XMM-Newton

Users' Guide
to the
XMM-Newton Science Analysis System
(for EPIC & OM)

Issue 2.0
Edited by: E. Verdugo

22.11.2002

Based on contributions from M. Ehle, A. Talavera, E. Verdugo,
J. Ballet, M. Freyberg, M. Kirsch, L. Metcalfe, J. Osborne, W. Pietsch, R. Saxton, M. Smith

Revision history

Revision number	Date	Revision author	Comments
Issue 2.0	22.11.2002	E. Verdugo	Official release
Issue 1.0	10.07.2000	Ph. Gondoin	Official release

Contents

List of Figures

List of Tables

Glossary

Acronym	Explanation
AO	Announcement of Opportunity
AOCS	Attitude and Orbit Control and Measurement Subsystem
AMS	Archive Management Subsystem
CAL	Calibration Access Layer
CCD	Charge Coupled Device
CCF	Current Calibration File
CTE	Charge Transfer Efficiency
CTI	Charge Transfer Inefficiency
CVZ	Continuous Viewing Zone
DAL	Data Access Layer
DEC	Declination, δ(J2000)
EPIC	European Photon Imaging Camera
ERMS	EPIC Radiation Monitor Subsystem
ESOC	European Space Operations Centre
ESTEC	European Space Research and Technology Centre
FITS	Flexible Image Transport System
FOV	Field Of View
FWHM	Full Width at Half Maximum
GMT	Greenwich Mean Time
GO	Guest Observer
GT(O)	Guaranteed Time (Observer)
GTI	Good Time Interval
GUI	Graphical User Interface
HEASARC	(NASA) High Energy Astrophysics Science Archive Research Center
HEW	Half Energy Width
HER	RGS high event rate (selectable mode)
HTR	High Time Resolution (mode of RGS)
LSF	Line-spread Function
MIME	Multipurpose Internet Mail Extensions
MOS	Metal Oxide Semi-conductor
OCB	On-Chip Binning
ODF	Observation Data File
OGIP	(NASA's) Office of the Guest Investigator Program
OM	Optical Monitor
OTAC	Observatory Time Allocation Committee
PHA	Pulse Height Analyser
PI	Principal Investigator
PSF	Point-Spread Function
QE	Quantum Efficiency
RA	Right Ascension, α(J2000)
RFC	RGS Focal Camera
RGA	Reflection Grating Assembly (of the RGS)
RGS	Reflection Grating Spectrometer
RPE	Relative Pointing Error
SAS	Science Analysis System
SOC	XMM Science Operations Centre
SSC	Survey Science Centre
ToO	Target of Opportunity
UHB	XMM Users' Handbook
URL	Unique Resource Location
UT	Universal Time
W90	90% energy width
WCS	World Coordinate System
XMM	X-ray Multi-Mirror Mission

1 Introduction

1.1 Scope of this manual

After observation or upon request to the XMM Science Operation Center (SOC), observation data from the ESA XMM observatory are sent to guest investigators on CD-ROMs. The CD ROMs contain raw and calibrated event files collected by the European Photon Imaging Cameras (EPIC), the Reflection Grating Spectrometers (RGS) and the Optical Monitor (OM) on-board XMM. Since April 15, 2002 the available public data can also be retrieved through the XMM-Newton Science Archive (XSA) user interface (`http://xmm.vilspa.esa.es/xsa/`). PIs of XMM−Newton observations can also retrieve their own proprietary data, through a password-protected account. The purpose of this document is to guide XMM guest investigators in analysing these data.

1.2 Required software environment

The analysis of XMM data includes quick look analysis of raw data, calibration of raw event dataset, screening of the calibrated data, extraction of images, extraction of spectra, extraction of time series, source detection and scientific analysis of the calibrated products. A large fraction of these tasks has to be conducted with the XMM Science Analysis System (SAS) which has been developed by a team of scientists located at the ESA XMM SOC and at the XMM Survey Science Centre (SSC). Informations on how to download and install the SAS software package are available on the XMM web server at:

`http://xmm.vilspa.esa.es/sas/`

1.3 More information on the web

Before analysing XMM data, XMM guest investigators should get familiar with the basic characteristics and operation modes of the scientific instrumentation on board XMM−Newton. This information is available in the XMM−Newton Users Handbook at:

`http://xmm.vilspa.esa.es/external/xmm_user_support/documentation/uhb_frame.shtml`

An other important document is the XMM Data File Handbook which provides a detailed description of the data distributed to guest observers from the XMM−Newton archive. This handbook includes in particular information about the format, structure and content of individual XMM−Newton files. The document can be downloaded from:

`http://xmm.vilspa.esa.es/pub/odf/data/tmp/XMM-SOC-DFHB-2_3.ps.gz`

The Calibration Access and Data Handbook[1] describes the calibration files and the calibration access routines which are used by the SAS to access these files. The document is accessible from the XMM−Newton calibration web page at:

`http://xmm.vilspa.esa.es/docs/documents/CAL-MAN-0001-2-1.ps.gz`

The CCF interface document describes the XMM current calibration files necessary to reduce

[1] Christian Erd, Phillipe Gondoin, David Lumb, Rudi Much, Uwe Lammers, and Giuseppe Vacanti. Calibration Access and Data Handbook. Technical Report XMM-PS-GM-20, ESA/SSD, Jan 14 2000.

and analyze the data. The document is accessible from:

`http://xmm.vilspa.esa.es/docs/documents/GEN-ICD-0005-3-4_1.ps.gz`

The present guide complements detailed descriptions of individual SAS tasks which can be accessed through the SAS web page or by pushing the "Help Task" button of the SAS GUI.

1.4 Structure of the document

The structure of the present document is as follows:

- Chapter 2 describes the successive SAS analysis steps which, in general, have to be conducted prior to the scientific analysis of XMM EPIC final data products.
- Chapter 3 describes the SAS preparation of OM data before analysis using standard astronomical software packages.

1.5 Reporting problems

Problems in using the SAS shall be reported through an observation report at:

`http://xmm.vilspa.esa.es/sas/feedback/`

If the observation report points to a software problem, the XMM-Newton SOC will raise a Software Problem Report (SPR), which will be fixed in forthcoming software versions. Any inquiry on the status of the observation reports, as well as any general question about the XMM-Newton data analysis, must be issued only through the XMM-Newton HelpDesk:

`http://xmm.vilspa.esa.es/xmmhelp/`

2 Analysis of EPIC camera data

In this chapter we describe the analysis of XMM-Newton data sets obtained with the European Photon Imaging Camera (EPIC).

In § 2.1 the structure of the EPIC related observation data files (ODF) is discussed.

The EPIC products created by the SSC standard Pipeline Processing are listed in § 2.2.

§ 2.3 explains in a rather technical way how an EPIC ODF can be re-processed up to the level of generating calibrated event lists. Such re-processing should be performed by the investigator only if the calibration has improved significantly between the time of the pipeline processing and the current time. While users may want to specify different time selections or different energy selections to those used in the pipeline, in general the SSC pipeline offers the best analysis possible, as it is the result of years of experience tuning the individual task parameters.

All following sections in this chapter assume that calibrated event lists exists (either from the pipeline or from a re-processing) and demonstrate how to create data products like images, spectra, rate-curves and source lists for further scientific analysis.

2.1 The EPIC data package

After reception of the XMM-Newton observation data package on CD-ROMs or from the **XMM-Newton** Science Archive (XSA), the guest investigator wishing to analyse EPIC data is advised to verify the EPIC related content of the data package and to inspect the EPIC pipeline products.

2.1.1 The EPIC Observation Data Files

The ODF names for the EPIC data will look something like:

- mmmm_iiiiiijjkk_aabeeeccfff.zzz, where
  - mmmm: revolution number
  - iiiiiijjkk: observation number
  - aa: detector ID (M1 - MOS1, M2 - MOS2, PN - pn)
  - b: flag for scheduled (S), unscheduled (U) observations, or (X) for general use files
  - eee: exposure number within the observation
  - cc: CCD identifier
  - fff: data identifier (see Table 1)
  - zzz: Format (FITS - FIT, ASCII - ASC)

2.1.2 The EPIC MOS Observation Data Files

The most relevant files for scientific analysis of a MOS observation are (depending on the observing mode) the imaging mode and/or the timing mode event list files together with the auxiliary

Data Identifier	Contents
IME	Event list for individual CCDs, imaging mode
RIE	Event list for individual CCDs, reduced imaging mode
CTE	Event list for individual CCDs, compressed timing mode (MOS)
TIE	Event list for individual CCDs, timing mode
BUE	Event list for individual CCDs, burst mode (pn)
AUX	Auxiliary file
CCX	Counting cycle report (auxiliary file)
HBH	HBR buffer size, non-periodic housekeeping (MOS)
HCH	HBR configuration, non-periodic housekeeping
HTH	HBR threshold values, non-periodic housekeeping (MOS)
PEH	Periodic housekeeping (MOS)
PTH	Bright pixel table, non-periodic housekeeping (MOS)
DLI	Discarded lines data (pn)
PAH	Additional periodic housekeeping (pn)
PMH	Main periodic housekeeping (pn)

Table 1: EPIC ODF data file identifiers (see Table 38 in the **XMM-Newton** Data Files Handbook)

file providing a detailed description of each frame recorded during the exposure. The structure of these files is described in the **XMM-Newton** Data Files Handbook. One imaging mode event list file is produced per CCD by the "Full Frame" and "Partial Window" instrumental modes. In the "Partial Window" mode, the imaging area of the central chip (CCD 1) is reduced to 100×100 ("Small Window" mode) or 300×300 ("Large Window" mode) pixels. In timing mode, the central chip collects spatial info only in one dimension (the X axis) whereas the Y axis is a measure of the detection time. The six surrounding CCDs always produce imaging mode event list files using the 600×600 pixel area.

2.1.3 The EPIC pn Observation Data Files

The most relevant files for scientific analysis of a pn observation are (depending on the observing mode) the imaging mode, the timing mode or the burst mode event list files together with the auxiliary file providing a detailed description of each frame recorded during the exposure. One imaging mode event list file is produced per CCD by the "Full Frame" and "Large Window" instrumental modes. Only CCD 4 is active in "Small Window", "Timing" or "Burst" mode. Hence these modes produce event list data only for a single chip. The other chips do not collect any data. In timing and burst mode, CCD 4 collects spatial info only in one dimension (the X axis) whereas the Y axis is a measure of the detection time.

2.2 The EPIC pipeline products

The Pipeline Processing produces quite a number of useful products which allow a first look at the data. With each grouping of the pipeline products (Table 2) there is an HTML (.HTM extension) file which lists the associated files and gives a short description of those files. The HTML file names are of the following format:

- PPiiiiiijjkkAAAAAA000_0.HTM, where
  - iiiiii: proposal number
  - jj: target ID - target number in proposal
  - kk: exposure ID - exposure number for target
    NOTE: The ten-digit combination of iiiiiijjkk is the observation number and is used repetitively throughout the file nomenclature
  - AAAAAA: Group ID (Table 2)

The data file names are of the form (see Table 39 in the **XMM-Newton** Data Files Handbook):

- PiiiiiijjkkaablllCCCCCCnmmm.zzz, where
  - iiiiiijjkk: observation number
  - aa: detector, M1 - MOS1, M2 - MOS2, PN - pn, CA - for files from the CRSCOR group
  - b: S for scheduled observation, U for unscheduled, X for files from the CRSCOR group (and any product that is not due to a single exposure)
  - lll: exposure number
  - CCCCCC: file identification (Table 2)
  - n: exposure map band number, unimportant otherwise for EPIC data
  - mmm: source number
  - zzz: file type (e.g., PDF, PNG, FTZ, HTM)
    - ASC: ASCII file, use Netscape, other web browser, or the "more" command
    - ASZ: gzipped ASCII file
    - FTZ: gzipped FITS format, use e.g. ds9, xmmselect, fv
    - HTM: HTML file, use web browser
    - PDF: Portable Data Format, use Acrobat Reader
    - PNG: Portable Networks Graphics file, use web browser
    - TAR: TAR file

Group ID	File ID	Contents	File Type
PPSOBS	OBX000SUMMAR	Observation Data File Summary (also copied into INDEX.HTM)	HTML
	EPX000SUMMAR	EPIC Processing Summary	HTML
	PPSSUM	PPS processing summary	HTML
	SCRLOG	PPS script log	gzipped ASCII
	ATTTSR	Attitude Time Series	gzipped FITS
ESKYIM	IMAGE_8	Sky image 0.2 - 12.0 keV	gzipped FITS
	IMAGE_1	Sky image 0.2 - 0.5 keV	gzipped FITS
	IMAGE_2	Sky image 0.5 - 2.0 keV	gzipped FITS
	IMAGE_3	Sky image 2.0 - 4.5 keV	gzipped FITS
	IMAGE_4	Sky image 4.5 - 7.5 keV	gzipped FITS
	IMAGE_5	Sky image 7.5 - 12.0 keV	gzipped FITS
	IMAGE	EPIC observation image	gzipped FITS, HTML
EANCIL	EXPMAP8	Exposure map 0.2 - 12.0 keV	gzipped FITS
	EXPMAP1	Exposure map 0.2 - 0.5 keV	gzipped FITS
	EXPMAP2	Exposure map 0.5 - 2.0 keV	gzipped FITS
	EXPMAP3	Exposure map 2.0 - 4.5 keV	gzipped FITS
	EXPMAP4	Exposure map 4.5 - 7.5 keV	gzipped FITS
	EXPMAP5	Exposure map 7.5 - 12.0 keV	gzipped FITS
	EXSNMP	Exposure sensitivity map	gzipped FITS
EEVLIS[1]	MIEVLI	MOS imaging mode event list	gzipped FITS
	PIEVLI	pn imaging mode event list	gzipped FITS
	TIEVLI	EPIC timing mode and also pn burst mode event list	gzipped FITS
EPICEXP	FBKTSR	EPIC flare background time-series	gzipped FITS
ESRLIS	EBLSLI	Box-local detect source list	gzipped FITS
	EBMSLI	Box-map detect source list	gzipped FITS
	EMSRLI	Max-like detect source list	gzipped FITS
	OBSMLI	Summary source list	gzipped FITS, HTML
CRSCOR	FCHART	Finding chart	PDF
	ROSIMG	Overlay EPIC & ROSAT image	PDF
	SNNNNN[2]	EPIC Source cross-correlation results	gzipped FITS
	DNNNNN[2]	Catalog descriptions	HTML
	FNNNNN[2]	EPIC FOV cross-correlation results	gzipped FITS

[1]:Files for only those modes which were active will be included
[2]:NNNNN - Alphanumeric ID

Table 2: Pipeline Processing data files relevant for EPIC

2.3 Running the EPIC pipeline processing

The data package received by the investigator contains EPIC pn and EPIC MOS calibrated event lists generated by the SSC pipeline processing. As the SSC pipeline is the result of years of experience tuning the individual task parameters, in general no re-processing is needed. The script of the pipeline processing (SCRCLOG) is made available to the investigator to check how the pipeline processing created EPIC products. Only if the calibration has improved significantly between the time of the pipeline processing and the current time, the investigator should reprocess the EPIC data by running the default pipeline processing meta tasks `emproc` and `epproc` (also known as `epicproc`) or the chain tasks `emchain` and `epchain`. These tasks run the calibration part of the EPIC MOS and pn pipelines in their simplest form processing all ODF components without any interaction from the user and creating calibrated EPIC event lists. The chain tasks produce the same event lists as the pipeline processing. As they are scripts they can easily be edited and they allow to produce several additional output files (mainly for diagnostic purposes). The processing tasks are run by specifying the path to the ODF directory and applying the task commands.

```
setenv SAS_CCF ‘‘path’’/ccf.cif
setenv SAS_ODF ‘‘path’’/odf
emproc / emchain
```

or/and

```
epproc / epchain
```

Input files are looked for in the directory entered via the SAS_ODF environment variable, which must also contain the general ODF files (attitude, time, summary file). Output files are created in the current directory.

At the beginning of the `epicproc` meta tasks, the task `atthkgen` creates an attitude history file, which will be used by `attcalc`

The `withinstexpids` and `selectccd` parameters allow to run `epicproc` on a subset of the data files (select a single exposure, a single instrument or a single CCD). The processing of timing modes for cases in which the pointing is offset with respect to the source, requires that the user specifies the additional parameters `withsrccoords`, `srcra`, and `srcdec`. For pn optionally one can use `timingsrcposition` in RAWY coordinates. In most other cases the default values are adequate.

The pipeline processing creates in the current directory output data sets with the following naming convention:

Every output file name starts with a string composed by one or more of the following parts, each separated by an underscore character (_):

- rrrr: the revolution number
- iiiiiijjkk: the observation identifier (see § 2.2)
- EPN or EMOS1 or EMOS2: the instrument name

- blll: the exposure identifier. For instance: S001 (see § 2.2)
- cc: the CCD number
- nn: the node number

These parts are hierarchically structured, so that, say, any data set name that contains the exposure identifier will also contain the revolution number, the observation identifier, and the instrument name.

Additional strings indicate the contents of the data set:

- AttHk: attitude housekeeping
- Gti: GTI
- AuxGti: GTI based on the auxiliary file
- FrmGti: GTI created by `emframes` or `epframes`
- HkGti: housekeeping GTI
- Evts: the data set contains an event list
- ImagingEvts: imaging mode event list
- TimingEvts: timing mode event list
- BurstEvts: burst mode event list
- Badpixels: Badpixel files created by `badpixfind` and used by `badpix`
- Temp: a temporary data set, usually removed

All the pipeline processing task inside the proc meta tasks can also be run individually by the user.

2.3.1 Running the EPIC MOS processing chain

A description of the functional organigram for the EPIC MOS processing is given in Figs. 1 and 2. The `emproc` meta task chains and loops over all first-level EPIC MOS tasks to produce calibrated event lists for all selected exposures.

The main program loops over all selected exposures and instruments (MOS1/MOS2) present in the input directory. The main subroutine creates one (or two, if a CCD is operated in TIMING mode) event lists for a single exposure, from all relevant ODF material and (if they exist) the good time intervals generated by `tabgtigen` and the list of bad pixels produced by `badpixfind`. In a first step it loops over all CCD/nodes, calling in sequence:

1. `emframes` on the auxiliary file, the event file and the HK GTI file (if any), creating a frame file as expected by `emevents` and a CCD/node specific GTI file which will be reinjected in the final call to `evselect`.

2. `badpix` on the event list, adding the BADPIX extension. If a bad pixels file exists, it is used instead of the CAL calls.

3. `emevents` on the event list, the offset/variance file and the frame file, creating a new event list which will be propagated through `attcalc` and `emenergy` to `evlistcomb`.

4. `gtialign` on the HK GTI file and the event file, then the task `gtimerge` to merge the resulting aligned GTI and the CCD/node specific GTI.

5. `attcalc` on the new event list, filling the X/Y columns.

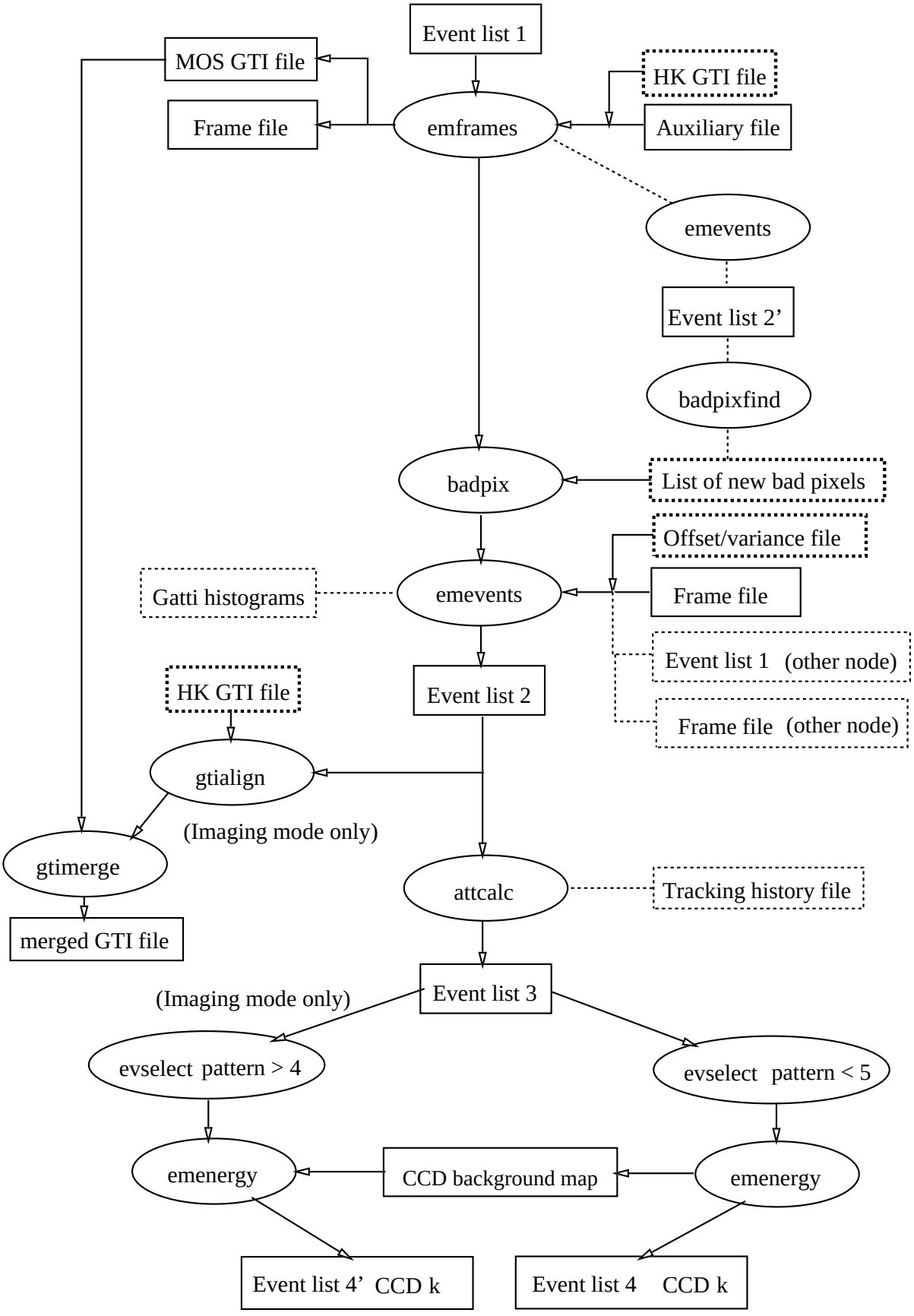
6. `emenergy` on the new event list, filling the FLAG, PHA and PI columns.

Then `evselect` is called on the resulting event list(s) applying (by default) the destructive filter selection "(#XMMEA_EM) && (FLAG & 0x762a0000)". Note that in case of `emchain`, it is not true that "(#XMMEA_EM)" is applied: here events flagged as OUT_OF_FOV and REJECTED_BY_GATTI are kept in the list (as they are useful for background assessments and flare screening, respectively). For a description of the event attribute based selection, refer to the documentation of the SAS package `evatt`.

All the event list files created (one per CCD/node) are merged by `evlistcomb`, creating one event list per mode (IMAGING, TIMING). Finally, `evselect` is called once more on the resulting merged event list(s) selecting all those events arriving in good time intervals.

In the EPIC MOS imaging mode, the EVENTS binary table of the calibrated event list files contain 12 columns i.e TIME, RAWX, RAWY, DETX, DETY, X, Y, PHA, PI, FLAG, PATTERN and CCDNR. DETX and DETY are the event position in the focal plane array. X and Y are the event position in sky coordinates. PHA is the pulse analyser channel and PI the pulse independent channel. CCDNR is the CCD number. For a description of the FLAG column, see the documentation of the SAS package `evatt`. The PATTERN definition is given in the documentation of the task `emevents` (see also Fig. 4).

EPIC/MOS chain per CCD



EPIC/MOS chain per MOS

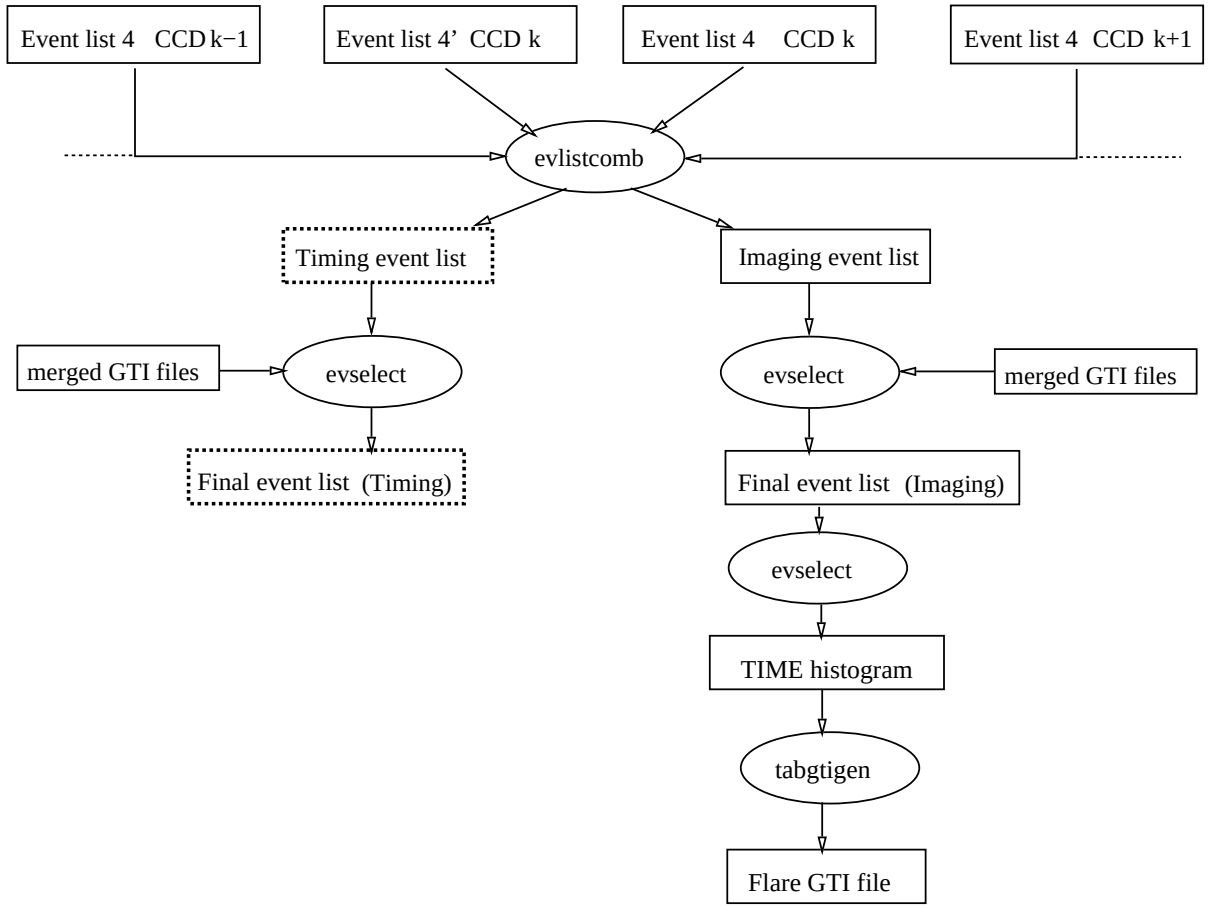


Figure 2: Organisation of the EPIC-MOS chain: merging the event lists of different CCDs.

2.3.2 Running the EPIC pn processing chain

The EPIC pn meta task `epproc` chains and loops over all first-level pn tasks to produce calibrated event lists. The pn processing is sketched out in Fig. 3. The main subroutine (`epframes`, `badpixfind`, `badpix`, `epevents` and `attcalc`) creates one event list for a single exposure and for all selected CCDs from all the relevant ODF material and bad pixel lists calling in sequence:

1. `epframes` to process a CCD, exposure and datamode specific ODF file, creating the output raw event list and GTI data set,
2. `badpixfind` to find new bad pixels,
3. `badpix` to process the raw event list, adding the BADPIX extension,
4. `epevents` to process the event list file, flagging trailing events, performing split events pattern recognition, CTI and gain correction to create the calibrated event list,
5. `attcalc` to calculate the X and Y sky coordinates.

Finally, making use of the task `evlistcomb`, the CCD specific data sets are merged into a single event list. `evselect` selects all those events arriving in good time intervals, applies (by default) the destructive filter selection "(#XMMEA_EP) && (PI > 150)" and writes the output file. For a description of the event attribute based selection, refer to the documentation of the SAS package `evatt`.

In the EPIC pn imaging mode, the EVENTS binary table of the calibrated event list files contain 14 columns i.e TIME, RAWX, RAWY, DETX, DETY, X, Y, PHA, PI, FLAG, PATTERN, PAT_ID, PAT_SEQ and CCDNR. For a description of the PAT* columns, refer to the documentation of the task `epevents`. The definition of the PATTERN values is also visualized in Fig. 5.

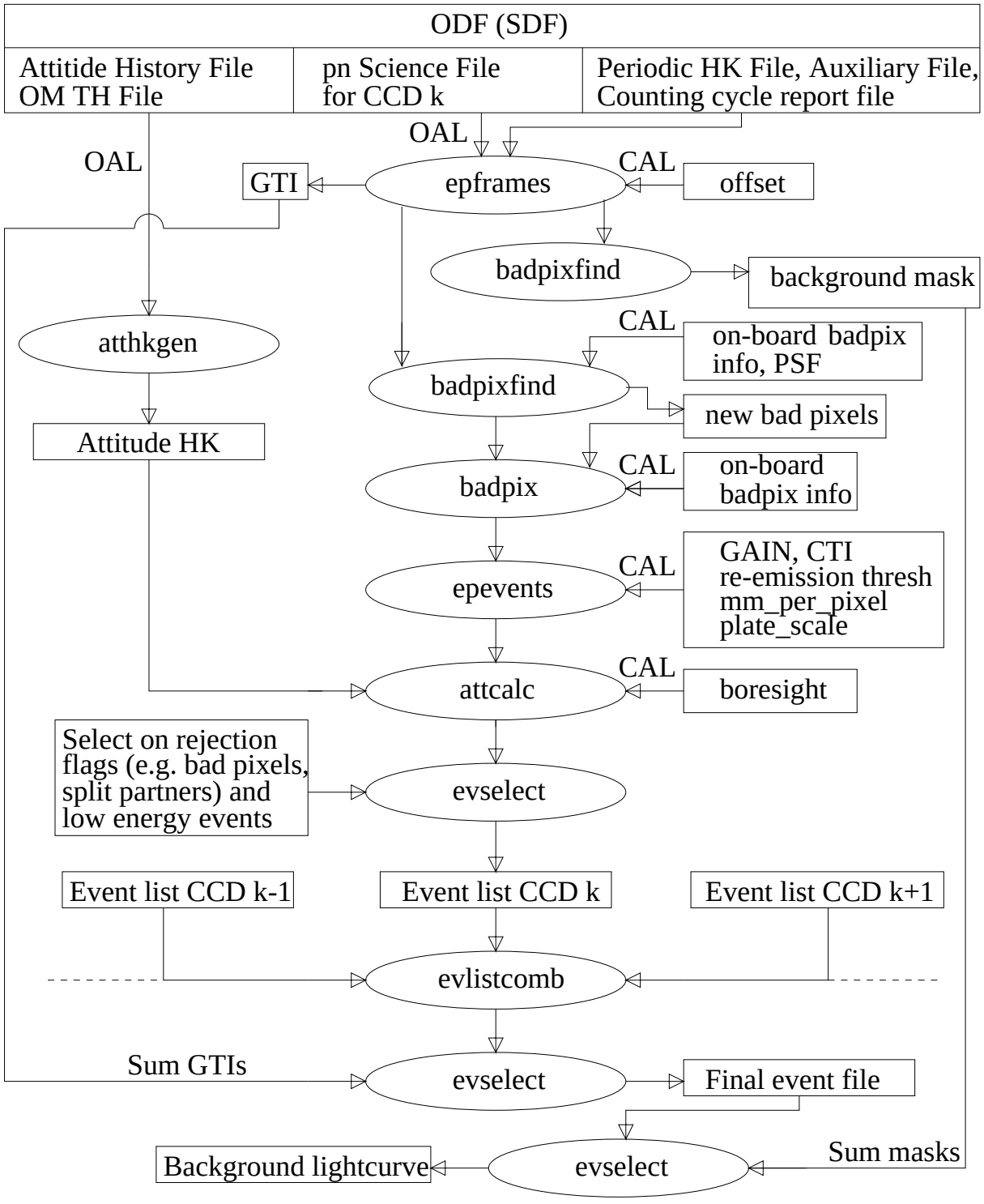


Figure 3: Pipeline processing of the EPIC pn observation data.

2.4 Filtering calibrated EPIC event lists

EPIC calibrated event lists must be processed to generate data products including images, spectra and rate curves.

All product creations can be accomplished using the `evselect` task or (via a user friendly Graphical User Interface; GUI) the `xmmselect` task. The event list is filtered on the fly according to user-specified selection criteria. The filtered event list can be stored on disk and/or products can be extracted from this filtered event list.

The filtering process is carried out on an event-by-event basis controlled by user-specified selection criteria. These criteria take the form of character strings which can be composed of a variety of elements including numerical and logical constants, operators, functions, and attributes. Arithmetic and logical operations, file-based filtering, new column or arrays construction and symbolic reference to other arrays can be performed.

Events for which the selection expression does not evaluate to true will be marked as invalid or discarded from the data set and shall not be considered in the product extraction. `xmmselect` (and `evselect`) support selections based on intrinsic event attributes. These attributes are present in the input event list table as corresponding columns, the names of which are to be used in the selection expression. This group further falls into the following sub-categories.

1. Spatial selections in raw pixel (RAWX/RAWY), camera coordinates (DETX/DETY) or sky pixel (X/Y) space, or in any spatial coordinate system. Special pre-defined region shapes exists such as CIRCLE, ELLIPSE, ANNULUS, BOX, POLYGON. Spatial filtering with mask images or region files is possible as well.
2. Energy selections in the form of one or more interval specifications in either PHA or PI space.
3. Time selections in the form of interval specifications or Good Time Interval (GTI) files created with e.g. `tabgtigen`.
4. Any other event attributes which are present (e.g. PATTERN or CCDNR)

The user should consult the `selectlib` package description which details the syntax and semantics of the expressions driving the above operations and gives sample expressions to demonstrate typical usage scenarios.

Note, while all of the data products generation can be done on the original event list, it can save significant computing time and memory first to create a filtered event list (e.g. removing high flaring background periods, restricting the analysis to certain regions and energies) and operate afterwards on the reduced filtered event file.

2.4.1 Filtering EPIC MOS concatenated event lists

Each of the two EPIC MOS cameras consists of seven individual 600 x 600 pixel CCDs. Because of telemetry constraints, a real time on-board recognition scheme filters out cosmic-ray tracks and exclusively transmits to the ground the information supposedly related to X-ray events. The on-board recognition scheme looks for a local enhancement of signal in flat fielded images. The signal enhancement is searched in 5 x 5 pixel matrix which is scanned over the full image. The signal is defined with respect to a threshold value set by telecommand for each observation. An event is identified, if in the 5 x 5 pixel matrix, pixels above thresholds formed a predefined pattern.

32 patterns have been predefined (see Fig. 4). They each correspond to an isolated event i.e. to a zone above threshold completely encircled by pixels below threshold. There are however two exceptions. Pattern 30 can be connected to a pixel above threshold on the diagonal of the center pixel. Pattern 31 can have any of the border pixels above threshold. Pattern 30 and pattern 31 are designed to quantify the amount of cosmic-rays extended tracks.

On-ground calibration has shown that soft X-rays mainly generate patterns 0 to 12 corresponding to compact regions of X-ray energy deposition. Pattern 0 events are single pixel events. These comprise most of the valid X-ray events with the most accurate energy resolution. Patterns 0 to 12 are the canonical set of valid X-ray events which are well calibrated. Selection of these patterns constitutes the best trade-off between detection efficiency and spectral resolution. However, because they deposit energy below the CCD depletion zone, higher energy X-rays also generate pattern 31 events with a probability of 20% and 50% respectively at 6 keV and 9 keV. Pattern 31 comprises largely cosmic ray events but can also include pile-up X-ray events. A large density of pattern 30, 31 events (and $26-29$ diagonal events) in the core of the telescope point spread function (PSF) is the signature of a piled response which needs careful analysis (see § 2.7).

The users are encouraged to test different filtering schemes which could be better suited to their own observation. However, general recommendations are as follows:

- The selection expression (#XMMEA_EM) shall be used to filter out artifacts from the calibrated and concatenated dataset.
- Only patterns 0 to 12 should be kept as X-ray events since patterns with higher numbers are not created by X-rays or are highly contaminated by cosmics.

2.4.2 Filtering EPIC pn concatenated event lists

The EPIC pn camera consists of twelve 64 x 200 pixel CCDs on a single wafer. For full frame and extended full frame modes the first 12 rows at the readout node are not transmitted to ground (are set to "bad", equivalent to "bad pixels"). In contrast to the MOS, all non bad pn events supposedly related to X-rays are transmitted to the ground and the pattern recognition and recombination of split partner is done off-line by the task `epevents`. Filtering of an EPIC pn dataset is entirely performed by the EPIC pn pipeline processing. The user is only advised to check the background level as a function of time, with as main aim the identification of any periods of enhanced low-energy proton flux.

For pn, 13 valid patterns have been defined. As in case of the MOS, pattern 0 events are single

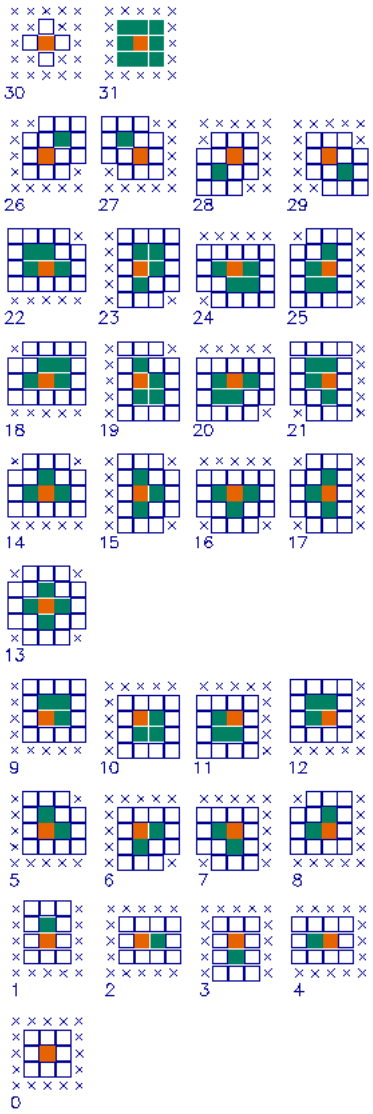


Figure 4: List of EPIC-MOS patterns: the figure should be interpreted as follows: each pattern is included in a 5x5 matrix used for proximity analysis, a pattern is centered by definition on the pixel with highest charge, this central pixel is colored in red, the other pixels above threshold in the pattern are colored in green, all pixels colored in white must be below threshold, the crossed pixels are indifferent (they can be above threshold). The philosophy for patterns 0-25 is that a good X-ray pattern must be compact, with the highest charge at the center, and isolated (all pixels around are below threshold). Patterns 26-29 are the so-called diagonal patterns, not expected from a genuine X-ray, but which can arise in case of Si-fluorescence or of pileup of two monopixel events.

pixel events. These comprise most of the valid X-ray events with the most accurate energy resolution. Patterns 0 to 12 (see Fig. 5) are the canonical set of valid X-ray events which are well calibrated. Selection of these patterns constitutes the best trade-off between detection efficiency and spectral resolution. All higher patterns are not created by single X-ray photons and are due to pattern pileup.

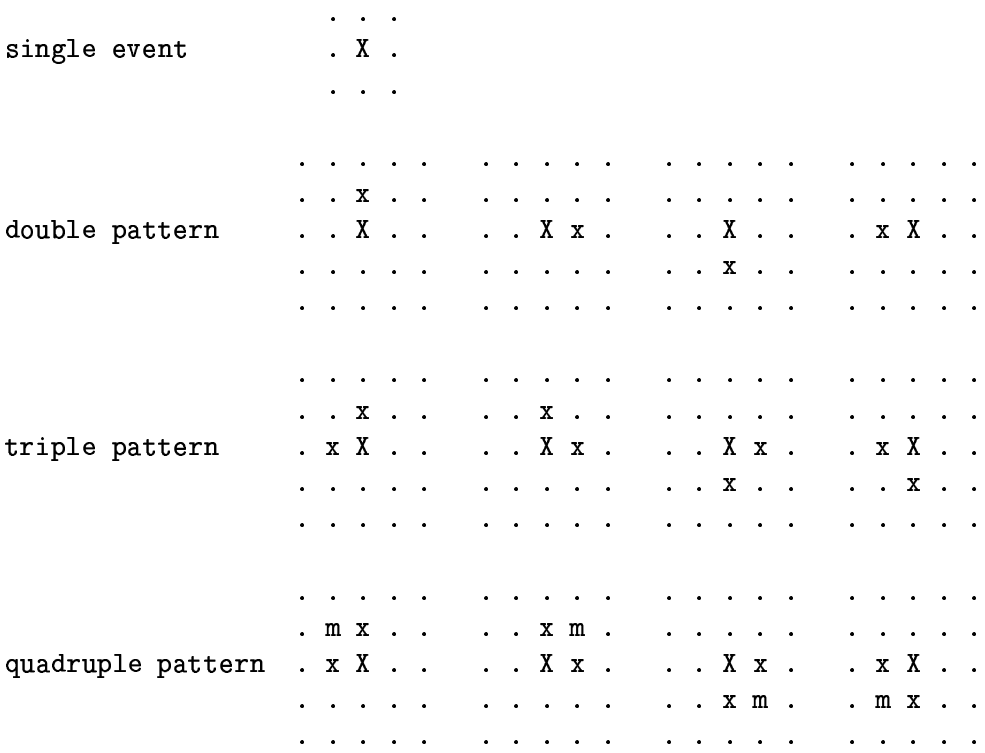


Figure 5: List of valid EPIC-pn patterns (cf. Fig. 4). Here "." marks a pixel without an event above threshold, "X" is the pixel with the maximum charge ('main pixel'), 'x' is the pixel with a non-maximum charge, "m" is the pixel with the minimum charge. These 13 figures refer to the SAS PATTERN codes 0 (singles), 1-4 (doubles), 5-8 (triples) and 9-12 (quadruples), respectively. The RAWX co-ordinate is running rightward and the RAWY co-ordinate running upward.

The users are encouraged to test different filtering schemes which could be better suited to their own observation. However, general recommendations are as follows:

- The selection expression (#XMMEA_EP) should be used to filter out artifacts from the calibrated and concatenated dataset.
- Only patterns 0 to 12 should be kept as valid X-ray events since patterns with higher numbers are not created by X-ray or are highly contaminated by pileup.

Users need to beware of the spatial non-uniformity of low energy multi-pixel events, e.g. when defining source and background accumulation regions (§ 2.9).

2.4.3 Filtering high background periods

The user is advised to produce a histogram of TIME values (a rate curve) in order to identify the useful period of low background level when the focal plane is not illuminated by low-energy protons.

Before any filtering, the user is also advised to inspect the background lightcurves produced by SSC pipeline processing (FBKTSR) where all sources and bright features/pixels have been

masked out (see task descriptions of `emchain` and `epchain` for further details). If bright hard point-like sources are in the FoV, such sources should be excluded prior to the interactive rate curve generation as well.

In order to check for and remove any additional high background periods from the event list, the user can apply the following recipe:

- Build a rate curve of the time column in the calibrated event list using only valid single events with energy greater than 10 keV. This can be performed using `xmmselect` by entering 10000 in the PI lower limit box, 0 in the PATTERN upper limit box, pressing the PI and PATTERN button, and in addition appending "&& #XMMEA_EM" in case of MOS, or "&& #XMMEA_EP" in case of pn in the filter expression box. The round radio button in the TIME row should then be pressed. A suitable time bin needs to be selected for the OGIP rate curve accumulation, e.g. 25, 50 or 100 seconds, depending on the exposure duration.
- Examine the rate curve produced and identify periods with a constant low count rate and periods with a high background level. Select a suitable count rate threshold which lies a little above the low background rate. Recommended values are 0.35 counts/s for the MOS and 1 count/s for the pn camera, respectively, but depend on the science the user wants to do.
- Assuming that the rate curve is named `rates.fits`, a new GTI file can be created from the selected count rate threshold using a command line as follows (changing the count rate value for pn accordingly):

  ```
  tabgtigen table=rates.fits gtiset=BKG_GTI.fits \
            expression='RATE < 0.35'
  ```

- This GTI selection can then be passed to the existing event list using:

  ```
  evselect expression='gti(BKG_GTI.fits,TIME)' ...
  ```

  When using `xmmselect` from the GUI, the above expression should be typed in the selection window.

Low background intervals might differ significantly between different EPIC cameras and therefore a GTI created for one EPIC camera should not blindly be used for all the other cameras as well.

Users can also make use of GTIs which were created to make the pipeline images (only these - not the pipeline created event lists - have the flaring background removed):

```
evselect table=inevlist.fits \
         filtertype=dataSubspace dssblock=image.fits:PRIMARY \
         keepfilteroutput=yes withfilteredset=yes filteredset=outevlist.fits
```

where `inevlist.fits` is the input event list (e.g. from the pipeline), `image.fits` is the pipeline image from which to take the GTI and `outevlist.fits` is the output flare GTI filtered event list. If the user prefers the task `xmmselect` to apply such pipeline processed GTI to an input event list, the "Filtered table" product selection needs to be started. On the "General" `evselect` GUI parameter page, the "Filtering" method must be changed to "dataSubspace" and other `evselect` parameters need to be set as in the example above.

2.5 Merging event lists

The task `merge` merges EPIC event lists or additional files (attitude, orbit) from two exposures (even from different observations and instruments), pointing in the same or an adjacent direction. The output files can be used by `evselect` for the product generation and higher level detection and time analysis tasks.

Files to be merged may be the PPS product EPIC event lists or two other additional files (attitude files, orbit files etc.). In the attitude and orbit cases, any files can be merged together, whereas in the events case, some care should be taken: For instance event files from different modes should not be merged, though in some cases event files from different instruments can. Warnings are given when the event files are deemed not entirely compatible (i.e. their INSTRUME or FILTER keywords are different). Here it is up to the user to interpret the output files sensibly.

Also, in the event files case, the two pointings need to be close to each other. The accuracy of the reprojection degrades with increasing offset between the two pointings, and warnings will be given when this offset becomes significantly large. Only if the input files pass all the checks does the merging proceed and an output file (event list, attitude file, orbit file) is produced. In the event files case, the sky coordinates are deprojected and reprojected again onto a common sky position (given either by the user, via the parameters `ra` and `dec`, or as the mean of the two input pointings), and new attributes are calculated. The size of the image is given by the parameter `imagesize`.

For example, two event files can be merged reprojecting X/Y coordinates to a mean reference point using the following command.

```
merge set1=inevlist1.fits set2=inevlist2.fits outset=outevlist.fits
```

The two event files can also be merged reprojecting X/Y coordinates to a reference point defined by the user.

```
merge set1=inevlist1.fits set2=inevlist2.fits outset=outevlist.fits \
      withradec=Y ra=1.1 dec=-0.7
```

As this task is still in an experimental stage, care should be taken in using any output for a quantitative analysis. There are still many open questions regarding merging files over from different exposures, and the handling of EXPOSURE, GTI and BADPIX extensions (check the task description of `merge` for further details).

2.6 EPIC-pn Out-of-time events

For the EPIC imaging observing modes, photons are not only registered during the actual integration interval but also during the readout of the CCD (shift of charges along a column towards the readout node). These so called Out-of-Time (OoT) events get a wrong RAWY value assigned and thus finally a wrong energy correction (the correction for charge transfer inefficiency (CTI) depends on the distance from the readout node).

The effect of OoT events broadens spectral features and can be seen in EPIC images as a strip of wrongly reconstructed event positions in RAWY. The fraction of Oot events scales with the mode-dependent ratio of integration and readout time and is highest for the pn full frame (6.3 %) and extended full frame (2.3 %) mode (see the UHB for further details).

If OoT events are a problem for the analysis (if highest spectral resolution is required or if a clean image of e.g. extended emission surrounding a bright source is needed), the tasks `epproc` (§ 2.3.2) and `epchain` offer the possibility to simulate OoT events based on the original event list. Note that OoT events do not need to be removed for the purpose of source detection as they are dealt with in the background characterisation stage there (§ 2.12.3 and task description of `esplinemap`). The pipeline tasks need to be run twice, first creating an OoT event list and then the "normal" calibrated event list. The OoT event list is produced by calling the subtask `epevents` with the non-default parameter setting `withoutoftime=yes`. The OoT event list is created treating all events as out-of-time events: after the pattern recognition for the same TIME, PHA, and RAWX a new RAWY value is simulated by randomly shifting the pattern along the RAWY axis and performing the gain and CTI correction afterwards.

`epchain` resembles largely the `epproc` task but can speed up the process to remove OoT events as raw intermediate files from the first run (to create the OoT event list) can be kept for the second run (to create the "normal" calibrated event list) allowing the user to skip some of the already performed processing steps.

To produce an OoT event list from a pn imaging mode ODF, the task `epchain` should be called in the following way:

```
epchain runbackground=N keepintermediate=raw withoutoftime=Y
```

This will create an event list that contains the same number of OoT events as there are events in the "normal" event file. The Out-of-Time event file will have a name like PiiiiiijjkkPNbll-lOOEVLInmmm.FIT (cf. § 2.2).

The "normal" event list is created with:

```
epchain runatthkgen=N runepframes=N runbadpixfind=N runbadpix=N
```

In order to save time one does not have to re-create the attitude file, re-run `epframes`, `badpixfind` and `badpix` either. `epchain` will create an output event file with the name PiiiiiijjkkPN-blllPIEVLInmmm.FIT (the standard name of a calibrated pn imaging mode event list).

The two event lists can now be taken to create output products like images and spectra:

2.6.1 Removing Out-of-Time events from pn images

The effect on images is shown in Fig. 6 (btw. note the arc-like structures due to single mirror reflections (stray light) at the top left of the FoV).

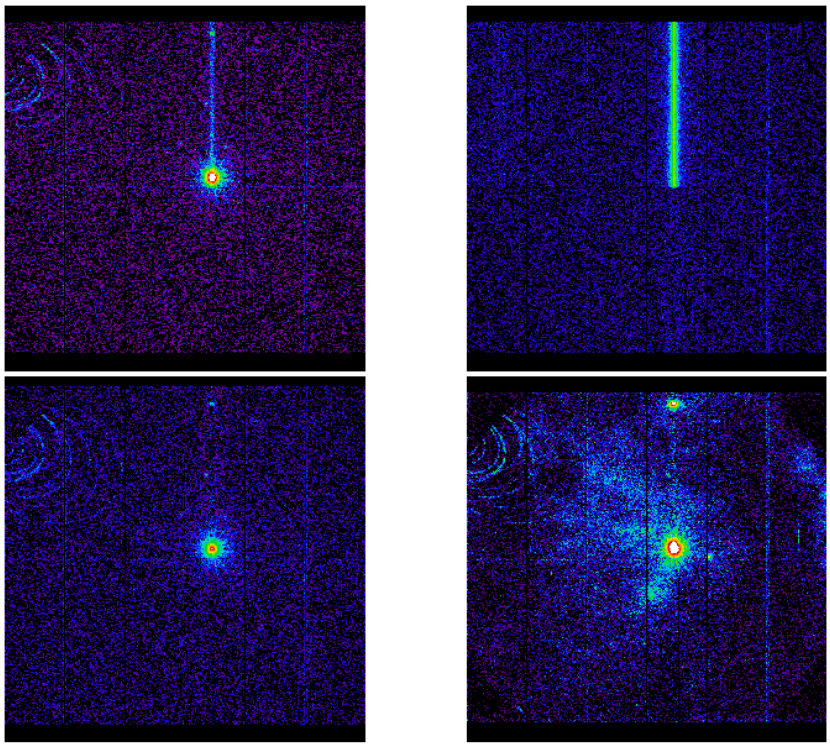


Figure 6: Effect of OoT events on images: The upper left panel contains a 2-10 keV band image of a pn observation of a bright source in full frame mode with the OoT events visible as a strip running along the length of the CCD. The upper right panel depicts the modeled OoT event distribution whereas in the lower left panel these are subtracted from the original image. The lower right panel shows the distribution of cleaned events in the soft (0.2-2 keV) energy band for comparison.

As mentioned above, 6.3 % of all events in a full-frame mode event file are OoT events. Because the OoT event list contains the same number of events as the original event list, the OoT image needs to be multiplied by 0.063 before subtracting it from the original image. Assuming that the user has created two images (see § 2.8) from the OoT and the "normal" event lists (named `image_oot.fits` and `image.fits`, respectively), the necessary image arithmetic can be performed with the `FTOOL` task `farith`:

1. The OoT image is multiplied by 0.063 creating the output file `image_oot_scaled.fits`:

   ```
   farith image_oot.fits 0.063 image_oot_scaled.fits MUL
   ```

2. The scaled OoT image is subtracted from the "normal" image:

   ```
   farith image.fits image_oot_scaled.fits image_clean.fits SUB
   ```

   The resulting image `image_clean.fits` is more or less cleared from OoT events.

2.6.2 Removing Out-of-Time events from pn spectra

The handling of Out-of-Time events in spectra relies on dealing with FITS tables. As in the case of image cleaning of OoT events (see § 2.6.1 above), the necessary table manipulation and arithmetic can be performed with `FTOOL` tasks, e.g. `fv` and `faddcol`. The example we give here is from an exposure of the internal calibration source. Due to the strong lines in this source it is easier to see the effects that OoT events have on a spectrum. First create two spectra, one from the "normal" event file, e.g. `P0122320101PNS003PIEVLI0000.FIT` and the other one from the OoT event file `P0122320101PNS003OOEVLI0000.FIT`:

```
evselect table=P0122320101PNS003PIEVLI0000.FIT withspectrumset=yes \
         spectrumset=source.fits energycolumn=PI \
         withspecranges=yes specchannelmin=0 specchannelmax=20479 \
         spectralbinsize=5 \
         expression='(FLAG==0)&&(PATTERN==0)'

evselect table=P0122320101PNS003OOEVLI0000.FIT withspectrumset=yes \
         spectrumset=source_oot.fits energycolumn=PI \
         withspecranges=yes specchannelmin=0 specchannelmax=20479 \
         spectralbinsize=5 \
         expression='(FLAG==0)&&(PATTERN==0)'
```

The next step is to copy the column `COUNTS` from the OoT event spectrum `source_oot.fits` into the source spectrum file `source.fits` (which is to be cleaned from OoT events): Then multiply this new column in the file `source.fits` by 0.063 (same explanation as given above for images). Then, as the last step, subtract this column from the original column COUNTS of the source spectrum file `source.fits`. The result is shown in Fig. 7.

The effect of OoT events basically is that they broaden the shoulders of a line and can cause a blending of lines. OoT events do not change the Full-width-at-half-maximum. The red spectrum shows that the lines in a spectrum cleared from OoT events are much easier to separate than in an untreated spectrum (black data points). Please keep in mind that the spectrum is displayed in a double logarithmic scale and restricted to a small energy band so as to emphasize the effect. Furthermore, the example shown here is data from the internal calibration source, which is very bright. **For most targets a correction for OoT events in a spectrum should not be necessary. In any case, a correction is only necessary if OoT events overlap the source being investigated.**

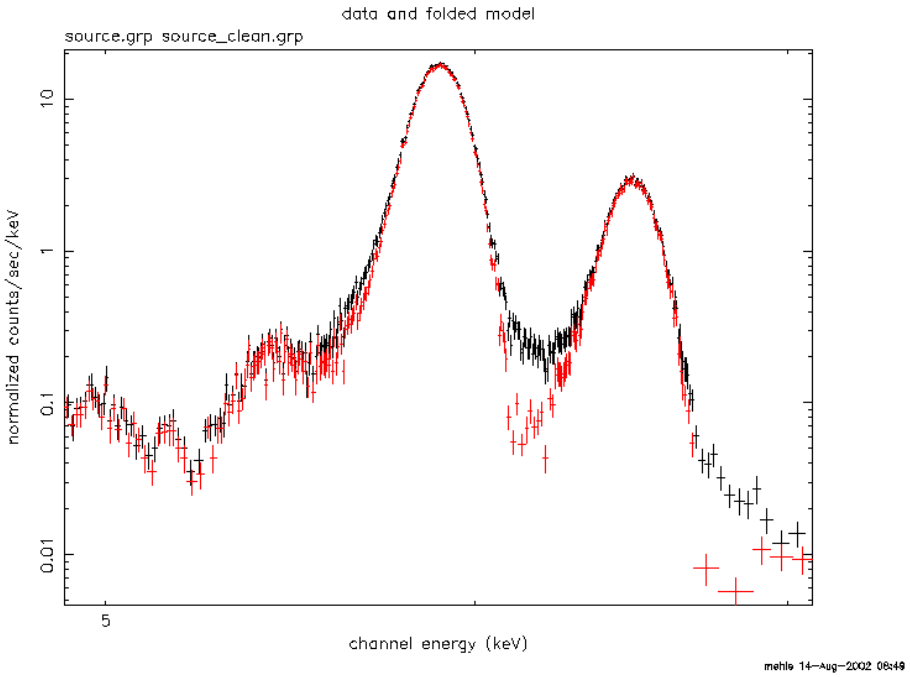


Figure 7: Effect of Out-of-Time events on a source spectrum (in this case the internal calibration source): the black data points display the source spectrum which is still contaminated by OoT events, the red points mark the source spectrum being cleared from OoT events. The energy range displayed is 5-7 keV.

2.7 Pile-up

Pile-up occurs when a source is so bright that there is the non-negligible possibility that two or more X-ray photons deposit charge packets in a single pixel ("photon pile-up"), or in neighboring pixels ("pattern pile-up", i.e. singles pileup to doubles etc.), during one read-out cycle (i.e. one frame). In such a case these events are recognized as one single event having the sum of their energies. If this happens sufficiently often, this will result in a hardening of the spectrum as piled-up soft events are shifted in the spectrum to higher energies.

In addition, pile-up leads to a more or less pronounced depression of counts in the central part of a bright source, resulting in flux loss. Pile-up also affects lightcurves, suppressing high count rates.

The XMM-Newton UHB lists (readout mode dependent) maximum count rates above which a source suffers from pile-up. In general the MOS camera is much more susceptible to pile-up than the pn.

To check whether pile up indeed is a problem, use the SAS task `epatplot`. To run `epatplot` one needs to create an event file for the source as described below in step 4) of § 2.9. The input event file name (e.g., `src_evlist.fits`) must to be specified via the `epatplot` task `set` parameter. If the resulting plot shows the model distributions for single and double events diverging significantly from the measured distributions, this is a strong indication that pile-up has occurred. Fig. 8 shows an example of a bright source observed in pn full-frame mode which is strongly affected by pile-up. Due to "pattern pile-up" more doubles are produced at the expense of single pixel events.

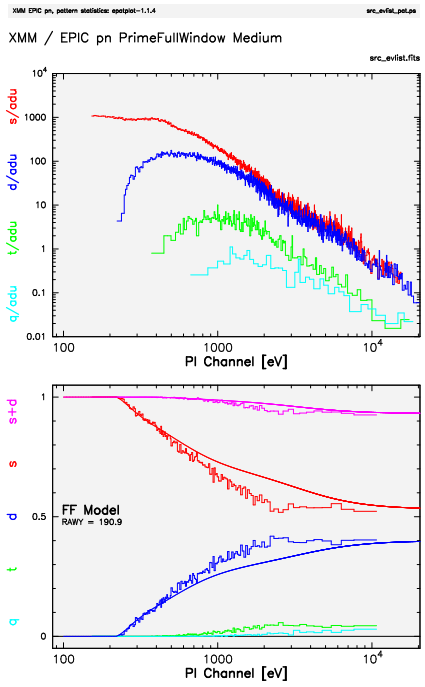


Figure 8: Plot of the pn pattern distribution with energy as produced by `epatplot`. The deviations of the single, double and single+double distributions from the model are clearly visible.

The spectral analysis of the X-ray emission from a source suffering from pile-up can still be

performed by excluding the inner part of the source emission from the event list used for the creation of a spectrum.

This can be achieved via the `xmmselect` task, first displaying the whole source region as an image (see § 2.8) and then defining an annulus (via two concentric circle regions, selecting all regions in the `ds9:Region` menu, importing these regions into the `xmmselect` selection expression via the "2D region" button (see § 2.9, step 5)). With this selection expression, an event list, named e.g. `src_annulus_evlist.fits`, can be created with `xmmselect`. Finally `epatplot` should be called again now with the `src_annulus_evlist.fits` as input data set. Inspecting the created pattern distribution curves, the inner circle region should be increased as long as the pattern distribution agrees with the model. Note, excluding the inner part of the source from the analysis will of course reduce the number of events for further analysis. So an iterative process for finding the best exclusion circle should be performed.

Fig. 9 shows the pattern distribution of the same source as above, but after exclusion of the inner part of the source. The pattern distributions now agree with the model curves and a resulting spectrum will be free of pile-up effects.

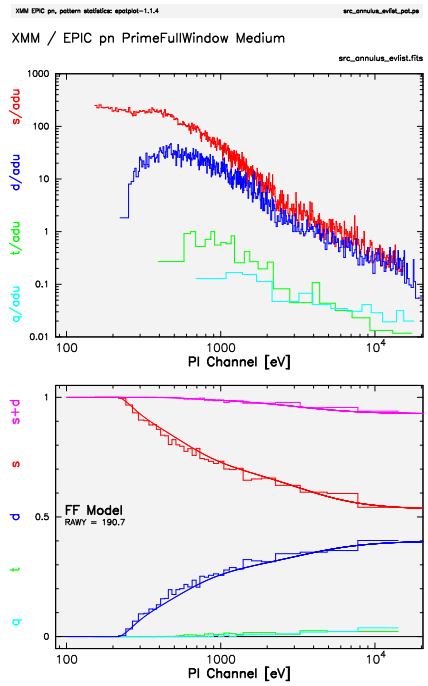


Figure 9: Plot of the pn pattern distribution with energy as produced by `epatplot`. After exclusion of the inner part of the source, the pattern distributions are in agreement with the model curves.

2.8 Generating EPIC images

EPIC images can be created from an event file with the `evselect` task from the command line or with the `xmmselect` task in an interactive GUI driven way.

2.8.1 Image generation with `evselect`

The task `evselect` creates a simple output image file in the following way:

```
evselect table=inevlist.fits xcolumn=X ycolumn=Y imagebinning=binSize \
         ximagebinsize=100 yimagebinsize=100 \
         withimageset=true imageset=image.fits
```

where `table` specifies the input event list, `x/ycolumn` the coordinates used for the image (here sky coordinates), `imagebinning` is a switch for an automatic or user defined binsize, `x/yimagebinsize` is the user defined bin size of the image in each direction, `withimageset` is the switch to create an image and `imageset` defines the name of the output image file. In this example the bin size is set to 100 which corresponds in the sky pixel system (units are 0.05 arcsec) to 5 arcsec.

The image can be displayed with the command:

```
ds9 image.fits
```

2.8.2 Image generation with `xmmselect`

The more comfortable way is to create images via the `xmmselect` GUI which is started with the command:

```
xmmselect table=inevlist.fits
```

the following interactive steps need to be performed to create an image:

1. Choose X and Y in the Column selection (input columns for the image, e.g. X and Y (sky coordinates), RAWX and RAWY (raw CCD-specific coordinates) or DETX and DETY (camera-specific coordinates)) and click in the Product selection part on the "Image" tab. This will start the general window of the `evselect` task (see Fig. 10).

   Caution: if `x/ycolumn` in `evselect` are the CCD raw coordinates RAWX and RAWY, all selected CCDs will be projected on top of each other. If an image in RAWX and RAWY is needed e.g. for diagnostic purposes, an additional filter expression selecting a single CCD only should be applied. In order to get an image of all active CCDs in the camera, the `x/ycolumn` should be set to X and Y or DETX and DETY.

2. In the `evselect` main window one now has the option to apply further events filtering (if not yet specified in the `xmmselect` main window). If one is happy with the set-up, the next step is to click on the "Image" button on the top of the GUI (see Fig. 11).

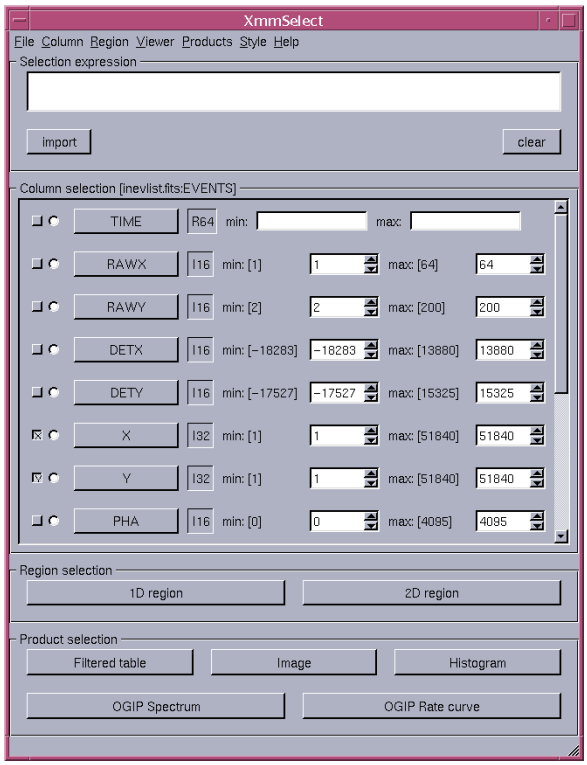


Figure 10: In the `xmmselect` main window, an image can be extracted by selecting X/Y as image axis and by pushing the "Image" button.

3. In the currently visible `evselect` window with the image creation related parameters, one needs to specify a name for the output image (parameter `imageset`). By default, events will be binned into an image with 600×600 square pixels. The image size can be modified as well as the binning mode: setting `Binning` to `binSize` allows e.g. to project events onto a grid with specified pixelsize (a value of `x/yimagebinsize`=100, e.g. will result in 5×5 arcsec pixels), see Fig. 12.

4. Click on the "Run" button to start the image creation process. After the image is created and stored in the working directory, `xmmselect` will automatically launch the image viewer. The `ds9` viewer allows the user to zoom in on a region of special interest, to change the intensity scale and the color. The position of the mouse pointer is displayed (in RA and Dec in case of a sky image, or in linear coordinates in case of a camera or raw coordinate image). For details on the image viewer (developed by the R&D Group at the Smithsonian Astrophysical Observatory), follow the links to the viewer manuals by clicking on the `ds9` "Help" button.

Note, if running the image creation for a second time with the same name for the output image (parameter `imageset`), the previously created image file will be overwritten. This can be avoided by starting `xmmselect` with the "-c" (no clobber) option (see documentation of the `taskmain` SAS package).

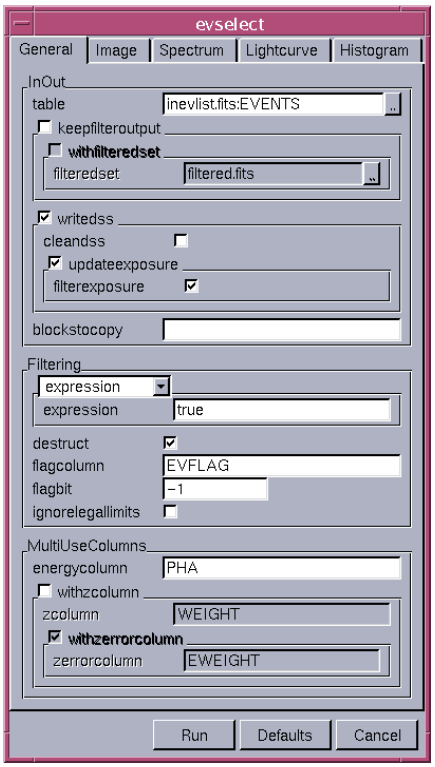


Figure 11: The `evselect` main window where e.g. selection criteria still can be modified.

2.9 Generating EPIC spectra

EPIC calibrated event lists must be filtered to generate spectra. As for images, this filtering is normally performed in two steps. First, the event lists are screened to reject spurious data and to select only events which contain information of sufficient quality for further scientific analysis. Secondly, the screened data are filtered to construct data subsets adapted to specific spectral analysis. These screening and filtering activities can be performed in an interactive manner with the SAS task `xmmselect`:

1. With `xmmselect` running on a filtered (or the original) event list, create an image (see § 2.8.2).

2. In the `ds9` window, create a region for the source of interest. Click once on the `ds9` image and a region circle will appear (other region shapes are available as well). Click on the region circle and the region will be activated, allowing the region to be moved and its size to be changed. Having created, placed, and sized the region appropriate for the source, click the "2D region" button in the `xmmselect` GUI. This transfers the region information into the "Selection expression" text area in `xmmselect`. A newly defined `ds9` region file can optionally be saved to disk via the `ds9` "Region" → "Save Regions..." menu and re-loaded for further analysis steps via the "Region" → "Load Regions..." option.

   Note 1: For serious spectral analysis the string `'&& (FLAG == 0)'` should be added to the selection expression. This will exclude events next to the edges of the CCDs and next to bad pixels which may have incorrect energies.

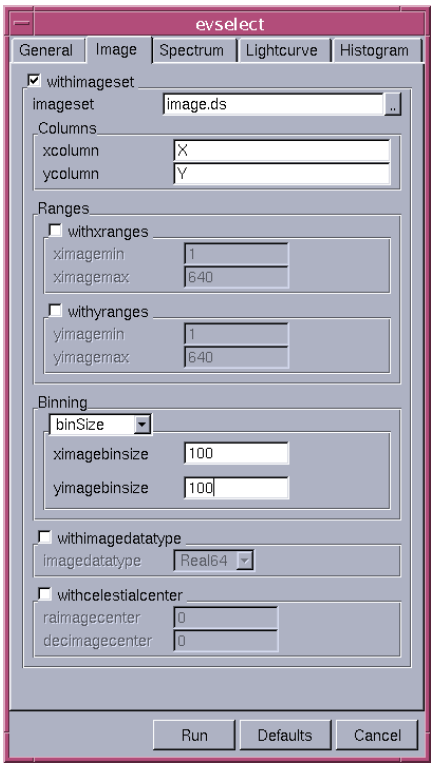


Figure 12: The `evselect` window with the image related parameters, where e.g. the output image name and the binning of the events can be modified.

Note 2: For pn data, add the string `'&& (PATTERN <= 4)'` which will select single and double events (which have the best energy calibration). For MOS data, use `'&& (PATTERN <= 12)'` to select all valid events.

Note 3: For pn data of bright sources and of sources with narrow lines it might be better to extract two spectra and corresponding backgrounds, response and ancillary files: one set for single pixel events `(PATTERN == 0)` and another set for doubles `(PATTERN IN [1:4])`. Fitting these two spectra simultaneously will show if there are any problems with pile-up (see § 2.7) and - as the energy calibration for singles is slightly better than the one for doubles - will show the line features at highest energy resolution in the single events spectra. However, in case of pn timing mode observations (where the ratio of single to double events depends on the source position) one should always create and fit a spectrum of the combined single and double events.

3. To extract the spectrum, first click the circular button next to the PI column in the `xmmselect` GUI. Next click the "OGIP Spectrum" button. Select the "Spectrum" page of the `evselect` GUI to set the file name and binning parameters for the spectrum. For example, set `spectrumset` to `src_spectrum.fits`. If the canned responses are going to be used (see § 2.10), the `spectralbinsize` must be set to 15 for the MOS or 5 in case of the pn. `withspecranges` must be checked, `specchannelmin` set to 0, and `specchannelmax` set to 11999 for the MOS or 20479 for the pn. Fig. 13 shows an example spectrum.

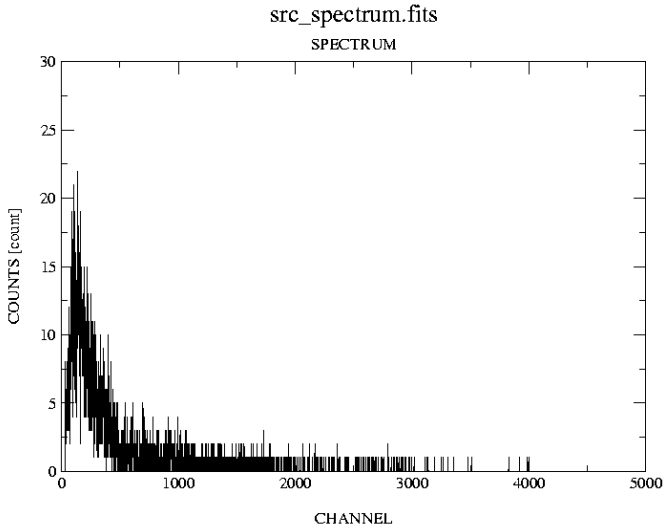


Figure 13: Spectrum of a source.

Note: `xmmselect` performs the calculation of the `BACKSCAL` factor, which takes into account CCD gaps, bad pixels and the size of the extraction region, on the fly during the spectrum generation. If `evselect` is used instead for the extraction of the source and background spectra, the `BACKSCAL` factor must be calculated explicitly by executing the `backscale` task before any subsequent quantitative analysis, e.g. with `Xspec`, can be performed. For a purely qualitative check of the source spectrum, however, the `BACKSCAL` computation step can be skipped to save significant computing time especially in case of large extraction areas.

4. For checking whether pileup (§ 2.7) might be a problem, create a source event file by checking the `keepfilteroutput` and `withfilteredset` boxes on the `evselect` "General" page and provide a `filteredset` name, e.g. `src_evlist.fits`, for the resultant file. For this event file, remove the `&&(PATTERN <= 4)` phrase for the pn so that single, double, triple, and quadruple events are all included. This filtered event list should be used as input file for the pattern analysis to be performed by `epatplot` (see § 2.7).

5. To extract a background spectrum from an annulus surrounding the source, first clear the "Selection expression". Next repeat step 2) except using now two circles defining the inner and outer edges of the background annulus and then click the "2D region" button. This will transfer the region description of both circles to the "Selection expression". To define an annulus, this selection expression still needs to be edited: e.g. for an annulus the modified selection expression shall look something like ((X,Y) IN circle(26144.5,22838.5,1200))&& !((X,Y) IN circle(26144.5,22838.5,600)). This filter will include data within a circle of radius 60" but not within a concentric circle of 30". Finally, repeat step 3) except setting now the `filteredset` parameter to a different file name, e.g. `bkg_spectrum.fits`. The inclusion and exclusion of regions can also be set directly via the pull-down `ds9` menu ("Properties" under the "Region" menu).

Note: Background extraction from an annulus around the source might be a problem in case of crowded fields. As a general advice for MOS, the source-free background region should be extracted at roughly the same off-axis angle as the position of the source. For pn, the recommendation is to select a source-free background region at the same RAWY position on the chip as the source (RAWY being the long axis of the CCD). So if the pn source is located e.g. at RAWY line 150 on CCD 4, one should aim for selecting the background from around line 150 on e.g. CCD 1.

In case of a bright source observed in small window mode or extended sources, where virtually no emission free regions exist on the CCD, the user is advised to make use of EPIC background files available through the SOC pages at
http://xmm.vilspa.esa.es/ccf/epic/

An alternative approach for MOS might be to extract source-free background regions from the outer CCDs (which always are collecting data in imaging mode).

In the case of pn timing or burst mode data, where the RAWY coordinate is not giving spatial but timing information, background regions should be extracted in strips parallel to the readout direction, excluding the region with registered events from the source (visible as a bright strip when plotting RAWX against RAWY).

Many users have asked for background data sets with which to analyse extended objects, particularly where determination of background is difficult from their own data sets. There is a SAS task in development to provide this, however until available in a satisfactory state, XMM-Newton SOC has provided some event lists for each EPIC camera that could be used ad interim. These files have an extension containing a calibrated event list in the same format as produced by SAS. The EPIC background files are available at http://xmm.vilspa.esa.es/ccf/epic/
together with an explanatory note (CAL-TN-0016-2-0) on how to use them.

At the time of writing, the spectral analysis of extended sources (e.g. clusters of galaxies) is not yet investigated enough to update this section with a detailed analysis recipe.

2.10 Creating EPIC response matrices

Analysis of EPIC data products is generally performed by specialised software packages including `Xspec` or `Xronos` (see http://heasarc.gsfc.nasa.gov/docs/xanadu/xanadu.html). In addition to the calibrated products, some of these packages require the generation of specific files. In particular, the spectral fitting technique used by `Xspec` requires a characterization of the EPIC detector response to simulate an output spectrum observed by EPIC. The response function gives the probability that an incoming photon of energy E will be detected in a channel I. This discrete function can be calculated as a product of a Redistribution Matrix File (RMF) by an Auxiliary Response File (ARF). These response files shield the user from the complexity of the EPIC instrument response which varies across the field of view.

There are currently two approaches to obtain RMF redistribution matrix files:

1. The user can make use of ready made (canned) response matrices made available by the EPIC team and accessible through the SOC pages at
   http://xmm.vilspa.esa.es/ccf/epic/
   They are virtually identical to the files produced by the SAS task `rmfgen`.

Special care must be taken in choosing the appropriate canned matrices as they depend on the EPIC mode, the pattern selection and (in case of the pn) on the distance from the readout node where the spectrum was extracted.

Note: in SAS 5.3.3 `rmfgen` does not yet support pn timing or burst mode data and so ready-made matrices must be used for these modes.

2. The user can also create the response matrix using the `rmfgen` task (even though `rmfgen` can take a long time to run). The input spectrum file contains the necessary ancillary information to allow the correct response to be made. The `rmfgen` task can then reformat the detector response and energy bounds according to the information provided by the calibration access layer. It corrects for instrumental effects specific to the events and writes the result to a specified dataset. It groups response data above a threshold value and accounts for spectral distortion due to pile-up and charge transfer inefficiency (CTI) (both not yet maintained).

The ARF response file of the EPIC camera shall then be generated by the task `arfgen`. This task calculates an effective area curve as a function of energy, to be used in conjunction with the RMF file generated before. For each row of the RMF there is a corresponding element in the 1-D ARF. This is normally adjusted by specifying the previously generated response matrix as an input file to the `arfgen` task.

The `arfgen` task generates an ARF file taking into account the following effects:

1. Telescope effective area including vignetting by the RGA structure for the MOS cameras,
2. EPIC filter transmission,
3. EPIC CCD quantum efficiency,
4. Fraction of the PSF in the accumulation region (including chip gap and bad pixel effects),
5. Pile-up effect *(while a basic pile-up correction implementation is available, this feature is not currently being maintained)*,
6. Out-of-time events smearing (pn).

The above effects generally depend on the source position in the EPIC field of view. Spatial response variation over an extended source is also taken into account.

2.11 Detecting EPIC X-ray sources

The SAS metatask `edetect_chain` is a powerful tool to perform source detections on EPIC datasets. It allows searching for sources simultaneously in several energy bands, applying different source detection methods, creation of a final combined source list and the computation of sensitivity maps which contain info on point source detection upper limits. It performs the same kind of analysis as occurs in the SSC pipeline to make PPS products.

Task	Purpose	Input files	Output files
`eexpmap`	creation of exposure maps	attitude file, event list, image	exposure maps
`emask`	creation of detection masks	exposure map	detection mask
`eboxdetect` (local mode)	sliding box detection	images, exposure maps, detection mask	box detect source list
`esplinemap`	creation of background maps	image, exposure map, detection mask, local box list	background map
`eboxdetect` (map mode)	box detection using background maps	images, exposure maps, detection mask, bkg. maps	map detect source list
`emldetect`	maximum likelihood fitting	images, exposure maps, bkg. maps, map detect list	final source list
`esensmap`	creation of sensitivity maps	exposure map, detection mask, bkg. map	sensitivity map

Table 3: Schematic overview of the EPIC source detection chain and related input and output data sets

The several subtasks called from `edetect_chain`, their main purposes, needed input files and created output data sets are summarized in Table 3.

The final output of the EPIC source detection process is a merged source list FITS file which - for every detected source - lists among other parameters the source identification number, the instrument and energy band where it was detected, the source counts, source position and extent, source flux and count rate as well as hardness ratios. The user should read the task description of `emldetect` for a complete list of the output source list columns.

Another output file is a sensitivity map giving (rough) point source detection upper limits (vignetting corrected source count rate corresponding to the likelihood of detection as specified in the parameter file) for each image pixel. **Caveat:** at the time of writing, the sensitivities calculated by `esensmap` are sometimes a factor of 2 to 3 higher than count rates and likelihoods calculated for sources detected by `emldetect`, i.e. values in the sensitivity map can be too high.

In § 2.12.3 we show an example of how `edetect_chain` and related subtasks actually perform the EPIC source detection.

The task `ewavelet` provides an alternative source detection method based on a Mexican hat wavelet algorithm which is especially suited well for the detection also of extended sources. As the determination of source parameters computed by `ewavelet` is not (yet) in general as reliable than those obtained from `edetect_chain`, we leave it to the reader to evaluate its performance on their specific datasets.

2.12 Processing examples of EPIC data

2.12.1 Quick start example

After a proper installation of the SAS software package, a quick check of the SAS installation can be performed according to the following recipe:

- create and move to a working directory
- point to an observation data file (ODF)

  ```
  setenv SAS_ODF /'path to the selected ODF'/
  ```

- generate a calibration index file (CIF) from a repository of current calibration files (CCFs).

  ```
  cifbuild
  ```

- point to the newly created `ccf.cif` calibration index file

  ```
  setenv SAS_CCF ccf.cif
  ```

- extend the ODF summary file with data extracted from the instrument housekeeping data files and the calibration database creating a new summary file

  ```
  odfingest outdir=$SAS_ODF odfdir=$SAS_ODF
  ```

- run e.g the EPIC MOS or/and pn pipeline processing

  ```
  emproc &
  epproc &
  ```

  or, alternatively, use the chain tasks

  ```
  emchain &
  epchain &
  ```

  After some time, a set of calibrated event files will have been created in the working directory including the merged event lists of all active MOS and pn CCDs, respectively.

- In order to inspect the content of these files, the Graphical User Interface (GUI) of the `xmmselect` task can be started:

  ```
  xmmselect -d &
  ```

- click the double dots button located behind `table` and select (by double-clicking with the left mouse button) one of these newly created MOS or pn files on the left panel of the dataset browser. Then search for the table called EVENT on the right panel of the dataset browser. Click on the right mouse button, press "Select" and "OK" at the bottom of the dataset browser panel.

- press "Run" in the `xmmselect` window. A large `xmmselect` window appears.
- click on the square boxes in front of X and Y and click "Image". Select the image tab and, after having checked the parameter settings for the image generation, click "Run". An image will be displayed in true sky coordinates via `ds9`.
- in `xmmselect` click on the circular box in front of PI and click "OGIP-Spectrum". Select the spectrum tab and, after eventually having modified the parameter settings for the spectrum generation, click "Run". The PI spectrum will be displayed in a `grace` window.

2.12.2 Example of EPIC product generation: images, spectra & lightcurves

The following example illustrates a simple analysis session which aims to extract an image, spectrum and lightcurve of a point source within the field of view of a single EPIC camera.

We assume that a calibrated EPIC event list (either from the pipeline processing or from running the EPIC chain tasks) is present in the working directory and that the setup steps (`cifbuild`, `odfingest` and the definition of the SAS environment variables, see § 2.12.1) already were performed.

The following analysis steps are described:

1. Create light curves to check for times of background flares
2. Filter the EPIC event list to exclude bad events and periods of high background
3. Create images of the EPIC data
4. Extract source and background spectra for a bright point-like field source
5. Create RMF and ARF files for this source
6. Prepare the spectra for further analysis with `Xspec`
7. Create a light curve for the source

Commands to analyse the spectrum using `Xspec` and the light curve using `Xronos` are beyond the scope of this document. Please, check the manuals of these program packages for further info.

2.12.2.1 Working from the command line

The user can perform the described analysis with command lines by executing the following steps:

In order to avoid long names and path, it might be convenient to first define the name of the input calibrated event list as a variable:

```
set evfile=/path to PPS directory/P0123700401PNS003PIEVLI0000.FIT
```

Here we are using the pn event list from the pipeline processing of the public Lockman Hole data set.

1. Create a light-curve for the observation to check for flaring high background periods (which are best visible above 10 keV):

```
evselect table=$evfile withrateset=yes rateset=rates.fits \
         timecolumn=TIME timebinsize=10 maketimecolumn=yes \
         expression='#XMMEA_EP && (PI>10000) && (PATTERN==0)'
```

   Note, in case of MOS #XMMEA_EM shall be used.

   Plot the light-curve (see Fig. 14):

```
dsplot table=rates.fits x=TIME y=COUNTS &
```

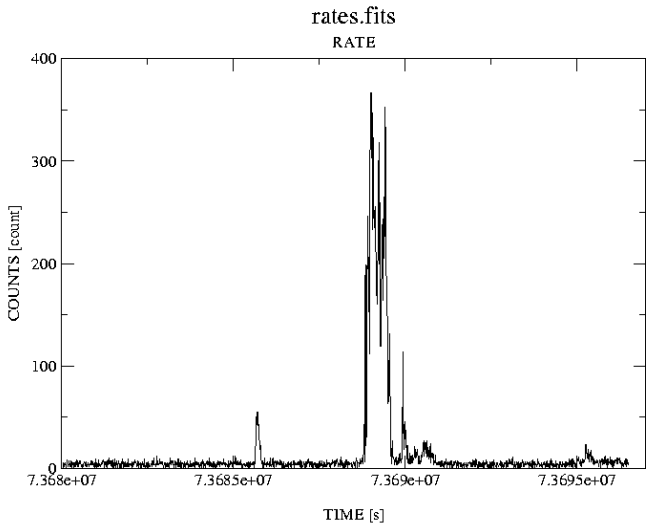


Figure 14: Light-curve of the example data set. Flaring high background periods are clearly visible.

   Determine a threshold on the light-curve counts, defining "low background" intervals (in our example: 1.0 cts/s) and create a corresponding good time interval (GTI) file (as `timebinsize=10` was chosen above, one needs to select `COUNTS<=10` for the threshold of 1.0 cts/s):

```
tabgtigen table=rates.fits expression='COUNTS<=10' gtiset=gti.fits
```

   Note: the recommended cut value for MOS observations is 0.35 cts/s but the choice of the thresholds strongly depends on the science the user is interested in.

2. Create an event list which is free of high background periods. Also this might be the place to restrict the further analysis to the well calibrated patterns and energy band:

```
evselect table=$evfile withfilteredset=true filteredset=filtered.fits \
         keepfilteroutput=true destruct=true \
         expression='(gti(gti.fits,TIME) && (PI in [100:15000]) && \
                     (PATTERN<=12))'
```

Create a new light curve to make sure that the flare was removed:

```
evselect table=filtered.fits withrateset=yes rateset=rates_new.fits \
         timecolumn=TIME timebinsize=10 maketimecolumn=yes \
         expression='#XMMEA_EP && (PI>10000) && (PATTERN==0)'
```

Note, in case of MOS #XMMEA_EM shall be used.

Plot the new light-curve (see Fig. 15):

```
dsplot table=rates_new.fits x=TIME y=COUNTS &
```

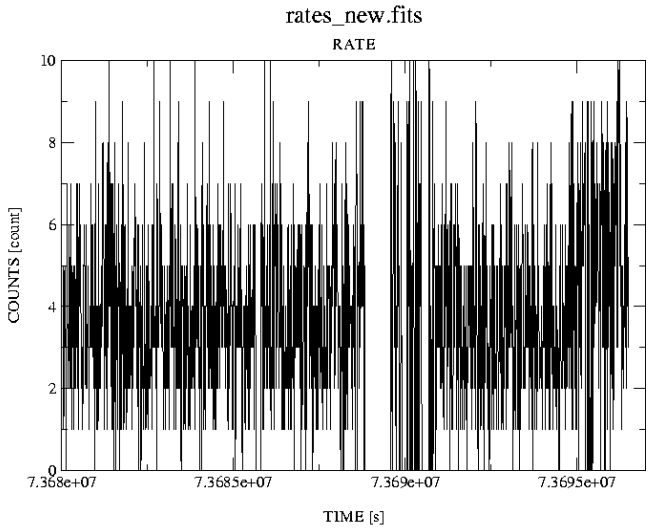


Figure 15: Light-curve of the example data set after removal of flaring high background periods.

From now on, the filtered events file (`filtered.fits`) will be used for further analysis. In case of the example data set, the number of events in the filtered event list is less the half of what the original event list had, a fact which will speed up further processing significantly. The size of the event list (especially in case of bright sources) can be further reduced a lot by also excluding the following columns: RAWX/Y, DETX/Y, PHA.

3. Create a sky image of the filtered data set:

```
evselect table=filtered.fits withimageset=true imageset=image.fits \
         xcolumn=X ycolumn=Y \
         imagebinning=binSize ximagebinsize=80 yimagebinsize=80
```

and display the image with `ds9`:

```
ds9 image.fits &
```

The parameter settings `imagebinning=binSize` and `x/yimagebinsize=80` bin the image into squared pixels of $80 \times 0.05 = 4$ arcsec (0.05 arcsec being the unit of the X, Y sky coordinate system).

Specifying an additional selection expression of the form `expression='PI in [..:..]'` allows creation of images in different energy bands. E.g. Fig. 16 shows the image in the energy range 0.5-7 keV.

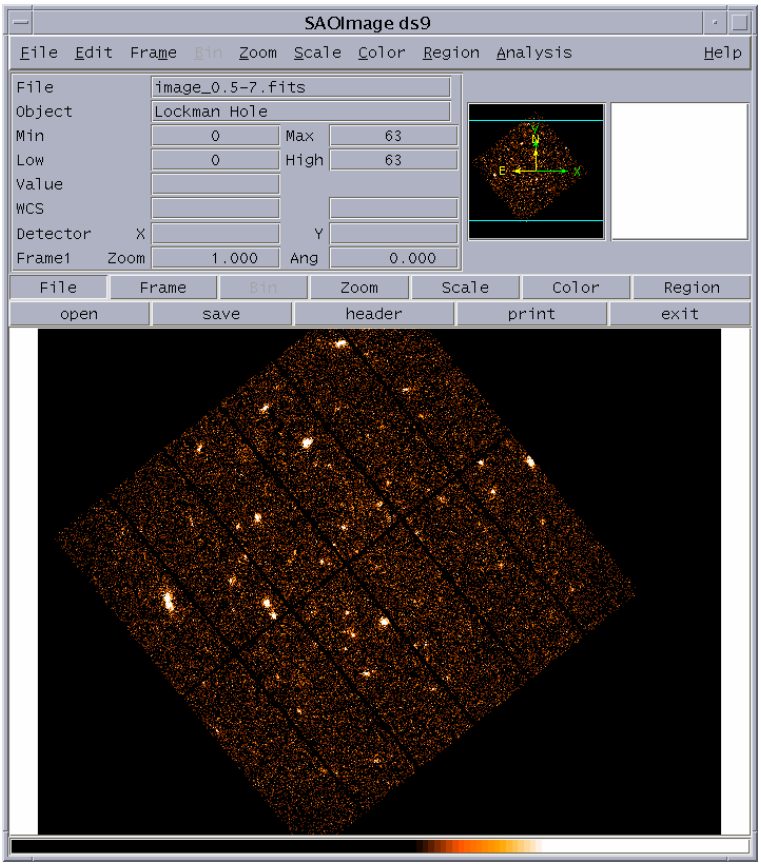


Figure 16: Sky image of the example data set in the energy range 0.5-7 keV displayed with `ds9`.

4. Extract source and background spectra for a bright point-like field source: in the displayed image a region around the source can be selected by positioning the cursor on the source center. Keeping the left mouse button pressed, the size of the circular extraction region can be changed. Clicking afterwards again on the region selects it. With the left mouse button the position and size of the region can interactively be changed. It is also possible to change characteristic region parameters via double clicking on the region as well as to save a region via the "Region/Save Regions..." menu in `ds9`.

   To apply the specified region as a spatial filter expression in `evselect`, the properties of the region (e.g. Fig. 17) (set "Coordinates → Image" to switch to image

coordinates) need to be converted into sky coordinates. To do this, coordinates of the center (329.251,286.249) and the radius (6.2514) need to be multiplied by the binning (which was chosen to be 80 in our example).

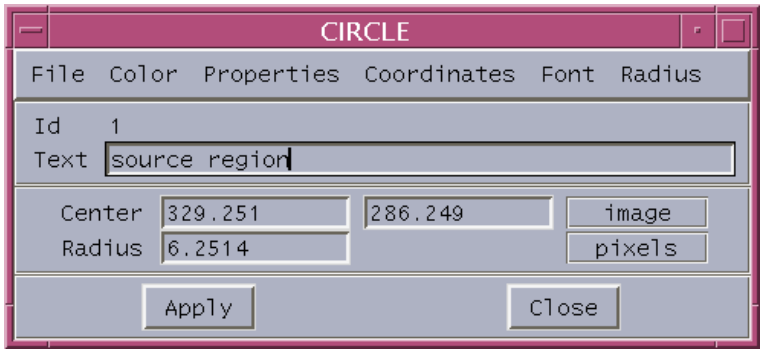


Figure 17: Selection region properties window, popped-up by double-clicking on the region in the main `ds9` window.

Hence the spatial selection expression for the source spectrum will be '(X,Y) IN circle(26340.1,22899.9,500.1)'. The extraction radius is 25 arcsec. Roughly these values would also be propagated into the selection expression by pressing the "2D Region" button in `xmmselect`.

The source spectrum is extracted with the following command line:

```
evselect table=filtered.fits withspectrumset=yes \
         spectrumset=spectrum.fits energycolumn=PI \
         withspecranges=yes specchannelmin=0 specchannelmax=20479 \
         spectralbinsize=5 \
         expression='((X,Y) IN circle(26340.1,22899.9,500.1)) && \
                     (FLAG==0) && (PATTERN<=4)'
```

Parameters `specchannelmax=20479` and `spectralbinsize=5` are the recommended settings for a pn data set. In the case of MOS data, they should be set to `specchannelmax=11999` and `spectralbinsize=15`. This setup allows the usage of the ready made response matrices as well.

Including `(FLAG==0)` in the selection expression is recommended for pn. This selection is even more restrictive than the `#XMMEA_EP` event attribute flag as it rejects, in addition, events which are close to CCD gaps or bad pixels. In the case of MOS, the filter expression `#XMMEA_EM` might be sufficient.

For pn we restrict the spectral analysis to the best calibrated single and double events (`(PATTERN<=4)`). In the case of MOS, all valid patterns (`(PATTERN<=12)`) might be included.

The source spectrum can be displayed with the following command:

```
dsplot table=spectrum.fits &
```

In the next step, one needs to extract a background spectrum:

In our pn example the source is located close to the edge of a CCD at a RAWY position of about 138. A surrounding annulus for the background extraction would include the CCD gap and eventually also part of a neighbouring CCD. Hence in this case it might be better to define the background via a circle on the same CCD where the source is located at roughly the same distance from the readout node (same RAWY as the source) placed in a source free region (see Fig. 18 for the selected source and background regions parameters):

```
evselect table=filtered.fits withspectrumset=yes \
         spectrumset=background.fits energycolumn=PI \
         withspecranges=yes specchannelmin=0 specchannelmax=20479 \
         spectralbinsize=5 \
         expression='((X,Y) IN circle(25260.3,21979.9,500.1)) && \
                     (FLAG==0) && (PATTERN<=4)'
```

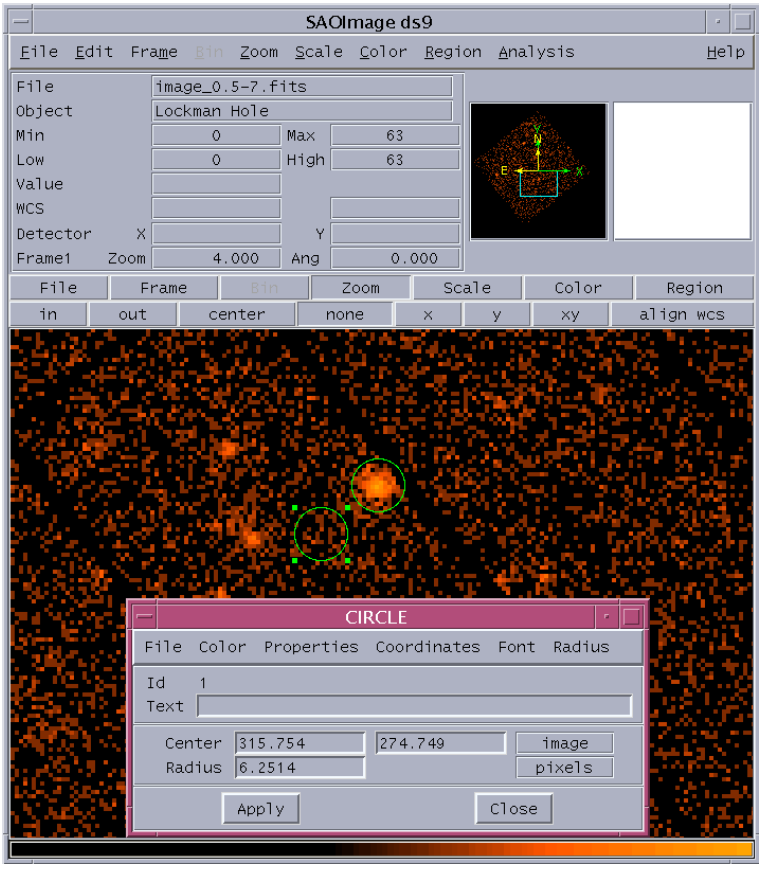


Figure 18: Selection regions for the extraction of source and background spectra.

The background spectrum can be displayed again with the following command:

```
dsplot table=background.fits &
```

In the next step, the area of the extraction regions used to make the source and background spectral files must be calculated to take into account CCD boundaries

and bad pixels. The area is written into the header of the SPECTRUM table of the input file as the keyword BACKSCAL:

```
backscale spectrumset=spectrum.fits badpixlocation=filtered.fits

backscale spectrumset=background.fits badpixlocation=filtered.fits
```

Note, if spectra are created via the `xmmselect` task, `backscale` will have automatically been applied "on the fly" during the product generation process.

5. Create response matrix files (Redistribution Matrix File (RMF) and Ancillary Response File (ARF)) using the `rmfgen` and `arfgen` tasks:

   ```
   rmfgen spectrumset=spectrum.fits rmfset=spectrum.rmf
   ```

   An alternative approach to obtain a RMF file is to use the ready-made "canned" response matrices available on the SOC web at the following URL:
   http://xmm.vilspa.esa.es/ccf/epic

   ```
   arfgen spectrumset=spectrum.fits arfset=spectrum.arf \
          withrmfset=yes rmfset=spectrum.rmf \
          badpixlocation=filtered.fits detmaptype=psf
   ```

   Note, `arfgen` reads the pattern range from the data subspace (DSS) information in the spectrum dataset, and accumulates the quantum efficiency curves over those patterns, which are then combined to the other constituents of the ARF. Be aware that the entire range of allowed patterns (0-12 for the pn and 0-31 for the MOS, respectively) are assumed if no pattern range was found in the DSS.

6. Prepare the spectra for further analysis with `Xspec`

   The created spectrum should be grouped (binned) now depending on the available signal to noise ratio in the data and the science the user wants to perform. The `FTOOL` task `grppha` allows regrouping of the source spectrum based on different criteria. Here we show an example of how to perform a grouping based on a minimum number of counts in each bin. In addition, `grppha` offers the possibility to associate the RMF, ARF and background spectrum with the source spectrum which is convenient to do before using `Xspec` for further analysis:

   ```
   grppha
   Please enter PHA filename[] spectrum.fits
   Please enter output filename[] spectrum.grp
   GRPPHA[] chkey BACKFILE background.fits
   GRPPHA[] chkey RESPFILE spectrum.rmf
   GRPPHA[] chkey ANCRFILE spectrum.arf
   GRPPHA[] group min 25
   GRPPHA[] exit
   ... written the PHA data Extension
   ...... exiting, changes written to file : spectrum.grp
   ```

Note, that after having read the `spectrum.grp` into `Xspec` the `ignore bad` command shall be issued to ignore spectral bins which did not fulfill the grouping condition used in `grppha`.

7. Create a light-curve for the source:

```
evselect table=filtered.fits withrateset=yes \
         rateset=lightcurve.fits timecolumn=TIME timebinsize=1 \
         expression='((X,Y) IN circle(26340.1,22899.9,500.1))'
```

The `Xronos` program package can now be used to produce a binned light-curve, to calculate a power spectrum, search for periodicities etc.

The data analysis based on command lines was described here, as combining command lines into a script might offer a good method for further re-performing of (part of) a data analysis session (but see also § 2.12.4 for a discussion of possible advantages of a GUI based analysis).

2.12.2.2 Inspection of spectra or timeseries using the command line

Although `xmmselect` interacts in a convenient way with the external plotting program `grace`, the user might want to plot, inspect or export spectra and timeseries which have been created in an already closed `xmmselect` session or via `evselect` from the command line. As the produced output files are FITS files, the `FTOOL` task `fv` and `fplot` offer many possibilities here, but also the SAS task `dsplot` can be used:

- The spectra or timeseries table can be plotted with the `grace` package using the `dsplot` command. We assume that the spectrum or timeseries are named `product.fits`:

```
dsplot table=product.fits
```

- Postscript files can be created.

```
dsplot table=product.fits plotter='gracebat -printfile product.ps'
```

- Spectrum and timeseries files can also be generated in ASCII format using the command `dstoplot`.

```
dstoplot table=product.fits > product.asc
```

2.12.3 Source detection example

In the following section (§ 2.12.3.1) we show an example on how to perform EPIC source detection calling in sequence the individual `edetect_chain` subtasks. This gives detailed information about the detection process. If, however, the user is happy with the default parameter settings for a standard source detection session, he/she might skip this section and continue directly with § 2.12.3.2 where the source detection is performed in one go via the `edetect_chain` task.

In the following we assume that the user has created a calibrated EPIC event list (or uses one from the pipeline processing) and has cleaned it for high flaring background periods. The assumed name of this filtered event list is `filtered.fits`.

The user can make use of the pipeline produced images (IMAGE_1 to IMAGE_5, see Table 2). Alternatively, images in these or different energy bands need to be generated first with e.g. the following command (see also § 2.12.2.1):

```
evselect table=filtered.fits withimageset=true imageset=image_b1.fits \
         xcolumn=X ycolumn=Y imagebinning=binSize ...\
```

in case of MOS, a binsize of 22 = 1.1 arcsec fits well the camera pixel size, all valid patterns should be included, and the MOS specific bit mask should be used:

```
         ... ximagebinsize=22 yimagebinsize=22 \
         expression='(PI in [200:500])&&(FLAG==0)&&(PATTERN in [0:12])&&#XMMEA_EM'
```

in case of the pn, a binsize of 82 = 4.1 arcsec fits well the camera pixel size, all single and double events should be included, and the PN specific bit mask should be used:

```
         ... ximagebinsize=82 yimagebinsize=82 \
         expression='(PI in [200:500])&&(FLAG==0)&&(PATTERN in [0:4])&&#XMMEA_EP'
```

To perform source detection in all the standard pipeline processing energy bands, four more images need to be generated, modifying the energy selection expression in the following way: for `image_b2.fits` use (PI in [500:2000]), for `image_b3.fits` use (PI in [2000:4500]), for `image_b4.fits` use (PI in [4500:7500]) and for `image_b5.fits` use (PI in [7500:12000]). The energy bands are defined via PI intervals in units of eV. Note that such a multi-band approach to source detection is not essential (it could also be performed in a single energy band).

With images extracted in the required energy bands, the user now can start the EPIC source detection process, either performing it step by step (§ 2.12.3.1) or in one go via the `edetect_chain` metatask (§ 2.12.3.2).

2.12.3.1 EPIC source detection performed via single task commands

The user can further make use of the pipeline produced exposure maps (EXPMAP1 to EXPMAP5, see Table 2). Alternatively, exposure maps in different energy bands need to be generated with the `eexpmap` task. This task makes use of calibration information on the spatial quantum efficiency, filter transmission, mirror vignetting and field of view to calculate for each attitude bin the exposure time values projected onto the sky. Assuming that the attitude information is available from the file `attitude.fits` (created by the `atthkgen` task; alternatively, the attitude file exists as a pipeline product with file identification ATTTSR), exposure maps `image_biexp.fits` (i = 1 to 5) are generated in the following way:

```
eexpmap attitudeset=attitude.fits eventset=filtered.fits imageset=image.fits \
        expimageset='image_b1exp.fits image_b2exp.fits image_b3exp.fits \
                     image_b4exp.fits image_b5exp.fits' \
        pimin='200 500 2000 4500 7500' pimax='500 2000 4500 7500 12000'
```

The imageset `image.fits` can be any of the already created energy band images. It is only used to extract information from the FITS header about the Instrument ID, Mode/Submode, filter ID, GTI and WCS keywords.

In the next step, a detection mask needs to be generated. The detection mask is a FITS image containing the integer values 0 and 1 where 1 marks the image area on which subsequent source searching will be performed. The following command should be performed:

```
emask expimageset=image_biexp.fits detmaskset=mask.fits
```

The expimageset `image_biexp.fits` can be any of the previously created exposure maps. The user should inspect the created mask with `ds9` to see if the area selection is appropriate for further source detections. If not, the values of the threshold parameters `threshold1` and `threshold2` can be tuned. By default, the detection mask contains 0 where the value of the exposure map is less than 30% of the maximum exposure (`threshold1`) and where the gradient of the exposure map is steeper than 0.5 (`threshold2`).

Now all necessary files have been generated to start the sliding box source detection in the so-called local mode. The purpose of the local detection step is to provide an input list of source positions for task `esplinemap` (see below) which then constructs a background map from the non-source locations. Source counts are accumulated from a 3×3 or 5×5 (controlled by parameter `boxsize`, default value = 5, also used in the SSC pipeline) pixel window and the background is determined from the surrounding 40 (7×7 pixel window) or 56 pixels (9×9 pixel window), respectively. Detection of moderately extended objects (up to several times the PSF size) is achieved by searching the image in up to four consecutive detection runs each doubling the pixel size (parameter `nruns`, default value = 3).

Following the definition which was, e.g., used by the ROSAT mission, detection likelihoods (per energy band and total) are given for each source in the form $L = -\ln p$ where p is the probability of Poissonian random fluctuation of the counts in the detection cell which would have resulted in at least the observed number of source counts. The value of p is calculated as a function of raw source counts and raw background counts in the detection box (see `eboxdetect` task description for further info on the detection algorithm).

The local mode sliding box source detection is performed executing the following command:

```
eboxdetect usemap=no likemin=8 withdetmask=yes detmasksets=mask.fits \
           imagesets='image_b1.fits image_b2.fits image_b3.fits \
                      image_b4.fits image_b5.fits' \
           expimagesets='image_b1exp.fits image_b2exp.fits image_b3exp.fits \
                         image_b4exp.fits image_b5exp.fits' \
           pimin='200 500 2000 4500 7500' pimax='500 2000 4500 7500 12000' \
           boxlistset=eboxlist_l.fits
```

The local mode detection is switched on setting `usemap=no`. It is recommended to use a detection threshold of `likemin=8` to provide a complete source list as input for `esplinemap`. Note, the parameter `ecf` will need to be specified if one already at this stage of the source detection chain is interested in the computation of source fluxes (see section on `emlselect` below).

In the next step, the task `esplinemap` makes use of the `eboxdetect` generated local mode source list to derive spline background maps from the non-source regions. Sources found in

the local detection step at significance levels (column SIGMA of the source list) exceeding a user-specifiable threshold (input parameter `mlmin`) are removed from the image using a suitable PSF and source brightness dependent cut-out radius (determined to be the radius at which each source contributes more than a user-specifiable number of counts/arcsec^2 to the background; parameter `scut`):

```
esplinemap bkgimageset=image_b1bkg.fits imageset=image_b1.fits \
           boxlistset=eboxlist_1.fits scut=0.005 nsplinenodes=16 \
           withdetmask=yes detmaskset=mask.fits \
           withexpimage=yes expimageset=image_b1exp.fits
```

This commands needs to be performed for all energy bands (b1 to b5) separately. An optionally diagnostic output photon image where sources have been masked out (the so called cheesed image) can be created setting `withcheese=yes` and defining the name of the cheesed image via `cheeseimageset`. In addition, `esplinemap` is able to determine the background caused by OoT events registered during the readout process of the pn CCDs (§ 2.6). If the flag `withootset` is set, the photon event table specified in `ooteventset` is read and the background caused by OoT events is included in the output background map.

To improve the detection sensitivity reached before with the local mode detection, the sliding box source detection now should be performed in map mode (`usemap=yes`). Here the background will be taken from the background maps that were determined by `esplinemap`:

```
eboxdetect usemap=yes likemin=8 withdetmask=yes detmasksets=mask.fits \
           imagesets='image_b1.fits image_b2.fits image_b3.fits \
                      image_b4.fits image_b5.fits' \
           expimagesets='image_b1exp.fits image_b2exp.fits image_b3exp.fits \
                         image_b4exp.fits image_b5exp.fits' \
           bkgimagesets='image_b1bkg.fits image_b2bkg.fits image_b3bkg.fits \
                         image_b4bkg.fits image_b5bkg.fits' \
           pimin='200 500 2000 4500 7500' pimax='500 2000 4500 7500 12000' \
           boxlistset=eboxlist_m.fits
```

The next step is to generate the final source list with the task `emldetect`: a simultaneous maximum likelihood PSF fit is performed to the source count distribution in all energy bands of each involved EPIC telescope with the following free parameters: source location (RA, Dec), source extent (Gaussian sigma) and source count rate. Source location and extent are constrained to the same best-fit value in all energy bands whereas count rates are the best-fit values in each band. The PSF fitting may either be performed in single source (default) or in multi-source mode (parameter `nmaxfit` > 1). Unless `nmaxfit` > 1, `emldetect` will not detect new sources, but characterizes the detected sources by making a PSF fit. Energy conversion factors (ECFs) can be supplied for a conversion of source count rates into flux values. The ECFs for each energy band depend on the pattern selection and the filter used during the observation (see SSC-LUX-TN-0059). Here we are assuming a MOS thin filter observation where patterns 0 - 12 are considered:

```
emldetect imagesets='image_b1.fits image_b2.fits image_b3.fits \
                     image_b4.fits image_b5.fits' \
```

```
expimagesets='image_b1exp.fits image_b2exp.fits image_b3exp.fits \
              image_b4exp.fits image_b5exp.fits' \
bkgimagesets='image_b1bkg.fits image_b2bkg.fits image_b3bkg.fits \
              image_b4bkg.fits image_b5bkg.fits' \
pimin='200 500 2000 4500 7500' pimax='500 2000 4500 7500 12000' \
boxlistset=eboxlist_m.fits \
ecf='1.8035 1.9872 0.76318 0.26770 0.029085' \
mlmin=10 mllistset=emllist.fits
```

Especially in cases where an expected X-ray source could not be detected by the source detection steps described above, the user might be interested in using task `esensmap` to generate a sensitivity map giving (rough) point source detection upper limits (vignetting corrected source count rate corresponding to the likelihood of detection as specified in the parameter file) for each image pixel. The task `esensmap` may either be called for individual energy bands or combinations of energy bands and instruments. The upper limits are calculated by assuming Poissonian count statistics in each 3×3 pixel detection cell, using the exposure and background values read from the input images. In the case of multiple input energy bands the upper limits are expressed in units of counts per seconds for the combined band (i.e. they refer to the detection sensitivity which would be achieved by adding up the photons observed in the individual bands). Below we give an example on how to compute a sensitivity map for a single energy band:

```
esensmap expimagesets=image_b1exp.fits bkgimagesets=image_b1bkg.fits \
         detmasksets=mask.fits mlmin=10 sensimageset=image_b1sen.fits
```

This command eventually should be performed for all energy bands (b1 to b5) separately.

Alternatively, if one is interested in a sensitivity map for the combined energy range, the command to be issued could be:

```
esensmap expimagesets='image_b1exp.fits image_b2exp.fits image_b3exp.fits \
                       image_b4exp.fits image_b5exp.fits' \
         bkgimagesets='image_b1bkg.fits image_b2bkg.fits image_b3bkg.fits \
                       image_b4bkg.fits image_b5bkg.fits' \
         detmasksets=mask.fits mlmin=10 sensimageset=image_sen.fits
```

2.12.3.2 Running the EPIC source detection chain

Assuming that the user either wants to make use of the pipeline generated image products or that he/she has created images in different energy bands, the task `edetect_chain` can be called to perform all source detection steps described in § 2.12.3.1 via a single command.

The attitude information is assumed to be available from the file `attitude.fits` (created by the `atthkgen` task). The example below further assumes that the source detection will be performed on a background cleaned pn thin filter observation (the name of the calibrated and filtered event list being `filtered.fits`). As noted above, the ECFs for each energy band depend on the EPIC camera, the pattern selection and the filter used during the observation (see SSC-LUX-TN-0059).

The complete EPIC source detection chain is started with the following command:

```
edetect_chain eventsets=filtered.fits attitudeset=attitude.fits \
              imagesets='image_b1.fits image_b2.fits image_b3.fits \
                         image_b4.fits image_b5.fits' \
              pimin='200 500 2000 4500 7500' pimax='500 2000 4500 7500 12000' \
              ecf='10.596 6.816 2.054 0.995 0.259' \
              eboxl_list=eboxlist_l.fits eboxm_list=eboxlist_m.fits \
              esp_nsplinenodes=16 eml_list=emllist.fits esen_mlmin=10
```

After completion of the `edetect_chain` task, the following output files have been created: exposure maps (from task `eexpmap`), detector mask images (from task `emask`), background maps (from task `esplinemap`), `eboxdetect` source list (local mode), `eboxdetect` source list (map mode), `emldetect` source list, source maps, if `eml_withsourcemap=yes` (from task `emldetect`) and sensitivity maps (from task `esensmap`).

To verify the quality of the performed source detection, the user might want to check generated maps with the image viewer program `ds9`. In addition, the task `srcdisplay` can be used to overlay all the different derived source lists onto an image. Circles are used to depict the source positions. The radius of these circles can be set (in degrees) using the `sourceradius` parameter. An optional ID label can also be displayed, corresponding to the row number of that source in the input source list. This can be enabled through the `uselabel` parameter. Plotting the ID helps the user to refer back to source properties documented in the source list.

As the plotted circles are in fact `ds9`-type regions, they can be written out to a file for future use (for example, when running a later `ds9` session) by setting `withregionfile` to true, and specifying the desired file name via the `regionfile` parameter (set to `regionfile.txt` by default).

To show e.g. the generated merged source list overlaid onto an EPIC image (called `pn_image` in the example) the following command can be issued:

```
srcdisplay boxlistset=emllist.fits imageset=pn_image sourceradius=0.01
```

Fig. 19 shows an example of such a `srcdisplay` run.

2.12.4 Working with the GUI

The user can perform the described analysis (instead of typing commands) also with the graphical user interface (GUI) of the SAS. In general, the analysis via the GUI might be judged more user friendly for an interactive analysis session. But the GUI, too, allows the saving of performed analysis steps in a log file (default name "sas_log") as well as the saving of performed commands as a shell script (via the "File: Save as script" menu) which later can be used for reprocessing.

For that purpose, the `sas` command shall be run first. The SAS GUI appears and the user can select a task. The `xmmselect` task interacts in a convenient way with the `evselect` task to extract images, spectra, timeseries or filtered event lists.

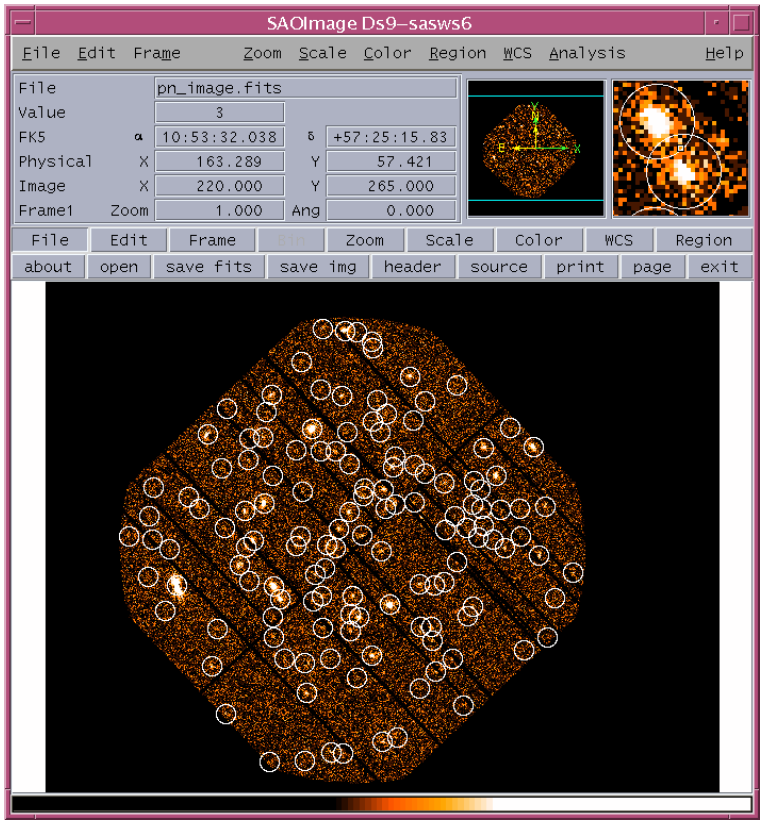


Figure 19: Resulting ds9 display of a `srcdisplay` command issued to overlay a final source list onto a pn image.

3 Analysis of OM optical monitor data

3.1 The OM data

3.1.1 OM observing modes and data types

The OM can be operated in two basic science modes: image and fast mode. These modes can be used separately or combined, in different instrument configurations:

- Default modes: image_only and image+fast. These are predefined window configurations covering nearly 92% of the field of view with five exposures with/without one fast mode window centered at the boresight location.
- User defined image and/or fast mode windows: up to 5 windows (2 in fast mode) can be defined in pixels or in (RA, Dec)
- In addition, the OM detector can be operated in full frame mode to obtain images of the entire FOV, either at high resolution (0.5 arcsec/pix, 2048×2048 format) or at low resolution (1 arcsec/pix, 1024×1024 format).

In the windowed modes, the detector windows are re-centered to correct for pointing errors (Field Acquisition, FAQ). Also a shift_and_add mechanism is used to compensate for spacecraft tracking stability. Full frame modes do not have these capabilities.

These observing modes determine the type of data obtained with the OM: 2-D images for image modes and Event lists if fast mode is used.

The default modes mentioned above will produce the following data types. Since each exposure has associated two image windows, there will be two image files per exposure which correspond to a small high resolution window at the centre of the FOV and a large low resolution one placed at different locations in each exposure. If the default included fast mode, there will be an additional event list file corresponding to the fast window.

In the user defined windows configuration there will be one image file or event list corresponding to each of the windows.

The full frame images consist of a single image file if high resolution was used and four adjacent image files in case of low resolution. (It should be noted that the four image files correspond to a unique exposure).

In addition to these image and/or event list files, for each exposure there will be several auxiliary files containing instrument configuration parameters. There will be also files containing OM house-keeping data for the whole observation. All these files together with the spacecraft attitude files, common for all instruments, constitute the ODF for OM data.

Table 4 contains a summary of the possible OM files contained in an ODF (for one exposure).

OM images files have a file name ending with a "IMI.FIT". There is one imaged file per image mode window per exposure. Raw image files can be displayed with e.g. the ds9 image viewer. It is also possible to combine all processed image windows of an OM default configuration before viewing them using the SAS task `ommosaic`.

OM fast mode event lists have a file name ending with "FAE.FIT". There is one dataset for each fast window. Inspection of the data files using e.g. the ftool viewer fv is possible.

File name	Description of the file content
0001-0000000000-OMS00100IMI.FIT	image window 1
0001-0000000000-OMS00101IMI.FIT	image window 2
0001-0000000000-OMS00102FAE.FIT	event list (if fast mode was used)
0001-0000000000-OMS00100THX.FIT	tracking history
0001-0000000000-OMS00100WDX.FIT	priority window data
0001-0000000000-OMS00100PFX.FIT	priority fast mode data
0001-0000000000-OMS00100RFX.FIT	priority reference frame data
0001-0000000000-OMS40000PAX.FIT	field acquisition data
0001-0000000000-OMX00000PEH.FIT	periodic house-keeping
0001-0000000000-OMX00000NPH.FIT	non-periodic house-keeping

Table 4: ODF files associated with a single OM exposure.

OM reference frame data have a file name ending with a "RFX.FIT". Generally one file is present for each exposure, except if tracking was switched off. This is the case for certain OM engineering modes and certain filter elements (e.g. grism).

OM tracking history file data have a file name ending with a "THX.FIT". Generally one file is present for each exposure, except when tracking was explicitly switched off, or when no suitable stars for tracking were found in the reference frame at the beginning of the exposure. Inspection of this files with e.g. the ftool viewer fv is possible. Plot of column DX and DY show the attitude stability during the exposure. The PPS product file whose name ends as " TSHPLT.PDF" provides a visualiation of the pointing stability during the exposure.

OM field acquisition data have a file name ending with a "PAX.FIT". There is only one field acquisition dataset present per observation. In order to get a feeling on the absolute pointing accuracy, the user can look at this field acquisition datatset (PAX.FIT). The paramaters DX, DY provide the absolute pointing error in units of 0.001 subpixel.

OM priority window data have a file name ending with a "WDX.FIT". One priority window dataset is present for each exposure. This file is used, e.g. to compute the exposure dead-time fraction.

OM priority fast mode data have a file name ending with a "PFX.FIT". There is one dataset for each exposure containing fast mode window(s).

3.1.2 Listing the OM Current Calibration Files

Table 5 provides a list of the OM calibration files. For a detailed description of these files and of their usage, the user is refered to the CCF interface document and to the Calibration Access and Data Handbook (see § 1.3). Some of these files are only used by the PHS and QLA systems, although they are contained in the SAS file ccf.cif. The SAS tasks using a given CCF are indicated as well.

File name	File Content	Usage
ASTROMET	coefficients for geometric distortion correction	`omatt`
BADPIX	bad pixels position, type of defect and the severity level	`omcosflag`
BORESIGHT	alignement of the instruments and star tracker	`omatt`
COLORTRANS	coefficients for color transformation into a standard system	`ommag`
HKPARMINT	house keeping parameter ranges	`ommag`
LARGESCALESENS	flat field map (set to unity)	`omflatgen`
LINCOORD	parameters for calculations of instrument configuration	not used
PHOTTONAT	correction coefficients of count rates for detector non-linearity and aging	`ommag`
PIXTOPIXSENS	flat field map (set to unity)	`omflatgen`
PSF1DRB	point spread function for each filter	`ommag`
DARKFRAME	dark current map	QLA
DIFFUSEGALA	intensity map of diffuse galactic emissions	PHS
QUICKMAG	a rough count to magnitude conversion table for different spectral types	QLA
ZODIACAL	intensity map of the zodical light and average spectrum	QLA

Table 5: Calibration files of the Optical Monitor.

3.2 Description of the OM image data processing chain `omichain`

The processing of OM data is run in pipeline mode at the SSC in Leicester. Two processing chains, `omichain` and `omfchain`, are used for image and fast mode data respectively. The process can be repeated by the users by running the same chains at their home institute. The chains are Perl scripts which concatenate the individual SAS tasks so as to go from the input raw data up to the final output results without further interaction. The tasks can also be run one by one having more control on all needed parameters.

The whole processing of image and fast mode data is depicted in Fig. 20 and Fig. 21.

In order to process the OM image data successfully an input dataset is required. Input datasets are not changed during the OM processing with `omichain`. They are copied to intermediate datasets and keywords. Extensions are added as needed. Intermediate task products, which are passed from one task to another, are generally not described.

The processing of OM image data can be divided in three parts (see Fig. 20)

- **Data preparation.** Before real processing, the four files composing a full frame low resolution exposure are combined into a single one using `omcomb`. Also a flat-field for later usage is obtained with `omflatgen`. (See note about OM flat-field at the end of this section)

- **Image processing.** All corrections, source detection, astrometry and photometry are applied to each image file, exposure by exposure. `omichain` will process automatically all image data consecutively. However, the user can process single image files by applying the different tasks one by one in a semi-interactive way.

  This core of the image processing can be divided as well

  - **Preparation of tracking corrections**
  - **Corrections: bad pixels, fixed pattern, flat-field**
  - **Source detection, astrometry and photometry**

- **Combined results.** The source lists corresponding to different exposures, using different filters, are combined with `omsrclistcomb`. Colour indices are also calculated. The images corresponding to different windows observed with the same filter are mosaiced in a single image file using `ommosaic`.

As it can be seen in the flow chart of the OM processing chain (Fig. 20), the OM tracking information for each exposure is treated before and independently from the image data processing. In the following, for each task description, a reference is given to the analysis step in the processing example.

3.2.1 Data preparation

- `omcomb` is used before real processing, to combine the four files composing a full frame low resolution exposure into a single one which will be the subsequent input for the process.

- `omflatgen` generates the flatfield file for the observation. The flatfield file is distributed as a PPS product (FLAFLD). In absence of a FLAFLD.FIT file in the PPS products, a dummy file can be generated by the task `omflatgen`. In this case the flatfield response is set to unity.

It should be noted that the characteristics of the OM detector make that flat field correction has to be treated in a diferent way than in a normal CCD. The physical pixels are subsampled by the centroiding algorithm, thus the concept of pixel to pixel variations changes since the real pixel cannot be recovered. A fixed pattern appears instead and it is corrected at a later stage in the processing.

It has been shown by studying the accuracy of the OM photometric calibration, which is based in observations of very many stars with OM and from the ground, and the statistical errors of the measurements that the results obtained with OM are accurate within a few percent without flat field correction. Furthermore, observations of the same stars in different parts of the detector show that large scale sensitivity variations are smaller than two per cent. Therefore the SAS uses a unity flat field file for processing OM data. Studies of flat field correction in OM are being done to further demonstrate the validity of this approach.

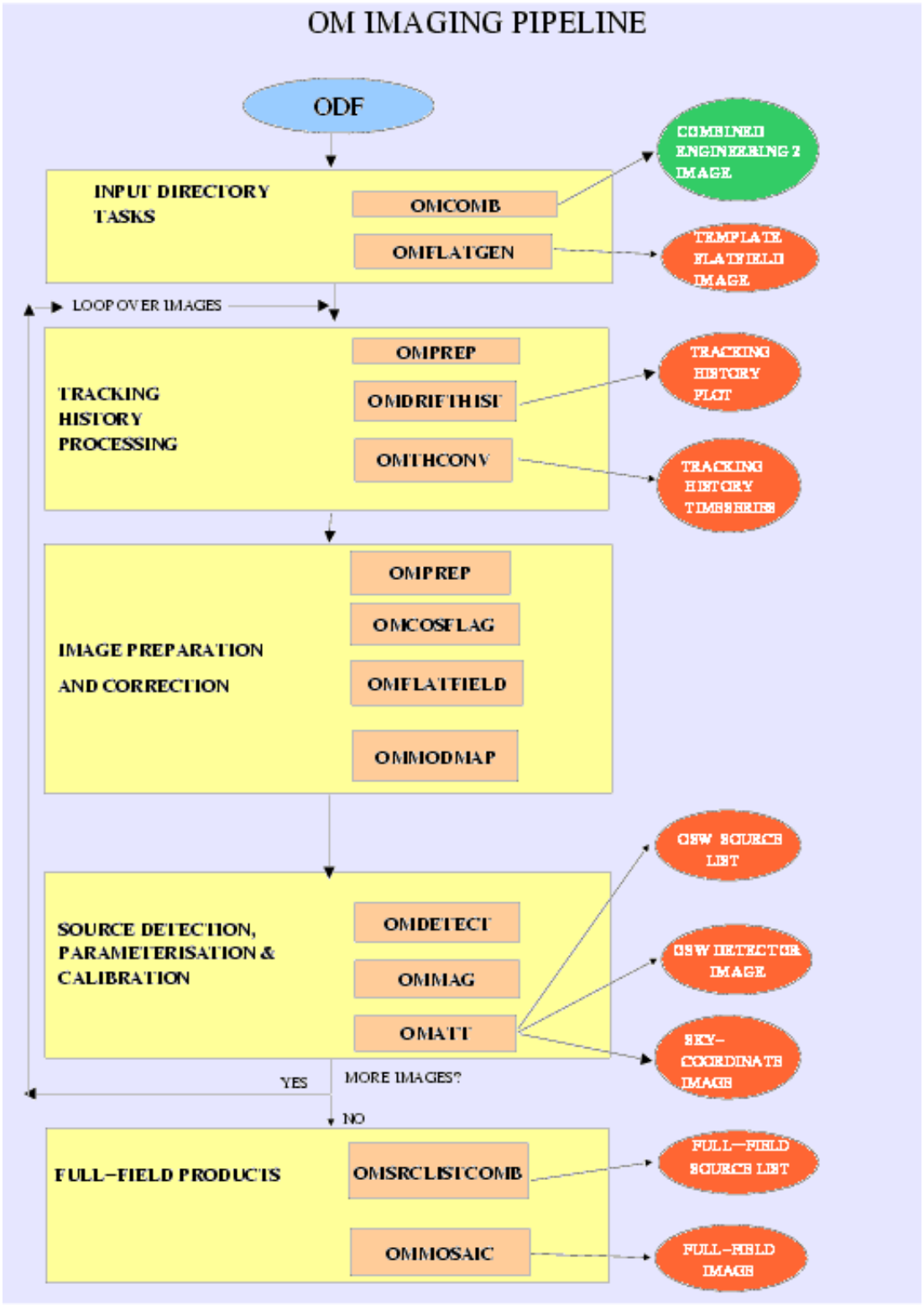


Figure 20: Pipeline processing of data acquired in the OM imaging mode.

3.2.2 Handling of tracking data

Two output dataset related to tracking data are distributed as pipeline products, namely the TSHPLT.PDF and the TSTRT.FIT file. They are created as follows.

- `omprep` (step1). The task `omprep` does not generate final data products but only intermediate files. OM images, tracking history files and fast mode files if any, are copied into temporary files. Further information is added via keywords. Intermediate products are generated, which carry information extracted form the tracking history file. If no tracking history file was produced, these are generated on the fly assuming perfect stable pointing. Tracking history data can be missing if no suitable tracking stars were available during observation. This can also occur when certain instrument modes (engineering modes) or when some filter elements (e.g. grisms) are used.
- `omdrifthist` (step2). This task creates the TSHPLT.PDF file, which provides an overview of the pointing stability during an exposure. The file contains diagrams illustrating the absolute and incremental OM drift.
- `omthconv` (step3). This task creates the TSTRTS.FIT file, which contains the tracking star time series, i.e the average count rate (cts/s) per tracking frame for each tracking star.

3.2.3 Handling of corrections applied to image mode data

As pointed out before, an observation with OM may be composed of several exposures which contain different observed windows. There are three final data products per observed science window (OSW), namely IMAGE_.FIT, SIMAGE.FIT and SWSRLI.FIT. Two additional products FLAFLD.FIT and OBSMLI.FIT. are generated on a per observation basis. The task `omichain` loops over all image mode data files of all exposures in an observation. The following tasks are applied to each individual observation window.

- `omprep` (step4). The task `omprep` copies the raw images into an intermediate output file. It adds several keywords to this intermediate output file. Although this file is not a final data product, it is carried forward within the processing of OM image data and is eventually contained within the final IMAGE_ product.
- `omcosflag` (step5). The task `omcosflag` generates a quality map. Bad pixels are flagged taking also into account the tracking history information. The quality array (bad pixels are flagged as non-zero there) itself is written into the quality extension of the intermediate image files. It is used later by other tasks and it is propagated eventually into the IMAGE_.FIT PPS output file.
- `omflatfield` (step6). The task `omflatfield` applies the tracking history file (as contained in the intermediate processing product of the tracking history task) to the OM flatfield file (FLAFLD). The resulting drift corrected flatfield file is applied to the intermediate OM image file. The flatfield corrected OM image is written as an intermediate data product, which is used as an input to `ommodmap`. Since the current flat field is unity, this correction has no effect on the data.

- `ommodmap` (step7). The task `ommodmap` applies the modulo 8 correction to an OM image. The modulo 8 background fluctuations is introduced to an OM image by imperfections of the photon centroiding algorithm. An intermediate image file is generated, which acts as input to the task `omdetect`.

3.2.4 Source detection, astrometry and photometry

- `omdetect` (step8). The task `omdetect` applies a source detection algorithm to find the sources present in the image. First the background is modelled. Then the algorithm looks for sources matching the criteria defined by the task parameters. Aperture photometry is applied to the detected sources. The output is a source list (SWSRLI) which contains the source position in absolute detector pixel and the calculated source and background count rates (cts/s). An optional output file called the region file can be generated if the parameter outputregionfile is set to true. The region file describes the area associated with each identified source in the image. It is used by the image display tool.

- `ommag` (step9). The task `ommag` converts count rates into the instrumental OM photometric system. Count rates are corrected for coincidence losses and detector deadtime. The corrected count rates are converted into instrumental magnitudes. The task requires input from the source list file SWSRLI and the priority window files (WDX). The output is added as extra column to the source list file (SWSRLI). The sensitivity limit of the specific exposure is written as a keyword in the source list.

- `omatt` (step10). The task `omatt` reads the intermediate image generated by ommodmap and reads the source list file generated by `omdetect` and `ommag`. It converts detector position from pixel coordinates into ra/dec positions based on information from the attitude history file, the instrument boresight file and the OM filter distortion map. The celestial position are added as two new columns to the SWSRLI.FIT file. A skyimage (SIMAGE.FIT) is generated from the input image, by resampling the image, applying the distortion correction and north aligning the image. The skyimage provides the most accurate positional information of an image in the OM image data processing chain. `omatt` also adds WCS to the input flatfielded and mod8 corrected OM image which is renamed into IMAGE_.FIT by `omichain` and is delivered as a PPS product. Note that this IMAGE_ has no distortion correction and therefore its positional information is less accurate than in the SIMAGE.

3.2.5 Final combined results

- `omsrclistcomb` (step11). The source lists of the different OSW's of all exposures contained in an observation are combined into one single observation slource list file using `omsrclistcomb`, which is invoked when the analysis of the invidual science windows is finished. The input parameter "nsigma" defines the criteria for the spatial matching of two sources in the merging of the source lists. `omsrclistcomb` generates a master source list with ra/dec coordinates, ra/dec errors and galactic coordinates. The significance of detection, the instrumental magnitude and its error, the semi-axis of the source detection ellipse and a set of flags are written in the source

list for each filter used in the observation. The colour transformations are applied and the source brightness is also listed in a standard photometric system.

- `ommosaic` (step 12). In case of multiple OM exposures of the same field, `ommosaic` is used to combine the corresponding skyimages into a single skyimage covering the full field of view. This task is used for the default image mode, to combine the windows of the five exposures obtained with the same OM filter, but it can also be used to mix windows observed with different filters.

3.2.6 Further notes on processing of OM image data

With the default settings, `omichain` overwrites temporary files at several stages, e.g when looping over multiple exposures or even within the analysis of a single window. This can be avoided by specifying unique output file names. It is recommended to clear out (removing or renaming) products of previous `omichain` runs before re-invoking `omichain`.

3.3 Description of the OM fast mode data processing chain `omfchain`

The example input dataset given in Table 4 contains also fast mode data files. To process them with SAS we have to use the task `omfchain`. As for image data, the input data do not change through the fast mode chain.

The process is similar to the one of image data. It can be followed in Fig. 21. For each fast mode window of the different exposures in the observation (ODF) there will be

- **Data preparation: tracking correction.** The tracking data, which are common for image and fast mode data, are treated to derive the corrections to be applied to the events detected in an exposure.
- **Event selection and corrections.** The event list file is scanned through to find the photon events. Then tracking and flat field corrections are applied.
- **Source detection and astrometry.** The source is located within the fast mode window (eventually more than one sources) and astrometry conversions are applied.
- **Light curve.** Count rates are derived for the source(s) and the background. Then a light curve is produced giving the net countrate as a function of time for each fast window in each exposure. A graphical output is produced as well. In the future all light curves obtained with the same OM filter will be merged.

3.3.1 Data preparation: tracking correction

The same tasks used in the image chain are repeated here:

- `omprep` (step1). The task `omprep` does not generate final data products but only intermediate files. OM images, tracking history files and fast mode files if any, are copied into temporary files. Further information is added via keywords. Intermediate products are generated, which carry information extracted form the tracking

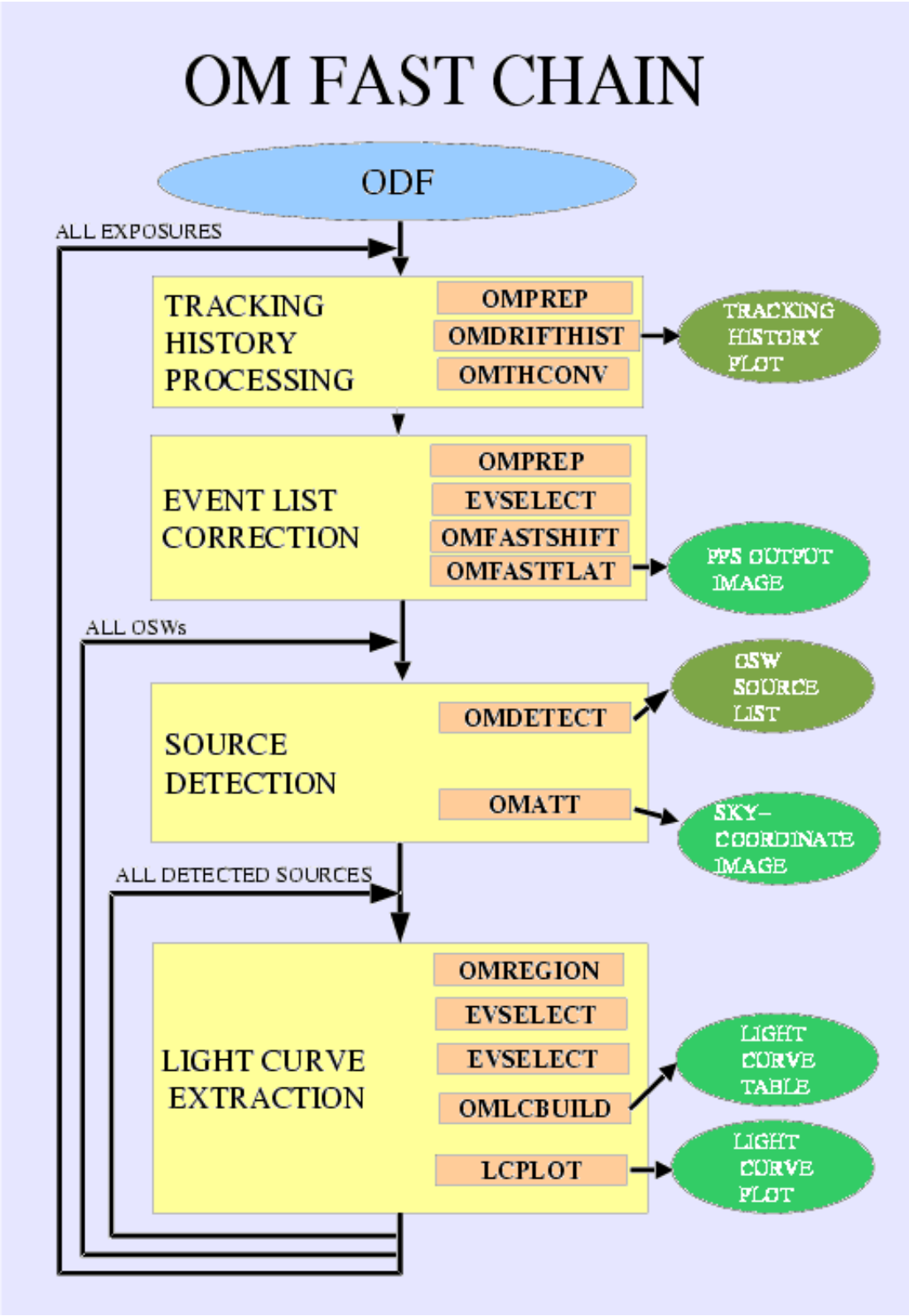


Figure 21: Pipeline processing of data acquired in the OM fast mode.

history file. If no tracking history file was produced, these are generated on the fly assuming perfect stable pointing. Tracking history data can be missing if no suitable tracking stars were available during observation. This can also occur when certain instrument modes (engineering modes) or when some filter elements (e.g grisms) are used.

- `omdrifthist` (step2). This task creates the TSHPLT.PDF file, which provides an overview of the pointing stability during an exposure. The file contains diagrams illustrating the absolute and incremental OM drift.
- `omthconv` (step3). This task creates the TSTRTS.FIT file, which contains the tracking star time series, i.e the average count rate (cts/s) per tracking frame for each tracking star.

3.3.2 Event selection & corrections

The following tasks are applied to each individual fast mode window

- `omprep` (step4). The task `omprep` applied to the event list file will augment its header contents with house-keeping information. A modified intermediate event list file will be created.
- `evselect` (step5). The task `evselect` extracts the photon events from the event-list file and builds a pseudo image corresponding to the fast mode window.
- `omfastshift` (step6). The task `omfastshift` uses the tracking history file (as contained in the intermediate processing product of the tracking history task) to correct the position of the photon events for drifts of the spacecraft during the exposure. New columns are added to the event-list file with corrected coordinates.
- `omfastflat` (step7). The task `omfastflat` applies the tracking history file to the OM flatfield file (obtained from `omflatgen`) to build a shifted flat field corresponding to the fast mode pseudo image.

  We point out again the peculiarity of the flat field for OM. As for image data, a unity flat field is applied to fast mode data as well.

3.3.3 Source detection & astrometry

- `omdetect` (step8). The task `omdetect` applies the source detection algorithm to the fast mode pseudo image to locate the existing sources. It builds a source list similar to the image data one (SWSRLI) which contains the source position in absolute detector pixel and the calculated source and background count rates (cts/s).
- `omatt` (step9). The task `omatt` performs astrometry conversions from pixel positions to astronomical coordinates in the source list and rotates the pseudo image to obtain a north aligned skyimage.
- `omregion` (step10). The task `omregion` produces the regions for the source and background needed later for `evselect`

- `evselect` (step11-12). The task `evselect` is run twice consecutively on the event-list file to extract the photon events for the source and the background in the specified time sampling. Intermediate rate files are created for the source and the background.
- `omlcbuild` (step13). The task `omlcbuild` generates coincidence loss and dead-time corrected source timeseries using the source and background rates produced by `evselect`. The background rates are substracted and the photometry corrections are applied. If the source is offset from the centre, the knowledge of the PSF is used to account for possible missing photons. A light-curve is obtained as final result for each of the fast mode windows in the exposure.
- `lcplot` (step14). The task `lcplot` reads the ligh-curve file and makes some plots. Some statistics is also applied to assess possible source variablity.

3.4 OM pipeline products

This section provides a general description of the final pipeline products which are distributed to the user. Similar files are generated by running the chain scripts, but the user shall be aware that they may have subtle differences compared to the pipeline products. Intermediate files are deleted in the pipeline processing and also when the `omichain` or `omfchain` are used. By running the tasks interactively, the user can maintain these intermediate products for a better understanding of the whole process.

OM pipeline products are named as "PooooooooooOMueeetttttsxxx.fff", where:

- oooooooooo stands for the observation ID,
- u is the exposure flag (S:scheduled, U:unsched, X:not applicable)
- eee is the exposure ID,
- tttttt is the data type,
- s is the OM window within the exposure
- xxx is the source number within the window
- fff is the file type (FIT, FTZ, PDF, PNG etc.)

In the following we refer to a data type by the tttttt identifier, e.g. such as tttttt= SIMAGE for the OM sky image product.

There are two pipeline products associated with the tracking history files, namely the TSHPLT.PDF and the TSTRTS.FIT file. There are three pipeline data products per OSW, namely IMAGE_.FIT, SIMAGE.FIT and SWSRLI.FIT. If fast mode data are present, there are two products per OSW, namely TIMESR.FIT and its graphics version TIMESR.PDF. There are two pipeline products generated on a per observation basis, which are FLAFLD.FIT and OBSMLI.FIT.

It should be noted that the pipeline produces compressed FITS files named .FTZ while the output obtained when the chains are run by the user is not compressed , .FIT.

TSHPLT.PDF provides a visualisation of the tracking history file. It gives an overview of the pointing stability in the course of an exposure. The drift history of an exposure is displayed as a vector diagram. Each data point represents the average attitude solution of one tracking frame, lasting typically 20 s. The pointing drift is calculated with respect to the OM attitude at the time where the reference frame was defined. The reference attitude of an exposure is determined at the beginning of a science exposure. Histograms at the side of the vector diagram show the projection of the OM pointing drift onto the x- and y-direction. The second output page shows the incremental OM drift between 2 subsequent tracking frames. A clustering of points at one location would indicate a systematic drift into one direction. Typically the pointing stability is better than 1 arcsec during one exposure, which corresponds to 2 pixel on the diagrams. There is one TSHPLT PDF file produced for each THX file.

TSTRTS.FIT file:. There is one TSTRTS FITS file produced for each THX file. The binary extension comprises n columns, i.e one for each tracking star used. Each column list the count rate (cts/s) of a tracking star averaged over the tracking frame duration. Each row indicates a new tracking frame and one column corresponds to a timeseries of a specific star. The number of columns corresponds to the number of tracking stars used in an exposure. It can be any number up to 15.

IMAGE_.FIT file: The Primary array contains the flatfield and mod8 corrected image. The header contains the World Coordinate System parameters applied to the raw image. The RAW extension contains the original raw OSW image converted into real number represenation. The MODES extension lists the details of the window configuration. The QUALITY extension contains in an image the quality array. Any non-zero entry in the quality array reflects a bad pixel notified in the calibration bad pixel map or any location of the image where problems were encountered during the image processing.

SIMAGE.FIT file: The primary array of this dataset contains the north aligned sky image. The image is flatfielded, mod8 corrected, resampled and distortion corrected. The image is north aligned. WCS coordinates are contained as header keywords.

SWSRLI.FIT file: The primary header of an OSW source list file contains beside the definition of the OSW within the exposure and the observation, the sensitivity limit of the plate expressed in count rates and in instrumental magnitudes. The binary extension holds the list of detected sources. There are the following entries for each detected source: identifier within the OSW position within the OSW in pixel, the source position in ra/dec and galactic coordinates, the positional uncertainty, the extracted source count rates and its error estimate, the significance of the source detection, the brightness in instrumental magnitudes, the shape of the source expressed as the two semi-axes of an ellipse (for point sources the two axes should agree), the orientation of the ellipse and three quality flags, describing whether the source is thought to be extended, confused or affected by bad pixels. The last entry is the source identifier in the final combined source list.

FLAFLD.FIT file: Image either part of the pipeline products or generated by `omflatgen`. There is one file per observation containing an image extension. The image describes the flatfield response covering the whole detector at coarse resolution. It is set to unity.

IMAGEF.FIT file: This is the pseudo image which corresponds to the fast mode window. It is obtained from the original event-file list and it is equivalent to the IMAGE file produced for an OSW in image mode.

SIMAGF.FIT file: This represents the fast mode window as a skyimmage, north aligned. It

is equivalent to the SIMAGE file produced for an OSW in image mode.

TIMESR.FIT file: The time series contains source and background rates and their error estimates as a function of time, within the specified time sampling.

TIMESR.PDF file: The time series is plotted here to produce a graphic light curve, together with some statistics performed on the data.

OBSMLI.FIT file: This file contains the combined source list for all OSWs of an observation. Information of sources whose spatial position coincides within the specified "nsigma" are merged together. An observation source identifier is assigned to each source, which can be used to look up the sources in OSW source lists (SWSRLI files). The list contains, observation source identifier, position in ra/dec and galactic coordinates, positional uncertainty, the detection significance in the different filters, the source brightness expressed in instrumental magnitudes and its uncertainty, the quality, source extension and confusion flag for each filter, the characterisation and orientation of the spatial extent (described as in the SWSRLI files by the two semiaxes and the orientation angle in respect to the image x-axis) for each filter and finally if a transformation was possible the source brightness in a standard photometric system, as well as the colour indices.

3.5 Running the OM data processing

The data package received by the investigator contains OM data which normally do not necessitate further processing for the purpose of calibration. However, a user may want to apply the most recent calibrations, or to change some default parameters to e.g. improve the source detection on his/her data. It may not be necessary to run the complete chains, but just some tasks. All this can be done interactively.

The SAS_ODF environment variable shall be set to a directory containing the data and access to calibration files shall be set through `cifbuild`.

In addition to the house-keeping files listed in Table 4, the working directory needs to contain spacecraft files of the following form:

```
0070_0123700101_SCX00000SUM.SAS  - ASCII observation summary file
0070_0123700101_SCX00000TCS.FIT  - Spacecraft Time correlation file
0070_0123700101_SCX00000ATS.FIT  - Spacecraft Attitude file
```

Invoking `omichain` or `omfchain` will automatically start the processing of all OM data in the working directory. The duration of the process will depend on the number of exposures and windows and at the end we shall obtain the processed files described before. No intermediate file will be preserved.

Currently the chains can accept parameters to limit the processing to a given filter or a given exposure. Some of the default parameters used by individual tasks can also be tuned.

3.5.1 Example of image data processing

To process task by task a single exposure in image mode as 0070_0123700101_OMS00400IMI.FIT, the working directory shall contain the following files:

```
0070_0123700101_OMS00400WDX.FIT  - Exposure priority window file
0070_0123700101_OMS00400THX.FIT  - Exposure tracking history file
0070_0123700101_OMS00400IMI.FIT  - Exposure image file
```

and the process will be run in the following way

```
step1> omprep set=0070_0123700101_OMS00400THX.FIT \
              nphset=0070_0123700101_OMX00000NPH.FIT \
              pehset=0070_0123700101_OMX00000PEH.FIT \
              wdxset=0070_0123700101_OMS00400WDX.FIT \
              outset=tmp_tracking
```

In case there is no THX file, then set=DUMMYTHX.FIT. `omprep` will generate a dummy file needed for the rest of the chain, with zero drift in it.

```
step2> omdrifthist set=tmp_tracking plotfile=P0123700101OMS004TSHPLT0000.ps

step3> omthconv thxset=tmp_tracking nphset=0070_0123700101_OMX00000NPH.FIT \
               outset=P0123700101OMS004TSTRTS0000.FIT \

step4> omprep set=0070_0123700101_OMS00400IMI.FIT \
              nphset=0070_0123700101_OMX00000NPH.FIT \
              pehset=0070_0123700101_OMX00000PEH.FIT \
              wdxset=0070_0123700101_OMS00400WDX.FIT \
              outset=tmp_image_1

step5> omcosflag set=tmp_image_1 thxset=tmp_tracking
```

Although as we have said before no real flat field correction exists for OM, nor is it necessary, the processing requires such a file which can be generated using the task `omflatgen` as follows. (This task can also be run at the beginning of the processing)

```
step6.0> omflatgen outset=P0123700101OMX000FLAFLD0000.FIT
```

The output flatfield (primary extension) will be set to unity.

```
step6> omflatfield set=tmp_image_1 thxset=tmp_tracking \
       inorbitflatset=P0123700101OMX000FLAFLD0000.FIT \
       tsflatset=tmp_flat_field outset=tmp_image_2

step7> ommodmap set=tmp_image_2 mod8product=yes mod8set=tmp_mod8 \
                outset=tmp_image_3 nsig=5

step8> omdetect set=tmp_image_3 outputregionfile=true \
                regionfile=region_file outset=P0123700101_OMS004SWSRLI1000.FIT
```

```
step9> ommag set=osw_list wdxset=0070_0123700101_OMS00400WDX.FIT

step10> omatt set=tmp_image_3 sourcelistset=osw_list \
              ppsoswset=P0123700101OMS004SIMAGE1000.FIT usecat=F
```

Note that currently the usage of a catalogue for source crosscorrelation is disabled. The detected sources can be overlaidd on the OSW as follows:

```
step10a> implot set=P0123700101OMS004SIMAGE1000.FIT withsrclistset=yes \
                srclistset=P0123700101OMS004SWSRLI1000.FIT itf=1 \
                radius=5 maxsrc=200 colour=2 device='/XW'
```

In the standard automatic SSC pipeline processing, the tmp_image files are re−used and thus overwritten. In the task by task processing, they are distinguished so that intermediate stage output can be looked at if desired. In the processing by `omichain`, the file tmp_image_3 is renamed to P0123700101OMS004IMAGE_1000.FIT. Elsewhere in the above example, product names are used where they are similarly used in the SSC pipeline.

If multiple images are processed, the source lists can be combined as follows.

```
step11> omsrclistcomb sourcelistsets=P0123700101OMS004SWSRLI1000.FIT,..... \
                      nsigma=3 outset=P0123700101OMS000OBSMLI0000.FIT
```

where ..... is a continuing list of SWSRLI files.

Finally, the sky images of the multiple exposures corresponding to the default image configuration can be combined in a single sky image of the field of view. Note that this mosaiced image cannot be used for photometric purpose.

```
last step> ommosaic imagesets=``list of sky images'' \
                    mosaicedset=myfield_in_myfilter.fit
```

3.5.2 Example of fast mode data processing

In a similar way, if an exposure contains fast mode data, the working directory shall contain something like

```
0070_0123700101_OMS00400WDX.FIT  - Exposure priority window file
0070_0123700101_OMS00400THX.FIT  - Exposure tracking history file
0070_0123700101_OMS00401FAE.FIT  - Exposure fast mode data file
```

and the process can be run task by task in the following way

```
step1> omprep modeset=1 set=0070_0123700101_OMS00400THX.FIT \
               nphset=0070_0123700101_OMX00000NPH.FIT \
               pehset=0070_0123700101_OMX00000PEH.FIT \
               wdxset=0070_0123700101_OMS00400WDX.FIT \
               outset=tmp_tracking
```

As for image data, if there is no THX file, then set=DUMMYTHX.FIT. `omprep` will generate a dummy file needed for the rest of the chain, with zero drift in it.

```
step2> omdrifthist set=tmp_tracking plotfile=P0123700101OMS004TSHPLT0000.ps

step3> omthconv thxset=tmp_tracking modeset=1 \
               nphset=0070_0123700101_OMX00000NPH.FIT \
               outset=P0123700101OMS004TSTRTS0000.FIT

step4> omprep modeset=1 set=0070_0123700101_OMS00401FAE.FIT \
              nphset=0070_0123700101_OMX00000NPH.FIT \
              pehset=0070_0123700101_OMX00000PEH.FIT \
              wdxset=0070_0123700101_OMS00400WDX.FIT \
              outset=event_list.fit
```

In its second run, `omprep` is invoked for fast data (*modeset=1*) and the FAE raw event data is transformed into a modified event list to be used as input for `evselect`

```
step5> evselect table=event_list.fit xcolumn=RAWX ycolumn=RAWY \
               withimageset=true imageset=raw_image.fit
```

The pseudo-image corresponding to the fast mode OSW has been created by this first run of `evselect`.

```
step6> omfastshift set=event_list.fit thxset=tmp_tracking \
                   nphset=0070_0123700101_OMX00000NPH.FIT
```

The X- and Y- coordinates of the photon events are corrected for spacecraft drift. New columns are added to the event list with the corrected values.

```
step7> omfastflat set=event_list.fit \
                  slewflatset=P0123700101OMX000FLAFLD0000.FIT \
                  oswflatset=fast_flat.fit fastimgset=image.fit
```

Here again, as for image mode processing, the system is prepared to apply a subset of the `omflatgen` generated flat field to the fast mode window data, taking into account the spacecraft drift. The flat field is set to one, and therefore this correction has no real effect. The task generates the tracking shifted fast_flat.fit (only for the fast mode window) and the corrected pseudo image image.fit.

```
step8 >omdetect set=image.fit outputregionfile=yes \
               regionfile=region_file outset=osw_sourcelist.fit
```

The output region will allow the user to check the proper detection of the source in the small fast window. The PSF information is used to parameterise the detected source.

```
step9> omatt set=image.fit sourcelistset=osw_sourcelist.fit \
            ppsoswset=s_image.fit usecat=no
```

Astrometry is applied as for image data. The pseudo-image is north aligned too.

```
step10> omregion set=osw_sourcelist.fit srcnumber=1 \
                bkgfile=back_region.fit srcfile=source_region.fit
```

These regions will be used by `evselect` to filter out the event list, extracting the corresponding photon events for the source and the background. Optional parameters can be used to fine-tune the definition of the regions.

```
step11> evselect table=event_list.fit xcolumn=CORR_X ycolumn=CORR_Y \
                expression=‘‘((WIN_FLAG .eq. 0) .and. \
                (region(source_region.fit, CORR_X, CORR_Y)))’’ \
                maketimecolumn=T withrateset=Y timebinsize=10 \
                rateset=source_rate.fit

step12> evselect table=event_list.fit xcolumn=CORR_X ycolumn=CORR_Y \
                expression=‘‘((WIN_FLAG .eq. 0) .and. \
                (region(back_region.fit, CORR_X, CORR_Y)))’’ \
                maketimecolumn=T withrateset=Y timebinsize=10 \
                rateset=back_rate.fit
```

And finally the light curve can be obtained and plotted. The photometric corrections are also applied (coincidence loss, dead-time, PSF, magnitude conversion)

```
step13> omlcbuild srcregionset=source_region.fit bkgregionset=back_region.fit \
                  srcrateset=source_rate.fit bkgrateset=back_rate.fit \
                  sourcelistset=osw_sourcelist.fit \
                  wdxset=0070_0123700101_OMS00400WDX.FIT \
                  outset=lightcurve.fit

step14> lcplot set=lightcurve.fit binsize=1 plotfile=lightcurve.ps
```

3.6 Analysing OM data

As it has been pointed out already, OM data are fully processed by the XMM-SAS Pipeline, or the equivalent running of the image and fast mode chains: for each exposure of a given observation all necessary corrections are applied to the data files. Then a source detection algorithm is used to identify the sources present in the image. Standard aperture photometry is applied to obtain the count rates for all detected sources. These rates are corrected for coincidence losses and dead time of the detector and finally OM instrumental magnitudes and standard colour corrections are computed. A final source list is obtained from all exposures and filters. The detector geometric distortion correction and astrometric corrections are applied to each source's position and also the whole image is converted to sky co-ordinates space and

rotated so as to have the North on the top. Default image windows are also combined to obtain a mosaic (per filter) of the FOV.

However, one has to be aware of some peculiarities of OM data before starting the analysis.

- **Artifacts**

  OM images show several artifacts. These artifacts are generallly only visible when viewing the OM images in a high contrast and in logarithmic display. Artifacts can however affect the accuracy of a measurement, e.g. by increasing the background level. The following artifacts may be present in the raw OM images.

  - Out of time events: very bright sources with count rates of several tens of counts per second show a strip of events along the readout direction. These events correspond to out of time photons arriving while the detector is read out.
  - Smoke rings: bright sources generate smoke ring like structures, which are located radially away from the center of the field of view. These rings are caused by photons which are reflected from the detector entrance window back onto the detector photocathode.
  - Fixed pattern noise: raw OM images show a modulo 8 regular pattern originating from imperfections of the event centroiding algorithm in the the instrument electronics. This modulo 8 pattern is removed during image analysis by the task `ommodmap`.
  - Straylight features: straylight is caused by a chamfer in the OM detector housing which reflects light from outside the OM FOV onto the active detector area. This reflected celestial background light sums up onto a circular area with an increased background rate in the center of the OM field of view. The light coming from bright sources and reflected by the chamfer can also produce loop like structures in an OM image. These loops can degenerate into long streaks depending on the source positions.

The following points should also be kept in mind:

1. The OM operates in photon counting mode but images are accumulated on board. Good time intervals (GTI) have no sense in OM data. Either the full exposure is selected or discarded.

2. Contrary to the X-ray instruments, an OM exposure does not provide direct energy information except when grisms are selected.

3. As it has been explained, a flatfield response of unity is assumed.

4. Coincidence losses can occur which depend on the source brightness and on the CCD framerate. The framerate itself depends on the selected configuration of the detector science and tracking windows. If two or more events are located close to each other within the same CCD readout frame, they are detected and counted as one event. Also when photon splashes overlap, a mislocation is given by the centroiding algorithm. Another consequence of coincidence losses is that event depletion occurs at high count rates around the central source position.

5. The OM filters do not form a proper photometric system. However the photometric calibration of the instrument, based on observations with OM and from the ground, allows the SAS to obtain standard U, B, V magnitudes and colours in the Johnson system. This applies to stars. Extended objects should be treated with care.

6. The OM point spread function (PSF) has wide wings. This is taken into account in the application of the calibration with SAS through the proper setting of photometric apertures. A similar approach has to be taken if one wants to make an independent processing and analysis of OM data using any other data reduction package.

7. Straylight features, already mentioned, complicate the background subtraction in some cases, specially when the target star lays in or close to a straylight feature.

In principle, after SAS has been run there is no need for further data reduction. The observation source list should contain the calibrated data with their errors. Therefore the user can proceed with the analysis and interpretation of the processed data. However, some checking is convenient to verify the consistency of the data output.

The whole data processing can be repeated easily by any Guest Observer or Archives User, should any calibration file be updated, and what is more important, in case of doubtful results: the pipeline applies default options in all SAS tasks which eventually can be changed by the GO in order to improve the quality of the results. In particular, the source detection is very sensitive to the artifacts which are common in OM.

We outline here the checkings that any User should perform on OM processed data (by the standard Pipeline or by running SAS) and the use of one of the tasks, `omdetect` where the user can modify parameters affecting the source detection and therefore the overall results of the data analysis.

1. Checking `omichain` output products:

   - The first thing to do is to overlay the image source list onto the sky image. The task `implot` will do it:

     ```
     implot set=P0125320701OMS002SIMAGE2000.FTZ withsrclistset=y
            srclistset=P0125320701OMS002SWSRLI2000.FTZ \
            itf=1 radius=5 maxsrc=200 colour=2 device='/XW' \
     ```

     This will overplot the detected sources on the corresponding image, and it will allow us to check that all sources are real and that the one(s) we are interested in have been properly identified.
     If the background is strongly affected by straylight features, this check is very important

   - Inspection of the combined source list (e.g. using fv) will allow us to check that the sources of interest have been picked up in all the filters where they are visible and that the combined list contains colours and standard magnitudes for them.
   - Check the tracking corrections: although the pointing stability of XMM is very good, one can verify it by examining the corresponding tracking history PDF file.

2. Checking `omfchain` output products:

   - Inspection of the light curves:
     One can look at the PDF files containing the light curves to check that there is some signal detected and measured (both in the source and in the background).
   - Checking the presence of source(s) in the fast window pseudo-image.
     This can be done easily by displaying this image with SAOImage or fv. Another possibility is to overlay the detected sources onto the pseudo-image. The task `implot` will do it

     ```
     implot set=P0125320701OMS002SIMAGF1000.FTZ withsrclistset=y \
            srclistset=P0125320701OMS002SWSRLI1000.FTZ \
            itf=1 radius=5 maxsrc=100 colour=3 device='/XW'
     ```

3. Improving the source detection:

   - For image data, if the source of interest is close to straylight features or to other sources it may not be detected with the default settings of the `omdetect` task. We have to change its parameters: *boxscale, smoothsize, nsigma* and *contrast* (see SAS documentation, or run `omdetect -h` for details)

     ```
     omdetect set=your_image.fit outset=your_sourcelist.fit \
              outputregion=yes regionfile=your_region.dat \
              boxscale=n smoothsize=m nsigma=p contrast=q
     ```

     where 'your_image.fit', should have been produced by `ommodmap` The default values for *boxscale, smoothsize, nsigma* and *contrast* are 3, 64, 6 and 0.001 respectively.
     Invoking `omdetect` with *outputregionfile=yes region=your_region.dat* will allow us a fast checking by overlaying the newly detected sources positions on the image with SAOImage using the created region file.
   - In case of fast mode, if there are more than one sources in the fast window, then the detection will be affected and therefore the whole light curve too. We have to change some parameters in `omdetect`: *boxscale, smoothsize, nsigma* and *contrast* (see SAS documentation, or run `omdetect -h` for details).
     Invoking `omdetect` with *outputregionfile=yes region=your_region.dat* will allow us a fast checking by overlaying the newly detected sources positions on the image with SAOImage using the created region file.

3.6.1 Astrometry

The task `omatt` takes an OM OSW source list, with source positions in pixels, and converts to sky coordinates. These sky coordinates are then used to produce a sky coordinate image. The task requires the user to enter the name of an OSW source list and its corresponding OSW image. The positional accuracy obtained in this way is of the order of 2 arcsec, this error being due in part to the residuals of the geometric distortion correction.

In its second part, which can be run optionally by the user (it is not used in the pipeline nor in `omichain`), `omatt` employs the USNO SA1 catalogue (it has to be provided by the user).

The OM pointing vector and field of view are used as parameters to a search of the USNO, which lists all catalog stars in the field of view. Two synthetic images are constructed from the source and list and the catalogue stars. A two-dimensional cross-correlation is performed on the images, and this 2D cross-correlation function is searched for its maximum, which gives x and y offsets in pixels between the OSW source list and the catalogue. This offset is used to correct the catalog tangent plane coordinates. The nearest catalog star to each source in the source list is then found. If this distance is smaller than a tolerance times the positional error, then it is assumed to be that catalog star. If sufficient catalog stars are identified, then a least-squares fit is performed to the catalog positions, to yield more accurate astrometry. The source positions, in RA and DEC, are then written to the source list, along with the positional error (the pixel coordinates are retained in the source list).

3.6.2 Counts conversion to magnitudes and fluxes

The task `ommag` converts the count rates to magnitudes in the appropriate instrumental bandpasses. The rates are taken from a source-list produced by `omdetect` or from a timeseries produced by `evselect`. The output file will be a FITS file identical to the input source-list or time-series except that the count-rates have been converted into instrumental magnitudes (in the specified filter bandpass - U, B, V, UVW1, UVM2, UVW2) and then written as an extra column in the original FITS file. The program also calculates the limiting magnitude and writes the value in the FITS header.

In the near future, SAS will include a conversion from rates to fluxes in the corresponding bandpasses of the OM filters. In the meantime the following recipe shall be followed by any XMM-Newton users, who need or prefer to obtain fluxes rather than magnitudes in the UV filters (expressed, e.g., in erg cm^{-2} s^{-1} $Å^{-1}$):

- Vega is used as a reference to determine the magnitude to flux conversion. The UV flux of Vega should correspond to a magnitude of 0.025. The Vega fluxes in the OM UV filters are listed in Table 6.
- the standard formula can be applied

  $$0.025 - mag(UV) = -2.5 \log [Flux(Vega)/Flux(source)]$$

  where UV stands for UVW1, UVM2 or UVW2.

Filter	Effective wavelength (nm)	Flux (erg cm^{-2} s^{-1} $Å^{-1}$)
UVW1	291	3.7391873E-09
UVM2	231	4.4609810E-09
UVW2	212	5.3884371E-09

Table 6: Vega fluxes in the OM UV filters.

Table 6 reflects the current status of the OM on-flight calibrations, and are therefore being constantly verified and updated. The users are invited to check periodically the content of the calibration page, to ensure that the most updated calibrations are always being employed.

3.6.3 Analysis of grism data

The analysis of data obtained with the OM grisms is currently not supported by the SAS.