The EP-FXT Users Guide

v2.0

EP-FXT Science Data Center

Online manual

Contents

1 Introduction **1**
- 1.1 Scope . . . 1
- 1.2 Organization of this Guide . . . 3

2 FXTSoft v2.0 Release Notes **4**
- 2.1 About this release . . . 4
- 2.2 New data analysis tools . . . 4
- 2.3 Effective-area calculations with fxtarfgen . . . 5
- 2.4 Calibration database updates . . . 5
- 2.5 Third-party code and licensing . . . 6

3 FXT modes **7**
- 3.1 The Full Frame mode . . . 7
- 3.2 The Partial Window mode . . . 8
- 3.3 The Timing mode . . . 10

4 Data Files **12**
- 4.1 Level 1 Data Product . . . 12
- 4.2 FXT file naming convention . . . 13
- 4.3 Main columns in FXT FITS event files . . . 13

5 Analysis Guide **16**
- 5.1 Data reduction . . . 16
- 5.2 Point source . . . 20
  - 5.2.1 FF & PW modes . . . 20
  - 5.2.2 TM mode . . . 29
- 5.3 Extended sources . . . 33
  - 5.3.1 Imaging analysis . . . 33
  - 5.3.2 Spectral analysis . . . 48

6 Calibration Files **53**
- 6.1 Introduction . . . 53
- 6.2 CALDB v2.0 release and maintenance . . . 54
- 6.3 Current CALDB v2.0 file inventory . . . 54

6.3.1 Detector geometry and event processing . . . 55
6.3.2 Energy calibration . . . 55
6.3.3 Throughput and spectral response . . . 55
6.3.4 Spatial response . . . 56
6.3.5 Analysis support . . . 57
6.4 Calibration file structure, naming, and selection . . . 57

7 FXT Pipeline: fxtchain tool **59**
7.1 Usage . . . 70

A Installation of FXTSoft v2.0 **73**
A.1 Source-code installation . . . 73
A.1.1 Ubuntu . . . 73
A.1.2 RHEL-compatible Linux (AlmaLinux, Rocky Linux, CentOS Stream) . . . 76
A.1.3 macOS . . . 76
A.2 FXT installation with an existing HEASoft environment . . . 77
A.3 Updating the installed Python tools . . . 78
A.4 Docker installation . . . 79

B FXTSoft v2.0 Command Reference **81**
B.1 Core mission tasks . . . 81
B.1.1 `fxtcheck` . . . 81
B.1.2 `fxtcoord` . . . 81
B.1.3 `fxttimecorr` . . . 82
B.1.4 `fxtparticleidentify` . . . 83
B.1.5 `fxtbadpix` . . . 83
B.1.6 `fxthotpix` . . . 84
B.1.7 `fxtgtigen` . . . 84
B.1.8 `fxtgticorr` . . . 85
B.1.9 `fxtgrade` . . . 86
B.1.10 `fxtootest` . . . 86
B.1.11 `fxtbary` . . . 87
B.1.12 `fxtchain` . . . 87
B.2 Official v2.0 workflow and calibration tasks . . . 87
B.2.1 `fxtarfgen` . . . 87
B.2.2 `fxtarfmerge` . . . 89
B.2.3 `fxtbkggen` . . . 89
B.2.4 `fxtbkgmap` . . . 90
B.2.5 `fxtcaldb-update` . . . 91
B.2.6 `fxte2pi` . . . 92
B.2.7 `fxteefgen` . . . 92
B.2.8 `fxtevtmerge` . . . 92

B.2.9 fxtexpogen . . . 93
B.2.10 fxtexpomerge . . . 94
B.2.11 fxtimage . . . 95
B.2.12 fxtimgmerge . . . 97
B.2.13 fxtlccorr . . . 98
B.2.14 fxtlcmath . . . 99
B.2.15 fxtobsmerge . . . 99
B.2.16 fxtpical . . . 102
B.2.17 fxtplotgrade . . . 103
B.2.18 fxtpsfgen . . . 103
B.2.19 fxtpsfmap . . . 104
B.2.20 fxtpsfmdcopy . . . 105
B.2.21 fxtregmaskfill . . . 105
B.2.22 fxtrmfgen . . . 106
B.2.23 fxtrmfmerge . . . 107
B.2.24 fxtspecmerge . . . 108
B.2.25 fxtsrcdet . . . 108
B.2.26 fxtsrcproducts . . . 114
B.2.27 fxttool-update . . . 117
B.3 Compatibility and support names . . . 118
B.3.1 fxtpsf . . . 118
B.4 Absorbed command names . . . 118

List of Figures

1.1 FXT mirror modules . . . 2
1.2 FXT PN-CCD layout . . . 2

3.1 Partial Window readout region . . . 9
3.2 Timing-mode readout region . . . 10
3.3 Readout-induced periodic signals . . . 11

4.1 FXT observation IDs and data files . . . 12
4.2 FXT event grades . . . 14

5.1 Field light curve . . . 19
5.2 FF source and background regions . . . 23
5.3 FF source light curve . . . 24
5.4 Pattern distribution with pile-up . . . 24
5.5 Pattern distribution after core exclusion . . . 25
5.6 PSF profile and model . . . 26
5.7 TM source and background regions . . . 32
5.8 TM field light curve . . . 33
5.9 M87 in sky and detector coordinates . . . 35
5.10 Exposure maps at 4 keV . . . 38
5.11 OoT image subtraction . . . 40
5.12 Stray light from the Crab Nebula . . . 41
5.13 Blank field before background subtraction . . . 42
5.14 Three-color image of M87 . . . 44
5.15 Point-source detection in 3C 273 . . . 46
5.16 A3571 spectrum . . . 52

7.1 fxtchain data-reduction flow . . . 60
7.2 Input directory hierarchy . . . 67
7.3 fxtchain inputs and outputs . . . 68

Chapter 1

Introduction

1.1 Scope

This document is the user guide and task reference for FXTSoft v2.0. It describes how Follow-up X-ray Telescope (FXT) data are calibrated, filtered, and converted into science products.

Einstein Probe is an X-ray astronomy mission of the Chinese Academy of Sciences (CAS). It carries the Wide-field X-ray Telescope (WXT, 0.5–4 keV) and the FXT. FXT follows up sources discovered by WXT and observes targets selected from the ground. It has two mirror modules, each with 54 nested Wolter-I paraboloid-hyperboloid shells (Figure 1.1). Each module focuses X-rays onto a PN-CCD detector, which records photon energy, position, and arrival time over 0.3–10 keV. The two PN-CCDs are oriented at 90° to one another in the focal plane.

Figure 1.2 shows the layout of the PN-CCD. The PN-CCD has an imaging area and a frame-store area, each containing 384 × 384 pixels. The pixel size is 75 μm × 75 μm in the imaging area and 75 μm × 51 μm in the frame-store area. Charge collected in the imaging area is transferred to the frame-store area and read out in parallel by three CAMEX chips.

This guide describes the principles of FXT data processing and reduction. The FXT Data Analysis Software Package (FXTDAS) produces images, spectra, light curves, Ancillary Response Files (ARF), Redistribution Matrix Files (RMF), exposure maps, and background products. Its tasks follow the FTOOLS parameter convention and operate within an initialized HEASoft environment.

This guide assumes that the data have already been downloaded from the archive and that FXTSoft v2.0, HEASoft, and CALDB v2.0 have been installed and initialized. The maintained online edition is available at https://epfxt.ihep.ac.cn/docs/fxtsoft/.

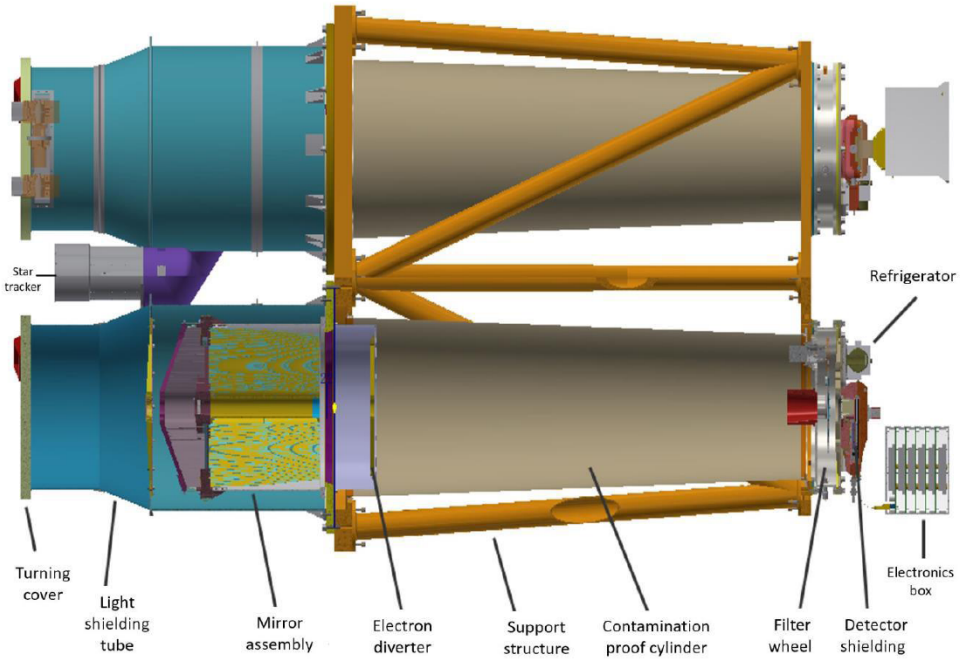

Figure 1.1: FXT mirror modules.

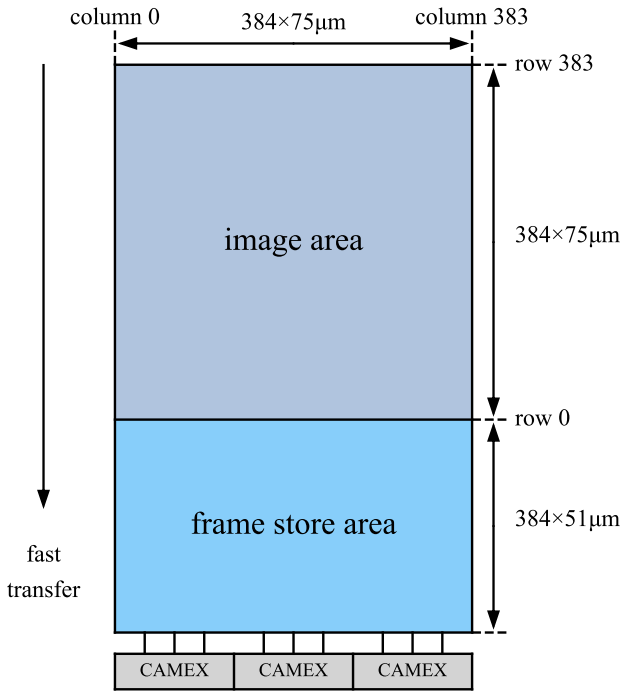

Figure 1.2: Layout of the FXT PN-CCD.

1.2 Organization of this Guide

The Release Notes describe FXTSoft v2.0, command availability, and third-party code and licenses.

FXT Modes describes the observing modes and their implications for data reduction.

Data Files explains the observation layout, file naming, and event columns.

The Analysis Guide gives workflows for point sources, timing analysis, and extended sources.

Calibration Files describes CALDB v2.0 and its maintenance. The `fxtchain` Pipeline chapter documents the routine reduction from Level 1 to Level 2.

Appendix A covers installation and initialization. Appendix B lists the commands and their parameters.

FXT was developed by the Institute of High Energy Physics (IHEP), Chinese Academy of Sciences (CAS), with participation from the Technical Institute of Physics and Chemistry, CAS, and contributions from the European Space Agency (ESA) and the Max Planck Institute for Extraterrestrial Physics (MPE).

Chapter 2

FXTSoft v2.0 Release Notes

2.1 About this release

On 6 November 2026, the EP-FXT team released FXTSoft v2.0 and CALDB v2.0. Building on v1.30, the release adds data analysis tools and incorporates in-orbit measurements into the detector and telescope calibrations. Use FXTSoft v2.0 with the matching CALDB v2.0. After a calibration update, regenerate the science products that depend on it. The EP-FXT release announcement is available in Chinese and English.

2.2 New data analysis tools

Point-source detection `fxtsrcdet` detects point sources with multiscale wavelets or maximum-likelihood fitting. The wavelet method adapts CIAO's `wavdetect`. The likelihood method draws on the eROSITA/eSASS `erbox`, `erbackmap`, and `ermldet` tasks.

Combining observations `fxtobsmerge` combines event lists, images, and exposure maps across observations. It can also combine compatible spectra with their backgrounds and response files. See the `fxtspecmerge` help for the spectral weighting and background options.

Images and masks `fxtimage` bins events into images and can add, subtract, multiply, or divide images. `fxtregmaskfill` builds source-region masks and fills masked pixels.

Light curves `fxtlccorr` scales the background light curve, subtracts it from the source light curve, and propagates the uncertainties.

Particle background `fxtbkgmap` produces maps of the instrumental particle background.

Software and calibration updates Use `fxttool-update` for the Python tools component and `fxtcaldb-update` for the calibration database. Both commands can check for updates, install them, and verify the installed files.

2.3 Effective-area calculations with fxtarfgen

- For point sources more than 3 arcmin off axis, `fxtarfgen` uses an updated point-spread function (PSF) model. Point-source PSF correction is enabled by default (`psfcor=1`).
- The task can calculate ancillary response files (ARFs) for extraction regions with complex shapes.
- For merged observations, the ARF calculation uses both exposure and telescope attitude to weight the response contributions.
- For extended sources, set `extend=1` and choose a uniform, Gaussian, or beta surface-brightness profile, or supply an image template.

2.4 Calibration database updates

Energy scale The energy–channel calibration corrects residual energy offsets after the charge-transfer inefficiency (CTI) correction and accounts for differences between event grades. CALDB also supplies the FXT-B calibration for observations after detector heating.

Energy redistribution Revised redistribution matrices (RMFs) better model the low-energy shelf and the detector's energy resolution. The v2.0 revision `caldb-v2.0-20261106-r1` supplies dedicated Partial Window files labelled `DATAMODE=PW`. Their numerical redistribution matches the Full Frame response.

Effective area Separate `ARFCORR` tables give the effective-area correction as a function of photon energy and time.

PSF and encircled energy Updated PSF and encircled-energy fraction (EEF) models include new tables of off-axis model parameters.

Vignetting The revised vignetting calibration describes how telescope throughput changes with off-axis angle.

Blank-sky observations CALDB now includes blank-sky observation data for background analysis.

See Calibration Files for response selection and calibration validity intervals, and the Command Reference for task parameters.

2.5 Third-party code and licensing

The `fxtsrcdet` implementation includes modified Python translations of CIAO 4.16 `wavdetect`, `wtransform`, and `wrecon`, developed at the Chandra X-ray Center, Smithsonian Astrophysical Observatory, and routines adapted from eROSITA/eSASS, developed at the Max Planck Institute for Extraterrestrial Physics and the Leibniz Institute for Astrophysics Potsdam. Work on response generation also draws on methods and documentation from XMM-Newton SAS.

The CIAO-derived code is distributed under GPL version 3 or later, with `LICENSE.SAO` retained. The adapted eSASS sources are licensed under GPL version 2 or later; the FXTSoft source-detection port is distributed under GPL version 3 or later. These terms apply to the corresponding components, not automatically to all of FXTSoft.

The source distribution includes the original `NOTICE_CIAO`, `NOTICE_EROSITA`, and `COPYING.GPL-3.0` files. They are the authoritative component records for source versions, author credits, modifications, and license terms. Redistribution must retain the applicable copyright and license notices and meet the GPL requirements for providing Corresponding Source. Refer to the included license texts, the CIAO license information, and the GNU GPL version 3 for the full terms.

Chapter 3

FXT modes

Einstein Probe's Follow-up X-ray Telescope (FXT) performs prompt follow-up observations of sources detected by WXT and pointed observations of other selected targets. Table 3.1 summarizes the FXT parameters.

Table 3.1: The FXT characteristics.

FXT parameters	Value
Energy range	0.3–10 keV
Mirror	Wolter-I
Detector	PNCCD
Effective area	600 cm^2 @1.25 keV (on-axis)
Field of view	$60' \times 60'$
Detector elements	384×384
pixel scale	$75\mu m \times 75\mu m$
HPD	$< 30''$ (on axis)
Sensitivity	2.4×10^{-13} erg cm^{-2} s^{-1} (1500 s exposure time)

FXT can select a readout mode automatically according to the source count rate, or use a mode commanded from the ground. Full Frame (FF) and Partial Window (PW) modes retain two-dimensional imaging; Timing (TM) mode trades one spatial dimension for higher time resolution. Table 3.2 gives the timing properties and source-brightness limits. These limits depend on the filter and the acceptable pile-up fraction.

3.1 The Full Frame mode

Full Frame mode records the full imaging area with a 50 ms frame cycle: 49.8848 ms of integration followed by approximately 0.1152 ms of frame transfer. The charge is then read from the frame-store area by three CAMEX chips (CAMEX_L, CAMEX_M and CAMEX_R).

Table 3.2: FXT science mode

Mode	Image capability (x×y)	Readout timing	Flux level (thin filter, pileup<2%)	Fraction of OoT
FF	384 × 384, 2D	50 ms	< 2 mCrab	0.23%
PW	128×61, 2D	2.2 ms	< 40 mCrab	<1%
TM	128×384, 1D	23.68 μs (Read time per row)	< 100 mCrab (spectral analysis) < 1 Crab (timing analysis)	-

In the FF mode, the energies and the positions of the signals exceeding the threshold are measured in the whole image area at a frame rate of 20 frames/s. During the exposure time, the frame area is read out before transfer of the next image to the frame store area.

The pixels are processed on board where the bias is subtracted and the lower level discriminator is applied. However, the events are not reconstructed.

Photons registered during charge transfer produce out-of-time (OoT) events. They are assigned incorrect positions along the readout direction. Because gain and Charge Transfer Inefficiency (CTI) corrections depend on position, their PI values may also be incorrect, broadening spectral features. The OoT fraction is approximately 0.23% in FF mode (Table 3.2).

Initial processing includes:

- Flag bad pixels
- Calculate and Flag hot and flickering pixels
- Assign grade and PI values

FXT also records events in the frame-store area while the imaging area integrates. These events are mainly produced by charged particles and are identified by `fsaevt` in their filenames. They can be used to estimate the instrumental background in the imaging area.

3.2 The Partial Window mode

Partial Window mode reads a central 128×61-pixel region through the middle CAMEX (CAMEX_M). Each 2.2 ms cycle includes approximately 2.1319 ms of integration and 0.0681 ms of transfer. Figure 3.1 shows the readout region.

For this mode, the sub-image can be transferred to frame-store area by 227 serial clock shifts (about 0.0681 ms) and during this time, the OoT events are still registered in the CCD. However, the photons only located in the region of 162–222 rows can be read out and other photons will be discarded. Therefore, the fraction of OoT correlates with the source position. The fraction for a source in the sub-image

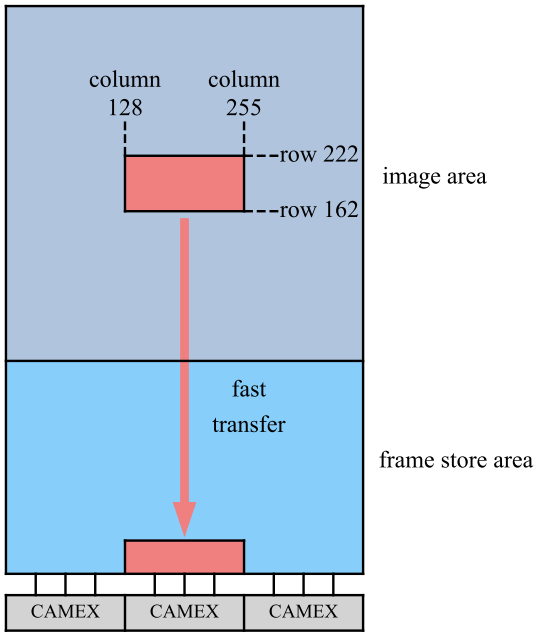

Figure 3.1: Readout region in Partial Window mode.

can be estimated by $f = 0.832\% - (N - 161) \times 0.01364\%$, where N means the Y position of the source and has the range from 162 to 222.

In this mode, there are also OoT events from the outside of the sub-image region and the fraction can be obtained if the source position is known. If the source's position is in rows 0–161, the probability of a photon being recorded is $61 \times 0.3/2200 = 0.832\%$. If the source's position is in rows 223–318, the photon will not be recorded. If the source's position is in rows 319–379, the probability of a photon being recorded is $(N - 318) \times 0.01364\%$, using the same 0.3/2200 fraction per row as above. If the source's position is in rows 380-383, the probability of a photon being recorded is $61 \times 0.3/2200 = 0.832\%$.

The pixels are processed on board where the bias is subtracted and the lower level discriminator is applied. However, the events are not reconstructed.

Initial processing includes:

- Flag bad pixels
- Calculate and Flag hot and flickering pixels
- Assign grade and PHA values

3.3 The Timing mode

Timing mode is used for bright sources and high-time-resolution measurements. Rows are read consecutively at approximately 23.68 μs per row. After 384 rows, the readout node resets for about (6×23.68+200) μs, or 342.08 μs. Figure 3.2 shows the readout region.

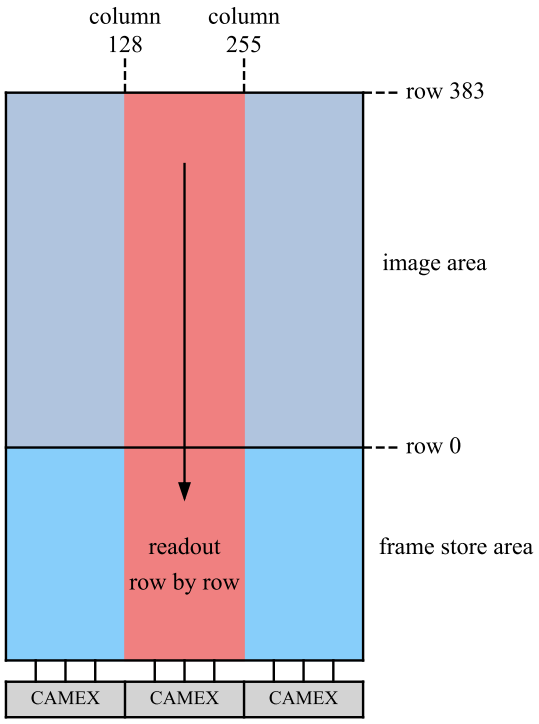

Figure 3.2: Readout region in Timing mode.

Source events are spread along the readout direction, with an enhancement at the source position caused by photons collected during the reset interval.

The recorded frame time and readout order at CAMEX_M are used to reconstruct each event's arrival time during ground processing.

The pixels are processed on board where the bias is subtracted and the lower level discriminator is applied. However, the events are not reconstructed.

Initial processing includes:

- Assign the proper arrival time to each event
- Flag bad pixels
- Calculate and Flag hot and flickering pixels
- Assign grade and PHA values

In the timing mode, there is an inherent periodic signal with a period of approximately 9.4352 ms ($\sim$ 105.986 Hz). Figure 3.3 illustrates the profile of this periodic signal. It arises from the readout cycle and event-time reconstruction. Apparent variability near this frequency should be checked for an instrumental origin.

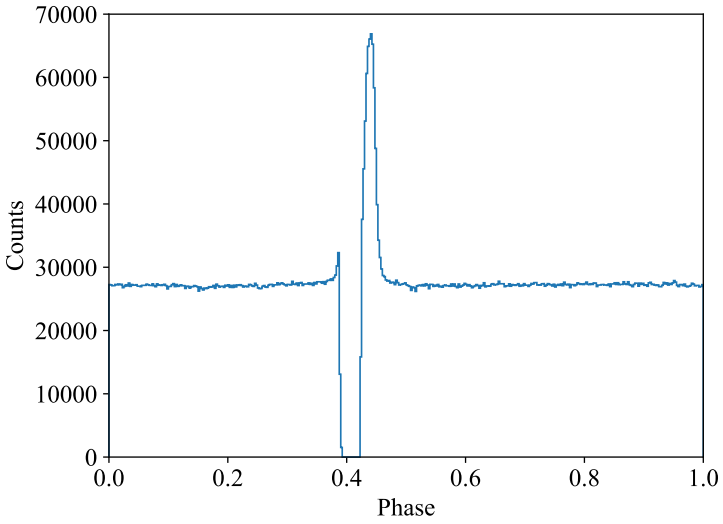

Figure 3.3: Periodic signals caused by the readout system.

Charge loss during transfer depends in part on the incident flux, so the energy-channel relation can change at high count rates. For TM spectral analysis, the recommended source flux is below 100 mCrab; spectra above this level may have additional systematic errors.

Chapter 4

Data Files

FXT records science data in Full Frame (FF), Partial Window (PW), and Timing (TM) modes, described in FXT Modes. This chapter explains the observation directory, file naming convention, and principal event columns.

4.1 Level 1 Data Product

Scientific analysis starts with the Level 1 data products. Each observation directory is named after its Observation ID and is organized as follows:

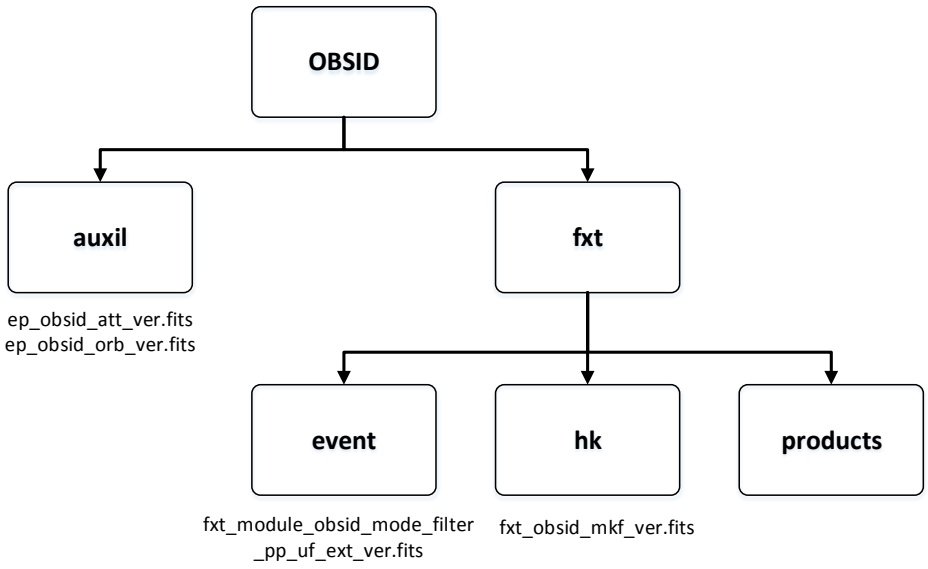

Figure 4.1: FXT observation IDs and data files. From Zhao et al. (2025).

The `auxil` directory stores attitude and orbit data. Under `fxt`, the `event` directory contains the science event files, and `hk` contains the make-filter file (MKF).

4.2 FXT file naming convention

The file name format for the FXT science files uses the following convention:
fxt_[module]_[obs_id]_[mode]_[filter]_[pp]_[uf]_[ext]_[lev].fits.
The filename contains several keywords:

- [fxt] is a prefix to indicate the mission name.
- [module] identifies FXT-A or FXT-B (a/b).
- [obs_id] is the 11-digit Observation ID.
- [mode] identifies the readout mode. The standard science modes are ff, pw, and tm; the current `fxtchain` filename matcher also recognizes auxiliary codes df, dt, dp, sv, and ds.
- [filter] identifies the filter position. Codes 01/02/03 are commonly used for the thin, medium, and hole positions; other mission-defined filter codes can also occur and should be read from the file metadata.
- [pp] identifies the observing context. `po` denotes pointing data and `sl` denotes slew data when present in an archive product. The current `fxtchain` matcher processes `po` event files; raw `sl` files are not handled by this matcher. For already prepared slew event files, `fxtobsmerge inputmode=slew` accepts an explicitly supplied event list for merging; see that task's help for the file-level workflow. This chapter does not document a raw-slew reduction chain.
- [uf] marks an unfiltered file.
- [ext] indicates the file type and is typically set to 'evt'. However, it can sometimes be set to 'fsaevt', which means events originating from the frame-store area.
- [lev] gives the file version.

The file version has three characters. The first indicates data integrity (2 = incomplete, 3 = complete, and 6 = merged); the second records the data version, or the number of productions under a given software version; and the third identifies the software version.

4.3 Main columns in FXT FITS event files

This section describes the important columns found in the event files.

RAWX and RAWY give the CCD pixel coordinates recorded on board. Ground processing calculates the focal-plane coordinates DETX and DETY and the sky coordinates X and Y. In FF and PW modes, both RAW coordinates contain spatial

information, with RAWY along the readout direction. In TM mode, RAWY is a counter incremented each time a row is read out.

The TIME column contains the event time in seconds after the reference time.

`CCDFrame` contains the frame number.

Channel records the detector channel, ranging from 0 to 4095. Energy calibration produces ENG, the event energy in keV, and PI, the pulse-invariant channel used for spectra. PI channels do not have a fixed width of 10 eV; use `fxte2pi` to convert energy limits to PI channel limits.

GRADE is assigned by `fxtgrade` from the event's pixel pattern. Figure 4.2 defines the FXT grades.

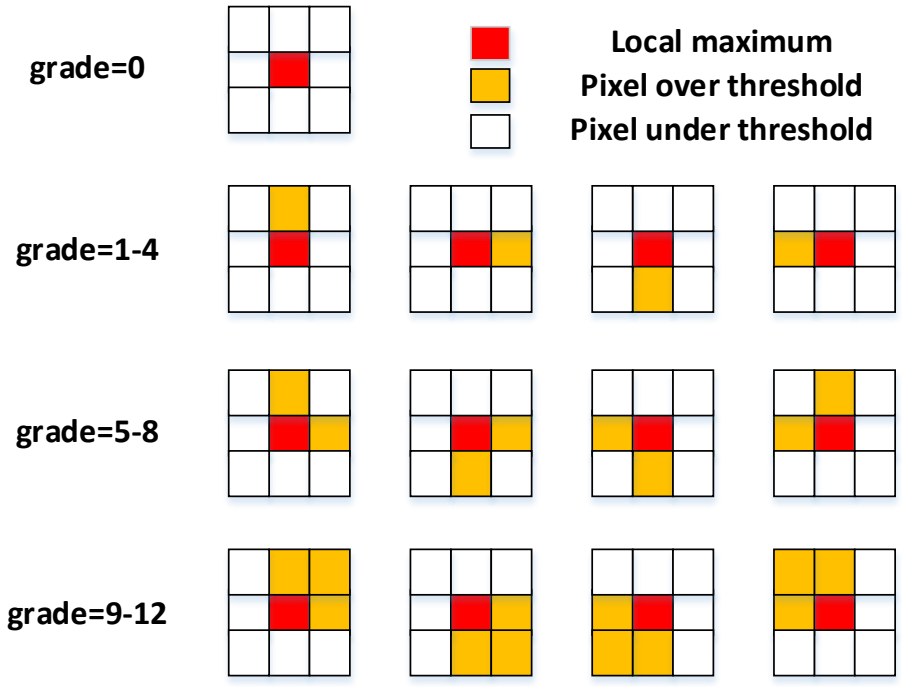

Figure 4.2: FXT event-grade definitions. From Zhao et al. (2025).

The STATUS column contains a bit mask flag describing the quality of the event. The column is populated during the data calibration in the "fxtgrade" step. The possible flags are:

- GOODSTATUS 0x00: good event
- GROUNDSTATUS 0x01: bad pixel from on-ground CALDB bad pixel file
- BOARDSTATUS 0x02: disabled pixel from on-board software

- USERSTATUS 0x04: bad pixel in the file provided by the user
- HOTSTATUS 0x08: pixel hot found in the current observation
- FLICKSTATUS 0x10: pixel flickering found in the current observation
- NEIGHBORSTATUS 0x20: event in the inner 3×3 neighborhood of a bad pixel
- NNEIGHBORSTATUS 0x40: event in the outer ring of the 5×5 neighborhood of a bad pixel

The standard XSELECT screening expression `STATUS==b0` rejects both neighborhood flags and therefore uses the 5×5 bad-pixel footprint in exposure-map generation. A selection that rejects the direct bad-pixel and inner-neighborhood bits but retains 0x40 uses the 3×3 footprint.

Chapter 5

Analysis Guide

5.1 Data reduction

Start with the Level 1 event files described in Data Files. Calibrate and screen them to produce Level 2 event files, then extract images, spectra, and light curves. The cleaned events can be reused while the calibration and screening choices remain unchanged. Reprocess the original inputs when a relevant software or CALDB update changes the calibration.

The following example uses optional symbolic links to shorten the filenames. You may use the original filenames directly. Start in the `fxt/event` directory:

```
ln -s fxt_a*uf* fxta-ori.fits
ln -s fxt_b*uf* fxtb-ori.fits
cd ../hk
ln -s *hk* hk.fits
ln -s *mkf* mkf.fits
cd ../../auxil
ln -s *att* att.fits
ln -s *orb* orb.fits
cd ../fxt/event
```

method1: FXTSoft batch reduction. The `fxtchain` task runs the standard calibration and screening sequence. Replace `input_obs` with the observation directory and `reduced_obs` with the output directory:

```
fxtchain indir=input_obs outdir=reduced_obs
```

method2: explicit FXTSoft and HEASoft steps. Run the individual tasks below to inspect intermediate products or apply custom screening.

The following example produces a Level 2 event file for FXTA:

- *fxtcoord* updates the SKY coordinate of each event using the latest calibration file and recalculates the GTI of events based on the event file and attitude file

```
fxtcoord evtfile=fxta-ori.fits attfile=../../auxil/att.fits outfile=fxta-coord.fits
```

- For TM mode, correct event times and readout positions before energy calibration. FF and PW data skip this step. Replace `SOURCE_RA` and `SOURCE_DEC` with the target coordinates in degrees:

```
fxttimecorr evtfile=fxta-coord.fits outfile=fxta-timecorr.fits \
    attfile=../../auxil/att.fits ra=SOURCE_RA dec=SOURCE_DEC
```

- Apply energy calibration with *fxtpical*. For FF/PW data, use the coordinate-corrected file:

```
fxtpical evtfile=fxta-coord.fits outfile=fxta-pi.fits
```

  For TM data, use the time-corrected file instead:

```
fxtpical evtfile=fxta-timecorr.fits outfile=fxta-pi.fits
```

- *fxtparticleidentify* to identify fake events from cosmic rays:

```
fxtparticleidentify evtfile=fxta-pi.fits outfile=fxta-rp.fits xlength=11 ylength=11
```

  Set both `xlength` and `ylength` to 11 for FF/PW data or 35 for TM data, where particle tracks extend across more rows.

- *fxtbadpix* and *fxthotpix* search bad and hot pixels, respectively.

```
fxtbadpix evtfile=fxta-rp.fits outfile=fxta-badpix.fits
fxthotpix evtfile=fxta-badpix.fits outfile=fxta-hotpix.fits
```

- *fxtgrade* calculates the event grades:

```
fxtgrade evtfile=fxta-hotpix.fits outfile=fxta-grade.fits
```

- *fxtgtigen* produces the Good Time Interval (GTI) file:

```
fxtgtigen mkffile=../hk/mkf.fits module=fxta outfile=fxta.gti expr=DEFAULT
```

  The `expr=DEFAULT` means that the selection criterion for the GTIs is `ELV>5 && DYE_ELV>30 && SAA==0 && STABILITY==3 && ANG_DIST<0.2`.

  Last, the events file can be filtered with selected grades, status, and GTIs with `xselect`:

```
xselect
> Enter session name >[xsl]
xsl:SUZAKU > read event fxta-grade.fits
> Enter the Event file dir >[./]
> Reset the mission ? >[yes]
xsl:EP-FXTA-FF > filter grade 0-12
xsl:EP-FXTA-FF > select event "status==b0"
xsl:EP-FXTA-FF > filter time file fxta.gti
xsl:EP-FXTA-FF > extract events copyall=yes
xsl:EP-FXTA-FF > save events fxta-clean.fits
xsl:EP-FXTA-FF > exit
> Save this session? >[no]
```

Use `copyall=yes` to retain the other event-file extensions. The equivalent HEASoft selection is:

```
ftselect "fxta-grade.fits[events]" fxta-clean.fits "grade >= 0 && grade <=12 \
    && status==b0 && gtifilter('fxta.gti')" copyall=yes clobber=yes
```

Extract images, spectra, and light curves from this filtered event file. Intermediate files may be removed after the products have been checked.

GTI Checking

Inspect a light curve of the full field for background flares that remain after GTI screening:

```
xselect
> Enter session name >[EP] EP  # session name, will be the prefix of later output
> set datadir .
> read events fxta-clean.fits
> set datamode FF/PW/TM   # when the mode are not recognized correctly.
> set binsize 20
> show status
> extract curve
> plot curve
PGPLOT file/type: /xw        # set the display device as X window
PLT> device lc.eps/CPS       # switch the output device to a color PS file: lc.eps
PLT> plot
PLT> quit
> save curve lc.fits clobber=yes    # save curve data with a fits file
> quit
```

More details could be found at the "Timing Analysis" section of the "ROSAT

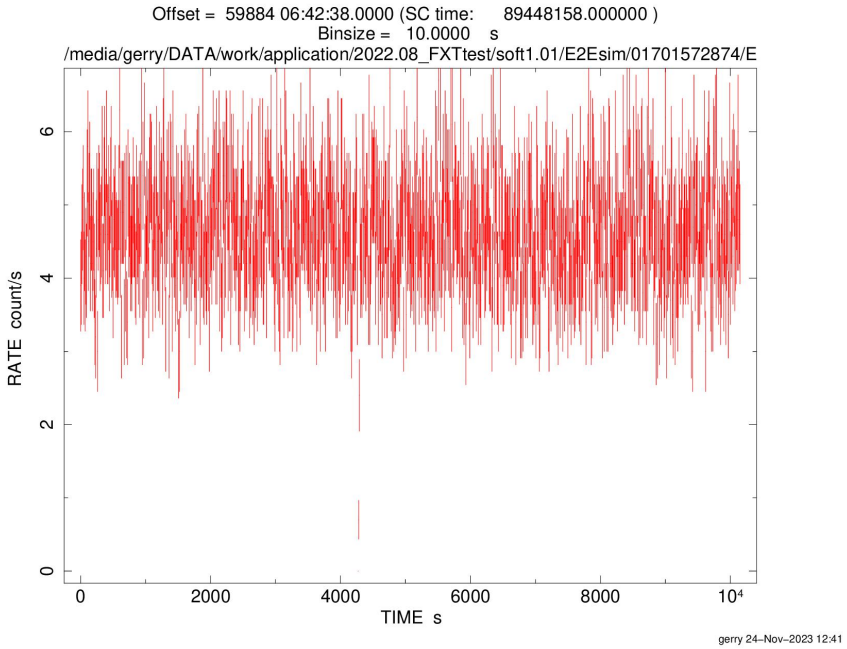

Figure 5.1: Light curve of the field.

Xselect Guide"[1]. Exclude intervals affected by background flares before further analysis.

[1]https://heasarc.gsfc.nasa.gov/docs/rosat/ros_xselect_guide_v1.1/node5.html

5.2 Point source

Use XSELECT to extract a point-source image, spectrum, and light curve from the cleaned event file (`fxta-clean.fits`). The resulting products can be analyzed with XRONOS, XIMAGE, and XSPEC. The examples below describe region selection, pile-up checks, response generation, and timing corrections.

There are 5 main steps to create high-level products:

1. Extract images, spectra, and light-curves.
2. Pileup estimation.
3. Generate the exposure map.
4. Create RMF and ARF files for this source.
5. Barycentric Correction.

5.2.1 FF & PW modes

FF and PW are imaging modes with similar initial processing. Photons can arrive during CCD readout as well as during integration. These Out-of-Time (OoT) events are assigned incorrect RAWY positions and therefore incorrect position-dependent CTI corrections. The OoT fraction depends on the observing mode; see OoT events for the image-correction procedure and its limitations.

When a cleaned event file such as `fxta-clean.fits` is available, use the explicit XSELECT route below. It remains the reference route for custom regions and specialized screening.

- `Extracting light curve & spectrum`

  Use XSELECT's `filter` command to select energy, region, or time intervals, and `clear` to remove a filter. Create products with `extract` and write them with `save`.

  Event grades. Use grades 0–12 for the standard selection, or apply a grade selection appropriate to the analysis.

  Region filtering. An image is extracted from event file read in XSELECT using the `'extract image'` command. Plot the image with the `'plot image'` command which invokes DS9. Both the source and the background region files are created with DS9. The source photons are extracted from a circular shape whose size depends on the brightness of the source. A radius of about 1 arcmin is an illustrative starting point, not a fixed enclosed-energy guarantee. Choose the aperture using the PSF/EEF for the module, energy, and off-axis angle, and account for aperture losses in the ARF. The region filter in XSELECT is processed with the command `'filter region source.reg'`. Alternatively,

the background photons should be extracted from a source-free region away from the strip associated with the OoT events.

Energy filtering. For a band-limited image or light curve in XSELECT, filter on PI channels. Use `fxte2pi` to convert energy limits to PI; it accepts space-separated energies or a comma-separated `energy=` list. The calibrated `ENG` column is also available for event-based imaging, but spectra are extracted in PI. For example, the channels for 2 and 10 keV are:

```
fxte2pi 2.0 10.0
$-PI value for fxt at 2.0 keV: 274
$-PI value for fxt at 10.0 keV: 925
fxte2pi energy=2.0,10.0
$-PI value for fxt at 2.0 keV: 274
$-PI value for fxt at 10.0 keV: 925
```

Therefore, the energy selection can be achieved by the command, e.g. `'filter pha_cutoff 274 925'`, which filters all data within the channel range 274–925, corresponding to 2–10 keV. **Caution: do not apply the PHA filter in the spectrum extraction.**

Time filtering. There are three ways to set the time filter in XSELECT: 1. Specifying the time interval with the mouse cursor, using the command `'filter time cursor'`; 2. Entering the start and the stop times at the keyboards, using command `'filter time UT'` or `'filter time SCC'` or `'filter time MJD'`; 3. Selecting the time intervals with a GTI file containing a series of desired start and stop times, using command `'filter time file filename'`.

The default time resolution of the light curve is 1 second.

method1: FXTSoft light-curve background correction (recommended). For source and background light curves extracted with identical time filters and bin sizes, the official `fxtlccorr` task creates the background-corrected net light curve:

```
fxtlccorr srcfile=source.lc bkgfile=back.lc outfile=net.lc alpha=AUTO
```

With `alpha=AUTO`, a unique matching record in `scaling_manifest` is used when that file is supplied; otherwise the input `BACKSCAL` and `EXPOSURE` values determine the scale. A numeric `alpha` can also be supplied. The input bins must align exactly: `fxtlccorr` does not interpolate or rebin and does not apply exposure or vignetting correction.

method2: HEASoft light-curve arithmetic. The external `lcmath` task provides the corresponding generic arithmetic route when its scaling parameters are set explicitly.

The image, spectrum and light curve of the field can be obtained by the following operation:

```
xselect
> Enter session name >[xsl]
xsl:SUZAKU > read event fxta-clean.fits
> Enter the Event file dir >[./]
> Reset the mission ? >[yes]
xsl:EP-FXTA-FF > extract all
xsl:EP-FXTA-FF > plot image
xsl:EP-FXTA-FF > filter region source.reg
xsl:EP-FXTA-FF > extract spectrum
xsl:EP-FXTA-FF > save spectrum source.pha
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region back.reg
xsl:EP-FXTA-FF > extract spectrum
xsl:EP-FXTA-FF > save spectrum back.pha
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region source.reg
xsl:EP-FXTA-FF > filter pha_cutoff 6 551
xsl:EP-FXTA-FF > extract curve
xsl:EP-FXTA-FF > save curve source.lc
xsl:EP-FXTA-FF > clear region
xsl:EP-FXTA-FF > filter region back.reg
xsl:EP-FXTA-FF > extract curve
xsl:EP-FXTA-FF > save curve back.lc
xsl:EP-FXTA-FF > exit
> Save this session? >[no]
```

Earth contamination can cause sharp count-rate increases near GTI boundaries. Inspect these intervals and exclude affected times from the analysis.

- `Pile-up estimation`

  Point sources with a count rate above the critical value produce severe pile-up in the central region of the Point Spread Function (PSF). In this case, there is a significant probability that two or more soft X-ray photons are registered as a single higher-energy photon (i.e. energy pile-up) of higher pattern type (i.e. pattern pile-up). A quick estimate can be obtained with the online simulation tool (`Filter & Window Mode Evaluation`[2]). Two complementary imaging-mode checks are documented below: pattern migration, similar to the method

[2]https://epfxt.ihep.ac.cn/simulation

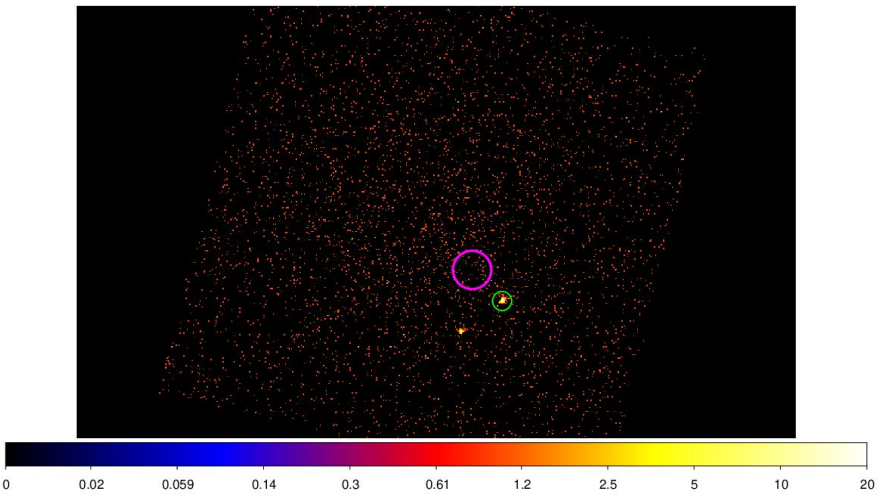

Figure 5.2: FF image with source (green circle) and background (cyan circle) regions.

used for *XMM-Newton* EPIC data[3], and photon loss inferred from the observed PSF shape, similar to the *Swift*/XRT procedure[4].

1. **Pattern-migration method.** Apply the task `fxtplotgrade` on the filtered source event file, source.evt, to inspect the observed versus expected pattern distribution. The default display-only form does not save a PNG:

   ```
   fxtplotgrade evtfile=source.evt show=yes
   ```

   Omitting `outfile` (or using `outfile=NONE`) suppresses PNG output for that invocation, even if PFILES contains a learned output name. To save a PNG, specify the output file explicitly:

   ```
   fxtplotgrade evtfile=source.evt outfile=grade.png show=no
   ```

   The second positional form `fxtplotgrade evtfile.fits grade` also saves the plot as `grade.png`. The figure plots grade fractions as a function of event energy. Reference curves follow the last recognizable processing version in the `EVENTS HISTORY` cards, in append order: T18V1 for legacy processing and T18V2 for current processing. If `HISTORY` has no recognizable version, T18V2 is used. The selected reference must be available in CALDB or its matching `.dat` file. Figure 5.4 shows an example of observation affected by pile-up. In order to reduce pile-up, the filtered event file is extracted from an annular region. As can be seen in Figure 5.5, the pile-up effect becomes negligible after excising the core of the PSF.

[3]https://www.cosmos.esa.int/web/xmm-newton/sas-thread-epatplot
[4]https://www.swift.ac.uk/analysis/xrt/pileup.php

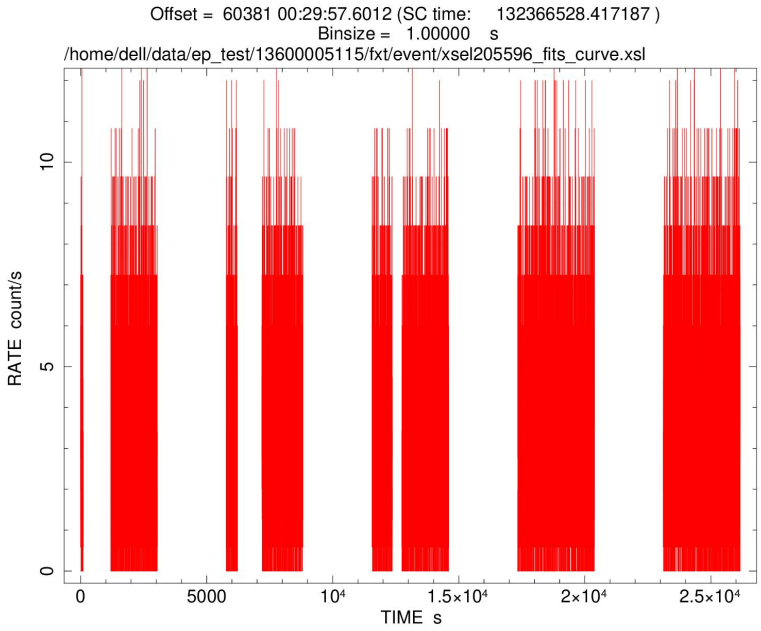

Figure 5.3: Light curve extracted from the FF source region.

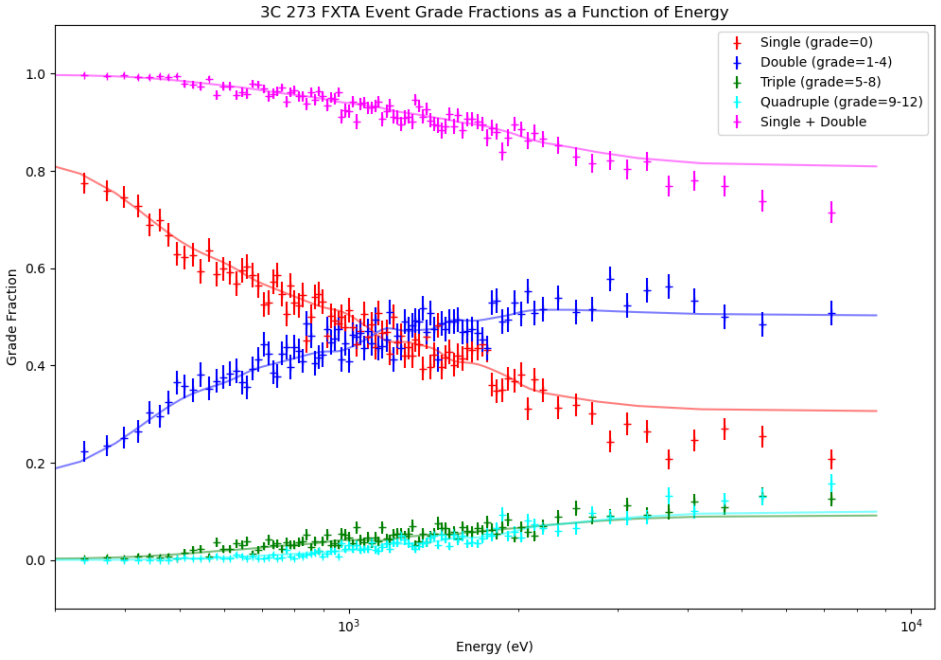

Figure 5.4: Pattern distribution from `fxtplotgrade`, showing pile-up.

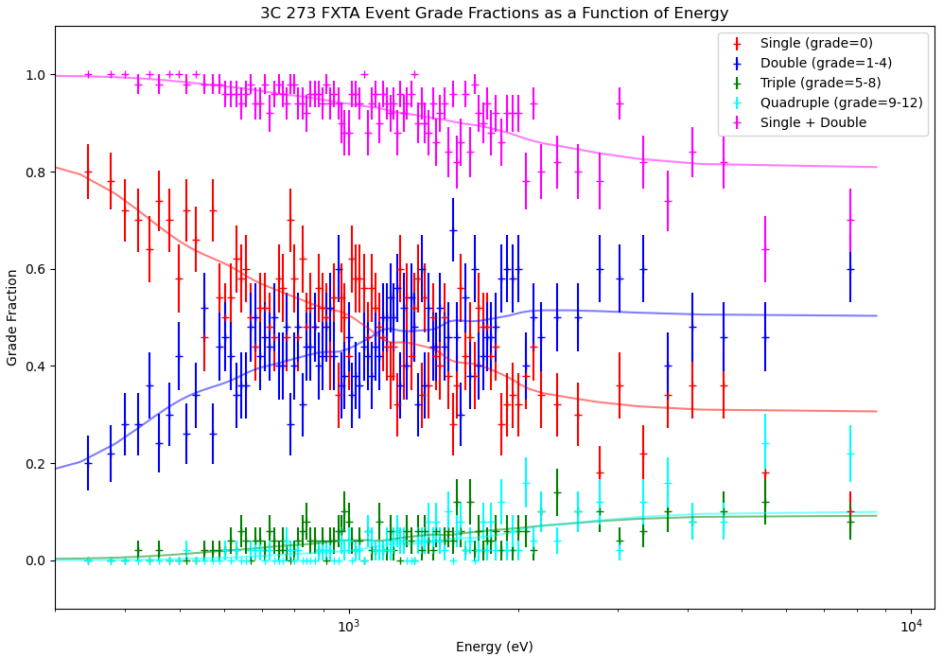

Figure 5.5: Pattern distribution after excluding the bright core; the pile-up signature is negligible.

2. **PSF-shape method.** This method compares the observed radial profile with a model fitted to the unaffected outer wings. For the FXTA example below, copy the detector-specific XIMAGE model pair, `fxta_psf_2beta.dat` and `fxta_psf_2beta.cod`, from CALDB with `fxtpsfmdcopy model=2beta det=fxta`. The 2beta model is the default; this example requires the matching detector model pair in the installed CALDB. The model choice is written explicitly so that it can be checked against the observed profile. For FXTB, use `det=fxtb`, `DETNAM=FXTB`, and the corresponding `fxtb_psf_2beta.dat`/`fxtb_psf_2beta.cod` files. The installed `fxtpsf` command is a deprecated compatibility alias that uses the same implementation and parameter file as `fxtpsfmdcopy`. It emits a deprecation warning on every invocation and will be removed in a future

release; use `fxtpsfmdcopy` in all new procedures.

```
fxtpsfmdcopy model=2beta det=fxta outdir=. clobber=yes
ximage
> CHMDB/ADD/TEL=EP/INST=FXT/DETNAM=FXTA/KEY=psffile/VALUE=fxta_psf_2beta.dat
> read fxta-clean.fits
> cpd /xtk
> disp
> back
> psf/cursor
> col off 1 2 3 4 6
> rescale x 30
> model fxta_psf_2beta.cod
> fit
> rescale
> p
```

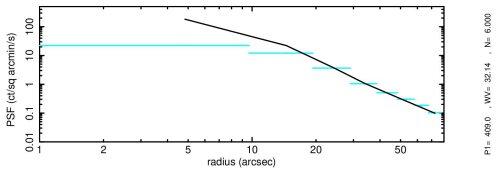

Figure 5.6: The PSF plot. The model and data diverge below about 20 arcsec.

The software XIMAGE is called to read and plot the filtered event or image file generated by XSELECT. The mean background level (counts per pixel) over the entire image is estimated with the task `back`. The user should zoom into the source of interest, run the command `psf/cursor`, click in the centre of the source and where the PSF fades into the background. This opens an interactive plot window with panels showing both the Encircled Energy Function and the PSF. Run `col off 1 2 3 4 6` to turn off all datasets except the cyan points, which are then fitted by the PSF model. Because pile-up mostly affects the PSF centre, fit the model to the outer wings and extrapolate it inward to determine where the data and model diverge (Figure 5.6). This estimates the size of the bright core that should be excluded.

Both methods are applicable to imaging-mode (FF and PW) data. The loss of counts caused by using an annulus can be estimated with the PSF correction by setting `psfcor=1` in `fxtarfgen` when generating the Ancillary Response File.

- `fxtexpogen`

  Generates the exposure map, accounting for detector pixels that did not collect usable data. It requires the event file and the MKF file from the `fxt/hk` directory.

  This command generates two exposure maps: one without vignetting correction and one with vignetting correction applied. The exposure map without vignetting correction can be identified by the addition of the field `-without_vign` in the output file name. In v2.0, `fxtexpogen` also documents optional effective-area scaling (`area_scale=yes`) and clustering parameters (`use_clustering`, `ra_threshold`, and `pa_threshold`) that can accelerate long calculations while preserving the exposure-map geometry.

  Example:

  ```
  fxtexpogen mkffile=../hk/mkf.fits evtfile=fxta-clean.fits \
      outfile=fxta-expo.fits
  ```

- `fxtpsfmap`

  The v2.0 `fxtpsfmap` task generates a PSF-size map directly from an exposure map for a given energy and enclosed energy fraction (EEF). The output image has the same geometry as the exposure map and records the PSF radius in arcseconds at each position. If the filter is not supplied explicitly, the task can read it from the FITS header.

  Example:

  ```
  fxtpsfmap expmap=fxta-expo.fits outfile=fxta-psfmap.fits energy=2.0 eef=0.5
  ```

- `fxtarfgen`

  This command creates the Ancillary Response File (ARF) for the extracted spectrum, taking into account the exposure map generated by the task `fxtexpogen` and, when requested, the Point Spread Function (PSF) correction. For a point-like source (`extend=0`), set `psfcor=1` when the selected region does not contain the full PSF, for example for a small or annular region; this correction is not limited to piled-up sources. With `extend=0 psfcor=1`, the point-source PSF is integrated against the live source-region mask for each exposure or attitude

sample, so region and detector losses are PSF- and exposure-weighted. With `psfmodel=legacy`, the current release uses the CALDB beta model inside about 3 arcmin of the optical axis and tabulated CALDB EEF curves at larger off-axis angles. With `psfmodel=dual-beta`, it uses the indexed CALDB beta model near the axis and the indexed off-axis two-beta extensions at larger angles. Supplying either model enables PSF correction unless an explicit `psfcor` value overrides it. If neither `psfmodel` nor `psfcor` is supplied, the default model is `dual-beta` while `psfcor=0`, so no PSF aperture correction is applied. For extended sources (`extend=1`), these point-source PSF settings are ignored; see the extended-source guide below. The input exposure map for static ARFs should normally be the file with suffix `-without_vign`.

Example:

```
fxtarfgen specfile=source.pha expfile=fxta-expo-without_vign.fits \
      psfcor=0 outfile=source.arf
```

- `fxtrmfgen`

  creates the Redistribution Matrix Files (RMF) for a given spectrum.

  Example:

  ```
  fxtrmfgen specfile=source.pha outfile=source.rmf
  ```

  method1: HEASoft spectrum grouping. First run `fxtarfgen` and `fxtrmfgen` so that ANCRFILE and RESPFILE are present, and ensure that BACKFILE is set when a background spectrum is used. Group the spectrum with the standard HEASoft `grppha` command:

  ```
  grppha infile=source.pha outfile=source_grouped.pha \
  comm='group min 10 & exit' clobber=yes
  ```

  method2: HEASoft response metadata. The same `grppha` command can group the spectrum and attach response and background metadata in one step.

  Associate the source spectrum with its background, RMF, and ARF before fitting in XSPEC. The example groups to at least 10 counts per bin; this is an illustrative grouping choice, not a guarantee that χ^2 statistics are appropriate. Choose the grouping and fit statistic for the count level and background treatment (see the XSPEC guidance on low-count spectra).

```
grppha infile=source.pha outfile=source.pi \
comm=' group min 10 & chkey RESPFILE source.rmf & chkey ANCRFILE source.arf
& chkey BACKFILE back.pha & exit' clobber=yes

xspec
>cpd /xw
>data source.pi
>setplot energy
>ignore bad
>ignore 0.0-0.5
>ignore 10.0-**
>show rate
>plot ldata
>model tbabs*powerlaw
>fit 100
>plot ldata del
>flux 0.5 10.0
>quit
```

5.2.2 TM mode

TM mode provides high time resolution for bright sources. The steps below start from the cleaned event file (`fxta-clean.fits`):

- `Extracting light curve & spectrum`

  Use XSELECT's `filter` command to select energy, region, or time intervals, and `clear` to remove a filter. Create products with `extract` and write them with `save`.

  Event grades. Use grades 0–12 for the standard selection, or apply a grade selection appropriate to the analysis.

  Region filtering. An image is extracted from event file read in XSELECT using the `'extract image'` command. Plot the image with the `'plot image'` command which invokes DS9. For observations made in the TM mode, only one dimensional imaging is preserved. Therefore, a rectangular box of ~ 3 arcmin long and ~ 1 arcmin wide can be adopted to extract the source photons, and a source-free rectangular region of the same width is adopted for the background. The roll angle of observation can be found in the header of the event file ('PA_PNT', or 270°-'PA_PNT'). Caution: for the TM mode observations, do not use a circle/annulus region to extract photons. Both the source and the background region files are created with DS9. The region filter in XSELECT is processed with the command `'filter region source.reg'`. The default time resolution of the light curve is 1 second, while higher time resolution is also available.

method1: FXTSoft light-curve background correction (recommended). For source and background light curves extracted with identical time filters and bin sizes, the official `fxtlccorr` task creates the background-corrected net light curve:

```
fxtlccorr srcfile=source.lc bkgfile=back.lc outfile=net.lc alpha=AUTO
```

With `alpha=AUTO`, a unique matching record in `scaling_manifest` is used when that file is supplied; otherwise the input `BACKSCAL` and `EXPOSURE` values determine the scale. A numeric `alpha` can also be supplied. The input bins must align exactly: `fxtlccorr` does not interpolate or rebin and does not apply exposure or vignetting correction.

method2: HEASoft light-curve arithmetic. The external `lcmath` task provides the corresponding generic arithmetic route when its scaling parameters are set explicitly.

Energy filtering. For a band-limited image or light curve in XSELECT, filter on PI channels. Use `fxte2pi` to convert energy limits to PI; it accepts space-separated energies or a comma-separated `energy=` list. The calibrated `ENG` column is also available for event-based imaging, but spectra are extracted in PI. For example, the channels for 2 and 10 keV are:

```
fxte2pi 2.0 10.0
$-PI value for fxt at 2.0 keV: 274
$-PI value for fxt at 10.0 keV: 925
fxte2pi energy=2.0,10.0
$-PI value for fxt at 2.0 keV: 274
$-PI value for fxt at 10.0 keV: 925
```

Therefore, the energy selection can be achieved by the command, e.g. `'filter pha_cutoff 274 925'`, which filters all data within the channel range 274–925, corresponding to 2–10 keV. Caution: do not apply the PHA filter in the spectrum extraction.

Time filtering. There are three ways to set the time filter in XSELECT : 1. Specifying the time interval with the mouse cursor, using the command `'filter time cursor'`; 2. Entering the start and the stop times at the keyboards, using command `'filter time UT'` or `'filter time SCC'` or `'filter time MJD'`; 3. Selecting the time intervals with a GTI file containing a series of desired start and stop times, using command `'filter time file filename'`.

The following session saves the source-region event file for barycentric correction, spectra without a PI-band cut, and 0.5–2.5 keV light curves. For this band, `fxte2pi 0.5 2.5` gives PI limits 73 and 328:

```
xselect
> Enter session name >[xsl]
xsl:SUZAKU > read event fxta-clean.fits
> Enter the Event file dir >[./]
> Reset the mission ? >[yes]
xsl:EP-FXTA-TM > set datamode TM
xsl:EP-FXTA-TM > extract all
xsl:EP-FXTA-TM > plot image
xsl:EP-FXTA-TM > filter region source.reg
xsl:EP-FXTA-TM > extract events copyall=yes
xsl:EP-FXTA-TM > save events fxta-source.evt
xsl:EP-FXTA-TM > extract spectrum
xsl:EP-FXTA-TM > save spectrum source.pha
xsl:EP-FXTA-TM > clear region
xsl:EP-FXTA-TM > filter region back.reg
xsl:EP-FXTA-TM > extract spectrum
xsl:EP-FXTA-TM > save spectrum back.pha
xsl:EP-FXTA-TM > clear region
xsl:EP-FXTA-TM > filter region source.reg
xsl:EP-FXTA-TM > filter pha_cutoff 73 328
xsl:EP-FXTA-TM > extract curve
xsl:EP-FXTA-TM > save curve source.lc
xsl:EP-FXTA-TM > clear region
xsl:EP-FXTA-TM > filter region back.reg
xsl:EP-FXTA-TM > extract curve
xsl:EP-FXTA-TM > save curve back.lc
xsl:EP-FXTA-TM > exit
> Save this session? >[no]
```

- `fxtexpogen`

  Generates the exposure map from the event file and the MKF file in `fxt/hk`, accounting for detector pixels that did not collect usable data.
  Example:

  ```
  fxtexpogen mkffile=../hk/mkf.fits evtfile=fxta-clean.fits \
      outfile=fxta-expo.fits
  ```

- `fxtarfgen`

  creates the Ancillary Response File (ARF) for the extracted spectrum, taking into account the exposure map generated by the task `fxtexpogen`.
  Example:

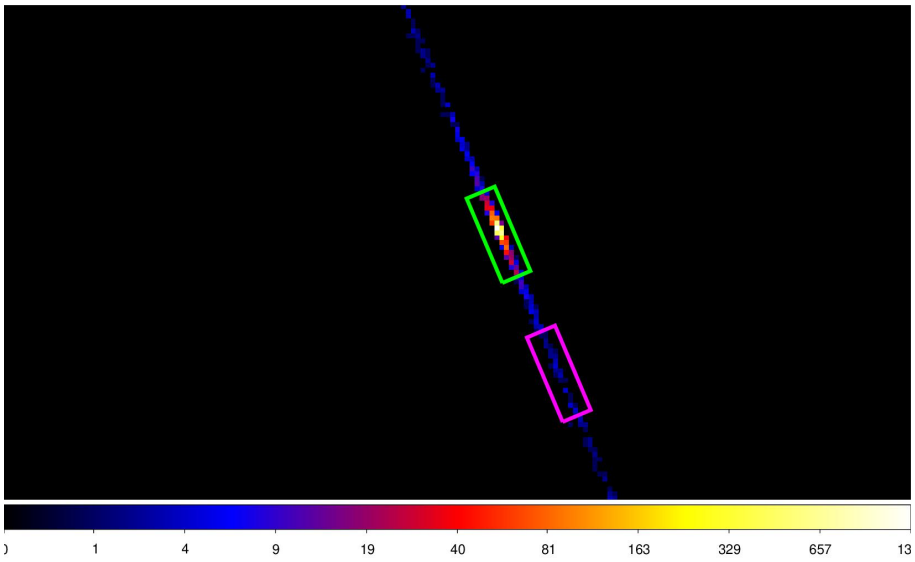

Figure 5.7: TM image with source (green rectangle) and background (cyan rectangle) regions.

```
fxtarfgen specfile=source.pha expfile=fxta-expo-without_vign.fits \
    outfile=source.arf psfcor=0
```

PSF correction is not available for TM data. Keep `psfcor=0` in this mode.

- `fxtrmfgen`

  creates the Redistribution Matrix Files (RMF) for a given spectrum.
  Example:

  ```
  fxtrmfgen specfile=source.pha outfile=source.rmf
  ```

  Associate the source spectrum with its background, RMF, and ARF before fitting in XSPEC. The example groups to at least 10 counts per bin; this is an illustrative grouping choice, not a guarantee that χ^2 statistics are appropriate. Choose the grouping and fit statistic for the count level and background treatment (see the XSPEC guidance on low-count spectra).

- `fxtbary`

  applies barycenter correction to FXT event data files before a precise timing analysis.
  Example:

  ```
  fxtbary evtfile=fxta-source.evt outfile=fxta-source-bary.evt \
      orbitfile=../../auxil/orb.fits \
      ephemfile=$HEADAS/refdata/de421.bsp ra=342.1721004 dec=-51.1657098
  ```

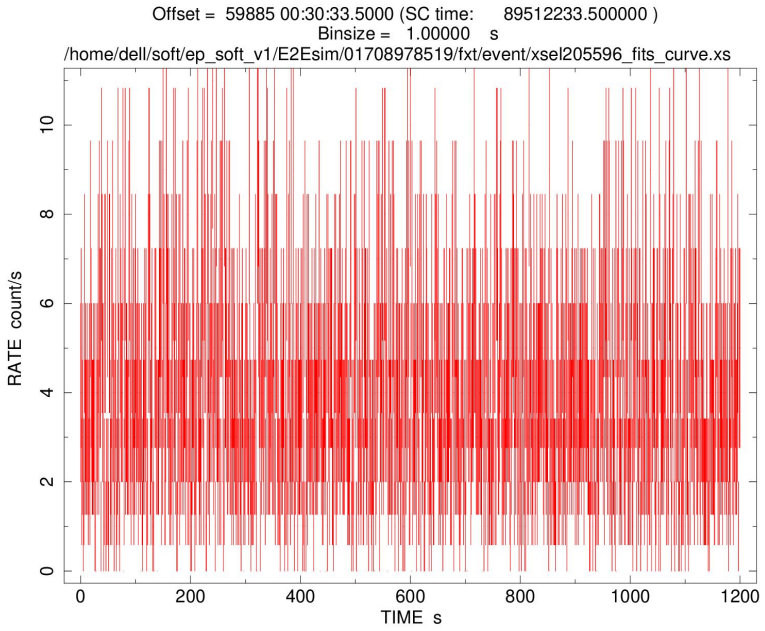

Figure 5.8: Light curve of the TM field.

5.3 Extended sources

5.3.1 Imaging analysis

FXT's 1° field of view and low instrumental background support observations of diffuse objects, including supernova remnants, superbubbles, Galactic plasma, nearby galaxies, and galaxy clusters. Extended sources often have low surface brightness and may span several pointings. Their images require pixel masking and exposure corrections before scientific analysis. This section covers image extraction, masking, vignetting correction, and, when needed, smoothing and mosaics.

FXTSoft provides image extraction and arithmetic through `fxtimage`, exposure maps through `fxtexpogen`, and reprojection and mosaics through `fxtimgmerge`. The examples below pair these tools with HEASoft/FTOOLS alternatives. CIAO[5] and Astropy[6] provide additional image-analysis tools.

Analysis methods. In this section, **method1** uses FXTSoft 2.0 tools and **method2** uses HEASoft/FTOOLS. Both use the same screened event and exposure products. Starting from a clean event file, method1 proceeds as follows:

1. Extract the count image with `fxtimage mode=bin`.

2. Create the exposure maps with `fxtexpogen`.

[5]https://cxc.cfa.harvard.edu/ciao/
[6]https://www.astropy.org

3. Create the OoT event list with `fxtootest` and bin it with `fxtimage mode=bin`.
4. Subtract the OoT image with `fxtimage mode=oot`.
5. Build the particle spectrum and image with `fxtbkggen` and `fxtbkgmap`.
6. Subtract the particle image with `fxtimage mode=sub`.
7. Run `fxtimage mode=rate` on the net-count image for exposure thresholding, masking, and rate calculation.

The same scientific order applies to method2; only the image arithmetic and threshold commands differ. The dedicated background and mosaic tasks are retained because `fxtimage` does not replace background modeling or cross-observation reprojection.

Image extraction

An X-ray event file lists the arrival time, detector position, energy information, and grade of each recorded event. An image is a two-dimensional histogram of the selected events. FXT provides three coordinate pairs: `RAWX/RAWY` in detector readout coordinates, `DETX/DETY` in detector coordinates, and `X/Y` in sky coordinates. Each module has one CCD, and the RAW and DET coordinates differ by one pixel in their origins:

$$\texttt{DETX} = \texttt{RAWX} + 1; \texttt{DETY} = \texttt{RAWY} + 1. \tag{5.1}$$

The `SKY` plane maps `X` and `Y` to equatorial coordinates and is normally used for science images. The `RAW/DET` plane has 384×384 pixels; the default `SKY` image has 600×600 pixels, each subtending approximately $9.66''$. Figure 5.9 compares the coordinate planes.

Start from the screened event file for each FXT module. Use `ENG`, the event energy in keV, to select an imaging band; use `PI` rather than `ENG` when extracting a spectrum. The examples below extract a 0.4–7.0 keV count image from *fxta-clean.fits*. **method1** (FXTSoft 2.0). The `fxtimage` task bins the filtered events directly:

```
fxtimage mode=bin evtfile=fxta-clean.fits \
    outfile=fxta-0.4-7.0-cts.fits emin=0.4 emax=7.0 clobber=yes
```

method2 (original HEASoft/FTOOLS route). The original `ftcopy` command is retained for users who want to inspect the binning expression explicitly[7]:

```
ftcopy "fxta-clean.fits[eng=0.4:7.0][bin x=0.5:600.5,y=0.5:600.5]" \
    fxta-0.4-7.0-cts.fits clobber=yes
```

Images can also be extracted interactively with XSELECT and displayed in SAOImage DS9.

[7] https://heasarc.gsfc.nasa.gov/lheasoft/ftools/ftools_subpacks.html

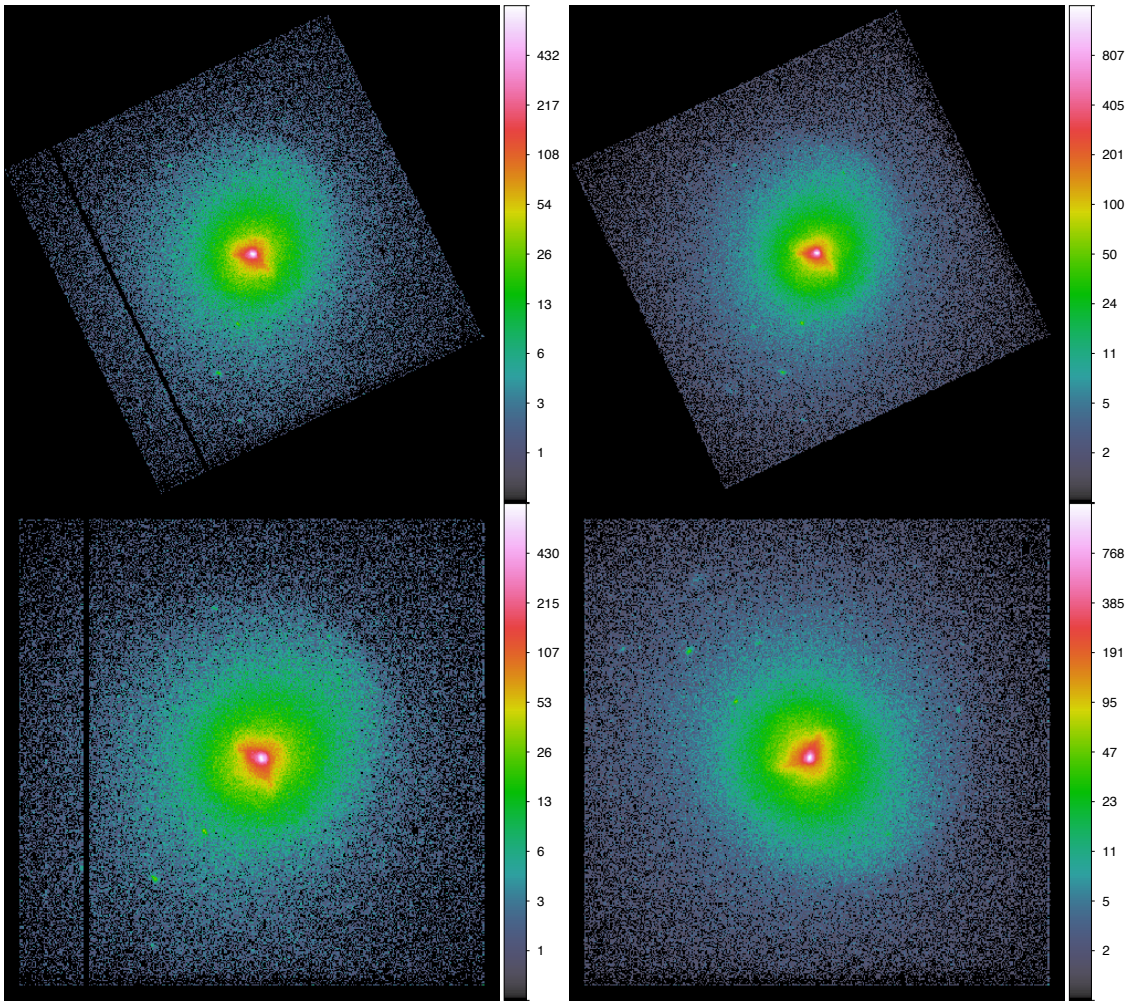

Figure 5.9: M87 first-light images: FXTA (left) and FXTB (right), in SKY (top) and RAW/DET (bottom) coordinates. Bad pixels are masked.

Exposure map: pixel masking and vignetting correction

An exposure map records the usable exposure at each sky pixel, including pointing motion, pixel masks, and optionally vignetting. Dividing a count image by the matching exposure map corrects these spatial variations. With `area_scale=yes`, the map also includes effective area and has units of $\mathrm{cm}^2\,\mathrm{s}$ rather than seconds; the resulting rate units are described below.

Pixel masking

Detector defects and edge effects can distort an image. Mask the affected pixels in both the event selection and exposure calculation. The main cases are:

Bad pixels: The calibration includes one bad column in FXTA and several bad pixels in both modules. `fxtbadpix` flags the affected locations and records them in

a FITS extension. Bad pixels also affect reconstructed 3×3 event islands in their neighborhood, extending the flagged region by two pixels. The screened event file produced by `fxtchain` excludes these events.

Detector edge: The readout electronics do not record events at `RAWX=0` or `RAWX=383`. Event reconstruction can overestimate counts at `RAWX`=1, 382 and `RAWY`=0, 383 (Figure 5.9). Incomplete 3×3 event islands also affect grades in the two adjacent rows or columns; mask these edge regions.

Shading: A light-blocking board protects the frame-store area but also shades nearby detector pixels. In both modules the effective area begins to decrease around `RAWY`$\approx$20 and is almost zero at `RAWY`=1 (bottom panels of Figure 5.9). The workflow below masks this region with `edgecovermask`.

Vignetting correction

The effective area generally decreases away from the optical axis, with an energy-dependent change called vignetting. A matching exposure map accounts for this variation when converting the count image to a rate image.

Common exposure-map step. By default, the CCD edge and bad pixels are automatically masked and exposure maps with and without vignetting correction (filename with a suffix "-without_vign") are created at the same time. Such an example is Figure 5.10. Applying the parameter *edgecovermask*, pixels with `RAWY`<20 affected by the shading, and two rows/columns on the detector edge will be masked in the exposure map as well. The exposure map can be created by the script *fxtexpogen* with a given effective area:

```
fxtexpogen evtfile=fxta-clean.fits mkffile=../hk/mkf.fits \
    outfile=fxta-0.4-7.0-exp.fits edgecovermask=1
```

In v2.0, the parameter *area_scale* can be used to multiply the on-axis effective area into the exposure map if it is set as "yes". This changes the unit of the exposure map from "s" to "cm^2 s", and also takes the difference in the effective area between FXTA and FXTB into consideration. For long observations, `fxtexpogen` can also speed up the calculation by grouping similar attitudes with `use_clustering=yes`; the parameters *ra_threshold* and *pa_threshold* control the clustering tolerance.

method1: FXTSoft workflow (recommended). For routine vignetting correction, the official FXTSoft task `fxtimage` uses the existing count or net-count image and exposure map from the common exposure-map step; it does not replace `fxtexpogen`. Set `mode=rate` to write the rate image together with thresholded count, exposure, and mask products:

```
fxtimage mode=rate imgfile=fxta-0.4-7.0-net.fits \
    expfile=fxta-0.4-7.0-exp.fits \
    outfile=fxta-0.4-7.0-rate.fits expfrac=0.15 \
```

```
    outcounts=fxta-0.4-7.0-thresh.fits \
    outexp=fxta-0.4-7.0-expthresh.fits \
    maskfile=fxta-0.4-7.0-mask.fits clobber=yes
```

If OoT or particle-background subtraction was skipped, the raw count image `fxta-0.4-7.0-cts.fits` can be supplied as `imgfile` instead. Run this rate step only after all required count-space subtraction.

method2: explicit FTOOLS workflow. The commands below apply the same 15% exposure threshold and show each intermediate product. Use the net-count image after the required background subtractions, or substitute the raw count image if those subtractions were skipped.

```
exp=‘ftstat fxta-0.4-7.0-exp.fits | grep "maximum" | awk ’{print $3}’‘
locut=‘echo "0.15*$exp" | bc -l‘
fimgtrim fxta-0.4-7.0-exp.fits $locut INDEF 0 INDEF \
    fxta-0.4-7.0-expthresh.fits clobber=yes
farith fxta-0.4-7.0-expthresh.fits fxta-0.4-7.0-expthresh.fits \
    fxta-0.4-7.0-mask.fits DIV clobber=yes
fimgtrim fxta-0.4-7.0-mask.fits 0 INDEF 0 INDEF fxta-0.4-7.0-mask.fits clobber=yes
farith fxta-0.4-7.0-net.fits fxta-0.4-7.0-mask.fits \
    fxta-0.4-7.0-thresh.fits MUL clobber=yes
```

Dividing the count map by the exposure map produces a vignetting-corrected count rate map. Both methods preserve the same exposure-unit convention: an exposure map in seconds gives `count/s`, while `area_scale=yes` gives an area-time exposure and `count/(cm**2*s)`. If the exposure-map file is present but its BUNIT is missing or blank, the FXTSoft method treats it as seconds, emits a warning, and writes a `HISTORY` record. It rejects scaled or otherwise unsupported BUNIT values and does not process a missing exposure-map file.

```
farith fxta-0.4-7.0-thresh.fits fxta-0.4-7.0-expthresh.fits \
    fxta-0.4-7.0-rate.fits DIV clobber=yes
fimgtrim fxta-0.4-7.0-rate.fits 0 INDEF 0 INDEF fxta-0.4-7.0-rate.fits clobber=yes
```

The edge depressions in the unvignetted exposure maps (Figure 5.10) reflect pointing motion. The modules have different effective areas and vignetting profiles.

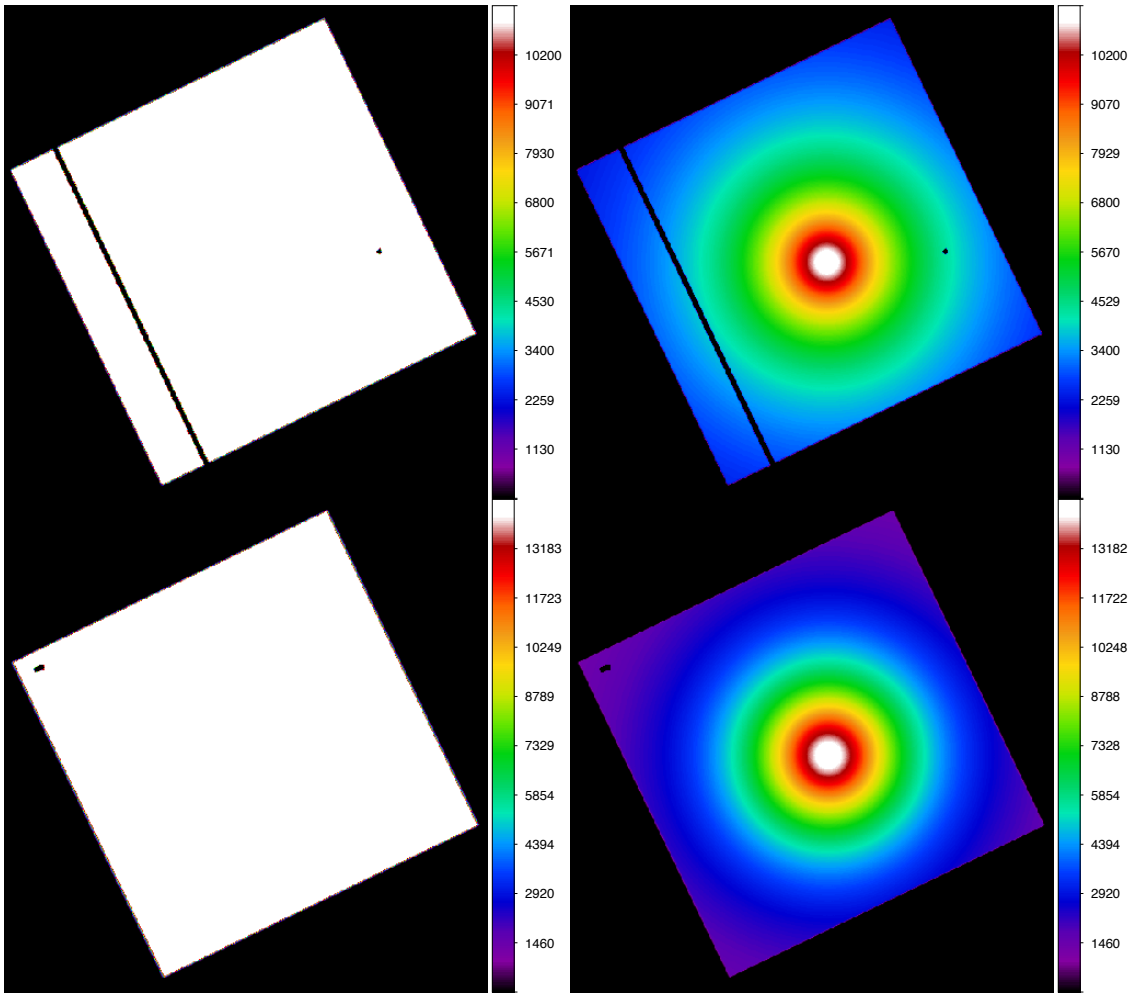

Figure 5.10: Exposure maps at 4 keV: FXTA (top) and FXTB (bottom), without (left) and with (right) vignetting. Bad pixels are masked.

Background Noise

Instrumental background, stray light, and OoT events contaminate the recorded image. They require different treatments: model the components supported by the tools below, and exclude regions where the residual contamination makes the intended analysis unreliable.

OoT events

Photons arriving during CCD readout are assigned incorrect `RAWY` positions. In Full Frame mode, the readout lasts about 0.1152 ms and OoT events account for about 0.23% of the recorded events (see FXT Modes). A bright source can therefore produce a trail along its detector columns. Where observation planning permits, choose a roll angle that keeps the diffuse target away from those columns. Generate

a modeled OoT event list with `fxtootest`; this step is common to both routes:

```
fxtootest evtfile=fxta-clean.fits outfile=fxta-oot.fits \
    attfile=../../auxil/att.fits
```

method1 (FXTSoft 2.0).

```
fxtimage mode=bin evtfile=fxta-oot.fits \
    outfile=fxta-0.4-7.0-oot.fits emin=0.4 emax=7.0 clobber=yes
fxtimage mode=oot imgfile=fxta-0.4-7.0-cts.fits \
    ootfile=fxta-0.4-7.0-oot.fits outfile=fxta-0.4-7.0-suboot.fits \
    scale=0.0023 clobber=yes
```

method2 (original HEASoft/FTOOLS route). Use the same SKY bin edges as the count image so that the two inputs to `ftimgcalc` have identical image geometry.

```
ftcopy "fxta-oot.fits[eng=0.4:7.0][bin x=0.5:600.5,y=0.5:600.5]" \
    fxta-0.4-7.0-oot.fits clobber=yes
ftimgcalc fxta-0.4-7.0-suboot.fits 'A-0.0023*B' a=fxta-0.4-7.0-cts.fits \
    b=fxta-0.4-7.0-oot.fits clobber=yes
```

On a common grid, `fxtimage mode=sub` with `bscale=0.0023` gives the same numerical subtraction. Use `mode=oot` to retain the OoT-specific alignment handling and output history. Subtract OoT counts before applying vignetting correction.

OoT events are modeled, not identified individually. Avoid spectroscopy of an extended source strongly contaminated by a nearby source's OoT trail: subtracting a modeled spectrum does not fully account for the associated uncertainties.

Pile-up introduces another uncertainty. The simulated OoT list is derived from the observed events, which may already be piled up, whereas the short readout interval makes the actual OoT events much less affected. The simulation can then underestimate the OoT count rate. Image subtraction does not remove this limitation for spectroscopy of a contaminated diffuse source.

In the upper panels of Figure 5.11, the arcs may be stray light, and some residuals may be caused by pile-up.

Stray light

FXT normally focuses photons after two mirror reflections, but photons from a bright source outside the field can reach the detector after a single reflection. The resulting stray light can form arcs or knots across the image (Figure 5.12), even when the source lies several degrees away. Exclude affected regions, especially for spectroscopy; the current workflow does not provide a reliable stray-light subtraction model.

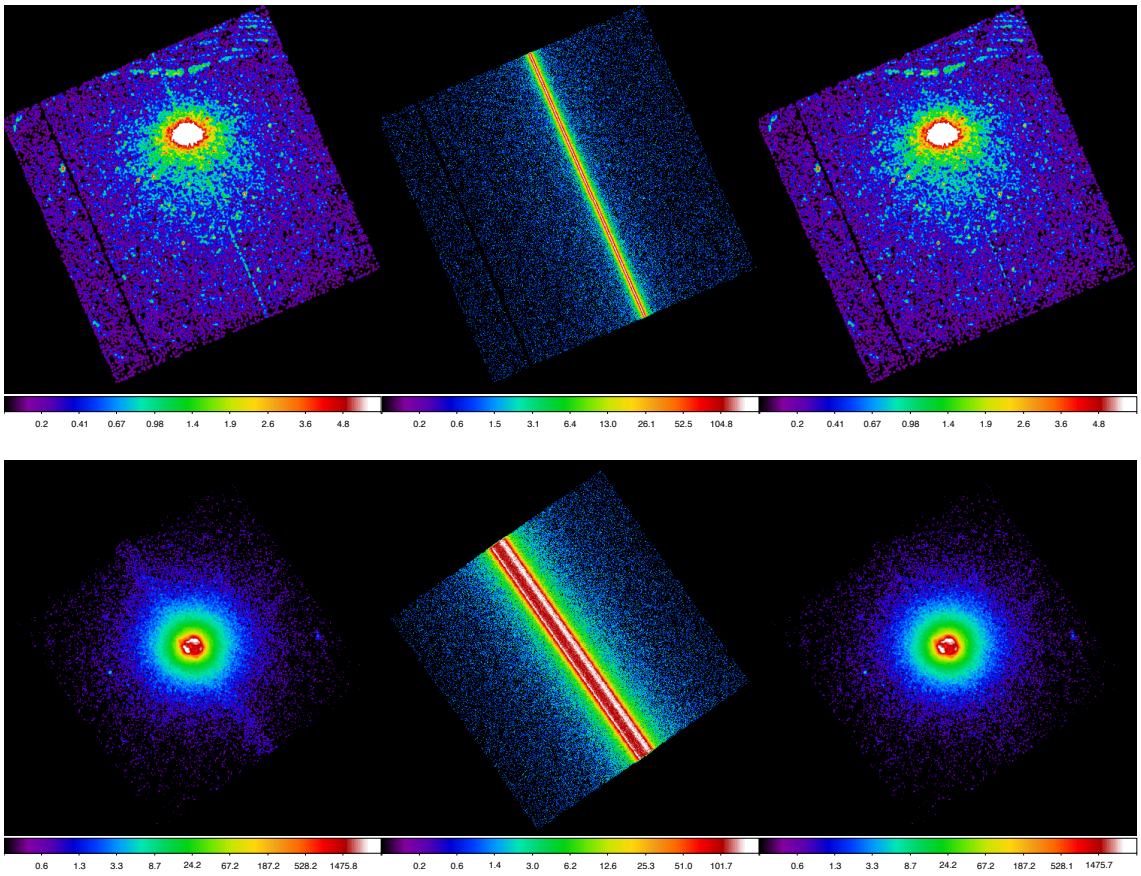

Figure 5.11: Raw counts, simulated OoT counts, and OoT-subtracted images (left to right), for a point source (top) and an extended source (bottom).

Instrument background

The instrumental background has a relatively flat spectrum, particularly above 2 keV, and can dominate faint diffuse emission. Its roughly uniform detector distribution becomes artificially brighter near the edges after vignetting correction if it is not subtracted first (Figure 5.13). The updated CALDB v2.0 package will also provide blank-sky reference observations in `data/ep/fxt/bcf/bkg/blanksky.tar.gz` for background studies. The `fxtbkggen` workflow below estimates the instrumental particle background from FSA spectra using the calibration templates under `bcf/bkgmodel/`. Estimate the background spectrum for each module from the FSA data with *fxtbkggen*:

```
fxtbkggen infile=fxta_fsa.pi outfile=fxta_bkg.pi
```

The spectrum can be converted into a particle-background image with *fxtbkgmap* using the non-vignetted exposure map produced by *fxtexpogen*:

```
fxtbkgmap infile=fxta_bkg.pi \
```

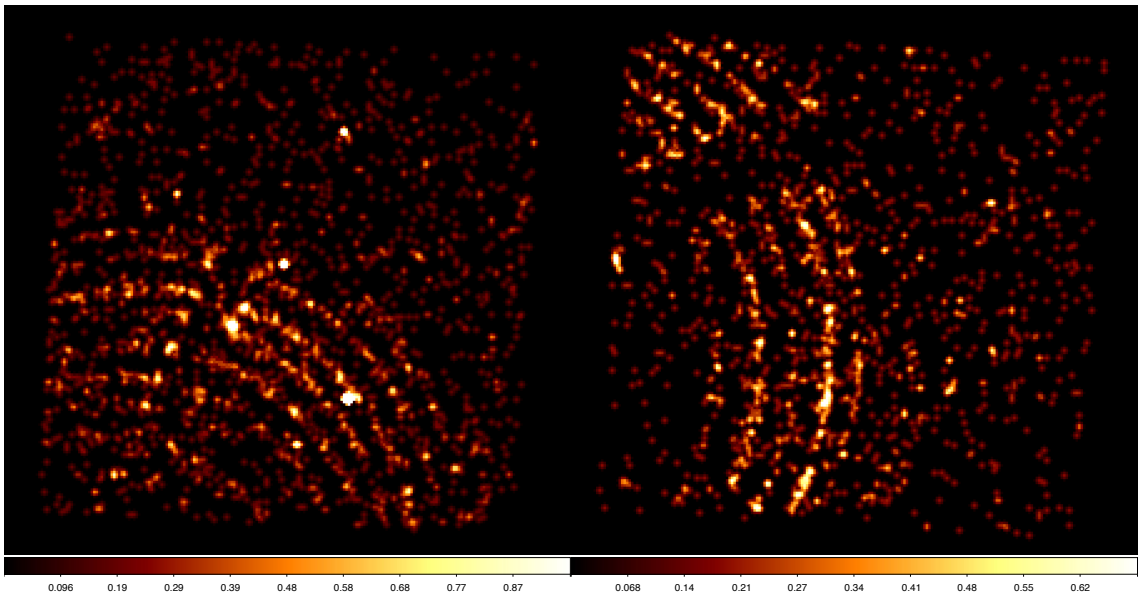

Figure 5.12: Smoothed 0.4–7.0 keV images of stray light from the off-field Crab Nebula. The arcs and knots are not celestial sources.

```
    expfile=fxta-0.4-7.0-exp-without_vign.fits \
    outfile=fxta-0.4-7.0-particle.fits emin=0.4 emax=7.0
```

method1 (FXTSoft 2.0).

```
fxtimage mode=sub imgfile=fxta-0.4-7.0-suboot.fits \
    imgfile2=fxta-0.4-7.0-particle.fits \
    outfile=fxta-0.4-7.0-net.fits \
    ascale=1 bscale=1 outunit=AUTO clobber=yes
```

method2 (original HEASoft/FTOOLS arithmetic).

```
ftimgcalc fxta-0.4-7.0-net.fits 'A-B' \
    a=fxta-0.4-7.0-suboot.fits \
    b=fxta-0.4-7.0-particle.fits clobber=yes
```

`fxtimage` only performs arithmetic subtraction using an existing particle map; it does not replace `fxtbkggen`/`fxtbkgmap` particle-spectrum and particle-map modeling. Particle subtraction precedes vignetting correction. The non-vignetted exposure map is used only to construct the particle map; apply the vignetting-corrected exposure map later to the net counts.

Smoothing

Smoothing can make extended-source structure easier to see. Use smoothed images for display, not for quantitative analysis. **method1** (FXTSoft 2.0).

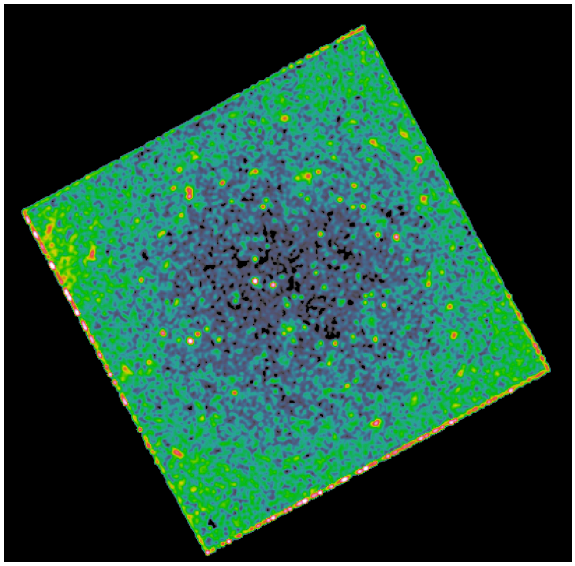

Figure 5.13: Vignetting-corrected blank field before instrumental-background subtraction.

```
fxtimage mode=smooth imgfile=fxta-0.4-7.0-rate.fits \
    outfile=fxta-0.4-7.0-rate-smooth.fits method=gauss sigma=2 \
    clobber=yes
```

method2 (original HEASoft workflow). HEASoft provides the scripts *fgauss*, *fboxcar*, and *fadapt*. The first two smooth the picture on a fixed scale while the latter applies adaptive smoothing. The native `fxtimage` route covers fixed-kernel Gaussian and box smoothing; `fadapt` remains an external adaptive option and is not equivalent to the native fixed-kernel operations.

RGB image

To check the photon distribution in different energy ranges, we can assign different colors to the photons of different energies for easy viewing. Here we withdraw three images from different band ranges: 0.3-1 keV as the soft band, 1-3 keV as the middle band, while 3-5 keV as the hard band.

method1: FXTSoft band extraction. The official `fxtimage` task can create the three band images directly from the cleaned event file:

```
fxtimage mode=bin evtfile=evt-a-cl.fits outfile=evt_soft.im \
    emin=0.3 emax=1.0 clobber=yes
fxtimage mode=bin evtfile=evt-a-cl.fits outfile=evt_mid.im \
    emin=1.0 emax=3.0 clobber=yes
fxtimage mode=bin evtfile=evt-a-cl.fits outfile=evt_hard.im \
    emin=3.0 emax=5.0 clobber=yes
```

method2: XSELECT band extraction. Extract the same soft, middle, and hard bands with XSELECT. The command `fxte2pi 0.3 1.0 3.0 5.0` gives PI limits 38, 150, 378, and 560 for the following example; do not assume a fixed 10 eV channel width.

```
xselect
> Enter session name >[EP] EP  # session name, will be the prefix of later output
> set datadir .
> read events evt-a-cl.fits
> set datamode  FF/PW/TM   # when the mode are not recognized correctly.
> filter time file gti.fits
> set binsize 1.
> filter pha_cutoff 38 150
> extract image xysize=601 xybinsize=1 xcenter=300.5 ycenter=300.5 copyall=yes
> save image evt_soft.im
> clear pha_cutoff
> filter pha_cutoff 150 378
> extract image xysize=601 xybinsize=1 xcenter=300.5 ycenter=300.5 copyall=yes
> save image evt_mid.im
> clear pha_cutoff
> filter pha_cutoff 378 560
> extract image xysize=601 xybinsize=1 xcenter=300.5 ycenter=300.5 copyall=yes
> save image evt_hard.im
> quit
```

With the images of different bands, a false-color image can be created by either route.
method1 (FXTSoft 2.0).

```
fxtimage mode=rgb redfile=evt_soft.im greenfile=evt_mid.im \
    bluefile=evt_hard.im outfile=rgb.png clobber=yes
```

The `fxtimage mode=rgb` operation only composes the already extracted images; it does not perform the band filtering or exposure correction.
method2 (original DS9 display workflow). The original interactive display command is retained:

```
ds9 -rgb -red evt_soft.im -green evt_mid.im -blue evt_hard.im
```

Figure 5.14 shows a three-color, vignetting-corrected, adaptively smoothed M87 image from four observations with different filters. Count rates depend on effective area, so combining modules or filters without matching exposure corrections can leave artifacts near the FXTA bad column or the detector edges. Stacking

can improve the signal-to-noise ratio when the images and exposure maps are combined consistently. The faint patterns at the lower left and lower right may reflect differences between the modules' effective areas.



Figure 5.14: M87 first-light image, vignetting-corrected, stacked, and adaptively smoothed. Red: 0.4–0.7 keV; green: 0.7–1.1 keV; blue: 1.1–2.3 keV. The color scale is logarithmic.

Mosaic

Usually, more than one observation would be mosaicked or stacked to extend the imaging area or raise the signal-to-noise ratio. **method1** (FXTSoft 2.0). The `fxtimgmerge` task places multiple count images and their matching exposure maps on a common sky WCS, sums each product, and can optionally write the merged count-rate image. The image and exposure-map lists must use the same order and energy band. For example, FXT-A and FXT-B can be combined with:

```
fxtimgmerge imgfiles=fxta-0.4-7.0-cts.fits,fxtb-0.4-7.0-cts.fits \
    expfiles=fxta-0.4-7.0-exp.fits,fxtb-0.4-7.0-exp.fits \
    outimg=fxtab-0.4-7.0-cts.fits \
    outexp=fxtab-0.4-7.0-exp.fits \
    outrateimg=fxtab-0.4-7.0-rate.fits expmin=1.0 clobber=yes
```

Effective area differs between filters and modules, so the same source can have different count rates. Generate every input exposure map with `fxtexpogen` using the same `area_scale` setting and the same exposure-unit convention; `fxtimgmerge` rejects mixed conventions. Use `area_scale=yes` to include the on-axis effective area when combining modules or filters. The `outrateimg` BUNIT is `count/s` for

exposure maps in seconds and `count/(cm**2*s)` for maps in $\mathrm{cm}^2\,\mathrm{s}$. Only the latter is an effective-area-normalized rate.
method2 (original CIAO workflow). If CIAO is preferred, its `reproject_image_grid` task can reproject and combine images on a common sky grid. Use `method=sum` for count images (and count-based background maps, if needed) and `method=average` for exposure maps when regridding. Reproject and combine the count images separately from the exposure maps before calculating the merged count-rate image:

$$rate = \frac{(\Sigma cts - \Sigma bkg)}{(\Sigma exp)} \tag{5.2}$$

The count or net-count images and the exposure maps are merged separately before division. The `fxtimage` task does not replace `fxtimgmerge` for construction of a cross-observation common WCS, reprojection, and merging.

Point-source detection and masking

Point sources should be identified before quantitative analysis of diffuse emission.
method1: FXTSoft point-source detection. The official FXTSoft task `fxtsrcdet` accepts a counts image and its matching exposure map, and writes a FITS source table, a DS9 region file, and a significance map. For the 0.4–7.0 keV images created above, a basic point-source run is:

```
fxtsrcdet imgfile=fxta-0.4-7.0-cts.fits \
    expofile=fxta-0.4-7.0-exp.fits outfile=fxta-sources.fits \
    regfile=fxta-sources.reg sigfile=fxta-significance.fits \
    srcclass=point sigcalfile=AUTO clobber=yes
```

method2: CIAO point-source detection. An external alternative is the official CIAO `wavdetect` task, which can be used when a CIAO-compatible wavelet workflow is required:

```
wavdetect infile=fxta-0.4-7.0-cts.fits \
    outfile=fxta-wavdetect.fits scellfile=fxta-scell.fits \
    imagefile=fxta-wavdetect-image.fits defnbkgfile=fxta-wavdetect-bkg.fits \
    regfile=fxta-wavdetect.reg expfile=fxta-0.4-7.0-exp.fits \
    scales="1 2 4 8" sigthresh=1e-6 clobber=yes
```

Figure 5.15 uses 3C 273 observation 13600005907: FXTB in Full Frame (FF) mode, filter 02, 0.4–7.0 keV, and a nominal exposure of 2205.2 s. With `srcclass=point` and `sigcalfile=AUTO`, the catalogue contains 13 candidate detections.

The ellipses use the source-table fields `X`, `Y`, `R`, and `ROTANG`; the corresponding DS9 region file is a public output. The nearest detection to the target is `SRC_ID 7`, with an offset of about 3.2″ and significance 128.4944. These are detection diagnostics, not an independent astrometric calibration.

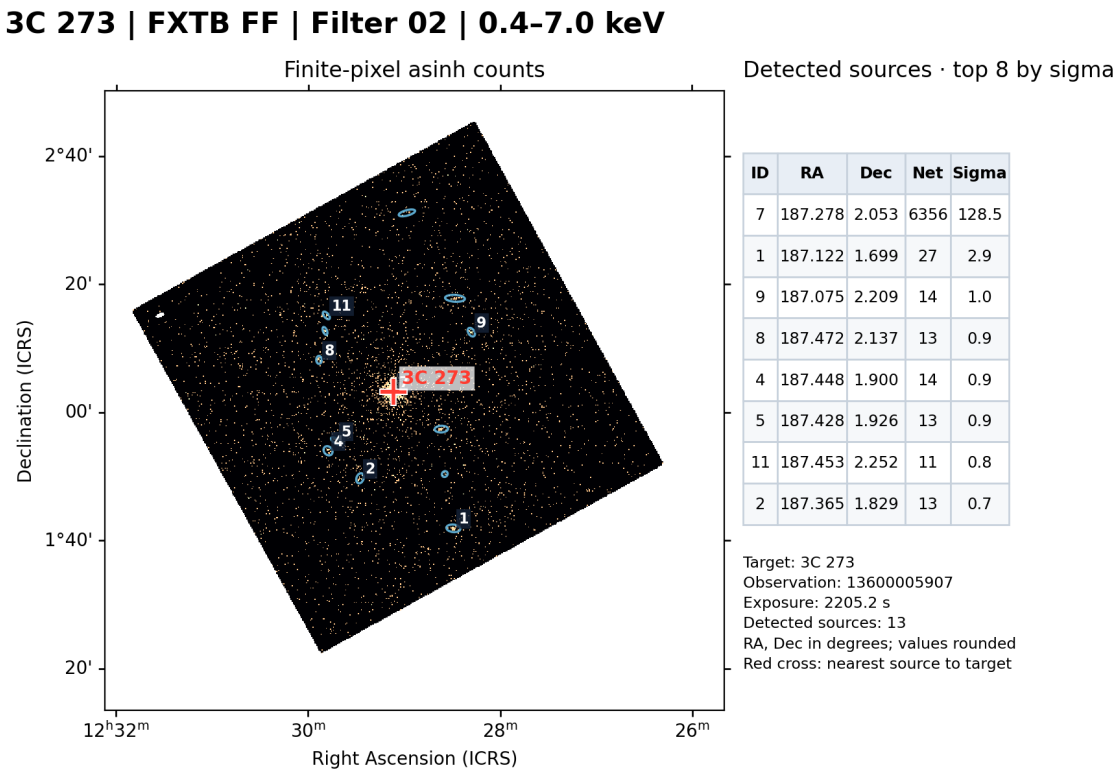

Figure 5.15: `fxtsrcdet` detections in 3C 273. Ellipses mark candidate sources; the red cross marks the detection nearest the target. The right panel lists the eight highest-significance detections.

The value `sigcalfile=AUTO` requires a matching `SRCDETSIGCAL` entry in the configured FXT CALDB and fails if it is unavailable. Use `sigcalfile=CIAO` only for an explicit CIAO comparison. Inspect the source table, significance map, and DS9 regions together before masking: compact features embedded in structured diffuse emission can require manual adjustment. The reviewed point-source regions can then be excluded consistently from count and background products before smoothing, mosaicking, or measuring the extended emission. Point-source fluxes can be estimated with the extraction and response methods described in the Point source section.

method1: FXTSoft source masking and filling. After reviewing the source table and regions, the official `fxtregmaskfill` task can create source and valid-analysis masks and fill the masked regions with a local Poisson estimate:

```
fxtregmaskfill imgfile=fxta-0.4-7.0-cts.fits \
    srcfile=fxta-sources.fits maskout=fxta-source-mask.fits \
    validmaskout=fxta-valid-mask.fits fill=yes method=poisson \
    outfile=fxta-source-filled.fits randseed=123 clobber=yes
```

method2: CIAO source masking and filling. The corresponding CIAO `dmfilth`

route uses the reviewed DS9 source and background regions:

```
dmfilth infile=fxta-0.4-7.0-cts.fits outfile=fxta-source-filled.fits \
    srclist=fxta-sources.reg bkglist=fxta-background.reg method=POISSON
```

The `fxtregmaskfill` task does not support excluded DS9 regions. Its Poisson fill is stochastic, so the result is not expected to be byte-identical to CIAO. In structured diffuse emission, detections and the final masking regions still require manual review.

5.3.2 Spectral analysis

For extended-source spectra, use the explicit DS9/XSELECT response workflow below. Select the source and background regions with DS9/XSELECT, generate the extended-source ARF with `fxtarfgen extend=1`, create the RMF with `fxtrmfgen`, and use `grppha` to attach response metadata and group the spectrum.

Spectra extraction

Select the source and background regions in DS9 and save them as `source.reg` and `bck.reg`. Use these files to extract the spectra in XSELECT. **Caution: do not apply an energy-channel filter during spectrum extraction.** Retain the full channel range so that the spectrum matches its response files.

```
xselect
> Enter session name >[EP] EP  # session name, will be the prefix of later output
> set datadir .
> read events evt-a-cl.fits
> set datamode  FF/PW/TM   # when the mode are not recognized correctly.
> filter time file gti.fits
> filter region source.reg
> extract spectrum
> save spectrum fxta.pi
> clear region
> filter region bck.reg
> extract spectrum
> save spectrum fxta_bck.pi
> quit
```

Response file generation

Then we can calculate the corresponding Ancillary Response File (ARF) with the command *fxtarfgen*. For an extended-source spectrum, the response must use the same extraction aperture as the input PHA and an explicit brightness assumption. The baseline below retains the default uniform-brightness model:

```
cp fxta.pi fxta_uniform.pi
fxtarfgen specfile=fxta_uniform.pi expfile=expmap-without_vign.fits \
  outfile=fxta_uniform_arf.fits extend=1 srcmodel=uniform \
  regfile=source.reg
```

Each variant below starts from a separate PHA copy because `fxtarfgen` updates the supplied PHA's `ANCRFILE`; do not run morphology comparisons sequentially against one PHA.

Reuse exactly the region used to extract `fxta.pi`, including excluded holes. If the PHA contains the matching FITS REGION, an explicit `regfile` is not needed; do not replace an irregular region by an area-equivalent circle. The static `expfile` has a different role from a source-brightness image: for `aspcor=0` it should be the no-vignetting exposure map (for example, the `-without_vign` product with `VIGNAPP=F`), because `fxtarfgen` applies vignetting itself. A vignetting-corrected image used to estimate morphology is not an `expfile`. Products carrying `MERGED=T` or `ASPREQ=T` force attitude weighting; this static `expfile` recipe does not disable it. For merged inputs, follow the response-context and merge instructions in the `fxtarfgen` help.

The current `fxtarfgen` also accepts the supported headerless CIAO/DS9 region convention. Sexagesimal positions or explicit `h/d` units are inferred as celestial FK5/J2000, whereas plain numeric positions retain the physical-coordinate default. Add an explicit `fk5` or `icrs` directive when the frame matters, and include angular units on sizes for inferred sky regions. This is a supported subset of CIAO/DS9 syntax, not a general region-language implementation.

Choosing the spatial model. `srcmodel=uniform` is the default assumption, appropriate when the source brightness is approximately constant across the selected aperture. A circular `gaussian` can describe a smooth, centrally concentrated source; a `beta` profile has a core and longer outer tail. These are assumptions to check against imaging, not automatic choices based on the object name. The `gaussian` and `beta` models require positive, finite, explicitly selected widths; there is no universal 30 or 60 arcsec value, and `fxtarfgen` does not fit a width. Every number in the examples below is illustrative, not a best-fit result for A1795 or any other particular observation.

For a Gaussian, `srcsigma` is the intrinsic-model standard deviation in arcsec: `srcsigma=FWHM/2.35482`. Thus an illustrative intrinsic FWHM of 120 arcsec gives `srcsigma=50.9593`. Estimate the morphology first, with background, exposure, vignetting, and PSF blurring accounted for; a width measured directly from a raw counts image is not necessarily an intrinsic width.

```
cp fxta.pi fxta_gaussian.pi
fxtarfgen specfile=fxta_gaussian.pi expfile=expmap-without_vign.fits \
  outfile=fxta_gaussian_arf.fits extend=1 srcmodel=gaussian \
  srcsigma=50.9593 srccut=8 regfile=source.reg
```

For a beta model, `srcrc` is the core radius in arcsec and the profile is fixed to `S(r)=S0*[1+(r/srcrc)**2]**(-1.5)`, corresponding to `beta=2/3`. There is no free beta-index parameter. A core radius fitted with a different beta in the literature is therefore not interchangeable with `srcrc`.

```
cp fxta.pi fxta_beta.pi
```

```
fxtarfgen specfile=fxta_beta.pi expfile=expmap-without_vign.fits \
  outfile=fxta_beta_arf.fits extend=1 srcmodel=beta \
  srcrc=60 srccut=8 regfile=source.reg
```

For Gaussian and beta models, supply both `srcx` and `srcy` as positive FITS 1-based coordinates in the original FXT SKY/physical pixel system to set the brightness center. These are not RA/Dec in degrees or cropped-image array indices. With both left at `-1`, an included `POLYGON` uses the PHA SPECTRUM `RA_PNT/DEC_PNT`, converted through the PHA celestial/physical WCS. This default is the observation pointing, not a measured brightness peak or the cropped image center. Missing or invalid pointing coordinates require manual center values; other observation metadata required by ARF generation must still be valid.

Without an included polygon, the center is derived from the region; an exclusion-only polygon does not trigger the pointing default. The uniform model and image-model WCS behavior are unchanged, and the PHA extraction region still determines which sky pixels enter the ARF. For an included polygon in `source.reg`, the examples above use the pointing default. To override it, append, for example, `srcx=280.5 srcy=310.5`, replacing these illustrative values with the actual center.

Irregular morphology and image templates. Use `srcmodel=image` when a circular Gaussian or beta profile is not defensible. Fit a source-only morphology after accounting for background, exposure, vignetting, and PSF effects; for an irregular or complex source, a nonnegative fitted template is preferable to a raw-count image. The `srcmap` input must be a celestial-WCS FITS image of finite, nonnegative relative source flux **PER PIXEL**. It is source brightness only, not raw counts, a background image, an exposure map, or a net-count image with negative pixels. Negative net-count pixels are not silently clipped. Bad-pixel or no-exposure holes are not evidence that the intrinsic source brightness is zero; model those regions rather than encoding an unjustified zero. The map must cover the modeled aperture, since pixels outside the map have zero brightness.

```
cp fxta.pi fxta_image.pi
fxtarfgen specfile=fxta_image.pi expfile=expmap-without_vign.fits \
  outfile=fxta_image_arf.fits extend=1 srcmodel=image \
  srcmap=source_template.fits regfile=source.reg
```

All current `extend=1` modes use one energy-independent brightness template, normalize it over the selected extraction aperture, and keep point-source PSF correction disabled. Nonuniform-model ARFs record `SRCNORM=APERTURE`. No full extended-source PSF convolution is applied: small holes, narrow annuli, and compact structures are not repaired by an EEF/PSF factor, and no total flux is extrapolated into excluded subregions or outside the aperture. Multiplying every map pixel by the same positive amplitude does not change the normalized ARF.

Cutoff and sensitivity checks. `srccut` is a dimensionless cutoff multiple, with default 5. The cutoff radius in arcsec is `srccut*srcsigma` for a Gaussian or `srccut*srcrc` for a beta model. Choose `srccut` so that the intended emission reaches the aperture edge; the beta profile has a longer tail. The Gaussian example above reaches about 407.7 arcsec. For illustration, `srcrc=60 srccut=8` reaches 480 arcsec while an independently selected extraction region might extend only 300 arcsec. These are analysis choices, not measurements of A1795. Vary widths over the imaging confidence range and other defensible morphology choices, then compare the ARFs in the fitted energy band and compare the resulting spectral parameters or fluxes. Make a copy of the input PHA before each comparison because `fxtarfgen` updates its `ANCRFILE`. There is no universal fixed pass threshold; report the sensitivity and the reason for the adopted model.

Related tutorials. The following official eROSITA materials are useful method references, not claims of instrument equality or full eSASS-algorithm equivalence: the eSASS cookbook, Section 2.10 (where `exttype=TOPHAT extpars=60` specifies a 60-arcsec disk outer radius, not a Gaussian sigma or beta core radius); the eSASS `srctool` manual, "extpars", "Extended source handling", and "Examples"; and the Jeremy Sanders MPE lecture, pp. 24–30 and p. 38, which summarizes forward modeling, source models, and MAP use for bright nearby systems with PSF convolution disabled. Use the FXT parameter names and normalization rules above when running `fxtarfgen`.

The RMF files could be generated with the command *fxtrmfgen*

```
fxtrmfgen specfile=fxta_uniform.pi outfile=fxta_uniform_rmf.fits
```

For another morphology, replace the uniform PHA, ARF, and RMF filenames below with the corresponding paired files. We could then link the spectrum and its response files with the command *grppha* as below.

```
grppha infile=fxta_uniform.pi outfile=fxta_gr.pi chatter=0 \
comm="group min 10 & chkey RESPFILE fxta_uniform_rmf.fits \
& chkey ANCRFILE fxta_uniform_arf.fits \
& chkey BACKFILE fxta_bck.pi & exit" clobber=yes
```

Spectra fitting

Fit the spectrum in XSPEC as shown below. See the XSPEC manual for details of the commands and models.

```
xspec
>cpd /xw
>data fxta_gr.pi
>show rate
>query yes
```

```
>setplot energy
>abund angr
>notice **-**
>ignore bad
>ignore 0.0-0.5
>ignore 8.0-**
>setplot rebin 3 20
>plot ldata
>statistic cstat
>model tbabs*apec
>freeze 1 4
>thaw 2 3 5
>renorm
>fit 20 1e-2
>setplot device spec.eps /cps
>setplot rebin 3 20
>plot ldata res
>quit
```

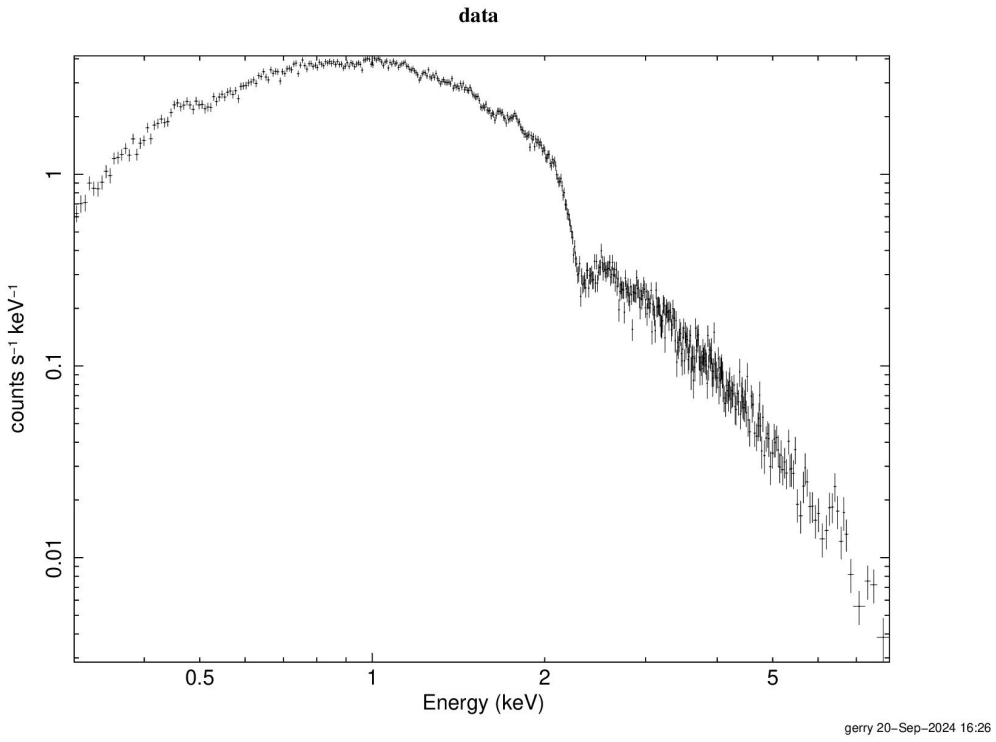

Figure 5.16: Spectrum of the PV target galaxy cluster A3571.

Chapter 6

Calibration Files

6.1 Introduction

The Einstein Probe Follow-up X-ray Telescope (EP-FXT) Calibration Database (CALDB) contains the files used to calibrate and analyze EP-FXT data. These files describe the detector, mirror, filter, and other instrument properties needed by the FXTSoft tasks. The database follows the standard CALDB directory structure, and the CALDB index selects the product appropriate for each observation.

The principal calibration products are:

1) Ancillary Response Files (ARF): Effective collecting area as a function of energy.

2) Redistribution Matrix Files (RMF): The probability distribution of measured channels for an incident photon energy, including detector energy resolution.

3) Point Spread Function (PSF) files: The spatial response to a point source.

4) Encircled Energy Fraction (EEF) files: The fraction of a point source's signal enclosed within a given radius.

5) Telescope Definition (TELDEF) files: Detector and mirror geometry and alignment.

6) Vignetting files: The dependence of effective area on off-axis angle.

7) Filter Transmission (FTRANS) files: Energy-dependent filter transmission.

8) Quantum Efficiency (QE) files: Detector photon-detection efficiency as a function of energy.

9) Gain and Charge Transfer Inefficiency (GAIN) files: Channel-to-energy calibration and charge-transfer corrections.

10) Mirror effective area (EFF) files: Mirror collecting area as a function of energy.

11) Background files: Models and reference products for instrumental and astrophysical backgrounds.

For reproducibility, record the CALDB release together with the FXTSoft version and input observation for every archived product.

6.2 CALDB v2.0 release and maintenance

FXTSoft v2.0 is paired with CALDB v2.0. The command `fxtcaldb-update` can inspect, install, check, or verify a public CALDB release over anonymous HTTPS. Status and verification are read-only operations; installation writes only below the selected installation root.

The current formal revision is `caldb-v2.0-20261103-r1`, released on 2026-11-03. It publishes the already activated formal snapshot; no new scientific data edits were made during packaging. Some historical calibration filenames and FITS provenance keywords retain `beta2` or `2.0_BETA2` to identify their model origin; these labels do not name the current formal release. The root `VERSION` identifies the formal release; the complete archive also includes its provenance in `RELEASE.json`. The complete package and its checksum are available from the CALDB v2.0 download page.

```
fxtcaldb-update command=status install_root=/path/to/CALDB_v2.0
fxtcaldb-update command=verify install_root=/path/to/CALDB_v2.0
fxtcaldb-update command=install ref=latest \
    install_root=/path/to/CALDB_v2.0
```

Use `ref=caldb-v2.0-20261103-r1` instead of `ref=latest` when reproducing this specific revision; `latest` can advance. A stable release ID means that complete identifier, not the literal value `stable`. The 2026-10-20 date retained in the calibration inventory below refers to an earlier validity baseline, not to the current release ID.

After installation, initialize the standard CALDB variables in the same shell as FXTSoft:

```
export CALDB=/path/to/CALDB_v2.0
export CALDBCONFIG=$CALDB/caldb.config
export CALDBALIAS=$CALDB/software/tools/alias_config.fits
```

Do not combine files from different CALDB releases in one analysis unless that comparison is intentional and documented. Re-run affected products after a CALDB update rather than silently reusing products created with an older calibration set.

6.3 Current CALDB v2.0 file inventory

The inventory below describes the paired CALDB v2.0, which contains calibration products for both FXT modules and all supported observing modes. Check the `VERSION` file in the CALDB root for the installed revision. The families below group the indexed content by analysis purpose. A filename shown here is a family or representative product; let the CALDB index select the product for a particular observation.

6.3.1 Detector geometry and event processing

Detector geometry and event reconstruction use TELDEF, bad-pixel, threshold, E2PI-boundary, and GRADEFRACTION products. The threshold inputs include the `fxt_a_threshold_diff*.dat` family, while the E2PI boundary inputs use the `bcf/instrument/e*.txt` family. GRADEFRACTION products record the indexed grade-selection calibration; superseded intermediate text inputs are not selected as CALDB products.

6.3.2 Energy calibration

Energy assignment uses module-, mode-, and date-dependent GAIN products. For FXT-B, the heating boundary is 2026-04-26 15:04:00 UTC. The post-heating GAIN files with `20260520` in their names have an effective start of 2026-04-27 00:00:00 UTC. The date 2026-05-20 records the Fe55 calibration; it is not the heating boundary. Users should allow the CALDB index to choose the valid product for the observation rather than selecting a dated gain file by name.

The indexed `GAIN_POLICY` file specifies the time intervals over which GAIN products may be selected or interpolated. It is read by `fxtpical`; users do not need to edit its rules for routine processing.

The selected GAINCORR file contains the residual energy calibration as well as the time-dependent gain terms. The current revision includes `PWSHIFT_E` and `TMSHIFT_E` energy-dependent curves for PW and TM, together with the legacy `PWSHIFT` and `TMSHIFT` extensions. The preferred curve and its legacy alternative are not added together. FF data do not use the PW/TM residual layer. With an older GAINCORR file lacking both supported residual extensions for a mode, that layer is recorded as unavailable; a malformed extension stops processing. See the fxtpical help for the correction order and output records. To apply updated energy calibration, start from the original input events rather than recalibrating already corrected events.

6.3.3 Throughput and spectral response

Spectral-response files include filter transmission (FTRANS), quantum efficiency (QE), mirror effective area, ancillary response (ARF), and redistribution matrix (RMF) products. These files describe energy-dependent transmission, collecting area, and detector redistribution.

Response calibration distinguishes three standard grade selections:

- `g0`: GRADE 0
- `g4`: GRADE 0–4
- `g12`: GRADE 0–12

The source spectrum, ARF, and RMF should use the same grade selection. Use CALDB to resolve the response for the module, observing mode, filter, and grades in use, rather than choosing a baseline file from its name alone.

The 2026-10-20 calibration baseline established the date-dependent RMFs for FXT-A and the two FXT-B calibration epochs retained in the current snapshot. Their dated filenames describe calibration nodes; the index determines which matrix applies. Generate a new RMF and ARF for the spectrum being fitted after updating the response calibration.

A static ARF provides the baseline effective area. The independent `ARFCORR` product supplies an energy- and time-dependent multiplicative correction to that baseline. It is not a complete response file and should not be applied twice.

The `ARFCORR` files are stored in `bcf/arfcorr/`, with separate extensions for FXT-A and FXT-B with thin or medium filters. The index records the observing mode, reference grade range, filter, and validity interval of each correction. `fxtarfgen` selects the applicable time-dependent correction through `CAL_CNAM=ARFCORR`. Its `arfcorr` option controls whether that correction is applied when available, required, or disabled. The output ARF records the selected static response and correction.

In this release the active correction is a precomputed `ARFCORR1` table with three epoch files. Each has thin- and medium-filter extensions and is indexed for FF, GRADE 0–12. The UTC validity intervals are:

- FXT-A, `ep_fxt_fl1q_beta2_A_arfcorr.fits`: 2024-01-09 07:03:05 through 2026-09-25 21:41:22.
- FXT-B before the heating boundary, `ep_fxt_fl1q_beta2_Bpre_arfcorr.fits`: 2024-01-09 07:03:05 through 2026-04-26 15:04:00.
- FXT-B after the heating boundary, `ep_fxt_fl1q_beta2_Bpost_arfcorr.fits`: 2026-04-26 15:04:00 through 2026-09-25 21:41:23.

Process FXT-B observations crossing that boundary in their respective calibration epochs. These ARFCORR intervals use the same 2026-04-26 15:04:00 UTC heating boundary as the GAIN_POLICY thermal regimes.

The runtime interpolates the tabulated correction; it does not evaluate a new absorption model or extrapolate it to future observations. Beyond the selected file's `VALIDEND`, `arfcorr=auto` skips this correction and records `SKIPPED`, while `arfcorr=required` stops with an error. Thus an ARF produced in `auto` mode is not by itself evidence that the time correction was applied. Inspect its correction status and provenance, especially for observations after 2026-09-25. The release date does not extend the calibration's time coverage.

6.3.4 Spatial response

The spatial-response content includes the compressed `cpf/psf/*.fits.gz` PSF grids, the `fxt?_beta.fits` family, `fxt_psf.cod`, the EEF model `fxt_psf.dat`, and

the vignetting products indexed in CALDB v2.0. PSF, EEF, and vignetting selections depend on the observing configuration and should be resolved through the index.

The module-specific two-beta PSF models and their EEF tables are:

- FXT-A: `fxta_psf_2beta.cod` and `fxta_psf_2beta.dat`
- FXT-B: `fxtb_psf_2beta.cod` and `fxtb_psf_2beta.dat`

They are indexed as `PSFMODEL` and `EEFMODEL`. `fxtpsfmdcopy` copies these A/B model pairs by default; the generic `fxt_psf.cod` and `fxt_psf.dat` pair remains available for legacy use. The off-axis parameter tables `fxta_beta_offaxis.fits` and `fxtb_beta_offaxis.fits` describe the two-beta PSF across energy bands and off-axis angles. These parameter tables are distinct from the sampled PSF images.

The updated thin-filter vignetting files are:

- `ep_fxt_a_normal_vign_20201101_v4.fits`
- `ep_fxt_b_normal_vign_20201101_v4.fits`

Their `VIG_COEF` tables contain `ENERGY`, `COEF0`, `COEF1`, and `ALPHA`; the added `ALPHA` column allows the radial dependence to vary in the in-flight model. The vignetting correction enters exposure-map and response calculations. Use a matching FXTSoft version that reads this column, and let the index choose the file for the filter in use. Older tables without `ALPHA` retain the previous model with an exponent of 2.

6.3.5 Analysis support

Background products are stored in `bcf/bkgmodel/`. Source-detection significance uses `fxt_srcdet_sigcal_v001.fits`. This file is stored in `cpf/instrument/` and indexed as `SRCDETSIGCAL`. It is used by `fxtsrcdet` for calibrated detection thresholds; it is not a background model. An updated CALDB v2.0 package will add blank-sky reference observations as `data/ep/fxt/bcf/bkg/blanksky.tar.gz`. Older packages may not contain this directory; consult the documentation or file manifest supplied with the downloaded package to confirm its presence and contents.

6.4 Calibration file structure, naming, and selection

The calibration database root is selected by the `CALDB` environment variable. FXT content is stored below `data/ep/fxt/`.

The configured index, `data/ep/fxt/caldb.indx`, is stored directly under the FXT directory, not in a separate `index/` subdirectory. Basic calibration files are stored in `bcf`, and calibration product files in `cpf`.

Base path	Instrument	Directory	Contents
$CALDB/data/ep/	fxt/	(root)	caldb.indx
$CALDB/	(root)	bkg/	blank-sky observation data (updated packages)
$CALDB/data/ep/	fxt/	bcf/	badpix/, teldef/, effarea/, ftrans/, qe/, gain/
$CALDB/data/ep/	fxt/	bcf/	arfcorr/, bkgmodel/, instrument/
$CALDB/data/ep/	fxt/	cpf/	psf/, arf/, rmf/, eef/, vign/, instrument/

Table 6.1: CALDB directory structure

The naming convention for calibration files stored in FXT's calibration database follows this format:

`FXT_module_mode_datatype_[date]_v<version>.ext`

Here are the components of this naming convention:

Module: Represents different modules, labeled as 'a' or 'b'.

Mode: Describes the observation mode. 'cal' stands for calibration, 'ff' for full frame, 'pw' for partial window, and 'tm' for timing mode.

Datatype: Indicates the type of data. For example, 'gain' corresponds to a gain file, and 'psf' corresponds to a PSF (Point Spread Function) file.

Date: The applicable date of the calibration file, formatted as YYYYMMDD.

Version: Represents the version information.

Ext: Represents the extension name. By default, it is 'fits'. Exceptions are 'RMF' for response matrix files, 'ARF' for effective area files (also known as auxiliary response files), and 'teldef' for telescope definition files.

The calibration index selects products using the module, observing mode, filter, and observation date recorded for the dataset. When several dated products exist, use the indexed validity interval instead of choosing the newest-looking filename.

Vignetting files are stored in `data/ep/fxt/cpf/vign/`. The index records this directory in `CAL_DIR` and identifies the file type as `CAL_CNAM=VIGNET`.

For the design of the EP-FXT Calibration Database, see Li et al. 2025 [1].

Chapter 7

FXT Pipeline: fxtchain tool

The `fxtchain` task performs the standard FXT data reduction. It searches the input directory for EVT, MKF, ORB, and ATT files, then runs `fxtcoord`, `fxttimecorr` (TM mode only), `fxtpical`, `fxtparticleidentify`, `fxtbadpix`, `fxthotpix`, `fxtgrade`, `fxtgtigen`, and `xselect` for each selected EVT or FSAEVT file. For ordinary EVT files with `exitstage=2`, it also runs `fxtexpogen`, `fxtarfgen`, and `fxtrmfgen`. The task saves the resulting products in the output directory.

By default, `fxtchain` produces cleaned Level-2 events. Set `exitstage=2` to request the higher-level products described below. For FSAEVT input, the stage-2 branch retains the cleaned FSA event and spectrum products but does not create the ordinary event light curve, image, exposure map, ARF, or RMF products. Keep the observation ID, software and CALDB versions, and analysis settings with the output products.

The bundled HEASoft XSELECT has long-path limitations. Use short output and work paths and a short writable PFILES user directory. This release does not modify XSELECT. Native fxtcoord TNULL metadata warnings also remain; no C++ metadata or scientific calculation is changed in this maintenance update.

Figure 7.1 shows the execution sequence. The command templates below illustrate the pipeline's task calls: replace brace- and angle-bracket placeholders with actual values, and join wrapped filename segments before running an individual command. The Usage section provides examples of the `fxtchain` entry point.

- fxtcoord
  fxtcoord is the first task to be executed, and the command executed in fxtchain is as follows:

```
$fxtcoord  evtfile={your_input_dir}/fxt/event/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_uf_evt_<ver>.fits \
    attfile={your_input_dir}/auxil/ep_<obsID>_att_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID>_<datamode> \
    _<filter>_po_evt_coord_<ver>.fits
```

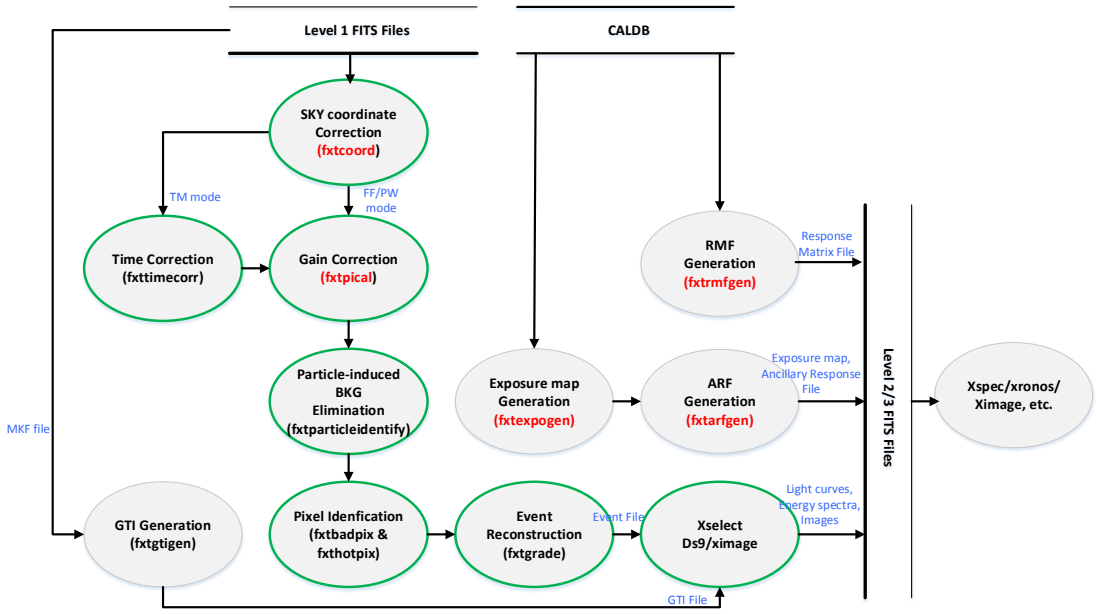

Figure 7.1: Data-reduction flow of `fxtchain`.

If `module` and `datamode` are specified, the tasks run only on matching EVT files; otherwise, all EVT files are processed. This applies to the subsequent tasks as well.

- fxttimecorr
  This task is only applicable to EVT data in TM mode, and the command to execute in fxtchain is:

```
$fxttimecorr ra={RA} dec={DEC} \
    evtfile={your_output_dir}/fxt_<module>_<obsID>_<datamode> \
    _<filter>_po_evt_coord_<ver>.fits \
    attfile={your_input_dir}/auxil/ep_<obsID>_att_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID>_<datamode> \
    _<filter>_po_evt_timecorr_<ver>.fits
```

  Set `ra` and `dec` together when passing them through `fxtchain`; if either value is missing, both are ignored and `fxtchain` reads the position from the EVT file.

- fxtpical
  For EVT data in TM mode, the command to execute is:

```
$fxtpical evtfile={your_output_dir}/fxt_<module>_<obsID> \
```

```
    _<datamode>_<filter>_po_evt_timecorr_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID>_<datamode> \
    _<filter>_po_evt_pi_<ver>.fits
```

For data in FF and PW mode, use:

```
$fxtpical evtfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_coord_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_pi_<ver>.fits
```

The input EVT file is the only difference; TM mode also runs `fxttimecorr`.

- fxtparticleidentify
  The command to execute in fxtchain is:

```
$fxtparticleidentify evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_evt_pi_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_particle_<ver>.fits \
    xlength={X_LENGTH} ylength={Y_LENGTH}
```

  `fxtchain` sets `xlength` and `ylength` automatically: both are 35 for TM and 11 for FF/PW.

- fxtbadpix
  The command to execute in fxtchain is:

```
$fxtbadpix evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_evt_particle_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_badpix_<ver>.fits \
    userbpfile={USERBPFILE}
```

  If `userbpfile` is specified for `fxtchain`, it is passed to `fxtbadpix`. Otherwise, `fxtbadpix` uses only `evtfile` and `outfile`.

- fxthotpix
  The command to execute in fxtchain is:

```
$fxthotpix evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_evt_badpix_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_hotpix_<ver>.fits
```

- fxtgrade
  The command to execute in fxtchain is:

```
$fxtgrade evtfile={your_output_dir}/fxt_<module> \
    _<obsID>_<datamode>_<filter>_po_evt_hotpix_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_evt_grade_<ver>.fits \
    pithresh=0
```

- fxtgtigen
  The command to execute in fxtchain is:

```
$fxtgtigen mkffile={your_input_dir}/fxt/hk/fxt \
    _<obsID>_mkf_<ver>.fits \
    outfile={your_output_dir}/fxt_<module>_<obsID> \
    _<datamode>_<filter>_po_gti_<ver>.fits \
    module={MODULE} expr={EXPR} usergtifile={USERGTIFILE}
```

  For `fxtgtigen`, `module` is taken from the EVT file being processed. Its values are `fxta` and `fxtb`, rather than `a` and `b`. The `expr` and `usergtifile` values are passed through from `fxtchain`.

- xselect
  The command to execute in fxtchain is:

```
$xselect @{your_output_dir}/xselect_run_<module> \
    _<obsID>_<datamode>_<filter>_po_<ver>.txt
```

  The commands used during the xselect filtering process are written into a file, and its contents are mainly influenced by the parameters exitstage, srcffregion, srcpwregion, srctmregion, bkgffregion, bkgpwregion, and bkgtmregion.
  When exitstage is unassigned or set to 1, and the parameters srcffregion, srcpwregion, srctmregion, bkgffregion, bkgpwregion, and bkgtmregion are all unassigned, the contents of the file are as follows:

```
EP
set datadir {your_output_dir}
read events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_evt_grade_<ver>.fits
yes
set binsize {binsize}
filter grade {grade}
filter time file ./fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits
select event "status==b0"
extract events copyall=yes
```

```
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits
no
# exit xselect
clear all proceed=yes
quit
no
```

The variables `your_output_dir`, `binsize`, and `grade` in the file are taken from the corresponding `fxtchain` parameters. If `binsize` is not assigned, it defaults to 1; if `grade` is not assigned, it defaults to 0–12.
When `exitstage`=2 and none of `srcffregion`, `srcpwregion`, `srctmregion`, `bkgffregion`, `bkgpwregion`, and `bkgtmregion` is assigned, the file contains:

```
EP
set datadir {your_output_dir}
read events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_evt_grade_<ver>.fits
yes
set binsize {binsize}
filter grade {grade}
filter time file ./fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits
select event "status==b0"
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits
no
extract spectrum copyall=yes
save spectrum ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pha
filter pha_cutoff 38 925
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_tmp_<ver>.fits
no
extract curve copyall=yes
save curve ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.lc
extract image xysize=601 xybinsize=1 xcenter=300 ycenter=300 copyall=yes
save img ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.img
clear pha_cutoff
clear events
# exit xselect
clear all proceed=yes
quit
no
```

When `exitstage` is unassigned or equals 1, and selected regions for the currently processed EVT data are provided through `srcffregion`, `srcpwregion`,

srctmregion, bkgffregion, bkgpwregion, and bkgtmregion, the file contains:

```
EP
set datadir {your_output_dir}
read events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_evt_grade_<ver>.fits
yes
set binsize {binsize}
filter grade {grade}
filter time file ./fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits
select event "status==b0"
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits
no
# if src region is defined
filter region {your_src_region_file}
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_src_<ver>.fits
no
clear region
clear events
# if bkg region is defined
filter region {your_bkg_region_file}
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_bkg_<ver>.fits
no
clear region
clear events
# exit xselect
clear all proceed=yes
quit
no
```

When exitstage=2 and selected regions for the currently processed EVT data are provided through srcffregion, srcpwregion, srctmregion, bkgffregion, bkgpwregion, and bkgtmregion, the file contains:

```
EP
set datadir {your_output_dir}
read events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_evt_grade_<ver>.fits
yes
set binsize {binsize}
filter grade {grade}
```

```
filter time file ./fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits
select event "status==b0"
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits
no
# if src region is defined
filter region {your_src_region_file}
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_src_<ver>.fits
no
clear region
clear events
# if bkg region is defined
filter region {your_bkg_region_file}
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_bkg_<ver>.fits
no
clear region
clear events
extract spectrum copyall=yes
save spectrum ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pha
filter pha_cutoff 38 925
extract events copyall=yes
save events ./fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_tmp_<ver>.fits
no
extract curve copyall=yes
save curve ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.lc
extract image xysize=601 xybinsize=1 xcenter=300 ycenter=300 copyall=yes
save img ./fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.img
clear pha_cutoff
clear events
# if src region is defined
filter region {your_src_region_file}
extract all copyall=yes
save spectrum ./fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.pha
save curve ./fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.lc
clear region
clear events
# if bkg region is defined
filter region {your_bkg_region_file}
extract all copyall=yes
save spectrum ./fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.pha
save curve ./fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.lc
```

```
clear region
clear events
# exit xselect
clear all proceed=yes
quit
no
```

- fxtexpogen
  The command to execute in fxtchain is:

  ```
  $fxtexpogen evtfile={your_output_dir}/fxt_<module> \
      _<obsID>_<datamode>_<filter>_po_cl_<ver>.fits \
      outfile={your_output_dir}/fxt_<module>_<obsID> \
      _<datamode>_<filter>_po_<ver>.expo \
      mkffile={your_input_dir}/fxt/hk/fxt \
      _<obsID>_mkf_<ver>.fits
  ```

  The evtfile here is the cleaned event file obtained after executing xselect.

- fxtarfgen
  The command to execute in fxtchain is:

  ```
  $fxtarfgen {your_output_dir}/fxt_<module> \
      _<obsID>_<datamode>_<filter>_po_<ver>.pha \
      {your_output_dir}/fxt_<module>_<obsID> \
      _<datamode>_<filter>_po_<ver>_without_vign.expo \
      {your_output_dir}/fxt_<module>_<obsID> \
      _<datamode>_<filter>_po_<ver>.arf
  ```

- fxtrmfgen
  The command to execute in fxtchain is:

  ```
  $fxtrmfgen specfile={your_output_dir}/fxt_<module> \
      _<obsID>_<datamode>_<filter>_po_<ver>.pha \
      outfile={your_output_dir}/fxt_<module>_<obsID> \
      _<datamode>_<filter>_po_<ver>.rmf
  ```

- powspec
  The command to execute in fxtchain is:

  ```
  $powspec norm=1 \
      cfile={your_output_dir}/fxt_<module>_<obsID> \
      _<datamode>_<filter>_po_cl_tmp_<ver>.fits \
      window=- dtnb=<DTNB> nbint=<NBINT> nintfm=INDEF \
  ```

```
rebin=-1.005 plot=no \
outfile={your_output_dir}/fxt_<module>_<obsID> \
_<datamode>_<filter>_po_<ver>.pds
```

`fxtchain` does not accept user input for `dtnb` or `nbint`; their values depend on the mode of the EVT file being processed:

Parameter	FF	PW	TM
dtnb	0.05	0.0022	0.001953125/4.0
nbint	2048	4096*8	4096*32

Table 7.1: The values of parameters.

Most `fxtchain` processing options are optional and use the defaults listed below. The command-line interface requires `indir`, `outdir`, `exitstage`, and `expr`; if any of these are omitted, interactive mode prompts for them before starting the reduction.

PARAMETER

The parameters of this tool are listed below:

- indir [string]
  Directory containing the EVT, MKF, ORB, and ATT files in the observation layout shown below. Missing required files stop the reduction.

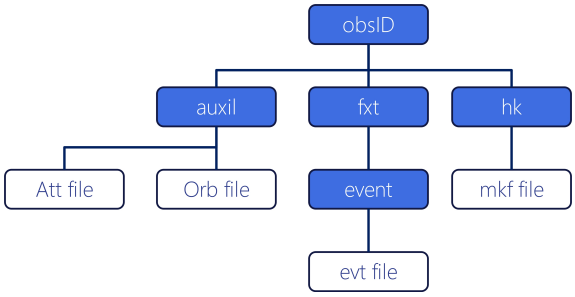

Figure 7.2: Input directory hierarchy.

  If omitted, `fxtchain` prompts for this directory.

- outdir [string]
  Directory for reduced events, GTIs, and any requested spectra, images, and light curves. You need write permission; `fxtchain` creates the directory if it does not exist. Figure 7.3 relates the output products to the input options.

  If omitted, `fxtchain` prompts for this directory.

FileName	Parameter(The relationship between multiple parameters is "AND")					
	exitstage=	module=	datamode=	datatype=	src<datamode>region=	bkg<datamode>region=
fxt_<module>_<obsID>_<datamode>_<filter>_po_gti_<ver>.fits	1 or 2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_<ver>.fits	1 or 2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.img	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.img.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.lc	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.lc.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pha	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pha.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.expo.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pds	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.pds.png	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>-without_vign.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.arf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_<ver>.rmf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_src_<ver>.fits	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.lc	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.pha	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.img	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>-without_vign.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.arf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_src_<ver>.rmf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	{src region file}	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_cl_bkg_<ver>.fits	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.lc	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.pha	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.img	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>-without_vign.expo	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.arf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_bkg_<ver>.rmf	2	<module> or unassigned	<datamode> or unassigned	evt or unassigned	-	{bkg region file}
fxt_<module>_<obsID>_<datamode>_<filter>_po_fsa_cl_<ver>.fits	1 or 2	<module> or unassigned	<datamode> or unassigned	fsaevt	-	-
fxt_<module>_<obsID>_<datamode>_<filter>_po_fsa_<ver>.pha	2	<module> or unassigned	<datamode> or unassigned	fsaevt	-	-

Figure 7.3: Mapping between `fxtchain` input parameters and output files.

- exitstage [int]
  Set `exitstage=1` (default) to stop after cleaned events, or `exitstage=2` to generate higher-level products. For `datatype=fsaevt`, stage 2 does not generate ordinary event exposure or response products.

- module [string]
  Process only the selected module: `a` or `b`, for example `module=a`. If omitted, both modules are processed.

- datamode [string]
  Process only matching modes, case-insensitively. The standard science modes are FF, PW, and TM; the current filename matcher also recognizes DF, DT, DP, SV, and DS auxiliary mode codes. Separate multiple modes with commas, for example `datamode=FF,PW`.

- datatype [string]
  Specifies the file types to process. The default is `evt`, so only EVT files are processed. The other supported value is `fsaevt`. To process both types, separate them with a comma, for example: `datatype=evt,fsaevt`.

- binsize [float]
  Bin size (seconds) for the light curve, default value is 1.

- ra [float]
  Source right ascension in degrees for TM time correction. Supply `ra` and `dec` together; if either is missing, the pipeline uses the EVT header position instead.

- dec [float]
  Source declination in degrees for TM time correction. Supply it together with `ra`.

- grade [string]
  The value of grade is for missions with CCD detectors whose events are assigned a grade. The grade can be specified as a single number (eg 0), a range (eg 0:2 or 0-2), an upper limit (eg <4), or a lower limit (eg >3). Specifications should be separated by commas (eg 0,2-6) and enclosed in double quotes if given on the command line instead of when prompted.

- expr[string]
  The selection expression is applied to the attitude and instrument HK data. Use `DEFAULT` (the default) to load the standard expression from CALDB. To avoid parsing issues with special characters (for example, &, or >), enclose the value in single quotes (for example, `expr='ELV>-10'`). Quotes may be omitted when the value contains no special characters, as in `expr=DEFAULT`.

- usergtifile[string]
  The path of the input GTI file (FITS format) or NONE for none

- userbpfile[string]
  The path of the user BADPIX file, or `NONE` for none.

- srcffregion[string]
  This region is used to define a source filter when the currently processed EVT file is in FF mode. The region filter is in ds9 format. Region filters for the filter region command can be created graphically via the plot image command, which spawns a ds9 session.

- srcpwregion[string]
  DS9 source-region file for PW data. Create the region with DS9, for example through XSELECT's `plot image` command.

- srctmregion[string]
  DS9 source-region file for TM data. Create the region with DS9, for example through XSELECT's `plot image` command.

- bkgffregion[string]
  This region is used to define a background filter when the currently processed EVT file is in FF mode. The region filter is in ds9 format. Region filters for the filter region command can be created graphically via the plot image command, which spawns a ds9 session.

- bkgpwregion[string]
  DS9 background-region file for PW data. Use a source-free region appropriate to the observation.

- bkgtmregion[string]
  DS9 background-region file for TM data. Use a source-free region appropriate to the observation.

- outputlist[string]
  Controls whether an output product list is written. Values `yes` or `Y` write the list; `no` or `N` do not. The default is `N`.

- savetmp[string]
  Controls whether temporary files are kept. Values `Y` or `yes` keep them; `N`, `no`, or an unassigned value remove them.

7.1 Usage

1. The basic `fxtchain` command uses these input parameters:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=1 \
    expr=DEFAULT
```

   Here `your_input_path` contains the observation files and `your_output_path` is the output directory. The default run produces cleaned events and GTIs for FF, PW, and TM using the CALDB selection expression. For TM data, provide both coordinates of the intended source. If they are omitted, the pipeline reads `RA_OBJ`/`DEC_OBJ`, then falls back to `RA_PNT`/`DEC_PNT`; the pointing position may differ from the target position. For example:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_dir} \
    exitstage=1 \
    expr=DEFAULT \
    ra=161.265 \
    dec=-59.685
```

2. The following command produces the higher-level products for ordinary EVT input; `exitstage` is set to 2:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_dir} \
    exitstage=2 \
    expr=DEFAULT
```

3. The following command shows how to obtain module a/b files:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    module=a
```

4. In this example fxtchain only runs a data mode:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    datamode=ff
```

5. `fxtchain` can also process FSAEVT files:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    datatype=fsaevt
```

   For FSAEVT input, this stage-2 run produces the cleaned FSA event and spectrum products. It does not create the ordinary event light curve, image, exposure map, ARF, or RMF products; use the dedicated particle-background workflow for subsequent FSA analysis.

6. Extract source and background spectra:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    datamode=ff \
    srcffregion={your_ff_src_region} \
    bkgffregion={your_ff_bkg_region}
```

   This example processes FF data, so PW and TM region options have no effect. To process all available modes, omit `datamode` and supply a region for each required mode:

```
$fxtchain indir={your_input_path} \
    outdir={your_output_path} \
    exitstage=2 \
    expr=DEFAULT \
    srcffregion={your_ff_src_region} \
    bkgffregion={your_ff_bkg_region} \
    srctmregion={your_tm_src_region} \
    bkgtmregion={your_tm_bkg_region} \
    srcpwregion={your_pw_src_region} \
    bkgpwregion={your_pw_bkg_region}
```

7. Interactive mode only accepts four mandatory parameters:

```
$fxtchain



The input directory : {your_input_path}
The argument indir has been set to {your_input_path}

The output directory : {your_output_path}
The argument outdir has been set to {your_output_path}

The exit stage [1] :
The argument exitstage has been set to 1

The value of expression [DEFAULT] :
The argument expr has been set to DEFAULT
```

When prompted, enter values for `indir`, `outdir`, `exitstage`, and `expr`. Defaults are provided for `exitstage` and `expr`; press Enter to accept them. A parameter already given on the command line is not requested again in interactive mode.

Appendix A

Installation of FXTSoft v2.0

The following prerequisites are required to install this software:

- Sufficient disk space for the source, build tree, installed software, and CALDB. Requirements depend on the platform and selected HEASoft packages; consult the HEASoft installation guides for the chosen release.
- X11 / X Windows:
  Graphical tools such as XSPEC, XIMAGE, FV, and fplot require X11. See X.Org or, on macOS, XQuartz.
- CPython 3.11 or newer. The selected interpreter must be used both when configuring the software and at runtime. The Python workflows also use NumPy, SciPy, Astropy, and Matplotlib.

A.1 Source-code installation

A.1.1 Ubuntu

The current FXTSoft v2.0 source package is published at `https://epfxt.ihep.ac.cn/downloads/fxtsoft/2.0/`. It can be compiled on supported Ubuntu and other Linux systems with the corresponding HEASoft prerequisite packages.

1. **Prerequisite packages**

   Before the first installation, install the packages required to build FXTSoft from the source distribution:

```
sudo apt-get -y install libreadline-dev
sudo apt-get -y install libncurses5-dev
sudo apt-get -y install libx11-dev
sudo apt-get -y install libxt-dev
sudo apt-get -y install ncurses-dev
```

```
sudo apt-get -y install curl
sudo apt-get -y install libcurl4
sudo apt-get -y install libcurl4-gnutls-dev
sudo apt-get -y install xorg-dev
sudo apt-get -y install make
sudo apt-get -y install gcc g++ gfortran
sudo apt-get -y install perl-modules
```

2. **Unpacking**

   Unpack the downloaded archive, for example:

   ```
   tar zxf fxtsoft_v2.0.tar.gz
   ```

   This creates a `fxtsoftv2.0/` directory containing the source code. Keep the downloaded archive unchanged so that the unpacked source can be restored if a build must be restarted.

3. **Building**

   Select one CPython 3.11-or-newer executable before configuration. Bind both `PYTHON` and `FXT_PYTHON` to its absolute path so that configure-time and run-time selection agree. Preserve the executable path inside a selected virtual environment; resolving its final symlink to the base Python would bypass that environment. Then run the main configure script in `BUILD_DIR`. Replace `/path/to/install` with a user-writable installation prefix.

   ```
   python3.11 -VV
   export PYTHON="$(python3.11 -c \
       'import os,sys; print(os.path.abspath(sys.executable))')"
   export FXT_PYTHON="$PYTHON"
   cd fxtsoftv2.0/BUILD_DIR
   ./configure --prefix=/path/to/install > config.txt 2>&1
   make > build.txt 2>&1
   make install > install.txt 2>&1
   ```

   Choose the final installation prefix before running `configure`. A source installation records absolute installation and build paths in system parameter files and development metadata such as `BUILD_DIR/hmakerc`, pkg-config files, libtool archives, Tcl/Tk/FFTW configuration, and local-model build settings. A complete source build in a user's own final prefix does not inherit another user's paths. Moving an existing tree, changing only `HEADAS` or an initialization script, or changing ELF RPATHs does not rewrite these text files. Keep the source and build dependencies needed by XSPEC `initpackage` and `hmake`, or rebuild and install into the final prefix.

Before using a copied or moved tree, run the read-only installation audit from the source package:

```
python3 distribution/relocate_install.py \
    --root /absolute/path/to/fxtsoft-install \
    --report relocation-audit.json
```

The audit must be clean before release. If a move is intentional, provide explicit old and new prefixes and an installation-tree-external backup directory; the tool validates referenced destinations before changing supported text configuration. This is a text-configuration repair, not an ELF, ABI, or native-library migration. It does not replace missing build dependencies or overwrite learned user parameter files. Back up a user's `PFILES` files before using `punlearn` to reset selected tasks, then repeat the audit and the command, XSPEC local-model, and science smoke checks.

Python prerequisites for `caldb_tools` The FXT Python tools require the following minimum package versions:

```
NumPy >= 1.25
Astropy >= 6.1
SciPy >= 1.15
regions >= 0.11
reproject >= 0.20
```

After selecting the interpreter above, install these dependencies from the FXTSoft source tree:

```
"$PYTHON" -m pip install -r fxt/caldb_tools/requirements.txt
```

4. **Initialization**

   For Bourne-compatible shells, initialize the installed platform directory before running FXT tasks. Replace `PLATFORM` with the platform directory created by the build.

   ```
   export HEADAS="/path/to/install/PLATFORM"
   source "$HEADAS/headas-init.sh"
   ```

   These settings apply only to the current shell. Add them to your .bashrc file if they should be available in new shells.

```
export HEADAS="/path/to/install/PLATFORM"
alias fxtinit="source $HEADAS/headas-init.sh"
```

You can then initialize FXTSoft with the `fxtinit` command.

5. **Configure CALDB v2.0**

   CALDB is distributed separately from the software. The formal CALDB v2.0 release page is https://epfxt.ihep.ac.cn/downloads/CALDB_v2.0.html; download the archive from https://epfxt.ihep.ac.cn/downloads/CALDB_v2.0.tar.gz, install or unpack it in a stable location, and set the following variables in the initialized FXTSoft shell:

   ```
   export CALDB=/path/to/CALDB_v2.0
   export CALDBCONFIG=$CALDB/caldb.config
   export CALDBALIAS=$CALDB/software/tools/alias_config.fits
   ```

   The updated CALDB package will provide blank-sky reference observations under its package-root `bkg/` directory; older packages may not include this directory, so check the downloaded package documentation or file manifest.

   Confirm the installation and local offline help before processing science data:

   ```
   fxtcaldb-update command=verify install_root=$CALDB
   fhelp -m=cat -l fxtchain
   ```

Note: Do not compile or install the code on an NTFS partition. NTFS does not allow colons (:) in filenames, which can cause the following CFITSIO compilation error: "Can't write-open blib/man3/Astro::FITS::CFITSIO.3pm"

A.1.2 RHEL-compatible Linux (AlmaLinux, Rocky Linux, CentOS Stream)

The current IHEP FXTSoft v2.0 runtime has been verified on AlmaLinux 9.8. On AlmaLinux and other RHEL-compatible RPM-based systems, install the corresponding compiler and development packages with `dnf`, and select CPython 3.11 or newer before configuration. Rocky Linux and CentOS Stream use the same package-family instructions; validate the complete build on the target release before use.

A.1.3 macOS

On macOS, use a supported Xcode command-line toolchain and a CPython 3.11-or-newer environment. Set `PYTHON` and `FXT_PYTHON` to the same interpreter before running configure.

Python prerequisites for `caldb_tools` The FXT Python tools require the following minimum package versions:

```
NumPy >= 1.25
Astropy >= 6.1
SciPy >= 1.15
regions >= 0.11
reproject >= 0.20
```

Install these dependencies with the interpreter selected for the build:

```
"$PYTHON" -m pip install -r fxt/caldb_tools/requirements.txt
```

You can unpack the file you downloaded using e.g.:

```
tar zxf fxtsoft_v2.0.tar.gz
```

A `fxtsoftv2.0` directory containing the source code will be created.

Configure the software for your platform from the `fxtsoftv2.0/BUILD_DIR` directory. The configure script checks for the required libraries, headers, and compilers and generates the Makefile. Set `--prefix` to the installation directory. Keep the architecture of the compilers, Python, and libraries consistent. Use `arch -x86_64 /bin/bash` only when intentionally building an Intel-compatible environment with matching dependencies.

```
cd fxtsoftv2.0/BUILD_DIR
./configure --prefix=/path/to/install
make
make install
```

A.2 FXT installation with an existing HEASoft environment

The FXT component can be compiled into a separate installation prefix using an existing HEASoft environment. Use the `fxt/BUILD_DIR` configuration entry and supply the installed HEASoft platform directory through `--with-hea-dir`. The HEASoft installation must provide its development headers, libraries, and installed `BUILD_DIR/hmakerc` metadata for `heacore`, `heatools`, `attitude`, and `heagen`. If the metadata refers to WCSLIB headers in the original HEASoft build directory, retain that directory. Compilers, Python, and HEASoft libraries must use matching architectures.

Select a Python 3.11-or-newer interpreter with the dependencies listed above. Use absolute paths without spaces or shell special characters. The following example starts from a source package containing `fxt/BUILD_DIR` and `fxt/caldb_tools`:

```
export HEADAS=/path/to/heasoft/PLATFORM
. "$HEADAS/headas-init.sh"
export PYTHON=/path/to/python3
export FXT_PYTHON="$PYTHON"
export FXT_DIR=/path/to/fxt-install
cd /path/to/fxt/BUILD_DIR
"$PYTHON" -m pip install -r ../caldb_tools/requirements.txt
./configure --prefix="$FXT_DIR" --with-hea-dir="$HEADAS"
./hmake
./hmake install
. "$FXT_DIR/fxtdas-init.sh"
```

Check that each command succeeds before proceeding. The installation puts the FXT commands, Python package, help pages, and system parameter files under `FXT_DIR`. The initialization script preserves `HEADAS`, CALDB settings, and the user parameter directories, and adds the FXT command, Python, and system parameter paths. Initialize HEASoft followed by `fxtdas-init.sh` in each shell that will run FXT commands. Configure CALDB separately as described above. The external HEASoft XSELECT mission database must contain the EP/FXT definitions when XSELECT is used; this installation does not modify that database.

```
command -v fxtpical fxtsrcdet fxtchain
"$FXT_PYTHON" -c \
    'import caldb_tools; print(caldb_tools.__file__)'
fxtsrcdet --help
```

FXT commands should resolve to `FXT_DIR/bin`, and the Python package to `FXT_DIR/lib/python`. Verify compilation, initialization, and the required data-processing workflow on the target operating system and architecture.

A.3 Updating the installed Python tools

`fxttool-update` manages the published `caldb_tools` component: its Python commands, libraries, system parameter files, and help. CALDB data are managed separately by `fxtcaldb-update`. After initializing a complete FXTSoft installation, use:

```
fxttool-update command=status
fxttool-update command=check
fxttool-update command=install
fxttool-update command=verify
```

For an independent FXT prefix, specify the directory containing `bin/`, `lib/python/caldb_tools/`, and `syspfiles/`:

```
fxttool-update command=status install_root="$FXT_DIR" layout=single
fxttool-update command=check install_root="$FXT_DIR" layout=single
fxttool-update command=install install_root="$FXT_DIR" layout=single
fxttool-update command=verify install_root="$FXT_DIR" layout=single
```

`status` reports local state without network access. `check` reads remote release metadata and computes the update plan without downloading payload files or changing installed files. `install` downloads changed files, validates them, and installs them transactionally; `dry_run=yes` displays the plan without applying it. `verify` checks files against the recorded installed release. `command=rollback` restores the preceding transaction when a rollback is available. Use the same `install_root`, `layout`, and any explicitly chosen `state_dir` for subsequent operations. A complete FXTSoft installation uses the standard top-level and FXT platform directories; `layout=single` is intended for an independent prefix.

A.4 Docker installation

Docker provides a containerized runtime for FXTSoft and its software dependencies.

Install Docker Engine before running FXTSoft in a container. Once Docker is ready, obtain the FXTSoft image from the FXTSoft team and import it. Replace `fxtsoft-image.tar` below with the downloaded filename:

```
docker load -i fxtsoft-image.tar
```

After loading, list the local images with:

```
docker images
```

Start a container from the image name or ID shown by `docker images`. Replace `repository:tag` and the host paths below with your local values:

```
docker run -v /path/to/CALDB_v2.0:/caldb \
    -v /path/to/input:/input \
    -v /path/to/output:/output \
    -it repository:tag /bin/bash
```

The `-v` option mounts a host directory inside the container; the host and container paths are separated by a colon. The example uses three mount points, but fewer can be used when appropriate. Mount the CALDB directory at `/caldb` when possible, because the container is configured to use that path. If CALDB is mounted elsewhere, update the CALDB environment after starting the container.

The `-it` option combines interactive mode (`-i`) with a pseudo-terminal (`-t`).

On startup, the container executes `/bin/bash` and opens a shell in which FXT data-reduction tasks can be run.

After leaving a container, find its ID and re-enter it with the commands below. Replace `CONTAINER_ID` with the value reported by `docker ps -a`.

```
docker ps -a
docker start CONTAINER_ID
docker exec -it CONTAINER_ID /bin/bash
```

If a container is no longer needed, stop and remove it as follows. First save any needed files that are not in a mounted host directory; removing the container deletes its writable layer.

```
docker stop CONTAINER_ID
docker rm CONTAINER_ID
```

Appendix B

FXTSoft v2.0 Command Reference

This appendix lists the commands and parameters available in FXTSoft v2.0. The online command help gives detailed usage, examples, and limitations. Older command names retained for compatibility are listed separately from the commands recommended for new scripts.

B.1 Core mission tasks

B.1.1 `fxtcheck`

Checks the HEADAS and CALDB environment, finds FXT event files in an observation directory, and lists them.

`indir` Input directory Type f; mode a.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.2 `fxtcoord`

Adds or updates detector, focal-plane, and sky coordinates in an FXT event file.

`evtfile` Name of the input Event FITS file Type f; mode a.

`outfile` Name of the output Event FITS File or DEFAULT for standard name Type f; mode a.

`attfile` Name of the attitude Event FITS file Type f; mode a.

`teldef` Name of XRT teldef calibration file or CALDB Type f; mode h. Default: `CALDB`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.3 `fxttimecorr`

Corrects TM-mode event times and detector rows, removes low-channel row noise, and reconciles event and attitude GTIs.

`evtfile` Name of the input Event FITS file Type f; mode a.

`outfile` Name of the output Event FITS File or DEFAULT for standard name Type f; mode a. Default: `DEFAULT`.

`attfile` Name of the attitude Event FITS file Type f; mode a.

`teldef` Name of XRT teldef calibration file or CALDB Type f; mode h. Default: `CALDB`.

`ra` Source RA (degrees) Type r; mode a. Default: `-1`.

`dec` Source DEC (degrees) Type r; mode a. Default: `-91`.

`readout` total readout time in Counts Type r; mode h. Default: `236.8`.

`resolution` half position resolution (in pixel) Type i; mode h. Default: `3`.

`waitrd` waiting time (in pixel) Type i; mode h. Default: `6`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.4 fxtparticleidentify

Identifies and removes particle-induced tracks, showers, and instrumental-effect events in FXT data.

`evtfile` Name of the input Calibration Event FITS file Type f; mode a.

`outfile` Name of the output Event FITS File Type f; mode a.

`searchwidth` Search width of particle cluster Type i; mode h. Default: `3`.

`bkgnum` Background threshold Type i; mode h. Default: `5`.

`efrac` Mininum fraction of High Energy particles (>12keV) to all Type r; mode h. Default: `0.3`.

`xlength` Mininum length along X caused by instrumental effect Type i; mode h. Default: `11`.

`ylength` Mininum length along Y caused by instrumental effect Type i; mode h. Default: `11`.

`particlefile` Name of the output particle-induced Event FITS File Type f; mode h. Default: `NONE`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.5 fxtbadpix

Flags events from known bad pixels using the applicable FXT bad-pixel calibration.

`evtfile` Name of the input Event FITS file Type f; mode a.

`outfile` Name of the output Event FITS file or NONE to overwrite the input file Type f; mode a. Default: `NONE`.

`groundbpfile` Name of the input BADPIX file or CALDB or NONE for none Type f; mode h. Default: `CALDB`.

`boardbpfile` Name of the input on-board BADPIX file or CALDB or NONE for none Type f; mode h. Default: `CALDB`.

`userbpfile` Name of the user BADPIX file or NONE for none Type f; mode h. Default: `NONE`.

`outbpfile` Name of the output Bad Pixel file or DEFAULT to use standard name or NONE for none Type f; mode h. Default: `NONE`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.6 `fxthotpix`

Detects and flags transient hot or flickering pixels in an FXT event file.

`evtfile` Name of the input Event FITS file Type f; mode a.

`outfile` Name of the output Event FITS File or NONE to overwrite the input file Type f; mode a. Default: `NONE`.

`bpfile` Name of the input Bad Pixel file Type f; mode h. Default: `NONE`.

`outhpfile` Name of the output Hot Pixel file or NONE for none or DEFAULT for standard name Type f; mode h. Default: `NONE`.

`cellsize` Search cell size in pixels Type i; mode h. Default: `5`.

`binsize` Bin size (seconds) for count image generation Type i; mode h. Default: `2000`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.7 `fxtgtigen`

Builds FXT good-time intervals from housekeeping and configured screening criteria.

`mkffile` Name of the input MKF FITS file Type f; mode a.

`module` Name of the module of FXT (fxta/fxtb) Type s; mode a. Default: `fxta`.

`usrgtifile` Name of the user input GTI file (FITS format) or NONE for none Type f; mode h. Default: `NONE`.

`evtfile` Name of the event file (its GTI extension will be used) Type f; mode h. Default: `NONE`.

`outfile` Name of the output GTI Event FITS file or DEFAULT for standard name Type f; mode a. Default: `DEFAULT`.

`expr` Selection Expression for attitude and instrument HK or DEFAULT for standard (from CALDB) Type s; mode a. Default: `DEFAULT`.

`hkrangefile` Name of the input fits HKRANGE file or CALDB Type s; mode h. Default: `CALDB`.

`prefr` Pre-Time Interval factor [0,1] Type r; mode h. Default: `0.5`.

`postfr` Post-Time Interval factor [0,1] Type r; mode h. Default: `0.5`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.8 `fxtgticorr`

Refines good-time intervals using counts per CCD frame and writes a GTI FITS file.

`evtfile` Name of the event file Type f; mode a.

`gtifile` Name of the GTI Event FITS file Type f; mode a.

`outfile` Name of the output GTI Event FITS file Type f; mode a.

`maxnumber` Maximum counts/frame value for clean Type r; mode a. Default: `0`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.9 `fxtgrade`

Reconstructs FXT event patterns and assigns event grades from split-pixel signals.

`evtfile` Name of the input Calibration Event FITS file Type f; mode a.

`outfile` Name of the output Event FITS File or DEFAULT for standard name Type f; mode a. Default: `DEFAULT`.

`pithresh` Use event pixels with PI greater than or equal to the split threshold Type i; mode h. Default: `0`.

`covfile` Name of the input E2PI configure file or CALDB (default=CALDB) Type f; mode h. Default: `CALDB`.

`clobber` Overwrite an existing output file (yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.10 `fxtootest`

Evaluates FXT out-of-time events and writes the corresponding event flags.

`evtfile` Name of the input Event FITS file Type f; mode a.

`outfile` Name of the output Event FITS File or DEFAULT for standard name Type f; mode a. Default: `DEFAULT`.

`attfile` Name of the attitude Event FITS file Type f; mode a.

`seed` Input seed for random number generator Type i; mode h. Default: `0`.

`teldef` Name of teldef calibration file or CALDB Type f; mode h. Default: `CALDB`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`history` Write HISTORY keywords in output file?(yes/no) Type b; mode h. Default: `yes`.

`chatter` Chatter Level Type i; mode h. Default: `1`.

B.1.11 `fxtbary`

Applies solar-system barycentric time correction to an FXT event file.

`evtfile` Name of the input Event FITS file Type f; mode a.

`outfile` Name of the output Event FITS file Type f; mode a.

`orbitfile` Name of the input orbit file Type f; mode a.

`ephemfile` The JPL Ephemeris filename Type f; mode a.

`ra` Source RA (degrees) Type r; mode a. Default: `-1`.

`dec` Source DEC (degrees) Type r; mode a. Default: `-91`.

B.1.12 `fxtchain`

Runs the standard FXT observation reduction chain from raw inputs to calibrated products.

No public system parameter file is supplied for this command.

B.2 Official v2.0 workflow and calibration tasks

B.2.1 `fxtarfgen`

The task attempts to update the input PHA in place: the SPECTRUM header receives an ANCRFILE reference to the generated ARF. If the PHA is read-only, a warning is issued and ARF generation continues without changing the PHA.

`specfile` Input PHA is updated with ANCRFILE; writable input required Type s; mode q.

`expfile` The exposure map file Type s; mode h.

`outfile` The output arf file Type s; mode q.

`aspcor` Apply aspect weighting (0/1; merged inputs force 1) Type i; mode h. Default: `0`.

`extend` the source is extend? yes:1,no:0 Type i; mode h. Default: `0`.

`srcx` Source x in original 1-based physical SKY; paired with srcy Type r; mode h. Default: `-1`.

`srcy` Source y in original 1-based physical SKY; paired with srcx Type r; mode h. Default: `-1`.

`psfcor` Point-source PSF-weighted region/exposure correction; disabled for extend=1 Type i; mode h. Default: `1`.

`psfmodel` PSF correction model: dual-beta (default) or legacy Type s; mode h. Default: `dual-beta`.

`evtfile` The cleaned event file when aspcor=1 Type s; mode h.

`mkffile` The MKF attitude file when aspcor=1 Type s; mode h.

`weight` Attitude weighting mode: aspect or event Type s; mode h. Default: `aspect`.

`thetabin` Theta histogram bin width in arcmin; <=0 means no binning Type r; mode h. Default: `0.05`.

`regfile` Optional DS9/CIAO-style source region file Type s; mode h.

`clobber` Overwrite existing output file? (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`arfcorr` Time-dependent ARF factor: auto, required, or none Type s; mode h. Default: `auto`.

`respctx` Merged response context JSON; defaults to FITS RESPCTX Type s; mode h.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

`srcmodel` Extended-source brightness model: uniform, gaussian, beta, or image Type s; mode h. Default: `uniform`.

`srcsigma` Gaussian sigma in arcsec (FWHM/2.35482); positive and finite Type r; mode h. Default: `0.0`.

`srcrc` Beta core radius rc in arcsec (beta=2/3); positive and finite Type r; mode h. Default: `0.0`.

`srccut` Cutoff multiple of sigma or rc (not arcsec); positive and finite Type r; mode h. Default: `5.0`.

`srcmap` Relative source flux-per-pixel image with celestial WCS for srcmodel=image Type s; mode h.

`resp_pos` Point-source response position: region, centroid, or max Type s; mode h. Default: `region`.

B.2.2 `fxtarfmerge`

fxtarfmerge combines compatible EP/FXT ARF files by applying one scalar weight to the SPECRESP column of each input file. The default weightmode=exposure reads EXPOSURE from the matching PHA files in the same order as the ARFs and normalizes those values before merging. When the input exposures differ, the command records a warning and the exposure-weighted result is intentional. Set weightmode=equal explicitly when equal weighting is scientifically intended; that mode does not require specfiles. The input ARFs must share the same physical ENERG_LO and ENERG_HI grid. Energy units are converted to keV and effective-area units to cm**2; files without these column units use those same defaults for compatibility. Non-finite, reversed, or unordered grids and effective areas are rejected. The merged output is written as an OGIP-style SPECRESP table with checksums through an atomic output transaction.

`arffiles` Input ARF files, comma-separated or @listfile Type s; mode q.

`outfile` Output merged ARF file Type s; mode q.

`weightmode` Weight mode: exposure (default), equal, coverage, or weights Type s; mode h. Default: `exposure`.

`weights` Weights, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`specfiles` Input PHA files in ARF order (required for exposure or coverage weights) Type s; mode h. Default: `NONE`.

`clobber` Overwrite existing output file (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.3 `fxtbkggen`

For normmethod=imghe, first choose a source-free monitor region from the cleaned IMG events and extract an ungrouped, unsubtracted PI COUNTS PHA. Supply that spectrum with imgspecfile. Prefer a large valid region within DETX=3:382, DETY=3:382, excluding detected sources and bad/hot pixels or columns. The monitor region need not be the scientific source-extraction region. Do not select the lowest-count patch in the high-energy image. Ensure residual source/sky counts are negligible in the chosen band, including hard-source wings or diffuse emission. The PHA area metadata must describe the region actually retained after masks and detector-edge exclusions.

`infile` input the spectral file extracted from the cleaned frame store area events Type s; mode q.

`outfile` output file name of the instrumental background spectrum Type s; mode q.

`clobber` Overwrite existing output file (yes/no) Type b; mode h. Default: `yes`.

`dateobs` Missing DATE-OBS fallback (YYYY-MM-DDThh:mm:ss); AUTO reads the spectrum Type s; mode h. Default: `AUTO`.

`dateend` Missing DATE-END fallback (YYYY-MM-DDThh:mm:ss); AUTO reads the spectrum Type s; mode h. Default: `AUTO`.

`grade` Missing grade selection fallback (0:12); AUTO reads the spectrum Type s; mode h. Default: `AUTO`.

`datamode` Missing DATAMODE fallback (FF); AUTO reads the spectrum Type s; mode h. Default: `AUTO`.

`normmethod` Particle normalization: fsa or imghe Type s; mode h. Default: `fsa`.

`imgspecfile` PHA from a source-free cleaned FF IMG region; hidden unless normmethod=imghe Type s; mode h. Default: `NONE`.

`hemin` IMGHE E2PI lower endpoint in keV; hidden unless normmethod=imghe Type r; mode h. Default: `9.0`.

`hemax` IMGHE E2PI upper endpoint in keV exclusive; limited by the input E2PI table; hidden unless normmethod=imghe Type r; mode h. Default: `10.0`.

`heareascal` Effective particle monitor/template area fraction or AUTO; hidden unless normmethod=imghe Type s; mode h. Default: `AUTO`.

`heregion` Existing DS9 IMG extraction region for provenance only; hidden unless normmethod=imghe Type s; mode h. Default: `NONE`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.4 `fxtbkgmap`

The command queries CALDB for the E2PI calibration appropriate to infile. It maps each energy endpoint through the E2PI rule ELOW <= energy < EHIGH, then integrates the PHA COUNTS for the explicit CHANNEL values from the lower endpoint PI through the channel immediately below the upper endpoint PI. Thus the requested energy interval is half-open: [emin, emax).

`infile` Input instrumental background spectrum from fxtbkggen Type s; mode q.

`expfile` Input non-vignetted exposure map Type s; mode q.

`outfile` Output instrumental background map Type s; mode q.

`emin` Lower energy endpoint in keV Type r; mode h. Default: `0.5`.

`emax` Upper energy endpoint in keV Type r; mode h. Default: `10.0`.

`bkgimgpixels` Reference detector area in pixels or AUTO from spectrum BACKSCAL Type s; mode h. Default: `AUTO`.

`clobber` Overwrite existing output file Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query and learn mode Type s; mode h. Default: `ql`.

B.2.5 `fxtcaldb-update`

Synchronize, inspect, and verify the public EP/FXT CALDB through anonymous HTTPS, without an IHEP account, access token, or interactive login.

`command` CALDB updater command: install, status, check, verify, or rollback Type s; mode q. Default: `install`.

`source` Anonymous HTTPS base URL containing CALDB releases Type s; mode h. Default: https://epfxt.ihep.ac.cn/downloads/caldb.

`ref` latest or stable CALDB release ID caldb-vX.Y-YYYYMMDD[-rN] Type s; mode h. Default: `latest`.

`install_root` Target CALDB_v2.0 installation root Type s; mode h.

`state_dir` Private updater state, staging, and backup directory Type s; mode h.

`validate` Validate checksums and FITS files before installation Type b; mode h. Default: `yes`.

`dry_run` Resolve and plan the installation without changing files Type b; mode h. Default: `no`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.6 `fxte2pi`

The task loads the default FXT E2PI calibration from CALDB and uses the same channel-bin convention as the legacy tool: an energy is assigned to the first channel whose ELOW/EHIGH interval contains that energy. Values outside the calibration range are reported as PI channel 0.

`energy` Input energy in keV; use comma-separated values for multiple energies Type s; mode q.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.7 `fxteefgen`

'fxteefgen' generates EEF (Encircled Energy Fraction) curves based on detector, filter, position, and energy parameters. The EEF curve represents the fraction of energy contained within a given radius from the center of a point source, which is essential for aperture photometry and understanding PSF characteristics.

`det` detector (fxta/fxtb) Type s; mode q.

`filter` filter (thin/medium/hole) Type s; mode q.

`rawX` the rawX position of eef center Type r; mode q.

`rawY` the rawY position of eef center: Type r; mode q.

`energy` the energy of eef Type r; mode q.

`outfile` the name of the output eef file Type s; mode q.

`clobber` Overwrite existing output file (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.8 `fxtevtmerge`

fxtevtmerge merges a list of FXT event files into one event file. It is also used as a stage helper by fxtobsmerge.

`evtfiles` Input event files, comma-separated or @listfile Type s; mode q.

`infile` Alias for evtfiles Type s; mode h. Default: `NONE`.

`outfile` Output merged event FITS file Type s; mode q.

`clobber` Overwrite existing output file Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.9 `fxtexpogen`

fxtexpogen generates exposure maps that account for bad pixels and columns, pointing variations, and telescope vignetting.

`evtfile` The input event file Type s; mode q.

`mkffile` The mkf file Type s; mode q.

`outfile` The output exposure map file Type s; mode q.

`energy` Average energy in keV; values outside CALDB vignetting nodes use the nearest endpoint Type r; mode h. Default: `1.5`.

`specweight` Use spectrum-weighted exposure map (0/1) Type i; mode h. Default: `0`.

`hires` Use high-resolution and/or enlarged-FOV sky projection (0/1) Type i; mode h. Default: `0`.

`hirescale` Sky-pixel resolution scale; pixel size is divided by hirescale when hires=1 Type r; mode h. Default: `1.0`.

`fovscale` Sky field-of-view scale; RA/Dec width is multiplied by fovscale when hires=1 Type r; mode h. Default: `1.0`.

`projmethod` High-resolution method: bilinear or polygon; noninteger hirescale uses polygon Type s; mode h. Default: `bilinear`.

`area_scale` Whether to apply scaling to the effective area (yes/no) Type b; mode h. Default: `no`.

`edgecovermask` mask edge occluded range for merging Type i; mode h. Default: `0`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`use_clustering` Use clustering to accelerate computation (yes/no) Type b; mode h. Default: `yes`.

`ra_threshold` RA/DEC angular distance threshold for clustering (arcsec) Type r; mode h. Default: `2.0`.

`pa_threshold` Roll angle difference threshold for clustering (deg) Type r; mode h. Default: `0.01`.

`specfile` The input spectrum file when specweight=1; use NONE to weight by event energies Type s; mode h.

`speccol` Energy column used when specweight=1 and specfile is empty Type s; mode h. Default: `ENG`.

`emin` Lower energy limit in keV; samples outside vignetting nodes use the nearest endpoint Type r; mode h. Default: `0.3`.

`emax` Upper energy limit in keV for spectrum weighting Type r; mode h. Default: `10.0`.

`badpixsize` RAW bad-pixel mask width: auto from recorded STATUS screening or explicit 3/5 Type s; mode h. Default: `auto`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`outmode` Compute/write exposure maps: both/vign/novign (case-insensitive) Type s; mode h. Default: `both`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.10 `fxtexpomerge`

fxtexpomerge accepts two input paths. If expfiles are supplied, it merges those existing exposure maps onto a common sky grid. If evtfiles and mkffiles are supplied instead, it first generates one exposure map per event input with fxtexpogen, then merges the generated maps. The per-observation maps remain in outdir; the final product is prefix_merged.fits, or prefix_<tile>_merged.fits for a tiled output.

`outdir` Output directory Type s; mode q.

`evtfiles` Input event files used to generate exposure maps before merging Type s; mode h. Default: `NONE`.

`mkffiles` Input MKF files used to generate exposure maps before merging Type s; mode h. Default: `NONE`.

`expfiles` Input exposure map files to merge Type s; mode h. Default: `NONE`.

`ra` Output center RA in degrees or auto Type s; mode h. Default: `auto`.

`dec` Output center Dec in degrees or auto Type s; mode h. Default: `auto`.

`width` Output sky width in degrees or auto Type s; mode h. Default: `auto`.

`height` Output sky height in degrees or auto Type s; mode h. Default: `auto`.

`pixsize` Output pixel size in arcsec or auto Type s; mode h. Default: `auto`.

`tile_size` Exposure tile size in pixels Type i; mode h. Default: `2048`.

`prefix` Output filename prefix Type s; mode h. Default: `expo`.

`energy` Mono energy in keV for generated exposure maps Type r; mode h. Default: `1.5`.

`expomaskfrac` Per-input fraction of center exposure; merged maps skip the mask; <=0 disables Type r; mode h. Default: `0.1`.

`clobber` Overwrite existing output files Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.11 `fxtimage`

fxtimage bins screened event files into images and provides image arithmetic, exposure correction, OoT subtraction, smoothing, and RGB displays. It accepts screened events or existing FITS images; event screening must be completed first, for example with xselect.

`mode` Operation mode: bin, rate, oot, smooth, rgb, add, sub, mul, or div Type s; mode h. Default: `bin`.

`evtfile` Input event FITS file for mode=bin Type s; mode h.

`imgfile` Input image FITS file for mode=rate, oot, smooth, add, sub, mul, or div Type s; mode h.

`imgfile2` Second input image FITS file for mode=add, sub, mul, or div Type s; mode h.

`expfile` Input exposure map for mode=rate Type s; mode h.

`ootfile` Input OoT image for mode=oot Type s; mode h.

`redfile` Input red-channel image for mode=rgb Type s; mode h.

`greenfile` Input green-channel image for mode=rgb Type s; mode h.

`bluefile` Input blue-channel image for mode=rgb Type s; mode h.

`outfile` Output file Type s; mode q.

`emin` Minimum energy in keV for mode=bin Type r; mode h.

`emax` Maximum energy in keV for mode=bin; inclusive for ENG or ENERGY Type r; mode h.

`xcol` Event X column for mode=bin Type s; mode h. Default: `X`.

`ycol` Event Y column for mode=bin Type s; mode h. Default: `Y`.

`energycol` ENG, ENERGY, or PI; auto selects in that order Type s; mode h. Default: `auto`.

`xmin` Minimum inclusive X bin edge for mode=bin Type r; mode h. Default: `0.5`.

`xmax` Maximum exclusive X bin edge for mode=bin Type r; mode h. Default: `600.5`.

`ymin` Minimum inclusive Y bin edge for mode=bin Type r; mode h. Default: `0.5`.

`ymax` Maximum exclusive Y bin edge for mode=bin Type r; mode h. Default: `600.5`.

`binsize` Image bin size for mode=bin; bounds must span an integer number of bins Type r; mode h. Default: `1.0`.

`ra` Output TAN center right ascension in degrees for mode=bin; pair with dec Type r; mode h.

`dec` Output TAN center declination in degrees for mode=bin; pair with ra Type r; mode h.

`pixsize` TAN projection scale in arcsec for mode=bin; inferred from event WCS when blank Type r; mode h.

`outcounts` Optional thresholded counts output for mode=rate Type s; mode h. Default: `NONE`.

`outexp` Optional thresholded exposure output for mode=rate Type s; mode h. Default: `NONE`.

`maskfile` Optional exposure mask output for mode=rate Type s; mode h. Default: `NONE`.

`expfrac` Exposure cut as fraction of max exposure for mode=rate Type r; mode h. Default: `0.15`.

`expmin` Absolute minimum exposure for mode=rate Type r; mode h.

`scale` Scale factor for mode=oot Type r; mode h. Default: `0.0023`.

`ascale` Scale factor applied to imgfile in generic arithmetic Type r; mode h. Default: `1.0`.

`bscale` Scale factor applied to imgfile2 in generic arithmetic Type r; mode h. Default: `1.0`.

`outunit` Output BUNIT policy: AUTO, NONE, or an explicit BUNIT label Type s; mode h. Default: `AUTO`.

`method` Smoothing method: gauss or box Type s; mode h. Default: `gauss`.

`sigma` Gaussian sigma for mode=smooth Type r; mode h. Default: `2.0`.

`boxsize` Boxcar size for mode=smooth Type i; mode h. Default: `3`.

`vmin` RGB lower display cut Type r; mode h.

`vmax` RGB upper display cut Type r; mode h.

`clobber` Overwrite existing output files (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

B.2.12 `fxtimgmerge`

The tool reads multiple FXT image and exposure-map pairs, builds a common sky WCS that covers all input pointings, reprojects each pair to that grid, and sums them into merged image and exposure products.

`imgfiles` Input image files, comma-separated or @listfile Type s; mode q.

`expfiles` Input exposure map files, comma-separated or @listfile Type s; mode q.

`outimg` Output merged image FITS file Type s; mode q.

`outexp` Output merged exposure FITS file Type s; mode q.

`ra` Output image center RA in degrees; -999 = auto from inputs Type r; mode h. Default: `-999.0`.

`dec` Output image center Dec in degrees; -999 = auto from inputs Type r; mode h. Default: `-999.0`.

`xpix` Output image X dimension in pixels; 0 = auto from inputs Type i; mode h. Default: `0`.

`ypix` Output image Y dimension in pixels; 0 = auto from inputs Type i; mode h. Default: `0`.

`pixsize` Output pixel size in arcsec; 0 = use input file pixel scale Type r; mode h. Default: `0.0`.

`padding` Extra padding pixels around merged sky footprint Type i; mode h. Default: `5`.

`interp` Compatibility setting; heterogeneous WCS uses exact area-overlap reprojection Type s; mode h. Default: `bilinear`.

`outrateimg` Optional output merged count-rate FITS file; use NONE to skip Type s; mode h. Default: `NONE`.

`expmin` Minimum exposure threshold in input exposure maps; pixels below it are ignored Type r; mode h. Default: `1.0`.

`clobber` Overwrite existing output files (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.13 `fxtlccorr`

fxtlccorr requires aligned source and background bins. It converts background RATE and ERROR to the source RATE unit, calculates RATE_out = RATE_src - alpha * RATE_bkg, and propagates errors in quadrature. RATE units must be declared and compatible. alpha is a rate multiplier: with AUTO, the tool uses a unique rate-domain manifest match or BACKSCAL_src / BACKSCAL_bkg. Each RATE already includes its own exposure normalization, so EXPOSURE is not multiplied again.

`srcfile` Input source-region light curve Type s; mode q.

`bkgfile` Input background-region light curve Type s; mode q.

`outfile` Output background-corrected light curve Type s; mode q.

`alpha` Background rate multiplier; AUTO uses manifest or BACKSCAL ratio Type s; mode h. Default: `AUTO`.

`errmethod` Error method: auto uses trusted Poisson counts; input uses ERROR; gehrels requires trusted counts Type s; mode h. Default: `auto`.

`scaling_manifest` Manifest with matching source/background pair and rate_alpha Type s; mode h. Default: `NONE`.

`clobber` Overwrite an existing output file (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.14 `fxtlcmath`

fxtlcmath requires identical row counts, TIME, TIMEDEL, and timing-reference metadata. It does not interpolate or rebin. Define A=scale1*RATE1 and B=scale2*RATE2; operation selects A+B, A-B, A*B, or A/B. Errors are propagated directly from the two independent input errors.

`infile1` First input RATE light curve Type s; mode q.

`infile2` Second input RATE light curve Type s; mode q.

`outfile` Output arithmetic light curve Type s; mode q.

`operation` Arithmetic operation: add—sub—mul—div Type s; mode q.

`scale1` Finite scale applied to infile1 RATE Type r; mode h. Default: `1.0`.

`scale2` Finite scale applied to infile2 RATE Type r; mode h. Default: `1.0`.

`clobber` Overwrite an existing output file (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.15 `fxtobsmerge`

combine=none retains the default detector/filter groups. With combine=images, requested image families are also summed within each readout mode under merged/ff/, merged/pw/, or merged/tm/. With combine=spectra response=yes, each source's per-observation PHA, background, ARF and RMF bundles are combined under merged/MODE/srcproducts/ using fxtspecmerge checkfilter=no. Every input spectrum must still match its own responses, channel grid and selections. Cross-group physical/image regions require identical event WCS; use FK5 or ICRS regions when grids differ. combine=all combines only the requested image and spectral families and does not enable response extraction automatically.

`inputmode` Input source mode: obs, events, images, mixed, or slew Type s; mode h. Default: `obs`.

`indirlist` Input observation directories, comma-separated or @listfile; use NONE for manual file lists Type s; mode h. Default: `NONE`.

`evtfiles` Input event files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`mkffiles` Input MKF attitude files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`attfiles` Input attitude files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`orbitfiles` Input orbit files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`expr` GTI screening expression for raw event/FSA preparation; DEFAULT uses the selected CALDB HK expression Type s; mode h. Default: `DEFAULT`.

`imgfiles` Input image files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`expfiles` Input exposure-map files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`expofiles` Alias for expfiles; input exposure-map files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`autoexp` Generate exposure maps from events and MKF files (yes/no) Type b; mode h. Default: `yes`.

`expenergy` Mono energy in keV used by fxtexpogen when autoexp=yes Type r; mode h. Default: `1.5`.

`emin` Minimum image energy in keV; blank leaves this end unbounded Type r; mode h.

`emax` Maximum image energy in keV; ENG/ENERGY inclusive and PI uses CALDB E2PI endpoints Type r; mode h.

`outdir` Output run directory; final products are written under outdir/merged Type s; mode q.

`mode` Merge mode: auto, small, large, or slew Type s; mode h. Default: `auto`.

`ra` Output center RA in degrees, or auto to infer from input coverage Type s; mode h. Default: `auto`.

`dec` Output center Dec in degrees, or auto to infer from input coverage Type s; mode h. Default: `auto`.

`width` Output sky width in degrees, or auto to cover input footprint Type s; mode h. Default: `auto`.

`height` Output sky height in degrees, or auto to cover input footprint Type s; mode h. Default: `auto`.

`pixsize` Output pixel size in arcsec, or auto to infer from FITS WCS Type s; mode h. Default: `auto`.

`expomaskfrac` Per-input fraction of center exposure; merged maps skip the mask; <=0 disables Type r; mode h. Default: `0.1`.

`bkgimgpixels` AUTO derives the image-area denominator from particle-spectrum BACKSCAL; a positive number overrides it Type s; mode h. Default: `AUTO`.

`bkgemin` Particle-background minimum energy in keV (inclusive) Type r; mode h. Default: `0.5`.

`bkgemax` Particle-background maximum energy in keV (exclusive) Type r; mode h. Default: `10.0`.

`normmethod` Merged particle-spectrum normalization: fsa or imghe Type s; mode h. Default: `fsa`.

`heregion` Source-free FK5/ICRS IMG monitor region for merged high-energy normalization Type s; mode h. Default: `NONE`.

`hemin` IMGHE E2PI normalization lower endpoint in keV Type r; mode h. Default: `9.0`.

`hemax` IMGHE E2PI normalization upper endpoint in keV (exclusive); limited by the input E2PI table Type r; mode h. Default: `10.0`.

`products` Requested product families: all or comma-separated reproject,evt,img, expo,bkg,rate,response Type s; mode h. Default: `all`.

`combine` Cross-group aggregation: none, images, spectra, or all; default none Type s; mode h. Default: `none`.

`datamodes` Datamodes to include in directory or grouped file discovery: all or comma-separated FF,PW,TM,DF,DT,DS; DF selects the FF group Type s; mode h. Default: `all`.

`forcechain` Rerun the installed fxtchain to exitstage=2 from raw UF inputs (yes/no); requires inputmode=obs with indirlist Type b; mode h. Default: `no`.

`resume` Reuse completed manifest stages when hashes and parameters match (yes/no) Type b; mode h. Default: `no`.

`dryrun` Write a separate planned-output JSON without creating science products (yes/no) Type b; mode h. Default: `no`.

`manifest` Manifest path, or auto to write fxtobsmerge_manifest.json in outdir Type s; mode h. Default: `auto`.

`mergepolicy` Compatibility policy for stage gates: strict, warn, or force Type s; mode h. Default: `strict`.

`caldb_policy` CALDB provenance policy for resume: strict, warn, or ignore Type s; mode h. Default: `strict`.

`keepintermediate` Keep stage intermediate products (yes/no) Type b; mode h. Default: `yes`.

`validate` Run built-in product validation and record results (yes/no) Type b; mode h. Default: `yes`.

`response` Run fxtsrcproducts source-product orchestration after event merge (yes/no) Type b; mode h. Default: `no`.

`sources` JSON source list for fxtsrcproducts; required when response=yes and srcregion=NONE Type s; mode h.

`spectral_mode` Spectral response strategy: stack_then_merge only (per-observation responses; compatible stacking or joint fit) Type s; mode h. Default: `stack_then_merge`.

`srcregion` Source region for optional spectral stack products Type s; mode h. Default: `NONE`.

`bkgregion` Background region for optional spectral stack products Type s; mode h. Default: `NONE`.

`psfcor` PSF correction switch for optional point-source ARF generation Type i; mode h. Default: `1`.

`clobber` Overwrite existing output files (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

B.2.16 `fxtpical`

fxtpical assigns calibrated event energies and PI channels to FF, PW, and TM data, correcting for position-dependent gain, Charge Transfer Inefficiency (CTI), and changes in gain with time.

`evtfile` The input evt file Type s; mode q.

`outfile` The output evt file Type s; mode q.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`gain_extrapolation` Gain/CTI extrapolation outside CALDB date nodes (caldb/hold/linear) Type s; mode h. Default: `caldb`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.17 `fxtplotgrade`

Plot event grade fractions, selecting the reference from PICALVER or retained processing history, and otherwise using T18V2; PNG output is saved only when an output filename is supplied.

`evtfile` The input event file Type s; mode q.

`outfile` Optional output PNG file (NONE disables saving; specify each run) Type s; mode h. Default: `NONE`.

`plotbins` Maximum number of equal-count energy bins Type i; mode h. Default: `100`.

`show` Display the plot window?(yes/no) Type b; mode h. Default: `yes`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.18 `fxtpsfgen`

'fxtpsfgen' generates PSF (Point Spread Function) files based on detector, filter, position, and energy parameters. The PSF characterizes how a point source of light spreads out in the detector due to the telescope optics and detector response.

`det` detector (fxta/fxtb) Type s; mode q.

`filter` filter (thin/medium/hole) Type s; mode q.

`rawX` the rawX position of psf center Type r; mode q.

`rawY` the rawY position of psf center Type r; mode q.

`energy` the energy of psf Type r; mode q.

`outfile` the name of the output psf image file Type s; mode q.

`clobber` Overwrite existing output file (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

`infile` Input image/FITS file for SKY-coordinate PSF generation Type s; mode h. Default: `NONE`.

`ra` Sky right ascension in degrees; provide with dec Type r; mode h.

`dec` Sky declination in degrees; provide with ra Type r; mode h.

`psfcontext` Frozen response context JSON for COMPOSITE PSF generation Type s; mode h. Default: `NONE`.

`imageid` COMPOSITE image identifier; valid only with psfcontext Type s; mode h. Default: `NONE`.

B.2.19 `fxtpsfmap`

'fxtpsfmap' generates a PSF (Point Spread Function) size map based on an input exposure map. The tool calculates the PSF radius at each position in the exposure map, accounting for off-axis angle variations.

`expmap` Input exposure map file Type s; mode q.

`outfile` Output PSF map file Type s; mode q.

`energy` Effective energy in keV Type r; mode q.

`eef` Enclosed Energy Fraction (0 < eef <= 1) Type r; mode q.

`filt` Filter ID (0:open, 1:thin, 2:medium, 3:hole). If not specified, read from input file FILTER keyword Type i; mode h. Default: `-1`.

`clobber` Overwrite existing output file (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.20 `fxtpsfmdcopy`

fxtpsfmdcopy is the upgraded replacement for the deprecated fxtpsf command. Use fxtpsfmdcopy in new scripts; the fxtpsf compatibility name will be removed in a future release.

`outdir` Output directory for copied PSF metadata files Type s; mode h. Default: ..

`psffile` Input PSF calibration file or auto Type s; mode h. Default: `auto`.

`eeffile` Input EEF calibration file or auto Type s; mode h. Default: `auto`.

`manifest` Output JSON manifest path; NONE skips output Type s; mode h. Default: `NONE`.

`clobber` Overwrite existing output files Type b; mode h. Default: `no`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

`model` PSF model family: 2beta or legacy Type s; mode h. Default: `2beta`.

`det` Detector selection: all/fxta/fxtb Type s; mode h. Default: `all`.

B.2.21 `fxtregmaskfill`

fxtregmaskfill creates source or exclusion masks from image-coordinate or WCS-aware sky-coordinate regions and writes masks on the same pixel grid as the input image. It accepts DS9 region files, fxtsrcdet-style FITS source tables, or JSON source lists.

`imgfile` Input image FITS file Type s; mode q.

`srcfile` Input DS9 image region file, FITS source table, or sources.json Type s; mode q.

`maskout` Output source mask FITS; 1=source/hole, 0=keep Type s; mode q.

`validmaskout` Output valid analysis mask FITS; 1=valid, 0=source/outside Type s; mode q.

`basemask` Optional input exposure/FOV mask; nonzero pixels are eligible Type s; mode h. Default: `NONE`.

`maskscale` Multiplicative scale applied to source-region radii Type r; mode h. Default: `1.0`.

`fill` Write filled image outfile (yes/no) Type b; mode h. Default: `no`.

`outfile` Filled output image when fill=yes Type s; mode h. Default: `NONE`.

`method` Fill method: constant, nan, dist, poisson, global, poly, or bilint Type s; mode h. Default: `constant`.

`fillvalue` Constant fill value when method=constant Type r; mode h. Default: `0.0`.

`bkginner` Inner local-background radius scale for dist/poisson/poly Type r; mode h. Default: `1.0`.

`bkgouter` Outer local-background radius scale for dist/poisson/poly Type r; mode h. Default: `2.5`.

`randseed` Random seed for dist/poisson/global; 0 uses non-deterministic seed Type i; mode h. Default: `0`.

`manifest` Output JSON manifest path, or NONE to skip Type s; mode h. Default: `fxtregmaskfill_manifest.json`.

`clobber` Overwrite existing output files (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.22 `fxtrmfgen`

fxtrmfgen generates an OGIP-style response matrix file for EP-FXT spectral analysis. The response matrix maps photon energy to detector channel and is used by XSPEC together with an ARF.

`specfile` Input PHA is updated with RESPFILE; writable input required Type s; mode q.

`outfile` The output response file Type s; mode q.

`rmfmode` RMF source mode (quarterly/nodes) Type s; mode h. Default: `quarterly`.

`clobber` Overwrite existing output file (if outfile!=NONE)?(yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.23 `fxtrmfmerge`

fxtrmfmerge combines compatible EP/FXT RMF files using scalar weights, or energy-dependent ARF-times-exposure weights in weightmode=arf_exposure. In inputmode=spec, the task uses each PHA spectrum observation time and detector information to resolve the corresponding CALDB RMF with the same rules used by fxtrmfgen. In inputmode=rmf, the task merges explicitly supplied RMF files. MATRIX and EBOUNDS energy columns are normalized to keV (missing units default to keV). The input channel grid, finite non-negative MATRIX values, sparse group spans, and DETCHANS consistency are validated before an OGIP-classified, checksummed RMF is published atomically. CALDB-selected MATRIX extension numbers are retained.

`inputmode` Input mode: spec or rmf Type s; mode h. Default: `spec`.

`specfiles` Input spectrum files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`rmffiles` Input RMF files, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`outfile` Output merged RMF file Type s; mode q.

`weightmode` Weight mode: exposure, coverage, equal, weights, or arf_exposure Type s; mode h. Default: `exposure`.

`weights` Weights, comma-separated or @listfile Type s; mode h. Default: `NONE`.

`arffiles` ARF files used by weightmode=arf_exposure Type s; mode h. Default: `NONE`.

`exposures` Exposure weights used by weightmode=arf_exposure Type s; mode h. Default: `NONE`.

`allow_scalar` Allow diagnostic scalar averaging of distinct RMFs (not for science fitting) Type b; mode h. Default: `no`.

`caldb_root` Optional CALDB root override Type s; mode h. Default: `NONE`.

`caldb_config` Optional CALDB config override Type s; mode h. Default: `NONE`.

`clobber` Overwrite existing output file (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.24 `fxtspecmerge`

fxtspecmerge combines compatible OGIP PHA spectra on the same CHANNEL grid. It is intended for source spectra, FSA count spectra, and the floating-point background spectra produced by fxtbkggen. Input files are never changed. The output retains floating-point COUNTS; explicit STAT_ERR columns are propagated using independent-input variance addition. Shared template, calibration, and other cross-channel covariance is not represented by the marginal STAT_ERR column and is recorded in HISTORY rather than claimed to be removed. The OGIP POISSERR=T, STAT_ERR=0 keyword-only placeholder is accepted; a true STAT_ERR column or nonzero error keyword with POISSERR=T is rejected.

`infiles` Input PHA files, space/comma-separated or @listfile Type s; mode q.

`sumtype` Spectrum combination mode: TIME or AREA Type s; mode h. Default: `TIME`.

`outfile` Merged PHA output; prefix with ! to overwrite Type s; mode q.

`backfiles` Optional ordered background PHA list, NONE, or % for PHA BACKFILE Type s; mode h. Default: `NONE`.

`arffiles` Optional ordered ARF list, NONE, or % for PHA ANCRFILE Type s; mode h. Default: `NONE`.

`rmffiles` Optional ordered RMF/RSP list, NONE, or % for PHA RESPFILE Type s; mode h. Default: `NONE`.

`clobber` Overwrite existing output files (yes/no) Type b; mode h. Default: `no`.

`bkgmethod` Background combination: AUTO, TIME, ASCA, COUNTS, or WEIGHTED Type s; mode h. Default: `ASCA`.

`checkfilter` Require matching FILTER across input spectra Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`mode` Query and learn mode Type s; mode h. Default: `ql`.

B.2.25 `fxtsrcdet`

detmethod selects the source-detection method. The default wavdetect value runs the multiscale wavelet path; mldetect runs likelihood fitting. The aliases ciao, erosita, and esass select the corresponding methods.

Usage

```
fxtsrcdet imgfile=<image.fits> expofile=<expo.fits|NONE> outfile=<sources.fits> \
          regfile=<sources.reg> sigfile=<significance.fits> [options]
```

Usage

```
fxtsrcdet detmethod=mldetect imgfile=<image.fits|A.fits,B.fits> \
          expofile=<expo.fits|A_exp.fits,B_exp.fits|NONE> \
          outfile=<likelihood_sources.fits> [likelihood options]
```

`detmethod` Detection backend: wavdetect or mldetect; aliases ciao erosita and esass Type s; mode h. Default: `wavdetect`.

`imgfile` Input counts image FITS file Type s; mode h.

`expofile` Input exposure map FITS file or NONE Type s; mode h. Default: `NONE`.

`infile` CIAO-style input counts image FITS file Type f; mode a.

`expfile` CIAO-style input exposure map FITS file; blank means none Type f; mode h.

`outfile` Output source-table FITS file Type f; mode a.

`regfile` Output DS9 region file Type f; mode h.

`sigfile` Output significance-map FITS file Type s; mode h. Default: `..`

`scellfile` Output source-cell label image FITS file; blank means none Type f; mode a.

`imagefile` Output reconstructed source image FITS file; blank means none Type f; mode a.

`defnbkgfile` Output normalized background FITS file; blank means none Type f; mode a.

`psffile` Optional PSF-size map from fxtpsfmap; blank means none. Main counts/rates are not EEF-corrected. Type f; mode a.

`extoutfile` Optional extended-source table FITS file or NONE Type s; mode h. Default: `NONE`.

`extregfile` Optional extended-source DS9 region file or NONE Type s; mode h. Default: `NONE`.

`bkgfile` Optional background-map FITS file or NONE Type s; mode h. Default: `NONE`.

`bkginput` CIAO-style input background map; blank means none Type f; mode h.

`maskfile` Optional detection-mask FITS file or NONE Type s; mode h. Default: `NONE`.

`boxlist` AUTO or FITS candidates; X/Y zero-based; wavdetect SRCDET X/Y and X_IMA/Y_IMA one-based; priority X/Y then X_IMA/Y_IMA then RA/DEC Type s; mode h. Default: `AUTO`.

`autopipeline` AUTO pipeline: esass or fxtsrcdet compatibility; dual images require esass Type s; mode h. Default: `esass`.

`boxsize` ERBOX half-size in image pixels Type i; mode h. Default: `4`.

`boxnruns` ERBOX compression runs Type i; mode h. Default: `3`.

`boxlikemin` Minimum ERBOX detection likelihood Type r; mode h. Default: `6.0`.

`boxmaskfrac` Minimum valid ERBOX aperture fraction Type r; mode h. Default: `0.55`.

`bkgscut` ERBACKMAP PSF surface cut Type r; mode h. Default: `1e-4`.

`bkgmlmin` Minimum likelihood for ERBACKMAP source masking Type r; mode h. Default: `6.0`.

`bkgmaxcut` Minimum retained detector fraction Type r; mode h. Default: `0.5`.

`bkgfitmethod` ERBACKMAP fit method: smooth or spline Type s; mode h. Default: `smooth`.

`bkgsnr` ERBACKMAP adaptive-smoothing target SNR Type r; mode h. Default: `40.0`.

`bkgnfitrun` Spline clipping iterations Type i; mode h. Default: `3`.

`bkgexcesssigma` Spline residual clipping threshold in sigma Type r; mode h. Default: `1000.0`.

`bkgnsplinenodes` Spline spatial bins per dimension Type i; mode h. Default: `36`.

`bkgdegree` Spline degree Type i; mode h. Default: `2`.

`bkgsmoothflag` Use a smoothing spline Type b; mode h. Default: `yes`.

`bkgsmooth` Initial ERBACKMAP smoothing sigma or spline smoothing value Type r; mode h. Default: `30.0`.

`localboxoutfile` Optional single local SRCLIST path; requires eSASS boxlist=AUTO bkgfile=AUTO Type s; mode h. Default: `NONE`.

`boxoutfile` Optional single map SRCLIST path or NONE Type s; mode h. Default: `NONE`.

`diagnosticfile` Optional single FXT diagnostics path for autopipeline=esass Type s; mode h. Default: `NONE`.

`bkgoutfile` Optional background FITS: A then B paths or . for module names Type s; mode h. Default: `NONE`.

`cheesemask` Optional cheese-mask FITS: A then B paths or . for module names Type s; mode h. Default: `NONE`.

`srcmap` Optional fitted source model FITS: A then B paths or . for module names Type s; mode h. Default: `NONE`.

`residfile` Optional residual FITS: A then B paths or . for module names Type s; mode h. Default: `NONE`.

`psfmodel` PSF model: CALDB or GAUSS or COMPOSITE (mldetect point sources only) Type s; mode h. Default: `CALDB`.

`psfcontext` Frozen imaging-response context JSON for COMPOSITE Type s; mode h. Default: `NONE`.

`energy` PSF energy in eV Type r; mode h. Default: `1000.0`.

`detnam` Detector name or AUTO Type s; mode h. Default: `AUTO`.

`filter` Filter name or AUTO Type s; mode h. Default: `AUTO`.

`psfinterp` FXT CALDB PSF interpolation: nearest or linear with boundary clamping Type s; mode h. Default: `nearest`.

`psfsigma` Gaussian PSF sigma in image pixels Type r; mode h. Default: `2.0`.

`scales` Detection scales in image pixels; CIAO whitespace comma colon and semicolon separators are accepted Type s; mode a. Default: `2.0 4.0`.

`sigthresh` Detection significance threshold Type r; mode h. Default: `1e-06`.

`expmin` Optional absolute exposure floor; 0 disables this FXT-only filter Type r; mode h. Default: `0.0`.

`corrtype` Detection kernel type: mexhat; gauss is incompatible with required significance calibration Type s; mode h. Default: `mexhat`.

`bkgradius` Background annulus inner-radius factor Type r; mode h. Default: `2.5`.

`falsesrc` Allowed false sources per scale; <=0 disables adaptive thresholding Type r; mode h. Default: `-1.0`.

`bkgsigthresh` Background-cleansing threshold; sigma if >1 else probability Type r; mode h. Default: `0.001`.

`sigmamode` Sigma calculation mode: ciao keeps wavdetect-aligned behavior; effarea applies the experimental effective-area correction Type s; mode h. Default: `ciao`.

`expthresh` CIAO-style relative exposure threshold after exposure-map normalization Type r; mode h. Default: `0.1`.

`ellsigma` Ellipse scaling for fxtsrcdet source-pixel R/ROTANG output Type r; mode h. Default: `3.0`.

`maxiter` Maximum source-cleansing iterations (positive integer) Type i; mode h. Default: `2`.

`iterstop` Fractional stopping threshold for cleansing-mask changes Type r; mode h. Default: `0.0001`.

`likemin` Minimum per-source detection likelihood Type r; mode h. Default: `8.0`.

`extlikemin` Minimum extent likelihood Type r; mode h. Default: `6.0`.

`fitposition` Fit subpixel source positions after boxlist coordinates are converted to internal zero-based pixels; no keeps converted positions fixed and fits rates only Type b; mode h. Default: `yes`.

`fitextent` Test extent; no fixes external EXT and disables twostage Type b; mode h. Default: `yes`.

`extentmodel` Extent model: beta or gauss/gaussian Type s; mode h. Default: `beta`.

`extmin` Minimum tested extent in image pixels Type r; mode h. Default: `1.5`.

`extmax` Maximum tested extent in image pixels Type r; mode h. Default: `40.0`.

`nmaxfit` Maximum simultaneous sources per fit cluster Type i; mode h. Default: `3`.

`multrad` Neighbour control; POINTING preserves upstream scutrad path Type r; mode h. Default: `15.0`.

`cutrad` Fit radius; below 1 upstream passes multrad to EEF radius Type r; mode h. Default: `15.0`.

`fitbackend` Fit backend: mldetect likelihood core (based on eROSITA/eSASS ermldet 1.35); GPL-3.0-or-later Python core or legacy regression; dual images require mldetect Type s; mode h. Default: `mldetect`.

`fitmaxiter` Maximum mldetect optimizer iterations Type i; mode h. Default: `50`.

`nmulsou` Maximum split components per input source Type i; mode h. Default: `2`.

`twostage` Second-stage split after extent trigger; disabled when fitextent=no Type b; mode h. Default: `yes`.

`splitthresh` Apply a source-splitting threshold; requires nmulsou >= 2 Type b; mode h. Default: `no`.

`splitthreshval` Source-splitting threshold value Type r; mode h. Default: `30.0`.

`splitthreshcol` Threshold column: explicit LIKE RATE or SCTS; no score fallback Type s; mode h. Default: `LIKE`.

`psfthreshold` Minimum in-mask PSF fraction Type r; mode h. Default: `0.15`.

`maskthreshold` Minimum valid fraction for FXT fit and wide source-map regions Type r; mode h. Default: `0.0`.

`extlike_slope` Extent-likelihood intercept and slope Type s; mode h. Default: `0,0`.

`srccoverage` Minimum valid-pixel fraction in the source support Type r; mode h. Default: `0.8`.

`bkgcoverage` Minimum valid-pixel fraction in the background support Type r; mode h. Default: `0.6`.

`residualsigthresh` Residual-pass detection threshold; sigma if >1 else one-sided probability Type r; mode h. Default: `3.5`.

`residualfalsesrc` Residual-pass false-source rate per scale; <=0 disables adaptive thresholding Type r; mode h. Default: `-1.0`.

`residualscalemax` Maximum scale used by the residual weak-source pass Type r; mode h. Default: `3.0`.

`fxtrecovery` Enable fxtsrcdet-only weak-source recovery passes outside CIAO wavdetect semantics (yes/no) Type b; mode h. Default: `no`.

`srcclass` Primary source-table class: point, extended, or all Type s; mode h. Default: `all`.

`extentcheck` Enable srcextent-style extended-source post-check (yes/no) Type b; mode h. Default: `no`.

`extentthresh` Minimum extent significance for confirmed extended sources Type r; mode h. Default: `5.0`.

`extentmincounts` Minimum source counts for extent classification Type i; mode h. Default: `15`.

`extentpsfblur` Additional PSF sigma blur in image pixels for extent classification Type r; mode h. Default: `0.0`.

`exptime` Source livetime for detection/background normalization; rates use count-weighted expfile seconds when supplied Type r; mode h. Default: `0.0`.

`bkgtime` Livetime in seconds for a predetermined background map; 0 means match exposure Type r; mode h. Default: `0.0`.

`outputinfix` Optional output filename infix Type f; mode h.

`sigcalfile` Required significance calibration file; AUTO and CALDB require FXT CALDB SRCDETSIGCAL, CIAO explicitly selects wtsimresult.fits, NONE is rejected Type f; mode h. Default: `AUTO`.

`interdir` Accepted for CIAO wavdetect compatibility; no intermediate stack is written Type f; mode h. Default: `..`

`bkgerrinput` CIAO-style switch; yes reads errors from DataModel block 2 (normally FITS extension 1) Type b; mode h. Default: `no`.

`log` Write fxtsrcdet.log as well as stdout (yes/no) Type b; mode h. Default: `no`.

`verbose` Logging verbosity: 0=warning/error; 1=normal info; 2=debug Type i; mode h. Default: `1`.

`clobber` Overwrite existing outputs (yes/no) Type b; mode h. Default: `no`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.26 `fxtsrcproducts`

fxtsrcproducts creates spectra, light curves, images, and response products from merged FXT events and their response information. Use a TOML configuration to specify the workflow, or the parameter interface when config=NONE or is omitted. Compatible completed stages can be reused with resume=yes.

`evtfile` Context-aware merged event file used as the user-facing container Type s; mode q.

`rspctx` Response context JSON produced from the context-aware event merge Type s; mode q.

`sources` JSON source specification with srcregion and optional bkgregion Type s; mode q.

`config` Optional TOML workflow configuration; NONE keeps the legacy parameter interface Type s; mode h. Default: `NONE`.

`outdir` Output source-products directory Type s; mode q.

`response` Run per-atom RMF/ARF command generation after xselect products (yes/no) Type b; mode h. Default: `yes`.

`response_mode` Response mode; context_merged is required for merged events Type s; mode h. Default: `context_merged`.

`expfiles` Optional exposure-map files, comma-separated or @listfile, matched to response atoms Type s; mode h. Default: `NONE`.

`stack` Stack mode: yes/auto, jointfit, combined, or no/none Type s; mode h. Default: `auto`.

`fallback` Fallback when response compatibility fails: jointfit or fail Type s; mode h. Default: `jointfit`.

`datamodes` Comma-separated datamodes to process; MVP supports FF,PW and skips TM Type s; mode h. Default: `FF,PW`.

`coverage_mode` Source coverage mode: auto, attitude, exposure, or obs_gti Type s; mode h. Default: `auto`.

`coverage_tstep` Maximum source-coverage attitude sampling step in seconds Type r; mode h. Default: `1.0`.

`coverage_accuracy` Spatial coverage sampling accuracy: fast, standard, or precise Type s; mode h. Default: `standard`.

`coverage_grid` Spatial coverage grid samples per axis Type i; mode h. Default: `16`.

`coverage_minfrac` Minimum valid area fraction for a time sample to enter SRCGTI Type r; mode h. Default: `0.01`.

`coverage_warnfrac` Warn when mean valid area fraction is below this value Type r; mode h. Default: `0.1`.

`arf_mode` ARF mode: auto, aspect, exposure, or none Type s; mode h. Default: `auto`.

`arf_weight` Aspect ARF weighting mode: aspect or event Type s; mode h. Default: `aspect`.

`arf_thetabin` Aspect ARF theta-bin width in arcmin Type r; mode h. Default: `0.05`.

`backscal_policy` BACKSCAL policy: source_effarea or region_area Type s; mode h. Default: `source_effarea`.

`background_policy` Background policy: local_sky or none Type s; mode h. Default: `local_sky`.

`background_region` Background region mode: auto, manual, or none Type s; mode h. Default: `auto`.

`bkg_inner_factor` Inner radius factor for auto annulus background regions Type r; mode h. Default: `2.5`.

`bkg_outer_factor` Outer radius factor for auto annulus background regions Type r; mode h. Default: `5.0`.

`bkg_min_area_ratio` Minimum auto-background area divided by source area Type r; mode h. Default: `5.0`.

`lc_binsizes` Comma-separated light-curve bin sizes in seconds; first bin keeps the legacy .lc name Type s; mode h. Default: `1`.

`bexpscale` Compatibility value 1000 only; background exposure is not rescaled Type r; mode h. Default: `1000.0`.

`products` Comma-separated xselect products to extract: spectrum, curve, image Type s; mode h. Default: `spectrum,curve,image`.

`response_kernel` Extraction kernel; xselect is used for PHA/LC/image extraction Type s; mode h. Default: `xselect`.

`xspec_smoke` Write XSPEC joint-fit smoke-test script (yes/no) Type b; mode h. Default: `yes`.

`xspec_run` Run XSPEC smoke test after writing script (yes/no) Type b; mode h. Default: `no`.

`xspec_run_mode` XSPEC run mode: stdin, atscript, or script_tty Type s; mode h. Default: `script_tty`.

`subtract_background` Write background-subtracted light curves after response generation (yes/no) Type b; mode h. Default: `no`.

`group_min_counts` Group spectra to this minimum count threshold; 0 disables grouping Type i; mode h. Default: `0`.

`resume` Reuse successful compatible stage manifests and outputs (yes/no) Type b; mode h. Default: `no`.

`dryrun` Write scripts and manifest without running xselect/response commands (yes/no) Type b; mode h. Default: `no`.

`keepintermediate` Keep rebuildable XSELECT scripts and source/background GTIs (yes/no); dryrun always keeps them Type b; mode h. Default: `yes`.

`clobber` Overwrite existing output files (yes/no) Type b; mode h. Default: `yes`.

`verbose` Logging verbosity Type i; mode h. Default: `1`.

`bkgmethod` Background combination method: auto, time, asca, counts, or weighted Type s; mode h. Default: `asca`.

`lc_errmethod` Light-curve error method: auto, input, or gehrels Type s; mode h. Default: `auto`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.2.27 `fxttool-update`

Check, install, validate, or roll back FXT software components over anonymous HTTPS.

`command` Updater command: install, status, check, verify, or rollback Type s; mode q. Default: `install`.

`component` Component to update Type s; mode h. Default: `caldb_tools`.

`source` Anonymous HTTPS release repository Type s; mode h. Default: `https://epfxt.ihep.ac.cn/downloads/fxttools`.

`ref` Release identifier or latest Type s; mode h. Default: `latest`.

`install_root` Installed platform prefix (HEASoft or standalone FXT); defaults to HEADAS Type s; mode h.

`layout` Installation layout; standard or single requires install_root Type s; mode h. Default: `auto`.

`state_dir` Private updater state directory Type s; mode h.

`validate` Validate release metadata and installed files Type b; mode h. Default: `yes`.

`dry_run` Plan without changing files Type b; mode h. Default: `no`.

`verbose` Logging verbosity: 0 warnings/errors; 1 concise operations; 2 diagnostics Type i; mode h. Default: `1`.

`mode` Query/learn mode Type s; mode h. Default: `ql`.

B.3 Compatibility and support names

B.3.1 `fxtpsf`

`fxtpsf` is an installed deprecated compatibility alias for `fxtpsfmdcopy`. It invokes the same implementation and uses the same system parameter file and parameter interface.

It emits a deprecation warning on every invocation. Deprecated compatibility alias; will be removed in a future release. Use fxtpsfmdcopy.

`fxtbkgmerge` is not installed for routine use.

`fxtchain_merged` is not installed for routine use.

`fxtfsaclean` is not installed for routine use.

`fxtsrc_addspec_compare` is not installed for routine use.

`fxtsrcmaskfill` is not installed for routine use. Use `fxtregmaskfill` instead.

B.4 Absorbed command names

`fxtasparfgen` has been absorbed by `fxtarfgen`.

`fxtsrcdet2` has been absorbed by `fxtsrcdet`.

Bibliography

[1] Li, C.K., Jia, S.M., Song, L.M., et al., 2025, RDTM

[2] Zhao H.S., Li, C.K., Wang, J., et al., 2025, RDTM

[3] Zhao, X., Xu, J., Cui, W., et al., 2024, PASP

[4] Zhao, X., Cui, W., Wang, H., et al., 2025, RAA

[5] Zhao, X., Ban, H., Cai, H., et al., 2025, EA

[6] Chen, Y., Wang, J., Yang, Y., et al., 2025, RDTM

[7] Wang, J., Zhao, X., Yang, X., et al., 2025, RDTM

[8] Wang, H., Cui, W., Zhao, X., et al., 2025, RDTM

[9] Zhang, J., Chen, Y., Jia, S., et al. 2025, Research in Astronomy and Astrophysics, 25, 11, 115019. doi:10.1088/1674-4527/ae05f8