory
      (slewcd = "slewcd$")      Einstein slew survey cd directory
       (ipcu1 = "ipcu1$")       Einstein IPC unscreened cd 1 directory
       (ipcu2 = "ipcu2$")       Einstein IPC unscreened cd 2 directory
       (ipcu3 = "ipcu3$")       Einstein IPC unscreened cd 3 directory
       (ipcu4 = "ipcu4$")       Einstein IPC unscreened cd 4 directory
       (ipcu5 = "ipcu5$")       Einstein IPC unscreened cd 5 directory
       (ipcu6 = "ipcu6$")       Einstein IPC unscreened cd 6 directory
       (ipcu7 = "ipcu7$")       Einstein IPC unscreened cd 7 directory
       (ipcu8 = "ipcu8$")       Einstein IPC unscreened cd 8 directory
```

```
       (ipcu9 = "ipcu9$")       Einstein IPC unscreened cd 9 directory
      (ipcu10 = "ipcu10$")      Einstein IPC unscreened cd 10 directory
      (ipcu11 = "ipcu11$")      Einstein IPC unscreened cd 11 directory
      (ipcu12 = "ipcu12$")      Einstein IPC unscreened cd 12 directory
      (ipcu13 = "ipcu13$")      Einstein IPC unscreened cd 13 directory
      (ipcu14 = "ipcu14$")      Einstein IPC unscreened cd 14 directory
      (ipcu15 = "ipcu15$")      Einstein IPC unscreened cd 15 directory
      (ipcu16 = "ipcu16$")      Einstein IPC unscreened cd 16 directory
      (ipcu17 = "ipcu17$")      Einstein IPC unscreened cd 17 directory
      (ipcu18 = "ipcu18$")      Einstein IPC unscreened cd 18 directory
 (ipcevt_info = "eincdromdata$ipcevt_info.tab") IPCEVT info table
 (hrievt_info = "eincdromdata$hrievt_info.tab") HRIEVT info table
 (eoscat_info = "eincdromdata$eoscat_info.tab") EOSCAT info table
 (hriimg_info = "eincdromdata$hriimg_info.tab") HRIIMG info table
   (slew_info = "eincdromdata$slew_info.tab") SLEW info table
   (ipcu_info = "eincdromdata$ipcu_info.tab") IPCU info table
 (ipcevtindex = "ipcevt1$lists/unix/seqnindx.asc") Einstein ipc event cd index
 (hrievtindex = "hrievt2$lists/unix/seqnindx.asc") Einstein hri event cd index
 (eoscatindex = "eoscat2$lists/unix/seqnindx.asc") EOSCAT cd index file
 (hriimgindex = "hriimg1$lists/unix/seqnindx.asc") Einstein hri image cd index
        (mode = "ql")
```

The path names must be correctly specified in order for eincdrom *tasks to work.* The default values are set for the HEAD CfA Network.

- **ecd2pros** is a task which converts CD-ROM FITS files to IRAF/PROS format, and puts the converted files in the users space. If an event list fits file is specified a qpoe file is created, if an image file is specified an IRAF image is produced. This task will access files on the latest *Einstein* CD-ROM set, the IPC Unscreened Data, including the auxiliary data files (see the help files "explain_eincdrom" and "help ecd2pros" for more details of auxiliary data). It replaces the tasks *fits_get* and *qp_get*.
- **ecdinfo** is a task which gives information on any FITS file given a sequence number or FITS file name. It gives the FITS root, the title of the observation, the RA and Dec of the field center, the livetime and the location of the CDROM directory.

3.4.2.2 Example – Creating a QPOE file from CD ROM Data Suppose you want to create a qpoe file (with standard screening) for sequence 2952. Use *ecdinfo* to tell you which CD ROM to use:

```
ei> ecdinfo
FITS filename OR sequence number (2952):
Einstein instrument (ipc|hri) (ipc):
Datatype (event|image|slew|unscreened) (unscreened): event

FITS root: i2357s64, extension: a, sequence 2952.

Title: I02952 MX2348-65 (MX2348-65: X-RAY TRANSIENTS AND NOVA-LIKE OBJECTS).

Ra=23:57:59.8, Dec=-64:05:59.5, livetime: 2020.57 seconds.

This sequence should be on the CDROM in the directory ipcevt4$.
```

The sequence you require is on the *4th* IPC Event Lists CD ROM. Load the CD ROM into your reader (assumed to be mounted as /cdrom), and set the location of the disk and with *eincdpar*

```
ei> eincdpar.ipcevt4="/cdrom/"
```

Then use *ecd2pros* to create the QPOE file;

```
ei> ecd2pros
input FITS file name OR sequence number: 2952
Einstein instrument (hri|ipc): ipc
datatype of FITS file (event|image|slew|unscreened): event
root name for output files (.): i2952

Using FITS file /cdrom/data/23h/i2357s64.xpa to create QPOE file.

Writing output QPOE file: i2952.qp

aux=.convtgr, type=convtgr -> i2952.qp
```

3.4.2.3 Example – Retrieve Unscreened Data from CDROM

Now retrive a QPOE file (AND auxiliary data files) for the 3C 327 (a radio galaxy) sequence 3476.

```
ei> ecd2pros.aux=yes                # get auxiliary data files
ei> ecd2pros
input FITS file name OR sequence number: 3476
Einstein instrument (hri|ipc): ipc
datatype of FITS file (event|image|slew|unscreened): uns
root name for output files (.): i3476

Using FITS file ipcu12$data/15h/i1559n02.upa to create QPOE file.

char strings not yet implemented - skipping
Writing output QPOE file: i3476_unsrtd.qp

i3476_unsrtd.qp[y=2:] (; no exp.) -> i3476.qp
Creating QPOE output file: i3476.qp

Converting FITS file : ipcu12$data/15h/i1559n02.upa[1]

Fits_file         IRAFNAME            Dimensions     BP   DATE   OBJECT
ipcu12$data/15h/  ... main FITS unit skipped.
                  i3476OBS01.tab      60    1     Ncols= 11 OBSTABLE
                  renamed to i3476OBS.tab

Converting FITS file : ipcu12$data/15h/i1559n02.bka[1]

Fits_file         IRAFNAME            Dimensions     BP   DATE   OBJECT
ipcu12$data/15h/  ... main FITS unit skipped.
                  i3476BKWCS01.tab    56    1     Ncols= 15 BK_WCS
                  renamed to i3476BKWCS.tab
.
.
.
.
.
.
```

.
.
.

3.4.3 MPC Data - *CfA Only*

The background-subtracted MPC spectra are on disk at CfA. To read MPC data into PROS, the data must be extracted from the archive and then converted to a table format suitable for the PROS spectral package. To extract the spectra from sequence 3388 type

```
cfa#: setenv MPCDATA /data/mpc

cfa#: /soft/mpc/mpcpros.macro 3388                        # outputfile is
                                                        # 3388.mpcpros

approximately 5 - 10 minutes are required to complete.
```

This is a **NON-IRAF task**

These data must now be converted to a table suitable for input to the PROS spectral package (**xlocal.mpc.mpcspec**).

```
mp> mpcspec
input MPC file: 3388.mpcpros
root name for output file [root_obs.tab]: m3388      #output file is
                                                     m3388_obs.tab
mp>
```

3.5 Other missions

PROS can read ASCA GIS and SIS event data into a QPOE data structure. Run *fits2qp* with the default parameters to translate the FITS data to QPOE format. See *help asca_data* for a description of analysis tools/paths available in PROS for ASCA data.

3.6 *xwfits, qp2fits, (fits2qp)* – Writing FITS files

3.6.1 Making a FITS primary array file or an ASCII table

xwfits can be used to make a fits file from any IRAF (.imh) image. It can also produce ASCII extension tables from any IRAF (.tab) table.

```
xd> xwfits visitor_b.imh mtn 1600          # writes the image 'visitor_b'
                                                 # onto tape at 1600 bpi.
```

3.6.2 Making a binary table of photons from a qpoe file

To make a FITS version of any qpoe file, use *qp2fits* . The file thus created can be recovered with *fits2qp* .

3.7 A Few Words About ROSAT Directories at SAO

To view an image or to work with data, you must be in the directory containing the data or specify the location of the data exactly (the pathname).

For example: Your directory might be /home/smith/ . At SAO, the ROSAT data for observation rp123456 will be in /data/rdata/godata/rp123456/ . The PROS package contains sample ROSAT and Einstein data which are in the directory /pros/xray/data which is loaded with the IRAF task "xray". To avoid typing long directory names within IRAF we have assigned "logical variables" which are recognized as aliases for the full directory name: "xdata$" for the directory with sample data.

Thus within an arbitrary working directory and within the IRAF environment, to look (see §5) at an image the command is,

```
tv> display "xdata$rp110590_im2" 1
```

If Einstein data are loaded at SAO they will appear in the directory /pool14/i1234/ . To view the data from Einstein IPC sequence 1234, type

```
tv> display /pool14/i1234/i1234.qp 1
```

4 QPOE Utilities

4.1 Summary of Tasks in this section

Name	Function	Packages Needed
evalvg	generate time history of satellite orientation	xray:xproto
qpappend	combine two or more *qp* files	xray:xdataio
qpcopy	copy *qp* file through filter	xray:ximages
qplintran	apply linear transformation to a *qp* file through filter	xray:ximages
qplist	lists file and header on screen	xray:ximages
qprotate	rotate *qp* file through filter	xray:ximages
qpshift	shift *qp* file by an offset in x and y	xray:ximages
qpsort	sort *qp* file through filter	xray:ximages

4.2 *qpcopy* – Manipulation of PROS/IRAF data files

Copy *.qp* file, usually through a filter, to make a new *qp* file. This example shows selection of two time intervals (or exclusion of all data not in these two intervals).

```
cl> xray
xr> ximages
xi> qpcopy

input qpoe file name: xdata$rp110590.qp[time=(1606720:1607484, 1611613:1612817)]}
region description: ``''    # This is the ``null string'' which will cause
                                            # entire image to be copied.
output filename: comal
event definition: medium # possibilities are: peewee, small, medium, large, region.
```

If the command line is too long, there may be problems. If many intervals are specified, the filter should be put in a file and the task directed to the file. *e.g.* Make a file *intervals* which contains the line:

time = (a:b,c:d,e:f,...).

Then type input file name

xdata$rp110590.qp[@intervals]

If the input file is time ordered (e.g., **_sti.qp*), there is a parameter which must be reset.

qpcopy.qp_mkindex = no

qpcopy will now operate on the time-ordered input file.

4.3 *qplist* – Inspection of *.qp* files

```
Input file:      xdata$snr.qp:

cfa#: cl                                        #enter IRAF environment

cl> xray
xr> ximages
xi> qplist xdata$snr events=no header=yes
```

This will display header on the screen.

To look at the contents of the file xdata$snr.qp, if in SUNTOOLS, first make sure page mode is enabled (press right button, cursor in window), then:

```
xi> qplist sdata$snr
```

Computer will list events in file.

Press space bar for next page, to quit type: ctrl c

If using xwindows, the above command will start the display of a (usually) long list of events. This display can then be stopped with `ctrl s`, moved through a limited page with the scroll bar and restarted with `ctrl q`. Again `ctrl c` will exit.

Or the file can be displayed one page at a time with

```
xi> qplist xdata$snr events+ header- last=20 | page

xi> qplist xdata$snr events+ header- first=1000 | page
```

Space bar will display next page, d will move down 1/2 page, q will quit.

To display temporal status interval records, use:

```
xi> qplist xdata$snr events- tsi+
```

4.3.1 *imheader* – Inspection of *.qp* files

The task *imheader* can also be used to display header.

```
input file:  xdata$snr.qp
```

```
cl> images
im> imheader xdata$snr long+   # long+ gives the long version of the output.
```

This will display information from header on screen including some items not given by *qplist*
.

4.4 *qpsort* — Sort a QPOE file

Sort a qpoe file using any event attribute. In addition, a filter and a region can be specified.

```
xi> qpsort
input qpoe file name: xdata$snr[pi=(1:7)]
region descriptor: ‘‘’’  # this is the ‘‘null string’’ which will cause
                                              # entire image to be copied
Output qpoe filename: test.qp
event definition: medium # possibilities are: peewee, small, medium, large, region
type of sort: position                          # sort the file spatially
```

This will sort the qpoe file with the y position as the primary key, and the x position as the secondary key. It is important to spatially qpsort a qpoe file that was not created from an IRAF/PROS FITS file (including MPE FITS files in the ROSAT archive). This is necessary if the qpoe file is to be used with a REGION mask.

To sort files with time as the sort key, the task *timsort* in the xtiming package (see §8.3) should be used.

To sort files according to other event attributes (e.g., pha, pi, etc.) the qp_mkindex parameter in qpsort must be changed to no.

```
xi> qpsort xdata$snr ‘‘’’ out.qp medium pi qp_mkindex=no
```

4.5 *qpshift* — Shift *qp* file by an offset in x and y

Transform a qpoe file according to the specified parameters. Position X1,Y1 is changed to:

$$X2 = X1 + Xshift$$
$$Y2 = Y1 + Yshift$$

where Xshift, Yshift is the amount of shift in pixels.

Any type of filtering can be applied to the input qpoe file, including region filtering. In addition, the size of the output event record can be different from the size of the input event record. (Of course, usually one makes the output record equal to or smaller than the input. If the output record size is made greater than the input record, a warning is issued and the "extra" part of the record is undefined.)

The following will shift the x-axis by 3.2 pixels and the y-axis by -1.65 pixels, making the resulting qpoe file of small size:

```
cl> qpshift xdata$rp110590.qp rp_shift.qp 3.2 -1.65 eventdef=''small''
```

The output *qp* file is generated with the same dimensions as the input, even when a region is applied. If no region is used, the translation may result in the loss of part of the image and a list of discarded events will appear. Note that QPEX warnings can be safely ignored.

4.6 *qprotate* — Rotate a QPOE image about the origin

Rotate qpoe image around an axis perpendicular to the image, normally about the origin (0,0). The angle of rotation is specified in degrees counter-clockwise. The sample shows rotation of the qpoe image by 20 degrees.

```
cl> xray
xr> ximages
xi> qprotate
input qpoe file name: xdata$snr.qp
region descriptor: <cr>
angle through which to rotate (0.): 20
```

```
output qpoe file name: xnr.rot.qp
event definition (large): <cr>
```

The output *qp* file is generated with the same dimensions as the input, even when a region is applied. If no region is used, the translation may result in the loss of part of the image and a list of discarded events will appear. Note that QPEX warnings can be safely ignored.

One practical application of *qprotate* is to align qpoe images in which the satellite had different roll angles. Two qpoe images can be aligned by rotating each of them by the negative of their roll angles. The roll angle, if present in the header, will be found in the parameter XS-DANG. This can be found by running *qphedit*, in the ximages package. For example, to find the roll angle of xdata$snr.qp, type:

```
xr> ximages
xi> qphedit xdata$snr.qp XS-DANG

The task will output the line:
```

xdata$snr.qp,XS-DANG=-51.3059

Then *qprotate* can be run on the qpoe file using +51.3059 as the angle. Applying the same process to another qpoe image will cause the support structure of the instrumentation to line up. Note that the roll angle is not available in all cases. When it is not, the parameter XS-DANG will generally have the value 0.0.

4.7 *qplintran* — Apply linear transformation to a *qp* file through filter

Transforms a qpoe file according to the specified parameters. Position X1,Y1 is changed to:

$$\begin{aligned} X2 &= Xout + ((X1 - Xin) * Xmag * cos(Angle) - (Y1 - Yin) * Ymag * sin(Angle)) \\ Y2 &= Yout + ((Y1 - Yin) * Ymag * cos(Angle) + (X1 - Xin) * Xmag * sin(Angle)) \end{aligned}$$

where Xout,Yout is the new reference point in pixels, Xin,Yin is the input reference point. Xmag,Ymag are the relative plate scales in X and Y (normally 1).

Any type of filtering can be applied to the input qpoe file, including region filtering. In addition, the size of the output event record can be different from the size of the input event record. (Of course, usually one makes the output record equal to or smaller than the input. If the output record size is made greater than the input record, a warning is issued and the "extra" part of the record is undefined.)

This task is modeled after *imlintran* but runs in qpoe files. Simple SHIFT and ROTATE functions can be performed with the specialized tasks above.

N.B.: The warning "QPEXwarning: bad attribute name 'time"' can be ignored.

```
    input file: xdata$rp110590.qp

    output file: arlac_lintran.qp


  xs> qplintran xin=7140 yin=7440 xout=7681 yout=7681

  input qpoe file name (xdata$rp110590n00.qp): xdata$rp110590n00.qp
  region descriptor (FIELD): FIELD
  angle (degrees) through which to rotate (0.): 20.0
  output qpoe file name (arlac_mag.qp): arlac_lintran.qp
  event definition: FULL

xdata$rp110590n00.qp (FIELD; no exp.) -> arlac_lintran.qp (xin=7140.00 yin=7440.00
                                  xrot=20.00 yrot=20.00 xout=7681.00 yout=7681.00)
Creating QPOE output file: arlac_lintran.qp
```

4.8 *qpappend* — Append a list of input qpoes to produce a single qpoe file

Read a list of input qpoe files (including filters) and produce an appended output qpoe file. The event lists and TSI records are appended; the DEFFILT (good time intervals) are merged; history is preserved, with additional history records for the append; ONTIME will be the sum of the ONTIMES; DTCOR will be the ONTIME weighted DTCORS; MJD-OBS,DATE-OBS,TIME-OBS will be the minimum of the input values. As currently implemented, many of the header keywords MUST match exactly in all of the append files. The output keywords which define WCS will be taken from the first file in the list. (See the help files for further details.)

Any type of filtering can be applied to the input qpoe file, including region filtering. In addition, the output event definition can be different (generally smaller) than the input event definition. If it is greater, a warning is issued and the "extra" part of the record is undefined.

The following will merge together the two observational pieces of AR LAC to form a single MERGED file. (This is obviously a contrived example, but can be used for testing.)

```
cl> xray
xr> xdataio
xp> page merge.lst        # an ascii file listing the two input qp files

xdata$rp110590.qp[time=(1606720.00:1607484.00)]
xdata$rp110590.qp[time=(1611613.00:1612817.00)]

xp> qpappend @merge.lst ‘‘’’ merge.qp large
```

@ is an iraf symbol to indicate that the task should use the contents of *merge.lst* as its input.

4.9 *evalvg* – Generate Time History of Satellite Orientation

This task computes the earth-satellite-target angle (EST) and the sun-earth-satellite angle (SES). The output is a table containing the time-history of these angles.

```
cl> xray
xr> xproto
xp> evalvg  xdata$rp110590r00 rp110590

Creating output file:  rp110590_ang.tab
```

The output table contains four columns: spacecraft time (sc_time), EST angle, SES angle, OBI number. This task works only with RDF format data and uses as input the tables: *_eph.tab, *_obi.tab.

5 Image Display

This chapter discusses image display, inspection and manipulation. Images can be displayed from IRAF both within the sunview (with Imtool) and Xwindow (with *SAOimage*) environments. Imtool is a standard IRAF tool provided by NOAO. *SAOimage* is part of the PROS package and will be described in full here.

5.1 Summary of Tasks in This Section

Name	Function	Packages Required
display	displays image on screen	images:tv
fixsaoreq	restore region to original image pixels	xray:xspatial
gridpars	allow edit of grid display parameters	xray:xplot
pspc_hrcolor	generate an x-ray color image	xray:xplot
rimcursor	coordinate conversion	lists
tabplot	plot tables, lists or image sections	xray:xplot
tvlabel	coordinates from cursor position	xray:xplot
tvmark	marks positions on screen	noao:proto
tvproj	overlays x,y projection on display	xray:xplot
tvimcontour	overlays contours/sources on display	xray:xplot
wcscoords	transform between various coordinates	xray:xspatial
xdisplay	displays image on screen	xray:xplot
xexamine	inspect, zoom, pan, project qpfile	xray:xplot
ximtool	initiate saoimage/imtool	xray:xplot

Non-IRAF tasks:

startx   puts user in x-window environment

5.2 Image Display in Xwindows

In any window, initiate iraf and load the xray package.

```
cfa#: cd iraf
cfa#: cl
cl> xray
```

The task *ximtool* which initiates the image display window is in the *xplot* package. It allows selection of one of three image displays:

```
xr> xplot
xp> ximtool
Display type = saoimage/ximtool/saotng(ximtool):saoimage
```

ximtool is a new Xwindow display tool developed at NOAO.
(Note: Please do not confuse the task and the tool which unfortunately have the same name.)

saotng i.e. "the next generation" SAOimage under development at SAO. It can be retrieved via anonymous ftp from sao-ftp.harvard.edu in the directory /pub/rd. The source tree is contained in a compressed tar file: saord.tar.z. The unpacked tar file requires ~45 Mbytes of disk space. The build requires ~90 Mbytes depending on the platform. For more information send email to: saotng@cfa.harvard.edu.

If you select *saoimage*, a window will appear, which has several sections:

- the main image display (containing the SAO logo)
- the menu
- the text window (upper left)
- a full image display showing which portion of the image is shown in the main display window (upper middle)
- a magnifier window displaying a close up of the region around the mouse position (upper right)
- a large text window (*SAOmonolog*) for status and verbose output

An image can now be displayed as described in §5.3 .

5.2.1 SAOimage for remote users

If you are using PROS/IRAF via remote logon to another computer, and if you have an imaging terminal, *SAOimage*, can be used to display data but <u>you must setup as follows</u>:

1. Initiate X windows on your home terminal before logging on to the remote computer. If you remote logon first, then initiate X windows, the windows appear on the remote computer, rather than your home terminal. This frequently kills the remote computer which then requires rebooting.

2. Set your computer name in the unix environment variable, DISPLAY (located in the .login file). To check this type the unix command:

```
cfa#: setenv DISPLAY=whouse.washington.gov:0.0
```

3. Initiate *SAOimage* using *ximtool* as described above. terminal.

```
xp> ximtool nasa#: 0
```

5.2.2 General usage.

The following list gives essential information for the new user. For more details see the following section.

- Menu: to select an option, click any mouse button on that option. Selected options are blackened. (Note: Several options bring up sub-menus).
- Quit: Select *etc* and *quit*
- New image: Select *etc* and *new*. Edit the command line to reflect the new image name (ignore the details!).
- Raise: To raise the window above its neighbors either:
  –type 'a' anywhere in the window
  –select *etc* and *raise*
  –click a mouse button on the upper border
  –hit the "front" key on your keyboard
- Cursor position: is displayed (usually, but depends on default setting) in "logical" coordinates) in the text window. To enable/disable select *etc* and *coord*.
- Pixel values: type 't' in the main display window to obtain a list of pixel values around the cursor position. The values appear in the *SAOmonolog* window.

- Close-up display: This display tracks the cursor position in the main display. To enable/disable the track, select *etc* and *track*.
- Color lookup table: to change the contrast of the display hold 'r' mouse button and move cursor in main window.
- Color map: to change the colormap (default is black and white) select *color* and *cmap* which brings up another sub-menu. *"B"* is a good first choice of colors.
- Color lookup display: to display look-up table, click any mouse button on color bar. To adjust red/green/blue individually, hold left/middle/right mouse buttons respectively and move mouse.
- text/error messages: appear in the *SAOmonolog* window.
- -pros/-imtool: This switch controls the type of output from the *region* menu option. It can be changed by selecting *etc* and *new* and editing the command line accordingly.

5.2.3 The SAOimage Menu (options within SAOimage)

When *SAOimage* is initiated, it will appear with the SAO logo which will be replaced by an image when one is (§5.3).

Menu items are chosen by clicking any mouse button (cf. Figure 5.1) on that item. There are several levels available, a level is entered and exited by clicking on the appropriate button. Table1gives a brief description of the function of each menu item. A few, more complex, topics are described in more detail below.

Fine Movement
To move the mouse one pixel at a time, use the arrow keys on the keyboard.

Saving regions
To save/delete regions from the current buffer:

"s"	save current cursor as an include region (counts within region will be included in subsequent calculations)
"e"	save current cursor as an exclude region (counts within region will be excluded from subsequent calculations)
"d"	delete most recent (or closest to mouse) region from save buffer

To save regions in current buffer to a disk file select *region,write*.

Annuli

Annuli around a region are spaced either using the mouse directly (middle/right to add/delete respectively) or, if the mouse is positioned far from the region, equally spaced for each addition/deletion.

Blinking

Three blink buffers are available, one for each mouse button. An image is placed in the left/middle/right buffer by clicking the left/middle/right mouse button on the *blink* menu item. When another image is displayed, the earlier one can be compared with it by pressing its respective mouse button with cursor in the main display window. When the button is released, the current, loaded image will be re-displayed.

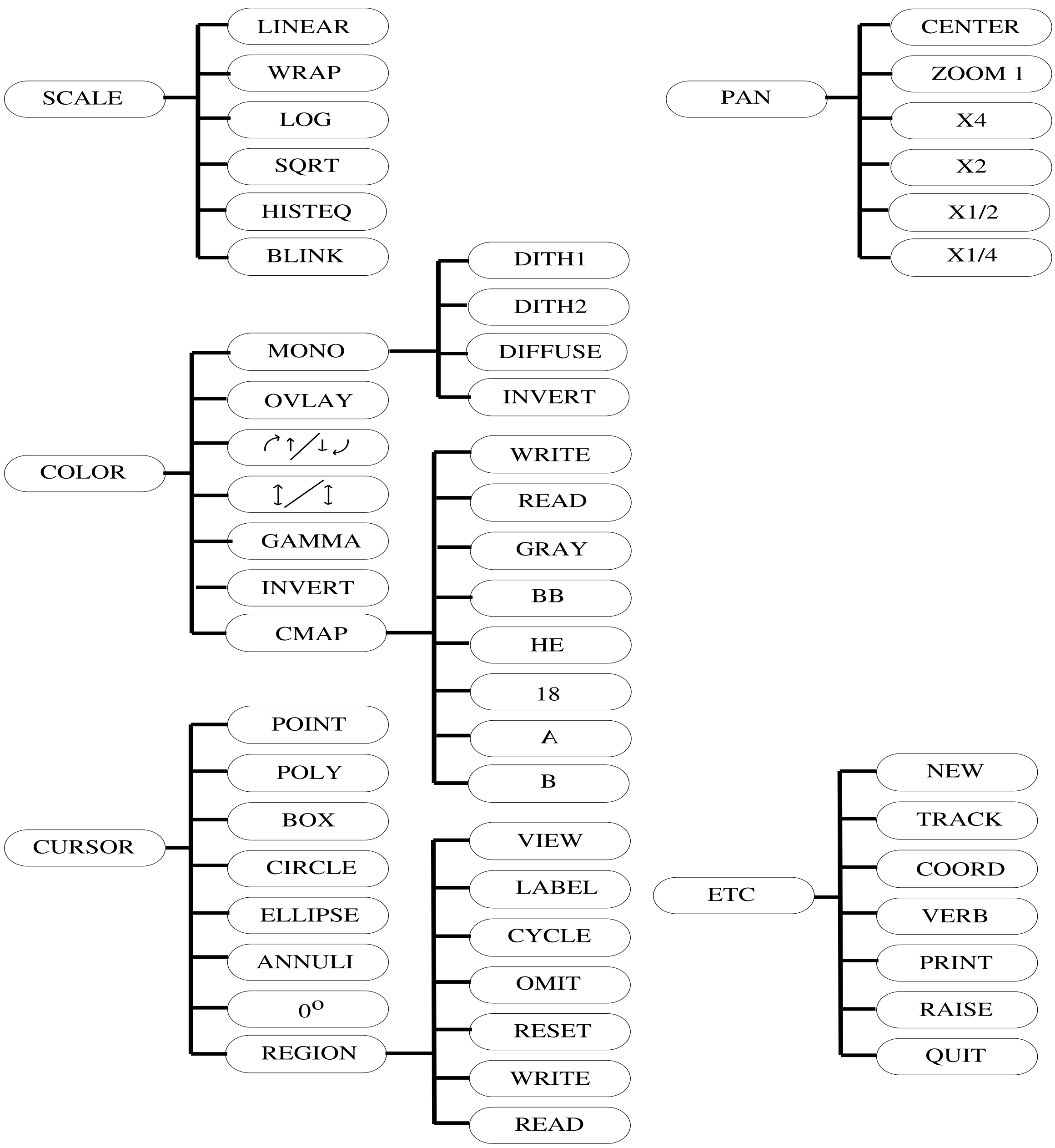

SCALE
LINEAR
WRAP
LOG
SQRT
HISTEQ
BLINK
PAN
CENTER
ZOOM 1
X4
X2
X1/2
X1/4
COLOR
MONO
DITH1
DITH2
DIFFUSE
INVERT
OVLAY
GAMMA
INVERT
CMAP
WRITE
READ
GRAY
BB
HE
18
A
B
CURSOR
POINT
POLY
BOX
CIRCLE
ELLIPSE
ANNULI
0°
REGION
VIEW
LABEL
CYCLE
OMIT
RESET
WRITE
READ
ETC
NEW
TRACK
COORD
VERB
PRINT
RAISE
QUIT

Figure 5.1

Table 1: Detailed list of menu options

Menu item	Sub-menu item	Description
scale	linear	linear scaling
	wrap	wrapped linear scaling
	log	logarithmic scaling
	sqrt	squareroot scaling
	histeq	histogram equalization scaling
		i.e. adjusting histogram of pixel values so that each interesting
		bin contains roughly equal number of pixels
	blink	save image in blink buffer, 3 buffers labelled with
		left/middle/right mouse button
color	mono	enter/leave mono sub-menu
	mono-dith1	apply dither map 1 to display colormap) These are grey-scales
	mono-dith2	apply dither map 2 to display colormap) composed of dots,
	mono-diffuse	apply error diffusion code to colormap) useful for publication.
	mono-invert	invert color map
	ovlay	toggles overlay mode. When "on" half available
		colors are saved for overlays and graphics
	$\nearrow\uparrow$ / $\downarrow\swarrow$	adjust threshold/saturation (horizontal/vertical mouse motion)
	$\updownarrow$ / $\updownarrow$	adjust contrast/bias (horizontal/vertical mouse motion)
	gamma	adjust intensity (horizontal mouse motion)
	invert	invert color map
	cmap	enter/leave colormap submenu
	cmap-write	write current colormap settings to specified disk file
	cmap-read	read current colormap settings from specified disk file
	cmap-gray	all pixels have same color, intensity varies
	cmap-BB	preset color map 1[a]
	cmap-HE	preset color map 2[a]
	cmap-18	preset color map 3, pseudocolor[a]
	cmap-A	preset color map 4[a]
	cmap-B	preset color map 5[a]
cursor[b]	point	point cursor
	poly	polygon cursor
	circle	circular cursor
	ellipse	elliptical cursor

Menu item	Sub-memu item	Description
	box	rectangular box cursor
	0^o	restore cursor to original rotation
	annuli	make annuli around circle/elipse/box cursor
$\leftarrow$	arrow cursor	left button positions, middle button rotates
abc	label (letters etc)	position label with left button, freeze with middle
	region	enter/leave region sub-menu
	region-view	toggles display of saved regions (and labels)
	region-label	toggles display of saved region labels
	region-cycle	cycle through saved regions in magnifier window highlighting each one in main display window
	region-omit	deletes region currently selected by cycle
	region-reset	deletes all labelled points
	region-write	write saved ("s") cursor positions to a specified file
	region-read	read cursor positions from specified file
pan	center	center display at position of mouse click
	zoom 1	return to zoom 1
	X4	zoom by a factor 4
	X2	zoom by a factor 2
	X1/2	expand by a factor 2
	X1/4	expand by a factor 4
etc	new	load new image, filename to be entered
	track	toggle tracking of magnifier window and real-time color graph modification
	coord	toggle cursor coordinate display
	verb	toggle verbose message mode
	print	send a hardcopy to the default printer[c]
	raise	raise window to front
	quit	exit SAOimage

Notes:

a: to view color graph window, click mouse button in color bar. Colors can be adjusted by moving left/middle/right (red/green/blue) mouse buttons on the graph.*

b: these menu items are symbols in this order

c: default printer is set in your .login file with:
SETENV R_DISPOSE "lpr lps -r -s %s" (where lps is the logical name for the printer).

5.2.4 To produce a region mask using *SAOimage* interactively.

SAOimage allows the interactive building of a spatial region that can be fed to other IRAF tasks (*e.g.*, imcnts, see §2.6 , 6.3) to sum up counts etc. Bring up *SAOimage* using (see §5.2) and display (see §5.3) the ROSAT test image xdata$rp110590_iml.imh.

N.B.: If you are working with a blocked or sectioned image, the region will be created in the corresponding coordinates. If required, the original image coordinates can be restored using (§5.2.5)).

5.2.4.1 Select a circle around the central point source

- select *cursor* and *circle*. When in the image window, the mouse now has a box shape with an arrow at the upper left corner. The mouse buttons act as follows:

  LEFT - to move the center of the circle at the position of the arrow point.

  MIDDLE - to size the circle.

  RIGHT - idle.

- center the circle at the point source by positioning the arrow point and clicking on the LEFT mouse button. A circle will appear at the arrow position. A verbose display will also appear in the window from which *SAOimage* was initiated, giving the center coordinates and radius of the circle.
- size the circle by holding the MIDDLE button and dragging the mouse until the desired size is obtained. Releasing it will fix the size of the circle at the value printed in the output window.
- save the results (in memory) by typing 's' while the mouse is in the display window. The center position and the outermost radial distance along the minor axis will be displayed if the button options *cursor, region, view* and *cursor, region, label* are ON.
- select *region* and *write*, to write the saved regions into a file for later use. Another small window will appear, with a filename *saoimage.reg* (default). Either hit carriage return for approval, or delete and type a new name. The file will be written in the directory from which *SAOimage* was initiated, unless specified otherwise in the filename.

The following text will be written to the specified file:

```
# xdata$rp110590_iml.imh - XRT/PSPC PSF AR LAC
# Thu Jul 25 11:20:17 1991
# shape x, y, [x dimension, y dimension], [angle]
 CIRCLE(232,313,10.00)
```

The circle has a radius of 10.00 pixels.

- Select *region* and *read* to display this or other regions by reading them from the specified file.

5.2.4.2 A box, to include extended emission

- select *cursor* and *box*. The three mouse buttons have been redefined as:

  LEFT - moves the center of the box to the position of the arrow point.

  MIDDLE - sizes the box. Horizontal and vertical size can be done independently.

  RIGHT - rotates the box.

- A box will appear centered with the circle. Position the arrow at the desired location and click on the LEFT button.
- size the box by holding the MIDDLE button and dragging the mouse both horizontally and vertically to define the two dimensions of the box. Release the button to fix the size of the box.
- save the results by typing 's' while the mouse is in the display window.
- select *region* and *write*, to write the saved regions into a file for later use (as above).

5.2.4.3 A Region mask for a radial profile in concentric, elliptical annuli

- select *cursor* and *ellipse*. When in the image window, the mouse now has a box shape with an arrow at the upper left corner. The three mouse buttons have been redefined as:

LEFT - moves the center of the ellipse with the position of the arrow point.

MIDDLE - sizes the ellipse.

RIGHT - rotates it.

- position the arrow where the center of the ellipse should be and click on the LEFT button. The ellipse parameters will appear on the window from which *SAOimage* was initiated.

- size the ellipse by holding the MIDDLE button and dragging the mouse both horizontally and vertically until the desired shape (major and minor axis) is achieved.

- rotate it by holding the RIGHT mouse button and dragging the mouse. Release the button at the desired angle.

- select the annulus button. Mouse buttons are now:

  LEFT – move region (as before)

  MIDDLE – add annulus

  RIGHT – delete annulus

  i.e., the axial ratio and orientation of the ellipse are fixed, only adding (or removing) ellipses of different area is possible.

- obtain different annuli by placing the mouse at some distance from the ellipse and clicking on the MIDDLE button to produce equally spaced, concentric ellipses towards the mouse position. This will work both outside and inside the ellipse. To change the spacing between ellipses, click with the MIDDLE button and drag to the desired position. To remove one or more ellipses use the RIGHT button: the ellipse closest to the cursor position will disappear.

- save the annuli, by typing 's'. The center position and the outermost radial distance along the minor axis will be displayed if the options *cursor>region>view* and *cursor>region>label* are ON.

- select *region* and *write*, to write the saved region to a file (as above).

Below is an example of the results obtained with the above procedure.

```
# xdata$rp110590_iml.imh - XRT/PSPC PSF AR LAC
# Thu Jul 25 11:42:57 1991
# shape x, y, [x dimension, y dimension], [angle]
 ELLIPSE(231,314,4.23,17.70,236.864)
 ELLIPSE(231,314,11.72,49.06,236.864) & !ELLIPSE(231,314,4.23,17.70,236.864)
 ELLIPSE(231,314,23.45,98.11,236.864) & !ELLIPSE(231,314,11.72,49.06,236.864)
```

Note: &! means "and not"; i.e., an elliptical annulus rather than a solid ellipse.

5.2.5 *fixsaoreg* — restore SAOimage region to original image (physical) coordinates

In order to view a large image/qp file with the SAOimage display, it is necessary to block or section that image. The SAOimage pixel readout is not in original image/qp coordinates (physical) but in blocked and perhaps sectioned coordinates (logical). In addition, any defined region (cursor) will also be in logical coordinates. To convert an SAOimage cursor file into original (physical) coordinates:

```
xr> xspatial
xs> fixsaoreg saoimage.reg fixsaoimage.reg
```

In the output file, your original region has been commented out and the same region in physical coordinates has been inserted. This region file can then be used as input to a PROS task, e.g., *qpspec, imcnts*, using the original image/qp file.

5.3 * display , xdisplay , xexamine * – Image Display

5.3.1 A small image (512 X 512) using display

To display an image, once *SAOimage* or *Imtool* is running:

```
cfa#: cl           # Initiate IRAF from within one of the windows
cl> images                             # Load the relevant package
```

```
im> xray                        # Necessary if demonstration file used.
               # Not necessary if you use file in current directory
xr> tv
tv> display xdata$snr             # will display central 512x512 of
                                    # a 1024x1024 Einstein image.
frame to be written into (1:4)(1): 1   # Although IRAF recognizes 4
                                 # buffers, SAO image only has one.
```

5.3.2 To compress and display a large (e.g. ROSAT) image

Since a ROSAT PSPC image is 15360×15360, the above command only displays a small part. To see the whole image, compress this by a factor of 30 to 512×512. The factor of 30 compression is called a "blocking factor" (ie. block=30). Blocking factors work only with QPOE (*.qp) files and not with image (*.imh) files.

```
tv> display ''xdata$rp110590.qp[block=30]'' 1
```

Note: the quotes are necessary when this modified file name is used in the command line mode.

5.3.3 *xdisplay* – Image Display with scaling appropriate to x-ray images

The threshold and saturation values are not automatically set well for most x-ray data files since the software samples a few pixels only and x-ray data has few photons. It is possible to set them by hand for an individual image:

```
tv> display ''xdata$rp110590.qp[block=30]'' 1 zscale- zrange- z1=0 z2=30
```

This turns off the default mapping of intensity to color (zscale-) and sets 0 counts/pixel at color-scale minimum and 30 counts/pixel at color-scale maximum.

or try

```
tv> display ''xdata$rp110590.qp [block=30]'' 1 ztrans= ''none''
```

This sets the z value = the number of counts in each pixel.

Another technique is to use the task *xdisplay* which is identical to *display* except that z1 and z2 are scaled appropriately for an x-ray image.

```
xr> xplot                        # load the relevant package
xp> xdisplay ''xdata$rp110590.qp[block=30]''
```

5.3.4 To view a section of the image at full resolution

Display just the section of interest:

```
tv> xdisplay ''xdata$rp110590.qp[6700:7200,9100:9600]''
                                         # range of x coord is 6700-7200,
                                         # range of y coord is 9100-9600.
```

5.3.5 To both compress and view a particular section

```
tv> display ''xdata$rp110590.qp[block=4][1673:1800,2273:2400]''
```

This is a tricky operation. The order of the instructions is critical. The image must first be compressed, then the section specified using the coordinates of the compressed image (logical coordinates). The above will display a 128×128 section.

5.3.6 To display a particular energy band

```
tv> xdisplay ''xdata$rp110590.qp[block=30, pi=7:40]''  # will display only data
                                                       # from pi channels 7-40.
```

PSPC PI channels available are 1:256, use of 7:240 only is recommended (see §6.5.3).

5.3.7 To display a particular time interval

```
tv> xdisplay ''xdata$rp110590.qp[block=30,time=(1611600:1612600)]''
```

N.B. Use *timfilter*(see §8.4) to obtain available times in the image

5.3.8 To display other sizes

The display defaults to a size of 512×512 pixels. The size of the display can be changed to match the image. Available sizes are:

```
1  2  512  512          OA# imt1|imt512
2  2  800  800          # imt2|imt800
3  2 1024 1024          # imt3|imt1024
4  1 1600 1600          # imt4|imt1600
5  1 2048 2048          # imt5|imt2048
6  1 4096 4096          # imt6|imt4096
7  1 4096 1024          # imt7|imt4x1
8  1 1024 4096          # imt8|imt1x4
9  2 1142  880          # imt9|imtfs   full screen (1152x900 minus frame)
```

To change the size:

```
tv> set stdimage=''imt3''   # sets the size of the screen buffer to 1024x1024
tv> xdisplay ''xdata$rp110590.qp[bl=15]''        # reduces the 15,360 square
                                                # image to a 1024x1024 array.
                                  # The central 512x512 section is displayed.
```

The full image is loaded into the buffer and a part of the image displayed depending on the physical size of the windows. The *par* and *center* menu option can be used to center on different sections of the image in the buffer or the SAOimage window can be physically resized to display the full 1024×1024.

5.3.9 * display , xdisplay * — Displaying an image in detector coordinates

The following command will display a *.qp* file in the original detector coordinates. In this case a ROSAT PSPC image is used so it is also blocked to a reasonable size:

```
xp> xdisplay ''xdata$rp110590.qp[block=30, key=(detx, dety)]''
```

Alternatively, higher resolution can be obtained with the following command which displays the bottom left hand quadrant of a1024 × 1024 array:

```
xp> xdisplay ''xdata$rp110590.qp[block=15, key=(detx, dety)] [1:512,1:512]''
```

5.3.10 * xexamine * — examine a qp file

If you have a large (spatial dimensions) qpoe file, the most convenient way to display it is using *xexamine* . This task works on the full-resolution data file, blocking as appropriate for the display window. The user can zoom, pan and project interactively using the menu items listed; the full-resolution data file is accessed each time:

NOTE: Running *xexamine* disables commands a user has available in their TV window. After 'quiting' *xexamine* the TV window commands are again enabled. (i.e. *SAOimage* keyboard commands 's', 'e', etc.).

```
xp> xexamine xdata$rp110590.qp
```

The full qpoe file is displayed, blocked appropriately and a single keystroke menu appears.

<u>Source detection</u>

The 's' option determines the maximum likelihood position of the source at the cursor position, tabulating RA, dec, original pixel coordinates, cellcounts etc.. Note that the default cell size is 12 arcsec, appropriate for an HRI image. This can be changed by typing:

```
:c 30                  # 30 arcsec cellsize, appropriate for PSPC/IPC
```

The user should refer to the BEPOS documentation for more information on detect.

5.4 Celestial Coordinates from Image Coordinates

There are several alternative methods for obtaining coordinates from a displayed image. If celestial coordinates are required, the simplest method is to use *tvlabel* .

5.4.1 * tvlabel , *xexamine* * — sky coordinates from a displayed image

This will calculate coordinates from header information (therefore epoch 1950 for *Einstein*, epoch 2000 for ROSAT) and will present coordinates in hms,dms.

```
st> xray
xr> xplot
xp> tvlabel ''xdata$rp110590n00.qp[block=10]''
```

d for coordinates at cursor position.

Type <ctrl>d in the display window to quit.

or

```
xp> xexamine ''xdata$rp110590n00.qp''
  c  for sky coordinates at cursor position
  q  to quit
```

xexamine has many additional options, "?" lists them (see § 5.3.10).

5.4.2 *skypix* — conversion of pixel position to sky coordinates

To convert from pixels to RA and Dec. first view image with *SAOimage*. Set cursor on object of interest. Read pixel coord from *SAOimage* (coord must be on). Run *skypix* in separate window. Type pixel coords in skypix.

```
xr> xspatial
xs> skypix.oformat = ''hours''  # to reset the default which is ''degrees''.
xs> skypix

input coordinate system (J2000): xdata$rh110267_img    # the image file name
output coordinate system (J2000): image            # to use coord conversion
                                                      # in image file header
or B1950 or J2000                             # epoch of output can be specified
input file list (STDIN): <cr>
```

...conversion explained, note epoch of output...

```
145  276<cr>                                        # pixels read from SAOimage
      22:08:41.6  45:44:38.0                                     # RA and dec.
256  265<cr>
      22:07:16.8  45:42:00.0
ctrl d                                                               # to quit
```

5.4.3 *wcscoord* — all purpose coordinate conversion

This task converts between any pair of the available coordinates:

logical raw pixel coordinates of current image or image section

physical pixel coordinates of original image from which current image was derived ($\equiv$logical if no blocking or sectioning have been performed)

world sky coordinates

tv pixel coordinates on the tv (*SAOimage*) display. These will $\neq$ logical if an image section is specified as input to the display task.

5.4.4 Image with sky grid overlaid

To produce a high-quality black-and-white figure of a source with a coordinate grid overlaid, use the following recipe:

1. Open an *SAOimage* window, display an image, click on the color/invert button, and adjust the colors to get a black-and-white image.

2. Set the following display parameters:

   display.xcenter=0.55

   display.ycenter=0.51

3. Display a 430×430 pixel section of the image (blocked or unblocked). (This allows room in the *SAOimage* display for coordinate labels outside of the figure. Labels can also be inside, in which case step 2 is unnecessary and the full 512×512 pixels can be displayed.)

   display "rp110590n00.qp[bl=15][249:679,409:839]" 1

4. Run *wlpars*, which will invoke *eparam* on one of the *wcslab* parameter files, and change "major_g" to no and "title" to blank (hit space then return). (The title option here may be used, but it requires that a smaller region be displayed. A different labeling option is used in step 7.) Packages "images" and "tv" must be loaded to run *wlpars* and *wcslab*.

5. Set the *wcslab* parameter "device" to imdB (for a black overlay). (Other colors are available, see below.)

   wcslab.device="imdB"

6. Run *wcslab* (overlays a world coordiante system).

   wcslab "rp110590n00.qp[bl=15][249:670,409:839]" 1

7. Make a hardcopy (from outside IRAF) using facilities available on your machine, e.g., screendump.

   AT SAO: xprint -saoskip -nobdrs -portrait -P lp -label "Fig. 986A" -font 24

5.5 *tvimcontour* – Overlay contours and/or source positions on the display

This task is a convenient amalgamation of *xdisplay* and *imcontour* which overlays contours and, if requested, source positions in sky coordinates onto the SAOimage display. As with *imcontour* the contouring works best on a smoothed (see §6.4) image.

input file: *rp90_ab_smo.imh*[2]

```
xp> xplot
xp> tvimcontour
IRAF image file:  rp90_ab_smo
Grid type (sky/pix):  sky
Contour units (pixel/peak/sigma): peak
Contour levels:  log 5 100 5
xp>
```

A convenient method for overlaying source positions on an image where you are not interested in the contours and/or an unsmoothed image is as follows:

```
xp> tvimcontour src_lis=xdata$rp110590_src.tab
IRAF image file:  xdata$rp110590_im1
Grid type (sky/pix):  sky
Contour units (pixel/peak/sigma): peak
Contour levels:  log 100 100 1              # single contour at the peak.
```

imcontour (See also §6.4) can be used to overlay contours on an existing display with proper setting of the parameters: scale_device and graph_device and perim. Type *help imcontour* for more information.

[2]See §6.4 for generation of this smoothed image.

5.6 *tvmark* – Mark pixel positions on the display

This can be used to overlay the position of fiducial stars onto an image, for example. Positions can be input manually or from a list.

To position marks interactively display the image and then

```
cl> noao
no> images
im> tv
tv> tvmark 1 STDIN interactive+

type . to put dot at cursor position
type + to put cross at cursor position
type c to put circle at cursor position (must specify hidden parameter ''radii'')
<ctrl>d to exit
```

See help file and parameters for other options.

To display marks from a list, "starpos", the list should have the format of a double column with positions in logical coordinates (= same as *saoimage* readout).

```
pr> tvmark 1 starpos interac-
```

5.7 *tvproj* — one-dimensional projection on display

This task combines *display* and *improj* to superpose x,y projections of the input image on the display of that image:

```
xp> tvproj xdata$rp110590_im1
```

Note that you will need to be in a directory in which you have write permission.

5.8 *tabplot* – Plotting tables, lists or image sections

tabplot is a convenient interface to the stsdas task sgraph which allows the user to generate line graphics of table columns, image sections or lists. It allows manipulation of point styles, inclusion of error bars, titles etc. via the *pltpar* parameter set. The help files for tabplot and sgraph provide more details.

e.g., To make a plot of a source spectrum as a function of energy including error bars on the counts:

```
xp> tabplot ``xdata$rp110590_sp4 lo_energy net''errcol=``neterr'' point-errax=2
               # where lo_energy, net, neterr are the column names in the table
```

5.9 *pspc_hrcolor* – generate an x-ray color image

This task generates a color map based on the x-ray color information in a PSPC image by dividing the input *.qp file into three color bands.

```
xp> pspc_hrcolor xdata$rp110590.qp 30 '' '' rp90
```

This creates a colormap rp90_color.map which can be accessed once the blocked image has been displayed using the *color_cmap_read* option of *SAOimage*:

```
xp> xdisp ``xdata$rp110590.qp[bl=30]''
```

N.B.: It is generally necessary to enter the full address of the colormap.

6 Spatial Analysis

This chapter deals with analyzing source positions and spatial structure and with image calibration. Depending on the specific scientific task, the user may wish to block or section the input data file for convenience (See also §5.3). See Table 6.1 for the dimensions of ROSAT and Einstein images:

Table 6.1 ROSAT and Einstein Field Dimensions

Inst't	Mission	Pixel Size	Dimension	Field of View	Block to reach 512*512
PSPC	ROSAT	$0.5''$	15360^2	2°	30
HRI	ROSAT	$0.5''$	8192^2	$40'$	16
IPC	Einstein	$8''$	1024^2	1°	2
HRI	Einstein	$0.5''$	4096^2	$40'$	8

6.1 Summary of Tasks in this Section

Task	Function	Package
detect	a suite of tools for source detection	xray:xspatial:detect
errcreate	create an array of errors	xray:xpatial
imarith	addition etc. of images	images
imcalc	arithmetic operations of arrays	xray:ximages
imcnts	extracts counts from region	xray:xspatial
imcontour	makes contours	xray:xplot
imcopy	copies arrays with format changes if desired	images
imexamine	photometry of sources	noao:proto
imlintran	translation, rotation, and magnification	images
immodel	build analytic functions	xray:xspatial
imsmooth	smooths arrays	xray:xspatial
isoreg	makes mask based on image intensity levels	xray:xspatial
magnify	magnification of images	images
makevig	generate a vignetting mask	xray:xspatial
plcreate	creates binary file describing region	xray:ximages
pllist	display pixel mask	xray:ximages
qpshift	perform a linear shift on a qp file	xray:ximages
qpsim	generate simulated HRI qp file	xray:xpatial
qplintran	perform a linear transformation on a qp file	xray:ximages
rosprf	build ROSAT point response functions	xray:xspatial
sgraph	a plotting task	tables
skypix	converts pixels to RA and dec	xray:xspatial
srcinten	determine HRI count rate	xray:xspatial
tprint	lists tabular information	tables
vigdata	correct an image for vignetting	xray:xspatial

6.2 Regions and Source Parameters

This section will deal with obtaining photons within specified regions of an image. Available regions are:

```
Annulus
```

```
Box
Circle
Ellipse
Field -- entire field
Irregular region based on intensity of map...
          i.e. region defined by contour levels.
Pie -- angular wedge
Point
Polygon
Rotbox -- rotated box
```

for more details type:

```
cl> help regions
```

The *imcnts* task in the *xray-spatial* package sums the source and background counts in specified regions, and subtracts one from the other properly normalized. The results are saved in a table file, which can be read, printed or converted into an ascii file using the *ttools* (see §6.2.7).

The sample data file, *xdata$rp110590.qp*, contains a point source whose position is at x = 6930, y = 9390. The counts will be taken within a circle of 240 pixel radius ($2'$ in the ROSAT PSPC. This is the recommended circle size for a PSPC point source which is not too far off axis). The examples requiring an extended source will use the data file, *xdata$snr.qp*, containing *Einstein* IPC data. Counts will be obtained in a box of 208 $\times$ 266 pixels, with exclusion of the point source at 503,513.

6.2.1 *plcreate* — Defining a Region

There are three alternative ways to define a region. The optimum choice depends on the scientific application.

1. input the region descriptor directly as a parameter to the PROS task in question e.g. *imcnts*

2. write a binary file using the task *plcreate* . This is faster and more convenient if the same region is to be applied repeatedly.

input file: *xdata$rp110590.qp*

output file: *rp110590.pl*

```
cl> xray
xr> ximages
xi> plcreate
region description: c 6930 9390 240
reference image file or 2 dims for mask:  xdata$rp110590.qp
output mask data file [root.pl]: rp110590            # =rp110590.pl file
```

3. via *saoimage* as described in §5.2.4. The region is written by *saoimage* to an ascii text file, default name *saoimage.reg* and can be directly input to a PROS task. To restore the coordinates to original image coordinates see §5.2.5.

Although the ascii output from *saoimage.reg* may be input directly to PROS tasks as a region descriptor, program execution is faster using the binary output of *plcreate* . This task may also be used to translate *saoimage.reg* into binary format:

input file: *saoimage.reg; xdata$rp110590.qp*

output file: *rp110590.pl*

```
xi> plcreate
region descriptor:  saoimage              # default output file from SAOimage
reference image file or 2 dims for mask:  xdata$rp110590.qp
output mask data file [root.pl]: rp110590                  # =rp110590.pl file
```

Region masks are specific to the reference file that was used to create them, and only work correctly for that file. They can be created using any of the coordinate systems available.

The following sections will include examples of the all three of these methods.

6.2.2 *pllist* – Look at the region

The region in a **.pl* file can be inspected:

```
xi> pllist rp110590
```

6.2.3 *isoreg* – A region defined by an isointensity contour

A region file can be made which defines the region enclosed within a specific intensity level contour. First the image must be smoothed:

```
cl> xray
xr> xspatial
xs> imsmooth ''xdata$snr.qp(400:600,400:600)'' snr gaus 5  # output is snr_smo.imh
xs> isoreg xnr_smo snr 0.2                                  # output is snr_iso.pl
xs> pllist snr_iso.pl                                            # shows the mask.
```

6.2.4 *imcnts* – Find background subtracted counts for a point source

input file: *xdata$rp110590.qp*

output file: *rp90_cnt.tab*

```
xi> xspatial                                          # Load the spatial package
xs> imcnts                                       # Run the task to count photons
source image file:  xdata$rp110590.qp      # Input file, use complete filename
source region descriptor:  c 6930 9390 240               # Circle at 6930 9390
                                                      # with radius of 240 pixels
background image file or cts/pixel: xdata$rp110590.qp      # From the same file
bkgd region descripton: a 6930 9390 300 1500
                                           # Annulus  of 300 to 1500 pixel radii,
                                          # centered at the point source position
root name for output file [root_cnt.tab]: rp90                     # output table
                                                                  # rp90_cnt.tab
```

6.2.5 *imcnts* – Find counts in an extended region, after removing an embedded point source

input file: *xdata$snr.qp*

output file: *snr_cnt.tab*

```
xr> xspatial
xs> imcnts
source image file:  xdata$snr.qp              # Again for the extended emission
source region descriptor:  box 459 510 208 266 &! c 503 513 22.5
                                # Box centered at 459, 510, of dimensions 208x266,
                     # with the point source circle excluded, &! means   'and not'
background image file or cts/pixel: 0                               # No background
root name for output file [root_cnt.tab]:  snr       # Output table snr_cnt.tab
```

N.B.: The use of *imcnts* to obtain counts in regions can lead to wrong results when more than one region is used in the same run, if there is any overlap between them, since each pixel can be assigned to ONE AND ONLY ONE region (the last one defined that includes that pixel. See "help regions").

6.2.6 *imexamine* – General analysis of image containing many sources

This task (developed for use with optical images, but also useful with x-ray data) will quickly compute location, size, ellipticity, and position angle of sources. Input position is from cursor that user places on desired source in display. Local background is subtracted and flux and magnitude are calculated. User can generate contour plots of any portion of the image and count rates in slices through the image.

The program is advertised as "quick [optical] photometry".

input file: *rp110590_im1.imh*

output file: *logfile*

```
cl> xray                          # needed for access to xdata$ directory
```

```
xr> noao
no> tv
```

6.2.6.1 Photometry

```
pr> rimexam # This puts user in an editor so parameters can be set to desired
 # values e.g.; for source photometry: radius = 10, buffer = 0, bkgd width =2
```

Set parameters by moving cursor to appropriate row.

```
pr> ctrl x                                        # to record changes and leave editor
pr> imexame xdata$rp110590_im1 keep+ log=logfile           # logfile necessary
                                                          # for permanent record.
                                                   # Results also displayed on monitor.
```

Set cursor on first source of interest and press “a” for photometric analysis. Repeat for other sources.

Press “q” to quit and to generate log file.

```
pr> lprint logfile                                                    # for hardcopy
```

6.2.6.2 Contour Plot To generate an 8-level contour plot of a 50×50 pixel region centered on the cursor.

```
pr> eimexam                                # put parameter file in editor, set
                                     # ncolumn = 50, nlines = 50, ncontour = 8
```

Set parameters by moving cursor to appropriate row.

```
ctrl x                                                      # to leave editor
pr > imexamine.graphics = ``stdgraph''                # for display on screen
```

or

```
pr> imexamine.graphics = ``stdplot''                          # for hard copy

pr> imexamine xdata$rp110590_im1
```

Set cursor at point desired for center of plot and press e.

Press q to quit

```
pr>  gflush                    # for hard copy if parameter was set as stdplot
```

6.2.6.3 Other Useful Functions Some other useful functions which also give graphical results are:

```
r  gives radial plot of surface brightness.
c  (column) gives brightness in a vertical slice through the cursor.
l  (line) gives brightness in a horizontal slice through the cursor.
```

6.2.7 *tprint* – Reading a Table File Created by *imcnts*

Table files like the ones created in the section on §6.2.4 can be displayed, printed and manipulated using the tasks contained in the *stsdas.ttools* package.

Tables are composed of a header section, followed by columns for storing data. Header and table information can be accessed (read, printed, copied) or modified through editing. Table columns can also be plotted or used in arithmetic calculations or statistics.

To read and print the tables obtained in the previous section:

```
sp> tables                                          # Load the relevant package
ta> tprint rp90_cnt prpar=yes                          # .tab extension assumed
                     # the hidden parameter prpar is set to display the header
                                                            # as well as the data

ta> tprint rp90_cnt                                 # Displays only the data info
ta> tprint rp90_cnt prdata- prpar+                # Displays only the header info
ta> tprint rp90_cnt prparam=yes | page            # To read both header and data,
                      # one page at a time. Note the alternate form of prpar+.
ta> tprint rp90_cnt prparam=yes | lprint                # Sends output to printer
ta> tprint rp90_cnt prpar+ > file.tab              # To write to a file, file.tab
```

6.2.8 *imcnts* – Use of the Sky Coordinate System

The following example reproduces that in §6.2.4, but using celestial coordinates (see §2.6.1), and using different size source and background regions.

input file: *xdata$rp110590.qp*

output file: *rps90_cnt.tab*

```
xr> xspatial                                         # Load the spatial package
xs> imcnts                                       # Run the task to count photons
source image file :  xdata$rp110590.qp                                # Input file
```

```
  source region descriptor :  c 22:08:41.6 45:45:04.7 3'
                               # Circle at RA=22h 08m 41.6s, Dec 45 deg 45' 04.7''
                                                          # with radius of 3 arcmin
  background image file or cts/pixel: xdata$rp110590.qp      # From the same file
  bkgd region descriptor:   a 22:08:41.6 45:45:04.7 3.3' 4.4'            # Annulus
                     # of 3.3 to 4.4 arcmin radii, centered at the source position
  root name for output file [root_cnt.tab]:  rps90    # Output table rps90_cnt.tab
```

It is also possible to combine pixel and sky coordinates. For example *c 232 313 2′* is equivalent to *c 232 313 240* for the ROSAT PSPC (0.5″ per pixel).

6.2.9 *srcinten* – Calculate Source Intensity

For ROSAT HRI data, this task allows computation of the count rate from a specified source region. The result is corrected for the point response function (using the accurate, off-axis PRF), vignetting and quantum efficiency effects. Note that the size of a region cannot be specified in celestial coordinates (e.g., arcminutes) for this task (1 pixel = 0.5″ for the ROSAT HRI).

```
xs> srcint
source qpoe file ( ):   xdata$rh110267.qp
source region descriptor ( ): c 2326 4406 120
background qpoe file ( ):
bkgd region descriptor ( ):    a 2326 4406 150 200
output root name (_ccr.tab):  (arlac)

 --- Creating aspect histogram table from qpoe ---

 --- Computing vignetting correction factor ---

 --- Computing prf correction factor ---

 --- Computing qe correction factor ---
```

```
--- SRCINTEN created Output table: arlac_ccr.tab ---
```

6.3 Radial Profiles and Projections

This section deals with obtaining the distribution of counts for a specific source. For detailed analysis with reference to background and exposure maps, vignetting corrections, etc., see §6.5.

6.3.1 *imcnts* – Radial Profile of an extended source at the field center

Obtain a radial distribution of counts for the source centered at x = 503, y = 513, in the data image *xdata$snr.qp*. The background is taken locally from an annulus outside the source region. Errors are evaluated from the counts in the image and include the error on the background counts.

input file: *xdata$snr.qp*

output file: *snr_cnt.tab*

```
cl> xray                     # Load the relevant packages (unless already loaded)
xr> xspatial
xs> imcnts                                                          # Run the task
source image file :  xdata$snr                             # xdata$snr.qp; make sure
          # there is no xdata$snr.imh file, or give the *.qp extension explicitly

source region descriptor:  a 503 513 0 50 n=5[3]

               # 5 annuli at 503,513, 0 to 50 pixel min/max radii, 10 pixel width
background image file or cts/pixel :   ''''                    # blank filename will
                                                 # force the task to use the same filename
bkgd region descriptor:  a 503 513 100 150          # Annulus centered at 503,513,
```

[3]This form specifies that the region between 0,50 pixels should be divided into 5 equal-width annuli. It is equivalent to: *a 503 513 0 10 20 30 40 50.*

```
                                              # 100 to 150 pixel inner/outer radii
  root name for output file [root\_cnt.tab]: snr                     # snr_cnt.tab
```

Region descriptors can also be given on the command line, but they must be enclosed by quotes.

```
  xs> imcnts xdata$snr ''a 503 513 0 50 n=5''
```

Either commas or spaces can be used to separate the parameters needed for a region.

The output tables can be read/printed with the *stsdas-ttools* (cf. §6.2.7) package tasks

N.B.: The region specifications are identified with letters in the table headers, and the resulting regions are identified with numbers in the data section. There might not be a one to one correspondence between specifications in the header and regions in the data section. For example, when annuli are specified, as in the example above, there is only one specification in the header, but there are 5 regions in the table. For some regions (annuli and pie slices), the beginning and ending radii or angles are entered in the table.

Note also that the number of region specifications in the header is limited by the number of ASCII characters available. If the number of specifications is larger, the table will contain the total number of rows and the correct results, but the header information will be unreliable.

6.3.2 *tabplot* – Displaying a Radial Profile

input file: *snr_cnt*

output file:

To plot the radial profile generated in the previous section:

```
    xr> xplot
    xp> tabplot ''snr_cnt rad1 cnt/pix''                      # sgraph must be given
                    # these 3 parameters (input table, label of column giving x,
                  # label of column giving y) as a group.  If in interactive mode,
```

```
                                             # the 3 parameters must also be given as a group.
                                                           # the second parameter is rad(one).
```

The plot will appear and will be autoscaled appropriately.

To make hardcopy type,

xp> =gcur

Put cursor within plot boundary and type = .

Type q to return to text window.

To add error bars to above graph, the error bars must be generated then overlaid on existing graph.

```
xp> pltpar                                    # puts plot parameter file in editor.
move cursor to ''erraxis'', type 2
move cursor to ''errcolu'', type err/pix
type ctrl x  to exit editor (for emacs, ctrl d  for vi)

xp> tabplot ''snr_cnt rad1 cnt/pix'' append=yes    # overlays data and error bars
```

N.B. Remember that this has permanantly changed the plot parameter file and the parameters must be reset for subsequent plots.

6.3.3 *imcnts* – Azimuthal profile of an extended source

An azimuthal profile is determined as follows. Input a spatially blocked and energy filtered image from above and use with the background determined from the broad-band background map.

```
cl> xray
xr> xspatial
xs> imcnts addbkg=no                              # Do not add the background error
```

```
Source image file:  xdata$snr.qp
Source region descripter:  c 503 513 50 & p 503 513 0 360 n=10
                                   # circular region centered on point source with
                           # 25 pixel (block=2) radius divided into 10 pie slices.
background image file or cts/pixel:  xdata$snr.qp
                                                      # broad band background map
bkgd region descriptor:  a 503 513 50 100                 # annulus around source
root name for output file [root_cnt.tab]:  azim
                                                # output table i.e.  azim_cnt.tab
```

The profile can then be inspected using *stsdas-stplot-sgraph* - see §6.3.2. To look at the table (necessary to determine column headings for plotting) use *stsdas-ttools-tprint* (see §6.2.7).

6.3.4 *imcnts* – Radial Profile in a Quadrant

Obtain a radial profile for a quadrant of the central point source:

input file: *xdata$snr*

output file: *snr_cnt.tab*

```
xs>  imcnts
source image file:  xdata$snr[pi=2:10]
source region descriptor:  pie 503 513 0 90 & a 503 513 0 50 n=5     # see below
background image file or cts/pixel:  xdata$snr.qp [pi=2:10]
background region descriptor:  a 503 513 100 150
rootname for output file [root_cnt.tab]:  snr

xs>
```

The source region descriptor defines a series of annuli within a 90° pie slice from 0° (+ y axis) to 90° (counter-clockwise) to generate a radial profile (see §6.3.2, for display).

6.3.5 *improj* – Make a one-dimensional projection of data within a rectangular band traversing an image

This makes a one-dimensional array (or table) giving intensity vs. position. We illustrate with a horizontal band of finite width such as might be used to examine the surface brightness profile of a source or to extract the dispersed spectrum produced by an objective grating.

input file: *xdata$rp110590_im1.imh*

output file: *rp110590_prj.tab*

```
xr> xspatial
xs> improj

image file : xdata$rp110590_im1.imh
region descriptor : b 231 313 100 10           # a box centered at 231,313 with
                                            # x dimension=100, y dimension = 10
output file .                                                # uses same root
bins in x projection  100
bins in y projection  1
```

The result is a table, rp110590_iml_prj.tab, containing the summed (or average) value of y as a function of x. This can be inspected using *sgraph* (which is in *tables*) or *tabplot* .

```
xs> xplot
xp> tabplot ``rp110590_im1_prj counts_x''
```

The task *improj* will use a rotated box. If the region descriptor is *r 231 313 100 10 45* the above box will be rotated 45° counter-clockwise. If only a graph of a horizontal or vertical projection is wanted, the task *tabplot* alone will do this.

```
xp> tabplot ``xdata$rp110590_im1[*, 310:319]'' axis=1  # plots sum of y=310--319
                                                              # as f(x)
```

6.4 Smoothing and Contouring

6.4.1 *imsmooth* – Smooth an image

Smooth the image with a gaussian function of $\sigma = 2$ pixels. A small (200×200) portion of the field will be used (smoothing takes longer on large arrays).

input file: *rp90_ab.imh*

output file: *rp90_ab_smo.imh*

```
xr> xspatial                                              # Load the spatial package
xs> imsmooth                                                      # Smooth the image
name of input image (dev$pix) :  rp90_ab         # rp90_ab.imh from Section 6.7.1
root name for output file [root_smo.imh] : rp90_ab              # =rp90_ab_smo.imh[4]

      resultant output image dimensions: 200 x 200
      internal image dimensions for fft: 256 x 256

type of convolving function (gauss): <cr>
function radius, sigma or width: 2                                  # In pixel units
```

6.4.2 Calculation of errors in smoothed arrays

Errors may be traced during the smoothing process. An error array (see §6.5.4) must first be generated. This example takes errors from the same region of the original QPOE file.

```
xs> errcreate ''xdata$rp110590.qp[6830:7029,9290:9489]'' rp90ab_err.imh
                                                # The error file is rp90ab_err.imh
     # The '''' on the command line prevent IRAF's confusion with a long argument
```

One then sets the hidden parameters in *imsmooth* .

[4]The output file is written to your *imdir* directory. If this directory does not exist, the program will crash at the end.

```
errors = yes
errarray = (filename of unsmoothed error array)
error_out = (filename of output error array)
```

6.4.3 *imcontour* – Iso-intensity contour map

The contours can be displayed in sky or pixel coordinates. The scale defaults to include the whole field, and is in arcsec/mm (sky grid) or pixel/mm (pixel grid). The user can change the plot scale by manually resetting the (hidden) parameter, "scale".

input file: *rp90_ab_smo.imh*

```
xr> xplot
xp> imcontour
IRAF image filename: rp90_ab_smo
Grid type  (sky|pixel) : sky      # sky coordinates will be read from the header
Units (pixel|peak|sigma):  pixel                # pixel countour levels in counts
                                                                # per image pixel
                                 # peak --> contour levels as % of peak intensity
                  # sigma --> contour levels as S/N ratio (with error array only)
Contour Levels: 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
```

The result will appear in the graphics window (see Figure 6.1). If you are using Xwindows, you are now in cursor mode:

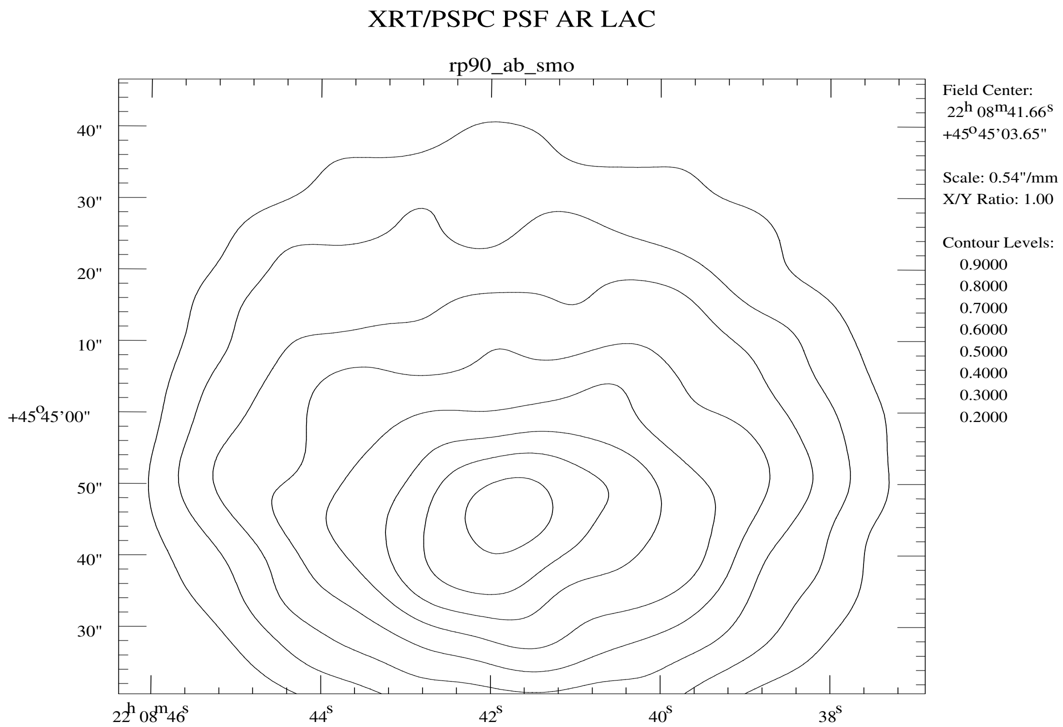


Figure 6.1: Output of the tasks "imsmooth" & "imcontour". this is an off-center point source and the assymmetry is due to coma caused by the optics. This is exactly the example used in Section 6.4.

```
=  sends a snapshot of the current graph to the printer.
q  exits from graphic mode.

xp> =gcur   to re-enter cursor mode
```

Alternatively contours can be specified logarithmically and using a range:

```
xs> imcontour
```

```
IRAF image filename: rp90_ab_smo
Grid type:  sky
Units (pixel|peak|sigma): peak
Contour Levels:  log 5 90 8          # 9 logarithmic contours in the interval
                                        # from 5% to 90% of the peak value
```

To send a contour plot directly to the printer, set the plot output device.

```
xs> imcontour.device=stdplot                          # Directly to the printer
                                              # nothing will appear on the screen
```

then run imcontour as above examples. If hardcopy does not appear type:

```
xs> gflush
```

N.B.: In imcontour, leaving the parameter *isystem* blank will read and use the coordinate system from the header. Problems will arise if *isystem* is given as one coordinate system and the .imh header is another coordinate system.

6.5 Calibration

In order to perform detailed analysis on an x-ray field, a user will need to access the various calibration files, apply corrections for vignetting and propagate errors.

6.5.1 Vignetting correction

Vignetting is a decrease in the effective area of the telescope mirror with increasing off-axis angle. It is due to a combination of a real decrease in the mirror area and a degradation of the mirror reflection efficiency at larger angles of reflection. It becomes significant for $> 12'$ (Einstein IPC) or $> 6'$ (ROSAT PSPC). Both the ROSAT and Einstein exposure maps (§6.5.2) are corrected for vignetting.

6.5.1.1 * makevig , vigdata * – Apply Vignetting Correction to an image

First generate a vignetting mask appropriate for an energy of 1.0 keV (∼central energy in the ROSAT PSPC) for the image *xdata$rp110590.qp*:

task: *makevig*

input file: *xdata$snr.qp*

output file *snr_vig.pl*

```
xr> xspatial
xs> makevig "xdata$rp110590.qp[bl=30]" arlac " " 1.              # generates
                                        # output mask arlac_vig.pl at energy 1.0 keV
```

Now apply vignetting correction to the image generating a vignetting corrected output image and error array:

task: *vigdata*

input files: *xdata$snr.qp, snr_vig.pl*

output files: *vigsnr.imh, vigsnr_err.imh*

```
xs> ximage                                                          # load package
xi> vigdata                                                # vignetting correction
Input image file :  xdata$110590.qp[bl=30]           # From the original QPOE file
Output vignetted image file : arlac                       # give *.imh extension!
Vignetting corrections mask : arlac_vig.pl              # All extensions explicit
fraction of image which is non-vign bkgd (x.xx format): 0.05
                              # 5% of the image counts are non-vignetted background
                                                       #  and will not be corrected
                                                          # The format is important
Input error image file : xdata$rp110590.qp[bl=30]           # The error is taken as
                                             # square root of n from the original data
Output vignetted error file : <cr>                                 # =vigsnr_err.imh
```

This method of applying vignetting assumes that background counts dominate the image. In the ROSAT PSPC the non-vignetted (charged particle) background is a minor contribution to the background with a typical rate of 4×10^{-6} cts s^{-1}arcmin^{-2}keV^{-1} (see Plucinsky et al., Ap.J., **418**, 519, 1993 for a detailed discussion).

6.5.2 Exposure Map

Both ROSAT PSPC and Einstein IPC have regions of their fields masked by the window support structure. In addition, vignetting results in a significantly lower effective exposure time toward the edge of the field.

An exposure map is provided for each ROSAT field (e.g., rp110590_mex.imh).

The simplest method of applying an exposure correction is to divide the data by the exposure map yielding a count rate image. The image must first be blocked to the same size as the exposure map, e.g., for ROSAT PSPC.

```
xi> imcalc
input file ('-' for STDIN) or imcalc commands () : -          # interactive mode
IMC > ctrate.imh = ''xdata$rp110590.qp[bl=30]'' / ''xdata$rp110590_mex.imh''
IMC > quit
```

Alternatively, when using imcnts to extract the counts in a region, the same region can be applied to the exposure map to determine an average exposure time.

Note: The ROSAT exposure map has $60'' \times 60''$ pixels *i.e.*, a 128×128 image. See §6.5.5 for an example of re-gridding this map to match the image data files.

6.5.3 Background Corrections

Background maps are provided by the standard processing for all ROSAT and Einstein instruments. For the PSPC and IPC these are in three standard bands: soft, hard, and broad (Table 6.2).

Table 6.2 Standard Bands

				After Dec 92	
Instrument	Band	Energy	PI Bins	Energy	PI Bins
ROSAT PSPC	Soft	0.1-0.4	7-40	0.1-0.4	11-41
	Hard	0.4-2.4	41-240	0.5-2.0	52-201
	Broad	0.1-2.4	7-240	0.1-2.4	11-235
Einstein IPC	Soft	0.2-0.8	2-4		
	Hard	0.8-3.5	5-10		
	Broad	0.2-3.5	2-10		

In order to make use of a background map, the data must first be blocked and PI-filtered appropriately.

The background templates for *Einstein* IPC data are obtained in a fixed energy band (Table 6.2), and have a spatial resolution that is 1/2 that of the corresponding QPOE file (*i.e.*, 16"/pixel). While the PROS tasks can take into account the different pixel size in the source and background files with the physical coordinate system (cf. §2.4), the energy filters must be explicitly applied. The background data can generally be excluded from the computation of the errors (the **rbkmap* files have been rescaled to match the background level of the source field, see *Scientific Specification of the Data Analysis System for the Einstein Observatory (HEAO-2) Imaging Proportional Counter*, by F.R. Harnden, D.G. Fabricant, D.E. Harris and J. Schwarz.), and the statistical error associated with them is negligible.

N.B.: ROSAT PSPC background maps in SASS 5-6 or higher are generated by smoothing a source-subtracted, exposure corrected image (see the ROSAT Data Products Guide). Goodness of fit depends on the amount of data left after source subtraction. Maps based upon short exposure images, those with extended sources or strong gradients or very crowded fields are likely to contain spurious structure. Background maps from earlier SASS versions were smoothed without correcting for the ribs and contain structure dominated by the ribs, these are not useful.

6.5.3.1 *imcalc* – Obtain a background subtracted image for ROSAT PSPC Data An energy filter and blocking factor will be applied to match those of the background map. Use the ROSAT broad band background which is a 512 $\times$ 512 array file at 1/30 the **.qp* file resolution, and contains data from PI bins 7–240.

input files: *xdata$rp110590.qp; xdata$rp110590_bk1.imh*

output file: *rp90_a.imh*

```
xr> ximages
xi> imcalc
input file ('-' for STDIN) or imcalc commands:  -             # Interactive mode
IMC)
IMC)  rp90_a.imh = xdata$rp110590.qp[bl=30,pi=7:240] - xdata$rp110590_bkl.imh
                             # Select same size and channels as the background map
                                                       # and subtract the background
 IMC) quit                                # To exit the interactive mode and imcalc
```

The output file is a 512 $\times$ 512 array containing data in the broad PSPC band.

6.5.3.2 Radial profile of an extended source for Einstein IPC data. Obtain a radial profile estimating the background level from a background map.

input files: *xdata$snr.qp; xdata$snr_bbk.imh*

intermediate file: *broad.imh*

output file: *broad_cnt.tab*

```
cl> xray
xr> image
im> imcopy ''xdata$snr.qp[pi=2:10,bl=2]'' broad.imh # Selecting PI channels 2--10
                                                      # and spatial blocking factor 2
im> bye                                                 # Unload unnecessary packages
                                                          # and load the relevant ones
xr> xspatial
```

```
xs> imcnts addbkgderr=no                        # Do not add the background error
source image file:  broad                                            # broad.imh
source region descriptor:  a 251.5 256.5 0 25 n=5
                            # 5 annuli at 251.5,256.5 0 to 25 pixel min/max radii
background image file or cts/pixel:  xdata$snr_bbk
                                    # Broad band background map xdata$snr_bbk.imh
bkgd region descriptor:  a 251.5 256.5 0 25 n=5
                                      # From the same region as the source counts
root name for output file [root_cnt.tab]:  broad   # Output file is broad_cnt.tab
```

Alternatively, the blocking and pi selection can be done as input to the task avoiding the creation of another data file.

input files: *xdata$snr.qp; xdata$snr_bbk.imh*

output file: *broad_cnt.tab*

```
xs> imcnts addbkg=no                                         # Run the task as above
                                                # Do not add the background error
source image file: xdata$snr[pi=2:10,bl=2]
        # energy filtering and spatial blocking on the original xdata$snr.qp file
source region descriptor:  a 251.5 256.5 0 25 n=5
                  # 5 annuli centered at 251.5,256.5, 0 to 25 pixel min/max radii
background image file or cts/pixel:  xdata$snr_bbk
                                    # Broad band background map xdata$snr_bbk.imh
bkgd region descriptor:  a 251.5 256.5 0 25 n=5
                                      # From the same region as the source counts
root name for output file [root_cnt.tab]:  broad                   # broad_cnt.tab
```

6.5.4 *errcreate* – Use of error arrays

It is possible to propagate statistical errors through an analysis procedure via the use of an error array. Error arrays can be used in such tasks as *imcnts* , *imcontour* and *imsmooth*. An error array contains the squares of the errors (it is the data file itself for Gaussian statistics).

With the release of PROS 2.3, library routines and tasks were changed so as to use Poisson statistics by default, rather than Gaussian statistics. For normal operation:

```
xs> errcreate ''xdata$snr.qp[bl=2]''
```

will make a 512×512 array with $16''$ pixels from the Einstein IPC image, SNR.qp. Each array element will contain the square of the error. The output file will be snr_err.imh.

For Gaussian statistics, one needs to change the value of keyword POISSERR in the image header. (For more information, see Section 2.6.2 in this Guide; see also *help errcreate* and *help explain_errors*.

6.5.4.1 Determine a background subtracted radial profile This example uses an error array to determine the errors in the radial profile.

task: *imcnts*

input files: *vigsnr.imh;vigsnr_err.imh;xdata$snr_exp.pl*

output file: *vigsnr_cnt.tab*

```
xs> imcnts err=vigsnr_err expo=xdata$snr_exp.pl expthr=95.0 bkgdex=xdata$snr_exp
    bkgdth=95.0 bkgderr=vigsnr_err
                        # Run the imcnts task with the appropriate parameters
source image file: vigsnr                                         # vigsnr.imh
source region descriptor: a 300 300 0 50 n=5
                   # 5 annuli centered at 300,300, 0 and 50 pixels min/max radius
background image file or cts/pixel: <cr>         # From the same file vigsnr.imh
bkgd region descriptor: a 250 256 100 150                   # Annulus at 250,256
                                                             # 100-150 pixel width
root name for output file [root_cnt.tab]: vigsnr                 # vigsnr_cnt.tab
xs>
```

The command line for *imcnts* is rather long, but most of these are hidden parameters and need to be explicitly given. A more convenient way to do this is to edit the *imcnts* parameter files, and change all the appropriate parameters. However, these will be permanently changed (until the next re-editing) for subsequent use of this task. To do so:

```
xs> eparam imcnts                                    # Edit with your favorite editor
xs> imcnts                                                   # (Set in loginuser.cl)
```

The task will prompt for all auto parameters as above.

6.5.5 To Resample Calibration Images

The ROSAT PSPC exposure and background maps are 128×128 images (60″ pixels) linearly replicated to yield 512×512 arrays (15″ pixels). [NB: The HRI background map is also 512 × 512, but with 8″ pixels.] The user can re-sample these maps using IRAF tasks as follows:

```
cl> images
im> magnify rp110590_mex rp110590_mex_demag dx=4 dy=4 inter=linear     # yields a
                                    # 128*128 exposure map, i.e., true resolution

im> magnify rp110590_mex_demag rp110590_mex_15 4 4 inter=spline3       # yields a
                                                             # 512*512 map with
                                                  # 15'' pixels generated using a
                                                 # 3rd order spline interpolation
```

The user can apply different interpolation functions to obtain the optimum result. Please note that the size of the final image may not be exactly 512*512 due to edge effects of the binning (type *help magnify* for more information). *skypix* can be used to check the sky coordinates against the original image.

6.6 Point Response Functions

6.6.1 On-Axis Point Response Functions (PRF)

Mathematical determinations of some PRFs can be found by typing:

```
xr> help prf
```

This in turn refers to several detailed help files for specific instruments.

For an example of constructing an analytical model PRF for ROSAT, see Section6.8.2.

6.6.2 ROSAT and EINSTEIN Calibration Fields

Three series of ROSAT calibration data are provided for general use. They provide examples of off-axis PRFs, and are situated in a directory with PROS alias *xrcal*. *xecal* has Einstein data.

The list files in the directory provide a key to the data files (also given in the help files discussed in §6.6.1).

6.7 Image Manipulation

6.7.1 *imcopy* – Extract a small image file from a larger image

input file: *rp90_a.imh*

output file: *rp90_aa.imh*

```
xr> images                                          # need this package
im> imcopy rp90_a[200:263,282:345] rp90_aa
                                           # extract 64x64 region from .imh file
rp90_a[200:263,282:345]-->rp90_aa
```

If the input file is a qp file it is necessary to specify the .imh extension on the output file. This prevents the program from trying to make an output .qp file which it cannot do. As an example we make an input file for the smoothing task which needs small arrays or it takes too long to run.

```
im> imcopy xdata$rp110590.qp[6830:7029,9290:9489] rp90_ab.imh
                                      # extract 200x200 region from the .qp file
```

6.7.2 *imcopy* – Compress an image and extract a small region

Note that, in one operation, it is possible to both compress an image and extract a small part of the compressed image.

```
im> imcopy ''xdata$rp110590.qp[block=10][683:702,929:948]'' rp90_ac.imh
```

The coordinates used are the compressed or blocked coordinates, i.e. 6830/10=683, etc.

To limit the range of pulse heights included, the first square brackets should contain e.g. [block=10,pi=7:21].

6.7.3 *imcalc* – Image Arithmetic

imcalc provides basic arithmetic operations plus a host of other functions described in the help file.

```
cl> images
im> imcalc ''imdd = 1.72*  imcc.imh ''              # multiply the value in each
                                                    # pixel of imcc.imh by 1.72.
                                                      # write output to imdd.imh

im> imcalc ''temp_sub=cyg_HSD.imh-40*prf_smo.imh''                 # 40 times the
```

```
                                        # (smoothed) model prf is subtracted from cyg_HSD.imh
```

See Section 6.5.3.1 for another example.

6.7.4 * imlintran , magnify * – Image Scale, Orientation, and Translation

To compare images from different instruments it is desirable for them to have the same scale and orientation. IRAF provides a number of tools which allow manipulation of images. For qp files, see Chapter 4: QPOE Utilities.

To magnify or shrink (both linear transformations), specify the reduction factor in both x and y.

```
  im> imlintran.ncols = 0                         # necessary to set values in empty
  im> imlintran.nlines = 0                       # pixels to zero. Use for shrinking.
  im> imlintran casa casa_m  0. 0. 2. 2. # input image casa.imh is shrunk factor of 2
                                                     # and result put in casa_m.imh
```

or this can be accomplished with,

```
  im> magnify casa casa_m 0.5 0.5.
```

To shift, specify the old and new position of the image center.

```
  im> imlintran casa_m casa_mt  0. 0. 1. 1. xin=256 yin=256 xout = 276 yout=456
                 # moves the image casa_m 20 pixels to the right and 200 pixels up.
```

To embed a small image in a larger (blank) array

```
  im> magnify casa casa_big 1 1 x1=-56  x2=200  y1=-56  y2=200
  # This example places the 144x144 array, ''casa'' at the center of a 256x256 array
                                                        # with no change of scale.
```

To rotate, specify angle of rotation in degrees (positive is counter clockwise).

```
  im> imlintran casa_m casa_mr 30. 30. 1. 1.                 # rotates image
casa_m.imh
                                                                  # 30 deg ccw
  im> imlintran casa_m casa _mf  0. 180 1. 1.              # flips the image,
                                                        # output is mirror image
                                               # There is 180 rotation of y axis
```

A flip about the y axis, however, is more easily accomplished by

```
  im> imcopy casa_m[*,-*] casa_mf
```

Note that several operations can be done at the same time

```
  im> imlintran casa casa_mrt 45. 45. 1.25 1.25 xin=512 yin=512 xout=492 yout=519
```

6.7.5 *qpsim* – Generate Simulated Event List

A simulated HRI event list is generated from a set of user-specified source intensities, shapes and positions. The source specification is input into an *iraf* table using *tcreate* and the PROS-supplied format description. Please refer to the *qpsim* help file for detailed information on the source parameters. Briefly, they are: x, y, counts/rate, intensity (as counts/rate), prf-type, prf-parameter:

```
tt> tcreate
name of table to be created:    (source.lis)
name of file containing column descriptions:  (xspatialdata$srclst.cd)
name of file containing data (STDIN):
Give table data ... then newline & EOF to finish.
4096 4096 rate 0.5 ros_hri INDEF
5000 2500 counts 300 ros_hri 20

tt>
```

Once this table has been created, *qpsim* is run to generate the qpoe file: Note that the "bad attribute name 'time"' message can be ignored, see help file.

```
tt> qpsim
Input source table file:   source
Reference QPOE file (xspatial$data/hdummy.qp):
Root name for output files:   simul

### Generating simulated data ###

Beginning run of SIMEVT ...

Using LIVETIME of 1000.00 seconds to determine source counts

Source number 3 at position (INDEF, INDEF) falls outside
of allowed field limits, [2048 : 6144].
Skipping this source ...

SIMEVT all done.

### Creating QPOE with simulated data ###

FITSNAME      FILENAME              DIMENS     BITPI BSCALE       BZERO        O
```

```
simul.fits    null_image          0
  BINTABLE    simul.tab           2Cx800R    8bin
QPEX Warning: bad attribute name 'time'
QPEX Warning: bad attribute name 'time'
QPEX Warning: bad attribute name 'time'
No GTI records in file - no TIME update done
writing output file  - simul_sim.qp

### Appending simulated data to reference QPOE ###

ref file:   RA: 9:00:00.0000 Dec: 45:00:00.0000 Roll: 0.0000
file # 2  RA: 9:00:00.0000 Dec: 45:00:00.0000 Roll: 0.0000
offsets(arcsec):  RA: -0.0000 Dec: -0.0000 Roll: 0.0000
file # 2  xref: 4096.0000 yref: 4097.0000
offsets(pixels):  xoff: 0.0000 yoff: 0.0000
Opening file #1 : xspatial$data/hdummy.qp
QPEX Warning: bad attribute name 'time'
WARNING: no TSI records found in file #: 1
QPEX Warning: bad attribute name 'time'
No GTI records in file - no TIME update done
No GTI records in file - no TIME update done

file: 1 xspatial$data/hdummy.qp events: 0 times: 0.0
QPEX Warning: bad attribute name 'time'
No GTI records in file - no TIME update done
No GTI records in file - no TIME update done

file: 2 simul_sim.qp events: 800 times: 0.0

@simulapp.lst3840b (; 2 files, no exp.) -> simul.qp
Creating QPOE output file: simul.qp

Renaming output file : simul.qp

tt>
```

6.8 Source Subtraction

For many types of spatial analysis it is useful to subtract spatial models in order to examine residuals for deviations from idealized morphologies. There are many general image processing tools in other parts of IRAF and we urge the user to explore them. Here we discuss a few tasks that are particularly suited for x-ray data.

6.8.1 *immodel* – subtracting a King model

In the following example we use a file which is scheduled for release in the archive at the end of 1994 (rh800021). Although it is possible to make subtractions on photon arrays, we have chosen to first smooth the data because it is easier to evaluate the residuals and intervening steps.

We use imsmooth (section 6.4) to extract a 512×512 array at full pixel resolution (0.5″) with a minimal (compared to the HRI PRF) Gaussian smoothing function; sigma = 4.24 p, producing 5″ FWHM. We then evaluate this image, determining the center of the cluster gas distribution and measure the total number of counts in order to assign an intensity to the model. NB: imsmooth conserves counts.

Running immodel, we provide the following parameters:

```
input  the array created by imsmooth
func King
arg1 70 this is in pixels, so the core radius is 35"
arg2 1.75 for a beta value of 3/4, the exponent is 1.75
normalize yes
scale -1 this is a global scaling; negative sign means subtract.
sources 257,256,5000 pixel coordinates and amplitude.
```

The normalization of intensity is somewhat obscure. The 5000 number is not counts; see below. The residual map can be examined with imcontour, tvproj, and the like to see if the model parameters need changing.

Another way to accomplish the same result is afforded by changing the input specification to "512 512" and the scale parameter to +1. This creates an output file with only the model present, thereby allowing one to measure the effective number of counts in the model with imcnts . In our example we specified 5000 as the intensity of the model. Measuring the effective counts in a circle with r=250 pix (i.e. almost the entire array), we find a value of 9342. Hence the term "obscure" used earlier. Since our data had approximately 10,700 counts in the same circle, we used a scaling factor (SF=1.145) with imcalc (section 6.7.3) in order to minimize the residual:

```
imcalc
input RESIDUAL = SMOOTHED_DATA - SF * MODEL
```

The help file for immodel lists other functions available as well as options for user supplied functions.

6.8.2 *rosprf* – Using the analytical expressions for the point response function

The point response functions for the ROSAT HRI and PSPC are described in their respective help files (prfroshri & prfrospspc). rosprf is a script which builds a model prf by repeating immodel for each functional part of the PRF. In the following example we build an HRI prf and subtract it from one of the HRI boresights (type "help prfroshri" for a list of available boresights). We commence by creating a 255 smoothed array, using

```
input xrcal$/rhlmcx1_03.qp[3458:3712,4304:4558]
[the center was chosen based on the coordinates provided in prfroshri]
arg1 2.54  pixels, equivalent to FWHM=3"
out lmcx1_3
```

We evaluate this map with tvproj (section 5.7) and imcnts (section 6.2), finding 7484 counts in a circle centered on the peak (127,127) with r=100 pixels. Next we generate the PRF.

```
rosprf
xdim 255
ydim 255
center 127 127
out hriprf
instr hri
```

We then measure the "counts" in the model (circle with radius=100 pixels) and find a total of 0.98. We thus derive a scaling factor of 7484/0.98=7651 and use imcalc to perform the subtraction.

```
imcalc,
input hriprf_resid = lmcx1_3_smo.imh - 7651 * hriprf.imh
```

The results look crumby since the source is egg shaped and we see that we are slightly offset. We use imshift to add 1 pixel to each coordinate of the model, hriprf.imh, effectively moving the PRF to the NW. This residual also looks crumby.

NB: For best results, we should have smoothed the PRF (with the same Gaussian used on the map) before subtraction.

6.9 Detect

The detect package contains several tasks which may be run seperately or all together (sequentially) by a script, LDETECT. Each task has an on-line HELP file, and a longer discussion is given in the on-line file EXPLAIN_DETECT. The current version of DETECT is based on the Einstein and ROSAT/HRI level 1 processing algorithms. It is a "sliding-box" routine which uses a detect cell and derives the background from a frame around the cell.

There are 5 basic tasks in the current DETECT package:

CELLMAP — prepares x-ray qp file for DETECT. The qp file is blocked into an image of resolution d/3 (where d = the chosen size of the detect cell). Any other desired x-ray filters (e.g. PHA or PI) are also applied.

SNRMAP — computes the signal-to-noise ratio (snr) map for the image prepared by CELLMAP.

LPEAKS — scans the snr map line by line for local peaks above a user-defined threshold.

BKDEN — computes the mean background density in counts/sq. arcmin. in a user-defined region of the image prepared by CELLMAP. Image pixels with values which deviate from the mean by more than a user-defined threshold are rejected.

BEPOS — determines the maximum liklihood position for each peak identified by LPEAKS, using the original qp file. Signal-to-noise ratios in detect cells centered on the final positions are computed and positions with snr's greater than a second user- defined threshold are identified as sources.

In addition, the following auxiliary tasks are provided:

LDETECT — runs the above 5 tasks in sequence, for a single detect cell size and threshold.

DETMKREG — makes a region file from Detect output positions (_ruf.tab, _pos.tab, _unq.tab, _mch.tab). The file(s), when overlayed onto SAOIMAGE are a visual tool indicating where source detection in the image was determined.

LBMAP — generates background and background variance maps at the same resolution as in CELLMAP. These maps are for illustrative purposes only and are not used in other DETECT tasks. The equations used to build the background map use the same algorithms as implemented in BEPOS.

LMATCHSRC — consolidates detections from one or more runs of ldetect on the same qpoe file into unique sources in a field. The distance between input detections are computed and those falling within a match tolerance are determined to be the same source.

In the examples below we give a minimal run to show the basic operation. Other parameters may be used by experimentation or by reading the help files.

EXAMPLE 1 - running LDETECT - ROSAT HRI

In this case we specified only the cellsize, leaving the default thresholds and default circle for the background measurement.

```
  xp> ldetect

qpoe file : xdata$rh110267.qp
Output table root: rh110267
```

```
ROSAT HRI Cellsize in arcsecs - 6|9|12|24|36|48: 24

... Running Cellmap ...

xdata$rh110267.qp[bl=16] -> rh110267.imh
add rh110267.imh,cellsize = 24
rh110267.imh updated
xdata$rh110267.qp

... Running Snrmap ...

Convolving 3x3 cells

Convolving 5x5 cells

Calculating Snrmap with equation:

snrmap=(1.7777*cell3-frame)/((1.7777**2*cell3+oframe)**0.5)

Writing Snrmap Image:
rh110267_snr.imh

... Running Lpeaks ...

src#  logx    logy     physx     physy    cellx/y     snr
---------------------------------------------------------


   1  149.00  177.00   2384.00   2832.00   24   24     5.954
   2  146.00  276.00   2336.00   4416.00   24   24    30.197

Writing to Output file:  rh110267_ruf.tab

... Running Bkden ...

Bkden region is circle 256.00 256.00 7.50'
```

```
Bkden = 12.8558464 counts/arcmin**2

... Running Bepos ...

*** PI bands must be Respecified with input QPOE ***


Input Qpoe: xdata$rh110267.qp
bkden = 12.8558464 (cts/sq arc-min), 0.0008928 (cts/sq pixel)

2 Rough Position(s) Input

 src#     ra            dec        physx    physy  cellcnts framecnts snr
--------------------------------------------------------------------------
   1    22:08:38.4    45:31:24.2   2380.1   2829.1      95       67     5.3
   2    22:08:41.4    45:44:33.4   2323.5   4407.7    1506      394    31.8

Writing 2 Sources to Output file:  rh110267_pos.tab
```

EXAMPLE 2 - running the 5 components separately - ROSAT PSPC

de> cellmap

```
Input xray Qpoe: xdata$rp110590.qp
Output xray image : rp110590_cmp
ROSAT PSPC Cellsize in arcsecs - 30 [h/b]| 45 [s]| 60 [s/h/b]| 120 [s/h/b]: 30
Energy Band (soft|hard|broad) (broad): hard

xdata$rp110590.qp[bl=20,pi=52:201][128:639,128:639] -> rp110590_cmp.imh
add rp110590_cmp.imh,cellsize = 30
rp110590_cmp.imh updated
xdata$rp110590.qp[pi=52:201]

de> snrmap
Input xray image : rp110590_cmp
Output snrmap image [root_snr.imh]: rp110590
```

```
Convolving 3x3 cells

Convolving 5x5 cells

Calculating Snrmap with equation:

snrmap=(1.7777*cell3-frame)/((1.7777**2*cell3+oframe)**0.5)

de> lpeaks
Input Snrmap Image: rp110590_snr
Output Rough Positions Table: rp110590

src#  logx    logy     physx     physy    cellx/y     snr
---------------------------------------------------------

   1  224.00  263.00   7020.00   7800.00   30   30     4.372
   2  220.00  340.00   6940.00   9340.00   30   30    12.945

Writing to Output file:  rp110590_ruf.tab

xp> bkden
Input Image: rp110590_cmp
Region Descriptor (default): BOX(226,115,344.00,94.00)
[This box was determined by examining the cellmap and finding
a source free region.]
Bkden region is BOX(226,115,344.00,94.00)

Bkden = 0.6401532 counts/arcmin**2

xp> bepos.snr_thresh=3.5
xp> bepos

*** PI bands must be Respecified with input QPOE ***

Input qpoe file: xdata$rp110590.qp[pi=(41:240)]
Field background density in cts/arcmin**2 (0.82384938001633):
```

```
Input rough pos table: rp110590_ruf.tab
Output bepos table file: rp110590

Input Qpoe: xdata$rp110590.qp[pi=52:201]
bkden = 0.6401532 (cts/sq arc-min), 0.0000445 (cts/sq pixel)


2 Rough Position(s) Input

 src#      ra            dec       physx    physy  cellcnts framecnts snr
---------------------------------------------------------------------------
   1    22:08:37.8    45:31:46.7   7008.0   7794.9      41       19     4.3
   2    22:08:41.6    45:44:45.4   6929.6   9352.5    1607     1865    11.9

Writing 2 Sources to Output file:  rp110590_pos.tab
```

EXAMPLE using LMATCHSRC -

Two table files are output. The first (_unq.tab) is a unique list of all input detections where each has a unique row reference; the second file (_mch.tab) is the output of the match. This file is a match reference list with one row for each source. Match columns refer to the detections listed in the _unq.tab file.

Match sources from 4 runs of ldetect at different detect cell sizes.

```
  de> lmatchsrc

Reference Qpoe file name: xdata$rh110267.qp
List of detect tables to match [_pos.tab]: rh12_pos,rh24_pos,rh36_pos,rh48_pos
Root name for Match Source table: rharlac

... Merging the list of input tables

... Sorting the list in y,x order
```

```
... Matching the detections into Sources

      Avg ra       Avg dec    Avg x     Avg y     Best x    Best y   Src ref
                              (physical pixels)   (physical pixels)  (row #)
------------------------------------------------------------------------------
  1 22:08:38.4   45:31:24.3  2380.332  2829.285  2380.510  2829.540  3 2 1
  2 22:09:01.3   45:37:14.3  1902.477  3531.475  1902.477  3531.475  4
  3 22:08:41.4   45:44:33.4  2323.537  4407.728  2323.539  4407.733  6 8 7 5 9

Writing to Match Results table:  rharlac_mch.tab
Writing to Match Reference table: rharlac_unq.tab
```

7 Spectral Analysis

The x-ray spectral analysis consists of a comparison between the observed pulse height spectrum and a pulse height spectrum calculated using a given spectral model. The model, chosen by the user, is convolved with the instrument and telescope responses.

To obtain the spectral parameters for an x-ray source, several steps need to be followed. For each source:

- extract counts (pulse height spectrum) from the source or region of interest
- select background region and extract counts
- obtain the distribution of the gain values – for *Einstein* IPC only
- obtain the best fit parameters for a spectral model - repeated for each model of interest

- obtain a 1-D or 2-D grid of χ^2 values around the best fit values - this can be done for as many models as desired
- determine the flux and luminosity of the source from the best fit model

Plots of the gain distribution, the pulse height distribution, the data compared to the model spectra, and of the χ^2 grids can be generated.

7.1 Summary of Tasks in this Section

Task	Function	Packages Needed
bal_plot	plots the BAL histogram(*Einstein* IPC only)	xray:xspectral
counts_plot	plots observed and model spectra on screen	xray:xspectral
dofitplot	script to run fit and counts_plot	xray:xspectral
downspecrdf	convert input spectral data from RDF to pre-RDF format	xray:xspectral
fit	minimizing fit of model to pulse height spectrum	xray:xspectral
grid_plot	plots contours of constant χ^2	xray:xspectral
hxflux	commute X-ray flux from HRI count rate	xray:xspectral
intrinsicspecplot	script to plot the emitted and observed spectrum	xray:xspectral
qpspec	extracts pulse height spectrum from chosen region	xray:xspectral
search_grid	calculates χ^2 for grid of parameters	xray:xspectral
show_models	display the current model parameters	xray:xspectral
singlefit	fixed parameter model fit	xray:xspectral
upspecrdf	convert input spectral data from pre-RDF to RDF format	xray:xspectral
xflux	compute x-ray from models	xray:xspectral

Non-PROS tasks:

proscon	conversion from pre-RDF PROS spectral tables to *xspec* input format	
xspec	XANADU spectral fitting package.	

7.2 A Quick Look at Spectral Fitting

This section describes a short procedure for finding the best fit parameters of a chosen model and the flux and luminosity of a source.

```
xs> qpspec xdata$rp110590 ''c(6930,9390,240)'' xdata$rp110590 ''a(6930,9390,300,600)''
          # extract the source counts from a 2' circle centered on the target and
      # the background counts from an annulus between 2.5' and 5' from the source

xs> dofitplot rp90 ''abs(20:22)*pow(0:3)'' print_table=yes   print_plot=yes
        # This command fits an absorbed power-law model to the data, brings up a
              # plot of the predicted and observed pulse height distribution and
           # generates hardcopies of both the plot and the table of observed and
                                                               # predicted counts.

xs> xflux fluxin=rp90 energy=''0.1:2.4'' dis=''1.0k''  # Calculate the integrated
   # flux betwen 0.1 and 2.4 keV and the luminosity using a distance of 1.0 kpc.
```

7.3 Notes on IPC and PSPC Spectral Fitting

The first step in any spectral analysis is to extract the photon distribution from both a source and a background region. Any size or shape of region can be specified. The photon distribution can be based on either the pulse height analyzer (PHA) channels or the pulse invariant (PI) bins. To convert the PHA distribution into a PI distribution, the instrument gain must be known. For ROSAT PSPC data, this conversion is done during the on-line (SASS) processing. For IPC observations, the instrument gain varies both across the detector and in time, and is applied during off-line processing (PROS) using the BAL histogram.

By default, spectra are extracted in PI bins for PSPC data and in PHA bins for IPC data.

The background is normalized to the source counts using the ratio of the source-to-background areas. The user has the option to adjust the normalization factor. For IPC data, the background should be obtained within the same field if possible, since the software assumes that the on-source BAL histogram applies to the background region as well.

The errors will be calculated directly from the counts, with the error on the background included. The user has the option of including a systematic error, which is added in quadrature; by default, IPC data has a 3% systematic error included.

Special techniques are required when using *Einstein* IPC data. If the source is a point source, the mirror scattering corrections have been obtained for a $3'$ radius circle (or 22.5 pixel). The corrections will be properly applied only if the source counts are obtained from a $3'$ radius circle.

As a consequence of the ongoing inflight calibration effort, there are several versions of the PSPC spectral response matrix available for fitting data in PROS. These are given in Table 7.1:

Table 7.1: Available calibration files for the ROSAT PSPC.

Response matrix	area file	filter area file	Comments
	PSPC-B	PSPC-B	(used since Jan 1991)
	PSPC-C	PSPC-C	(used prior to Jan 1991)
dtmat.ieee	offar.ieee	filter.ieee	Default in PROS 1.0 Pre-flight versions. non-detector specific
dtmat_5.ieee	offar.ieee	filter.ieee	First post launch matrix non-detector specific
dtmat_6.ieee	offar2_5.ieee offar1_5.ieee	filter2_5.ieee filter1_5.ieee	Default in PROS 2.0 Same as pspcb_mar11 and pspcc_mar11 in XSPEC.
dtmat_36.ieee	offar2_6.ieee offar1_6.ieee	filter 2_5.ieee filter 1_5.ieee	Default in PROS 2.2 and 2.3 same as 93jan12 in XSPEC.

Data taken with the boron filter are automatically fitted using the correct area files. Type *help pspc_fitting* for more information on these calibration files, and how to specify calibration files other that the default ones.

Note that for spectral analysis of off-axis or extended sources, PROS computes an "off-axis histogram" of the extracted photons, and the fitted model is appropriately corrected for vignetting.

For PSPC data the background spectrum can be corrected for the differential vignetting of the mirror. The charged particle component can also be removed for PSPC data. (Type *help pspc_fitting* for more details).

Although the PSPC detectors have 256 PI channels, these are binned into 34 channels for both on-line (SASS) and off-line (PROS) spectral analysis. The correspondence between the 256 channels and the 34 channels is given below.

BINNING SCHEMES

34-bin number	Energy Low - High	256-channels included	34-bin number	Energy Low - High	256-channels included
1	0.07 - 0.09	7 - 8	18	0.84 - 0.91	84 - 90
2	0.09 - 0.11	9 - 10	19	0.91 - 0.99	91 - 98
3	0.11 - 0.14	11 - 13	20	0.99 - 1.07	99 - 106
4	0.14 - 0.17	14 - 16	21	1.07 - 1.15	107 - 114
5	0.17 - 0.20	17 - 19	22	1.15 - 1.23	115 - 122
6	0.20 - 0.24	20 - 23	23	1.23 - 1.32	123 - 131
7	0.24 - 0.28	24 - 27	24	1.32 - 1.41	132 - 140
8	0.28 - 0.32	28 - 31	25	1.41 - 1.50	141 - 149
9	0.32 - 0.37	32 - 36	26	1.50 - 1.60	150 - 159
10	0.37 - 0.42	37 - 41	27	1.60 - 1.70	160 - 169
11	0.42 - 0.47	42 - 46	28	1.70 - 1.80	170 - 179
12	0.47 - 0.52	47 - 51	29	1.80 - 1.91	180 - 190
13	0.52 - 0.58	52 - 57	30	1.91 - 2.02	191 - 201
14	0.58 - 0.64	58 - 63	31	2.02 - 2.13	202 - 212
15	0.64 - 0.70	64 - 69	32	2.13 - 2.24	213 - 223
16	0.70 - 0.77	70 - 76	33	2.24 - 2.36	224 - 235
17	0.77 - 0.84	77 - 83	34	2.36 - 2.48	236 - 247

7.4 Model Spectra for Spectral-Fitting Program

For on-line information type: cl> *help using_spectral*

The models provided are:

- abs - absorption of x-rays by neutral gas with solar abundances.

  parameter 1 = logarithmic value of Galactic N_H in units of cm^{-2}
  parameter 2 = logarithmic value of Intrinsic N_H in units of cm^{-2}
  parameter 3 = redshift of source

  If only one parameter is given, it is assumed to be the Galactic N_H.

If only two parameters are given, they are assumed to be the Intrinsic N_H and the redshift.

- pow - power law

  parameter 1 = normalization - logarithmic value of the flux density at 1 keV in units of keV cm^{-2} s^{-1} keV^{-1}
  parameter 2 = energy index defined as $S_\nu \propto E^{-\alpha}$, where S_ν is the flux density

- bre - thermal bremsstrahlung (exponential plus gaunt factor)

  parameter 1 = normalization - logarithmic value of the flux density at 1 keV in units of keV cm^{-2} s^{-1} keV^{-1}
  parameter 2 = temperature in the source frame in KeV

- exp - pure exponential

  parameter 1 = normalization - logarithmic value of the flux density at 1 keV in units of keV cm^{-2} s^{-1} keV^{-1}
  parameter 2 = temperature in the source frame in KeV

- bla - black body

  parameter 1 = normalization - logarithmic value of the flux density at 1 keV in units of keV cm^{-2} s^{-1} keV^{-1}
  parameter 2 = temperature in the source frame in keV

- ray - optically thin thermal spectrum with metallic lines (Raymond code)

  parameter 1 = normalization - logarithmic value of the ratio of the volume emission measure to $4\pi D^2$, where D is the distance to the target. Units of the normalization are cm^{-5}.
  parameter 2 = temperature in the source frame in keV
  parameter 3 = abundance table
  c - cosmic
  m - Meyer
  parameter 4 = percentage of heavy elements (100 is the solar value)

- line - single gaussian line model

  parameter 1 = normalization - logarithmic value of the normalization of the gaussian in units of keV cm^{-2} s^{-1}.
  parameter 2 = line centroid in keV
  parameter 3 = FWHM of the line in keV

For detailed information on model parameters, please refer to the science specifications in *xspectraldoc$spectral.(tex,dvi)*.

The spectral model and its parameters are passed to the fitting tasks as:

$$absorption(params) * model(params)$$

More than one absorption and more than one spectral model can be combined in the following ways:

*absorption1(params)*model1(params)+absorption2(params)*model2(params)...*

or

absorption(params)(model1(params)+model2(params)...+modelN(params))*

Some basic rules for writing a model description:

- Commas or spaces separate parameters
- Lo:hi should be given when a parameter should vary within these boundaries. A single number indicates a fixed parameter value.
- There are no implicit parameters (*i.e.*, all parameters must be specified). The exception is the normalization. For a single model the normalization does not need to be specified and will be calculated. For a combination of models, however, either a range or a specific value for the normalization must be given for all but the first model. The normalization in the first model will be calculated as before. A single value for the normalization (i.e., fixed normalization) will force the program to use that value.
- Normalization for models 2 ... N are relative to that of model 1. To specify a fixed normalization for model 2, 50% that of model 1, the syntax is:

  abs(18:20) * (power(0:2) + brems (log(0.5), 0:20))

NOTE: When different spectral models with different units for the normalization are combined, the relative normalization of the second model can be quite large.

For example: ray(1:2,c,100)+pow(11:13,1:2)

specifies a power-law model with a flux of approximately 1% of the flux in the Raymond model.

See cl> *help models_spectral* for more information on syntax.

Note: The program *fit* starts at the midpoint of the parameter range specified, then changes the parameters to determine the direction in which χ^2 is decreasing and heads in that direction to find the parameter values that give the minimum χ^2. If the initial parameter range is large, only part of the parameter space may be tested, and the program can settle into a secondary minimum. The real minimum can be missed. Choose parameter limits carefully.

Examples:

1. abs(20:21)*pow(1:2)

   to describe a power law model spectrum, with energy index varying between 1 and 2, and a galactic absorption varying between 10^{20} and 10^{21} cm^{-2}.

2. abs(20:21,23,1.0)*(pow(1:2)+pow(log(0.5):log(0.6),1.5)

   for two power law models, and one absorption applied to both of them. The galactic absorption can vary between 10^{20} and 10^{21} cm^{-2}, the intrinsic absorption is fixed at 10^{23}, for an object at z=1. The energy index of the first power law model can vary between 1 and 2, with the best fit normalization calculated by the program. The second power law has a fixed index of 1.5, and its normalization is allowed to vary between 50 and 60 % of the first one.

3. abs(20:21)*pow(1:2)+abs(21:22)*pow(0.1,2:3)

   to describe two power law models, each with galactic absorption in two different ranges.

7.5 Structure of the Spectral Package

Several of the tasks in the spectral package share a common set of parameters. These parameters reside in the task *pkgpars*. Each task in the spectral package has a hidden parameter which points to the pkgpars parameter file. The parameters stored here include the observed data sets and the model descriptor, and instrument-specific parameters. Each task also has its own parameter file.

To examine the common parameter file, type:

```
xs> lparam pkgpars | page
```

The task *pkgpars* allows the user to edit the pkgpars.par file as if they had called *eparam pkgpars*.

```
xs> pkgpars
```

It is also valid to use the other IRAF methods to change this parameter file.

```
xs> pkgpars.<param>=<newvalue>
```

This will set the value of *param* to *newvalue*, overwriting the previous value.

With this structure, the data and model may be passed from one task to the next without further user intervention, but the user has the option to make a change at any time. Because the observed data set and the model are in the *pkgpars* parameter file and not in the parameter file for each task, it is not possible to use them as positional arguments on the command line as with other automatic parameters. But one may change them on the command line using the syntax of hidden parameters. For example, to use these parameters without the user being prompted, one could type:

```
xs> fit observed=rp90 model=''abs(21:23)*pow(4:7)''
```

The observed counts and error data set contained in the table file (**_obs.tab*) produced by *qpspec* (cf. §7.6) will be the input data for subsequent spectral tasks.

Spectral fits (tasks *fit* , *singlefit*) will output a distinct table file, the *_prd* table. The *_prd* table contains: the input data set and errors, the input model (imod), the output best fit parameters and corresponding count distribution, and the output model (omod). 'omod' is a string that contains the model centered on the best fit parameter values. This latter can be the input for the *search_grid* (cf. §7.9) task.

The task *fit* also generates a table for the best fit spectral model, with the extenstion *_int*.

The user has no control over the *_prd* file name: it will have the same root as the input file, with the new extension '*_prd*' instead of '*_obs*'. It will be written in the current directory by default, regardless of where the input *_obs* file is stored. (Everything to the left of the last "/" or "$" character in the input table name is the 'directory path', and it will not be part of the *_prd* table name.) The user has the option of changing the output directory by setting the parameter *prd_dir* in the *pkgpars* parameter file to the desired value (null string $\equiv$ current directory).

Since some of the spectral tasks require both the *_obs* and the *_prd* tables as input, but ask for only the observed data set, it is important that the 'root' name remain the same; otherwise the task will not find the predicted data set.

Subsequent runs of the model fitting routines will produce different best fit models. The user has the option of keeping only the results from the last run, or to store all of them in separate columns in the output table. In the latter case, subsequent runs will be numbered sequentially. The parameter *predict* in the *pkgpars* parameter file regulates this choice.

Multiple data sets can be fitted to the same model. Separate input data set names should be given at the prompt, separated by semicolons. Multiple *_prd* tables will be the output, with one for each input data set. The same rules for the output table names apply, so that each *_prd* will have the same 'root' name as the corresponding *_obs*, and all will be written in <u>the same</u> directory, regardless of whether the input tables are in the same or in separate directories.

The choice of spectral channels to be used in the fit is regulated by a hidden parameter in the pkgpars parameter file. The channels can be changed easily at run time, by specifying them within [ ] at the end of the input file name. The section on Limiting Energy Channels (§7.7.4) gives an example of this.

7.6 *qpspec* – Extraction of Photon Distributions from QPOE files

A pulse height distribution can be extracted from a QPOE file only if the event structure (see §2.5.2) is 'large', 'region', or 'full'. The source position in the field must be known, and the region where the background is to be obtained must be supplied. *SAOimage* (§5.2) can be used to display an image, and the cursor used to find the source coordinates. Some instructions for defining regions can be found in Section 6.2. The regions help file also contains instructions for defining regions, and a list of the available shapes.

With the new RDF format files, the output from *qpspec* for most instruments is split into three files:

_obs.tab	counts
_soh.tab	source off-axis histogram
_boh.tab	background off-axis histogram

For Einstein IPC, there are no off-axis histogram files, but there is a BAL histogram file, _bal.tab.

To convert to the pre-RDF form, the task downspecrdf (Section 7.12) should be used. To convert from pre-RDF form, use task upspecrdf (See Section7.12)

7.6.1 Spectral Extraction of Point Sources

input file: *xdata$rp110590.00.qp*

output files: *rp90_obs.tab, rp90_soh.tab, rp90_boh.tab*

```
cl> xray
xr> xspectral
xs> qpspec
source qpoe file:  xdata$rp110590                  # No *.qp extension is needed
source region descriptor:  c 6930 9390 240           # A 240 pixel radius circle
                                                        # centered at 6930,9390
background qpoe file: xdata$rp110590                   # From the same QPOE file
bkgd region descriptor:  a 6930 9390 300 600  # an annulus centered at 6930,9390
                                   # with inner and outer radii 300 and 600 pixels
root name for output file [root_obs.tab]:  rp90    # rp90_obs.tab, rp90_soh.tab,
                                                                 # rp90_boh.tab
```

If the user wants to specify a different normalization for background subtraction, or a systematic error, it must be done when using *qpspec* (see the *qpspec* help file).

7.6.2 Spectral Extraction of Extended Sources

When analyzing extended sources the background must be obtained from a region well separated from the source. To correct the background for the differential vignetting (i.e., the difference between source and background vignetting) of the ROSAT mirror, set the vign_corr flag in qpspec to yes. For example:

xs> qpspec vign_corr=yes

7.6.3 Charged Particle Subtraction

Charged particles can be removed from the source and background spectra using the master veto (MV) rate. (Type *help pspc_fitting* for a description of how to determine the average MV rate during an observation). For example, if MV rate = 100, the following command will remove the charged particles in the background and source spectra before determining the net spectrum.

xs> qpspec vign_corr=yes avg_mvr=100

Note that the avg_mvr parameter is effective only if vign_corr is yes.

7.6.4 Error Determination

The errors (neterr) in each channel as listed in the *_obs.tab* file are given by:

```
neterr=sqrt (src_less_part_err**2 + bkgd_norm_err**2)
```

where src_less_part_err is the error on: the source counts less the source charged particle counts; and bkgd_norm_err is the error on: the background counts less the background charged particle count, normalized by the ratio of source area to background area, by the vignetting ratio of effective areas, and by any user-supplied factor.

The error on each count is estimated from the count on the assumption that it arises from a Poisson process (by default). The error on each sum or difference of counts is estimated as the sum in quadrature of the estimated errors on the constituent counts.

If a systematic error is applied (this is the default for the IPC: *pkgpars.ein_ipc_syserr="0.03"* and *qpspec.syserr=""*) then

```
final_neterr=sqrt(neterr**2+[net*syserr]**2)
```

Parameter names are, as far as possible, identical to those in the *_obs.tab* file.

7.7 *fit* – Fit Data to a Specified Model

With PROS2.3 all fitting is performed using Poisson statistics (see Section2.6.2), rather than Gaussian, so that results will be correct even for small counts (≤ 10) in some channels.

7.7.1 Power Law fit to a Point Source

The power law model with galactic absorption will be applied to ROSAT data the extracted in §7.6.1. The results will be stored in the *_prd* table in the current directory.

input files: *rp90_obs.tab, rp90_soh.tab, rp_boh.tab*

output file: *rp90_prd.tab*

NOTE: If an RDF SASS output file is used as input to *fit* (name of the form: *rp<ROR#>_sp###.tab*) then *fit* looks for an accompanying *_oah###.tab* source histogram file and runs without the background histogram.

NOTE: The *rp_boh.tab* file is not needed for the calculation of the fitted model. It is only used for the execution of fit.

```
xs> fit
Performing the Simplex minimization fit.
observed spectrum [root_obs.tab]: rp90              # will fit ROSAT data using
                                           # the default PSPC channels, 3:34.
model descriptor or ? for help: abs(19:21)*pow(0.5:1.5)         # IRAF will not
      # accept .5 as a valid fraction.  Numbers <1 must be entered starting with
                                                          # a zero, e.g., 0.5.
```

7.7.2 Raymond Model

The model is a Raymond thermal spectrum, temperature range is 1 to 4 keV, element abundance table is cosmic (the other choice is Meyer), abundance of heavy elements (atomic number > 2) is 100% of the value in the table (i.e. solar). Type "help models" for a list of other possible abundance fractions.

```
sp> fit

Performing the simplex minimization fit.
Observed spectrum [root_obs.tab]:} rp90
Model descriptor or ? for help:  abs(17:19)*ray(1:4,c,100)  # free parameters are
                                    # absorption, temperature, and normalization.
```

7.7.3 Two-Temperature Raymond Model

The model is a two-temperature Raymond thermal spectrum, low temperature range is 0.55 to 0.65 keV, high temperature range is 1 to 2 keV, element abundance table is cosmic, abundance of heavy elements (atomic number > 2) is 50% of the value in the table. Note that $0.7 = \log(5.0)$, specifies that the second component normalization will be 5 times the normalization of the first component. A multiple component fit requires that this normalization (or a range of normalizations) be specified for the 2nd, 3rd, etc. components. Type *help models_spectral* for a list of possible choices.

```
xs> fit

Performing the simplex minimization fit.
Observed spectrum: rp90
Model descriptor: abs(20)*(ray(0.55:0.65 c 50)+ray(0.7 1:2 c 50))      # The free
                        # parameters are now 2 temperatures and one normalization.
 # Note that IRAF will not accept .7 as a valid fraction. The number must be 0.7.
```

7.7.4 Limiting Energy Channels Input to the Fit

The energy range covered by the default channels depends on the instrument gain. To limit the fit to channels 7 to 30, specify the file with observed spectrum as:

```
rp90[7:30]                   # will fit as before but using PSPC channels 7 to 30
                                                         # in the 34 binning scheme
```

7.7.5 Multiple Data Sets

Any number of data sets (_obs.tab files), from the same or different instruments, can be fitted simultaneously by separating the root file names with semi-colons.

rp90; xdata$snr

7.7.6 Verbose Mode

There is a hidden parameter in *fit* that controls the amount of output sent to the screen. By setting verbose=yes the results from each individual iteration of the fit will appear on the screen. The output will have the number of the iteration, the chi-square value, and the free parameters (including the normalization factor calculated after the fit to reduce the total chi-square).

```
  xs> fit verbose=yes
```

7.7.7 Fit a source with a small number of net counts

If the number of net counts for a source is less than ~ 100 it is impractical to fit the data in all 34 energy bits. However, if the flux is required, then a fit must be performed first. This can be done by setting the hidden parameter *rebin=yes*, which bins the data into a single bin before fitting for the normalization only. The user must specify all the parameters in the chosen model except the normalization. *Fit* will then calculate the normalization of the model and xflux (see Section7.11) can be used to determine the flux and luminosity of the source from the best fit model.

```
xs> fit rebin=yes
```

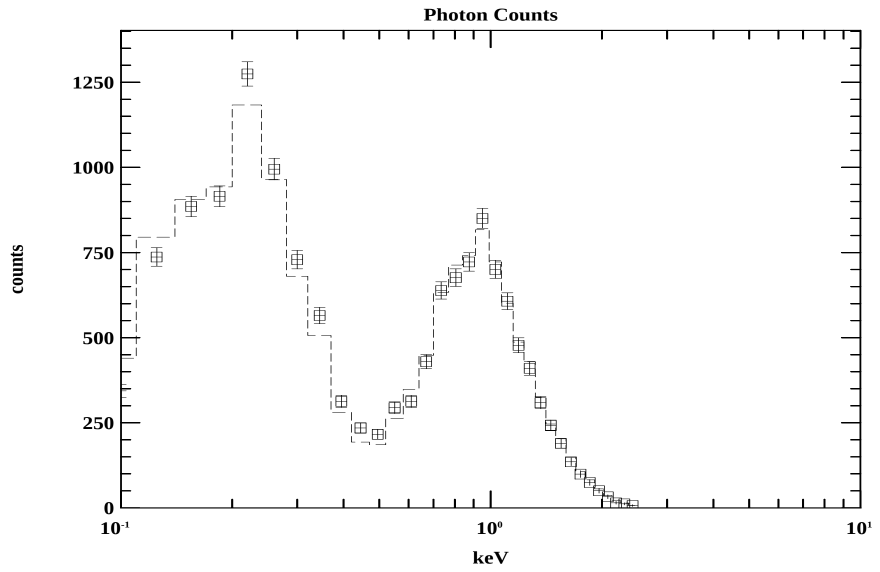
7.7.8 Inspection of results

The table of results obtained with *fit* can be listed with the *stsdas.ttools* task *tprint* .

```
cl> tables
ta> ttools
tt> tprint rp90_prd                                # input file is rp90_prd.tab
```

The table will be listed on the screen.

```
tt> tprint rp90_prd prparam=yes             # will list the table and its header
                      # parameters, which include the best fit spectral parameter
```

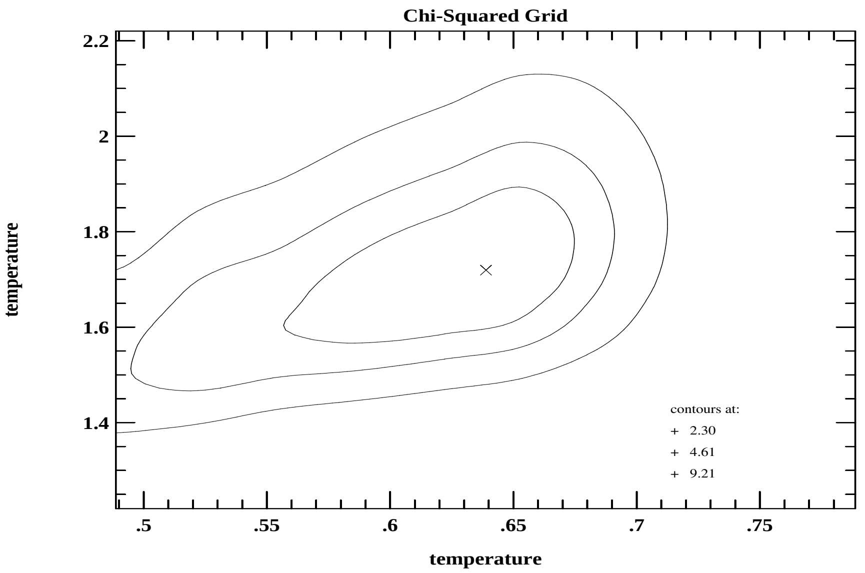


(a)

min Chi-Squared = 60.69 for abs(20.0000)*ray(11.9984 0.6389 cosmic 50)+ray(0.7000 1.7195 cosmic 50)

using abs(20.00)*ray(0.59:0.69 cosmic 50.00)+ray(0.70 1.22:2.22 cosmic 50.00)

rp90_obs[3:34]



(b)

min Chi-Squared = 60.69 for abs(20.0000)*ray(11.9984 0.6389 cosmic 50)+ray(0.7000 1.7195 cosmic 50)

using abs(20.0000)*ray(0.4887:0.7887 cosmic 50)+ray(0.7000 1.2200:2.2200 cosm

rp90_obs.tab[3:34]012

Fig. 7.1a. Output from *fit* showing the data and best fit model folded through the instrument response (dashed line). Fig. 7.1b. Output from *search_grid* showing the best fit and confidence contours, at $\Delta\chi^2$ shown, for a two-temperature Raymond thermal model fit.

7.8 *counts_plot* – Inspection of Results

The results can be displayed graphically:

```
xs> counts_plot
observed spectrum [root_obs.tab]  rp90                        # input is rp90_prd.tab
```

A plot will appear on the screen, showing data points with error bars, and the predicted model as a histogram when one is present. Best fit model parameters are printed at the bottom. The task will leave the user in graphics mode so that various plotting options can be applied.

? lists the graphics options.

q exits the task.

Options include:

p to undisplay/display the predicted counts,

l to remove labels,

d and *s* for plots of 'observed-predicted' and '(observed-predicted)/error' respectively, and

= for hard copy.

These options are specific to counts_plot. Fig. 7.1a. shows an example plot.

7.9 *search_grid* – Grid of χ^2 - default mode

After the best fit is obtained, the confidence limits on two parameters can be obtained by running *search_grid* . This task calculates a grid of χ^2 values for a given input model. The task will also calculate the absolute minimum [within the limits of the grid or the limits from the input model spectrum, whichever is larger], and use it to draw the contours (χ^2_{min}+$\Delta\chi^2$).

In the simplest case the user can hit <cr> at the first few prompts. Then *search_grid* will select the last dataset used (from *pkgpars*), will use the best fit from the last run of *fit* (the null string "" at the model prompt will always select the best fit from the *_prd.tab* file), and will place the results in a *_grd.tab* file with the same root as the observed file. A grid will be built around those values, incrementing out to ± *delta* (provided by the user) in n_x steps in the x dimension and n_y steps in the y dimension (n_x and n_y provided by the user). *search_grid* will return $n_x \times n_y$ χ^2 values, a χ^2_{min} on the grid points, and a true minimum (which will coincide with the central grid point in this simple case).

input files: *rp90_prd.tab; rp90_obs.tab*

output file: *rp90_a_grd.tab*

```
xs> search_grid
Performing a grid search.
observed spectrum [root_obs.tab] = (rp90):  <cr>
                                                  # input data from rp90_obs.tab
model descriptor [null --> last best fit]:  <cr>                  # The best fit
                                              # parameters are from rp90_prd.tab
root name for output file [root_grd.tab]:  rp90_a
                                       # output is rp90_a_grd.tab.  Root changed
       # because you will probably want to repeat this with different parameters
```

The best-fit model parameters (from previous fit) will be listed on the screen.

```
Please choose the grid axes:

  model component number for the first axis (1): <cr>
  first axis parameter (norm|index|temp|galacticNh|intrinsicNh):  temp
  half range of first axis parameter:  0.5
  number of steps in the first axis (1:):  10
  model component number for the second axis (1):  2
  second axis parameter (norm|index|temp|galacticNh|intrinsicNh):  temp
  half range of second axis parameter:  0.5
  number of steps in the second axis (1:): 10
```

The best-fit parameters from this current grid of fits will be listed on the screen.

The output file is in table format, and can be listed with the *tables.ttools* task *tprint* (cf. §6.2.7).

7.9.1 *search_grid* – Customized χ^2 grid

The grid of χ^2 values can be built around any parameter by explicitly giving the center point of the grid (in response to the model descriptor prompt). If the model descriptor provides only fixed values (as in the example below), the search for the true minimum will start from the grid χ^2_{min}, and the parameters can vary within 'best-fit-parameter$\pm\delta$', where δ is the user input to define the range of the χ^2 grid (half range of first axis parameter).

input file: *rp90_obs.tab*

output file: *rp90_b_grd.tab*

```
  xs> search_grid

Performing a grid search.
  observed spectrum [root_obs.tab]: rp90
  model descriptor:  ray(0.3 c 100)+ray(0.67 1.26 c 100)
  root name for output file [root_grd.tab]:  rp90_b

The chisquare grid will be built around the following model parameters:
```

These are listed on the screen.

```
Please choose the grid axes:
```

```
  model componet number for the first axis (1):  <cr>
  first axis parameter (norm|index|temp|galacticNh|intrinsicNh):  temp
  half range of first axis parameter (0.4):  0.25
  number of steps in the first axis (1:) (7):  5
  model component number for the second axis (1):  2
  second axis parameter (norm|index|temp|galacticNh|intrinsicNh):  temp
  half range of second axis parameter (0.2):  0.5
```

```
  number of steps in the second axis (1:) (7):  5

Best grid fit found at grid point: 3 3
       First parameter value =     3.000E-1
      Second parameter value =     1.260E0
      Chi-square = 48.181
```

Final fit model, best fit parameters, χ^2 will be listed on screen.

7.10 *grid_plot* – Plot Results on Screen

input file: *rp90_a_grd.tab*

```
  sp> grid_plot rp90_a
```

Plot will appear in graphics window. With cursor in window, type = for hard copy, type *q* to exit from graphics mode. Fig. 7.1b. shows a sample result.

7.11 * xflux , hxflux * – Calculation of X-ray Flux

The conversion of count rate to flux depends on the intrinsic spectrum of the source, so *xflux* must be used in conjunction with *fit* . *Fit* generates a table of the best fit intrinsic spectrum (*_int.tab* extension). *xflux* uses this table to calculate either the integrated flux between two energy limits or the flux density at a specific energy. In order to calculate the flux of a source with only a few counts, or for *Einstein* or ROSAT HRI data, *fit* must be run with *rebin=yes* and the spectral model given as input leaving only the normalization as a free parameter (see §7.7.7). For IPC or PSPC sources with a sufficient number of counts the task *fit* should be run as before.

The distance required to calculate the luminosity can be given in kpc or redshift z. In order to compute the luminosity correctly using *xflux* , the SAME distance must be input to both *fit* and *xflux* . Note that the only way to input a distance/z to *fit* is to also specify an intrinsic absorbing column; it should be negligible (~10**17 cm^{-2}) if such a column is not otherwise required.

To compute the integrated flux:

1. First run *qpspec* (see §7.6 for details) to obtain counts from desired region. This produces "..._obs.tab" file.

2. Then run *fit* (with *rebin=yes* for low count or HRI data). This is where you specify the spectral shape: e.g. abs(21.51)*pow(0.85); see Section 7.7 for details. The output file is *..._int.tab*, which contains the amplitude ("normalization") corresponding to the number of net counts gathered by *qpspec* .

3. Finally, run *xflux* , which will give you flux (or flux density) for a user-specified energy band from the model spectrum.

```
xs> xflux
spectrum intermediate file:  rp90
energy or range of energies in keV:  0.1:2.4     # flux between 0.1 and 2.4 keV
flux between 0.1 and 2.4 keV
distance e.g., 5 kpc or 1.0z:  1.0 kpc              # this distance must be the
                                                    # same as that used in fit.
```

```
Output file:  rp90_flux.tab
```

To compute the flux density at 1 keV:

```
xs>  xflux

spectrum intermediate file (rp90):
energy or range of energies in keV (0.1:2.4):  1.0
distance e.g., 5 kpc or 1.0z (1.0kpc):
```

```
Output file:  rp90_fd.tab
```

Detailed spectral analysis is not possible for sources observed with the ROSAT HRI, because of the detector's poor energy resolution. However, fluxes for HRI sources may still be calculated, provided one assumes a spectral shape. The task hxflux executes qpspec, fit, and xflux, as described above, with all spectral parameters except normalization held fixed at values specified by the user.

```
   input files:   rh0110267.qp

   output files:  rh110267_flux.tab or rh110267_fd.tab

xs> hxflux

source qpoe file (xdata$rh110267.qp): xdata$rh110267.qp
source region descriptor (c 2322 4408 60):
background qpoe file (xdata$rh110267.qp):
bkgd region descriptor (a 2322 4408 120 240):
root name for output files [root_obs.tab, etc.] (rh110267):
spectral model, e.g. abs(logNH)*pow(alpha) (abs(19.0)*ray(1.0 cosmic 100)):
energy or range of energies in keV (0.1:2.4):
distance or redshift e.g. 5kpc or 1.0z (0.047):

ds:                     0
file:                   rh110267_obs.tab
scale:                  0.00
mission:                20
seqno:                  CA110267.N2
instrument:             ROSAT_HRI
sub-instrument:         0
no. pha channels:       1
mean off-axis angle
   of events (arcmin):  28.84
radius (arcmin):        0.50
filter:                 0
livetime (sec):         2153.14
x, y:                   2322.00, 4408.00
livetime corr:          0.99
src area (sq. pix.):    11277.00
cts_tot (count):        1993.00
bkgd area (sq. pix.):   135692.00
```

```
ccts_bkg (count):       16.21
net (count):            1976.79
neterr (count):         45.67
Output table file(s):  rh110267_obs.tab, rh110267_soh.tab, rh110267_boh.tab
Performing the Simplex minimization fit.
Found 1 free parameter(s).
Found 1 dataset(s).

 Data set #1 from file: rh110267_obs.tab

 PHA energy range   observed     error   predicted (pred-obs)/error
 --- ------------   --------     -----   --------- ----------------
   1  0.01->2.57      1976.8 *    45.7      1976.8        0.0
               * indicates use in Chi-square calculation.

The fitted net counts are: 1976.79; with error: 45.67.

Chi-square =      0.000
Predicted Data file: dummy_prd.tab
Intermediate Spectra file: rh110267_int.tab

Model: abs(19.000)*ray(1.000 cosmic 100.000)

Model Component 1: Raymond Thermal
          Cosmic abundance at 100 percent.
          temperature = 1.000 (fixed)
 normalization (log) = 12.2580032 (calculated)
    galactic Nh (log) = 19.000 (fixed)
    intrinsic Nh (log) = 0.000 (fixed)
          redshift = 0.000 (fixed)


Output file: rh110267_flux.tab
Model: abs(19.0000)*ray(12.2580 1.0000 cosmic 100)
Absorption type: Morrison & Maccammon
Units:
        flux (ergs/cm**2/s)
```

```
        luminosity (10**34 ergs/s)
        energy (keV)
        H_0 (km/s/Mpc)

Compute flux and luminosity for energy range :

  flux    unabs_f      L      lo_energy  hi_energy    D_kpc    z    H_0   q_0
  ----    -------     ---     ---------  ---------    -----   ---   ---   ---
3.52E-11  3.73E-11   9.86E-4     0.10       2.40     4.70E-2 INDEF  50.0  0.00
```

7.12 * upspecrdf , downspecrdf * – Transform between RDF and pre-RDF spectral tables

With the advent of RDF, the output from *qpspec* (for most instruments) has been divided into three tables:

_obs.tab	counts
_soh.tab	source off-axis histogram
_boh.tab	background off-axis histogram

Tasks *downspecrdf*, and *upspecrdf* translate between RDF and pre-RDF formats. *upspecrdf* will be useful to re-analyze previously extracted spectral data; *downspecrdf* is currently necessary for use with *proscon* (version 2.2), the translation program for input into *xspec,* which does not work with RDF data.

input files: *rp90_obs.tab, rp90_soh.tab, rp90_boh.tab*

output file: *rp90_prdf_obs.tab*

```
 xs> downspecrdf  rp90_obs
```

7.13 *xspec* – External Fitting Program

The spectral analysis system *xspec* , which is part of the XANADU software originally written for analysis of EXOSAT data, provides a larger variety of spectral models than those supported in PROS. PROS format spectral data can be input into *xspec* using the XANADU program *proscon* .

Users should note that for spectral data generated in PROS 2.3 or later, the PROS program *downspecrdf* must be run to generate the input file to *proscon* (see §7.12). Also, the current version of *proscon* (v. 2.3) requires the full 256-channel pulse height spectrum for the PSPC.

Thus, in IRAF,

```
xs> qpspec full=yes display=0     # generate full 256 channels and
  # suppress printout of spectrum
source qpoe file: xdata$rp110590n00
source region descriptor: c 6940 9390 240
background qpoe file: xdata$rp110590n00
bkgd region descriptor: a 6930 9390 300 600
root name for output files [root_obs.tab, etc.]: rp90
xs>

xs> downspecrdf display=0
list of root names for files to convert [root_obs.tab]: rp90
Output (pre-RDF) table file:  rp90_prdf_obs.tab
xs>
```

and at the operating system (OS) level,

```
OS-prompt> proscon
 PROSCON version 2.3
Input file : rp90_prdf_obs.tab
OS-prompt>
```

Output of *proscon* will be the pha and rsp files rp90_prdf_obs.pha and rp90_prdf_obs.rsp, which can be directly input into *xspec* .

Both *proscon* and *xspec* are provided and supported by the HEASARC at GSFC. For more information, contact *xanprob@legacy.gsfc.nasa.gov*. Software may be obtained via anonymous ftp to *legacy.gsfc.nasa.gov*, in the subdirectory */software/xanadu*.

HEASARC provides a set of IRAF tools for translating PROS spectral data into *xspec* format. The task *stw2pha* in the *caltools* package of *ftools* v. 2.9 will convert FITS versions (generated with the STSDAS task *stwfits*) of the _obs.tab, _soh.tab, and _boh.tab *qpsec* output files into an OGIP standard FITS file, which can be read by *xspec* . Using this pha FITS file, the task *pcarf* in the *rosat* package will create the position-dependant part of the response. Which then can be used in conjunction with the detector's calibration response matrix in *xspec* .

7.14 *qpspec* – Time Resolved Spectral Fitting

For a source with a large number of counts in a long observation it is sometimes useful to study its spectrum as a function of time. This is done by extracting the spectrum within specified time intervals. To determine the time intervals available run *timfilter* (see Section 8.4):

```
xt> timfilter xdata$rp110590n00.qp
Ouput Time Filter list (): arlac

Reading from Qpoe file: xdata$rp110590n00.qp
Writing to Time Filter list: arlac.lis

        Good Time Intervals

start           end             duration

1606720.00      1607479.00      759.00

1611613.00      1612817.00      1204.00
```

qpspec can be run on the input qpoe file using times within these ranges

```
xs> qpspec
source qpoe file (xdata$rp110590n00.qp): xdata$rp110590n00.qp[time=(1606720.00:1607479.0
source region descriptor ():c 6947 9325 2'
background qpoe file ( ):
bkgd region descriptor (): a 6947 9325 3' 4'
root name for output files [root_obs.tab, etc.] (): arlac_1


ds:                       0
file:                     arlac_1_obs.tab
scale:                    0.00
mission:                  20
seqno:                    CA110590P.N6
instrument:               ROSAT_PSPC
sub-instrument:           1
no. pha channels:         34
mean off-axis angle
   of events (arcmin):    16.39
radius (arcmin):          2.00
filter:                   0
livetime (sec):           728.84
x, y:                     6947.00, 9325.00
livetime corr:            0.96
src area (sq. pix.):      180905.00
cts_tot (count):          54.00  129.00  264.00  324.00  315.00  495.00  329.00
                          252.00  198.00  108.00  76.00  82.00  105.00  108.00
                          153.00  244.00  259.00  273.00  312.00  259.00
                          249.00  193.00  128.00  109.00  107.00  70.00  62.00
                          35.00  22.00  16.00  9.00  5.00  6.00  0.00
bkgd area (sq. pix.):     316676.00
ccts_bkg (count):         0.22  0.00  0.66  0.44  0.22  0.66  0.88  0.66  0.66
                          0.22  0.44  1.10  0.44  0.44  0.44  0.66  0.22  0.22
                          1.10  1.10  0.22  0.66  0.44  0.66  0.44  0.22  0.00
                          0.44  0.44  0.00  0.00  0.44  0.00  0.00
net (count):              53.78  129.00  263.34  323.56  314.78  494.34  328.12
                          251.34  197.34  107.78  75.56  80.90  104.56  107.56
```

```
                        152.56  243.34  258.78  272.78  310.90  257.90
                        248.78  192.34  127.56  108.34  106.56  69.78  62.00
                        34.56  21.56  16.00  9.00  4.56  6.00  0.00
neterr (count):         8.41  12.40  17.28  19.03  18.78  23.27  19.17  16.91
                        15.11  11.44  9.78  10.12  11.30  11.44  13.41  16.66
                        17.12  17.55  18.70  17.13  16.81  14.93  12.36
                        11.49  11.40  9.43  8.93  7.00  5.80  5.11  4.14
                        3.45  3.62  1.91
Output table file(s):  arlac_1_obs.tab, arlac_1_soh.tab, arlac_1_boh.tab
xs>
```

8 Timing Analysis

Photons detected with the Einstein IPC and HRI were time-tagged to a precision of 63 μsec and 8 μsec, respectively. Photons detected with the ROSAT PSPC and HRI are time tagged to precision of 61 μsec. The *xtiming* package contains tasks which make light curves and search for periodicity.

The input for most timing tasks is a time-sorted *.qp* file, which is made from a QPOE file by the task *timsort*. Region masking is usually applied to select only portions of the image, in which case, sorting will occur faster. Outputs of the other tasks are typically ST tables. A one line description of available tasks is given below and can be obtained by typing 'help xtiming'.

Trial data sets can be found in the files *xdata$snr.qp* and *rp110590n00.qp*, and examples using these are given below. The file *rp110590n00.qp* is the ROSAT calibration image of the RS CVn star AR Lac. Although not a periodic source, the spacecraft wobble causes the image of the source to be periodically eclipsed by the detector window-support mesh and there is a period of ≈100 sec in the resultant signal.

8.1 Summary of tasks in this section

Name	Function	Packages Needed
apply_bary	apply barycentric corrections to photon arrival times	xray:xtiming:timcor
calc_bary	compute barycentric corrections to photon arrival times	xray:xtiming:timcor
chiplot	plot chisquared as a function of period	xray:xtiming
fft	fast fourier transform period search	xray:xtiming
fftplot	plot the Fourier coefficients	xray:xtiming
fldplot	plot the folded light curve	xray:xtiming
fold	make a folded light curve	xray:xtiming
ftpplot	plot the distribution of Fourier coefficients	xray:xtiming
ksplot	plots result of K-S variability test	xray:xtiming
ltcplot	plot the light curve	xray:xtiming
ltcurv	makes table of counts vs. time	xray:xtiming
period	period search by folding	xray:xtiming
qpphase	calculate photon phase for assumed period	xray:xtiming
timfilter	list start and stop times of good data intervals	xray:xtiming
timplot	plot light curves or other tabular information	xray:xtiming
timprint	list tables on screen	xray:xtiming
timsort	extraction of time sorted event	xray:xtiming
vartst	variability test	xray:xtiming

8.2 An example of a typical timing analysis

As an example we use the Einstein Observation of CTB109, a supernova remnant with bright internal pulsar. Data are found in the PROS package directory "xdata". First the source events are extracted from the rest of the field using a circular region centered on the source. Events will be time ordered in the output file.

```
xt> timsort xdata$snr psr ``c 503 513 24'' none
```

A light curve is then generated and plotted so we can inspect the gross properties of the sample (the sequence of good-data intervals and gaps) and any source variability.

```
xt>  ltcurv psr none psr 500

xt>  ltcplot psr_ltc
```

Finding this sample satisfactory, we search for periodicity with a fast Fourier transform. We use 2^{14} bins so the search will cover periods ≥ 1 s in this 7700 s data interval.

```
xt>  fft psr_sti.qp none psr src 16384
```

The result is viewed by plotting power as a function of frequency and by listing the table containing significant coefficients.

```
xt>  fftplot.y_range = "0 20"

xt>  fftplot psr_fft

xt>  timprint psr_pwr 1-20
```

We find significant power at low frequency because of the gaps in the data record and, at a frequency of 0.2868^{-1}, the pulsar. The period can be verified and measured more precisely by epoch folding over a range of trial periods.

```
xt>  period psr none psr 3.45 3.50 0.0003 5
```

The result can be plotted and listed,

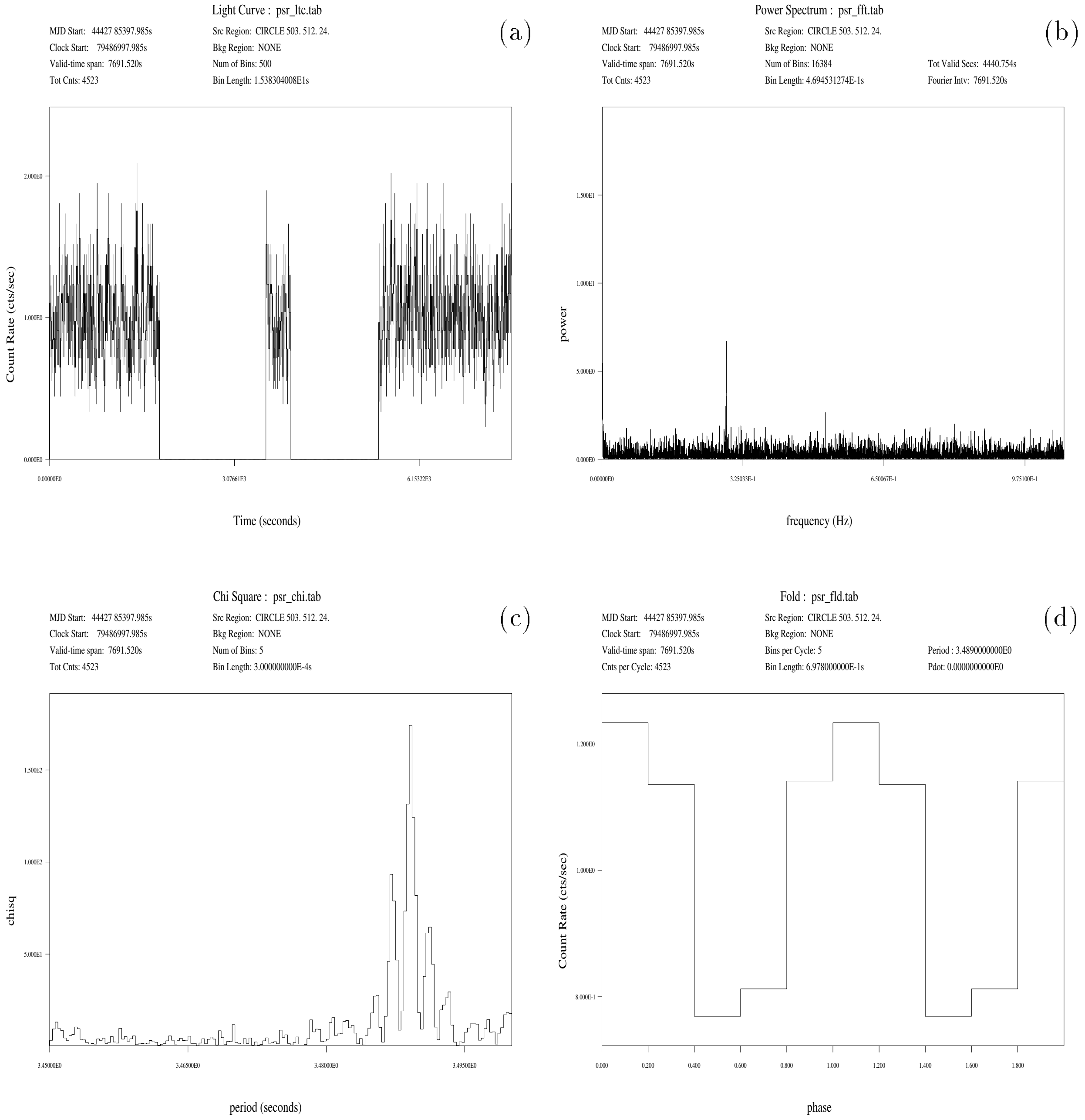
```
xt>  chiplot psr_chi

xt>  timprint psr_chi  110-150
```

and shows the period to be 3.4890 s (freq=0.2866). A folded light curve for the most promising period is also generated and can be viewed.

```
xt>  fldplot psr_fld
```

We are finished. We have the period and pulse shape. This example uses a strong source with rather slow period (the real period is actually 6.978 s, but the largest coefficient is that of the 2nd harmonic). In practice it is usually necessary to use barycenter-corrected data and much longer Fourier transforms. The following sections give more detail for each task.

Graphics from timing-analysis example:



8.3 *timsort*

8.3.1 Extraction of time-sorted events

Restrict data to a region at the position 6940, 9390, with radius of 240 pixels, and make a file of time-sorted events.

input file: *rp110590n00.qp*

output file: *rp90_sti.qp*

```
cfa#: cl                                              # enter iraf environment
cl> xray                                        # Load x-ray analysis programs
xr> xtiming                                     # Load timing analysis programs
xt> timsort
input qpoe file name: xdata$rp110590n00               # The original *.qp file
output time ordered filename: rp90         # will assign the name rp90_sti.qp
source region descriptor: c 6940 9390 240    # Circle centered at 6940 9390,
                                                 # with radius of 240 pixels
output time ordered bkgd filename: <cr>                         # No bkg data
```

8.3.2 Extraction of limited time intervals

```
xt> timsort

Input QPOE file name: xdata$rp110590n00.qp[time=(1611613:1612817)]
Output time-ordered file name:  rp90_sect2
```

An alternate approach which saves the typing of long numbers is to choose the appropriate interval by editing the file rp90_tim.lis output with *timfilter* in Section 8.4 and specifying the input file as

xdata$rp110590n00.qp[@rp90_tim.lis]

8.4 *timfilter* – List start & stop times of good data intervals

The data example used above has two intervals of good data separted by a long gap. Before doing further time analysis you may want to know the boundaries of these intervals.

input file: *rp90_sti.qp*

output file: *rp90_tim.lis*

```
xt> timfilter
Input QPOE file:  rp90_sti          # the default extention, .qp, is known
Output Time Filter List:  rp90_tim
```

Start and stop times of data intervals are displayed on screen and written to the file rp90_tim.lis. To use multiple intervals in subsequent filtering run *timfilter* with parameter merge=yes. To use one interval, run the merge=no and uncomment.

Type *help timfilter* for instructions.

8.5 Counting rate vs. time

8.5.1 *ltcurv* – Generation of the light curve

Produce a light curve for the source at 6940 9390, using timesorted QPOE file made in Section 8.3.1.

input file: *rp90_sti.qp*

output file: *rp90_ltc.tab*

```
xt> ltcurv                                          # In the timing package
Input Source Timing File:  rp90              # Only the root name is needed,
                                            # the default extension is known
Input Background Timing File (NONE): <cr>                     # No bkg data
rootname for output file:  rp90 or .       # ``.'' will assign same root name
```

```
Number of Bins:  500                                              # 500 data bins
xt>
```

8.5.2 *ltcplot* – Plot the light curve

Plot the light curve file made in Section 8.5.1.

input file: *rp90_ltc.tab*

```
xt> ltcplot
Input Table File:  rp90_ltc
```

Plot appears with crosshair cursor and the program will be in cursor mode for interactive readjusting of the plot. Type = for hardcopy. Type q to leave plot and return to PROS.

Note: You may want to adjust the following parameters:

ecolumn = "err"	to plot error bars
y_range = "0 12"	will set the vertical scale to these values.
	"auto" will scale from min y to max y.
num_plots = 2	

8.6 *timprint* – Inspect tables on screen

Print the light curve file made in Section8.5.1 .

input file: *rp90_ltc.tab*

```
xt> timprint

Table file name:  rp90_ltc
Range of rows to print (-): <cr>              # will show header and all rows
                            or 1-20      # will show header and only rows 1-20
```

If data overflows screen, ctrl s will stop and ctrl q will start again.

```
xt> timprint | page      # prevents data from overflowing screen in suntools
xt> timprint | lpr                                          # for hard copy
```

8.7 *calc_bary, apply_bary* – Barycentric Time Corrections

Precise timing analysis requires the conversion of input QPOE event times from raw spacecraft clock units to UTC, and the correction of photon arrival times at the satellite to times at the solar system barycenter. The satellite ephemeris is a necessary input.

```
xt> timcor
ti> help using_timcor             # gives general instructions for obtaining
                                       # _ephem tables and using these tasks
```

8.7.1 *calc_bary* – Computing Barycentric Corrections

The satellite ephemeris is used to compute arrival time corrections from the satellite to the solar system barycenter, appropriate to an input celestial position. If the ephemeris data file is in Rev. 0 format, it is converted to RDF format.

input files: *rp110590_eph.tab (Rev. 0) or rp110590n00_eph.tab (RDF)*

output files: *rp110590n00_cor.tab, rp110590n00_ephem.tab*

```
ti> calc_bary
input ephemeris table : xdata$rp110590_eph.tab              # A Rev. 0 file
root name for output file(s) [root_cor(ephem).tab] : .
input RA (hh:mm:ss.s) : 22:08:41.6
input DEC (dd:mm:ss.s) : 45:44:47.3

Converting Rev0 US format EPH file to RDF format ...
Opening file: xdata$rp110590_eph.tab
```

```
40 rows written to output table.
Output file: ./rp110590_ephem.tab
Output ephemeris table: ./rp110590_ephem.tab

Computing Correction table ...
Opening file: ./rp110590_ephem.tab
Output Correction table: ./rp110590_cor.tab
```

Alternatively, for RDF input

```
ti> calc_bary
input ephemeris table: xdata$rp110590n00_eph.tab                # an RDF file
root name for output file(s) [root_cor(ephem).tab]: .
input RA (hh:mm:ss.s): 22:08:41.6
input DEC (dd:mm:ss.s): 45:44:47.3

RDF input ... No conversion!!
Output ephemeris table: ./rp110590n00_ephem.tab

Computing Correction table ...
Opening file: ./rp110590n00_ephem.tab
Output Correction table: ./rp110590n00_cor.tab
```

8.7.2 *apply_bary* – Applying Barycentric Corrections to QPOE times

The barycentric corrections computed in 8.7.1 are applied to the event times in a time-sorted QPOE file. In addition, the times are converted from spacecraft-clock units to UTC.

input files: *rp90_sti.qp, rp110590n00_cor.tab*

output files: *rp90_out.qp*

```
ti> apply_bary
```

```
input qpoe file name: rp90_sti.qp[time=(1606720.0:1612809.8)]    # see Note 1&2
input correction table filename: rp110590_cor.tab
output qpoe file name: rp90_out
event definition: medium

SCC->UTC conversion table is calibrated to SCC 110883482.0000

first qpoe gti time   = J.D. 2448062.97604379,  SCC 1606720.00000000
last qpoe gti time    = J.D. 2448063.04652758,  SCC 1612809.80000000
first corr table time = J.D. 2448062.97569444
last corr table time  = J.D. 2448063.04652778

Barycenter Corrections applied to event times in rp90_out.qp
using correction table rp110590_cor.tab
with reference position at RA =  22:08:41.60, DEC =  45:44:47.30
Creating QPOE output file: rp90_out.qp
```

NOTE: The output QPOE times are in seconds from noon of the MJD of the start of the first OBI.

To convert the times to MJD use the following formula:

$$MJDREFI + MJDREFF + (photon\_time / 86400)$$

The values MJDREFI and MJDREFF can be found in the header of the output QPOE file. (MJDREFF should be 0.5)

NOTE 1: The orbit data records in the root_eph.tab file are generated every 60 seconds during the observation. The apply_bary task calculates the barycenter timing corrections using a linear interpolation between the two orbit records for which an event time falls. Occasionally there is not a final orbit record, so that there may be events with arrival times occurring after the last orbit record. When calculating the barycenter timing corrections for these events, the apply_bary task uses a linear extrapolation from the previous two orbit records and gives the user a warning that an extrapolation is being used. This extrapolation should effect at most 60 seconds of data. The apply_bary task prints the times of the first and last event and the first and last orbit record, so that the user may assess the appropriateness of the extrapolation being performed.

For this dataset, there are 62 events that arrive up to ~7 seconds after the last orbit record. The example above uses a time filter on the input qpoe file to filter out these events. The user may also use the qpcopy task to perform the filtering.

NOTE 2: The apply_bary task takes as input a time-sorted qpoe file, and allows the user to specify a region descriptor. There is a known bug when using large region descriptors with time-sorted qpoe files. The error usually manifests itself in the form of "ERROR: PLIO: reference out of bounds on mask" or "ERROR: PLIO: stack overflow (plr_open)". Since the standard qpoe file is spatially sorted, the user is encouraged to apply region descriptors when running the timsort task (as in example 8.3.1).

8.7.3 *scc_to_utc* – Converting Spacecraft time to UTC

QPOE event times are typically expressed in units of an on-board spacecraft clock. Because of clock drifts and resets, the relationship between spacecraft time and UTC may be non-linear, and may even vary with time. For Einstein and ROSAT, the relationship is expressed in terms of interpolating polynomials determined through in-flight calibration. The conversion is automatically performed as part of the timing corrections in 8.7.2. The stand-alone task scc_to_utc allows conversion of an arbitrary spacecraft time to UTC.

input files: *xtimingdata$scc_to_utc.tab*

```
  ti> scc_to_utc tbl_fname = xtimingdata$scc_to_utc.tab                  # ROSAT
                                                             # conversion table

input spacecraft clock time (1606720.): 1606720.0

SCC->UTC conversion table is calibrated to SCC 110883482.0000

SCC time = 1606720.00000000
UTC time (integer part) 2448062  (real part) 0.97604379056608

  ti>
```

8.8 *fft* – Period search using Fast Fourier Transform

8.8.1 A "short" FFT

Small numbers ($\leq 2^{14}$) of time bins may be analyzed directly from the output of ltcurv, Section8.5.1 .

input file: *rp90_ltc.tab*

output files: *rp90_fft.tab, rp90_ftp.tab, rp90_pwr.tab*

```
xt> fft
Input source file (table or timsorted qpoe file): rp90_ltc
root name for output file [root_fft(ftp,pwr).tab]: .  # will assign root name
                                                        or rp90

Input table column heading:  src    # label of column giving number of counts

FFT done with 512 bins!
Creating raw fft output file: rp90_fft.tab                            # columns:
                                         # freq, rfreq, ifreq, power, normpower
Creating histogram output file: rp90_ftp.tab         # columns: logel,  phist
Creating fft power output file: rp90_pwr.tab                          # columns:
                                                  # bin, freq, power, normpower
xt>
```

To list only significant coefficients, run *timprint* as in §8.6 with the *_pwr.tab* file as input.

8.8.2 *fftplot* – Plot power vs. frequency

```
xt> fftplot

input table file: rp90_fft
```

A plot of Fourier coefficient value vs. frequency will appear on screen. To make hard copy, type = with cursor within plot, to return cursor to command window, type q.

Note: The plot is set to autoscale and will display the largest coefficient. To better display smaller coefficients, change the parameter file to specify exactly the limits of the y axis, e.g., from 0 to 2 (assuming you are plotting "normpower" for a large number of bins).

fftplot y_range = "0 2"

8.8.3 A "long" FFT

Compute FFT from the input QPOE file from Section8.3.1 .

input file: *rp90_sti.qp*

output files: *rp90_ftp.tab, rp90_pwr.tab*

In searching for short periods using a long data interval many bins are necessary [(no. of bins) $= 2 \times (data\,interval)(min\,period)^{-1}$] and generation of a _ltc.tab file may prove impractical. In such cases, the QPOE file may be input. Only the largest Fourier coefficients, but not the complete list of (*e.g.*, 5×10^5) coefficients should be used, so we set power_thresh to exclude all but the largest from the table [root]_pwr.tab. The normalization has been adjusted so the value of the power coefficient, normpower, does not depend strongly on number of bins or number of counts. A significant signal will have a value of ~ 4 for a 10^3 bin transform and a value of ~ 8 for 10^5 bins. Time required for a 10^5 bin transform is 2–3 minutes.

```
xt> fft.fftcoeff=no             # don't generate a long list of coefficients
xt> fft.power_thresh=7                      # set the significance threshold

xt> fft

input source file (table or time-sorted qpoe file):  rp90_sti.qp
root name for output files [rrot_fft(ftp,pwr).tab]: rp90lng
background qpoe file: <cr>
input table column heading:  src
number of bins 2048 # The task uses the closest power of 2 to the input value
```

FFT done with 2048 bins!

```
creating histogram output file rp90_ftp.tab
creating fft power output file rp90_pwr.tab
```

8.8.4 A summed or incoherent FFT

The time span of some data sets is too large to perform a single fft to search for short periods. Some ROSAT observations contain gaps of days length, for example. The data can be divided into a set of individual intervals, all of the same length. The fft will transform each interval and sum the power coeff for all intervals. Gaps can be eliminated or minimized by clever selection of interval length. However, the shorter the interval the less sensitive the transform of the data set.

First make a file (e.g., *rp90_sum.lis*) which defines the .qp files of each data interval to be used as input to the transform,

e.g. rp90_sti.qp[time=(1606720.0:1607232.0)]
rp90_sti.qp[time=(1611613.0:1612125.0)]

Note all intervals are 512 sec long in this example. FFT is the only timing task that will take more than one .qp file as input.

Now run the FFT

```
st> fft
input source file (table or time_sorted qpoe file): @ rp90_sum.lis
                                                     # the ``@'' is important
rootname for output file [root_fft(ftp,pwr).tab]: rp90_sum
input bgnd qpoe file (name): <cr>
input table column heading (src): <cr>
number of bins: 0                                              # a power of 2
bin length: .025             # important to specify bin length when intervals
                                        # are not full of data (contain gaps)
```

The screen will report each transform as it is completed and the generation of the _fft, _pwr, _ftp output tables. Note that as long as all intervals do not contain gaps (regions of bad data) at start and/or end, fft may be run with number of bins specified. If there are gaps at start and/or end of any interval, the bin length must be specified.

8.8.5 *ftpplot* – Distribution of FFT coefficient values

The distribution of coefficient values for a single FFT is exponential if the original data contain no periodic signal. This distribution can be plotted and inspected. Significant coefficients will appear beyond the high-value end of the normal distributions. The width of the histogram bins can be adjusted (before running *fft*) to get a nice looking plot.

To plot the power density distribution created by *fft* in Section8.8.1.

```
xt> fft.histogram_bi=0.5  # if necessary, adjust number for an aesthetic plot

xt> ftpplot
input table file: rp90_ftp
```

The plot will appear on the screen. To make hard copy, type = with cursor within plot. To return to text window type q.

8.9 *period* – Period search using epoch folding

Search for periodicity using the timesorted output file of Section8.3.2 .

This task determines the period which produces the folded light curve with maximum deviation (i.e. chi sq) from that of a constant source. It outputs the folded light curve as a function of phase in rp90_sec2_fld.tab, and the run of chisquare vs. period in rp90_sect2_chi.tab.

The sample interval must be set short enough so the sampling is complete. To search for a period p seconds in an observation of length t seconds, sorting into n bins, the period interval, Δp, should be $\Delta p = p^2/nt$. If the user specifies a period increment of “0”, the computer will calculate this interval.

The parameter get_gintvs=yes requires that an exposure be calculated for each interval, important if the period is comparable to the duration of gaps and good-data intervals.

input file: *rp90_sect2_sti.qp*

output files: *rp90_sect2_fld.tab, rp90_sect2_chi.tab*

```
xt> period.pdot=4.22e-13    # enter value of period derivitive (units, s s-1)
                                                # if known.  Default is 0.
```

Note: For those sources with a period which changes linearly with time, enter dp/dt value (4.22e-13 is the value for the Crab pulsar).

```
xt> period
Input Source Timing File: rp90_sect2             # assumes extension _sti.qp
Input Background Timing File:  none
Root name for output files [root_fld.tab, root_chi.tab]:  rp90_sect2
                                             # Assigns name rp90_sect2_fld.tab
Period start:  90                                          # units are seconds
Period stop:  110
Increment (0 for auto increment):  0.1
Number of bins:  10       # trial periods will be divided into 10 phase bins.
```

Results may be viewed using

```
xt> chiplot rp90_sect2_chi                                 # command line mode
```

8.10 *fold* – Make a folded light curve

Run fold on the timesorted output of Section8.3.2 . This task computes the average or folded light curve for a single cycle of known period.

input file: *rp90_sect2_sti.qp*

output file: *rp90_sect2_fld.tab*

```
xt> fold
input source timing file:  rp90_sect2
input background timing file (none): <cr>
root name for outfile [root_fld.tab]: rp90_part2   # chosen so name of output
                                          # file different from that of Section 8.9.
fold period in seconds: 100.14                    # this value chosen by user.
number of bins:  10
```

The folded light curve may be displayed with the task *fldplot* .

```
xt> fldplot rp90_part2_fld
```

Note: For error bars on plot set the parameter ecolumn="err".

8.11 *vartst* – Test for Source Variability

This task applies two non-parametric statistical tests, the Kolmogorov-Smirnov and Cramer-Von Mises tests, to the cumulative distribution of photon arrival times (with data gaps removed), to test the hypothesis that the source rate is constant. Input file is a timesorted qp file. Results are listed on the screen and can be plotted with the task *ksplot* .

input file: *rp90_sti.qp*

output files: *rp90_iq1.cmd,rp90_iq2.cmd,rp90_var.tab*

```
xt> vartst

Input source timing file:  rp90
```

```
Root name for output files: rp90
Number of Events: 14625
```

Ks-text Thresh. : 90% = 0.010, 95% = 0.011, 99% = 0.013

Max diff = 0.02448, Variability detected with 99% confidence level

CvM Thresh. : 90% = 0.347, 95% = 0.461, 99% = 0.743

CvM = 3.57285, Variability detected with 99% confidence level

```
Creating igi Ksplot cmd file:  rp90_ig1.cmd
Creating igi Conf band cmd file: rp90_ig2.cmd
Creating Var file:  rp90_var.tab

xt> ksplot
Root name for input file(s): rp90             # plot appears in grapic window

xt> =gcur                                   # give control to graphics cursor
```

Type = for hard copy, q to return control to text window.

8.12 *qpphase* – Append phase to events in QPOE file

This task computes the phases of individual QPOE events for an assumed period P and rate of change of period Pdot according to the formula

$$PHI(t) = mod[t/P - 0.5 * Pdot * (t**2/P**2), 1.0].$$

The phases are appended to the individual event records by defining a new event structure for the output QPOE file.

input files: *rp90_out.qp*

output files: *rp90_ph.qp*

```
xt> qpphase
input qpoe file name: rp90_out
region descriptor: <CR>                                    # region selection
                                                              # already done
Nominal period in seconds: 100.14
Rate of change in period (0.): <CR>
output qpoe file name: rp90_ph
event definition ({s:x,s:y,s:pha,s:pi,d:time,d:phase,s:detx,s:dety}): <CR>

rp90_out.qp (; no exp.; period=100.140000; pdot=0.000000) -> rp90_ph.qp
Creating QPOE output file: rp90_ph.qp
```

Note, the phases are only valid for the period and Pdot chosen.

9 Glossary

- **rawx** raw detector x position (*i.e.*, before rotation and aspect correction)($\equiv$dx for Rev0)
- **rawy** raw detector y position (*i.e.*, before rotation and aspect correction)($\equiv$dy for Rev0)
- **detx** detector x position after correction
- **dety** detector y position after correction
- **pha** pulse height analyzer channel. Accepted values 0-15 (*Einstein* IPC), 1–256 (ROSAT PSPC)
- **pi** pulse invariant bin (energy). Accepted values 0-15 (*Einstein* IPC), 1–256 (ROSAT PSPC) and 1–34 (ROSAT PSPC, SASS output).
- **region** a selected region of the image of various forms. Used to select only a portion of the image, within which the current operations should be applied.
- **time** photon arrival time in sec.
  - from Jan. 0, 1978, for *Einstein* data
  - Spacecraft clock time (can be converted to UT) from 21:06:50.2 June 1, 1990 for ROSAT data
- **x - y** pixel position in sky pixel (*i.e.*, after aspect correction). Values start from 1 in the lower left corner

10 List of Related Documents

- * *iraf Tidbits*) [Belinda Wilkes] - general IRAF information, warnings about common pitfalls etc. Available from SAO.
- * *A User's Introduction to the iraf Command Language* [Peter Shames and Doug Tody] - General introduction to IRAF. Available from NOAO with the IRAF distribution.
- * *The IRAF Data Reduction and Analysis System.* [Doug Tody] - General description of the IRAF system and its capabilities. Available from NOAO.
- *An Introductory User's Guide to IRAF Scripts* [Ed Anderson] - Basic introduction to the writing of IRAF scripts *i.e.* macros. Available from NOAO.
- **CL Programmers Manual* [Downey, Tody and Jacoby] - Detailed programmers guide for the IRAF command language. Available from NOAO or within IRAF under *iraf$doc*Coordin.
- * *siteguide* [Belinda Wilkes and Wendy Roberts] - Site-dependent aspects of running IRAF on the SAO HEAD computer system, could also be used for similar configurations at other sites. Available from SAO.
- IPCSPECS - IPC data reduction specification (rev1B processing). Available from SAO. SAO special report # 393.
- *File Structure and Formats of Processed Einstein Data* [G. Fabbiano and J. McSweeney] - Details of the structure of the *Einstein* data files. [Vax format] Available from SAO.
- *Einstein Observatory Revised User Manual* [D. Harris] - User Manual for the Einstein Mission. Available from SAO.
- *The ROSAT Data Products Guide* [R. Downes, R. White, and G. Reichert] - Complete description of ROSAT data as distributed to users. Describes format and gives examples. Available from GSFC RSDC.
- *STSDAS User Guide* - Available from Science Software Branch STScI.
- *User Manual for SAOimage* [M. VanHilst] - Available from SAO.

*documents so indicated are also available as compressed postscript files in subdir /pub/pros/docs of SAO anonftp.

ftp sao-ftp.harvard.edu

11 List of PROS and Selected IRAF Tasks

This section lists all PROS tasks (some of which are not described in this guide) and some useful IRAF tasks which are included. It is our intent to add instructions for tasks that are particularly useful in x-ray analysis. A brief description of all tasks in a given package can be obtained by typing `help package-name` and a complete description of any task with `help task-name`.

Package	Task	Description	Section(s)
dataio	*t2d*	reads files from tape to disk	3.2.1
tables	*strfits*	converts FITS to images and/or tables	
	stwfits	converts images and/or tables to FITS	
	pltpar	display sgraph parameter file in editor	
	tprint	lists table on screen	
images	*hedit*	header editor	
	imcopy	copy an image	
	imdelete	delete a list of images	
	imheader	print an image header	4.3.1
	imlintran	linearly transform a list of 2-D images	6.7.4
	magnify	magnify a list of 1-D or 2-D images	6.7.4
images:tv	*display*	displays image on screen	5.3
	imexamine	examine images using image display, graphics, and text	6.2.6
	tvmark	mark objects on the image display	5.6
	wcslab	overlay a displayed image with a world coordinate grid	5.5.4
xray	*xapropos*	list all tasks related to a given subject	
xray:xdataio	*datarep*	convert binary number representations to host computer format	
	efits2qp	convert an Einstein FITS file into an IRAF/PROS QPOE file	
	fits2qp	convert from BINTABLE FITS format to PROS QPOE format	3.6.2
	hkfilter	generate a QPOE time filter string from an input housekeeping filter	
	mkhkscr	generate a housekeeping (TSI) filter	
	mperfits	convert from MPE/MIDAS FITS IMAGE to IRAF image format w/WCS	3.2.2

Package	Task	Description	Section(s)
xray:xdataio	*qp2fits*	Convert from PROS QPOE format to BINTABLE FITS Format	3.6.2
	qpaddaux	add additional BINTABLEs to existing QPOE file	
	qpgapmap	calculate the de-gapped HRI coordinates (detx,dety)	
	rarc2pros	Convert ROSAT archive files to PROS format	3.3.3
	rfits2pros	Convert ROSAT PI data tapes to PROS format	3.3.1
	upimgrdf	convert a list of IRAF/PROS IMAGE files into RDF compatible format	
	upqpoerdf	convert a list of IRAF/PROS QPOE files into RDF compatible format	
	xrfits	Read FITS data and convert to IRAF IMAGE or TABLE files	3.2.2
	xwfits	Writes a FITS output file from IRAF IMAGE or TABLE files	3.6
xray:xdataio:eincdrom	*ecd2pros*	copy or convert FITS files from the Einstein CD-ROMs	
	ecdinfo	display information about a specific Einstein dataset	
	eincdpar	edit the eincdrom package wide parameters	
	eindatademo	demonstration of Einstein IPC unscreened data	
xray:xdemo	*gspectral*	demonstration of spectral functions, with line graphics	
	gtiming	demonstration of timing functions, with line graphics	
	spatialdemo	demonstration of spatial functions, with line image display	
	spectraldemo	demonstration of spectral functions, with image display	
	timingdemo	demonstration of timing functions, with image display	

Package	Task	Description	Section(s)
xray:ximages	*imcalc*	Arithmetic operations on image files	6.7.3
	imcreate	Create an image from a binary file or a constant value	
	imcompress	Compress an IRAF image by pixel summation	
	imnode	Remove node information from IRAF image pixfile pathname	
	imreplicate	Expand an IRAF image by pixel replication	
	plcreate	create a pixel list from a region	6.2.1
	pllist	display a pixel list	6.2.2
	qpcopy	copy a qpoe file to another qpoe file	4.2
	qplintran	Apply linear transformation to a QPOE file	4.7
	qplist	list individual photon records in regions and aux. records	4.3
	qprotate	rotate a qpoe file outputting another qpoe file	4.6
	qpshift	Apply linear shift to a QPOE file	4.5
	qpsort	sort a qpoe file	4.4
	qphedit	edit header parameters in qpoe or image file	2.3.5
	xhadd	add x-ray history record to an image or qpoe	
	xhdisp	display x-ray history records	
xray:xplot	*imcontour*	display a contour map of an image using wcs	6.4.3
	pspc_hrcolor	Generate an x-ray color image	5.10
	tabplot	Plot tables, lists or image sections	5.9
	tvimcontour	load an image and overlay a contour map into the display window	5.5
	tvlabel	display an image with a wcs cursor readback in hh:mm:ss, dd:mm:ss	5.4.1
	tvproj	display an image with x and y projections	5.7
	xdisplay	load an xray image into the display window	5.3
	ximtool	invoke SAOIMAGE for use with IRAF/XDISPLAY	??
	xexamine	examine a qpoe using the image display and wcs coordinates	5.3.10
xray:xspatial	*errcreate*	generate a PROS error image from data image	
	fixsaoreg	convert an SAOimage region file to PROS physical coords	5.2.5
	imcnts	count photons (w/bkgd subtraction) in regions	6.2.8,6.3
	imdisp	display pixel values in regions	
	improj	perform rectangular projections on an image	6.3.5
	immodel	create a model array from a source list	6.8.1

Package	Task	Description	Section(s)
xray:xspatial	*imsmooth*	smooth an image	6.4.1
	isoreg	create region mask by applying intensity levels to image data	6.2.3
	makevig	generate a vignetting mask file from calibration data	6.5.1.1
	rosprf	Build ROSAT point response function	6.8.2
	skypix	process and convert between astronomical coordiante systems	5.4.2
	vigdata	apply vignetting correction to image data	6.5.1.1
	vigmodel	apply vignetting effect to an image model	
	wcscoords	use WCS to convert between IRAF coordinate systems (logical, physical)	
xray:xspatial.detect	*bepos*	determine the best source position from a rough position list	6.9
	bkden	determine the background density of a given region	6.9
	cellmap	block a qpoe file at detect cell size resolution	6.9
	detmkreg	make a region file in logical coords from detect tables	6.9
	lbmap	makes a local background map	6.9
	ldetect	'canned' local detect task	6.9
	lmatchsrc	Match sources between/within different cell sizes	6.9
	lpeaks	determine the rough source positions from a snrmap	6.9
	snrmap	determine the signal-to-noise of a given image	6.9
xray:xspectral	*bal_plot*	plot the current BAL histogram (Einstein IPC data only)	
	counts_plot	plot the observed and/or predicted count distributions	7.8
	dofitplot	do fit and then counts_plot	7.2
	fit	find best-fit model values using simplex minimization	7.7
	grid_plot	plot the chisquared grid with contour lines	7.10
	hxflux	compute x-ray flux from ROSAT HRI data	7.11
	intrinsicspecplot	plot the emitted and observed spectrum	
	pkgpars	spectral package parameter pset	7.5
	qpspec	create spectral data set from a qpoe file	7.6
	search_grid	perform a chisquared grid search	7.9
	show_models	display the current model parameters	
	singlefit	compare fixed-parameter model with data	
	xflux	compute x-ray flux from models	7.11

Package	Task	Description	Section(s)
xray:xtiming	*chiplot*	plot a chi-squre table (output of period)	8.9
	fft	compute Fast Fourier Transform on table or qpoe data	8.8
	fftplot	plot a fft table (output of fft)	8.8.2
	fldplot	plot a fld table (output of fold)	8.10
	fold	fold time data on a period and compute statistics	8.10
	ftpplot	plot a ftp table (output of fft)	8.8.5
	ksplot	Plots result of K-S variability test	8.11
	ltcplot	plot a ltc table (output of ltcurv)	8.5
	ltcurv	compute count rates of binned source and bkgd	8.5
	period	determine the period using epoch folding	8.9
	qpphase	Calculate photon phase for a QPOE	
	timfilter	write qpoe time interval filters to an ascii file	8.4
	timplot	strip plot of table format data	
	timprint	display timing table data	8.6
	timsort	create a time-sorted qpoe file for timing analysis	8.3
	vartst	Variability test	8.11
xray.xtiming.timcor	*apply_bary*	apply the barycenter time corrections to the time-sorted qpoe file	8.7.2
	calc_bary	calculate the barycenter time corrections	8.7.1
	scc_to_utc	convert from ROSAT space-craft clock to UTC	8.7.3

12 DETAILED LIST OF ROSAT FITS FILES

12.1 Rev0, US-Processed Data

Prior to June 1994 data processed in the U.S. were supplied in the following files:

PSPC

rp<seq#>.fits	Photon events, data quality, and good time intervals (binary table)
rp<seq#>_im1.fits	Compressed image for display — total energy band
rp<seq#>_im2.fits	Compressed image for display — hard energy band
rp<seq#>_im3.fits	Compressed image for display — soft energy band
rp<seq#>_bk1.fits	Background map — total energy band (image)
rp<seq#>_bk2.fits	Background map — hard energy band (image)
rp<seq#>_bk3.fits	Background map — soft energy band (image)
rp<seq#>_mex.fits	Exposure map with vignetting, ribs, and wobble (image)
rp<seq#>_sp.fits	Selected source spectra (ascii table)
rp<seq#>_sky.fits	Sky catalog identifications (ascii table)
rp<seq#>_src.fits	Detected source list (ascii table)
rp<seq#>_so.fits*	Satellite orbit information by OBI
rp<seq#>_evr.fits*	Events rate
rp<seq#>_asc.fits*	ASCII data corresponding to the hard copy outputs
rp<seq#>_hkb.fits*	Binned housekeeping files for the instrument condition
rp<seq#>_sa.fits*	Aspect files
rp<seq#>_asp.fits*	Aspect quality flags
rp<seq#>_par.fits*	Parameter sets used to create standard output(data processing)

HRI

rh<seq#>.fits	Photon events, data quality, and good time intervals (binary table)
rh<seq#>_img.fits	Compressed image for display
rh<seq#>_bkg.fits	Background map (image)
rh<seq#>_sky.fits	Sky catalog cross-identifications (ascii table)
rh<seq#>_src.fits	Detected source list and analysis (ascii table)
rh<seq#>_so.fits*	Satellite orbit information by OBI
rh<seq#>_ah.fits*	HRI Aspect Histogram
rh<seq#>_evr.fits*	Event rates
rh<seq#>_asc.fits*	ASCII files produced by SASS
rh<seq#>_hkb.fits*	Binned housekeeping files for the instrument condition
rh<seq#>_sa.fits*	Aspect files
rh<seq#>_asp.fits*	Aspect quality flags
rh<seq#>_evr.fits*	Event rates
rh<seq#>_sa.fits*	Aspect files

*These files are not PROS compatible. They are SASS level 1 output files (see Data Products Guide).

12.2 Rev0, MPE-Processed Data

Prior to June 1994, MPE-processed data were supplied in the following files:

PSPC

wp<seq#>_eventrates.tfit	Event rates
wp<seq#>_events.tfits	Photon event list
wp<seq#>_image1.ifits	Broad band image
wp<seq#>_image2.ifits	Hard band image
wp<seq#>_image3.ifits	Soft band image
wp<seq#>_mexmap.ifits	Exposure map (also seen as wp<seq#>_exposure.ifits)
wp<seq#>_imageec.ifits	Energy-coded image
wp<seq#>_orbit.tfits	Orbit information
wp<seq#>_quality.tfits	Data Quality
wp<seq#>_difevents.tfits	Rejected events
wp<seq#>_drmpspc.tfits	Response matrix
wp<seq#>_moimp.ifits	Instrument map
wp<seq#>_effarea_pspcb.tfits	Off-axis area, pspc B
wp<seq#>_effarea_pspcc.tfits	Off-axis area, pspc C
wp<seq#>_attitude.tfits	Attitude file

HRC

wh<seq#>_attitude.fits	Attitude
wh<seq#>_difevents.fits	Differential Events (rejected events)
wh<seq#>_eventrates.fits	Event rates
wh<seq#>_events.fits	Photon event list
wh<seq#>_orbit.fits	Orbit information
wh<seq#>_quality.fits	Data Quality
wh<seq#>_simbad.fits	SIMBAD source list

N.B. The file wp<seq#> file present in German format archived data should be deleted before running *rarc2pros*.

12.3 RDF Format

Data processed in US or at MPE after June 1994:

```
PSPC RDF tape list

tape002 -> rp110590n00_bas.fits      Photon event and good times
tape003 -> rp110590n00_im1.fits      Broad band image
tape004 -> rp110590n00_im2.fits      Hard band image
tape005 -> rp110590n00_im3.fits      Soft band image
tape006 -> rp110590n00_bk1.fits      Broad background image
tape007 -> rp110590n00_bk2.fits      Hard background image
tape008 -> rp110590n00_bk3.fits      Soft background image
tape009 -> rp110590n00_mex.fits      Exposure map
tape010 -> rp110590n00_ime.fits      Energy-coded image
tape011 -> rp110590n00_src.fits      Source analysis results
tape012 -> rp110590n00_anc.fits      Orbit, aspect and housekeeping data
tape013 -> rp110590n00_his.fits      Processing history
tape014 -> rp110590n00_prt.fits      Postscript information
tape015 -> rp110590n00_raw.fits      Uncorrected events list
tape016 -> rp110590n00_ltc.fits      Source light curves

HRI RDF tape list

tape002 -> rh110267n00_bas.fits      Photon event and good times
tape003 -> rh110267n00_im1.fits      Image
tape004 -> rh110267n00_bk1.fits      Background image
tape005 -> rh110267n00_src.fits      Source analysis results
tape006 -> rh110267n00_anc.fits      Orbit, aspect and housekeeping data
tape007 -> rh110267n00_his.fits      Processing history
tape008 -> rh110267n00_prt.fits      Postscript information
```

13 ROSAT HRI Recipes

13.1 Summary of tasks in this chapter

The purpose of this chapter is to provide various methods of dealing with aspects of ROSAT HRI data. Since in all cases, these apply only to the ROSAT HRI, the items presented here have not been incorporated into previous chapters.

Name	Function	Packages Needed
hkfilter	generate a QPOE time filter string from an input housekeeping file	xray:xdataio

13.2 Appearance of new HRI 'hotspots' in 1995

In 1995, two intermittent detector features (aka hotspots) particular to the ROSAT HRI have been noticed on the edge of the HRI field. These features give the appearance of a streak across a portion of the detector corresponding to the satellite wobble. For more specific analysis of the known hotspots please refer to The HRI Calbration Report . Below are the procedures for verifying and removing the features.

1. To view the hotspots:

```
 HS #1   > xdisplay "rh123456n00.qp[key=(rawx,rawy)][186:697,230:741]"
 HS #2  > xdisplay "rh123456n00.qp[key=(rawx,rawy)][3380:3891,297:808]"

OR      > xexamine rh...qp[key=(rawx,rawy)]

[HS #1, if present, appears as a spot in the lower left;
 HS #2 in the lower right.]
```

NOTE: The above coordinates are a small region centered on a hotspot. The object can be verified as a hotspot if the wobbled object in sky coordinates now appears pointlike in detector coordinates.

2. To make a KS plot, in the xtiming package,

```
        > timsort "rh....qp[key=(rawx,rawy)]"
                region: box 441 485 20 20
        > vartst
        > ksplot
```

3. To remove the hotspot, use qpcopy with the same [key=(rawx,rawy)] as above and specify a region to be excluded in an ascii file.

```
Region file: Create a region file with two lines as follows
     for hotspot #1.

FIELD
-circle(441,485,20)

> qpcopy "rh*_.qp" "rh*_no_hotspot.qp"

with the parameter "region" set with the region file name.
```

13.3 Correlating Spectral Data with Spatial Gain Variations

The mean pulse height analyzer channel (PHA) of a spectral distribution will vary depending on its location on the detector. Below is a method to compare observed data with that of preflight calibration measurements. Note however that this will give only approximate values since the actual data obtained with the wobble correspond to a finite band on the detector rather than a specific location.

For temporal gain variations, see The HRI Calbration Report .

1. Determine the roll angle of the satellite for the observation.

```
cl> imhead rh*.qp | match ROLL_NOM
```

   NOTE: The roll angle for US processed REV 0 data is given with different signs in the header compared to that of the printout (which is correct). Therefore the above method should be used only for RDF data.

2. Retrieve the gain maps (mean PHA) from anonymous ftp at SAO. A gain map for the elements carbon (0.28 keV), copper (0.93 keV) and aluminum (1.49 keV) are available.

```
> ftp sao-ftp.harvard.edu
> cd /pub/rosat/calfiles
> get README
> binary
> get gain_al.fits
> get gain_c.fits
> get gain_cu.fits
> quit
```

3. Convert the gain map files to Iraf image files using the task 'dataio.rfits'

4. Rotate one (or more) of the gain maps to the rotation of the data.

```
> rotate
```

NOTE: use the following parameters

rotation angle= (switch the sign of ROLL_NOM)
xin=258
yin=254
xout=INDEF
yout=INDEF
ncols=512
nlines=512
interpo=poly3
boundary=constant
constant=0.0

5. Store the new gain map image using "blink" in SaoImage. Under the item "scale", click on "blink" with the mouse button you would like the image stored under.

6. Display the image file created from rarc2pros (rh*_im1.imh, 8″ pixels).

```
> display rh*_im1.imh
```

7. Flip between both images to determine the position and intensity of an HRI detection on the gain map.

   NOTE: The wobble means that a general area of the detector/gain map will contribute to any give source.

8. Procedure for calculating the mean PHA channel

   (a) run qpspec on the qpoe file while selecting the region of interest

(b) calculate the mean of the distribution using the qpspec output with each channel being designated as (1.5,2.5,3.5,4.5....,15.5)

$$< i >= \sum_{i=1.5}^{15.5} i * N(i) \,/\, \sum_{i=1.5}^{15.5} N(i)$$

where $\sum$ is the summation from i=1.5 to i=15.5

$N(i)$ is the number of counts in channel i.

9. Compare between the observed and expected (calibration data) mean PHA.

   [But don't forget temporal gain variations!]

13.4 Quantum Efficiency Correction (a.k.a. 'flat-fielding')

Although the spatial variation of the quantum efficiency across the HRI is nominally within $\pm 10\%$ for most of the detector, occasionally it may be desirable to flat-field a map with preflight calibration measurements.

1. Determine the roll angle of the satellite for the observation.

```
cl> imhead rh*.qp | match ROLL_NOM
```

   NOTE: The roll angle for US processed REV 0 data is given with different signs in the header compared to that of the printout (which is correct). Therefore the above method should be used only for RDF data.

2. Retrieve the quantum efficiency map from anonymous ftp at SAO. This is a 512×512 pixel array with 8″ resolution.

```
> ftp sao-ftp.harvard.edu
> cd /pub/rosat/calfiles
```

```
> ascii
> get README
> binary
> get roshri_qe8.fits
> quit
```

3. Convert the QE map fits file to an Iraf image file using the task 'dataio.rfits'

4. Rotate the QE map to the rotation of the data.

```
> images
> rotate
```

NOTE: use the following parameters

rotation angle=	(switch the sign of the ROLL_NOM)
	xin=252.5
	yin=252.5
	xout=INDEF
	yout=INDEF
	ncols=512
	nlines=512
	interpo=poly3
	boundary=constant
	constant=0.0

5. Store the new gain map image using "blink" in SaoImage. Under the item "scale", click on "blink" with the mouse button you would like the image stored under.

6. Display the image file created from rarc2pros (rh*_im1.imh).

```
> display rh*_im1.imh
```

7. Flip between both images to determine the QE value at the source position by hitting the mouse button with the stored image. Note that the "scale" item in Saoimage must be selected.

8. Pixel intensities on both maps can be determined by the iraf task "listpix".

```
In Iraf:

>images
>listpix "rh*.imh[4050:4070,4070:4090] wcs=physical"

-where the coordinates are the image section of interest.
```

9. The source intensity can be relatively corrected to the field center. The quantum efficiency correction (QEC) is the inverse of the intensity determined from the QE map at the source position. The source intensity can be multiplied by the QEC factor to give the on-axis value. As well as point source corrections, this QEC factor can also be used for 'flat fielding' extended sources.

N.B. The source intensity correction given in the SASS output is the QEC factor.

The wobble means that a general area of the detector/QE map will contribute to any give source; hence the above method is approximate.

13.5 Filtering ROSAT HRI data on the basis of background level

For some observations, a significant amount of data may have been rejected during SASS processing because the background level was higher than the calculated threshold for rejection. It is also possible that for some requirements, an observer may wish to discard some data from the normally processed 'accepted' event list in order to have only very low background data. Starting with SASS processing version (tbd) in late 1996, additional information on background rejection and cumulative exposure times during (rejected) high background intervals is presented on the field summary page (also contained in the rh...._output.lst file generated from the RDF fits files). The MAXIMUM HIGH BACKGROUND (the largest level which will be accepted) can also be read with 'tprint' from the (RDF) generated table, rh..._stdqlm.tab. The units for MX_HIBACK (an integer) are E-7 counts/sec/pixel.

REJECTED EVENTS

Photons can be rejected for two reasons: either by time filters (e.g. high background intervals) or by their spatial coordinates. In the later case (e.g. the edges of the detector which are normally masked out, or in areas defined by the hotspot masks), the 'status' attribute of each photon is used to designate 'accepted' (status=0), rejected because of hotspot (status=1), and rejected because it is in the edge masked area (status=2).

The following discussion and recipes are restricted to RDF formatted ROSAT data since Rev0 processing handled rejected events differently.

The recipe will allow the user to include rejected events in the generation of a new qpoe file and filter on the background level. If the SASS rejected events are not required, then skip steps 1 and 3, using conventional filenames and reasonable selection values in step 2.

1. Run fits2qp on the rh*_bas.fits file to create two qpoe files.
   (a) Standard processed events (rh*_stnd.qp):
   with the parameter set as:
   "which_e=standard"
   (b) Rejected events (rh*_rej.qp):
   with the parameter set as:
   "which_e=rejected"
   "which_gti=all"
2. Run the task 'hkfilter' to create a time filter for the rejected events qpoe file.

```
>hkfilter "rh*_rej.qp[hiback=(10:20)]"

     - where the HIBACK values are your choice of minimum and maximum values.

>qpcopy "rh*_rej.qp[@hk.flt,status=(0)]" qpoe=rh*_filt_rej.qp

     -this produces the filtered (rejected) qpoe file
```

3. Create the new standard and rejected qpoe file

```
     - create an ascii file called merge.lst listing the
       two files to be appended (rh*_stnd.qp and rh*_filt_rej.qp).

>qpappend @merge.lst
```

4. Other options

   It is possible to run "rarc2pros" or "rfits2pros" for the first step with the parameter 'unscreened=yes'. This creates a qpoe file with standard + rejected events. Then all is needed is step 2 running the tasks hkfilter and qpcopy.