emframes

January 12, 2017

Abstract

Analyse frame information for one node of one EPIC-MOS CCD over one exposure.

1 Instruments/Modes

Instrument	Mode
EPIC MOS	IMAGING, TIMING

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

emframes analyzes the frames (auxiliary) file, adding four columns (a quality flag, the dead time fraction, the time and the GATTI value), performing various checks and computing the specific (to the CCD node) Good Time Intervals. The flag value allows to identify why the frame was flagged (using binary coding). **emframes** does not normally modify the events file (except in the case of frame renumbering as explained below). Events belonging to frames flagged as bad are themselves flagged for rejection by CUT_GTI in **emevents**.
emframes may be applied as is to slew data.

emframes may renumber the frames in the case of long telemetry drops. Frame renumbering allows to compute the GATTI value correctly, and to flag the events with truncated energy (as `REJECTED_BY_GATTI`) in **emevents**. Those flagged events are the best way to detect proton flares. Frame renumbering requires altering the events file as well.

When it is launched on an auxiliary file (straight from the ODF), **emframes** needs to access the ODF directory where the data comes from. This should be specified through the SAS_ODF environment variable, or the generic `odf` parameter on the command line. In that context, **emframes** also needs an events file (`odfeventset` parameter) to define the CCD/node and set the OAL state.
In that case **emframes** writes to the output frames file keywords filled using values taken from the

summary file via OAL calls. It also copies the FILTER keyword to the events file (for the CAL) if the events file was modified (newevent=Y or frame renumbering). If the events file was not modified the FILTER keyword is not copied to avoid the overhead of rewriting the full file.

On the other hand, **emframes** launched on its own output does not need an ODF directory nor an events file. All functions described below may be called that way except FRAMES.

emframes calls (in order) the following subroutines, all of which can be individually switched off:

- FRAMES. The time transmitted in the ODF is still in FTCOARSE/FTFINE form (FTFINE is in units of 40 μs). FRAMES starts by converting that in seconds (double precision real) and adding the sequencing delays, creating a TIME column. FRAMES computes the precise frame integration time and writes it into the FRMTIME keyword if it is significantly different from the original FRMTIME value. It adds the (constant) time offset to all frame times after wrap-around in the event of a long exposure ($>$ 32767 s), and converts the times into standard XMM times (running from 01/01/1998).
  In TIMING mode the times are corrected for the delay necessary to transfer the data through the 600 rows in the framestore area, and to transfer the data to the framestore area, assuming the vertical binning is 101. By default the source position is taken from the target coordinates (RA_OBJ, DEC_OBJ). This may be overridden by the user setting the source position manually via the parameters srcra and srcdec. If no RA_OBJ, DEC_OBJ keyword is present and srcra/dec are not set the source is assumed to be at the centre of the CCD. The NPIXEL column is recomputed to give the number of pixels above threshold per cycle (in the auxiliary file NPIXEL gives the integrated number of pixels above threshold since the beginning of the exposure).
  Finally, FRAMES tests whether the data was obtained with GATTI on or off and writes a GATTI_ON keyword to reflect that.

- Frame renumbering. The frame numbers are first scanned for regular increase, and a permutation array is created if some subset of frame numbers needs to be sorted. If a forward jump is detected then the time difference between the frames across the jump is examined. As the frame number is known only modulo 16 in the telemetry, a multiple of 16 frames may need to be inserted. How many times 16 frames is known by dividing the time difference by the frame integration time. This (-1) gives the most likely value of (and an upper bound to) the true number of missing frames (each extended frame reduces that number). If the interruption was due to counting mode, and the counting mode file was entered via the countingset parameter, then the number of missing frames is taken directly from the corresponding entry in that file. The number of missed frames is added to the FRAME columns in the frames and events files after the jump.
  If a frame number is repeated then FRAMES attempts to find one of the set with the expected time value; failing that, to find one of the set that might represent a cycling of the time; and failing that the frame with the smallest increase in time available. This "good frame" is marked with flag_mulfid and retained in the output data, while the other frames in the set are marked with flag_badfid and rejected.
  Frames with extended (n times) integration time are detected by looking at the time difference between successive frames (n times the frame integration time for extended frames). A statistical test on NPIXEL (comparing the local value with the overall distribution in the file) is performed to tell extended frames from undetected telemetry drops (integer multiple of 16 frames). NPIXEL (total number of pixels above threshold) must be larger in an extended frame.
  One row per frame (including dummies for missing frames) is created in output. This allows direct access by **emevents**. A dummy first row is always added (giving the estimated start time of the first frame). The first and last frames are flagged (they are usually not reliable).

- FLAG_HK. Outside normal operating conditions the instrument performances would presumably be degraded, possibly so much so that the following operations do not even make

sense. Some of the later tasks (CCDBKG in **emenergy**) build integrated images over the whole exposure, from which the frames judged bad by **tabgtigen** must be excluded. All frames whose start and end times are not inside the same good time interval (defined by **tabgtigen**) are flagged as bad.

- FIFO tests all the frames for a possible FIFO overflow flag and flags them as bad.
- VALID checks the number of events in the events file corresponding to a given frame versus the `NVALID` field in the frames file and flags as bad incomplete frames with the wrong number of events. In that case `NVALID` is set to the number of events in the events file.
- CR_DEAD. Every cosmic ray interaction with a CCD prevents a genuine X-ray from being detected there. This amounts to a source of dead time, which should be quite small on average ($< 1\%$). The number of pixels affected can be readily calculated by subtracting the number of pixels above threshold in the true events (known from their pattern, excluding "cosmic-ray" patterns) from the total number of pixels above threshold (`NPIXEL`) in the frame. The number of events below the lower EMDH threshold (`NBELOW`) should also be subtracted (those are mostly single noise excursions and incur no loss of efficiency). This allows to check the coherence of `NPIXEL` with the other data, since the subtraction must always remain positive. If it is not then the frame is flagged for rejection.
  Since all pixels neighbouring cosmic ray events are also lost, a statistical factor (COSMICSIZE field in the `XMM_MISCDATA` CCF file) must be applied. Another statistical correction must be applied because only part of the CCD surface is in view of the sky (because it is partly outside the field of view or because the edges are masked by other CCDs above), and because the dark part sees cosmic rays differently (COSMICOUTOVERIN field in the `XMM_MISCDATA` CCF file).
  The list of "cosmic-ray" patterns is derived from the patterns' definition by looking for all non-isolated patterns (where pixels around the central event are not necessarily below threshold) and is written into keywords `CRPATi`.
  In Compressed Timing mode where the pattern number is not transmitted to the ground it is taken to be 0 (single event, most likely value). As the true number of pixels per event can only be larger, this means the dead time is slightly overestimated in that mode.
- MAKE_GTI forms good time intervals for the current CCD, excluding all frames flagged as bad by FRAMES, FIFO, VALID or CR_DEAD (but not those flagged by FLAG_HK). It finally computes and writes the `LIVETIME` and `ONTIME` keywords corresponding to the total good time corrected or not for dead time, and rejecting the bad frames and those flagged by FLAG_HK.
- PUT_GATTI recomputes the GATTI value along its periodic triangle variation from 0 to 255 and back, using the GATTI flag as a marker, into `GATTIVAL`. This prepares SP_GATTI of **emevents**, which allows to check how the GATTI correction works. If the number of frames is less than 510 (this is often true for calibrations), no GATTI flag appears and this technique is not applicable. In that case the GATTI value is set to the input row number (best guess).
  In free run modes (Timing or free run Window) the GATTI is not used and PUT_GATTI just sets `GATTIVAL` to 0.

Recapitulation of flag definitions (flags below 64 are just warnings):

Value	Meaning
1	Extended frame integration time
2	Telemetry drop before this frame
4	Frame kept from a sequence of duplicated data with increasing IDs
8	No attitude available (Timing mode only)
16	Frame thought to contain good data, whose ID number was duplicated
64	Frame thought to contain bad data, whose ID number was duplicated
128	`NPIXEL` too small or `NABOVE` too large
256	Frame is not inside Good Time Interval from Housekeeping
512	FIFO overflow
1024	Number of events in the events file is not `NVALID`
2048	Time negative or aberrant leap in time tag
4096	Missing frame inserted after detection of telemetry drop
8192	First or last frame
16384	Time decreased unexpectedly

4 Parameters

This section documents the parameters recognized by this task (if any).

Parameter	Mand	Type	Default	Constraints
auxiliaryset	yes	dataset	' '	none
Name of input auxiliary or frames file				
odfeventset	no	dataset	' '	none
Name of events file				
countingtset	no	dataset	' '	none
Name of counting mode file				
newframeset	no	boolean	yes	yes/no
Create output frames file (no for overwriting input)				
frameset	no	dataset	'frames.out'	none
Name of output frames file (`newframeset`=yes)				
neweventset	no	boolean	no	yes/no
Create output events file (no for overwriting input)				
outeventset	no	dataset	'events.out'	none
Name of output events file. If this parameter is set, then `neweventset`=Y is automatic				
checkframes	no	boolean	yes	yes/no
Activate FRAMES ?				
withsrccoords	no	boolean	no	yes/no
Supply source position (Timing mode only) ?				
srcra	no	angle	0.	none
Source right ascension (J2000). Timing mode only				
srcdec	no	angle	0.	none

Source declination (J2000). Timing mode only

ingtiset	no	dataset	' '	none

Name of input good time intervals file from HK (activate FLAG_HK)

flagfifooverflow	no	boolean	yes	yes/no

Activate FIFO ?

checknvalid	no	boolean	yes	yes/no

Activate VALID ?

setdeadtime	no	boolean	yes	yes/no

Activate CR_DEAD ?

writegtiset	no	boolean	no	yes/no

Write good time intervals ?

outgtiset	no	dataset	'ccdgti.out'	none

Name of output good time intervals due to the CCD itself (`writegtiset`=yes)

setgatti	no	boolean	yes	yes/no

Activate PUT_GATTI ?

5 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

getParamValues01 *(error)*
`checkframes`=yes and no input events file

readEvents01 *(error)*
invalid input events file

frames01 *(error)*
`checkframes`=yes and non-ODF style input frames file

frames02 *(error)*
invalid input auxiliary file

frames03 *(error)*
no data for `CCDID`/`CCDNODE` of events file in auxiliary file

frames04 *(error)*
input events file incompatible with auxiliary file

frames05 *(error)*
non increasing frame numbers over entire auxiliary file

frames06 *(error)*
invalid input counting mode file

frames08 *(error)*
a timein value larger than the cycling time has been found

readFrames03 *(error)*
invalid input frames file

readFrames04 *(error)*
input events file incompatible with frames file

getParamValues10 *(warning)*
`checkframes`=yes and `newoutput`=no (overwrite). Skip FRAMES
corrective action: restart from auxiliary file if you wish to reapply FRAMES

getParamValues15 *(warning)*
`odfeventset` not set and `checknvalid`=Y or `setdeadtime`=Y. Skip VALID and CR_DEAD
corrective action: run **emframes** with `odfeventset` set if you wish to aply VALID or CR_DEAD

readEvents10 *(warning)*
invalid input events file. Skip VALID and CR_DEAD
corrective action: check input events file

flagHk10 *(warning)*
invalid input HK GTI file. Skip FLAG_HK
corrective action: check the HK GTI file

frames10 *(warning)*
time interval not constant in data, or input file has fewer than 4 frames. Do not check `FRMTIME`
corrective action: check `FRMTIME` manually

frames11 *(warning)*
the value of CLOCK_WRAP_AROUND in the summary file is invalid. Use 32767. This is normally all right
corrective action: check `FRMTIME` manually

frames12 *(warning)*
some frames were dropped from the end of the file in analysing the data due to non-increasing frame numbers
corrective action: check `FRMTIME` manually

frames13 *(warning)*
duplicate frame numbers found in the input file, all but one were dropped.
corrective action: check in the input auxiliary file that the proper frames were selected for analysis

frames14 *(warning)*
some frame numbers were found out of sequence, so the input data were sorted.
corrective action: none

frames15 *(warning)*
frame integration time significantly different from `FRMTIME` keyword. Update `FRMTIME`
corrective action: none

frames16 *(warning)*
aberrant time in the auxiliary file. Continue, flag such frames and ignore or set their time approximately
corrective action: none

frames17 *(warning)*
non-increasing time in the auxiliary file. Continue, flag such frames and ignore or set their time approximately
corrective action: none

valid10 *(warning)*
`checknvalid`=yes and VALID already applied. Option ignored
corrective action: restart from auxiliary file if you wish to reapply VALID

valid11 *(warning)*
`checknvalid`=yes and events have frame number outside frame interval in frames file. Continue
corrective action: none

crDead10 *(warning)*
`setdeadtime`=yes and NPIXEL − NBELOW − NABOVE < sum of pixels in valid events. Flag as bad and continue
corrective action: none

makeGti13 *(warning)*
all frames flagged as bad. Empty output GTI file. Continue
corrective action: check the log (with verbosity set to 4 at least) to know why those frames were rejected

testDrop12 *(warning)*
more than 50% time lost. Continue
corrective action: check in the log that nothing went wrong

putGatti10 *(warning)*
`setgatti`=yes, but less than 255 frames. set GATTI=ROW
corrective action: if this results in systematic spGatti11 warnings in **emevents**, you may add the offset manually

putGatti11 *(warning)*
`setgatti`=yes, GATTI on but no `GATTIFLG`=1. Skip PUT_GATTI. The truncated events cannot be identified
corrective action: none. This is hopeless

putGatti12 *(warning)*
`setgatti`=yes, GATTI on and `GATTIFLG` not found at expected periodicity. Continue, it may have occurred within a missing frame
corrective action: if this results in systematic spGatti11 warnings in **emevents**, do not trust the flare screening light curve

putGatti14 *(warning)*
`setgatti`=yes and interval between two successive GATTI flags not equal to 510. Skip PUT_GATTI. The truncated events cannot be identified
corrective action: check the log for frame renumbering messages. If you see where **emframes** erred, try renumbering the frames manually and run **emframes** again

createOutput10 *(warning)*
incoherence between `FRMTIME` and `GATTIFLG`. Continue anyway (`FRMTIME` is probably wrong, this does not affect the processing)
corrective action: inform SOC (this should not occur in any mode)

adjustTimetag10 *(warning)*
reconstructed source position is outside CCD. Flag frame and continue (most likely an attitude error)
corrective action: if this is systematic, the target position `RA_OBJ` / `DEC_OBJ` may be wrong. Try providing the source position manually setting `srcra/dec`

6 Input Files

1. EPIC MOS auxiliary file or frames file (from an earlier run of **emframes**). Uses all columns and keywords TELESCOP, INSTRUME, OBS_ID, EXP_ID, DATE-OBS, DATE-END.
2. EPIC MOS event list file for one CCD/node (from ODF/SDF) if odfeventset is set. Uses columns FRAME and PATTERN, and keywords TELESCOP, INSTRUME, OBS_ID, EXP_ID, CCDID, CCDNODE, WINDOWDX, WINDOWDY, FRMTIME, DATATYPE, plus those required by OAL_setState and CAL_setState.
3. EPIC MOS counting mode file if countingset is set. Uses columns HBRID, FRAMCNTR, ETCOARSE, and ETFINE.
4. good time intervals from housekeeping (from **tabgtigen**) if ingtiset is set.

The structure of files in the ODF is described in [1].

7 Output Files

1. frames file with additional rows and following items:
   - general keywords OBS_MODE, OBJECT, OBSERVER, RA_OBJ, DEC_OBJ, RA_NOM, DEC_NOM, REVOLUT taken from summary file, FILTER (taken from the HK), EXPIDSTR (taken from the original file name in the ODF), DATAMODE (deduced from DATATYPE in event list) and SUBMODE (Guest Observer mode, CCD 1 only) in the primary header (for PRODUCT: EPIC event list)
   - columns FRAME, NPIXEL, NBELOW, NABOVE, GATTIFLG, FIFOOVF and header propagated from auxiliary file
   - NVALID column updated (for **emevents**)
   - REAL*8 column TIME (s) (for **emevents**)
   - INTEGER*2 column FLAG (for **emevents**)
   - INTEGER*2 column GATTIVAL (for **emevents**)
   - REAL*4 column CRRATIO (dead-time fraction for **emevents**)
   - CCD specific keywords propagated from event list file
   - keyword CCDMODE expliciting CCD operating mode
   - keyword GAIN_CCD (when set to LOW, the PHA values are approximately divided by 10, and the data is not suitable for scientific use)
   - keyword FRMTIME (ms) (frame integration time, for PRODUCT: EPIC event list)
   - keywords TSTART, TSTOP, TELAPSE, ONTIME and LIVETIME (start and end times, exposure duration, sum of good time intervals and dead-time corrected on time), TIMEUNIT, TIMESYS, MJDREF, TIMEREF, TASSIGN, TIMEZERO and CLOCKAPP (origin and nature of times)
   - keywords ATT_SRC, ORB_RCNS, TFIT_RPD, TFIT_DEG, TFIT_RMS, TFIT_PFR set by the OAL.
   - keywords SRC_RAWX and SRC_RAWY (source position on CCD in Timing mode), with associated SRC_RA and SRC_DEC (source coordinates) if manually set
   - keyword GATTI_ON (boolean indicating whether the GATTI was on or off, for PRODUCT: EPIC event list and **arfgen**, **rmfgen**)
   - keywords CRPATi expliciting "cosmic-ray" patterns (for **emevents**)

- keywords detailing which subroutines were activated
- comment lines with names of input files

2. events file with same structure as in input, added `FILTER` keyword, modified `FRAME` column and possibly deleted rows (in case frames were renumbered).
3. GTI file specific to that CCD/node (for **evselect**, [2]), if `writegtiset`=yes. The GTI extension is called `STDGTInn`, where nn is the CCD number + 10 times the CCD node (as in **evlistcomb**). Keywords are copied from the frames file.
4. Temporary copy of the input auxiliary file (`emframes.temp`), normally removed by the task on exit.

8 Algorithm

```
MODULE em_frames_module

   Read the file names of frames files (input and output) and events file

   Opening of input auxiliary and events file
   Select CCD/node in auxiliary file
   Copy of input events file header keywords to output frames file

   parameters read : choice of sub_tasks to execute

   Loop over input task parameters
      Read the task parameter / 1 to perform the procedure,
                               0 not to perform it  /
   End loop

   Get the values of files columns in memory

SUBROUTINE FRAMES
      frame time calculation as difference between successive read-out times
             use only if constant 3 times in a row
             otherwise keep original value
      Read counting mode file if any
      Renumber frames and associated events if times indicate
             that frame number was lost.
      Build complete arrays of flags and times
             including missing frames and correcting wrong times
      Write columns to output file
      Put the keyword FRMTIME on file

SUBROUTINE FLAG_HK
      if flag_hk requested  then
           read the gti file name parameter
           Read GTI file
           Loop over the rows of frames file
               Loop over good time intervals
                  flag frame outside intervals
               end loop
           end loop
      endif
```

```
SUBROUTINE FIFO  check the FIFOOVF
   if fifo requested  then
      Loop over frame rows
         flag frames with FIFOOVF not equal to 0
      end loop
   endif

SUBROUTINE VALID    check the number of events for each frame
    if valid requested  then
      Loop over frame rows
        count the number of rows in the events file for each frame
        and compare with the NVALID value
        flag the frame if not equivalence
      end loop
    endif

SUBROUTINE CR_DEAD
   identify the ``cosmic-ray'' patterns,
   and get the number of pixels above threshold for each type of pattern
   for patterns CRPATi we put 0 to avoid subtracting cosmic-rays
   if cr_dead requested then
      Read cosmicsize and cosmicoutoverin values from CAL
      Loop over frame rows
        calculate crratio as
        ( NPIXEL(frame) - NPIXEL(events) - NBELOW ) / ( DX * DY )
        check  crratio > NABOVE / ( DX * DY )
        apply correction factor
        crratio = crratio * cosmicsize / (ratarea+(1-ratarea)*cosmicoutoverin)
        ratarea = fraction of window in open view of the sky
      end loop
      write CRRATIO column on file
    endif

SUBROUTINE MAKE_GTI    Build new GTI from unflagged frames
    if make_gti requested then
      Loop over the frames
        Reconstruct new GTI intervals excluding all flagged frames
      end loop
    endif

SUBROUTINE PUT_GATTI    check the gatti flag and write the gatti value
    if put_gatti requested then
       Loop over frame rows
          Look for GATTI flag = 1
          recompute the gatti value in previous 510 frames
       end loop
       Recompute GATTI in last frames
       write GATTIVAL column on file
    endif

    write FLAG column on file

    Closing files
```

```
end module
```

9 Comments

None.

10 Future developments

References

[1] ESA. XMM Interface Control Document: Observation and Slew Data Files (XSCS to SSC) (SciSIM to SOCSIM). Technical Report XMM-SOC-ICD-0004-SSD Issue 2.5, ESA/SSD, June 2000. Found at the URL: ftp://astro.estec.esa.nl/pub/XMM/documents/odf_icd.ps.gz.

[2] L.Angelini I.George. Specification of Physical Units within OGIP FITS files. Technical Report OGIP/93-001, NASA/GSC, May 1995.