SELECTING EVENTS FROM QPOE FILES

In this chapter we introduce operations on EUVE QPOE files, which are mostly aimed at producing an IRAF image, like a two-dimensional spectral image or a light curve, from your selected data set. This is the step when data quality selection actually takes place. The creation of new spectral images is where GO's can apply the data selection filters constructed from the telemetry in Chapter 1, to send only those events that meet the selection criteria to a task that creates an image. While QPOE's can be used for more extensive processing, it is usually more efficient for IRAF to operate on an image with the selected data set.

Section 2.1 introduces the basic concept of the QPOE interface expression: the language used to tell the QPOE interface what events you want from the QPOE file and how they should be binned when the file is opened for processing. We give some examples of expressions for selecting events based on time, position, the most common types of rules.

Section 2.2 describes the use of macros, which define substitution rules for use in QPOE interface expressions. The use of macros can expedite data selection and image creation, especially when one makes use of the specialized IRAF/EUV tasks for reading, editing, and creating QPOE I/O macro definitions. You may have already used the task **dqselect**, which is one of these. This section introduces tasks which make it possible to construct a macro from data in an ST table, to merge parts of several expressions together, and to edit the macros in a QPOE file.

Section 2.3 gives instructions for creating spectral images from EUVE QPOE file by using QPOE interface expressions and the specialized task **euvred.qpmkim**. This task also implements correction for primbsching and TIF deadtime.

Section 2.4 shows how to make light curves or images based on any QPOE attribute using the task **euvtools.qpbin**. We give an example of making a spectral light curve, and introduce a task for generating tables of good time intervals based on the phase of a periodic source.

Related topics

◇ This chapter does not describe QPOE files in general, or EUVE QPOE file format; please read Chapter 2 and section 4.2 of the *EGO Center Data Products Guide* for this information.

◇ The causes of primbsching and TIF deadtime losses are described in section 5.1 of the *Data Products Guide*.

2.1 QPOE I/O Interface Expressions

In this section, we will introduce the syntax of QPOE/IO interface expressions and common applications with EUVE QPOE files. These methods are a means to improve signal-to-noise, save considerable processing time, and even to produce very different types of data products. When a QPOE file is sent to a task as input, it is first processed by software known as the "QPOE I/O interface," or more simply, the "QPOE interface." This is because QPOE files have a structure different from IRAF images, with many pieces of information, called attributes, for each event. A single operation may use only a subset of the attributes for some subset of the events. For example, one might want to select events by their positions to make an image array, or by times and wavelengths to make a light curve.

The pre-processing of QPOE files by the QPOE interface is how the concept of data quality selection, as introduced in Chapter 1, is implemented. The QPOE interface parses the command line to determine how to bin the QPOE event list for input to a task. This information must be packaged in a specific format, called a QPOE I/O interface expression, or a QPOE expression, for short. QPOE expressions can be typed on the command line, in a way analogous to giving IRAF an image section. Parts of the expression may be supplied by macro definitions inside the QPOE file itself. One can perform many useful data selection tasks, "on the fly", but as selections become more sophisticated, the expressions can also become quite complex. The strings of start and end times created in examples 1.1 and 1.2 of Chapter 1 are parts of one type of expression, often called a filter.

2.1.1 General syntax

The most basic use of the QPOE I/O interface is to put the entire QPOE expression on the command line, as the input to some task. The syntax of a QPOE interface expression has the general form:

```
'filename.qp[keyword1=string1,keyword2=(string2)...]'
```

The part inside the [ ]'s is what we will call the QPOE expression; it is understood that a QPOE expression must be preceded by a QPOE file name to perform any action. Also note the quote marks: single quotes must surround QPOE interface expressions used on the IRAF command line. The quotes tell the **cl** to pass the expression to the QPOE interface to be parsed. Quotes need not be used when entering QPOE expressions into the parameter file using **epar**.

A fairly complete description of the expression syntax can also be found in the **xray.qpoe** and **xray.filter** help pages of the **xray** package. The attribute names and much of the example material are specific to ROSAT and EINSTEIN data.

Keyword expressions

The QPOE expression in its most general form is a comma-separated list of 'keyword expressions' with the form, `keyword=string`. The string will be called a 'keyword string.' There are two important categories of keyword expressions: attribute filters, and function keyword expressions. The keywords that are most useful with EUVE QPOE's and their syntax are described briefly in the sections below and the keyword expressions are summarized in table 2-1.

The entire QPOE expression is parsed before any action is taken. Most of the function keywords do not affect one another, with one exception noted below. Filter expressions, on the other hand, are applied to the event list in the order they appear in the QPOE expression. Some care must therefore be taken when using more than one filter, to be sure that their combined effects produce the desired selection. Filter string syntax is discussed in the next section.

> **Note on nomenclature:**
>
> While we attempt to use a consistent nomenclature in this *User's Guide*, various parts of the QPOE expression syntax are sometimes referred to elsewhere as 'QPOE expressions,' 'filter expressions', 'event attribute expressions,' 'QPOE filters,' and other things too fierce to mention. In particular, the **xray** QPOE documentation refers to any type of keyword expression as a filter, while this document reserves the term for expressions with an attribute as a keyword. The moral is that one must be careful to make clear exactly what part of the syntax is being referred to in conversations, e-mail, and other correspondence.

2.1.2 Attribute filters have attributes as keywords

The first type of keyword expression uses any attribute in the QPOE as the keyword. The string is called a 'filter string,' and is a logical expression for allowed values of the attribute, so that the keyword expression has the function of a filter. We will often refer to this type of keyword expression, and will call it a filter expression or filter for short. The string is bracketed by parentheses whenever the it is a complex series which contains commas (often). The interface allows any combination of the following possibilities as a (comma-separated) list:

```
a          equality test                dx=4
:a         open range (<= a)            dx=(:4)
b:         open range (>=b)             dx=(4:)
a:b        range (a to b inclusive)     dx=(4:16)
! expr     not this expression          dx=!(4:16)
@file      get filters from file        dx=(@foo.filter)
```

Note that although any QPOE attribute specification, like `d0` or `s8`, can be used as a filter keyword, in EUVE QPOE files the names for the attributes – `time`, `x`, `y,lambda, theta, dx, dy` – can also be substituted. For example, the filter:

```
theta=(-.5:-.4916,.4916:.5)
```

could appear inside the [ ]'s of a QPOE expression for an EUVE QPOE file, since every EUVE QPOE contains the macro `theta` for the imaging angle attribute, `r12`. This filter would select two sections of imaging angles in Y, from above and below image center for processing. The ( )'s are necessary any time a filter string is a list of expressions with commas. As a general rule, a comma between keyword expressions expressions functions as a logical "AND", and a comma between range expressions inside a filter string functions as a logical "OR"[1].

Table 2-1: QPOE interface keyword expressions.

keyword	**keyword string**	**function**	**examples**
rect	pair of colon-separated values for key attributes	select rectangular section	[rect=[57:230,500:524]] [key=(dx,dy),rect=[40:89,100:140]]
block	single integer	define bin size used to sort QPOE event list	[block=4]
filter	attr=(comma-separated range expressions): a (attr=a), :a (attr $\leq$ a); a:b (a $\leq$ attr $\leq$ b); b: (attr $\geq$ b); !expr (not expr)	select allowed values of attribute	[time=(1234:4567)] [s10=(100:144,160:)]
mask	mask=maskfile.pl	use PLIO to include or exclude regions	[mask=backgr.pl] [mask=b(1024,964,1620,100)]
key	comma-separated pair of integer attributes	define attributes to be used as binning indices	[key=(s8,s10)] [key=(dx,dy)]

[1]See also the discussion in section A.2.1.

QPOE I/O interface bug.

There is a bug in all versions of the IRAF/QPOE interface through **2.10.4**, that makes it dangerous to specify multiple filters on a single attribute, using any variation of the form: filename.qp[attr1=(string1)...attr1=(string2)]. It turns out that the interface *does not* merge same-attribute filters reliably. Tasks will run on such expressions without apparent errors, but the number of events processed is not consistent. Unfortunately, there is no simple rule to determine which filters will be merged properly and which will not, so one must always avoid letting the QPOE interface merge filters. A single filter can safely be used to specify a large number of intervals, with a complex logical expression, *as long as the attribute name is only used once.*

2.1.3 Useful function keywords

The second type of keyword expression uses a specialized keyword to perform a specific function, and requires a specific syntax in the string. Three of the most useful keywords for EUVE GO's are: `key`, `block`, and `rect`. The function and syntax of each is described below.

`rect=[x1:x2,y1:y2]`

The `rect` keyword allows one to specify a rectangular subregion of the QPOE file which is to be extracted. The keyword string is a region specification using almost the same syntax as an IRAF image section. `rect` operates rather like an attribute filter, except that it involves two attributes instead of one: it selects the specified ranges of the default key (`x,y` for EUVE QPOE's) unless some other key has been specified with `key`. For example, the expression:

```
'test.qp[rect=[100:199,1015:1034]]'
```

will select a region of 100 columns (x) $\times$ 20 rows (y) from an EUVE QPOE file, picking out a short section of the spectrum. Note: the `rect` string has its own set of [ ]'s, inside the square braces that surround the entire QPOE expression.

The wild card character '$*$' can be used to select all possible values. One difference between `rect` and the IRAF image section syntax is that the `rect` syntax does not allow a single row or column to be specified by a single number. A range *must* be given, so `rect=[*,5]` won't work; use `rect=[*,5:5]` to select all events in the fifth row.

Another important difference between `rect` and filters on position attributes `x` and `y` is that rect produces an image with the selected dimensions, and a WCS with logical coordinates. The filter:

```
[y=(a:b),x=(c:d)]
```

produces an image with the same size as the QPOE, with 0's outside the selected area. For this reason, position filters are rarely used. A combination of `rect` and `key` expressions can be used to select almost any section of the QPOE.

Since the `rect` expression is used to select a subset of the event list before it is input to the operating task, it is applicable with tasks that do not bin the event list, such as **qppage**. i This enables the user to list the events of a small section of a QPOE file, if desired. The command:

```
eu> qppage 'sw.qp[rect=[140:1800,970:1054]]'
```

will cause **qppage** to display only those events whose (`x,y`) values match the string, since (`x,y`) is the default key. Events are displayed in y-order, from 970 to 1054, including only those with x-values in the range 140–1800, which is about the width of the detector in the 2048×2048 pixel space.

Related topics

◇ The task **qppage** is used for scrolling the event list of a QPOE file, and is described in section 2.3.2 of the *EGO Center Data Products Guide*.

`block=pixsize`

For expressions using the keyword `block`, the keyword string is a positive integer which tells the QPOE interface the number of pixels to sum together when binning the event list. Like `rect`, it operates on the current key attributes. Its single argument is applied to both the column key and the row key; if `[block=4]` is specified, a 2048×2048 EUVE QPOE file would be binned into a 512×512 image. To display an EUVE QPOE file as a blocked image in your image tool, type:

```
tv> display 'sw.qp[block=4]' 1
```

Like `key`, the `block` keyword is usually used when the QPOE file is being binned for handling by an image task, such as **tv.display**, **qppage**, or **qpmkim**.

`mask=(<plio expr> or file.pl)`

There is also a keyword that allows the creation of more complex image sections than `rect`, using what are called PLIO (pixel list input/output) region specifications. The QPOE interface uses the keyword `mask` to refer to this kind of keyword string, which may be included literally, or stored in a PLIO region mask file (hereafter simply 'mask file'). The mask file may contains a list of many region specifications that select areas with various geometric shapes from the QPOE pixel space. Neither parentheses or the "@" symbol are needed, but the file must be a PLIO mask file and its name must end with *.pl*.

PLIO region mask files are made with the task **xray.ximage.plcreate**; please read the help page for instructions on using the task. The PLIO syntax and rules for specifying regions —the input to **plcreate**— are described in the **xray** documentation. Type 'help xray.regions', 'help xray.reggeometry', 'help xray.regalgebra', and 'help xray.regcoords' for an overview and details. The information needed for simple applications is summarized below.

Table 2-2 gives an incomplete list of shape definitions, from the help page **xray.regions**. These can be input to **plcreate** as a command line list separated by semicolons (;) to make a plio file. Arguments are (converted to) real numbers, and angles are in degrees counter-clockwise from the positive Y-axis. The region names are case-insensitive, and arguments may be listed with or without parentheses, separated by commas and/or spaces. Regions are inclusive by default, with the exception that preceding a region by a minus sign (e.g. –c xcen ycen radius) excludes that region from any other included region in the mask. If two regions overlap, the pixels in the overlap region are included or excluded according to the *last* expression to the right, so order can be important.

Table 2-2: Summary of PLIO region specifications. The PLIO interface permits parameters to be entered as comma- and/or space-separated lists, with or without parentheses.

Region type	abbreviation	arguments	examples
ANNULUS	ANN,A,a,ann	xcen ycen rad1 rad2	a 1029 1019 15 35
		[n=# OR rad3 rad4...]	ANN(1029,1019,15,35)
BOX	b,box	xcen ycen xwidth yheight [angle]	box 1020 1030 1800 12
			b(1020,1030,12,1800,90)
CIRCLE	CIR,C,c,cir	xcen ycen radius	cir 917 1012 10
			c(917,1012,10)
ELLIPSE	ELL,E,e,ell	xcen ycen xwidth yheight angle	E 34 129 40 60 30
PIE	pie	xcenter ycenter angle1 angle2...	pie(1200,1400,15,135)
POLYGON	POL,pol	x1 y1 x2 y2 x3 y3... xn yn	pol 40,120,40,220,65,170

Region expressions may use logical, physical, or world coordinates; logical coordinates, which may be changed by filters or other keyword expressions that precede the mask, are assumed as the default. This means that if a mask is preceded by a `block` expression, the region arguments must be given in the blocked coordinates.

More complex region expressions can be built up using shapes, the arithmetic operator –, and the logical operators ! (not), & (and), ∧ (exclusive or), and | (inclusive or). Type "help xray.regalgebra" for specifics and some examples.

`key=(sn,sn+2)`

The `key` expression is used implicitly or explicitly whenever the QPOE event list is binned into an image array. This happens when the QPOE is used as input to tasks that expect an image, like **display** or **imcopy**, and tasks that create an image as output, like **qpmkim**. Each QPOE has a default key: the positional attribute used to sort the event list when the file was created. If the QPOE is accessed with no explicit key specification, this default `key` tells the interface to bin the QPOE using the sort-attributes (usually row and column) as the indices. Therefore, using the default key is the most efficient way to bin a QPOE file.

When this keyword is used explicitly, the keyword string consists of a comma-separated pair of short integer attributes in parentheses. Only short integers can be used for the key, as they will become the IRAF image indices. The most common application is to specify alternate position attributes to use in binning the events when sending a QPOE to a task like **tv.display**. Sorting by a non-standard key can produce a different kind of image. For example, to create the detector images on the data tape, *(s,m,l)w_det.FITS* and *ds_det.FITS*, the assignment

```
[key=(dx,dy)]
```

is used. This is equivalent to the expression:

```
[key=(s20,s22)]
```

without the special names `dx` and `dy`. This tells the interface to bin the events using the raw detector coordinates for the column and row indices, rather than the default key coordinates, the remapped `x` and `y`.

2.1.4 Some QPOE expression rules

- The function keywords `rect`, `block`, and `mask`, normally operate on the default key. Specifying an alternate key causes such keywords to operate on the selected key attributes instead, *regardless of the order* in which the keywords appear in an expression.
- Any keyword string that contains a comma *must* be surrounded by parentheses, or QPOE will misinterpret the expression. Parenthesized expressions may be nested within a keyword string. See section A.2.1 for more information about how commas and parentheses interact.
- Blocking is performed after `rect` and attribute filter operations, so `rect` and filter expressions are written in unblocked coordinates. Later references to sections of displays or images made by blocking a QPOE file must be written in the *blocked* coordinates.
- Using a key other than the default key can cause a large increase in processing time, since the event list is already sorted according to the default key.

- One can make a region filter equivalent to a `rect` expression by using an attribute filter directly on the current sort key (see the item above). However, this will create a full-sized image (2048×2048 and 8 Mgb for EUVE QPOE's) with empty pixels outside the desired section, instead of a smaller image.

Related topics

◇ Appendix A of this *User's Guide* has more detailed descriptions of several aspects of the QPOE interface.
◇ Section A.2 in Appendix A has more details about QPOE expression syntax, including notes on constructing efficient filters to reduce processing time.
◇ Use the IRAF/EUV task **tfilter**, described in section 2.2.3 below, to merge several filters on a single attribute.
◇ Section A.1.3 introduces QPOE header parameters which can be used for storing default attribute filters and region masks.

2.2 QPOE Macros

QPOE expressions can become long, complex, and tedious to type. Because EUVE time tags have some 12 digits of accuracy, many filter strings are much too long to even attempt typing them on the command line. In Chapter 1, we saw that the task **dqselect** can install filter strings in QPOE files as macro definitions. The IRAF/EUV packages contain other tasks which can create filter strings from one or more ST tables, merge existing filter strings, install a string as a macro in a QPOE file, and edit existing QPOE macros. Before we introduce these tasks, we will introduce the concept of macros on a more general level, and discuss their role in EUVE data reduction. More detailed descriptions of the use of macros with the QPOE interface have been deferred to Appendix A.

2.2.1 Reading and writing macro definitions

Most QPOE macros are substitution rules with the form `NAME=string`. For example the definitions in EUVE QPOE files which allow you to use more mnemonic names for the attributes, `x`, `y`, `time`, etc., are simple substitution macros. Macros appear in the output from **imhead** as lines like:

'TIME = ' macro[002] macro definition'.

but **imhead** is not capable of showing the entire macro definition. only its length in characters ([002]) is given, and the macro name has been capitalized, to make it conform to FITS-card format. Most of this section is about data selection macros that contain part of a QPOE I/O interface expression, like a filter string, and are somewhat more complex.

Although we have seen how filter strings can be installed as macro definitions in a QPOE file, placing a macro in the file doesn't do any data selection. Macros have no effect *until the macro name is used in a QPOE interface expression* which causes the interface to select the events the filter specifies for some processing. For example, the two-dimensional spectral images on each data tape are made using a QPOE time-filter macro, called `(det)_gt`[2]. However, the events that go into the 2-d spectral images are not selected until the images are made, using the task **qpmkim**with macro name in a QPOE expression replacing the filter string. The QPOE interface performs the appropriate substitution, and selects the subset of QPOE events selected by the filter for binning. The macro can be reused any number of times to make new images from the QPOE, and provides a permanent record of the selection filter that travels along with the event list.

Related topics

- ◇ The construction and installation of the filter macros is described in section 4.3 of the *EGO Center Data Products Guide*, and in example 1.1 of this *User's Guide*.
- ◇ Macros can be defined in at least two places in IRAF, allowing for a hierarchy of precedence; see the discussion of global and local macros in section A.3.

2.2.2 QPMEDIT displays and edits QPOE macros

Suppose one wanted to inspect or change the macros that are installed by the EGO Center in the QPOE files. Users frequently find it convenient to use the task **imhead** to look at the global information in a QPOE file, but the information that **imhead** can supply on macros is minimal. You may recall from section 2.1 of the *EGO Center Data Products Guide* that this is because QPOE files are *not* images–the listing one gets by running **imhead** on a QPOE file is merely an attempt to represent the contents of the QPOE file in the FITS-like format of an IRAF image header.

The task **euvtools.qpmedit** operates on QPOES in much the same way that **images.hedit** operates on image header files. It provides capabilities for examining, adding, deleting, and changing the macro definitions in any QPOE file. Other parts of the QPOE file, such as the header parameters and the event list, cannot be changed with **qpmedit**. The **qpmedit** parameters also supply options for safeguarding existing macros from incorrect and inadvertent changes.

Figure 2-1 shows a parameter file setup to display the definition of the exposure filter macro `sw_gt`, using the special character "." for `value`, which means "the current value." One can view the complete names and definitions of all the macros in a QPOE file by using the wild card character (∗) for the macro name in **qpmedit**:

```
eu> qpmedit sw.qp * . update-
```

Explicitly setting the hidden parameter `update` on the command line insures that none of the macros will be changed, regardless of the settings of the parameters `add`

[2]where **`det`** is a substitute for `sw, mw, lw,` or **`ds`**, which is installed into each QPOE file during the EGO Center's standard batch processing.

```
eu> lpar qpmedit

   files = "sw.qp"        QPOE files to be edited
  macros = "sw_gt"        macros to be edited
   value = "."            value expression
    (add = no)            add rather than edit fields
 (delete = no)            delete rather than edit fields
 (verify = yes)           verify each edit operation
   (show = yes)           print record of each edit operation
 (update = no)            enable updating of the image header
   (mode = "al")
```

Figure 2-1: Parameter file for **qpmedit**. Typing ":go" in **epar** will start the task, displaying the macro "sw_gt" in full, gory detail on `STDOUT`.

or `delete`. To add or delete macros, either `add` or `delete` and the parameter `update` must be "yes." The command:

```
eu> qpmedit lw spec "y=1000:1040" add+ update+
```

will create a macro in the QPOE file *lw.qp* that substitutes the string `y=1000:1040` for the name `spec` in a QPOE expression, which will select a strip from the center of the detector when invoked in a QPOE expression. Note that another macro, the attribute name `y`, is used in the macro value definition. Existing macros can be edited by setting `add` and `delete` to "no", and giving the macro name and the new value. The following will redefine an existing exposure macro called `exposure` in an MW QPOE to be the contents of the file *mytimes*:

```
eu> qpmedit mw.qp exposure "@mytimes" add- delete- update+
```

The file *mytimes* should contain part or all of a correct QPOE expression. It is the user's job to keep track of which parts, and whether punctuation like ( )'s need to be used when the macro is referenced. If you looked at the macros in *sw.qp*, you may have noticed that the definition of "sw_gt" did NOT include the attribute name `time`. When the QPOE is sent to the task that creates the spectral images, the macro supplies only the filter string in the QPOE expression. The [ ]'s, filter keyword, and parentheses are also needed to make the complete expression:

```
'sw.qp[time=(sw_gt)]'
```

Whenever a macro is referenced, the string on the right of the "=" sign, must form a complete QPOE I/O expression with correct syntax when substituted into the command line. The person creating the macro decides what parts of the desired filter to include in the definition. All QPOE macros installed in EGO Center processing follow the convention of including only the filter string.

2.2.3 TFILTER creates QPOE time filters from ST tables

The type of data selection that has been implemented in the **euv** package uses filtering of the recorded times of QPOE events. This means that when a GO wishes to apply more than one criterion to data selection, for example selecting both for valid exposure times and for certain values of a periodic phase, two or more separate sets of time intervals may be required, one set to satisfy each of the criteria. Simultaneous application of all the criteria amounts to taking the intersection (logical and) of all the sets of time intervals to create a single specifically tailored "good times" filter. This is what the task **euvtools.tfilter** is designed to do.

> **Why should I use tfilter?**
>
> The main reason for having tasks that intersect sets of time intervals (e.g. **tfilter** and **dqselect**) is to work around a QPOE interface bug, which causes multiple filters on the same attribute to produce unreliable results. Tests have shown that the results of this type of expression depend in an unspecified way on the order of the filters in the QPOE expression.
>
> The main reason for EUVE GO's to want to use multiple criteria is to merge the nominal exposure filter `(det)_gt` with other time filters customized for their science. So GO's who wish to add their own time filters to the QPOE files should produce an ST table of start and end times for each criterion, and use **tfilter** to create a single time filter macro, rather than trying to AND several separate keyword expressions or filter strings together "on the fly." The `gtsave` command in the task **dqselect** creates ST tables of start and end times in the format required by **tfilter**.

tfilter converts table columns with the start and end times for good time intervals into a QPOE filter string, which can then be written to one of several possible output locations. Here we describe the option of inserting the string into a QPOE file as a new macro. A sample parameter file for **tfilter**, set up to create a new time filter string from two existing ST tables, is shown in figure 2-2.

```
    eu> lpar tfilter

     input = "det7filt,ds_gt" Input table name
   (output = eds.qp")        Output file
     (name = "bestime")      Name of output macro
    (start = "stime")        Name of starting column
      (end = "etime")        Name of ending column
  (cmdline = no)             Command line output flag
   (format = "13.3f")          Interval format control
     (mode = "al")
```

Figure 2-2: Parameter file for **tfilter** to merge two lists of time intervals. The filter string will be installed as a macro in the DS QPOE file.

Notes on running **tfilter**

- The value of `input` may be a comma-separated list of several table names; do not include the *tab* extensions.
- Each table in `input` should have columns of start and end times in the same order, named `start` and `end`. Tables that were not created by IRAF/EUV tasks may need to be edited, using an editing task such as **ttools.tchcol**, or **ttools.tedit**.
- `output` names the destination of the finished time filter string. If `output` has the *.qp* suffix, then the string is written into the QPOE file `output` as the macro `name`. Alternatively, `output` can be a text filename, to which the complete definition of the macro `name` will be written or appended.
- The `cmdline` parameter takes precedence over `output`. If it is "yes" (the default), the task output is directed to `STDOUT`. This option allows the user to make a "dry run" first, and check the output. `cmdline` *must be "no" to write the output to the destination specified by* `output`.
- The value of `format` specifies the format for the start and end times output in the filter string, using C language formatting conventions. The time stamps supplied in the EUVE QPOE files and good times tables have a nine-digit mantissa and numerous insignificant decimal places. Considerable space can be saved by truncating the times to 3 digits; this implies no loss of accuracy, as the times in the telemetry are known only to within ≈ 1 ms in WSZ mode data, or ≈ 8 ms for XY mode.

To create the QPOE time filter `bestime` from two existing tables and install it in the Deep Survey QPOE file:

```
eu> tfilter det7filt,ds_gt output=ds.qp name=bestime
```

tfilter may be used to define either a local macro in the QPOE file itself, as in the examples above, or a global QPOE macro in a macro definition file. See section A.3.1 in Appendix A for a short discussion of global QPOE macros. If the `output` file is a text file of global macros, the definition of the macro `name` is appended to the file. The `cmdline+` option also provides a way of applying a filter to the QPOE in a command substitution macro reference. Command substitution macros are described earlier in section A.3 of Appendix A.

Example 2.1 : Making a phase-limited spectrum

Suppose that a GO has determined the start and end times of some interesting event, perhaps a stellar flare during an observation, and wants to extract the short wavelength spectrum for that particular interval. The interval may span several orbits, including daytime gaps, and may contain interference from the SAA or another source of background. The GO should therefore combine the interval of interest with the valid times filter already installed in the QPOE file. A new spectral image must then be created using the new filter.

Specifically, if the GO wishes to extract a spectrum from the SW spectrometer taking only photons from the time interval JD=(762023822:762028792), he can use the task **tfilter** to combine this desired time interval with the standard exposure macro called `sw_gt` in the SW QPOE file. In order for **tfilter** to be able to read this interval it must be contained in a time intervals table, with the same format as the tables produced by the command `gtsave` in **dqselect**.

The easiest way to make the table is to modify another table of the same form, so the GO uses one of the tables of SW detector good times supplied on the data tape. The table for the SW spectrometer is called `sw_gt.tab`, so the GO uses the task **ttools.tselect** to make a new table caled *mytimes.tab*, containing a copy of the first row of the existing table:

```
ta> tselect sw_gt mytimes "row()=1"
```

The GO then uses the task **tedit** to change the start and end times in *mytimes.tab* to the times bracketing his flare event. If several non-overlapping time intervals were desired, rather than just one, more lines from *sw_gt.tab* could be included in *mytimes.tab*, and edited in the same way. See the help page for **tedit** for details on its use, or start the task and type "`control-d help`" on the prompt line to list **tedit** commands.

Next, the new table is intersected with the EGO Center valid exposure times in **tfilter**, and the resulting filter string is used to create a QPOE macro named `mytimes` in *sw.qp*:

```
eu> tfilter mytimes,sw_gt output=sw.qp name=mytimes cmdline-
format=13.3f
```

The parameter `cmdline` is turned off, so that the output is written to the QPOE file, instead of displayed on `STDOUT`. The `format` parameter tells **tfilter** how many significant digits of each timetag to use.

Once the appropriate filter string is installed in the QPOE file, the GO can create a new 2-d spectral image from the flare with the QPOE expression:

```
'sw.qp[time=(mytimes)]'
```

as input to the task **qpmkim** (see section 2.3.1).

Related topics

- ◇ Section A.3 in Appendix A introduces some additional ways to create, store, and use macros.
- ◇ See section 4.3.1 – 4.3.3 of the *EGO Center Data Products Guide* for a description of macros installed in EUVE QPOE files during EGO Center processing.

◇ If you have problems with a macro that is very long (> 64k characters), read section A.4.1, on configuring the QPOE interface to allow longer macros.

2.3 Making Spectral Images

Carefully made time filters select an optimal subset of event data for spectral extraction or other processing. However, the filtering of the QPOE file that must be done by the QPOE I/O interface to access the selected data set is time-consuming, especially if several filters are involved, and can take quite a long time with EUVE QPOE files. It is more generally more efficient to filter the QPOE through the interface only once, creating a standard IRAF spectral image that is used from then on. Creating a light curve is essentially the same task; the only significant difference being that the QPOE file is binned with `time` as one of the key attributes. Once a data subset is selected and placed in an image, it is not necessary to work with the entire QPOE to extract spectra and determine fluxes (Chapter 3), or for model comparisons (Chapter 4). The focus of this section is helping GO's create the best two-dimensional spectral image for their science, using the specialized time-filters created with **dqselect** and **tfilter**, and the **euv** task **qpmkim** for making corrected spectral images.

2.3.1 QPMKIM makes corrected EUVE spectral images

The task **euvred.qpmkim** is written specifically for making two-dimensional spectral images from EUVE QPOE files, with corrections for TIF deadtime and primbsching effects. Although it is possible to make images from QPOE files using **imcopy**, **qpmkim** adds the capability of correcting the exposure time or the resulting pixel values, for TIF deadtime and primbsching effects that might otherwise cause low count rates. In the default mode, corrections are based on the information in the processing table *primbsch.tab*, which is supplied on the standard data tape. See the help page for the task **mkprimbsch** and section 1.2.1 for details on how the corrections are calculated.

Do I need to run qpmkim?

If you have revised or remade the data selection filters for your QPOE file, using **dqselect**, **tfilter**, or **qpmedit**, you will need to re-create your two-dimensional spectral images to carry out the actual filtering on the data. For those GO's who find the EGO Center exposure filter adequate to their needs, this work is already done; they can move on to re-extracting the spectra from the 2-d spectral images, or reducing the spectra supplied on the tape.

A sample parameter file for **qpmkim** is shown in figure 2-3, set up to produce an exposure-corrected image like those on a GO data tape. We give some notes on running **qpmkim**, but read the help page for more complete information.

```
eu> lpar qpmkim

      input = "sw.qp[time=(mytimes),rect=[140:1779,950:1089]]" QPOE file
     output = "swimg_myti"    Image file
   (corrtab = "primbsch")     correction table name
   (refdata = "reference/detector") reference directory
 (applycorr = yes)            apply correction
(adjexptime = yes)            adjust exposure time
   (errband = no)             create error band
  (deadtime = yes)            do deadtime correction
  (primbsch = yes)            do primbsch correction
 (attribute = "time")         attribute used for exposure time
   (timecol = "")             optional alternate time column name
   (corrcol = "")             optional alternate correction column name
  (tepsilon = INDEF)          alternate epsilon value used for time comparison
   (deltime = INDEF)          alternate delta of correction times
  (infomsgs = yes)            give processing information messages
   (fixmwcs = yes)            apply QPOE mwcs workaround fix
 (timefield = "time")         QPOE event time field
   (dxfield = "dx")           QPOE event dx field
   (dyfield = "dy")           QPOE event dy field
      (mode = "al")
```

Figure 2-3: Parameters for creating an exposure-corrected image file with **qpmkim**, using the standard correction table *primbsch.tab*.

Notes on running **qpmkim**

- The value of `input` should be a QPOE interface expression, including any filters that should be applied to the event list. For EUVE QPOE's, the QPOE expression should specify at least an exposure filter, to eliminate the daytime gaps from the exposure time recorded in the image header. Read the next text on page 2-18 if you have processed your QPOE file with the **xray** package.
- The value of `refdata` should point to the table of detector data, supplied in the reference data on the tape. The default value should be correct in the standard EUVE data directory configuration (see figure 0-3 in the *Data Products Guide*).
- The default value of `corrtab` is the name for the table produced by the task **mkprimbsch** in the standard EGO Center data processing.
- If `applycorr` = "yes" **qpmkim** gives the user a choice of two different correction modes, depending on the value of `adjexptime`:
  1. When `applycorr` and `adjexptime` are both "yes," the task runs in adjusted exposure time mode. This means each two-frame interval of the selected exposure time is *divided* by the appropriate correction factor, from `corrtab`. Factors to correct for TIF deadtime and/or primbsching effects can be

specified using the boolean parameters `deadtime` and `primbsch`. Adjusted exposure time mode is used during EGO Center data processing, because it makes the most efficient use of the QPOE interface. It is recommended that GO's use this mode unless there is some compelling reason to use the adjusted pixel mode.

2. If `applycorr` is "yes" and `adjexptime` is "no," the task runs in adjusted pixel mode. Then each *event* is given a real value of 1.0 and *multiplied* by the correction factor before being added to the value of the pixel at its location. The result is an image with real-valued pixels, which requires twice as much memory as the usual short-integer pixels.

- In adjusted pixel mode, Poisson errors are calculated for each pixel, and can be saved in a separate image band by setting `errband` to "yes." Users should be aware that this effectively doubles the image size *again* over what is produced in exposure correction mode, as the error band is also real-valued. A 2048×2048 pixel-corrected image with error band requires 32 Mgb.

- **qpmkim** reads the `INSTRUMENT` header parameter from the input QPOE file to determine which detector's correction factors to use. If `deadtime` and `primbsch` are both "yes", it looks for a column called `Det#Q#dpc`. If only one is set, either column `Det#Q#dc` or `Det#Q#pc` will be used[3].

- If an alternate correction table is to be used instead of *primbsch.tab*, the name for the column of time stamps for each correction factor is loaded into the parameter `timecol`. Similarly, the parameter `corrcol` can be used to specify the name of the correction column in `corrtab`.

- The `fixmwcs` option enables or disables a fix to a QPOE mini-world coordinate system bug. The bug causes the values of the pixels in adjusted pixel mode to be shifted slightly. Until it is confirmed that the bug has been fixed, leave this parameter set to "yes".

- EUVE GO's should not normally need to change the values of `attribute`, `dxfield`, `dufield`, `timefield`, `dxfield`, or `dyfield`.

To create an image from one of the EUVE QPOE files, first determine the type of filtering needed and the necessary criteria for "good" data. Use the criteria to create filter strings from the telemetry monitors and save them in tables or in the QPOE header using **dqselect**, **tfilter**, and **qpmedit**. Determine what correction, if any, is needed, and make sure you have a suitable table of correction factors, such as *primbsch.tab*. You are then ready to run **qpmkim**.

[3]The Q#'s in the column names indicate filter quadrants. However, at this time **mkprimbsch** can only calculate correction factors for the quadrants which normally have spectra or the DS image.

Does the QPOE file have a `deffilt` card?

If the QPOE file has been processed with tasks in the **xray** package, it should have had the header parameter `deffilt` inserted. It is important to be aware of the action of this filter, which is always parsed by default in the QPOE interface *and merged with any other filters on the same attributes.* Because of the QPOE interface bug mentioned on page 2-5, some attempts by the interface to merge same-attribute filters may produce erroneous results. You may also wish to read section A.1.3 of Appendix A for guidelines on how to prevent the QPOE interface from merging multiple filters.

Simple calls to **qpmkim** that use the default table and correction type are just as easily made on the command line. For example, to create a spectral image of the entire detector area for the SW:

```
eu> qpmkim 'sw.qp[time=(mytimes)]' bigsw_myti
```

qpmkim receives the events, plus the spectral WCS from the QPOE, packed in an appropriate set of image header cards. If the task is run in exposure time correction mode, the corrected exposure is written into the header card `EXPTIME`, and the time calculated from the exposure filter is saved in the `RAWEXP` card.

Finally, do not attempt to use **qpmkim** to make a pixel corrected image from any QPOE file that has been resorted in other than `y,x` order (sorted first by row, then by column). If **qpmkim** is used on such a QPOE file in pixel correction mode, it will produce an incorrect number of events, because of the way the algorithm is optimized. This is most likely to come up for GO's who have created time-sorted versions of the QPOE file for timing analysis. **qpmkim** will not fail or produce any error message under these circumstances. The amount of error increases with the amount of primbsch correction needed in the data set. Use **qpbin** on the y,x-sorted QPOE file (section 2.4.1) if an image binned by event time is required.

2.3.2 Making images with IMCOPY

Making a spectral image from a QPOE file with the standard **images** task **imcopy** is equivalent to running **qpmkim** with the parameter `applycorr` set to "no." Data selection capabilities are still provided by the QPOE I/O interface in either case.

```
im> imcopy 'sw.qp[time=(mytimes),rect=[140:1779,954:1089]]' sw_spimg
```

is equivalent to:

```
eu> qpmkim 'sw.qp[time=(mytimes),rect=[140:1779,954:1089]]' sw_spimg ap
plycorr-
```

Related topics

◇ If you are using a QPOE that has been processed with the **xray** package, please see sections A.1.4 and A.1.3, and example A.1, on working with default filters.
◇ The subsection "Examples of specialized QPOE expressions" in section A.2.1 has some examples of QPOE expressions for creating different images.

2.4 Attribute Binning and Timing Analysis

Some targets have interesting periodic behavior in the EUV. Having the data in the form of time-tagged event list (QPOE) files makes it relatively easy to create light curves from EUVE data. Making a light curve corresponds simply to sorting the data with the attribute `time` as one of the keys. This section shows how to create light curves for all or part of a spectrum, or from the Deep Survey Lexan image.

2.4.1 QPBIN rebins on any attribute

The task **euvtools.qpbin** is used to rebin or "project" a QPOE file with respect to one of its attributes. Since rebinning along the coordinate axes is already easy to do using QPOE interface expressions, **qpbin** is most helpful for time binning. It sorts events from QPOE's into a one-dimensional distribution of counts vs. attribute value with an bin width specified by the user, and writes the array as an IRAF image. A typical parameter file for **qpbin** is shown in figure 2-4. The use of **qpbin** to make a background-subtracted light curve is illustrated in example 2.2.

```
eu> lpar qpbin


      input = "swdet.qp[rect=(300:350,720:1299)]" QPOE file
     output = "sw_83A"         Image file
 (attribute = time)            QPOE field name
        (x1 = INDEF)           Lowest value
        (x2 = INDEF)           Highest value
        (dx = 100.)            Delta value
        (nx = 64.)             Number of output bins
     (units = "seconds")       QPOE field units
 (integrate = yes)             Provide range information
      (mode = "al")
```

Figure 2-4: Parameters for **qpbin** to make a light curve with 100-second bins. The value of `nx` is superseded by that of `dx`.

Notes on running **qpbin**

- Many users want to make a light curve for only part of the spectrum, or even a single feature. Select the events from the interesting spectral region by using a QPOE interface expression such as a `rect` expression in `input`. Expressions may also be made using the attributes λ (wavelength) and Θ (imaging angle) as the sort key; this obviates having to locate the desired area in the (x,y) pixel space. The size of a spectral resolution element in each of the Spectrometer channels is about seven pixels.

- Any attribute in the QPOE file may be used to do the binning, and can be selected by setting the value of `attribute` to a QPOE attribute expression, like `d8`, or by using a macro already defined in the file, such as `time`.

- If the range parameters `x1` and `x2` are both "INDEF," **qpbin** will use the entire available range of `attribute`. This will override any setting of `nx`.

- At least one of the parameters `dx` and `nx` must always be defined (i.e. not INDEF). If the values of `dx` and `nx` conflict, `dx`, the bin size, takes precedence, and **qpbin** creates as many pixels as necessary to span the interval.

- The `units` of `attribute` can be specified for inclusion in the image header.

- Set the parameter `integrate` to create a second line with valid time information.

Data gaps and valid time integration

Gaps in the data cause problems in creation of equal-width time bins. Usually, **qpbin** divides the interval `x1-x2` by `dx` or `nx` to create time bins and sums together the events it finds within each bin. However, because there are always some gaps in the telemetry, at least for orbit daytimes, and additional time filters are often used, one cannot predict in advance how many seconds of any time bin include good, or even valid, data. For this reason, the parameter `integrate` should be turned on when any time filter, which may incorporate the standard exposure filter and user-defined filters, is applied to the input to **qpbin**, i.e., just about always.

When an exposure filter is used, and `integrate` = "yes", **qpbin** computes the number of seconds in each bin that passed the filter, and this information appears in the output image as the value of a corresponding pixel in a second line. The GO can then normalize the pixel values, by using the task **images.imarith** to divide the first line of the output image by the second[4]. This produces a light curve of the count *rate* versus time. Appropriate error bars can be computed using **imarith** and the information in the two lines of the output image. If the bins have very small effective exposures, each one will have few counts and the data will be noisy.

[4]Be sure to set `divzero` in **imarith** to"0" in case some bins correspond to no valid times.

Background subtraction for light curves

Background subtraction poses another complication. The background count rate can vary substantially during an EUVE observation, and **qpbin** makes no background correction. The user can perform this task by making a light curve of an appropriate background region with **qpbin**, and then using **imarith** to subtract the two light curves.

Example 2.2 : Making a light curve

This example demonstrates the use of **qpbin** in making a light curve of a selected spectral region. Suppose the GO wants to monitor the brightness of a group of spectral features between 106 and 128 Å. The correct QPOE expression must be input to the task **qpbin** to extract the light curve.

First, the GO displays the QPOE file *sw.qp* in an image window, the lines are identified "by eye", and the area they occupy is marked with the cursor. This part of the spectrum is pixels 633 – –956 in `x` and pixels 1015 – –1034 in `y`. This leads to the following QPOE expression, to be used as input to **qpbin**:

```
sw.qp[rect=[633:956,1015:1034],time=(sw_gt)]
```

The exposure filter string from the nominal EGO Center's reduction is accessed as the macro `sw_gt` stored in the file.

The GO then uses a single command to create the light curve and write it into an image called *scurve*:

```
eu> qpbin 'sw.qp[time=(sw_gt),rect=[633:956,1015:1034]]' scurve attri=
time dx=30 units=seconds integrate+
```

The same method is used to select a background region with the same width as the source region, but above the spectrum and 100 lines high. The GO extracts a background light curve with a similar call to **qpbin**:

```
eu> qpbin 'sw.qp[time=(sw_gt),rect=[633:956,1040:1139]]' bcurve attri=
time dx=30 units=seconds integrate-
```

Note that a line of normalization factors is not made for the background curve (`integrate-`). It is not necessary to normalize the curves independently; since they are both created with the same time filter, their norms would be the same:

$$\frac{scurve}{norm} - \frac{bcurve}{norm} = \frac{[scurve - bcurve]}{norm}$$

The task **images.imarith** is then used to normalize the background light curve by the ratio of the background sample area to the spectrum, and subtract the background curve from the source curve:

```
im> imarith bcurve / 5.  bcurve_n

im> imarith scurve[*,1] - bcurve_n[*,1] scurve_sub
```

where 5 if the ratio of heights of the background and the section used for the light curve. This puts the background-subtracted light curve into the image *scurve_sub*. Then the exposure is corrected, using the normalization values in line 2 of *scurve*:

```
eu> imarith scurve_sub / scurve[*,2] final_curve
```

This final step turns counts to count rate in each time bin, correcting for any exposure variations between the bins. The subtracted and normalized curve is shown in figure 2-5.

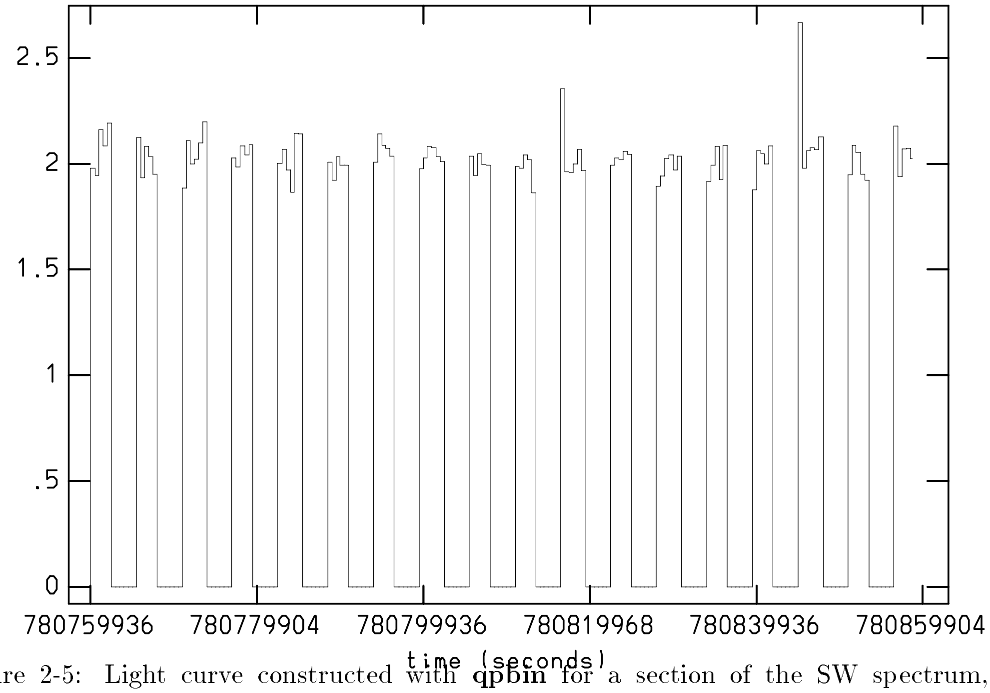

Figure 2-5: Light curve constructed with **qpbin** for a section of the SW spectrum, in background-subtracted normalized counts per second.

The count rate shown is from the entire spectrum area chosen with the `rect` QPOE filter. The bins with the very highest rates are actually derived from low-exposure low-count times bordering some rate shutdowns in the SAA, and so will have large error bars. The GO constructs error bars for the light curve from the signal and background counts in the light curves. Recall that the standard deviation of events accumulated as Poisson statistics in a bin with signal S and average background $\bar{B}$ is given by:

$$\sigma_S = \sqrt{S + \bar{B}\,(1 + 1/n)}$$

where n is the ratio of the background height to the spectrum height, Since *scurve* contains both source and background counts (S + B), and the GO has already formed the average of the background curve in *bcurve_n*, an image containing the deviation for each pixel of *final_curve* can be created by the commands:

```
im> imarith bcurve_n[*,1] / 5.  sigmaB

im> imarith scurve[*,1] + sigmaB variance

im> imfunction variance deviation sqrt

im> imarith deviation / scurve[*,2] error
```

The resulting image, *error*, contains the properly scaled Poisson errors for *final_curve*. The error bars are plotted in figure 2-6.

The smaller deviations in the plot show variations of up to 5%, due mainly to unremoved Primbsching/deadtime effects, so the curve shows no obvious variations at this time resolution.

A broad band light curve can be made from the Deep Survey QPOE file, using similar commands, and region masks which select the source area and a background region. For example:

```
ds.qp[time=(ds_gt),mask=image.pl]

ds.qp[time=(ds_gt),mask=backgr.pl]
```

where the PLIO mask files *image.pl* and *backgr.pl* contain the regions `cir(1024, 1024, 15)` and `ann(1024,1024,20,35)` respectively. See the help pages **xray.region** and **xray.reggeometry** for a detailed description of region masks.

2.4.2 PERIODINT calculates times for different phase points

The **euv** package supplies the task **euvtools.periodint** for generating tables of phase intervals for periodic sources. It creates a table of start and end times for a selected section of the object's phase cycle, over the period of the observation. The inputs

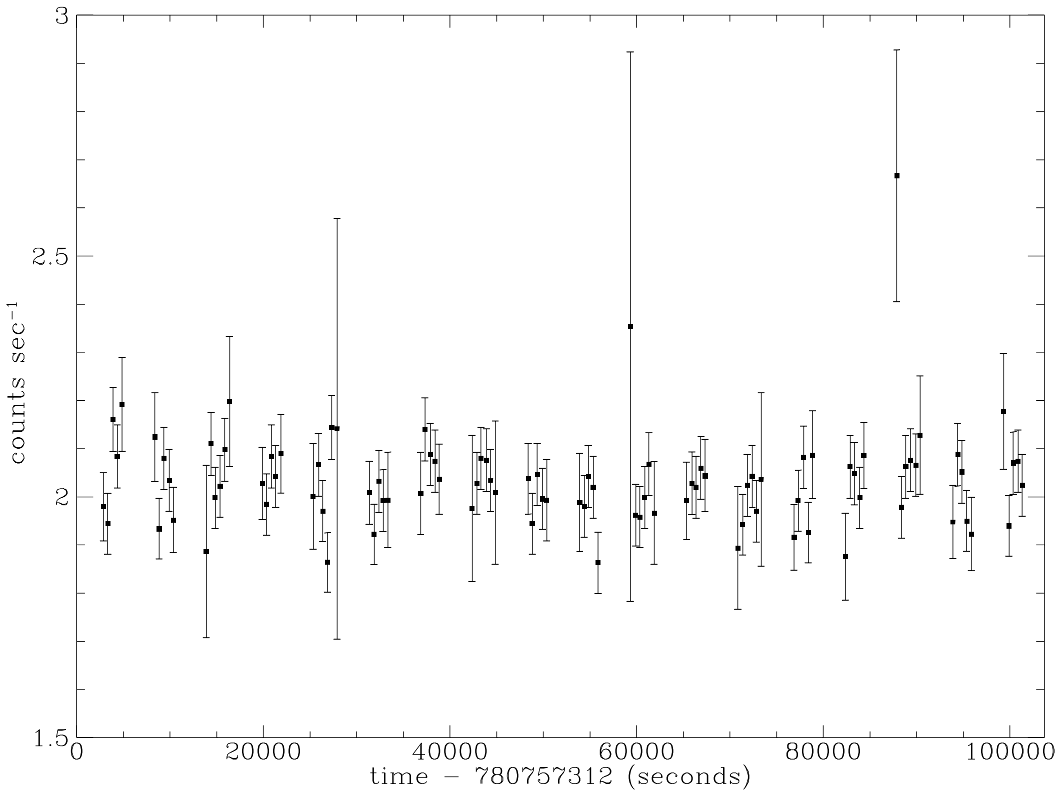

Figure 2-6: Light curve of a portion of a SW spectrum; same data as in figure 2-5, with errors.

required are the observation start and end times, a time associated with a phase of 0, and the start and end phases of an interval to be selected or excluded. The output, which can be ASCII or an ST table, has the same standard column names as good times tables output from **dqselect**, and is suitable for creating a QPOE time filter string with **tfilter**.

A sample parameter file for **periodint** is shown in figure 2-7, with values set to produce the a table of start and end times for the brightest 30% of a source period whose length is 132 minutes.

Notes on running **periodint**

- The range of `startphase` and `endphase` are defined to be $0 - 1.0$, where maximum brightness occurs at a phase of 0.
- The value of `epoch` must be $\leq$ the value of `startobs`, although they need not be stated in the same units. If you wish to use a time within your observation for `epoch`, you must make `startobs` equal to this time, and discard the portion of the observation prior to it.
- The boolean parameter `include` is used to determine whether the interval between `startphase` and `endphase` should be included in the time intervals generated

```
eu> lpar periodint

      output = "bright"          Name of output file/table
      (epoch = 2448921.0)        time of prior maximum point of period
 (epochunits = "jd")             epoch units [jd|mjd|euvesecs]
  (periodlen = 132.)             length of period
(periodunits = "minutes")        period units [seconds|minutes|hours|days]
 (startphase = 0.85)             start phase of interval
   (endphase = 0.15)             end phase of interval
    (include = yes)              include [yes] or exclude [no] phase interval
   (startobs = 781740329.41216) observation start time
     (endobs = 781794000.46016) observation end time
   (obsunits = "euvesecs")       observation units [jd|mjd|euvesecs]
      (table = yes)              make an ST table ([no] for ASCII)
       (mode = "al")
```

Figure 2-7: Sample parameters for the task **periodint**, to calculate start and end times of the brightest 30% of each phase cycle. The julian date given for `epoch` is a known peak in brightness from a previous observation.

or excluded from them. The task produces a list of start and end times for the "good" times selected, regardless of the value of `include`. For example, the following command line will produce a table equivalent to that produced with the parameters in figure 2-7:

```
eu> periodint notdim startph=0.15 endphas=0.85 include-
```

since excluding the phase from .15 to .85 is equivalent to including that from .85 to .15.

□ The definition of the units "Modified Julian Date" and "euvesecs" is given in the help page. The task **backmon** can be used to produce times which are equivalent to "euvesecs", in that they need only be converted from days to seconds. This is the output in the **backmon** column labeled `MMJD`.

Related topics

◇ The **xray.xtiming** package also has tasks that may be useful. See the IRAF help pages for general description, and section A.1.4 for tips on using these tasks with EUVE QPOE files.