**Swift BAT Ground Analysis Software Manual**
**Hans A. Krimm, Ann M. Parsons and Craig B. Markwardt**
**November 15, 2004**
**v2.0**

## FORWARD

This manual includes a description of the basic BAT FITS format data products, how they are produced and how they are analyzed. It includes a discussion of how the various BAT ground software tools are used. It is not intended to be a substitute for the manual pages provided with each of the tools. The manual pages remain the primary resource for the tools and the meaning and uses of their respective command line parameters.

## CHANGES FROM VERSION 1.0 OF THIS DOCUMENT

The document has been enlarged to include more details about the input and output file formats and the parameters for each tool. Examples (recipes) have also been added for most tools.

# TABLE OF CONTENTS

# I. INTRODUCTION

The BAT instrument is an imaging hard X-ray telescope, but it is not a focussing instrument. Therefore the primary (Level 0) files are not sky images, but rather images of the detector plane which must be processed (using a Fast Fourier Transform) to produce images of the sky.

Furthermore, unlike the Swift Narrow-Field Instruments, BAT has only one operating mode -- photon counting. Except when the event rate is too high for the processor to keep up (for instance during passage of the spacecraft through the South Atlantic Anomaly (SAA)), every photon is processed and becomes an "event," which has an associated time, detector number and energy. The events are stored in a ring buffer (which can store approximately ten minutes of event data at our predicted normal rate) and used to form gamma-ray burst triggers. Data throughput restrictions prevent sending the event buffer to the ground continuously. The event buffer is normally sent to the ground only following a detected gamma-ray burst. At all other times, each event is added to one of 32768 eighty-channel histograms. The histograms are accumulated for a time period that is normally five minutes and sent to the ground as a Detector Plane Histogram (DPH). The DPHs are thus the primary BAT data product and are used for ground-based transient searches and will be combined to produce the All-Sky Map for the BAT hard X-ray survey.

Physical Description

The BAT instrument consists of two basic elements: the detector plane and the coded mask. The detector plane consists of 32768 individual cadmium-zinc-telluride (CZT) detector elements. Each detector is 4 mm X 4 mm X 2 mm thick and arranged on a regular grid with 4.2 mm pitch. The detectors are arranged in the following hierarchy: 128 detectors form a *side* and two sides make up a *detector module*. There are eight detector modules in a *block* and sixteen blocks in the BAT array. Gaps between detector modules are integral numbers (two or three) of detector widths. The readout electronics includes digitization of the pulse height of each event along with time tagging with a time resolution of 100 μsec. The detector identity of each event is also reported. The only science output of the detector array is these event files. The other two lowest level BAT products are calibration events and housekeeping.

The coded mask (aperture) is located one meter above the detector plane. It consists of ~51,000 lead tiles, each 5 mm X 5 mm X 1 mm thick, arranged on a grid in a random pattern.

The effective energy range of BAT starts at approximately 15 keV and the nominal energy resolution is 6 keV. The effective area peaks at around 100 keV and falls off rapidly above that. Due to the 1 mm thickness of the lead tiles, the coded mask is transparent to photons above about 200 keV, cutting off the effective area for imaging at this energy. The overall effective area extends to above 1000 keV.

Overview of On-board Data Processing.

The BAT flight software runs as a set of interacting processing modules, which are here described very briefly. The primary modules are *data ingest* (di), *trigger* (tg) and *calibration* (ca). There are many other modules, including housekeeping and pulsar folding, that are not discussed here. Nor does this document contain any discussion of the Swift/BAT engineering software or low-level hardware interfaces.

Data ingest is the place in processing where event data from the array is ingested and sorted and where the calibration appropriate for each detector is used to convert from ADC channel to energy. The data ingest task fills the time-ordered event ring buffer. One of its most important tasks is binning the data, both to produce the Detector Plane Histograms and for the trigger process. The various rate files are also filled in this process.

The trigger process is central to the scientific mission of Swift. The basic procedure for rate triggers is as follows. Data is continually binned by the data ingest process into rolling time-based histograms (light curves) with time bins ranging from 4 ms to 16 seconds. Different histograms are accumulated for four energy bands (nominally 15-25 keV, 25-50 keV, 50-100 keV, and 100-350 keV) and for nine regions of the detector plane: the entire array, the four quadrants, and four halves (two halves sliced in different directions). The trigger process continually monitors the hundreds of histograms searching for rate increases in any of them. When a significant (typically 5-7 sigma) rate increase is found, the image processing software produces a sky image from the events in the high rate bin. It then subtracts a scaled background sky image from several seconds earlier. If a significant point source is found in the background subtracted image, then a final check is done to make sure the point source is not at the location of a known hard X-ray source. If the source is verified as new, it is reported as a burst and a message is sent to the spacecraft Figure of Merit (FoM) processor to initiate the Swift burst response.

The other part of the trigger process is the image trigger. In this mode, images are continually made on time scales ranging from sixty seconds to the duration of a pointing (snapshot). These images are searched for significant point sources which can also be reported to the FoM as bursts.

The calibration task typically runs during slews to pre-planned targets. There is a calibration pulser built in to the BAT hardware which can inject a known voltage into each detector in sequence. The calibration task has access to a lookup table (which can be updated from the ground) containing conversions from pulser voltage to energy. The task convolves the table with the ADC channel reading for each pulse in each detector to derive a conversion from ADC channel to energy. One point per detector is needed to derive an offset and two are needed to derive a gain. These offsets and gains are used in the data ingest task and are reported to the ground as calibration products.

Mask Tagging

Mask tagging is a technique for subtracting background from an event file or detector plane array in a coded aperture telescope. The technique takes advantage of the fact that background counts are randomly distributed across the detector plane, while

counts for a particular source fall in a calculable pattern. The mask tagging process involves forward projecting (ray tracing) photons from a known position in the sky through the coded mask onto the detector plane.

Mask tagging can either be applied to an event file (applied to event files with **batmaskwtevt**) or to a detector plane image (done in the flight code and in **batmaskwtimg**). In event tagging, for each event a single ray is traced from the source location to the particular detector in the event. The event is given a weighting from -1.0 to +1.0 based on the fraction of the detector that is shadowed by lead tiles. If the detector is fully shadowed it receives a weight of -1.0. If it is fully open it gets a weight of +1.0 and if half-shadowed, a weight of 0.0. Counts in the detector plane not coming from the source will receive weights averaging to zero. Counts coming from the source will only fall in open or partially open detectors and hence will receive a net positive weight. If the events are then binned into light curves based on their mask tagged weighting, the result will be a light curve that represents the net counts from the source.

Detector plane mask weighting is similar, but in this case, a ray is traced from the source to each of the 32768 detectors, each of which is given a weight. This technique is used for cleaning (see **batclean**) and is used in the flight processing to create a lookup table of weights which are applied to each event.

Overview of Data Products

There are many different Level 0 BAT data products, but they can be arranged into seven broad categories: (1) Event files, (2) Detector Plane Histograms, (3) Rate files, (4) TDRSS messages, (5) Calibration files, (6) Housekeeping, and (7) Diagnostic (or trend) data. The scientific user will typically only be concerned with the first four categories and those calibration files needed for analysis. The data in these seven categories are described in detail in the next section. In addition, the Science Data Center pipeline produces spacecraft attitude files and filter files, and there exists a Swift/BAT calibration database. In later sections, these files are described only to the extent necessary to use them in data analysis.

# II. DESCRIPTION OF BAT DATA PRODUCTS

All BAT data products are FITS files. The Level 0 and 1 products are either FITS tables or null files (header only). All files are named according to one of the following conventions ("sw" is Swift and "b" is BAT):

sw[Observation ID][Segment Number]b[code].[suffix]

-or-

sw[Time stamp]b[code].[suffix]

For example, an event file for Observation ID 00074651, segment 002, would have a name sw00074651002bevtstouf.evt, where the code for event files is "evtstouf" and the suffix is "evt." Similarly, a long trigger criteria trend file from MET 101809021 would have a name sw0101809021btblt.fits, where the code is "tblt" and suffix "fits."

This document will give only those codes and suffixes which apply to files being described in the document. A much more complete description of the BAT data products along with an exhaustive list of codes and suffixes, and the logic behind the codes, can be found in the document "BAT Data Products." The current version of that document is Version 3.1, 30 August 2004, by Hans Krimm.

## A. Event Files

These are time-ordered FITS tables, with each row corresponding to a single event. A fully processed event file will contain no duplicated events. Event files can be distinguished by the unique filename suffix "evt," and various codes refer to different types of events and different stages in processing.

The columns in a fully processed event file are:

| | | |
|---|---|---|
| HDU 2 EVENTS | BinTable | 8 cols x XXXXX rows |
| Col Name | Format[Units](Range) | Comment |
| 1 TIME | 1D [s] | TIME associated with event |

The time is reported in Mission Elapsed Time (MET), which is seconds from January 1, 2001.

| | | |
|---|---|---|
| 2 DET_ID | 1I | Detector Channel, DM, Side and Block |

This defines the location of the detector in the BAT electronics hierarchy. The result is a single integer between 0 and 32767. The coding is:

(2048 * Block) + (256 * DM) + (128 * Side) + (Channel).

$0 \le Block \le 15$ ; $0 \le DM \le 7$; $0 \le Side \le 1$; $0 \le Channel \le 127$.

| | | |
|---|---|---|
| 3 EVENT_FLAGS | 1B | DM Event Flags |

| | | |
|---|---|---|
| 4 PHA | 1I [chan] (0:4095) | Raw Pulse Height |
| 5 PI | 1I [keV] | Pulse Height Invariant Quantity |

This column is filled by ground processing (**bateconvert**) using the gain/offset calibration files. If this column has not been filled it will be absent.

| | | |
|---|---|---|
| 6 MASK_WEIGHT | 1E (-1.0:1.0) | label for field |

This is the weighting applied to each event (using **batmaskwtevt**) based on the location in the BAT field-of-view of a given source. Since event data is typically only downloaded for the time around a burst, this source is usually the burst. See discussion of mask weighting in the introduction. If this column has not been filled it will be absent.

| | | |
|---|---|---|
| 7 DETX | 1I (0:285) | BAT X coordinate |
| 8 DETY | 1I (0:172) | BAT Y coordinate |

The values in these columns give the geometric location of the detector producing each event. The geometric coordinates include gaps between detector modules so this space includes 286 X 173 = 49,478 positions of which 32,768 are filled by detectors. The mapping between (DETX, DETY) and DET_ID is quite complicated and can be done using the tool **batid2xy**.

Event files also include a Good Time Interval (GTI) extension giving the time spanned by events in the file.

B. Detector Plane Histograms

These files are the other primary science data product (along with event files). The bulk of the BAT data volume is Detector Plane Histograms (DPHs). Survey DPHs have suffix “dph” and are distinguished by the code “sv.” A typical file name also includes the gain/offset indices associated with the survey. For example, the file sw00074651000bsvo0002g0001.dph, would be for Observation ID 00074651, Segment 000, Offset index 0002 and gain index 0001.

The columns in a Detector Plane Histogram are:

HDU 2 BAT_DPH BinTable 11 cols x XXX rows

| Col Name | Format[Units](Range) | Comment |
|---|---|---|
| 1 TIME | 1D [s] | TIME associated with DPH |
| 2 EXPOSURE | 1D [s] | Exposure of the histogram |
| 3 DPH_COUNTS | 3958240I [count] | Detector Plane Histogram |

This is a three-dimensional data cube with 80 x 286 x 173 elements. The first dimension corresponds to the number of energy channels and can be varied, but will almost always be 80 for BAT surveys. The second and third dimensions indicate the geometric location of the detector in the BAT array plane. Since these dimensions

include gaps between DMs, approximately one third of the table is empty (filled with zeros).

| | | |
|---|---|---|
| 4 DATA_FLAGS | 1I | Data Quality 0=OK, 1=Problem |
| 5 GAIN_INDEX | 1J | Gain used in DPH |
| 6 OFFSET_INDEX | 1J | Offset used in DPH |
| 7 LDPNAME | 240A15 | BAT File Name |
| 8 BLOCK_MAP | 1I | Block Bit Mask |
| 9 NUM_DETS | 1J | Number of detectors |
| 10 APID | 1I | data came from ApID |
| 11 LDP | 1I | data came from LDP number |

The remaining columns 4-11 are mostly for bookkeeping and trend analysis, but the data could be filtered on some of these values.

The arrangement of data in the data cube was determined to maximize speed of filling the cubes from the flight data products. A consequence is that the data cannot easily be visualized with tools like **fv**. Two BAT tools, **batdph2dpi** and **batdph2pha** render the data cubes into, respectively, a single Detector Plane Image or 32768 detector spectra.

The DPH FITS files also include an EBOUNDS extension specifying the energy bounds of the histogram channels. There is also a GTI extension.

C. Rate Files

There are six types of rate files (light curves) produced by the BAT. Five of these are produced in final form as Level 0 products. One of them (mask-tagged light curves) requires further processing in the SDC pipeline using **batmasktaglc**.

The rate files are 1-second rates, quad rates, 64 msec rates, max rates, masktagged rates and burst light curves. The first five types are produced at all times when the data ingest task is running. The last type (burst light curve) is only produced in response to a burst. All rate files have suffix “lc”

1. One-second rate files (code “rt1s”) are single channel light curves representing the total count rate in the BAT array as a function of time. Time binning is one second. Since this file is derived from a hardware counter it continues to record the total BAT event rate even when the data ingest task is turned off (during SAA passage, for instance) and includes all events without any energy cuts.

2. Quadrant rates (code “rtqd”) contain four channel light curves for each of four quadrants of the BAT array. Time binning is 1.6 seconds and what is recorded is counts (not counts/second). The four channels correspond to the four nominal energy bands used in the trigger task. (15-25, 25-50, 50-100,100-350 keV). Events with

energies < 15 keV or > 350 keV are not included in these rates. The four quadrants are Blocks 0-3, 4-7, 8-11 and 12-15. These rates do not appear if the data ingest task is not running.

3. 64-msec rates (code “rtms”) contain a single four channel light curve giving the counts in the entire array with 64-msec time binning. The energy ranges are the same as for the quadrant rates. What is reported is counts (not counts/second).

4. Max rates (code “rtmc”) have five HDUs (plus a GTI extension). Each extension contains nine four-channel light curves with eight second time binning. The nine light curves in each extension correspond to the full array, the four quadrants (described under quadrant rates above) and four halves. The halves correspond to all possible adjacent pairs of quadrants, namely quadrants 0 &1, quadrants 0 & 2, quadrants 1 & 3, and quadrants 2 & 3. The five light curve extensions are:

| | | | |
|---|---|---|---|
| HDU 2 | MAX_COUNTS_04MS | BinTable | 11 cols x XX rows |
| HDU 3 | MAX_COUNTS_08MS | BinTable | 11 cols x XX rows |
| HDU 4 | MAX_COUNTS_16MS | BinTable | 11 cols x XX rows |
| HDU 5 | MAX_COUNTS_32MS | BinTable | 11 cols x XX rows |
| HDU 6 | MAX_COUNTS_64MS | BinTable | 11 cols x XX rows |

The light curves contain, for each eight second time bin, the maximum counts on the given time scale within those eight seconds. For example, regard the MAX_COUNTS_04MS extension. There are 2000 4-ms intervals within eight seconds. What is reported in the file is the number of counts in the 4-ms interval with the highest number of counts (in other words, the peak of the 2000 bin light curve). The same algorithm is used for the other four time scales.

This light curve is mostly used for diagnostics.

5. Mask -tagged light curves (code “mt”) are typically produced for three sources in the BAT field of view. In most cases, one source will be the burst being followed-up and the other two sources will be bright variable sources also in the field of view. The flight code weights each event for each source position (see “Mask Tagging” above) and bins the weighted events into light curves with 1.6 second time binning and four channel energy binning.

In order to reduce the time needed for mask tagging in flight, the weights are offset and scaled and the mask tagged light curves produced in flight are convolved with the quadrant rate files. All of this must be backed out in ground processing using the **batmasktaglc** tool. This tool reads in the raw mask tagged light curve, the corresponding mask weight file, and the quadrant rate file covering the same time period. It outputs a corrected mask tagged light curve with an EBOUNDS extension.

The raw light curve has only time and weighted counts columns (in four channels). The processed light curve has columns: TIME, RATE (counts/sec), ERROR (counts/sec) and BACKV (counts/sec). The last column is the background.

6. Burst light curves (code “msb” or “bhp”) are sent down in two channels, through TDRSS and in the high priority solid-state recorder ground pass channel. These files cover a time from 24 seconds before to 185 seconds after the burst. They have four energy channels. The time binning varies and is densest (0.128 seconds) closest to the trigger time and least dense (4.096 seconds) well after the trigger. Since these light curves come through the TDRSS channel they can be used to quickly determine high level burst properties.

The TDRSS light curves also have attitude data attached on the same time scale as the light curves. The attitude data is processed into separate files.

D. TDRSS Messages

In addition to the burst light curves (previous section), there are four TDRSS messages produced by the BAT. These are the burst alert (code “msbal”), burst position (ACK) (code “msbce”), no-position (NACK) (code “msbno”) and scaled maps (code “msbsm”). All rate triggers generated a burst alert. If no position is found, then a burst NACK follows. If a position is found, then a burst position message follows along with a scaled map. In either case (position or no position), a scaled map is sent down in the ground pass.

The burst alert, position and NACK messages are Null Arrays with all information contained in the header.

Scaled maps are detector plane images with a single number for each detector representing the counts in that detector during the time interval used to produce the image of the burst. Other columns contain diagnostic information about the burst.

E. Calibration Files

These include the files listed here.

1. Americium tagged source surveys (code “cbam,” suffix “dph”). Tagged calibration source events are segregated from normal events and binned in separate DPHs. The format is the same as the normal survey DPH, but the duration is much longer (typically 90 minutes). And since the tagged source position is fixed with respect to the detector plane, Am241 surveys can span pointings. These are used by the BAT team to derive the absolute energy calibration of the experiment.

2. Americium block spectra (code “cbam,” suffix “fits”) are eighty-channel histograms of tagged source events reported for each of the sixteen blocks. These are produced on the same time scale as the normal surveys (as opposed to Am241 surveys).

3. Calibration maps (code “cb_ “ suffix “dph”) contain the raw results from the pulser calibration runs used in on-board calibration. They are used by the BAT team to check the on-board calibration. Also if a series of more than two calibrations is commanded, the resulting calibration maps can be used to determine the non-linearity of the energy calibration.

4. Enable/disable maps (code “cbde,” suffix “fits”). These files contain a flag (0 or 1) for each detector indicating whether the detector is enabled or disabled in the flight processing. They are generated whenever additional detectors are disabled and during slews to bursts.

5. Gain/offset maps (code “cbo,” suffix “dph”). These files contain two detector plane maps giving the offset [keV] and gain[keV/channel] used in the flight code to calibrate the events. These are generated each calibration cycle (normally slews to pre-planned targets)

F. Housekeeping Files

These include engineering housekeeping files and the Detector Plane Array (DAP) housekeeping file which includes all “science” housekeeping from the BAT array.

G. Diagnostic and Trend Files

These include tables of Data Ingest (DI) commandables as well as tables listing the short and long trigger criteria and the trigger veto criteria. Other files in this category are the on-board source catalog, running sums tables for triggers, and tables giving diagnostics for rate and image triggers. The data log (shell file) is also included in this category.

# III. DESCRIPTION OF BAT ANALYSIS TOOLS

This section gives an overview of each of the BAT analysis tools and examples of how they are used. The tools are listed in alphabetical order.

## Very Brief Overview of Tools

(Note that DPH = Detector Plane Histogram, as described in the previous section)
**batbinevt:** makes light curve or spectrum from event file; also processes DPHs.
**batcelldetect:** finds both new and previously cataloged sources in image files.
**batclean:** cleans the background and (if desired) sources from BAT DPHs.
**batdph2dpi:** collapses DPH into a detector plane image.
**batdph2pha:** rewrites a DPH in the form of 32K individual detector spectra.
**batdrmgen:** creates a BAT detector response matrix for a given source location.
**bateconvert:** determines the energy (PI) from the ADC channel (PHA) in an event file.
**baterebin:** corrects DPH for non-linearity in the conversion from channel to energy.
**batfftimage:** derives a sky image from a DPH or makes a partial coding map.
**batgse2dpi:** internal tool to convert calibration data into flight-like format.
**bathotpix:** derives a map of hot (noisy) and cold pixels in the detector plane.
**batid2xy:** converts to and from detector order to geographic location in the array.
**batmasktaglc:** processes the raw masked tagged light curves from the flight data.
**batmaskwtevt:** calculates mask weighting for an event file.
**batmaskwtimg:** derives a mask weighting map for a particular sky location.
**batsumdph:** combines multiple DPHs.
**battblocks:** determines burst duration, peak flux, and interesting time intervals.

## Common Parameters

In addition to tool specific parameters which are described below in the section for each individual tool, all FTOOLs have the following hidden parameters which are described here:

| Parameter / Default / Data Type | Description |
|---|---|
| **(clobber)** = NO [boolean] | If the output file already exists, then setting "clobber = yes" will cause it to be overwritten. |
| **(chatter)** = 2 [integer, 0 - 5] | Controls the amount of informative text written to standard output. |
| **(history)** = YES [boolean] | If history = YES, then a set of HISTORY keywords will be written to the header of the specified HDU in the output file to record the value of all the task parameters that were used to produce the output file. |

# A. Batbinevt

This is a multipurpose tool which produces spectra and mask weighted light curves from BAT event files and detector histogram data. This tool can use the weights generated by **batmaskwtevt** or **batmaskwtimg** to make binned output light curves and spectra. It can also be used to add DPHs and collapse them (produce Detector Plane Images or DPIs).

## Input Files (batbinevt):

Hidden parameters are listed in parentheses. If these are required, then the code will use default values.

Required:

*infile:* Input file name containing a sky map. The input file must be either an event file or a DPH.

Event data must have the TIME and energy columns. If weighting is applied, then either the MASK_WEIGHT column must be present, or a separate mask weight image file must be supplied. An input event file must normally have been processed first by **batmaskwtevt** for a particular source position. Thus, **batbinevt** will only extract products for one source at a time, although it can compute many time or energy samples.

**Batbinevt** can also read detector plane histograms (DPHs). BAT DPHs have two spatial dimensions and one energy dimension. They are produced by the BAT flight software, and also perhaps previous runs of **batbinevt**. They must contain an EBOUNDS extension which describes the energy bin edges.

Optional:

*(gtifile):* Name of goodtime interval file. The user-supplied GTI file is logically intersected with the good times of the input files (i.e. only overlap portions are used). The default of NONE implies that all good time intervals from the input files will be used.

*(detmask):* Name of a detector quality map file. This should be an image file with the same dimensions as the detector plane map. A pixel value of 0 indicates the detector is enabled for imaging, and a non-zero value indicates disabled. A default value of NONE implies all detectors are on, except for the BAT detector gap regions.

## Output Files (batbinevt):

Required:

*outfile:* The output file types are either LC (light curve), PHA (counts spectrum), DPH or DPI. The most common output file types will be:

LC

Standard OGIP light curve (default: weighted=yes, outunits=RATE);

PHA

Standard OGIP spectrum (type II; default: weighted=yes, outunits=RATE);

**batbinevt** can also output histograms and images:

DPH

Detector plane histogram; for each temporal integration, a three dimensional histogram of the number of counts is constructed with two spatial dimensions (DETX and DETY) and one energy dimension. The EBOUNDS extension describes the energy binning. (default: weighted=no, outunits=COUNTS);

DPI

Detector plane image; a histogram of the number of counts in two spatial dimensions (DETX and DETY). Each FITS extension contains only one DPI; multiple images are stored in sequential FITS extensions. (default: weighted=no, outunits=COUNTS);

DPITAB

Detector plane image. Images are stored in rows of a FITS binary table (multiple images in one extension) The data is identical to the “DPI” option; only the FITS format is different. (default: weighted=no, outunits=COUNTS).

## Parameters (batbinevt):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

The table is found on the next page.

| Parameter / Default / Data Type | Description of batbinevt parameter |
|---|---|
| **infile** [filename] | Input file name containing BAT event or DPH data. |
| **outfile** [filename] | Output light curve or spectrum file name. |
| **outtype** [string] | Chooses output format :<br>light curve (outtype="LC")<br>spectrum (outtype="pha")<br>detector plane histogram (outtype="DPH"),<br>detector plane image (outtype="DPI")<br>detector plane image table (outtype="DPITAB"). |
| **timedel** = 1.0 [real] | Select a time bin size, in seconds. |
| **timebinalg** = uniform [string] | "uniform", for uniform binning<br>"snr", for constant signal to noise ratio<br>"gti" for binning according the GTI file<br>"infile" to mimic the input file binning |
| **energybins** = "-" [string] | Energy bin ranges, expressed as a comma-separated list of floating point number ranges, a file name containing energy bin ranges, or CALDB. |
| **(gtifile)** = NONE [string] | Name of goodtime interval file. |
| **(ecol)** = PI [string] | Column name to use for the energy value. |
| **(weighted)** = INDEF [string] | Set to boolean yes or no,depending on if mask weighting should be applied to binning operation. |
| **(outunits)** = INDEF [string] | Set to RATE, COUNTS or INDEF, whether the output histogram units should be "count/s", "count", or automatically determined (INDEF). |
| **(maskwt)** = NONE [string] | Mask weight map file. |
| **(tstart)** = INDEF [string] | Start time of the accumulation, in seconds, expressed in Mission Elapsed Time (MET). |
| **(tstop)** = INDEF [string] | Stop time of the accumulation, in seconds (MET). |
| **(snrthresh)** = 6.0 [real] | For the constsnr binning method, the signal to noise ratio to require for each time bin. |
| **(detmask)** = "NONE" [string] | Name of a detector quality map file. |
| **(tcol)** = "TIME" [string] | Name of TIME column in input file. |
| **(countscol)** = "DPH_COUNTS") | Name of counts column column in input file |
| **(xcol)** = "DETX" [string] | Name of X column in input file (event data). |
| **(ycol)** = "DETY" [string] | Name of XYcolumn in input file (event data). |
| **(maskwtcol)** = "MASK_WEIGHT" | Name of MASK_WEIGHT column (event data). |
| **(buffersize)** = 32768 [integer] | Size of internal event buffer for processing. |

**Further discussion of selected parameters for batbinevt:**

WEIGHTED or UNWEIGHTED? (parameter: *weighted*)

Applying mask weighting is equivalent to background subtraction. Detectors which are fully shadowed are assigned a -1 weight. Fully illuminated detectors a +1 weight, and partially illuminated detectors are assigned an prorated value. Thus, a weighted sum of the counts will automatically subtract the background.

Each output format, by default, is either weighted or not.

* light curves and spectra are weighted by default;
* histograms and detector images are not weighted by default.

Users can choose to change the default by setting *weighted* to "yes" or "no."

For event data, the default is to take the mask weighting values for each event from the MASK_WEIGHT column. This can be overridden by using the *maskwt* parameter, which gives a mask weight map to be used for all events. [This parameter should not be used for event data taken during slews.] For DPHs, the mask weighting is also specified using the *maskwt* parameter.

TIME BINNING (parameter: *timebinalg*)

The user can choose how to bin the data in time. For "uniform" binning, a non-zero time bin size indicates that every time bin should have this same size. A zero time bin size indicates that all input data should be summed into a single output time bin.

For "gti" binning, the user specifies the name of a file containing the desired good time interval bins using the *gtifile* parameter, which is in the standard GTI format. Adjoining good time intervals are not merged when using this method. The *timedel* bin size is ignored.

For "snr" binning, the user specifies a desired signal-to-noise ratio with the *snrthresh* parameter. When the total signal to noise ratio for a given time bin exceeds the threshold, a new bin is started. The *timedel* bin size is taken as the maximum bin size before a new bin is started; if timedel=0 then there is no maximum.

The "infile" binning method only applies when the input is a DPH. When the binning algorithm is set to "infile," then the time binning of the input file is preserved in the output file. This may be useful, for example, if each DPH row is to be flattened into a detector image, while otherwise preserving the individual exposures.

TIME SELECTION (parameters: *tstart, tstop, gtifile*)

By default, the input file is selected according to its own internal good time intervals, if any are present.

Crude time selection can be performed using the *tstart* and *tstop* parameters to the task, which give the start and stop times of accumulation.

More refined selections can be performed specifying the *gtifile* parameter. A good time interval file (GTI) describes an arbitrary number of intervals by specifying the start and stop times. The intersection of the input files' GTIs, the user-provided gtifile, and the user-provided tstart/tstop parameters, are used to select input data by time.

## Examples (batbinevt):

### 1. Accumulate detector plane images

To calculate the final DPIs for different times and energy bands, the appropriate **batbinevt** command is:

```
batbinevt infile.evt outfile.dpi DPI 0 u energybins = "15-25,25-
50,50-100,100-350"  detmask=qual_map ecol=ENERGY weighted=NO
outunits=COUNTS
```

where `infile.evt` is the name of the input event file `(event_rw/sw00100139000bevshpsuf.evt) outfile.dpi` is the name of the output detector plane image file `(event_rw/output/sw00100139000_4.dpi)` and *qual_map* = `event_rw/output/ sw00100139000.mask`. To make this dpi file we type:

```
my_computer> batbinevt event_rw/sw00100139000bevshpsuf.evt
event_rw/output/sw00100139000_4.dpi DPI 0 u energybins = "15-
25,25-50,50-100,100-350" detmask =
event_rw/output/sw00100139000.mask ecol=ENERGY weighted=NO
outunits=COUNTS
****************************************
         batbinevt v1.5
----------------------------------------
    Input Events: event_rw/sw00100139000bevshpsuf.evt
    Output File: event_rw/output/sw00100139000_4.dpi  (DPI)
    Detector Mask: event_rw/output/sw00100139000.mask
    Time Range: 108578897.000000 to 108579231.691780 (requested)
    Time Range: 108578897.000000 to 108579231.691780  (actual)
    Apply Weighting?: NO      Energy Column: ENERGY
    Energy Bins: 15-25,25-50,50-100,100-350
    Output Units: COUNTS
```

```
    Binning Method: UNIFORM
    Time Bin Size: 0.000000 (s)
-----------------------------------------
  DPIs written to event_rw/output/sw00100139000_4.dpi
  EBOUNDS written to event_rw/output/sw00100139000_4.dpi
    Number of Rows Processed: 281584
    Number Accepted/Rejected: 245047/36537
  Time Bins:          1      Energy Bins:    4
-----------------------------------------
my_computer>
```

and a new file is created in the `event_rw/output` directory: `sw00100139000_4.dpi` which has four BAT_DPI extensions corresponding to the four energy bins we defined. There is also an EBOUNDS extension that gives us the energy bin edges.

Note: the “bevshpsuf” file name code indicates that this event file contains the data taken before the slew. Similarly, the file name codes “bevshsluf” and “bevshpouf” indicate data taken during the slew and after the slew, respectively. Thus if you also wanted to make DPI’s for data taken during the slew and post-slew, you would use `infile.evt = event_rw/sw00100139000bevshsluf.evt` and `infile.evt = event_rw/sw00100139000bevshpouf.evt` in your call to **batbinevt**.

### 2. Extract Spectra

With an appropriate choice of parameters, **batbinevt** can be used to extract spectra from an event file. When the *outtype* parameter is set to “PHA,” a single spectrum is produced for each specified time interval. The output pha file is a standard OGIP spectrum file (type II; default: *weighted*=yes, outunits=RATE.)

By default, spectra are created using mask weighting which is equivalent to background subtraction. This is described in the first section of this document. The time interval(s) used for making spectra are, by default, selected according to the input file’s own internal good time intervals, if any are present.

The basic form of **batbinevt** used to create spectra from event data is given below:

```
batbinevt infile.evt outfile.pha PHA timedel=0 timebinalg=u
energybins=ebounds.file
```

where `infile.evt` is the input event file (for example `infile.evt = event_rw/sw00100139000bevsh psuf.evt`) and we’ll define `outfile.pha` as `event_rw/output/sw00100139000_preslew.pha`. If time binning is uniform (i.e. *timebinalg*=u), then a value of *timedel*=0.0 causes the tool to accumulate all data into a single time bin for each selected time interval.

The energy bins can be specified in the command line by either an ASCII list or by setting *energybins*=ebounds.fits where the energy bins are given by the EBOUNDS

extension in ebounds. For example, if the file ebounds.file contains an 80 row EBOUNDS HDU, then the output would be an 80-channel PHA file with a single time bin.

ASCII energy bin edges can be written on the command line as in this example:

```
my_computer>: batbinevt event_rw/sw00100139000bevshpsuf.evt
event_rw/output/sw00100139000bevshpsuf.pha PHA 0 u 15-20,20-25,25-
30,30-35,35-40,40-45,45-50,50-55,55-60,60-65,65-70,70-75,75-80,80-
85,85-90,90-95
*****************************************
         batbinevt v1.5
-----------------------------------------
    Input Events: event_rw/sw00100139000bevshpsuf.evt
    Output File: event_rw/output/sw00100139000bevshpsuf.pha  (PHA)
    Detector Mask: NONE
    Time Range: 108578897.000000 to 108579231.691780  (requested)
    Time Range: 108578897.000000 to 108579231.691780  (actual)
    Apply Weighting?: YES     Energy Column: ENERGY
    Energy Bins: 15-20,20-25,25-30,30-35,35-40,40-45,45-50,50-
55,55-60,60-65,65-70,70-75,75-80,80-85,85-90,90-95
    Output Units: RATE
    Binning Method: UNIFORM
    Time Bin Size: 0.000000 (s)
-----------------------------------------
  Spectrum written to event_rw/output/sw00100139000bevshpsuf.pha
  EBOUNDS written to event_rw/output/sw00100139000bevshpsuf.pha
    Number of Rows Processed: 281584
    Number Accepted/Rejected: 133601/147983
    Time Bins:          1      Energy Bins:   16
-----------------------------------------
my_computer>
```

A more elegant way to enter energy bin edges is to supply a FITS file with an EBOUNDS extension that contains the energy bin edges. The format of this extension is 3 columns x N rows (N being the number of energy bins.) The columns have the following names: CHANNEL, EMIN, and EMAX and contain the row number, the minimum energy for the bin, and the maximum energy for the bin. Suppose the file “ebounds.fits” with 80 bin edges has been placed in the bat subdirectory

```
batbinevt event_rw/sw00100139000bevshpsuf.evt
event_rw/output/sw00100139000bevshpsuf_2.pha PHA 0 u
energybins=ebounds.fits
*****************************************
         batbinevt v1.5
-----------------------------------------
    Input Events: event_rw/sw00100139000bevshpsuf.evt
    Output File: event_rw/output/sw00100139000bevshpsuf_2.pha(PHA)
    Detector Mask: NONE
```

```
    Time Range: 108578897.000000 to 108579231.691780  (requested)
    Time Range: 108578897.000000 to 108579231.691780  (actual)
    Apply Weighting?: YES     Energy Column: ENERGY
    Energy Bins: ebounds.fits
    Output Units: RATE
    Binning Method: UNIFORM
    Time Bin Size: 0.000000 (s)
----------------------------------------
  Spectrum written to event_rw/output/sw00100139000bevshpsuf_2.pha
  EBOUNDS written to event_rw/output/sw00100139000bevshpsuf_2.pha
    Number of Rows Processed: 281584
    Number Accepted/Rejected: 278504/3080
  Time Bins:          1      Energy Bins:   80
----------------------------------------
```

### 3. Extract light curves

Although we've already used the **batbinevt** tool to extract spectra from event files and to make detector plane images and histograms, **batbinevt** can also be used to extract light curves from event files. When the outtype parameter is set to "LC," a BAT light curve is generated for the energy intervals and time binning as specified by the user. The form to use for generating light curves is:

```
batbinevt infile.evt outfile.lc outtype= LC timedel timebinalg
energybins [detmask = quality.mask]
```

By default, the time interval for the light curve is selected according to the input file's internal good time intervals, if any are present. Crude time interval selection can be performed using the *tstart* and *tstop* parameters, which give the start and stop times of accumulation. **Batbinevt** also allows the user to specify his own time intervals by setting the parameter *gtifile* = newgtifile.fits. The time binning for the light curve within the selected time interval(s) is controlled by the *timedel* and *timebinalg* parameters. The user can select the time bin size in seconds with the *timedel* parameter. Note that If time binning is uniform, then a value of *timedel*=0.0 causes the tool to accumulate all data into a single time bin. This is fine if you're extracting spectra, but it will give you a very boring light curve with only one bin! *Timebinalg* defines the mode for time binning. If *timebinalg* = uniform (or u) the chosen time interval(s) are divided into bins of equal duration. There are other time binning modes such as "snr", for a constant signal to noise ratio and "gti" for time binning according to the intervals in the user specified GTI file. See the **batbinevt** help page for more detailed information.

Energy bin ranges are specified by the *energybins* parameter and can be expressed as either a comma-separated list of floating point number ranges, a file name containing energy bin ranges, or CALDB. A value of *energybins* = "-" indicates one all-inclusive energy bin. At least one energy bin must be given to make a light curve, and bins may not overlap. If a file name is given, it can either be an ASCII file containing the same comma-separated energy bin list, or a FITS file with an EBOUNDS

extension (such as a response matrix), containing columns E_MIN and E_MAX (in keV). If CALDB is specified, then the CALDB database is consulted for energy bins.

To eliminate the contribution of noisy pixels to the light curve, we can supply **batbinevt** with a quality map that describes which detectors are enabled. In this example, we'll set *detmask* = event_rw/output/sw00100139000.mask.

As with spectra, the default procedure is to create light curves using mask weighting. By using mask weighting, we are insuring that the light curve produced will only represent photons coming from the specified source position. As we discussed when using **batbinevt** to produce spectra, mask weighting is equivalent to background subtraction.

To make a 1 second time bin light curve for the four energy bands 15-25 keV, 25-50, keV, 50-100 keV, and 100-350 keV, from our example file event_rw/sw00100139000bevshpsuf.evt we type:

```
my_computer>: batbinevt event_rw/sw00100139000bevshpsuf.evt
event_rw/output/sw00100139000_pre_slew_1s.lc  LC  1.0 u 15-25,25-
50,50-100,100-350   detmask = event_rw/output/sw00100139000.mask
*****************************************
         batbinevt v1.5
-----------------------------------------
    Input Events: event_rw/sw00100139000bevshpsuf.evt
    Output File: event_rw/output/sw00100139000_pre_slew_1s.lc (LC)
    Detector Mask: event_rw/output/sw00100139000.mask
    Time Range: 108578897.000000 to 108579231.691780  (requested)
    Time Range: 108578897.000000 to 108579231.691780  (actual)
    Apply Weighting?: YES     Energy Column: ENERGY
    Energy Bins: 15-25,25-50,50-100,100-350
    Output Units: RATE
    Binning Method: UNIFORM
    Time Bin Size: 1.000000 (s)
-----------------------------------------
  Light curve written to
event_rw/output/sw00100139000_pre_slew_1s.lc
  EBOUNDS written to event_rw/output/sw00100139000_pre_slew_1s.lc
    Number of Rows Processed: 281584
    Number Accepted/Rejected: 245047/36537
  Time Bins:        335      Energy Bins:    4
-----------------------------------------
```

**4. Produce a light curve from an event file.** The default is to use the PI column in the event file for energy and the MASK_WEIGHT column to produce a mask weighted light curve. If the parameter *ecol* = PHA, then the PI column can be absent. Similarly if *weighted* = no, then the MASK_WEIGHT column can be absent. There must be a GTI extension.

```
batbinevt infile.evt outfile.lc outtype=LC timedel=1 timebinalg=u
energybins=25-50,50-100
```

This would produce an output light curve with one second, uniform time binning in two energy channels. An EBOUNDS and a GTI extension are produced.

**5. Produce a spectrum from an event file.**

```
batbinevt infile.evt outfile.lc outtype=PHA timedel=0 timebinalg=u
energybins=ebounds.file
```

If the file ebounds.file contains an 80 row EBOUNDS HDU, then the output would be an 80-channel PHA file with a single time bin. If *timedel*=1, then it would produce a PHA-II file with one-second time binning.

**6. Produce a DPH from an event file.**

```
batbinevt infile.evt outfile.lc outtype=DPH timedel=0 timebinalg=u
energybins=ebounds.file
```

This would produce a 80 energy channel DPH from the full time range of the event data.

**7. Produce a DPI from a DPH.**

```
batbinevt infile.dph outfile.dpi outtype=DPI timedel=0
timebinalg=u energybins=-
```

This would produce a DPI using all energy levels in the DPH.
Other options for timebinalg are described in the help file for **batbinevt**.

See the online help for **batbinevt** for many additional examples.

# B. Batcelldetect

This tool is used to find sources in a BAT sky image (as produced by **batfftimage**). It finds sources using a sliding "cell" method (hence the tool name) and when a source is found above the user specified threshold, then it fits a two-dimensional Gaussian function to derive a count value for the source.

This tool is appropriate for coded aperture imaging because: (1) it assumes Gaussian fluctuations, not Poissonian; (2) it measures the local background; and (3) it measures the local background standard deviation. A source is detected at a pixel if that pixel's value exceeds the background by more than *snrthresh* times the background standard deviation. Users can increase the significance of a detection and reduce false detections by requiring more than one adjacent pixel exceed the threshold, using the *nadjpix* parameter.

The background value is estimated by using a sliding window. The shape of the window is either circular or square, and the radius (or half-width) is specified by the *bkgradius* parameter. In determining the background, a circular region at the center of the window is excluded, whose radius is the *srcradius* parameter. Thus, the background does not include contamination from the source region.

In a single iteration, the sliding cell algorithm is less sensitive in a region around bright sources, because the background standard deviation becomes biased. To avoid this, the algorithm can be run in multiple iterations. After each iteration, the detected pixels are removed, and thus the bias can be significantly reduced.

A second stage fits a gaussian point spread function to regions of the image where sources are detected. This aids in refining the centroid of the source position, as well as in estimating uncertainties.

The default output is a catalog list of detected sources, plus various statistics about them. The tool can optionally output the background map, the background fluctuation map, and a significance map.

The user can also supply an input catalog. Sources in the input catalog, which are within the field of view of the image, are assumed to be fixed at their known positions, and fitted for intensity only during the PSF-fitting stage .

This routine uses a relatively straightforward brute-force convolution by FFT to compute the sliding-cell averages. Users should have a about a factor of eight more memory than the size of the image.

The simplest invocation is simply

```
batcelldetect infile.img outfile.cat snrthresh=6.0
```

The output is a file which lists, for each source found above a signal to noise ratio of 6.0, information including source location, counts, significance, background, and diagnostics about the search. The source location is reported in a number of formats

and coordinate systems including celestial (RA and declination), BAT image coordinates (tan_x and tan_y) and pixel coordinates.

It is also possible to input a source catalog. In this case the tool derives a count value for each source in the catalog in addition to those found using the search algorithm.

## Input Files (batcelldetect):

Hidden parameters are listed in parentheses. If these are required, then the code will use default values.

Required:

*infile:* Input file name containing a sky map.

Optional:

*(incatalog):* Name of input catalog of known sources. This catalog must have at least the columns RA and DEC, in degrees (J2000). Additional columns are also copied to outfile. Sources from this catalog which are in the field of view are held at a fixed position in order to determine the flux. A value of "NONE" indicates that no a priori catalog should be used. Specifying this keyword does not prevent the cell-detection based source detection algorithm from being applied.

*(pcodefile):* Name of optional partial coding map file. This may be any image with exactly the same dimensions and coordinate system as *infile*. Any region where the partial coding map exceeds *pcodethresh* is searched for sources; any region where it is below *pcodethresh* is ignored. The map need not be an actual partial coding map; any quantity that can be thresholded (even a map of 0s and 1s) can be used. If *pcodefile* is "NONE", then the entire image is searched.

## Output Files (batcelldetect):

Required:

*outfile:* Output source list.

Optional:

*(regionfile):* Optional name of source detection region file. Upon output, sources listed in the outfile catalog are also written to a standard "region" file, which can then be read into image display programs such as fv/POW or SAO ds9. Output units are degrees. The radius of the circle is twice the PSF gaussian sigma radius.

*(signifmap):* Optional output file name for the significance map.

(b*kgmap):* Optional output file name for the mean background map.

*(bkgvarmap):* Optional output file name for the background fluctuation map.

## Parameters (batcelldetect):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

The table is found on the next page.

| Parameter / Default / Data Type | Description of batcelldetect parameters |
|---|---|
| **infile** [filename] | Input file name containing a sky map. |
| **outfile** [filename] | Output source list. |
| **snrthresh** [real] | Signal to noise threshold for detection of sources. A value of 6.0 indicates that an excess must be 6 sigma above the background level to be a detection. |
| **(incatalog)** =NONE [string] | Name of input catalog of known sources. This catalog must have at least the cols RA and DEC, in degrees. Additional columns are also copied to outfile. |
| **(pcodefile)** =NONE [string] | Name of optional partial coding map file. |
| **(pcodethresh)** =0.01 [real] | Threshold to apply to the map in specified in pcodefile. |
| **(pospeaks)** =YES [boolean] | Constrain newly detected peaks to be positive. |
| **(niter)** =2 [integer] | Number of iterations of source detection to perform. |
| **(nadjpix)** =4 [integer] | Minimum number of adjacent pixels required for source detection. |
| **(bkgwindowtype)** ="circle" | Background window type, one of "circle" or "square". |
| **(bkgradius)** =30 [integer] | Background window radius in pixels. |
| **(srcradius)** =6 [integer] | Source radius in pixels. |
| **(regionfile)** =NONE [string] | Optional name of source detection region file. |
| **(srcdetect)** =YES [boolean] | Whether to attempt to detect new sources. |
| **(srcfit)** =YES [boolean] | Whether to fit newly detected and previously cataloged sources (if any) |
| **(posfit)** =NO [boolean] | Specifies whether batcelldetect fits the position of already known sources. |
| **(newsrcname)** ="UNKNOWN" | Name to be used for newly detected sources. |
| **(newsrcind)** =1 [integer] | Start index number to be used to label new sources. |
| **(signifmap)** =NONE [string] | Optional output file name for the significance map. |
| **(bkgmap)** =NONE [string] | Optional output file name for the mean background map. |
| **(bkvarmap)** = NONE [string] | Optional output file name for the background fluctuation map. |
| **(npixthresh)** =20 [integer] | Minimum number of pixels in the background window to enable source detection. |

## Examples (batcelldetect):

### 1. Refine the BAT GRB position

To use **batcelldetect** to refine a gamma ray burst position, we start by just having it find all the sources in the sky image. The basic form for using the **batcelldetect** tool is:

```
batcelldetect infile.img outfile.src snrthresh incatalog=NONE
pcodefile
```

where infile.img contains a sky image. Here we'll use the FITS sky image (pre-slew total band) made in the **batfftimage** recipe: event_rw/output/sw00100139000_1.dpi. The outfile.src file is an ASCII source list that **batcelldetect** produces. We'll use event_rw/output/source_list_1.txt. The parameter *snrthresh* is a real number representing the minimum signal to noise threshold for the detection of sources. For example, a value of 6.0 indicates that the excess must be 6 sigma above the background level to be considered a detection. For performance reasons, users should choose snrthresh > 3.5. Here, let's set *snrthresh* = 8. The *incatalog* parameter should be set to "NONE" (default). A value of "NONE" indicates that no a priori catalog should be used. Specifying this keyword does not prevent the cell-detection based source detection algorithm from being applied. Finally, the *pcodefile* parameter is the name of the appropriate partial coding map for this observation. Here we will use the map we made in the **batfftimage** recipe: *pcodefile* = event_rw/output/pcodemap.img.

To run **batcelldetect** this first time, we type:

```
my_computer>: batcelldetect event_rw/output/sw00100139000_1.dpi
event_rw/output/source_list_1.txt 8.0 incatalog=NONE pcodefile=
event_rw/output/pcodemap.img
*****************************************
# batcelldetect v1.6

-----------------------------------------
Input Image: event_rw/output/sw00100139000_1.dpi
Output Catalog: event_rw/output/source_list_1.txt
Input Catalog: NONE
Part. Coding Map: event_rw/output/pcodemap.img
(threshold=0.010000)
Back. Window: CIRCLE Radius: 30
Source Window: CIRCLE Radius: 6
SNR Threshold: 8.000000
Number of Iter.: 2
Min. Num. Pixels: 20
-----------------------------------------
Found 1 Images
Analyzing Image: 1
...read tanxy WCS keywords from image (suffix "T")...
```

```
IMX -- CRPIX=0.000000 CDELT=19.199402 CRVAL=0.000000
IMY -- CRPIX=7.474824 CDELT=19.051758 CRVAL=0.000000
...read def WCS keywords from image (suffix "")...
(X -- CRPIX=0.000000 CDELT=19.199402 CRVAL=0.000000)
(Y -- CRPIX=7.474824 CDELT=19.051758 CRVAL=0.000000)
ERROR: could not read image keywords
CFITSIO ERROR KEY_NO_EXIST: keyword not found in header
Task batcelldetect 1.6 terminating with status 202
```

Note to user: this obviously didn't run correctly. There was no time for further corrections before this draft was produced. Consult BAT Burst Advocate trainers for more up-to-date information.

# C. Batclean

This tool is used to clean out a background model and bright sources (from a catalog file) from a DPI. Currently the code operates only on a single DPI (not a DPH).

The tool currently incorporates a simple time-independent background model, which treats detectors on the edges of DMs differently from detectors in the center (due to their having greater projected surface area). The fitting method is a simple singular value decomposition. The fit to the background is subtracted from the detector plane.

To clean sources the tool uses the same forward projection method used in **batmaskwtimg** to produce a model of the source flux on the detector plane. This model is fit to the detector plane and subtracted. It is possible to clean the background only, by setting *incatalog*=NONE.

Source Cleaning:

If one or more sources are to be cleaned from the DPI (default), then the tool reads the source locations from the *incatalog* file and then calls the routine maskwtimg which uses a forward projection to create a model of the source exposure through the aperture onto the detector plane.

Background Cleaning:

The tool creates a simple eighteen parameter background model, which is described as follows:

1 = constant, 2 = proportional to X, 3 = proportional to Y
4 = proportional to X^2, 5 = proportional to Y^2
6 = proportional to X^Y,
7 = Detector on left (-X) side of sandwiches: constant,
8 = Left side detectors: proportional to X,
9 = Detector on right (+X) side of sandwiches (sws): constant,
10 = Right side detectors: proportional to X,
11 = Dets on front (-Y) side of sws (front half of array): const,
12 = Front side detectors (front half): proportional to Y,
13 = Dets on front (-Y) side of sws (back half of array): const,
14 = Front side detectors (back half): proportional to Y,
15 = Dets on back (+Y) side of sws (front half of array): const,
16 = Back side detectors (front half): proportional to Y,
17 = Dets on back (+Y) side of sws (back half of array): const,
18 = Back side detectors (back half): proportional to Y,

The code creates a background exposure map based on the model above (which can be optionally output using the *backexp* parameter).

The tool then uses a singular value decomposition method to fit the source exposure and background exposure maps to the data from the DPI. The coefficients from the fit are written to the output file as keywords.

The fit coefficients are then used to create a model background DPI. This is then subtracted from the original input DPI to produce a cleaned DPI.

## Input Files (batclean):

This code reads in a Detector Plane Image (DPI) file as well as an optional detector mask file. If no detector mask file is provided, then all detectors with non-zero data are included in the fit. The tool works on either flight or simulated flight data (sources at infinity) or ground calibration data (sources at a finite distance). See the description of the *bat_z* parameter for more about cleaning of sources at a finite distance.

Hidden parameters are listed in parentheses. If these are required, then the code will use default values.

Required:

*infile:* Input file name. The name of the input DPI file. This is expected to be an image array with dimensions equal to the number of detector columns by the number of detector rows (set to 286 X 173)

*incatalog:* (optional if *srcclean* = “NO”) Input source catalog. The format of the catalog file is that output by the BAT tool **batcelldetect**. In particular the catalog must contain columns "IMX," "IMY," "SOURCE_ID," "NAME," and "SNR." The tool reads the source position in BAT image coordinates from this file and uses the source position to forward project a model of the source flux onto the focal plane. This parameter may be omitted or set to "NONE" if only background cleaning is required (i.e., the *srcclean* parameter set to "NO").

(*aperture):* (optional if *srcclean* = “NO”) Aperture file. The default is the aperture file in the HEADAS/refdata area. This parameter may be omitted or set to "NONE" if only background cleaning is required (i.e., the *srcclean* parameter set to "NO").

Optional:

*(detmask):* Name of a detector mask file. This should be an image file with the same dimensions as the focal plane map. A pixel value of 0 indicates the detector is enabled for imaging, and a value of 1 indicates disabled, noisy, or otherwise selected for elimination from fits. A default value of NONE implies all detectors are on, except for the BAT detector gap regions. It is

strongly recommended to use a detector mask file if one is available. The quality of the fit, especially to calibration data is greatly improved if only working detectors are included in the fits.

*(wtmapin):* Name of an input error map. This is used to provide weights for the fitting in case one is cleaning a DPI which does not have Poisson statistics (such as an already cleaned DPI). The typical sequence would be to create an error map using the *wtmapout* parameter on the first cleaning and then read this error map in on the second cleaning. If an already cleaned image is cleaned without an input error map than uniform weighting is used.

## Output Files (batclean):

Required:

*outfile:* Output file name. This will be written in the DPI format. The default is to write out the cleaned image, but other outputs can be selected using the *outversion* parameter. Precede the output file name with an exclamation point, !, (or \!on the Unix command line), to overwrite a preexisting file with the same name (or set the *clobber* parameter to YES).

Optional:

(*wtmapout):* Name of an output error map. If a filename is specified then the tool writes out a DPI containing the calculated weights for each detector. This can be used on subsequent cleanings as the input wtmapin error map.

*(backexp):* If the name of a file is given here, then the background model will be output to this file. The format is a binary table with one column for each model parameter. The data is written in DPH format, or as a 286 X 173 element array, one number per detector. This table gives the model which was fit to the data, not the actual fit to the data. The final column in the table is the covariance matrix from the fit.

## Parameters (batclean):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

The table is found on the next page.

| Parameter / Default / Data Type | Description of batclean parameter |
|---|---|
| **infile** [filename] | Input file name. |
| **outfile** [filename] | Output file name. |
| **incatalog** [filename] | Input source catalog. |
| **(aperture)** [filename] | Aperture file. |
| **(detmask)** = "NONE" [filename] | Name of a detector mask file. |
| **(wtmapin)** = "NONE" [filename] | Name of an input error map. |
| **(wtmapout)** = "NONE" [filename] | Name of an output error map. |
| **(backexp)** = "NONE" [filename] | The background model will be output to this file. |
| **(bat_z)** = 0 [real] | The z-component of the distance to the source in BAT_Z coords (the vertical distance of the source from the origin of the BAT coordinate system). |
| **(bkgmode)** = "simple" [string] | This optional parameter indicates the background model to be fit. The only model currently supported is “simple.” |
| **(srcclean)** = "YES" [string] | Determines if sources are cleaned from the input DPI or only background. |
| **(leadedge)** = "NO" [string] | Leading edge detectors are fit separately from the rest of the detectors. |
| **(eff_edge)** = 0.2 [float] | The depth of edge (cm) to consider effective. |
| **(cleansnr)** = 6.0 [real] | Only sources with a signal-to-noise ratio as read from the incatalog higher than this value are cleaned from the DPI. |
| **(corrections)** = "none" [string] | Comma-separated list of corrections to apply to the mask weighting:<br>"flatfield" (basic flat fielding correction)<br>"cosine" (cosine effects of off-axis illumination)<br>"rsquare" (r-squared effects only)<br>"opaque" (activates a new algorithm for calculating the mask transmission function);<br>"sides" (effects of sides of dets not implemented). |
| **(outversion)** = "cleaned" [string] | what detector plane image is written to the outfile:<br>“cleaned”: cleaned DPI (input minus fit).<br>“original”: Write the input DPI filtered by det mask.<br>“fit”: Write the fit DPI.<br>“bkgcleaned”: Write the background cleaned DPI.<br>“bkgfit”: Write the fit to the background only. |

## Examples (batclean):

**1. Fit the simple background model** to a DPI with a detector mask file. Output the cleaned DPI and a background exposure map.

```
batclean input.dpi cleaned.dpi detmask=bat.mask
backexp=backexp2.dph srcclean=NO
```

**2. Fit a set of source models along with a simple background model** to a DPI, using the default aperture file. Output the fit to the focal plane instead of the cleaned focal plane. A flat-fielding correction is applied in the forward projection.

```
batclean input.dpi fit.dpi catalog.tbl detmask=bat.mask
outversion="fit"corrections="flatfield"
```

# D. Batdph2dpi

This is a simple tool which collapses the energy dimension of a DPH to produce a DPI. This functionality is also present in **batbinevt**, but this tool remains as a simpler way to accomplish this. The only options to the tool are the row number in the DPH file and the energy ranges in the DPH to be collapsed. Only one row in a DPH can be selected at a time.

## Input Files

Required:

*infile:* Input file name.

## Output Files

Required:

*outfile:* Output file name.

## Parameters

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

| Parameter / Default / Data Type | Description |
|---|---|
| **infile** [filename] | Input file name. |
| **outfile** [filename] | Output file name. |
| **rows** | Number of the row from which the data are taken for flattening the image. |
| **levels** | Ranges of energy levels that are used in flattening. |

## Example:

```
batdph2dpi infile.dph outfile.dpi 1 -
```

This would collapse all energy levels of the first row of the input file into a DPI.

# E. Batdph2pha

This tool extracts from a DPH one histogram for each detector in the array. The output is a table with 32768 rows, each containing a histogram with the binning from the original DPH. The EBOUNDS extension is copied to the output file.

This tool is necessary because there is no way to easily visualize the individual spectra using only standard FITS tools.

## Input Files

Required:

*infile:* Input file name. There is no need to include the extension name of the HDU since batdph2pha only operates on an HDU with name "BAT_DPH." The tool assumes that there is also and "EBOUNDS" extension in the input file.

## Output Files

Required:

*outfile:* Name of the output file. The output file is similar in form to a spectral file, but each row contains a spectrum for an individual detector, not for a particular interval of time. The spectra are contained in the "COUNT" column as vectors with as many elements as there are energy bins in the input DPH. The "EBOUNDS" from the input file is copied to the output file.

## Parameters

There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

| Parameter / Default / Data Type | Description |
|---|---|
| **infile** [filename] | Input file name. |
| **outfile** [filename] | Name of the output file. |
| **row** [integer] | The row number in the input DPH which is used to produce the output spectra. |

## Example:

```
batdph2pha infile='input.dph' outfile='output.spectrum' row=3
```

This produces a set of 32768 spectra from row 3 in the DPH.

# F. Batdrmgen

**Batdrmgen** is the BAT Detector Response Matrix (DRM) generator tool that computes the full BAT instrument response to incident photons, given the source position information read from an input PHA spectral file. The output FITS file contains a matrix that represents the mask-weighted summed response of all the active pixels in the entire BAT array. Since BAT is a coded-aperture hard x-ray imager, it is difficult to separate its response into the ARF and RMF files that are familiar to many FTOOLS users. Since the mask-weighted sum of the spectra from all of the active pixels is equal to the total background-subtracted source count rate measured by the BAT, this output DRM is the only file needed by XSPEC to perform spectral analysis of the BAT PHA data files.

The **batdrmgen** tool follows **batbinevt** in the burst spectroscopy analysis pipeline. The batbinevt output for a particular gamma-ray burst or other source is a FITS PHA spectrum file that becomes the input to batdrmgen. This input PHA file contains a full-array counts spectrum, the energy bin edges, and source position information. The **batdrmgen** tool will produce a DRM for only one source position (PHA file) at a time.

For each incident photon energy bin, batdrmgen computes the BAT counts spectrum that would be measured if a mono-energetic stream of photons of unit photon flux with an energy at the bin midpoint were incident on the array. Repeating this calculation for each incident energy bin produces an N x M matrix, where N is the number of incident photon bins and M is the number of PHA counts bins. While the incident photon energy bin edges can be defined by the user, the output PHA counts spectrum bin edges are read from the EBOUNDS extension in the input PHA file.

The BAT DRM is calculated using the set of quasi-physical calibration parameters that are appropriate for the particular position of the photon source. These parameters were determined from least squares fits of the spectral model to actual ground calibration data measured using radioactive gamma-ray sources that were mounted in a variety of positions within the BAT field of view (FOV). The calculation also makes use of the detector charge trapping parameters measured during ground calibration tests. The angle dependent calibration parameters and the detector charge trapping parameters are stored in a calibration file managed by the CALDB utility.

Since **batdrmgen** models the signal loss due to charge trapping at different absorption depths within the CZT detectors, it requires a knowledge of the interaction depth probability distributions for 10-500 keV photons coming from anywhere within the BAT FOV. Simple Monte Carlo simulations of the absorption of 10-500 keV photons in a 2 mm thick CZT detector were used to create 1000-element tables of the probabilities of a photon interacting in each of the 2 mm/1000 = 2 micron thick CZT detector "slices." These 1000-element depth distribution vectors were determined over a moderately dense grid of energies and incident angles within the BAT FOV. The depth distribution used for a particular source

position and energy is determined by interpolating within this table. This depth distribution information is highly compressed and stored in a second calibration file also managed using CALDB.

Users have the choice of 2 methods for DRM generation: The default method, "TABLE," accesses the table of calibration parameters for different source positions and sorts the table with respect to the angular separation between the source direction vector and the position vector for each ground calibration measurement. **Batdrmgen** then calculates the weighted average of the parameter values from the three closest calibration measurements. Thus, the ground calibration parameters from measurements closer to the source position will be weighted more heavily. The alternative method is "MEAN," where the spectral response is calculated using the average parameter values over the entire BAT FOV.

## Input Files (batdrmgen):

Hidden parameters are listed in parentheses. If these are required, then the code will use default values.

Required:

*infile:* Name of the PHA FITS input file containing the binned and mask-weighted BAT data as well as header keywords containing the source position and other information relevant to the response calculation.

*(calfile):* Name of the FITS file that contains the spectral parameters for the response function. The default value is "CALDB" thus allowing the file to be managed by the CALDB utility.

*(depthfile):* Name of the FITS file that contains the compressed photon interaction depth distribution generated by Monte Carlo simulations. The default value is "CALDB" thus allowing the file to be managed by the CALDB utility.

*(efile):* File name for the user-specified FITS file containing the user's custom incident energy bin edges (used for escale = FILE as discussed above). The default value, "CALDB" indicates that the default incident spectrum will be used. This default spectrum has the energy range of 10 keV to 500 keV divided into 200 logarithmically-spaced bins with bin edge adjustment made for the absorption edges of CdZnTe (26.711 keV for the Cd K edge and 31.814 for the Te K edge). As indicated, the file containing the default input bin edges is managed by the CALDB utility.

Optional:

*(detmask:)* File name for the detector quality mask (with dimensions of 286x173) which indicates which detectors were used to accumulate the spectrum. (Not yet implemented)

## Output Files (batdrmgen):

Required:

*outfile:* Name of the output FITS response file (*.rsp) that will contain the N x M DRM in a format appropriate for use with XSPEC.

## Parameters (batdrmgen):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

The table is found on the next page.

| Batdrmgen Parameters | |
|---|---|
| **Parameter / Default / Data Type** | **Description** |
| **infile** [filename] | Input file name. |
| **outfile** [filename] | Output file name. |
| **(calfile)** = CALDB [filename] | File that contains the spectral parameters for the response function. |
| **(depthfile)** = CALDB [filename] | File that contains the compressed photon interaction depth distribution |
| **(escale)** = FILE [string] | Desired form of the incident photon energy spectrum binning.<br>"FILE:" program reads a file containing the bin edges<br>"LIN" for linear binning<br>"LOG" for logarithmic binning. |
| **(detmask)** = NONE [filename] | File name for the detector quality mask. |
| **(method)** = TABLE [string] | Computation method for DRM.<br>Default method ("TABLE") accesses the table of calibration parameters for different source positions.<br>Alternative method ("MEAN") the spectral response is calculated using the average parameter values over the entire BAT FOV. |
| **(flight_data)** = YES [boolean] | Flag indicating whether the PHA file contains flight data as opposed to ground calibration data. |
| **(nphoton_bins)** = 160 [integer] | The number of bins desired in the incident photon energy scale |
| **(elimit_lo)** = 15.0 [real] | The desired lower energy limit for the incident photon energy scale |
| (**elimit_hi)** = 200.0 [real] | The desired upper energy limit for the incident photon energy scale |
| **(efile)** = CALDB [string] | File name for the user-specified FITS file containing the user's custom incident energy bin edges |
| **(fudge)** = INDEF [string] | A string specifying which corrections should be applied to the response matrix calculation. |

## Examples (batdrmgen):

**1. Generate a BAT response matrix**
If you wish to do spectral analysis with `event_rw/output/sw00100139000_preslew.pha` using XSPEC, you will need to generate a BAT response matrix matching the source position and counts spectrum energy bin edges from `event_rw/output /sw00100139000_preslew.pha` . The tool **batdrmgen** computes the BAT detector response (RSP) for a mask weighted PHA spectrum, such as the PHA output of the **batbinevt** tool. The standard form of the use of batdrmgen is given below:

```
batdrmgen infile.pha outfile.rsp calfile depthfile
```

**batdrmgen** obtains the burst position information and the energy bin edge values from the input PHA FITS file infile.pha. For this example, let `infile.pha = event_rw/output/sw00100139000_preslew.pha` and `outfile.rsp = event_rw/output/sw00100139000_preslew.rsp`. The last two entries, *calfile* and *depthfile* refer to BAT calibration parameter tables that are usually managed by CALDB, so let's assume that the calfile and depthfile parameters are both "CALDB."

```
my_computer>: batdrmgen event_rw/output/sw00100139000bevshpsuf.pha
event_rw/output/sw00100139000bevshpsuf.rsp (chatter=2)
----------Begin task batdrmgen --------------
****************************************
         batdrmgen v2.1
----------------------------------------
         PHA File: event_rw/output/sw00100139000bevshpsuf.pha
      Output File: event_rw/output/sw00100139000bevshpsuf.rsp
 Calibration File: CALDB
Depth Distribution File: CALDB
    Detector Mask: NONE
         Method is: TABLE
         Escale is: FILE
----------------------------------------
SPECRESP MATRIX written to
event_rw/output/sw00100139000bevshpsuf.rsp
EBOUNDS written to event_rw/output/sw00100139000bevshpsuf.rsp
----------batdrmgen task complete------------
```

**2. Additional Example:**

```
batdrmgen test.pha response.rsp bat_parms-2003-08-19.fits
bat_depth_dist_030526.fits.gz
```

The output is an RMF file suitable for use in XSPEC to derive a photon spectrum from the counts spectrum in the input file. The number of output photon bins can be set by the user.

# G. Bateconvert

This tool reads in an event file and one to three calibration files containing gains and offsets in a BAT Detector Plane array. For each event it calculates the energy in keV from the ADU value (PHA), gain and offset. The tool fills the PI and ENERGY columns in the event FITS file. PI is the same as energy, but expressed in units of 0.1 keV. If the event file does not have a PI column, then such a column is added to the file.

The default is to fill in or create the PI column in the input file, but if the optional "outfile" argument is given, a new file with the PI column filled is created.

## Input Files

Required:

*infile:* Input file name. There is no need to include the extension name of the HDU since **bateconvert** only operates on an HDU with name "EVENTS." Unless an outfile name is specified, **bateconvert** will automatically fill in the PI column in the infile and/or create a PI column if none exists.

*calfile:* (optional if *zeroit* = ”YES”) Name of the calibration file.This file is expected to contain the gain and offset derived in flight. This file is used for both LINEAR and QUADRATIC energy conversion. The file must exist and must include a "BAT_MAP" extension with "GAIN" and "OFFSET," columns containing calibration parameters in the proper format.

*residfile:* (optional if *calmode* = “LINEAR” or *zeroit* = ”YES”) Name of the file containing the residuals between a quadratic pulser gain fitting and a linear fitting. This file is typically derived from ground processing. This file is used for the QUADRATIC energy conversion. The file must exist and must include a "BAT_MAP" extension with "GAIN2", "GAIN," "OFFSET" columns containing calibration parameters in the proper format.

## Output Files

Required:

*None.*

Optional:

*(outfile):* Output file name. If an output file name is given, **bateconvert** will create a new file which is a copy of the input file with the PI and ENERGY columns filled in. The input file will remain unchanged. Precede the output file name with an exclamation point, !, (or \!on the Unix command line), to overwrite a preexisting file with the same name (or set the clobber parameter to YES).

## Parameters

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

| Bateconvert Parameters | |
|---|---|
| **Parameter / Default / Data Type** | **Description** |
| **infile** [filename] | Input file name. |
| **calfile** [filename] | Name of the calibration file. |
| **residfile** [filename] | Name of the file containing the residuals between a quadratic pulser gain fitting and a linear fitting. |
| **pulserfile** [filename] | Name of the pulser calibration file. |
| **(outfile)** [filename] | Output file name. |
| **(calmode)** =QUADRATIC [string] | The calibration mode.<br>LINEAR fit applies two parameters, gain and offset using the formula:<br>ENERGY=GAIN*(OFFSET-PHA).<br>QUADRATIC fit performs the linear fit and then adds in a residual to correct for the deviation of the true fit from linearity. |
| **(zeroit)** =NO [boolean] | Set all values in the PI and ENERGY columns to zero |
| **(scaled_energy)** =YES [boolean] | If YES (default), then the ENERGY column is written as 16-bit integers with a TSCAL scale factor which converts to keV.<br>If NO, 32-bit floating point values are written directly. |

## Examples (bateconvert):

### 1. Linear energy conversion

```
bateconvert infile='infile.evt' calfile='calfile.cal'
residfile=NONE pulserfile=NONE calmode=LINEAR
```

This would create a PI column in the infile if none already exists and apply a linear calibration. If the optional outfile parameter is specified, then the infile is unchanged and copied to the outfile, to which a PI column is added.

**2. Quadratic energy conversion**

```
bateconvert events.fits cal.fits resid.fits pulser.fits
```

Since the default calmode is quadratic, this example would use the calibration files resid.fits and pulser.fits along with the gain/offset map cal.fits to apply a quadratic correction. The calibration files would typically be found in the calibration database, so “caldb” is an appropriate value for these parameters. There are two calibration files because they are derived from different calibrations and because they can change independently.

This tool is used in the SDC pipeline to process the event files.

# H. Baterebin

This tool uses similar algorithms to **bateconvert** and takes similar input files. Since the onboard energy calibration is linear, this tool is used to rebin the DPHs to correct for the difference between a linear calibration and a quadratic calibration.

## Input Files

Required:

*infile:* Input file name.

c*alfile:* Name of the file that contains the linear calibration constants

*residfile:* Name of the file that contains the quadratic residuals to the energy correction

*pulserfile:* Name of the file that contains the pulser calibration constants.

## Output Files

Required:

*outfile:* Name of the output file.

## Parameters

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

The table is found on the next page.

| Baterebin Parameters | |
|---|---|
| **Parameter / Default / Data Type** | **Description** |
| **infile** [filename] | Input file name. |
| **outfile** [filename] | Name of the output file. |
| **calfile** [filename] | Name of the file that contains the linear calibration constants. |
| **residfile** [filename] | Name of the file that contains the quadratic residuals to the energy correction. |
| **pulserfile** [filename] | Name of the file that contains the pulser calibration constants. |
| **calmode** [string] | For baterebin, calmode should always be set to 1 |
| **wf** [string] | Selects whether the user wants whole {w} or fractional {f} binning. |
| **row** [integer] | The row number in the input DPH which is to be rebinned in energy. |

## Example (baterebin):

```
baterebin infile.dph outfile.dph calfile.fits caldb caldb row=1
wf=w
```

This would use the calibration database residuals and pulser calibration files to rebin the DPH, moving only whole counts from one bin to another.

# I. Batfftimage

**Batfftimage** constructs a sky image by deconvolving the observed detector plane image with the BAT mask aperture map. This tool is used to estimate the positions and intensities of previously unknown sources on the sky. Typically this tool may be run on a detector image after background cleaning with **batclean**, but this step is not required.

**Batfftimage** also has a secondary usage, which is to compute the partial coding map of the sky. This map represents the fractional exposure of the sky for a given detector/aperture configuration, and is similar to the vignetting profile for imaging telescopes. To compute the partial coding map, users should specify *infile*="NONE" and *pcodemap*="YES". A partial coding threshold can also be specified using *pcodethresh*, which is a fraction between 0.0 and 1.0. Partial coding values below pcodethresh are set to zero.

If an attitude file is specified, then a celestial coordinate system will be attached to the primary World Coordinate System (WCS) descriptors of the image. If no attitude file is specified, then the BAT instrument tangent plane coordinates are assigned instead.

**Batfftimage** will operate on single or multiple images. Multiple input images may be stored as multiple extensions in a single file, or as a FITS table containing one image per row. Individual images may be selected by using the 'rows' parameter. If a table of input images is supplied, then the column specified by 'countscol' is selected. If a column is not specified, then the first 2D column is selected. Output images are always stored as FITS image extensions.

If a background detector map is available, **batfftimage** can automatically subtract it. It automatically recognizes rate vs. count images, and applies an exposure correction to the background if necessary. If multiple images are operated on, then the background file must either have a single image (which is applied to all inputs); or the same number of images as the input file.

## Input Files (batfftimage):

Hidden parameters are listed in parentheses. If these are required, then the code will use default values.

Required:

*infile:* (Optional if *pcodemap*=YES) Input file name containing BAT detector plane image.

*(aperture):* BAT aperture map file name, which contains the coded mask pattern and alignment parameters. If the CALDB database is set up, then CALDB can also be specified.

Optional:

*attitude:* Swift attitude file name. If NONE is given, then a celestial coordinate system cannot be assigned to the image.

(*detmask):* Name of a detector quality map file. This should be an image file with the same dimensions as the focal plane map. A pixel value of 0 indicates the detector is enabled for imaging, and a non-zero value indicates disabled. A default value of NONE implies all detectors are on, except for the BAT detector gap regions.

*(bkgfile):* Name of a file containing background image data to be subtracted from the input. A value of NONE indicates no subtraction should be performed.

*(teldef):* BAT instrument telescope description file, which defines instrument-to-spacecraft alignments. Must be specified in order to assign celestial coordinates to the output image. A value of "NONE" disables celestial coordinates. By default the teldef file located in the HEADAS reference data area is used. If the CALDB database is set up, then CALDB can also be specified.

## Output Files (batfftimage):

Required:

*outfile:* Output sky image or partial coding map file name.

## Parameters (batfftimage):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

The table is found on the next page.

| Parameter / Default / Data Type | Description of batfftimage parameter |
|---|---|
| **infile** [filename] | Input file containing BAT detector plane image. |
| **outfile** [filename] | Output sky image or partial coding map file name. |
| **attitude** = "NONE" [string] | Swift attitude file name. |
| **(aperture) =** "CALDB" [filename] | BAT aperture map file name. |
| **(detmask)** = "NONE" [string] | Name of a detector quality map file. |
| **(bkgfile)** = "NONE" [string] | Name of a file with background image data. |
| **(oversampx)** = 2 [integer] | Oversampling factor of image in X direction. |
| **(oversampy)** = 2 [integer] | Oversampling factor of image in Y direction. |
| **(maskoffx) = 0.0 [real]** | Translational offset to apply to the mask along the BAT_X coord from its nominal design position. |
| **(maskoffy)** = 0.0 [real] | Translational offset to apply to the mask along the BAT_Y coord from its nominal design position. |
| **(maskoffz)** = 0.0 [real] | Translational offset to apply to the mask along the BAT_Z coord from its nominal design position. |
| **(rebalance)** = YES [boolean] | If YES, then the detector and aperture maps will be adjusted so that their additive means are zero. |
| **(pcodemap)** = NO [boolean] | If YES, then output will be a partial coding map. |
| **(pcodethresh)** = 0.01 [real] | Minimum allowable partial coding value in the partial coding map. |
| **(bat_z)** = 0 [real] | For testing with a near-field source. |
| **(origin_z)** = 0 [real] | For ground testing with a near-field source. |
| **(corrections)** = "default" [string] | Comma-separated list of corrections to apply to the image. See below for descriptions. |
| **(handedness)** = "left" [string] | The handedness of the image. |
| **(rows)** = "-" [string] | An index list of images to operate on. |
| **(countscol)** = "INDEF" [string] | If the input image is a FITS table, the column to analyze. |
| **(teldef)** = "CALDB" [string] | BAT instrument telescope description file. |
| **(maskwtswgain)** = 0.03 [real] | Mask weight technique software gain correction factor. |
| **(append)** = NO [boolean] | Append to existing output file? |

**Further description of the *corrections* parameter:**

Comma-separated list of corrections to apply to the image, or "none" if no corrections are to be applied. The possible corrections are:

default: Default corrections, which is shorthand for: "autocollim,flatfield,maskwt,ndets,pcode"

autocollim: Correct plate scale for autocollimation effect

flatfield: Apply corrections for projection effects

maskwt: Apply corrections for FFT technique

ndets: Normalize by number of exposed detectors

pcode: Apply partial coding corrections

## Examples (batfftimage):

### 1. Compute sky images

The BAT analysis tool **batfftimage** constructs a sky image from a detector plane images (DPI) using a Fast Fourier Transform to deconvolve the detector plane data from the coded mask pattern. **batfftimage** is the primary BAT imaging tool and can also produce a map of the coding fraction for each sky pixel. The coding fraction is defined as the fraction of the array which is exposed to the sky pixel through the coded mask. If an attitude file is specified, then a celestial coordinate system will be attached to the primary WCS descriptors of the image. If no attitude file is specified, then the BAT instrument tangent plane coordinates are assigned instead. The default for imaging is for the tangent plane to be at infinity. However, the tool can also be used for imaging at a finite distance from the detector plane, such as during ground calibration tests with radioactive sources, by setting the parameter bat_z to a value other than zero (zero signifies infinity in this context). The default aperture is the one in the refdata area of the BAT tools release. This version is always kept up to date and includes small deviations between the designed and as-built apertures.

The input to this tool is a detector plane image or *.dpi file. The basic form of the command-line use of **batfftimage** is:

```
batfftimage infile.dpi outfile.img attitude.fits
```

Where `infile.dpi` is the input *dpi file.

The next filename on the list is `outfile.img` which is the name of the sky image file we want to create and attitude.fits is the spacecraft attitude file. Additional parameters need to be used since we want the analysis to use a quality map (use *detmask* = `event_rw/output /sw00100139000.mask`) and we need to tell the computer where the teldef file is that contains the instrument alignment inforrmation. Also, remember that our example data was taken on the ground with a shuttered radioactive source. We must therefore specify the z position of the source as well as the offset of the origin in the BAT coordinates using the parameters *bat_z* and *origin_z*. Therefore, what we actually type into the computer is this:

```
my_computer> : batfftimage event_rw/output/sw00100139000_4.dpi
event_rw/output /sw00100139000_4.img
../aux/sw00100139000sat.fits.gz  detmask = event_rw/output/sw0010
0139000.mask     bat_z=229.354992870092 origin_z=100.354994869232
teldef= sw_bat_2003-02-23.teldef
****************************************
         batfftimage v1.6
----------------------------------------
     Input Image: event_rw/output/sw00100139000_4.dpi
 Background File: NONE
   Attitude File: ../aux/sw00100139000sat.fits.gz
     TelDef File: sw_bat_2003-02-23.teldef
  Aperture Image: CALDB
    Output Image: event_rw/output/sw00100139000_4.img
   Detector Mask: event_rw/output/sw00100139000.mask
    Mask Offsets:  +0.000 X    +0.000 Y    +0.000 Z (cm)
    Source BAT_Z: +229.355 (cm)
BAT_Z Ang Origin: +100.355 (cm)
    Oversampling:  2 X    2 Y
Rebalance Images: YES
     Corrections: autocollim=YES handedness=left  pcode=YES
flatfield=YES
----------------------------------------
  Sky image 1 written to event_rw/output/sw00100139000_4.img
  Sky image 2 written to event_rw/output/sw00100139000_4.img
  Sky image 3 written to event_rw/output/sw00100139000_4.img
  Sky image 4 written to event_rw/output/sw00100139000_4.img
    Sky images computed: 4
----------------------------------------
my_computer>:
```

The four sky images that result from running batfftimage image will each include the entire BAT field-of-view.

### 2. Computing partial coding map

To make a map of the coding fraction for each sky pixel, use **batfftimage** as above but with the parameter *pcodemap* = “YES”. A partial coding threshold can also be specified with the *pcodethresh* parameter, which is a fraction between 0.0 and 1.0.

Partial coding values below pcodethresh are set to zero. To make this partial coding map we type:

```
my_computer>:  batfftimage event_rw/output/sw00100139000_4.dpi
event_rw/output/pcodemap.img ../aux/sw00100139000sat.fits.gz
pcodemap=YES
detmask = event_rw/output/sw00100139000.mask
bat_z=229.354992870092 origin_z=100.354994869232 teldef=
sw_bat_2003-02-23.teldef
****************************************
         batfftimage v1.6
----------------------------------------
     Input Image: event_rw/output/sw00100139000_4.dpi
 Background File: NONE
   Attitude File: ../aux/sw00100139000sat.fits.gz
     TelDef File: sw_bat_2003-02-23.teldef
  Aperture Image: CALDB
    Output Image: event_rw/output/pcodemap.img
   Detector Mask: event_rw/output/sw00100139000.mask
    Mask Offsets:  +0.000 X    +0.000 Y    +0.000 Z (cm)
    Source BAT_Z: +229.355 (cm)
BAT_Z Ang Origin: +100.355 (cm)
    Oversampling:  2 X    2 Y
  Partial Coding: YES    Threshold: 0.010000
Rebalance Images: NO
     Corrections: autocollim=YES handedness=left  pcode=NO
flatfield=YES
----------------------------------------
  Sky image 1 written to event_rw/output/pcodemap.img
  Sky image 2 written to event_rw/output/pcodemap.img
  Sky image 3 written to event_rw/output/pcodemap.img
  Sky image 4 written to event_rw/output/pcodemap.img
    Sky images computed: 4
----------------------------------------
```

### 3. Additional Example

```
batfftimage infile.dpi outfile.img attitude.fits
```

This would produce an output image from the input DPI and use the supplied attitude file to derive a celestial coordinate system for the image. Since an infinite source distance is assumed, the output image would have a size of 1736 x 930 pixels, based on a detector plane size of 286 x 173 pixels and a coded mask size of 487 x 243 elements.

# J. Batgse2dpi (internal BAT team tool)

This tool is used to convert calibration files produced by the old (preflight) BAT GSE into the standard DPI format. It is unlikely to ever be used by the general scientific user.

This program reads in an ascii list (infile) of FITS files generated by the "Blue rack" GSE and combines them detector by detector to create a standard BAT Detector Plane Image (DPI)(outfile). There are several options to how the input files are used and these are described below under the discussion of the *histmode* and *windows* input parameters.

The default for this program is to correct the counts in each detector for the dead time in each particular detector module side (sandwich). The correction is applied by adding up the total number of counts in each sandwich and multiplying by the dead time per count to derive the dead time for that sandwich. The default dead time per count is 100 μsec, but this can be changed using the *deadpercount* parameter. The dead time is then subtracted from the exposure time (read from the "EXPOSURE" keyword in the GSE FITS files) to derive the live time (i.e., live time = exposure - dead time). The counts in each detector are multiplied by the ratio exposure/live time. Since this ratio is always greater than or equal to one, the counts written to the output file will always be greater than or equal to the actual counts recorded by the GSE. The dead time correction can be turned off if *deadapp* is set to "NO."

The primary HDU of the output file is the DPI, which is a two-dimensional image with one number per detector and zeros in the gaps between sandwiches. The second ("LIVETIME") HDU is a binary table containing, for each of the 256 BAT detector module sides, the EXPOSURE, LIVE_TIME and DEAD_TIME.

## Input Files (batgse2dpi):

Hidden parameters are listed in parentheses. If these are required, then the code will use default values.

Required:

*infile:* Name of an ascii file containing the names of the sixteen FITS files to be used in generating the dpi file. The names must be qualified with the name of the directory in which they are locate if necessary.

*(windows):* (Optional if *histmode* = “total") Name of FITS file containing the range of the counts values to be used for each of the 128 detectors in SPECTRUM extensions. An existing file in the correct format must be provided if *histmode*="window." The format of the file is a binary table with one row per detector, and columns giving BLOCK, DM, SIDE, and DET, along with WINDOW_LOW and WINDOW_HIGH in ADU channels.

## Output Files (batgse2dpi):

Required:

*outfile:* Output file name. Precede it with an exclamation point, !, (or \! on the Unix command line), to overwrite a preexisting file with the same name (or set the clobber parameter to YES). The contents of the output file are described above in the general description of the tool.

Optional:

*detmask:* Default value is "none". Name of the detector mask file if one is to be generated. This is a rather crude detector mask, which only masks out entire sandwiches or blocks, rather than individual detectors.

## Parameters (batgse2dpi):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

| Batgse2dpi Parameters | |
|---|---|
| **Parameter / Default / Data Type** | **Description** |
| **input** [filename] | File containing the names of the sixteen FITS files to be used in generating the dpi file. |
| **outfile** [filename] | Output file name. |
| **histmode** [string] | "total" for HITSMAP extension processing, "window" for SPECTRUM processing. |
| **(windows)** =" " [filename] | Name of FITS file containing the range of the counts values to be used for each of the 128 detectors in SPECTRUM extensions. |
| **(detmask)** [string] | Name of the detector mask file |
| **(deadpercount)** [real] | Value used to compute dead time. Default value is 100 microseconds. |
| **(deadapp)** [boolean] | "YES" apply dead time correction. |

## Examples (batgse2dpi):

1. Generate a dpi file using the HITSMAP extensions. The example below uses the default values for deadpercount and deadapp.

```
batgse2dpi 'file_list' 'hitsmap.out' histmode=total detmask=mask_t
clobber="Yes" chatter=2
```

2. Generate a dpi file using the SPECTRUM extensions. The example below uses the default values for deadpercount and deadapp.

```
batgse2dpi 'file_list'  'spectrum.out' windows=windows.fits
histmode=window detmask=mask_w clobber="Yes" chatter=2
```

# K. Bathotpix

**Bathotpix** locates hot pixels in a BAT detector plane image. Hot pixels can be a source of noise in subsequent image processing steps, and so should usually be excised.

This tool uses a histogram-based approach to locate the hottest and coldest pixels in the image. Given a distribution of counts, the "centermost" portion is selected as being good (the "keepfract" amount). If keepfract is 0.98, then 98% of the detectors are selected as good. The selection window is further enlarged by bands on either side using the guardfract and guardval parameters. Values outside the selection window are considered to be "cold" (too low) or "hot" (too high).

The output of the tool is a quality map of the same dimensions as the detector image. Values stored in the map are determined by the keywords below, but typically a value of 0 indicates a good pixel. This mask can be input into downstream image processing stages. The second extension, BADPIX, is a binary table containing a list of bad pixels.

Users can submit an existing detector mask using the detmask parameter. Pixels with bad quality are ignored by bathotpix in computing the distribution and selection windows. By default, the input mask is logically AND'd with the result, so that the output is the cumulative record of excised pixels. By default, BAT detector gaps are also excised.

## Input Files

Required:

*infile:* Input file name containing a detector plane image.

Optional:

*(detmask):* Input detector mask image. Pixels with bad quality are ignored in the **bathotpix** analysis.

## Output Files

Required:

*outfile:* Output detector mask image.

## Parameters

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

| Bathotpix Parameters | |
|---|---|
| **Parameter / Default / Data Type** | **Description** |
| **infile** [filename] | Input file name containing a detector plane image. |
| **outfile** [filename] | Output detector mask image. |
| **(detmask)** = "NONE" [string] | Input detector mask image. |
| **(keepfract)** = 0.98 [real] | Fraction of non-zero detectors to select as good, initially. |
| **(guardfract)** = 0.25 [real] | Fractional amount to enlarge initially selected detectors, relative to the median value. |
| **(guardval)** = 5.0 [real] | Further amount to enlarge the selection window, this time in counts. |
| **(applygaps)** = yes [boolean] | If yes, then positions of BAT detector gaps are excised. |
| **(mergemask)** = yes [boolean] | If yes, then the input mask is merged with the result before writing to the output. |
| **(zerothresh)** = 20.0 [real] | Threshold value. |
| **(goodval)** = 0 [integer] | Value to be used in detector mask maps to indicate a good pixel. |
| **(badval)** = 1 [integer] | Value to be used in detector mask maps to indicate a bad pixel, such as a detector gap. |
| **(hotval)** = 1 [integer] | Value to be used in detector mask maps to indicate a hot pixel. |
| **(coldval)** = 1 [integer] | Value to be used in detector mask maps to indicate a cold pixel. |

## Examples (bathotpix):

### 1. Generate a "quality map"

Since the BAT is a coded aperture telescope, the pattern of the relative count rates for each pixel of the detector array is the input to the FFT routine that produces a sky image. For a clear image, the relative brightness of each pixel (1 pixel = 1 CZT detector) must depend only on the exposure of the array through the coded aperture mask and not on detector-specific features. We must therefore make sure that the imaging programs know which detectors were either disabled (turned off) or were “hot” (had an abnormally high count rate) at the time the data were taken. This detector quality map is a simple binary table that indicates which detectors must be excluded from the imaging calculations and is used as input for most imaging tools

such as **batmaskwtevt**, **batmaskwtimg** etc. The method for finding the enable/disable map of interest is to locate the map whose timestamp is closest to the time of the trigger. This map is a binary FITS table stored in the sw[observationID]/data/bat/trend subdirectory for that observation. Enable/disable detector maps have the name code "bdecb." Thus this enable/disable detector map is found in a (compressed) FITS file with a name like sw[obsID][segID]/data/bat/trend/ sw[obsID][segID]bdecb.fits.gz. For our example burst, the oservation ID is "00100139" and the segment ID is "000"; thus, we must locate the FITS file named: `sw00100139000.011/data/bat/trend/sw00100139000bdecb.fits`.

Now that we have located the appropriate enable/disable map, we can use it to generate a detector quality map that is appropriate for the trigger time of our burst**.** The tool **bathotpix** locates hot pixels (detectors with an abnormally high count rate) in a BAT detector plane image. Hot pixels can be a source of noise in subsequent image processing steps, and so should usually be eliminated from the image calculation. **bathotpix** is well described in the help file for the tool, but basically it looks for outliers in a histogram of the counts in the array. The user can adjust the parameters specifying what fraction of pixels are considered good and what thresholds are applied. Reasonable defaults are set for these hidden parameters (see the help page bathotpix.html). When working with real (as opposed to simulated) data, this tool should always be run before any imaging is carried out. Nearly all tools which take DPIs or DPHs as inputs will accept a detector mask file.

The output of **bathotpix** can be combined with the enable/disable map generated in flight to produce a unified detector mask file.

In the following example we run **bathotpix** with a new dpi file (see **batbinevt** recipe 1.) and with the enable/disable file we just found to create a unified quality map:

```
bathotpix infile.dpi quality_map.mask detmask = apriori.mask
```

substituting in the right filenames we have:

```
my_computer>bathotpix event_rw/output/sw00100139000.dpi
event_rw/output/sw00100139000.mask detmask=
trend/sw00100139000bdecb.fits.gz
****************************************
         bathotpix v1.2
----------------------------------------
     Input Image: event_rw/output/sw00100139000.dpi
----------------------------------------
   Image maximum value: 66
      Number of zeroes: 45
        Selected range: 2.000000 - 17.000000 (without guard)
        Selected range: 0.000000 - 26.250000 (with guard)
    Approximate median: 8.000000
```

```
   Report of HOT/COLD pixels (note row & col start at 0)
   -------------------------
    Col.    Row  H/C    Pix Value    DETID  Bl DM  S Det
     127     49   H          62.0    23181  11  2  1  13
     205    104   H          66.0    12229   5  7  1  69
      23    161   H          27.0     1111   0  4  0  87
     Note: Cold pixels do not show up at chatter=2
  Mask image written to event_rw/output/sw00100139000.mask
-----------------------------------------
my_computer>
```

This unified quality mask “sw00100139000.mask” is used for scientific analysis.

**2. Additional Example:**

```
bathotpix image.dpi image.mask detmask=apriori.mask
```

This would run the hot pixel code to find hot pixels in the image.dpi and convolve this list with the apriori.mask files to produce a unified detector mask which could be input to other tools.

# L. Batid2xy

This is a basic tool which allows the user to transform back and forth between the two systems for identifying BAT detectors. These are the Block / DM / Side / Detector system (or DETID) and the geographical system (DETXY). One can either input a file containing a list of detector identifiers to transform or can calculate the transformations on the command line.

This code can either read in an input FITS file containing a list of detector identifications or (x,y) positions, or it can read these values from the command line (with the input file given as NONE).

The program calls a routine called batidconvert to figure out either the X (column) and Y (row) numbers associated with the particular detector or the reverse transformation. The output is always the full identification of the detector by DETID Block DM SIDE DET X Y.

There are three basic modes of conversion, which are abbreviated BDSD (for block, detector-module, side, detector), DETID and DETXY. The mode is selected either by including a "BATIDMOD" keyword in the input file, or by supplying valid ( > -1) values of the *detid, detx* or *dety* parameters.

BDSD: In this mode, the conversion goes from detector identification to BAT X and Y (location in the detector plane). The detector is identified by, respectively, block (0:15), detector module (0:7), side (0:1), detector (0:127). In the input file, these are supplied in columns named "BLOCK","DM","SIDE" and "DET." The keyword "BATIDMOD" is set to "BDSD." On the command line, these are supplied through the *block, dm, side* and *det* parameters. See example below.

DETID: In this mode, the conversion goes from detector identification to BAT X and Y (location in the detector plane). The detector is identified by detector ID, which is derived from the formula: detid = block*2048 + dm*256 + side*128 + detector. In the input file, this is supplied in a column name "DETID." The keyword "BATIDMOD" is set to "DETID." On the command line, this is supplied by setting the *detid* parameter equal to the detector ID. See example below.

DETXY: In this mode, the conversion goes from BAT X and Y (location in the detector plane) to detector identification. The detector location is given by BATX and BATY. In the input file, these are supplied in columns named "DETX" and "DETY." The keyword "BATIDMOD" is set to "DETXY." On the command line, these are supplied through the *detx* and *dety* parameters. See example below.

## Input Files

Required:

*none.*

Optional:

*infile:* Input file name. The input file must have an extension called "DETID" and columns commensurate with the "BATIDMOD" keyword (see general description above). If this keyword is missing, the code assumes that it is set to the value "BDSD." See a description of how to create an input file under Example 1 below. If the input file is given as NONE, then the detector values to be converted are supplied on the command line.

## Output Files

Required:

*none.*

Optional:

*outfile:* Output file name. It is not currently implemented to write the output to a FITS file.

## Parameters (batid2xy):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

| Parameter / Default / Data Type | Description of batid2xy parameter |
|---|---|
| **infile** [filename] | Input file name. |
| **(block)** [integer, 0 - 15] | Specify the block number. |
| **(dm)** [integer, 0 - 7] | Specify the detector module number. |
| **(side)** [integer, 0 - 1] | Specify the detector module side. |
| **(det)** [integer, 0 - 127] | Specify the detector number. |
| **(detid)** [integer, -1 to 32767] | Specify the detector id (block*2048 + dm*256 + side*128 + detector). |
| **(detx)** [integer, -1 to 285] | Specify the detector X location in detector units, starting from 0. |
| **(dety)** [integer, -1 to 1273] | Specify the detector Y location in detector units, starting from 0. |
| **(outfile)** [filename] | Output file name. |

The *block, dm, side* and *det* parameters are ignored if a valid input file is specified or if any of the *detid, detx*, or *dety* parameters are greater than -1.

For *detx* and *dety*, If the value provided corresponds to a gap in the detector array, then the code returns a TNULL value for each detector identification.

## Examples:

### 1. Input file.

```
batid2xy infile='detids.fits'
```

The input file must contain a DETID extension and a BATIDMOD keyword telling the tool what format the input table is in (DETID or DETXY). There are instructions in the help file for batid2xy for creating properly formatted files from ascii tables.

### 2. Command line

```
batid2xy NONE detx=259 dety=102
```

This would output the conversion from the given DETX and DETY to DETID and the breakdown into Block, DM, Side, Detector.

```
DETID   Block   DM   SIDE    DET       X         Y
15299        7       3          1        67      259     102
```

See the online (fhelp) file for **batid2xy** for further examples.

# M. Batmasktaglc

This tool is used to process a raw mask-tagged light curve to produce a scientifically useful mask-tagged light curve. It is a specialized tool which does only one task. Since it will be applied in the pipeline, it is unlikely to be needed by the general user.

The tool basically backs out the light curve processing that was carried out in the flight code to derive a background subtracted light curve and appropriate statistical error bars. A complete description of the algorithms used is found in comment fields in the source code.

See the help file and comments in the code for a more complete description.

## Input Files (batmasktaglc):

Hidden parameters are listed in parentheses. If these are required, then the code will use default values.

Required:

*infile:* Name of a masktag FITS file containing a MASK_TAG_RATES extension. This file is the raw mask tagged light curve as generated by the flight code.

*quadfile:* Name of the input quadrant rate file, which must contain a RATES extension. This file contains the uncorrected rates for each BAT quadrant on the same time and energy scales as the raw mask tagged rates. This file must be in the format produced by bat2fits from the flight derived raw quadrant light curves. Further, this must be matched to the raw light curve by matching the target ID of the mask tagged source in the file names. In other words the raw light curve sw00020002000bmt_00024651hp.lc must be matched to sw00020002000bmw_00024651.fits. The mask weight file contains the weighting factor for each detector as well as the sums and sums of squares of the weights which are used in the flight processing.

*maskwt:* Name of mask weight map file. The default value of this parameter is "file," which means that the tool looks for a MASK_TAG_WEIGHT HDU in the infile. This file is a detector plane histogram containing the flight generated weight for each detector.

*(ebounds):* The location of the EBOUNDS file describing the energy ranges in the light curves. The default is to use an EBOUNDS extension that is part of the infile.

Optional:

*(detmask):* Optional input detector plane mask. This is used to exclude detectors that are disabled from contributing to the light curves and errors.

This file should always be a flight code generated detector mask which represents only detectors actually turned off, rather than a mask derived from ground processing.

## Output Files (batmasktaglc):

Required:

*outfile:* Output light curve. The output file is a standard light curve file, containing TIME, RATE, ERROR and BACKGROUND columns, each entry of which is a four-dimensional vector, with one number for each of the energy ranges described in the EBOUNDS extension.

## Parameters (batmasktaglc):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

| Batmasktaglc Parameters | |
|---|---|
| **Parameter / Default / Data Type** | **Description** |
| **infile** [filename] | Name of mask tagged rate file. |
| **quadfile** [filename] | Name of the input quadrant rate file. |
| **maskwt** [string] | Name of mask weight map file. |
| **outfile** [filename] | Output light curve. |
| **(detmask)** = "none" [string] | Optional input detector plane mask. |
| **(ebounds)** = "file" [string] | The location of the EBOUNDS file describing the energy ranges in the light curves. |
| **(corrections)** = "none" [string] | Comma-separated list of corrections to apply to the mask weighting:<br>"none" if no corrections are to be applied.<br>"flatfield" basic flat fielding correction accounting for both cosine-type and r-squared-type effects<br>"cosine" cosine effects of off-axis illumination only<br>"pcode" partial coding effect only. |

## Example (batmasktaglc)::

```
batmasktaglc  "raw mask weight file”  “quadrant rates file" "mask
wt.map file"
"outfile" ebounds="ebounds file"
```

In this case there is no EBOUNDS extension on any of the input files so an external EBOUNDS must be specified. The output is a corrected mask-tagged light curve.

# N. Batmaskwtevt / Batmaskwtimg

These are sister tools and so are described together. These tools use a forward projection (ray tracing) algorithm to determine the weighting factor for each detector for a given source position. See the discussion of Mask Tagging in the Introductory section of this document.

The primary difference in the user interface is that while **batmaskwtevt** requires an input event file to be specified, **batmaskwtimg** takes no input file, but the user must specify an output file. In other words, **batmaskwtevt** will calculate a mask weighting for every event in the input file and write this to the MASK_WEIGHT column in the input file (creating the column if it doesn't exist). Alternatively, **batmaskwtimg** calculates the weighting for a hypothetical detector plane and output this weighting map.

Currently, this tool is geared towards applying mask weighting for sources in *instrument* coordinates. Eventually, when the spacecraft attitude file is incorporated, it will be possible to specify the coordinates of a source in RA/DEC, and they will be automatically converted to instrument coordinates.

## Input Files

Hidden parameters are listed in parentheses. If these are required, then the code will use default values.

Required:

*infile:* Input event file name. This file must contain BAT events, and have at least the columns TIME, MASK_WEIGHT, DETX and DETY. The MASK_WEIGHT column is overwritten by this tool, so the file must be writeable.

*(aperture):* BAT aperture map file name, which contains the coded mask pattern and alignment parameters. If the CALDB database is set up, then CALDB can also be specified.

*(detmask):* Name of a detector quality map file. This should be an image file with the same dimensions as the detector plane map. A pixel value of 0 indicates the detector is enabled, and a non-zero value indicates disabled. A default value of NONE implies all detectors are on, except for the BAT detector gap regions. This map is only used for computing the fraction of illuminated pixels. The output mask weight map contains a value for all detectors in the array, regardless of detmask.

*(teldef):* BAT instrument telescope description file, which defines instrument-to-spacecraft alignments. Must be specified when celestial coordinates are provided. If the CALDB database is set up, then CALDB can also be specified.

Optional:

*attitude:* File name of Swift attitude history, or NONE if none is used.

*(incatalog):* (**batmaskwtimg** ONLY) Input source catalog containing source positions. The catalog should contain one row per source. A value of NONE indicates that the source coordinates should be taken from the command line.

## Output Files

Required:

*outfile:* (**batmaskwtimg** ONLY) Output file name.

Optional:

*(auxfile):* (**batmaskwtevt** ONLY) Specifies an auxiliary output file which describes the position of the source in the BAT field of view as a function of time, including various ray tracing diagnostics. The columns are TIME (MET in seconds); BAT_X/Y/ZOBJ (source position in instrument coordinates in cm); IMX/Y (source tangent plane coordinates); PCODEFR (partial coding fraction); NGOODPIX (number of enabled detectors); MSKWTSQF (normalization parameter). One row is created for each ray tracing operation performed. This file is required for response matrix generation during slews.

## Parameters

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

The table is found on the next page. Note that parameters that apply only to **batmaskwtevt** are shaded in yellow and those that apply only to **batmaskwtimg** are shaded in pink.

| Parameter / Default / Data Type | Description |
|---|---|
| **infile** [filename] (batmaskwtevt ONLY) | Input event file name. |
| **outfile** [filename] (batmaskwtimgONLY) | Output image file name. |
| **attitude** [string] | File name of Swift attitude history. |
| **ra** [real] | Right ascension of source in dec degs, or other position (see coord_type below) |
| **dec** [real] | Declination of source in decimal degs. |
| **(aperture)** = "CALDB" [filename] | BAT aperture map file name. |
| **(coord_type)** = "sky" [string] | Method of specifying the source coordinates (see below for details) |
| **(time)** = 0 [real] (batmaskwtimgONLY) | Time associated with the mask weighted image. |
| **(rebalance)** = YES [boolean] | Adjust so that additive mean is zero. |
| **(corrections)** = "default" [string] | Comma-separated list of corrections to apply to the mask weighting (see below). |
| **(detmask)** = "NONE" [string] | Name of a detector quality map file. |
| **(auxfile)** ="NONE" [string] (batmaskwtevt) | Specifies an auxiliary output file. |
| **(bat_z)** = 0 [real] | Position of the source in BAT_Z coords. |
| **(incatalog)** = "NONE" (batmaskwtimg) | Input source catalog containing source positions. |
| **(racol)** = "RA_OBJ" (batmaskwtimg) | Col name of first coord value in input cat. |
| **(deccol)** = "DEC_OBJ" (batmaskwtimg) | Col name of 2nd coord value in input cat. |
| **(namecol)** = "NAME" (batmaskwtimg) | Col name of source name in input cat. |
| **(maskwtcol)** = "MASK_WEIGHT" [string] | Name of mask weight column. |
| **(distance)** = 1.0e7 [real] | Default bat_z position if bat_z is zero. |
| **(maskoffx / y / z )** = 0.0 [real] | Translational offset to apply to mask along the BAT_X/Y/Z coordinate. |
| **(maskthickness)** = INDEF [string] | Thickness of the lead mask tiles in cm. |
| **(buffersize)** = 32768 [int] (batmaskwtevt) | Size of internal event buffer for process. |
| **(subpixsize)** = 0.02 [real] | Size of detector subpixels used in computing mask weighting in cm. |
| **(teldef)** = "CALDB" [string] | BAT instrument telescope description file |
| **(gapval)**=0.0 [real] (batmaskwtimg ONLY) | Value to place in image cells where detector gaps are located. |
| **(origin_z)** = 0 [real] | For ground testing with a near-field srce. |
| **(maskwtswgain)** = 0.04 [real] | Mask weight technique software gain |

**Further description of the *coord_type* parameter:**

Both tools call for an attitude file and a source position. The default coordinates for the source position are RA and declination in degrees. This default requires a valid attitude file. Alternatively the user can specify the source location in another system (such as image tangent plane coordinates), in which case *attitude*=NONE is a valid parameter specification. The *coord_type* parameter (default "sky") is used to specify the system for the input source position according to the table below.

| COORD_TYPE | RA parameter | DEC parameter | Units |
|---|---|---|---|
| sky | Right Ascension | Declination | deg |
| cartesian | BAT_X position | BAT_Y position | cm |
| unit | BAT_X unit vector | BAT_Y unit vector | none |
| fswlonlat | Phi (flt. software) | Theta | deg |
| grmclonlat | Longitude (GRMC) | Latitude | deg |
| tanxy | Tan(theta_x) | Tan(theta_y) | none |

The coordinate types "fswlonlat" and "grmclonlat" are coordinate systems used in the flight software and instrument simulators. The coordinate type "tanxy" gives the tangent-plane angles of the source. Where necessary the source distance and/or BAT_Z position must also be provided (in centimeters).

**Further description of the *corrections* parameter:**

Comma-separated list of corrections to apply to the image, or "none" if no corrections are to be applied. The possible corrections are:

none: No corrections are to be applied.

default: Default corrections, which is shorthand for: "flatfield,ndets,pcode,maskwt"

autocollim: Correct plate scale for autocollimation effect

flatfield: Apply corrections for projection effects

maskwt: Apply corrections for mask weighting technique

ndets: Normalize by number of exposed detectors

pcode: Apply partial coding correction

subpixelate: Use slower but potentially higher fidelity algorithm

forward: Reserved for clean procedure

unbalanced: Reserved for clean procedure

nearfield: Near field corrections for ground calibration analysis only

The "pcode" and "ndets" corrections require the user to supply the detmask keyword.

## Examples:

### 1. Apply mask weighting

The tools used to generate the mask weighting information are **batmaskwtevt** and **batmaskwtimg.** Both of these tools use a forward projection (ray tracing) algorithm to determine the weighting factor for each detector for a given source position. The primary difference between the two tools is in the user interface: **batmaskwtevt** requires an input event file to be specified and calculates the mask weight for each individual event. **batmaskwtimg** creates the mask weight (shadow function) for the entire detector array at one time and outputs the information in DPI format. In other words, **batmaskwtevt** will calculate a mask weighting for every event in the input file and will write these weighting values to the MASK_WEIGHT column in the input file (creating the column if it doesn't exist). Alternatively, **batmaskwtimg** calculates the weighting for a hypothetical detector plane and outputs this weighting as a map.

The basic form of the command we'll need is:

```
batmaskwtevt input_file attitude_file ra dec detmask=qmap_file
```

Where `input_file = event_rw/sw00100139000bevshpsuf.evt` and the attitude files are found in the data/aux subdirectory with the suffix "sat.fits" The appropriate file for this example is thus `attitude_file=data/aux/sw00100139000sat.fits.gz.` The RA and dec position of the source can be extracted from keywords in the event file's primary header. Looking at the header information in the event file we've been using, `event_rw/sw00100139000bevshpsuf.evt`, we extract the following position:

RA_OBJ = 147.4987 / [deg] RA Object
DEC_OBJ = 10.24879 / [deg] Dec Object

Finally, we will also use the quality map we made in the recipe for **bathotpix.** We thus type:

```
my_computer> batmaskwtevt event_rw/sw00100139000bevshpsuf.evt
../aux/sw00100139000sat.fits.gz ra=147.4987 dec=10.24879 detmask =
event_rw/output/sw00100139000.mask bat_z=229.354992870092
origin_z=100.354994869232 teldef = sw_bat_2003-02-23.teldef
```

```
****************************************
         batmaskwtevt v1.3
----------------------------------------
    Input Events: event_rw/sw00100139000bevshpsuf.evt
   Attitude File: ../aux/sw00100139000sat.fits.gz
     TelDef File: sw_bat_2003-02-23.teldef
  Aperture Image: CALDB
   Detector Mask: event_rw/output/sw00100139000.mask
     Corrections: flatfield pcode ndets maskwt
     Coordinates: sky
       Longitude: 147.498700 (deg) Latitude: 10.248790 (deg; sky)
        Z Origin: +100.355 cm
----------------------------------------
  Number of Events Processed: 281584
----------------------------------------
my_computer>
```

and a column named “MASK_WEIGHT” is added to the event file.

**2. Mask weighting of an event file with an input attitude file.**

```
batmaskwtevt  sw00000001000bnone028.unf  attitude.dat  257.5500  -
28.118
```

This will calculate the mask weighting for a source at RA=257.5500, declination = -28.118 and fill the MASK_WEIGHT column in the input file.

**3. Mask weighting of an event file without an input attitude file**. The source position is specified in BAT image theta and phi.

```
batmaskwtevt  sw00000001000bnone028.unf  attitude.dat  -45.4293
16.1387  coord_type=fswlonlat
```

**4. Create a mask weight image** for a source at RA=257.5500, declination = -28.118

```
batmaskwtimg maskwt.img attitude.dat 257.5500 -28.118
```

# O. Batsumdph

This is a simple tool which sums together two or more DPHs in a single file to produce an output DPH. This functionality is also present in **batbinevt**, but this tool remains as a simpler way to accomplish this task. The only option to the tool is the row number range in the DPH file.

## Input Files

Required:

*infile:* Input file name. The file must contain a BAT_DPH extension.

## Output Files

Required:

*outfile:* Output file name. Precede it with an exclamation point, !, (or \! on the Unix command line), to overwrite a preexisting file with the same name (or set the clobber parameter to YES).

## Parameters

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

| Parameter / Default / Data Type | Description |
|---|---|
| **input** [filename] | Input file name. |
| **outfile** [filename] | Output file name. |
| **rows** | The ranges of rows, from the binary table of the BAT_DPH extension of the original file, to be summed. |

## Example:

```
batsumdph infile.dph sumfile.dph rows="1-3,5-7"
```

This would sum together rows 1-3 and 5-7 into a single output DPH.

# P. Battblocks

This is a primary tool for higher level analysis of burst data. It is primarily designed to operate on an event file, but can also be used to analyze a light curve. The output is a GTI file.

**Battblocks** estimates interesting time intervals (GTIs) based on time variable event or rate data. It does this using the Bayesian Block algorithm, which is intended to robustly compute time intervals based on Bayesian analysis. (see Scargle 1998, ApJ, 504, 405) There is a more detailed description in the tool help file. The user provides event data or a single-channel binned light curve.

The **battblocks** tool passes BAT rate data and light curves through a Bayesian Block algorithm to search for and characterize the bursts. Bayesian Blocks is a technique that detects and characterizes localized structures (bursts) in time series data. The algorithm determines the most probable segmentation of the input data into time intervals during which the photon arrival rate is statistically consistent with constant. Thus the lightcurve is reduced to a series of time intervals where the count rate is essentially steady. These time “blocks” will be shorter in duration where the rate change is fast such as during a burst and longer during times of slowly changing background rates.

The light curve should be a standard OGIP light curve FITS file (i.e. contain the proper HDUCLASn keywords which identify whether the light curve is background subtracted or not). Both RATE and COUNT type light curves are supported. It must be possible to multiply RATE by the TIMEDEL per-bin exposure to arrive at counts per bin. If the light curve has been background subtracted, then the ERROR column must be present.

**Battblocks** relies on the keyword information in the input file as being correct. In particular, HDUCLAS1 should be "EVENTS" or "LIGHTCURVE", depending on the input file type. For light curves, HDUCLAS2 should be "NET" or "TOTAL" depending on whether the file is background subtracted. For light curves, HDUCLAS3 should be "RATE" or "COUNT", although the user can override this with the *hduclas3* parameter. There must also be a way to determine the per-sample exposure, either with the TIMEDEL keyword, or a TIMEDEL/EXPOSURE column.

The Bayesian block algorithm can be used to estimate time intervals for any time series. For bursts, it can also estimate several measures of the burst duration, and the interval of the peak flux. The calculated duration of a burst depends, of course, on a judgment about when the burst starts and when it stops. We need a consistent method for making this determination, however, so that valid comparisons of the durations of different bursts can be made. The BAT tool **battblocks** is used to characterize the time structure of a burst and thus provides a consistent means for measuring the duration of every burst in the catalog. The burst duration is estimated by locating bracketing background intervals, and then locating intervals which enclose 90% and 50% of the total counts. These are stored in the *durfile* with extension names GTI_T90 and GTI_T50. Users should be aware that these algorithms are sensitive to systematic variations in the background. Some care must be taken to

filter the incoming light curve. Users an also choose their own percentage fluence with the *txx* keyword; the result is stored in the GTI_TXX extension.

The calculated duration of a burst also depends on the lightcurve used. The user should look at the final, complete lightcurve which contains times between 24 seconds before to 185 seconds after the burst. Since some bursts are long enough that the gamma-ray emission continues during the time of the slew and even after settling on target, the initial light curves sent down in the TDRSS messages may not contain the complete burst. In such cases, the user will need use the final lightcurve derived from data sent to the ground during Malindi passes.

The interval of the peak flux is estimated by sliding a time window across the light curve or event data and finding the epoch with the highest counts. The user can select the size of the window. The corresponding GTI is written to *durfile* with the extension name 'GTI_PEAK'.

For these measures, the user can request that **battblocks** perform background subtraction. The background is estimated based on the first and last Bayesian block intervals, and linearly interpolated to other points.

The global analysis time grows quadratically: as the input file size doubles, the computation time quadruples, and so on. Users can perform a local Bayesian block analysis by setting *tlookback* to a non-zero time window size. Only data within the time window at a particular instant are considered for the analysis at that instant. When *tlookback* is used, the resulting blocks are analyzed in a second pass to further consolidate blocks larger than the window size. Users of event data can also adjust the *nspill* parameter to decimate the events, at the expense of decreasing the time resolution proportionately.

## Input Files (battblocks):

Required:

*infile:* Input file name containing event data or light curve data. Events must be sorted in increasing time order. Light curve data must be OGIP-format (containing the HDUCLASn keywords), and either have the columns TIME and COUNTS (binned counts); or TIME, RATE and ERROR (binned data with error bars). Only single channel light curves are supported.

## Output Files (battblocks):

Required:

*outfile:* Name of output good time interval file. The primary output is a file which contains the interesting time intervals calculated for the event file in the form of a GTI extension. This GTI extension can then be used as input to **batbinevt** to produce either a light curve or PHA-II file based on these time intervals.

Optional:

(*durfile):* Name of output duration measures. If durfile is 'NONE' then the duration measures are not computed or written. If an optional output durfile is specified, then the tool will also output GTIs corresponding to the durations T90 and T50 (plus an optional user specified duration measure) derived from the data. GTI tables with extensions 'GTI_T90', 'GTI_T50' and 'GTI_PEAK' are created. Note that the method is sensitive to background fluctuations so the user should be careful to filter the input file.

*(diagfile):* Diagnostic output file, used for internal debugging purposes. NONE indicates no diagnostic output should be made.

## Parameters (battblocks):

Hidden parameters are in parentheses and need not be supplied on the command line. There are three additional parameters common to all FTOOLs which are described at the beginning of this section. The brief descriptions of the parameters given here are for reference only, for a full discussion of what the parameters do and what values they can take refer to the online help (fhelp) for this tool.

The table is found on the next page.

| Battblocks Parameters | |
|---|---|
| **Parameter / Default / Data Type** | **Description** |
| **infile** [filename] | Input file name containing event data or light curve data. |
| **outfile** [filename] | Name of output good time interval file. |
| **(durfile)** = 'NONE' [filename] | Name of output duration measures. |
| **(nspill)** = 128 [integer] | Number of events to skip per analysis cell. |
| **(gaussian)** = INDEF [string] | Boolean which determines whether the Bayesian blocks computation is performed using Gaussian statistics or not. |
| **(txx)** = 0.0 [real] | User-specified percentage of burst to estimate the duration for. |
| **(tpeak)** = 1.0 [real] | Size of sliding window, in seconds, used to determine the interval of the peak count rate. |
| **(tlookback)** = 0.0 [real] | A non-zero value indicates the lookback time window in seconds, for the local bayesian block analysis. |
| **(timedel)** = 100e-6 [real] | Internal time quantization size. |
| **(ncp_prior)** = 6.0 [real] | Log parameter prior for the number of "change points" between Bayesian blocks. |
| **(bkgsub)** = NO [boolean] | Automatic background subtraction, for computation of duration measures and fluence estimates. |
| **(timecol)** = "TIME" [string] | Name of the light curve time column. |
| **(countscol)** = "INDEF" [string] | Name of the light curve rate or counts column. |
| **(errcol)** = "ERROR" [string] | Name of the light curve gaussian error column. |
| **(expocol)** = "INDEF" [string] | Name of the light curve per-bin exposure column. |
| **(hduclas3)** = "INDEF" [string] | Default value of the HDUCLAS3 keyword. |
| **(diagfile)** = NONE [string] | Diagnostic output file, used for internal debugging purposes. |

## Examples (battblocks):

### 1. Estimate burst duration using a BAT TDRSS light curve.

BAT sends a light curve down through TDRSS several times. The final version includes all counts received by BAT from 24 seconds before the burst to 185 seconds after. There are 4 energy channels. The time binning varies and is densest (0.128 seconds) closest to the trigger time and least dense (4.096 seconds) well after the trigger.

The data for these TDRSS light curves can be found in files that end with “msb.lc(.gz)”. For example, TDRSS light curves for GRB 100139 can be found in the file named sw00100139000msb.lc.gz

The BAT TDRSS lightcurve is found in the TDRSS_LC extension of the FITS file. The contents are:

TIME-the Mission Elapsed Time in Seconds from the Swift Reference Time.

RELTIME-time since burst, ranging from -24 to 185 seconds.

TIMEDEL-a column describing the time between successive measurements, which is variable for the BAT light curve, ranging from 0.128 to 4.096 seconds.

RAW_COUNTS-An array of four vectors, each with same number of time steps and timedels as the TIME column, one vector for each of the 4 BAT energy bands reporting the raw counts received during that time interval.

RATE-An array of four vectors, each with same number of time steps and timedels as the TIME column, one vector for each of the 4 BAT energy bands reporting the count rate for that time interval. Raw counts are normalized by timedel to get the rate in counts/s.

ERROR-The error on the rate per time step.

TOT_COUNTS-For each time step, the sum of all four BAT raw_counts.

TOT_RATE-For each time step, the sum of all four BAT rates.

TOT_RATE_ERR-the error on the total count rate per time step.

In the example below, we show a call to battblocks using the tot_rate column of the TDRSS light curve file for burst 100139:

```
my_computer>  battblocks sw00100139000msb.lc.gz bb.gti
```

```
durfile='dur.gti'  bkgsub=yes countscol=TOT_RATE clobber=yes
 ****************************************
         battblocks v1.2
 ----------------------------------------
     Input Data: sw00100139000msb.lc.gz
     Output GTI: bb.gti
  Events to Skip: 128     Changepoint log(prob) Prior: 6.000000
   Internal Tick: 1.000000e-04 s
   Lookback time: 0 s
  Bkg. Subtract?: yes (for fluence/T50/T90 calculations)
 ----------------------------------------
  Estimated T90 duration: 2.688 s +/- 0 s
  Estimated T50 duration: 1.536 s +/- 0 s
  Estimated Peak Interval: MET   108579195.02399981 +/- 0.500000 s
  Estimated background rate 1:      497.794 near MET
108579183.680000 s
  Estimated background rate 2:      505.934 near MET
108579290.112000 s
  Estimated total duration: 2.944 s (for data selection)
       (from MET 108579194.560000 to MET 108579197.504000)
  Estimated total fluence: 45575.859515 count
  Created GTI with 5 entries
 ----------------------------------------
```

Outputs: bb.gti and dur.gti

As shown above, the **battblocks** tool reports estimates for T90, T50, peak Interval, total duration and total fluence for the burst. The output files generated by **battblocks** are also important. In the above example, these file names are bb.gti (bb= “Bayesian block”) and dur.gti (dur =“duration”).

To find out the contents of these files, type

```
my_computer>fstruct dur.gti
No. Type     EXTNAME    BITPIX Dimensions(columns)  PCOUNT  GCOUNT

     0    PRIMARY              16     0    0     1
     1    BINTABLE GTI_T90 8   16(2) 1    0     1

      Column Name       Format     Dims    Units     TLMIN  TLMAX
      1 START             D                   s
      2 STOP              D                   s

   2  BINTABLE GTI_T50      8  16(2) 1    0     1

      Column Name       Format     Dims    Units     TLMIN  TLMAX
      1 START             D                   s
      2 STOP              D                   s

   3  BINTABLE GTI_PEAK     8  16(2) 1    0     1
```

```
      Column Name        Format    Dims    Units     TLMIN  TLMAX
      1 START             D                  s
      2 STOP              D                  s

   4  BINTABLE GTI_TOT  8     16(2) 1                0       1

      Column Name        Format    Dims    Units     TLMIN  TLMAX
      1 START             D                  s
      2 STOP              D                  s
```

This tells you that the first extension contains the GTI_T90 information as a start and stop time. The second extension contains the GTI_T50 information again as a start and stop time. Finally, the third extension reports the block for which the burst experiences its peak total rate (GTI_PEAK) and provides the stop and start time of that Bayesian block.

Using the "fdump" command to see the values of the columns in the file:

```
my_computer>  fdump prhead=no dur.gti
 Name of optional output file[STDOUT]
 Names of columns[ ]
 Lists of rows[ - ]


         START                  STOP
         s                      s
      1.085791945599998E+08      1.085791972479999E+08


         START                  STOP
         s                      s
      1   1.085791951999998E+08  1.085791967359999E+08

         START                  STOP
         s                      s
      1   1.085791943999998E+08  1.085791956479998E+08
```

The start and stop times for the Bayesian blocks calculated can be found in bb.gti

```
my_computer> fstruct bb.gti
No. Type  EXTNAME   BITPIX Dimensions(columns) PCOUNT  GCOUNT

   0  PRIMARY         16     0                   0      1
   1  BINTABLE STDGTI  8     16(2) 5             0      1

      Column Name     Format     Dims       Units     TLMIN  TLMAX
      1 START            D                  s
      2 STOP             D                  s
```

```
my_computer> fdump prhead=no bb.gti
Name of optional output file[STDOUT]
Names of columns[ ]
Lists of rows[ - ]


        START                                  STOP
        s                                        s
      1   1.085791728000000E+08   1.085791945599998E+08
      2   1.085791945599998E+08   1.085791948799998E+08
      3   1.085791948799998E+08   1.085791973759999E+08
      4   1.085791973759999E+08   1.085791975039999E+08
      5   1.085791975039999E+08   1.085793827200000E+08
```

Examining the values in these files, or overplotting them on the TDRSS light curve can guide the user in deciding whether to update the parameter values used by the tool to get a more accurate estimate of the time scales of interest.

Caveats and Pointers:

Users should be aware that these algorithms are sensitive to systematic variations in the background. Some care must be taken to filter the incoming light curve. For example, consider TDRSS light curves (raw BAT rates). If there is a noise spike due to a noisy detector, or if during the slew a bright source enters the field of view, the raw rate will change, and this can effect the results of battblocks. The background is estimated based on the first and last Bayesian block intervals, and linearly interpolated to other points.

For event-based light curves, these sources of background can be subtracted. However, an N-sigma statistical fluctuation could remain, possibly significant enough to be considered as "source". If the real burst is 0.01 seconds long, but the fluctuation happens 100 seconds later, then T90 might be set to 100 seconds.

Users can also produce time bin edges defined by the percentage of the total fluence contained within each time bin. To do this, run **battblocks** with the 'txx' keyword; the result is stored in the GTI_TXX extension. For example, if you want the time bounds defining 67% of the fluence, call battblocks with txx=67.0. Note the value of this keyword runs from 0-100 and not 0.0-1.0.

The interval of the peak flux is estimated by sliding a time window across the light curve or event data and finding the epoch with the highest counts. The user can select the size of the window. The corresponding GTI is written to 'durfile' with the extension name 'GTI_PEAK'

The global analysis time grows quadratically: as the input file size doubles, the computation time quadruples, and so on. Users can perform a local Bayesian block analysis by setting tlookback to a non-zero time window size. Only data within the time window at a particular instant are considered for the analysis at that instant. When tlookback is used, the resulting blocks are analyzed in a second pass to further consolidate blocks larger than the window size. Users of event data can also adjust the 'nspill' parameter to decimate the events, at the expense of decreasing the

time resolution proportionately.

Summary: by running **battblocks**, looking at the results for the validity of the default parameters over which the calculation is done, and then rerunning, if necessary for better estimates, results in the user having the necessary information to characterize the GRB based on the TDRSS BAT lc data.

**2. Partition data into Bayesian Blocks**

```
battblocks burst.evt burst_bb.gti
```

**3. Extraction of Bayesian blocks from a light curve**; extraction of burst durations, including a user specified level of 68.3% of the burst (T90 and T50 intervals are always computed)

```
battblocks burst.lc burst_bb.gti durfile=burst_dur.gti txx=68.3
```

# IV. SAMPLE ANALYSIS CHAIN

This analysis chain starts with a time sorted and filtered event file and derives a sky image and source list along with mask tagged light curves and spectra. This chain does not use batdrmgen and battblocks, but these should be added.

(1. Apply the energy calibration to the event file)

```
bateconvert sw00074651000bgrb.unf  sw00074651000bcbo0001g0001.dph
calmode=LINEAR clobber=yes

bateconvert v2.00 completed
Input events file is sw00074651000bgrb.unf
Calmode: LINEAR
5861791 events converted from PHA to PI and written to file
sw00074651000bgrb.unf
```

(2. Make a DPI from the Event file)

```
batbinevt sw00074651000bgrb.unf sw00074651000bgrb.dpi DPI 0 u -
tstart=101807671.616000 tstop=101807672.640000 clobber=yes
*****************************************
         batbinevt v1.3
-----------------------------------------
    Input Events: sw00074651000bgrb.unf
     Output File: sw00074651000bgrb.dpi  (DPI)
   Detector Mask: NONE
      Time Range: 101807671.616000 to 101807672.640000
(requested)
      Time Range: 101807671.616000 to 101807672.640000  (actual)
Apply Weighting?: NO      Energy Column: PI
     Energy Bins: -
    Output Units: COUNTS
  Binning Method: UNIFORM
   Time Bin Size: 0.000000 (s)
-----------------------------------------
  DPIs written to sw00074651000bgrb.dpi
  EBOUNDS written to sw00074651000bgrb.dpi
    Number of Rows Processed: 5861791
    Number Accepted/Rejected: 17130/5844661
  Time Bins:          1      Energy Bins:    1
-----------------------------------------
```

(3. Make a sky image from the DPI)

```
batfftimage sw00074651000bgrb.dpi sw00074651000bgrb.img NONE
bat_z=0.000000 clobber=yes
*****************************************
         batfftimage v1.3
-----------------------------------------
     Input Image: sw00074651000bgrb.dpi
```

```
 Background File: NONE
   Attitude File: NONE
     TelDef File:
/home/heasfs/krimm/Swift/batgsw/krimm/headas/i686-pc-linux-
gnu/refdata/sw_bat_2003-02-23.teldef
  Aperture Image:
/home/heasfs/krimm/Swift/batgsw/krimm/headas/i686-pc-linux-
gnu/refdata/bat_aperture-2003-08-28.img
    Output Image: sw00074651000bgrb.img
   Detector Mask: NONE
    Mask Offsets:  +0.000 X    +0.000 Y    +0.000 Z (cm)
    Source BAT_Z: Infinity
    Oversampling:  2 X    2 Y
Rebalance Images: YES
     Corrections: autocollim=YES
-----------------------------------------
  Sky image 1 written to sw00074651000bgrb.img
    Sky images computed: 1
-----------------------------------------
```

(4. Find sources in the sky image)

```
batcelldetect sw00074651000bgrb.img sw00074651000bgrb.src 8.0
clobber=yes
*****************************************
         batcelldetect v1.4
-----------------------------------------
     Input Image: sw00074651000bgrb.img
  Output Catalog: sw00074651000bgrb.src
   Input Catalog: NONE
Part. Coding Map: NONE
    Back. Window: CIRCLE    Radius: 30
   Source Window: CIRCLE    Radius: 6
   SNR Threshold: 8.000000
 Number of Iter.: 2
Min. Num. Pixels: 20
-----------------------------------------
    Found 1 Images
    Analyzing Image: 1
    Detection Iteration 1
        Found 10 cumulative pixels
    Detection Iteration 2
        Found 10 cumulative pixels

    Detected 1 sources

   IMXcent  IMYcent    IMXwid   IMYwid    Peak Cts Bkg Var  SNR
   0.20192  0.20701    0.00291  0.00291   1536.7    92.8    16.4
-----------------------------------------
Theta is now 16.128608 (degrees)
Phi is now -45.712618 (degrees)
```

(5. Carry out the mask weighting in the event file)

```
batmaskwtevt sw00074651000bgrb.unf NONE ra=-45.712618
dec=16.128608 coord_type=fswlonlat distance=10000000.000000
clobber=yes
****************************************
         batmaskwtevt v1.1
----------------------------------------
    Input Events: sw00074651000bgrb.unf
  Aperture Image:
/home/heasfs/krimm/Swift/batgsw/krimm/headas/i686-pc-linux-
gnu/refdata/bat_aperture-2003-08-28.img
   Detector Mask: NONE
     Corrections:
   Subpixel Size: 0.020000
   Effective edge depth: 0.200000
     Coordinates: fswlonlat
       Longitude:   -45.712618 (deg) Latitude:    16.128608 (deg;
fswlonlat)
                          BAT_X               BAT_Y
BAT_Z
 Cartesian      +1939720.238841     +1988581.126014
9606405.701421  (cm)
 Unit Vect.           +0.193972           +0.198858
0.960641  (cm)
 Tangent              +0.201919           +0.207006
----------------------------------------
  Number of Events Processed: 5861791
----------------------------------------
```

(6. Create a mask-tagged light curve with 64 msec time binning)

```
batbinevt sw00074651000bgrb.unf sw00074651000bgrb.lc LC 0.064 u -
tstart=101807371.616000 tstop=101807971.616000 clobber=yes
****************************************
         batbinevt v1.3
----------------------------------------
    Input Events: sw00074651000bgrb.unf
     Output File: sw00074651000bgrb.lc  (LC)
   Detector Mask: NONE
      Time Range: 101807371.616000 to 101807971.616000
(requested)
      Time Range: 101807552.000000 to 101807971.616000  (actual)
Apply Weighting?: YES     Energy Column: PI
     Energy Bins: -
    Output Units: RATE
  Binning Method: UNIFORM
   Time Bin Size: 0.064000 (s)
----------------------------------------
  Light curve written to sw00074651000bgrb.lc
  EBOUNDS written to sw00074651000bgrb.lc
```

```
    Number of Rows Processed: 5861791
    Number Accepted/Rejected: 5689233/172558
  Time Bins:       6557      Energy Bins:    1
-----------------------------------------
```

(7. Create a  four channel mask-tagged light curve with 64 msec time binning)

```
batbinevt sw00074651000bgrb.unf sw00074651000bgrb.4lc LC 0.064 u
/local/data/gcn3a/caldb/data/swift/bat/bat_ebounds4-2003-08-
19.fits tstart=101807371.616000 tstop=101807971.616000 clobber=yes
*****************************************
         batbinevt v1.3
-----------------------------------------
    Input Events: sw00074651000bgrb.unf
     Output File: sw00074651000bgrb.4lc  (LC)
   Detector Mask: NONE
      Time Range: 101807371.616000 to 101807971.616000
(requested)
      Time Range: 101807552.000000 to 101807971.616000  (actual)
Apply Weighting?: YES     Energy Column: PI
     Energy Bins:
/local/data/gcn3a/caldb/data/swift/bat/bat_ebounds4-2003-08-
19.fits
    Output Units: RATE
  Binning Method: UNIFORM
   Time Bin Size: 0.064000 (s)
-----------------------------------------
  Light curve written to sw00074651000bgrb.4lc
  EBOUNDS written to sw00074651000bgrb.4lc
    Number of Rows Processed: 5861791
    Number Accepted/Rejected: 5102226/759565
  Time Bins:       6557      Energy Bins:    4
-----------------------------------------
```

(8. Create an eighty-channel PHA file from the event file)

```
batbinevt sw00074651000bgrb.unf sw00074651000bgrb.pha PHA 0 u
/local/data/gcn3a/caldb/data/swift/bat/bat_ebounds80-2003-08-
19.fits tstart=101807671.616000 tstop=101807672.640000 clobber=yes
*****************************************
         batbinevt v1.3
-----------------------------------------
    Input Events: sw00074651000bgrb.unf
     Output File: sw00074651000bgrb.pha  (PHA)
   Detector Mask: NONE
      Time Range: 101807671.616000 to 101807672.640000
(requested)
      Time Range: 101807671.616000 to 101807672.640000  (actual)
Apply Weighting?: YES     Energy Column: PI
     Energy Bins:
/local/data/gcn3a/caldb/data/swift/bat/bat_ebounds80-2003-08-
```

```
19.fits
    Output Units: RATE
  Binning Method: UNIFORM
   Time Bin Size: 0.000000 (s)
-----------------------------------------
  Spectrum written to sw00074651000bgrb.pha
  EBOUNDS written to sw00074651000bgrb.pha
    Number of Rows Processed: 5861791
    Number Accepted/Rejected: 16726/5845065
  Time Bins:          1      Energy Bins:   80
-----------------------------------------
```