ebadpixupdate

August 16, 2005

Abstract

Update event list with new bad pixels from the CCF or external files

1 Instruments/Modes

Instrument	Mode
EPIC MOS	IMAGING
EPIC PN	IMAGING

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

ebadpixupdate reads updated lists of bad pixels for each CCD (either provided by files or read in the calibration files) and flags a merged events list (like the calibrated events list delivered to the users as a PPS product) accordingly.

3.1 Introduction

The merged events list has already been processed for bad pixels during pipeline processing. However there are cases when this automatic procedure (using **badpixfind** for PN and **embadpixfind** for MOS) does not work perfectly and the user may wish to improve on it, without regenerating his events lists from scratch (**epchain** and **emchain** take a long time to run).

If a single bad pixel is obvious in the data, it is possible to eliminate the corresponding events in **evselect** by testing `CCDNR`, `RAWX` and `RAWY`. Then the user must manually update the `BADPIXnn` extension (where nn = CCDNR) to ensure that **eexpmap** and **arfgen** will account for the new bad pixel. If there are several such bad pixels, this becomes quite tedious. **ebadpixupdate** proposes to do that for you.

3.2 How to provide the bad pixels

Two sources of bad pixels are considered here:

1. the CCF; because the pipeline processing takes place within days of the observation, the CCF at the time of processing is necessarily always late (*i.e.* it was built on the basis of previous observations). A few months later, the user may note that the applicable bad pixels file (listed in the calibration index file generated by **cifbuild**) has changed.
2. bad pixels files; those may be obtained from the merged events list by running **emeventsproj** and **embadpixfind** for MOS, or **badpixfind** for PN, with user-defined settings and energy or time selection. They may also be written manually (respecting the same format described in **badpix**). Another possibility is to take the `BADPIXnn` tables from another events list.

3.3 What the task does

- If `fromccf`=Y, **ebadpixupdate** loops over all CCDs selected by the `ccds` parameter and reads all bad pixels for that CCD
- If `fromfiles`=Y, **ebadpixupdate** loops over all tables (or files) listed in the `badpixtables` parameter and reads the corresponding CCD in the header of the table.

In both cases, it flags all events belonging to that CCD and:

1. on a bad pixel, whether bright or dead (`ON_BADPIX`)
2. next to a dead pixel/line/column (`CLOSE_TO_DEADPIX`)
3. next to a bright pixel (`CLOSE_TO_BRIGHTPIX`)

CLOSE_TO means that the bad pixel is immediately next to the main pixel of the event (not diagonal).

As a second step, **ebadpixupdate** updates the bad pixels extensions (`BADPIXnn`) appended to the merged events lists, or adds a bad pixel extension if it is missing. This ensures that the bad pixel information will be propagated to the exposure map (by **eexpmap**) and the effective area (by **arfgen**). The `BADFLAG` column is set to 1 for uplinked bad pixels, to 2 for a CCF origin, to 3 for a file origin. For the case where the same bad pixel is present with both origins (*e.g.* a bad pixel with `BADFLAG`=3 originally and now present in the CCF) the smaller number is written (`BADFLAG`=2, CCF origin).

3.4 What the task does not do

ebadpixupdate does not delete any event. It only flags them. Running **evselect** with #XMMEA_EM or #XMMEA_EP will delete events **on** bad pixels. Running **evselect** with FLAG==0 will delete events close to bad pixels as well. Of course any more specific selection may be applied.

3.5 Replacing bad pixels instead of adding new ones

In the calibrated events lists generated by the pipeline, the events found to be on a bad pixel have already been removed. That operation is of course irreversible.

But **epchain** and **emchain** allow to generate events lists in which all events flagged for rejection (including those on a bad pixel) are still present. In that case the bad pixel flagging is reversible. Note that this is not completely true for MOS, because **emevents** by default removes the bright pixels and reanalyses events, so that a double event including a bright pixel will be changed into a single plus the (flagged) bright pixel. That behaviour may be suppressed (analysepatterns=N) in which case entire events are flagged, as PN does.

The `overwrite`=Y option may be used to replace the existing bad pixels with the new ones instead of adding the new bad pixels to the old ones. This means the events will be unflagged and the bad pixel lists reset, for all CCDs declared in `ccds` or corresponding to the tables in `badpixtables`.

The bright pixels previously declared on-board (`BADFLAG`=1) will not be removed from the list of bad pixels by the `overwrite`=Y option (the list of on-board bright pixels will be incremented). The corresponding pixels were removed on-board, so they cannot be recovered. Nevertheless, if the list of on-board bright pixels written by the PPS or **emchain** was wrong, the `replaceonboard`=Y option allows to replace them and reset the `CLOSE_TO_ONBOARD_BADPIX` flag. The other bad pixels will be incremented, unless `overwrite`=Y as well.

Use those option with caution. The program will try to check that the events flagged previously were not already removed, but this is not fool proof. In a non-interactive context, the `forcereplace`=Y option should be used to prevent activating the checks.

3.6 Examples

- Standard run (get bad pixels from the CCF) over all CCDs:

```
ebadpixupdate eventset=P0123700101PNS003PIEVLI0000.FIT
```

- Look for bright pixels in the low energy band and apply them (MOS, sh syntax):

```
evselect table=P0123700101M1S001MIEVLI0000.FIT keepfilteroutput=Y \
         withfilteredset=Y filteredset=events.lowenergy           \
  expression="(PI<500) && (TIME in GTI(P0123700101M1S001FBKGTI0000.FIT))"
for CCD in 01 02 03 04 05 06 07
do
   emeventsproj eventset=events.lowenergy mergedeventlist=Y \
                ccdnr=$CCD evimageset=evmap.in
   fmedian evmap.in medmap.in 5 5 boundary=reflect clobber=Y
   embadpixfind evimageset=evmap.in medianset=medmap.in \
                badpixset=badpix$CCD.out
done
ebadpixupdate eventset=P0123700101M1S001MIEVLI0000.FIT fromccf=N \
   badpixtables='badpix01.out badpix02.out badpix03.out badpix04.out  \
                 badpix05.out badpix06.out badpix07.out'
```

- Take additional bad pixels from another event list and the CCF, for CCDs 2 and 3:

```
ebadpixupdate eventset=P0123700101M2S002MIEVLI0000.FIT ccds='2 3' \
   badpixtables='P0111090301M2S002MIEVLI0000.FIT:BADPIX02  \
                 P0111090301M2S002MIEVLI0000.FIT:BADPIX03'
```

4 Parameters

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

Parameter	Mand	Type	Default	Constraints
eventset	yes	dataset	' '	none

name of events file (input/output)

fromccf	no	boolean	yes	yes—no

whether to look for new bad pixels in the CCF

ccds	no	list of integers	0	≥ 0

list of CCDs to update from the CCF. 0 to loop over all CCDs

fromfiles	no	boolean	no	yes—no

whether to look for new bad pixels in user-supplied files

badpixtables	no	list of tables or datasets	' '	none

input bad pixels tables or files (per CCD). If this parameter is set, then `fromfiles`=Y is automatic

overwrite	no	boolean	no	yes—no

replace the existing bad pixels with the new ones

replaceonboard	no	boolean	no	yes—no

replace bright pixels declared on-board

forcereplace	no	boolean	no	yes—no

allow `overwrite`=Y when no event flagged as `ON_BADPIX` or `replaceonboard`=Y when no event flagged at all

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.

InstrumentNotValid *(error)*
Instrument must be EMOS1, EMOS2 or EPN Either the instrument header is wrong or the input file is not from an EPIC camera Ensure that instrument header is correct and that the input file is an EPIC camera file

evflagImage11 *(warning)*
`replaceonboard`=Y and no event flagged at all for that CCD. Switching to incremental

mode.
corrective action: If you really want to proceed anyway, use `forcereplace`=Y

evflagImage12 *(warning)*
`overwrite`=Y and no event flagged as `ON_BADPIX` for that CCD. Switching to incremental mode.
corrective action: If you really want to proceed anyway, use `forcereplace`=Y

evflagImage13 *(warning)*
A bad pixel has invalid coordinates. It will be ignored.
corrective action: inform SOC (this is probably an error in the CCF) if `origin`=ccf. Look at what happened during bad pixel search if `origin`=files.

6 Input Files

1. merged events file (from **evlistcomb**) with bad pixels (`BADPIXnn`) extensions. This is a PPS product (EPIC event list) [1].
2. Bad pixels tables (for example `BADPIXnn` in anoter merged events file) or files (with `BADPIX` extension) output of **badpixfind** or **embadpixfind**.

7 Output Files

1. merged events file (same format as in input) with updated `FLAG` column and `BADPIXnn` extensions.

8 Algorithm

```
Read the events list

Define flags array, set to 0

if fromccf then
   Loop on ccds
      Read the bad pixels for that CCD via the CAL
      call flagEvents
      call updateBadpix
   end Loop
endif

if fromfiles then
   Loop on bad pixels tables
      Read bad pixels
      call flagEvents
      call updateBadpix
   end Loop
endif
end
```

```
subroutine flagEvents
   map = 0

   if overwrite and (forcereplace or events ON_BADPIX exist)
      unset ON_BADPIX, CLOSE_TO_DEADPIX and CLOSE_TO_BRIGHTPIX
   if replaceonboard and (forcereplace or flagged events exist)
      unset CLOSE_TO_ONBOARD_BADPIX

   Loop over bad pixels
      map(xb,yb) = map(xb,yb) OR ON_BADPIX
      Loop over pixels next to (xb,yb)
         if dead pixel then
            map(x,y) = map(x,y) OR CLOSE_TO_DEADPIX
         else
            map(x,y) = map(x,y) OR CLOSE_TO_BRIGHTPIX
         endif
      end Loop
   end Loop

   Loop over events for that CCD
      FLAG = FLAG OR map(RAWX,RAWY)
   end Loop
end subroutine flagEvents

subroutine updateBadpix
   if not (overwrite and replaceonboard) then
      call readBadpix on BADPIXnn table for current CCD/node in events file
      if overwrite and (forcereplace or events ON_BADPIX exist)
         remove rows where BADFLAG>1
      if replaceonboard and (forcereplace or flagged events exist)
         remove rows where BADFLAG=1
   endif
   add list of new bad pixels at the end of the list
   call mergeBad to remove redundancies
   call writeBadpix to BADPIXnn table
end subroutine updateBadpix
```

9 Comments

None.

References

[1] SSC. XMM Survey Science Centre to Science Operations ICD for SSC Products. Technical Report XMM-SOC-ICD-0006-SSC Issue 2.1, SSC, Mar 2000.