Part I

EUVE DATA PRODUCTS

Standard Data Organization

Throughout this User's Guide, we will assume that your data products are organized on disk in a particular manner. This is the organization that Guest Observers will find when they come to the EGO Center to use the IRAF/EUV packages with their data. For ease and consistency in using the EUV packages and this Guide, we recommend use of this organization at the Guest Observer's home institution as well. If a user prefers a different arrangement, however, all the software tasks will work as long as care is taken to correctly specify full path names when they are required in task parameters. Virtual path names can be set in IRAF as environmental variables. This saves some typing, but remember that the full path name will be expanded whenever the "path$" form is used. This can produce cryptic error message, "cannot access image/file" if the full path length exceeds the buffer length provided by IRAF. It's not that hard to do, so beware of far-flung directories! Treat any path longer than 50 characters with suspicion.

The assumed directory setup is pictured in figure 0-5. It is assumed that all FITS files have been converted to IRAF images, QPOE files, or ST tables, as appropriate, and that the separate directories that were created for the QPOE FITS files by **rtar** have been removed. Use the instructions in chapter 0 sections 0.3 and 0.3.1 now if this is not already done.

There is a top-level directory for each target being reduced, which contains the spectral images, QPOE files, and some ancillary files and analysis products. We will refer to this as the working directory.

The working directory has two subdirectories– *reference* and *observation*. The *reference* directory contains files and tables with information on the EUVE Spectrometer and its calibration, which are used by a number of the software tasks. This set of files is in a single tarfile on the data distribution tape, called *Reference* . The directory will also contain a file called *pipeline*, which was used in running the EGO Ceneter event pipeline on the data. The *pipeline* file should be moved to the directory containing the data products and observation log if it is to be used in re-running the comprehensive event pipeline task (see chapter 7).

When the files are tarred from the tape, the raw telemetry tables, which are named *table0.tab, table1.tab,...* through *table7.tab*, are placed in the subdirectory *observation*. *Table0[1-7]* started out in the *observation engineering* tarfile, and *table0.tab*, which contains all the event data, comes from the *observation/science* directory. There should also be a table directory file called *observation*.

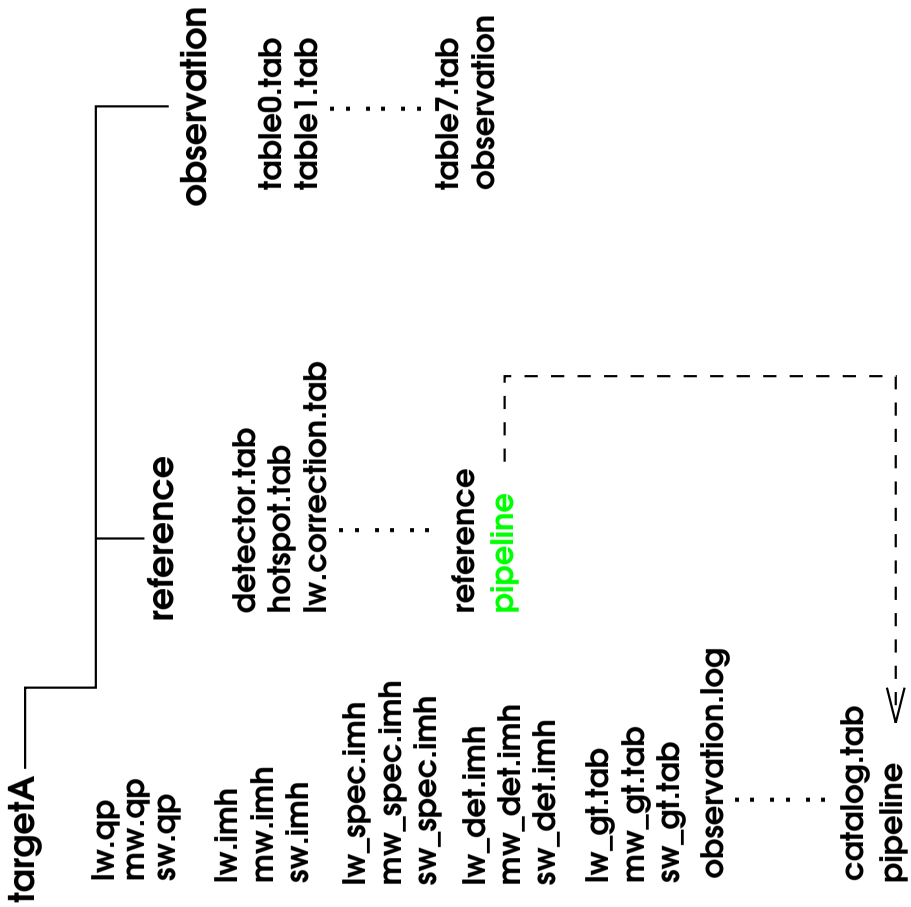

Figure 0-5: Suggested GO data directory structure for use with IRAF/EUV packages.

Chapter 1

ONE-DIMENSIONAL EUVE SPECTRA

You have received three one-dimensional, nominally extracted spectra, which were prepared by the EGO Center as a baseline reduction of your observation data. While these spectra have not been optimized, and it is recommended that all GO's re-extract their spectra from the QPOE files or two-dimensional spectral images, a look at the nominally extracted spectra can be a helpful starting point. Most GO's will do some reprocessing of the data, and will want to work with the two-dimensional spectral images and raw data tables. GO's should examine the 1-d spectra carefully, and read the description of EGO Center processing steps in Appendix B as they examine their data, to better understand what the data look like and why. Appendix B also contains some remarks about instrumental effects that cannot yet be removed from the data. We urge you to read these now, and then review them later, when you are more familiar with the data and its processing.

The spectra have been extracted from 2-dimensional spectral images, which are in turn derived from QPOE files, which are described in the next section. The QPOE files contain all photon event data, and have been processed to remove aspect changes and detector distortions and apply a wavelength solution. The functional difference between the 2-d spectral images and the QPOE files is that QPOE files are event lists with timing information, and the images have the events binned into an array, which facilitates display and manipulation. We will assume throughout this Guide that users are reasonably familiar with the display and manipulation of ordinary IRAF images. For now, you can think of the QPOE files as equivalent to any other 2-d image, like a ccd image, except that they occupy more disk space and take longer to access.

The spectra are 1-d images containing a background-subtracted, pixel-by-pixel summation over a standard aperture height perpendicular to the spectral direction. Background subtraction is performed using values obtained by averaging over a section of the image near the spectrum. The treatment follows the process described in section 3.3.3.1 of the *EUVE Guest Observer Handbook* (2nd release, May, 1993), using a standard background-to-spectrum height ratio of n=5.

See Appendix B for more detail and a cartoon of the apertures used.

1.1 Displaying the Spectra in IRAF

The files called sw_spec.imh, mw_spec.imh, and lw_spec.imh in your working directory are the 1-d spectrum header files. You can check the location of the image pixel files by typing:

```
cl> show imdir
```

You can examine the image headers to see the pointers to the pixel files and other information by typing:

```
cl> imhead *_spec.imh lo+
```

to display "long" format listings of the image headers.

The image header contains a set of world coordinate system (WCS) parameters, including a definition of the wavelength scale in Å. The flux scale is raw counts vs. wavelength; no time division or flux calibration has been performed.

1.1.1 Plotting in wavelength space with SPLOT

Many of the tasks in **onedspec** can also be used for examining and analyzing EUVE spectra. See the list in Appendix D, §D.1.2. The nominally extracted spectra and 2-d spectral images can be displayed either in pixel coordinates or wavelength coordinates with the task **noao.onedspec.splot**:

```
on> splot sw_spec line=1
```

A typical **splot** graphic display is shown in figure 1-1.

The **splot** tool is a versatile task with many options. Once you have displayed the spectrum in **splot**, you can perform a number of operations, like smoothing (s), zoom (z) and window (w) on the display, toggle between wavelength and pixel scales ($), or restore the original plot (c; r for simple redraw). When you are done with one spectrum, you can create a new image with your results (i) and get (g) any of the others without quitting (q) from **splot**. You can see more options by typing a "?" in your graphics window. Read the help page if you have not used **splot** before.

1.1.2 Plot multiple spectra with SPECPLOT

The task **specplot** can be used to view all three spectra together. It automatically scales the spectral and intensity axes and plots several spectra in a stacked format on your standard graphics device; see the sample output in figure 1-2. **Specplot** also has more sophisticated interactive labeling facilities than many of the other plotting tasks, so keep it in mind for creating illustrations of your final reduced spectra.

Most of the one-dimensional extracted spectra still contain a number of instrumental features and are far from optimized. They may contain regions of over-subtraction, noise near geocoronal resonance wavelengths, and unsubtracted excess counts at the ends. A few GO's may want to proceed directly to analysis after a few elementary operations, such as smoothing or rebinning, but most will want to re-extract their spectra subject to more stringent conditions with tracing and more detailed background subtraction. However, these operations will be easier and more fruitful if you take the time now to continue reading this Part of the Guide, which examines both the QPOE files and the ST tables, the two sources from which the spectra are derived.

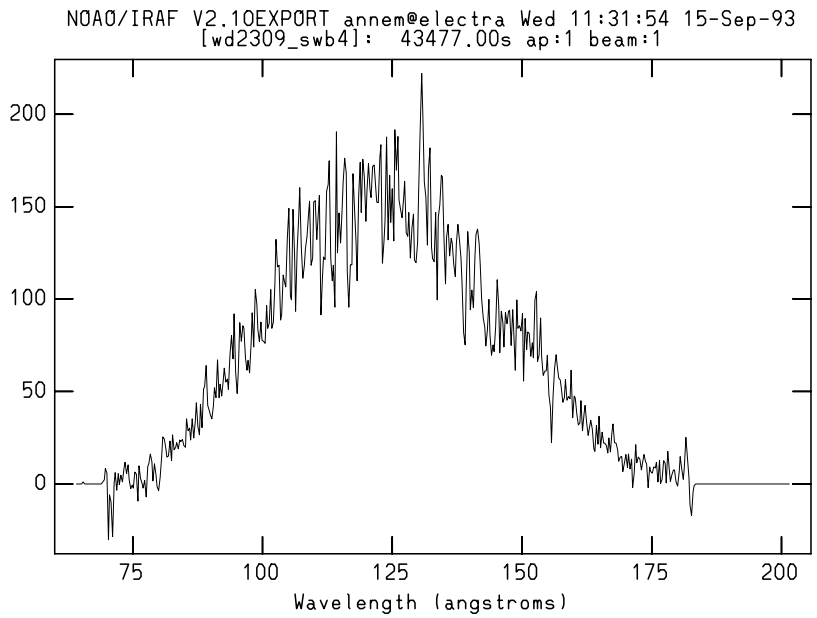

Figure 1-1: 1-d EUVE spectrum from the SW spectrometer displayed with **splot**.

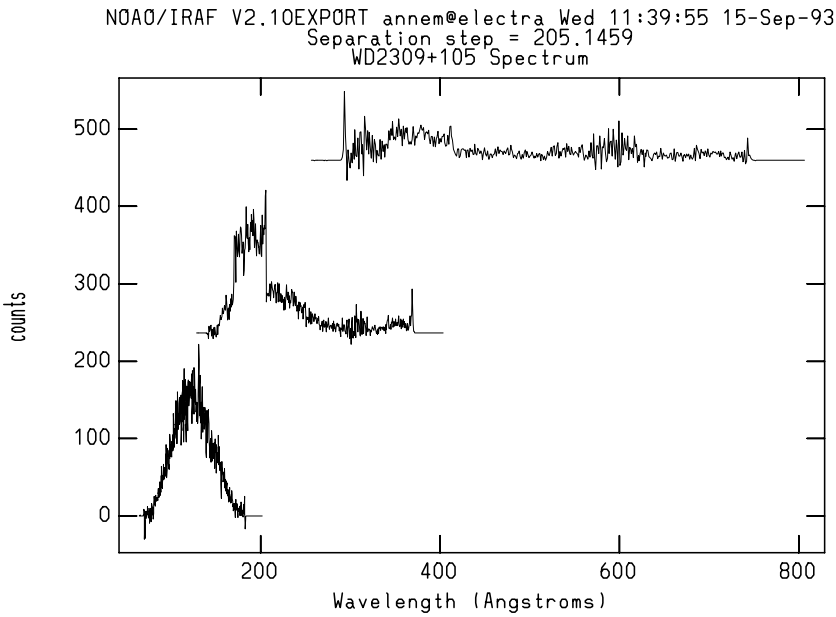

Figure 1-2: Graphic output from **specplot**.

PHOTON DATA IN QPOE FILES

Once you have unpacked the FITS files in the tarfile QPOE according to the directions in chapter 0, you should have three QPOE (Quick Position Oriented Event) files called *sw.qp, mw.qp,* and *lw.qp*: one for each detector. QPOE files can be treated somewhat like two-dimensional detector images; for example, they can be input to any of the tasks in the IRAF **images** package. But they are actually event list files; they contain more information than images, which can be selected and accessed through a specialized I/O interface.

Important:

QPOE files produced by the EGO Center under software version 1.3 and older were delivered as direct copies of the disk files. These QPOE files were only usable on machines compatible with the Sun/SPARC architecture. Starting with sofware version 1.4, this is no longer true. The QPOE files are transported in FITS format, and may be read by any FITS reader that can handle binary table extensions.
Two-dimensional spectral and detector images in FITS format, which have been made from the QPOE files, are still provided in the tarfile *Images*. The spectral images are created using **imcopy** on the QPOE to produce 2048 × 2048 IRAF images with the corrected wavelength, and image scales in the image header. The detector images are made by selecting the raw detector coordinates of the events, and so are not spectral images. Each image header also contains the corrected exposure time in seconds, but timing information for the individual events is not preserved in the images. Spectra may be extracted from the 2-d spectral images, if no other data selection is needed. Users who wish to extract spectra from a subset of the data, should make a new image using the QPOE file and data selection filters (see chapter 6).

This chapter will be devoted mainly to introducing QPOE files and their nomenclature. We will describe the particular format of EUVE QPOE files, and provide basic instructions for reading, displaying, and processing QPOE files and their subsets. Many of these procedures can be performed more quickly and easily on the two-dimensional spectral images, but operations which select data based on attributes other than the corrected coordinates (x,y) can only be performed on QPOE files. Chapter 5 in Part II continues, with more thorough and general instructions for using the IRAF/QPOE interface and macro facilities to select and process data subsets.

2.1 Introduction to QPOE Files

QPOE is an event list file format developed for use with photon-counting detectors. The QPOE format was developed by a collaboration between NOAO and the PROS group at the Harvard-Smithsonian Center for Astrophysics (CfA) in Boston. Most PROS tasks for manipulating QPOE

files can be found in the **xray** package. See Appendix D to find help on QPOE's and **xray**, but the topic-oriented general help pages will be the most useful, and most PROS tasks are not fully general, and cannot be used with EUVE QPOE files.

Although QPOE's are frequently used like IRAF images, the QPOE files contain more information. An IRAF image consists of a header file, containing FITS-like records and a separate pixel file with an array of intensity values. In contrast, each QPOE file contains a number of data objects, including global "header" information and a time-tagged list of the individual photon events, sorted in position order, as the name implies. Each line or "event record", of the event list can contain a number of columns, which are referred to as "attributes". The attributes usually include the event coordinates and time. The structure of the event record is specified by the program which creates the QPOE file.

As mentioned above, the position information in the file can be read by many IRAF tasks that usually operate on images. When a QPOE file is accessed as an image by a task like **imheader**, the event list is binned into an image array according to a "key", that indicates which attributes of the event list contain the event coordinates. Since the events are already in position order, a binning on these attributes gives an efficient translation into image array format.

2.1.1 Format for EUVE QPOE files

EUVE QPOE files are created by a pipeline processing program that reads the the photon events and spacecraft aspect from the telemetry and then calculates wavelengths and imaging angles from a standard set of reference data that describe the instruments. The wavelength and imaging angle scales are then linearized, and each event is assigned an (x,y) position in a two-dimensional spectral image. So, by the end of the pipeline, each event has associated with it raw detector coordinates and time, corrected spectral coordinates, and a wavelength and imaging angle.

The EUVE QPOE files delivered to guest observers have the following attributes:

time x y lambda theta dx dy

Each event record consists of the event time in seconds since JD 2440000.5, the corrected x and y coordinates (pixels), the corresponding wavelength (Å) and imaging angle (degrees), and the original detector coordinates, dx and dy. The coordinates can take integer values from 1–2048. Attribute names are case sensitive; EUVE attribute names are all lower case. The wavelength and imaging angle scales are stored in the header area of the file as a world coordinate system (WCS), which can be accessed by IRAF spectral analysis tasks such as those in **onedspec**.

2.1.2 Examining the QPOE files section with IMHEADER

Information about any QPOE file's contents, history, and format may be obtained in IRAF by running the standard image header task **images.imheader**:

```
im> imhead lw.qp lo+
```

This causes the QPOE interface to create a FITS-like list of header cards and send it to **imheader**

to display on the screen. Use of the ".qp" file extension is recommended anytime a QPOE file is to be accessed by an **images** task. An example of running **imheader** on a typical EUVE QPOE file is shown below, in figure 2-1.

Obviously, there are some similarities to a standard image header, and some differences. It is normal for **imheader** to report "[NO PIXEL FILE]", since the photon events are in the same file as the header. Note also that **imheader** can only print the parameter names in upper case. This may be normal for FITS header cards, many QPOE files contain macros, whose names can be upper and/or lower case, and the interface is case sensitive. The function of macros and ways to list and edit them are discussed further in chapter 5, §5.1.1.

QPOE Header parameters fall into three basic categories:

1. **Scalars** – Scalar parameters contain a single number or an ASCII string. They are read directly into the header without modification, and look just like FITS cards. For example, EUVE QPOE files contain cards called OBSERVAT, TELESCOPE and INSTRUME, which convey general information about the file's contents.
2. **Vectors and macro definitions**– QPOE vector parameters consist of arrays of values. The individual array elements can contain numbers, ASCII characters, or more complex structures. For example, every QPOE file has a vector which defines the structure of the event record, and the event list itself is a very long, complex vector parameter. In the **imheader** output, they are represented by the cards `EVENT` and `EVENTS`, which contain descriptive labels and the length of each vector. The position of the default key in the event record is found by looking for the markers ":x" and ":y" in the `EVENT` parameter. The number following `EVENTS` is the number of event records (lines) in the list. The actual vector's contents (the whole event list) are not displayed.

   The vector parameter `QPWCS`, which stores the WCS for the QPOE file, gets special treatment. The QPOE interface translates its contents into a set of header cards that specify the WCS in the usual IRAF image format.

   Macros are similar in appearance, being actually a subspecies of vector. The macro definition itself is an array of ASCII characters which may be quite long, and is not read into the display by **imheader**. Creating and editing macros will be described further when we discuss processing QPOE files in Chapter 5 of Part II.
3. **Quantities calculated when the QPOE file is accessed as an image** For example, `NAXES, AXLEN1, AXLEN2`, and `BLOCKFAC` describe the way the QPOE was translated into a 2-d image array. EUVE QPOE files are translated into a 2048 × 2048 image by default. `QPPFILT01--32` (there may be as many as 32 such items) summarize the event list, the key, blocking factor, and any filters that were used when the file was opened. These values will change if **imheader** is run on a file which was created using a QPOE expression. We will have a more to say about QPOE expressions presently, in section 2.2.3.

2.1.3 QPPAGE Makes an ASCII listing of events

Having looked at the QPOE global parameters, you may wish to see the event list itself, and compare its structure to the information in `EVENT`. You can create an ASCII listing of any QPOE file with

```
lw.qp[2048,2048][short]:
    No bad pixels, no histogram, min=0., max=0.
    Line storage mode, physdim [2048,2048], length of user area 12000 s.u.
    Created Fri 10:11:17 02-Apr-93, Last modified Fri 10:11:17 02-Apr-93
    Pixel file '' [NO PIXEL FILE]
    NAXES   =                    2
    AXLEN1  =                 2048
    AXLEN2  =                 2048
    BLOCKFAC=                    1
    QPFILT01= 'param=events,key=(s8,s10),block=1'
    EXPTIME =                INDEF
    WCSDIM  =                    2
    CTYPE1  = 'LINEAR  '
    CTYPE2  = 'LINEAR  '
    CRVAL1  =                256.
    CRVAL2  =    -1.24000000953674
    CRPIX1  =                   1.
    CRPIX2  =                   1.
    CDELT1  =           0.26953125
    CDELT2  =  0.00121093750931323
    CD1_1   =           0.26953125
    CD2_2   =  0.00121093750931323
    LTM1_1  =                   1.
    LTM2_2  =                   1.
    WAT0_001= 'system=world
    WAT1_001= 'label=Wavelength units=Angstroms
    WAT2_001= 'label = "Imaging Angle" units=Degrees
    QPWCS   = '        opaque[333] World coordinate system'
    EPOCH   = ' 2000.'
    OBSERVAT= ' EUVE'
    TELESCOP= ' DS/S'
    INSTRUME= ' LW'
    DEFATTR1= 'exptime = integral time:d'
    TIME    = '         macro[002] macro definition'
    X       = '         macro[002] macro definition'
    Y       = '         macro[003] macro definition'
    LAMBDA  = '         macro[003] macro definition'
    THETA   = '         macro[003] macro definition'
    DX      = '         macro[003] macro definition'
    DY      = '         macro[003] macro definition'
    EVENT   = '{d,s:x,s:y,r,r,s,s}[019] event type'
    EVENTS  = '         event[2212590] event list'
    DISPAXIS=                    1
    DC-FLAG =                    0
    DET6EXP = '         macro[896] macro'
    LW_GT   = '         macro[896] macro'
```

Figure 2-1: EUVE QPOE file information displayed with **imheader**.

the task **euv.euvtools.qppage**. Edit the parameter file to select an interactive screen display by setting `interactive` to "yes", or make a printed listing by setting `interactive` to "no" and specifying the `device` for the output. Set `first` to select the line where the display begins. **Qppage** supplies a header bar with bucket minimum and maximum information and column headers on each screenful. Several of the macro names defined in the QPOE appear as column headings labeling the various attributes.

In interactive mode, **qppage** supplies commands for scrolling through the file, moving left-to-right on the screen, and other common **page**-like features. Type "?" at any time to see the command set on the screen. Since QPOE files are divided into buckets, there are separate commands for scrolling by screenfuls or bucketfuls. Figure 2-2 shows an ASCII printout of few lines of a typical EUVE QPOE file.

electra

Event	time	x	y	lambda
Global Min	761834768.031	1	10	128.102050781
Global Max	762099004.548	2041	2039	402.978118896
Bucket Min	761835385.008	31	10	132.103302002
Bucket Max	762098856.003	1898	169	383.714355469
1	762075930.875	1123	10	279.319519043
2	762069905.835	203	12	155.290267944
3	762052330.082	512	25	196.986877441
4	761977505.041	550	31	202.010971069
5	761978415.521	1031	65	266.937408447
6	762052432.834	31	70	132.103302002
7	762075930.851	926	76	252.742340088
8	761881424.424	932	79	253.544387817
9	761990438.37	1044	94	268.605865479
10	762057862.659	1020	97	265.370544434
11	762040623.386	1041	100	268.181030273
12	762063791.011	922	102	252.2162323
13	762051835.082	997	102	262.308380127
14	762023873.714	909	103	250.434738159
15	761886956.48	1053	103	269.862518311
16	762058937.667	907	104	250.171707153
17	762041811.506	952	104	256.289367676
18	762063858.115	1020	105	265.369506836
19	761972480.273	809	107	237.002975464
20	762075762.747	844	107	241.67817688
21	761995066.17	1051	107	269.586639404
22	761966793.361	794	109	234.961257935
23	762046099.786	1022	109	265.611877441
24	762040622.778	966	111	258.086639404
25	762075866.395	1030	111	266.781188965

Figure 2-2: Display of a QPOE file in **qppage**. Various commands can be used to shift the window left or right, or page up and down, allowing the user to see each of the attributes.

2.2 Display and Plot the QPOE Files

QPOE files can be input to most image display tasks in IRAF. Filenames should include the ".qp" ending unless you are using a task that expects a QPOE. The QPOE files are quite large, so it is often useful to be able to select a subset for display. The regular image section syntax can be used, but this is relatively inefficient, as IRAF must bin the entire QPOE before cutting out the specified section. This section begins with general instructions for graphic display of the QPOE's, and we suggest that you look at the whole image at least once. Then we introduce QPOE expressions in the context of creating file subsets. The section is completed by sections on plotting and an introduction to a versatile tool that can both interpret expressions and create various graphic displays.

2.2.1 Passing a QPOE to an image tool with DISPLAY

You can use the task **images.tv.display** to view each QPOE file as a two-dimensional image. Figure 2-3 shows a parameter file setup for displaying a short wavelength EUVE QPOE file.

```
          im> lpar display


        image = sw.qp   image to be displayed
        frame = 1               frame to be written into
       (erase = yes)            erase frame
(border_erase = yes)            erase unfilled area of window
(select_frame = yes)            display frame being loaded
      (repeat = no)             repeat previous display parameters
        (fill = no)             scale image to fit display window
      (zscale = no)             display range of graylevels near median
    (contrast = 0.25)           contrast adjustment for zscale algorithm
      (zrange = no)             display full image intensity range
(nsample_line = 5)              number of sample lines
     (xcenter = 0.5)            display window horizontal center
     (ycenter = 0.5)            display window vertical center
       (xsize = 1.)             display window horizontal size
       (ysize = 1.)             display window vertical size
        (xmag = 1.)             display window horizontal magnification
        (ymag = 1.)             display window vertical magnification
       (order = 0)              spatial interpolator order (0=replicate, 1=line
          (z1 = 0)              minimum graylevel to be displayed
          (z2 = 200)            maximum graylevel to be displayed
     (ztrans = "none")          graylevel transformation (linear|log|none|user)
    (lutfile = "")              file containing user defined look up table
        (mode = "ql")
```

Figure 2-3: Sample parameters for **display** with an EUVE QPOE.

This parameter setup will not display pixels with very high intensities, but in most cases, only the "tops" of the stim pin[1] images will be cut off. Using the full grayscale range without scaling should provide enough contrast to make the interesting parts of most images visible, but the value of `z2` may also be limited to increase contrast for low-level areas.

Next, start the local image display utility. For example, at the EGO Center, *SAOimage* is often used, and is started by typing:

```
dr.ego% saoimage &
```

in a window with a UNIX prompt. To display the image, type:

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

Note that the string containing the filename and blocking command must be enclosed in (single or double) quotes. This is necessary when using any square-bracketed QPOE expression: to keep the CL from mis-interpreting information intended for the QPOE/IRAF interface. The second command line parameter is the frame number of the image display tool being used.

In this example, **display** will first block the image by a factor of four in both dimensions before sending it to the display tool. Blocking speeds the task up considerably, since the display will take less time. This blocking also allows all of the image to appear on a typical 512 X 512 image display. Figure 2-4 shows a typical EUVE QPOE file displayed in *SAOimage*.

If you wish to display the QPOE file without blocking, you may wish to set the IRAF environment variable *stdimage* to allocate a buffer large enough for a 2048 pixel image:

```
tv> set stdimage = "imt2048"
```

There are many other ways to use the **display** task, to enhance some features or compress the intensity range using logarithmic scaling, apply an automatic scaling which selects intensities near the mean, or look closely at a small range of intensities. If you are not familiar with **display**, you may want to experiment with it until you can predict the appearance of the display reasonably well from the parameter settings. Close reading of the help page can be useful.

2.2.2 World Coordinate display with WCSLAB

The special task **images.tv.wcslab** will interpret a world coordinate system and overplot it on your image display. This is often useful for locating regions and features of the image in the WCS coordinates. Use **wcslab** after you have put a QPOE file into **display**, but be aware that it is rather fussy. Here is a short step-by-step procedure:

[1] "Stim" pins provide periodic, artificial detector events throughout the observation, and are used to monitor the detectors.

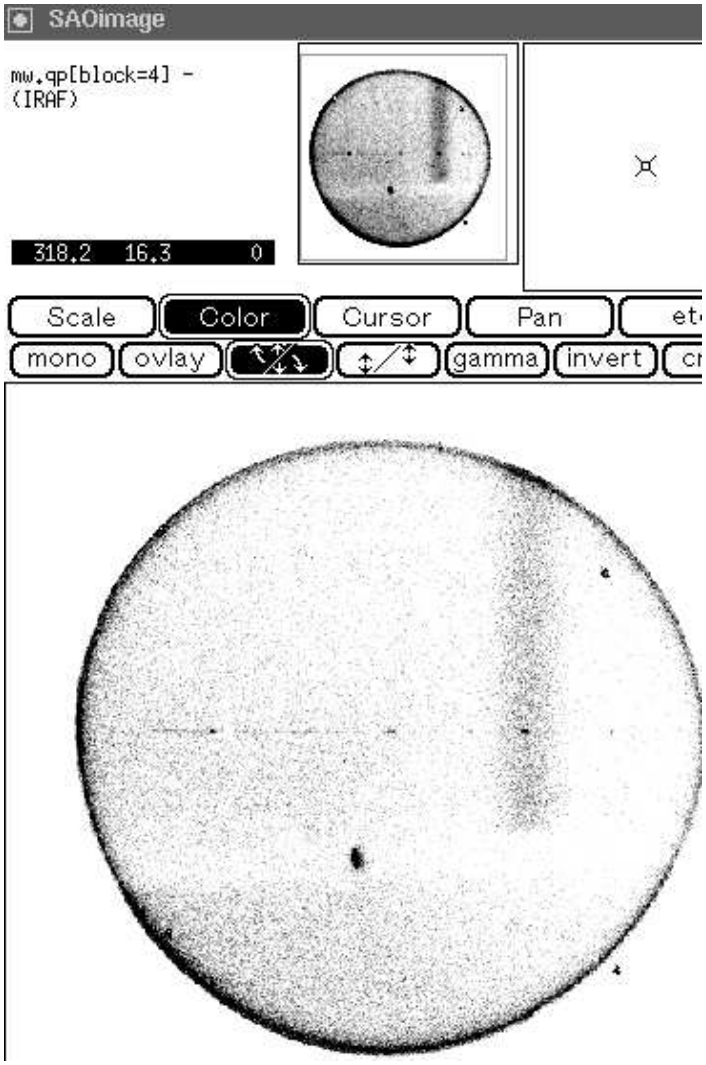

Figure 2-4: EUVE long wavelength QPOE file displayed in *SAOimage*

Show the WCS on the display tool

- Your displayed image must not be blocked, as **wcslab** reads only the QPOE global parameters, and cannot compensate for command line blocking. Set the number of pixels in your standard image display to match the size of the EUVE QPOE files by setting the variable *stdimage* as in section 2.2.1 above. If you are using *SAOimage*, this will cause the mouse position to be displayed in unblocked coordinates.

- Display your QPOE file in the image display without blocking:

  ```
  tv> display mw.qp 1
  ```

  make sure the entire image is visible in the frame.

- Edit the **wcslab** parameters, as shown in figure 2-5. The parameter `wcspars` should be empty, since you wish to use the WCS definition from the QPOE file. The parameter `wlpars` is actually another parameter set that governs axis plotting and labeling. The default values should be sufficient, so leave this line blank also. For later reference, the **wlpars** can be edited from within the **wcslab** parameters by positioning the cursor on the `wlpars` line and typing ":e". After editing, type ":q" to return to the **wcslab** parameter file.

- To run **wcslab** type:

  ```
  tv> wcslab mw.qp 1
  ```

  where you supply the "image" name and frame number, the same as in **display**. **Wcslab** reads only the WCS information, and plots coordinate axes for the entire image space in the display. This is why you need to display the entire image; **wcslab** assumes the internal value of `BLOCKFAC`, which is 1.

- You can also adjust the appearance of the axes by editing the parameter file *wlpars* after running **wcslab**. Typing: `wlpars` will open the parameter file for editing.

2.2.3 Using the QPOE Interface I: QPOE expressions

After you have examined the QPOE files in their entirety, you may want to take a closer look at a section of the file, or look at the image from a fraction of the observation. Because the QPOE format can store many pieces of information about each event, it is possible to create QPOE "sections" with masks on attributes other than the position, such as the event time. In this section, we will introduce these capabilities and describe them in some detail. These methods for handling QPOE files can save considerable processing time, by binning only the desired portion of the event list.

The QPOE interface supports powerful data selection capabilities for all attributes of any QPOE file, with built-in facilities to specify various subsets of a QPOE file on a task command line. There are several types of optional expressions that can be used to change the key or blocking or to "filter" on any attribute. The most general syntax of a QPOE file specification is:

```
tv> lpar wcslab


   image = "mw.qp"          Input image
   frame = 1                Default frame number for image display
 (usewcs = yes)             Use the world coordinate system definition para
(wcspars = "")              World coordinate system definition parameters
 (wlpars = "")              World coordinate system labeling parameters
   (fill = yes)             Fill the viewport ?
     (vl = INDEF)           Left edge of viewport (0.0:1.1)
     (vr = INDEF)           Right edge of viewport (0.0:1.0)
     (vb = INDEF)           Bottom edge of viewport (0.0:1.0)
     (vt = INDEF)           Top edge of viewport (0.0:1.0)
 (append = no)              Append to an existing plot ?
 (device = "imd")           Graphics device
   (mode = "ql")
```

Figure 2-5: Parameters for **wcslab**, using the existing WCS definition in the "header" of *mw.qp*.

'*filename*.qp[*QPOE expression*]'

A QPOE expression consists of a comma separated list of *keyword=expr* pairs. Not all the legal keywords will be described here, but some of the most useful ones are: `key, block, rect`, and `filter`. The four types of expressions are summarized in table 2-1. Combinations of expressions are permitted within the same set of [ ]'s.

The `key` expression has already been introduced; it is used to determine which attributes are used to define the position of an event when binning the event list into an image. The *expr* used with key should define a comma-separated pair of type short attributes, which will be used as the "x" position and "y" position of each event, or in IRAF image terms, the line and column in the image where the event is to be counted. For example, specifying `[key=(dx,dy)]` would direct the QPOE interface to create an image with the events binned according to their raw detector coordinates rather than the default key coordinates, x and y. This image is supplied on your datatape as *(s,m,l)w_det.FITS*.

For `block` expressions the *expr* is a positive integer setting the number of pixels the QPOE interface sums together when binning the events. This overrides the default value of 1 for EUVE QPOE files. For example, if `[block=4]` is specified, an EUVE QPOE file would be translated to a 512×512 image, rather than the default of 2048×2048. As such, the `block` keyword is only meaningful when the QPOE file is being accessed as an image, such as for **imcopy** or **display**.

The `rect` keyword allows one to specify a rectangular region of the QPOE file which is to be extracted. The *expr* is a square-bracketed region specification like that of an image section. For example, the command

```
im> imcopy 'test.qp[rect=[100:199,*]]' test.imh
```

will produce an image which has a size of 100×2048 from an unblocked EUVE QPOE file. One difference is that single row or columns are not allowed. A range must be given, so [*,5] won't work; use [*,5:5].

Also, since the `rect` expression is used to select a subset of the event list before binning, `rect` is applicable even when not accessing the QPOE file as an image; *e.g.*, with the task **qppage**. This enables the user to list the events of a small section of the 2-d spectral image, if desired.

The keyword `filter` specifies an event attribute filter to be applied to the QPOE when it is accessed. A range of values can be specified for any combination of attributes with a modified colon notation, as in:

```
im> imcopy 'sw.qp[filter=(time=123:456,lambda=(310:560,600:))]'
myimage
```

In this case the keyword is actually optional, so that one may type:

```
im> imcopy 'sw.qp[time=1234:4567,lambda=(310:560,600:)]'  myimage
```

for the filter in the example above. Like `rect`, `filter` causes the interface to create the image by binning only a selected subset of the event list.

A few notes on using basic QPOE filtering expressions:

- Any expression that contains a comma or an unmatched right brace, bracket, or parenthesis, *must* be surrounded by parentheses, or QPOE will misinterpret the expression.
- Blocking is performed after `rect` and `filter` operations, so `rect` and `filter` expressions are written in unblocked coordinates. Any references to sections of displays or images made from a blocked QPOE file must then 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 also make a region filter equivalent to a `rect` expression by using an attribute filter directly on the attributes in the default key. However, this will create a (2048×2048) image with empty pixels outside the desired section, rather than a smaller image.
- Be careful to check that any image you make by filtering a QPOE file contains the data subset you intended. For example, if a `key` specification comes first, it defines which set of coordinates `block`, `filter` and `rect` operate on. If the `key` is given at the end of the expression, events will be selected based on the default key, but binned according to the specified key.

Warning to users of IRAF 2.10.2 and earlier

There is a bug in the IRAF/QPOE interface that makes it impossible to specify multiple filters on a single attribute, of the form: filename.qp[d0=(a:b)...d0=(c:d)] It turns out that the interface *does not* merge filters reliably. Tasks run on such expressions without apparent errors, but the number of events processed is not consistent. A single filter may specify a large number of intervals, as long as the attribute name is only used once. This will become an important consideration in chapter 6, on data selection, which uses the "time" attribute as an index for many kinds of filters. The bug should be fixed in the 2.10.3 release, which is expected in the fall of 1993. Workarounds for this problem involve using the IRAF/EUV task **tfilter** to form an intersection of multiple filters for a single attribute. This process will probably not be necessary for initial QPOE inspections, so it is not described here. See sections 5.1 and 6.1, on QPOE macros and time filters.

Table 2-1: Basic QPOE file expressions

keyword	**expr**	**function**	**examples**
key	pair of short integer attributes	define event locations used in processing	mw.qp[key=(s8,s10)] mw.qp[key=(dx,dy)]
block	single integer	define bin size of image in QPOE "pixels"	sw.qp[block=4]
rect	bracketed ranges for current key attributes	select rectangular subregion	sw.qp[rect=[57:230,500:524]]
filter (kywd optional)	attr=(comma-separated ranges) including: :a (attr $\leq$ a); a:b (a $\leq$ attr $\leq$ b); b: (attr $\geq$ b); use ! to exclude	select values of attr	lw.qp[time=(1234:4567)] lw.qp[s10=(100:144,160:)]

A few more examples:

1. Make an image of a QPOE file with no filtering, using the default blocking:

```
im> imcopy test.qp test.imh
```

2. Make an image of a QPOE in raw detector coordinates and block the resulting image by a factor of 2. The **key** expression comes first, and the blocking is then performed on the detector coordinates. The output image will be 1024 × 1024 pixels in size.

```
im> imcopy 'test.qp[key=(dx,dy),block=2]' test.imh
```

3. Filter the QPOE according to the x coordinate. The only events that pass the filter are those from the intervals (100–1200) and (1400–2000).

```
im> imcopy 'test.qp[x=(100:2000,!1200:1400)]' test.imh
```

4. Block a central strip of a QPOE file in detector coordinates and copy it to an image.

```
im> imcopy 'sw1.qp[key=(dx,dy),block=4,rect=[*,1010:1050]]'
sw_blkspec.imh
```

2.2.4 Making plots of QPOE sections

EUVE QPOE files are compatible with many other spectrum plotting and analysis tools in IRAF. Most of the tasks in the **plot** and **onedspec** packages will accept QPOE files as input, and will reconstruct the wavelength scale from the WCS (world coordinate system) in the QPOE file. If you find yourself plotting the same QPOE subset repeatedly, then you might want to store it in an image and continue to use the plotting commands on the image to save processing time.

Both the interactive tasks **plot.implot** and **onedspec.splot** are handy for looking at the QPOE's. When used together with the right QPOE file expressions, they can produce many useful plots directly from the QPOE files. For example, plotting sections can help you decide which parts of a QPOE you want to save in an image for further processing. **Splot** will sum a number of lines, plot the sum, automatically put the wavelength scale on the x-axis, and allow you to smooth and rebin the spectrum. Before using **splot** on a two-dimensional image, set the number of lines to be summed by editing the parameter file for the **onedspec** package.

To plot a "raw" spectrum, change the value of the parameter **nsum** to the number of rows you want to plot; for example the spectrum is about 20–40 lines high, so you might set **nsum** to 25. You select the region by choosing which (central) line to plot:

```
on> splot sw.qp line=1024
```

This will give a reasonable approximation of your spectrum if the signal-to-noise is good, except that airglow features may dominate parts of the plot. Note that if you are **splot**ting a section of the image, the plot line should still be specified in physical coordinates, not relative to the section boundaries.

Implot is a general-purpose interactive plotting task, which will display an average of any number of lines or columns in the image, and has good facilities for zooming in and changing the plot

scales. To display in world coordinates in **implot** and other tasks in **plot**, type ":w world" in the graphics window after starting the task.

These techniques are useful for examining different areas of the image, and for previewing QPOE files as modified by the various types QPOE expressions discussed in section 2.2.3 above.

2.2.5 Having it all with IMEXAMINE

Another very versatile tool which you may want to try after you have some proficiency with the tasks above is the task **tv.imexamine**. Although it is a little bit slow to start on QPOE files, **imexamine** can make a number of different types of plots and do some simple analysis, like gaussian fitting and pixel statistics. This is done by putting IRAF in cursor mode and supplying commands that summon the other IRAF image plotting tasks.

To begin, simply type:

```
tv> imexam
```

This starts the task, and puts it in a quiescent state, waiting for your first command. If you haven't already loaded the desired image into your display, type 'd' for display. Imexamine will then prompt you for an image name. Type the QPOE file name with the extension, and again a blocking factor will speed things up at no great loss of resolution. There is no need to use quote marks when responding to prompts.

Displaying the QPOE takes some time, but once this is done, the user can create other graphic displays and perform analyses by choosing from a number of commands, many of which call other familiar graphics tasks. Text results appear in the alphanumeric window, and can be saved to an ASCII log if desired. Plots appear in a separate graphics window. A list of the key commands is shown in table 2.2.5.

The commands marked with a $\star$, are controlled by separate parameter files that specify the image section which **imexamine** sends to the task, and some set-up parameters. These files are named "`key`*imexam*", where `key` is the letter of the key-binding. They can be edited either beforehand or after entering **imexamine**, using the ':' notation. There are a number of other colon commands; see the help page for a complete list.

Table 2-2: Imexamine Cursor Keys. Keys marked with a ⋆ have separate parameter files named "`key`*imexam*".

?	Print help
⋆a	Circular photometry tool. Ouputs: col line mag flux sky npixels rmom ellip pa peak fwhm
b	Box coordinates from two cursor positions - c1 c2 l1 l2
⋆c	Column plot
d	Display an image
⋆e	Contour plot
f	Redraw the last graph
g	Graphics cursor
⋆h	Histogram plot
i	Image cursor
j	Fit 1D gaussian to image lines
k	Fit 1D gaussian to image columns
⋆l	Line plot
m	Statistics. Outputs:image[section] npixels mean median stddev min max
n	Next frame or image
o	Overplot
p	Previous frame or image
q	Quit
⋆r	Radial profile plot with gaussian fit and aperture sum values
⋆s	Surface plot
⋆u	Centered vector plot from two cursor positions
⋆v	Vector plot between two cursor positions
w	Toggle write to logfile
x	Print coordinates
y	Set origin for relative positions
z	Print grid of pixel values – 10 x 10 grid

Chapter 3

TELEMETRY IN SPACE TELESCOPE FORMAT TABLES

3.1 Format and Contents of EUVE ST Tables

EUVE satellite telemetry contains a great deal of information in addition to the photon arrival times and coordinates. The status of EUVE instrument and satellite subsystems are constantly reported via a large number of engineering monitors, which are reported as analog voltages or logic signals. At the start of observation processing at the EGO Center, the appropriate frames of telemetry are read from a central archive and restructured as ST Tables. ST Tables were developed to provide a flexible format in IRAF for tabular data, and are used to store the EUVE telemetry as time-ordered lists of events. The tables have a expandable multicolumn format that can be displayed and manipulated with tasks in the **tables** package in IRAF. Since the tables can have a large number of columns, each row can be used to store many events that share the same timestamp, such as a group of engineering monitors which are all reported at the same regular interval.

EGO Center software is used to decommutate and restructure the telemetry. Photon events are separated from engineering monitor data and sorted into separate ST tables. EUVE format ST tables are named *table0.tab, table1.tab....table7.tab*. If you are working at the EGO Center, they should be in a directory called *observation*, one level beneath the directory with the object name.

Tables 1 through 7 contain the health and engineering monitors from the instruments, electronics, and spacecraft, which show the status of many subsystems, and provide information on the operating environment, such as focal plane and detector temperatures and aspect reports. Appendix A gives a list of the most useful monitors and their limits, ordered by table number, and function, and rated on a scale from 1 to 5 of importance to GOs.

Different groups of monitors are reported at different intervals; some are reported every 1.024 seconds, some every 2.048 seconds, some every 4.096, and so on. For compactness, those monitors that are reported at the same frequency are grouped together and stored as the different columns of a single table. Each row contains a timestamp in one of the columns. All the information contained in the telemetry is thus saved in an ordered and easily accessible format which the GO can examine and use as a guide to data quality.

Table0 is by far the largest, as it contains all photon event data from all three detectors. Each row of *table0* represents an event on one of the detectors, and contains the time and detector coordinates of the event. The events are listed in time order in this format.

3.2 Accessing the ST Tables

3.2.1 Reading and printing tables: TREAD and TPRINT

The EUVE telemetry tables are standard format ST tables and all the **tables** tasks will work on them. For example, by using **tread** you can browse through a table. Enter **tread** by typing:

```
tt> tread table#
```

where # is a number between 0 and 7. The .tab extension is not usually necessary. Figure 3-1 shows a typical window of an ST table displayed in **tread**.

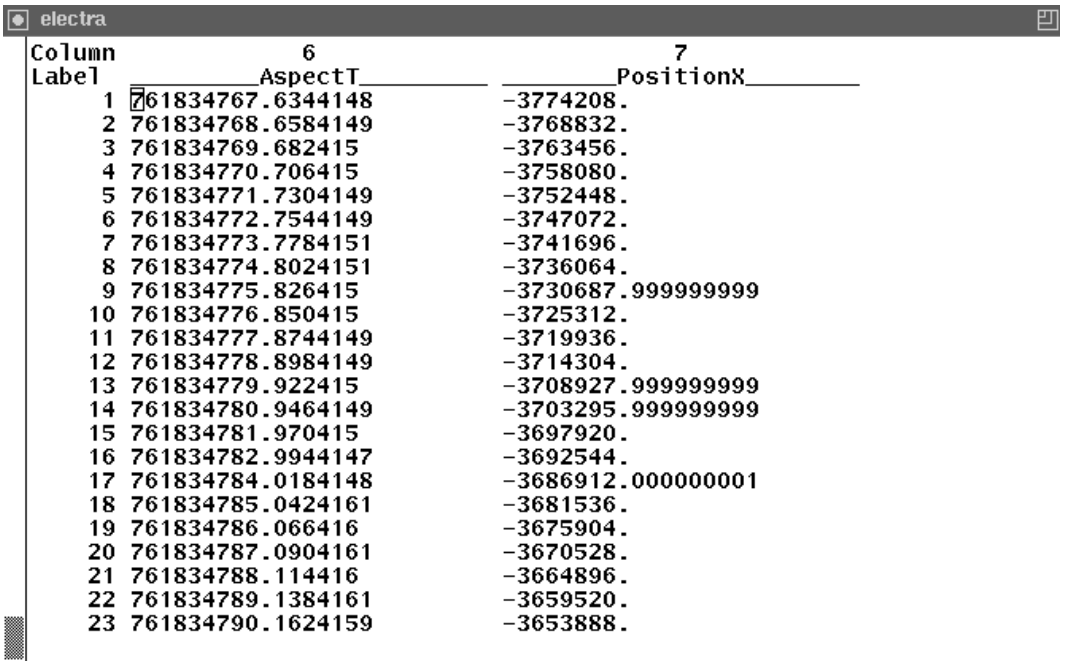

Figure 3-1: ST table displayed in **tread**

Being able to examine (and edit) ST tables will be quite useful in later stages of the reduction, as ST tables are used to store several intermediate data products created by other IRAF/EUV tasks, such as tables of valid observing times for each target.

The task **tprint** prints an ASCII version of any table to a file or printer. The tables are usually too long for a complete hardcopies to be useful, but ASCII conversion is handy if you wish to use part of a table as ASCII input to another task or program. We will see examples of this in later sections. See the IRAF help pages for the **tables** package to read more about ST tables.

3.2.2 DQSELECT creates time-ordered graphical displays

Of course tabular display of any kind of data can be less than informative; like all large data sets, the contents of the ST tables often can be understood better from a graphic display. Any of the monitors listed in a table can be plotted against the time column on your IRAF `stdplot` device with the task **euvred.dqselect**. Start **dqselect** from the directory *observation* in the standard data setup by typing the taskname and a template that includes all the monitor tables:

```
eu> dqselect table[1-7].tab
```

Note that the ".tab" extension is required when using a filename template. The task should notify you as each table is being loaded into memory. When it is ready to continue, **dqselect** will print a list of commands and give you a prompt. To make plots of any monitor, type `display`, followed by the monitor name (case insensitive). For example:

```
Command> display det4adct
```

will result in a plot of the short wavelength detector count rate. The display is shown in figure 3-2, after the `display` command has been repeated for detectors 5 (MW) and 6 (LW). Up to four monitors can be displayed on the screen at once. After they have been displayed, monitor plots can also the swapped in and out of the screen with the commands `hide` and `display`. Another helpful command is `list`, which lists the column names of the table named in its argument. For example:

```
Command> list table1
```

will list the names of all 47 columns in *table1*, several of which are very useful. Refer to Appendix A for a list of the monitor names and their meanings.

Note that since most tables are missing at least a few values, there may be some gaps in the plots. Gaps and what you can do about them are explained in Part II, where we discuss using **dqselect** to make time filters for data selection. For now, use the command `gap` to set the gap to 1, so you will see (practically) everything (but see also 6.1.1).

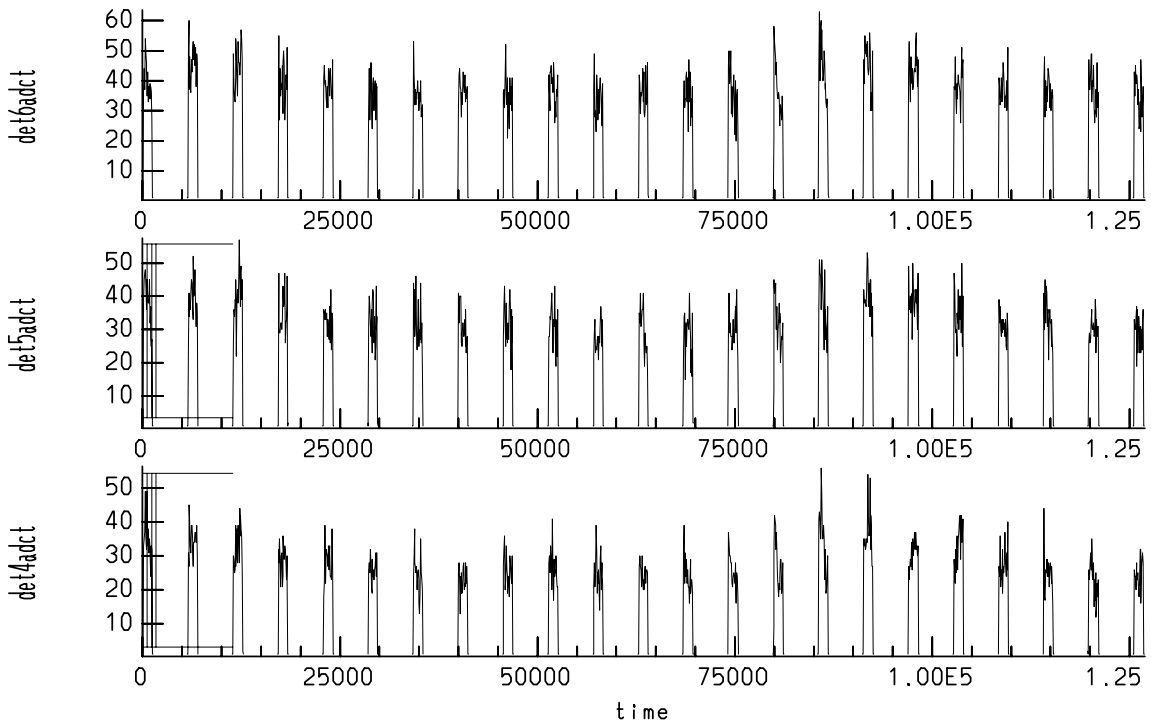

Figure 3-2: Detector count rates displayed with **dqselect**