The SWIFT BAT Software Guide

Version 2.1
30 Jan 2007
Matched to HEASoft 6.1.2

C. B. Markwardt,
S. D. Barthelmy, J. C. Cummings,
D. Hullinger, H. A. Krimm, A. Parsons,
(NASA/GSFC — BAT Instrument Team)
Contents

1 Introduction ............................................................. 1
  1.1 Scope ............................................................... 1
  1.2 The Basic Scheme ............................................... 1
  1.3 Organization of this Guide ..................................... 1
  1.4 New releases and Updates ...................................... 2

2 BAT Instrument ......................................................... 3
  2.1 Swift Overview ...................................................... 3
  2.2 Instrument Overview ............................................. 3
  2.3 The Swift Spacecraft ............................................ 3
  2.4 The BAT Instrument ............................................... 4
    2.4.1 Technical Description ..................................... 4
    2.4.2 Instrument Operations .................................... 8

3 BAT Operating Modes and Data Types .............................. 11
  3.1 Introduction ........................................................ 11
  3.2 GRB Products and Response .................................... 11
    3.2.1 Important Notes About Trigger Times ..................... 13
    3.2.2 Description of Standard Products ....................... 13
  3.3 Non-GRB Products ............................................... 14
    3.3.1 Survey Data ................................................... 14
    3.3.2 Rate Data ...................................................... 15
    3.3.3 Maps ........................................................... 17
    3.3.4 Trend Products .............................................. 17

4 Introduction to BAT Analysis ....................................... 19
  4.1 Introduction ....................................................... 19
  4.2 Coded Apertures: Basic Concepts ............................. 19
  4.3 Coded Aperture Analysis for X-ray Astronomers ............. 21
  4.4 How Sky Fluxes are Reconstructed ........................... 23
4.5 BAT Field of View and Partial Coding ........................................... 23
4.6 BAT Coordinates ................................................................. 24
4.7 Definition of BAT Count Units and Corrections ......................... 25
  4.7.1 Definition of BAT Mask-Weighted Counts .......................... 25
  4.7.2 Converting to a counts per area ................................... 26

5 BAT Analysis Procedures .......................................................... 27
  5.1 Introduction .......................................................................... 27
  5.2 Complete GRB Processing Script ......................................... 28
    5.2.1 Introduction ................................................................... 28
    5.2.2 Prerequisites ................................................................ 28
    5.2.3 Change to Data Directory ............................................. 28
    5.2.4 Creating GRB Products Automatically ....................... 28
    5.2.5 Results .......................................................................... 29
    5.2.6 When Things Go Wrong ................................................. 29
    5.2.7 Making Your Own Custom Products ............................. 29
    5.2.8 Spectral Fitting .............................................................. 30
  5.3 Energy Calibration of Events ................................................... 30
    5.3.1 Reasons to Re-run the Energy Calibration ....................... 30
    5.3.2 Prerequisites ................................................................ 30
    5.3.3 Initial Steps ................................................................... 31
    5.3.4 Compute Energy Calibration for BAT Events ................ 31
    5.3.5 What to do if no gain/offset map is present ................... 31
  5.4 Making a Detector Quality Map ............................................... 32
    5.4.1 Reasons to Create a BAT Quality Map ........................... 32
    5.4.2 Prerequisites ................................................................ 33
    5.4.3 Initial Steps ................................................................... 33
    5.4.4 Create a Detector Plane Image ...................................... 33
    5.4.5 Retrieve Known Problematic Detectors ....................... 34
    5.4.6 Find Noisy Detectors ................................................... 34
    5.4.7 Where to Use the Quality Map ..................................... 34
    5.4.8 The Difference Between the Enable/Disable Map and the Quality Map ......................................................... 34
    5.4.9 What to do if no detector enable/disable map is present .. 35
  5.5 Mask Weighting (Ray Tracing) ................................................. 35
    5.5.1 Introduction ................................................................... 35
    5.5.2 Reasons to Re-run the Mask Weighting ......................... 36
    5.5.3 Prerequisites ................................................................ 37
    5.5.4 Initial Steps ................................................................... 37
CONTENTS

6.2.5 Analysis: Earth and Moon Occultation ........................................... 54
6.2.6 batmaskwtevt: Slewing Raytracing Keywords Potentially Incorrect .......... 54

6.3 Old Issues ................................................................. 55

6.3.1 SDC: BAT data from SDC pipeline has incorrect DATE-OBS/END keywords ..................................................... 55
6.3.2 batoccultmap: incorrect exposure correction ..................................................... 56
6.3.3 batbinevt: light curves have slightly incorrect energy bins ......................... 56
6.3.4 Mask tagged light curve systematic flux errors ............................................. 57
6.3.5 XSPEC crashes with segmentation fault with BAT spectra ......................... 57
6.3.6 baterebin: Possible extra rows in EBOUNDS extension of output DPH ....... 58
6.3.7 batbinevt: Incorrect exposure for survey analysis ........................................ 58
6.3.8 CALDB: Incorrect Spectral Response < 20 keV ........................................ 59
6.3.9 CALDB: Incorrect Alignment Matrix ......................................................... 59
6.3.10 batbinevt: Incorrect EXPOSURE Keyword .............................................. 59
6.3.11 batcelldetect: Incorrect Error Bars ......................................................... 60
6.3.12 batbinevt: Incorrect Handling of float point DPHs .................................... 60
6.3.13 batmasktaglc: light curve error bars too small ........................................ 61
6.3.14 batmaskwtevt/batmaskwtimg: High-Energy Background Contamination ...... 61

A BAT Data Formats ............................................................... 63

A.1 Introduction ................................................................. 63
A.2 File Naming Overview ..................................................... 63
A.3 Event Files ................................................................. 63
A.4 Survey Detector Plane Histograms ............................................... 64
A.5 Rate Files ............................................................... 65

A.5.1 One-second Rates ......................................................... 65
A.5.2 Quadrant Rates ............................................................ 66
A.5.3 64-Millisecond Rates .................................................... 66
A.5.4 Maximum Rates .......................................................... 66
A.5.5 Mask-tagged Light Curves .................................................. 67
A.5.6 TDRSS GRB Light Curves .................................................. 68

A.6 Maps ............................................................... 68

A.6.1 Detector Enable/Disable Map ................................................... 68
A.6.2 Detector Quality Map .......................................................... 69
A.6.3 Gain / Offset Map ............................................................. 69
A.6.4 Mask Weight Map ............................................................. 70

A.7 TDRSS Messages ........................................................... 70

A.7.1 BAT GRB Alert ............................................................. 71
A.7.2 GRB Position Message (ACK or NACK) .............................................. 71
CONTENTS

A.7.3 FoM and S/C Will/Will Not Observe .............................................. 71
A.7.4 BAT Light Curves ............................................................................ 72

A.8 Housekeeping Files ........................................................................... 73
A.8.1 Americium Survey DPHs ................................................................. 73
A.8.2 Americium Block Spectra ............................................................... 73
A.8.3 Calibration Pulser Maps ................................................................. 73
A.8.4 Detector Array Panel (DAP) Housekeeping ..................................... 73
A.8.5 BAT Engineering Housekeeping ...................................................... 76
A.8.6 DI Commandables ......................................................................... 78
A.8.7 Debug Stream ................................................................................. 78
A.8.8 Trigger Diagnostic Tables ............................................................... 78

B BAT FITS Keywords ................................................................. 79
B.1 Introduction ....................................................................................... 79
B.1.1 Keywords Used ............................................................................ 79
B.1.2 Data Types ................................................................................... 81

C BAT Acronym and Terminology Glossary ........................................ 83

D BAT Software Tool Reference ......................................................... 87
D.1 BATBINEVT ..................................................................................... 88
D.1.1 NAME ......................................................................................... 88
D.1.2 USAGE ....................................................................................... 88
D.1.3 DESCRIPTION ........................................................................... 88
D.1.4 PARAMETERS ............................................................................ 90
D.1.5 EXAMPLES ............................................................................... 94
D.1.6 SEE ALSO .................................................................................. 98
D.1.7 LAST MODIFIED ...................................................................... 98
D.2 BATCELLDETECT ........................................................................... 99
D.2.1 NAME ......................................................................................... 99
D.2.2 USAGE ....................................................................................... 99
D.2.3 DESCRIPTION .......................................................................... 99
D.2.4 CATALOG OUTPUT ................................................................ 100
D.2.5 DISTORTION CORRECTION ..................................................... 103
D.2.6 PARAMETERS .......................................................................... 103
D.2.7 EXAMPLES .............................................................................. 108
D.2.8 SEE ALSO .................................................................................. 108
D.2.9 LAST MODIFIED ...................................................................... 108
D.15.6 PARAMETERS ......................................................... 160
D.15.7 EXAMPLES .......................................................... 164
D.15.8 SEE ALSO ............................................................. 164
D.15.9 LAST MODIFIED ...................................................... 164

D.16 BATOCCLGTI .............................................................. 165
D.16.1 NAME ................................................................. 165
D.16.2 USAGE ................................................................. 165
D.16.3 DESCRIPTION ........................................................ 165
D.16.4 PARAMETERS ......................................................... 165
D.16.5 EXAMPLES ............................................................ 168
D.16.6 SEE ALSO ............................................................. 168
D.16.7 LAST MODIFIED ...................................................... 168

D.17 BATOCCLMGT ............................................................ 169
D.17.1 NAME ................................................................. 169
D.17.2 USAGE ................................................................. 169
D.17.3 DESCRIPTION ........................................................ 169
D.17.4 PARAMETERS ......................................................... 169
D.17.5 EXAMPLES ............................................................ 171
D.17.6 SEE ALSO ............................................................. 171
D.17.7 LAST MODIFIED ...................................................... 171

D.18 BATPHASYSERR ......................................................... 172
D.18.1 NAME ................................................................. 172
D.18.2 USAGE ................................................................. 172
D.18.3 DESCRIPTION ........................................................ 172
D.18.4 PARAMETERS ......................................................... 172
D.18.5 EXAMPLES ............................................................ 172
D.18.6 SEE ALSO ............................................................. 173
D.18.7 LAST MODIFIED ...................................................... 173

D.19 BATTBLOCKS ............................................................ 174
D.19.1 NAME ................................................................. 174
D.19.2 USAGE ................................................................. 174
D.19.3 DESCRIPTION ........................................................ 174
D.19.4 PARAMETERS ......................................................... 174
D.19.5 EXAMPLES ............................................................ 177
D.19.6 REFERENCES ......................................................... 177
D.19.7 LAST MODIFIED ...................................................... 177

D.20 BATUPDATEPHAKW ...................................................... 178
Chapter 1

Introduction

1.1 Scope

This Guide describes the principles of the processing and reduction of Swift data taken with the Burst Alert Telescope (BAT). This includes instructions on retrieving and generating the proper auxiliary data, applying calibrations to the data, selecting sources of interest, and creating final science products such as images, light curves, spectra and spectral response matrices.

This guide assumes that the data have already been downloaded from the archive and that the Swift software and calibration data provided in CALDB are installed and initialized. The Swift BAT software package is a part of HEASOFT, also known as FTOOLS. Updates are distributed regularly from the FTOOLS web site [http://ftools.gsfc.nasa.gov/].

1.2 The Basic Scheme

Swift data are converted into FITS files at the Swift Data Center (SDC), which also runs a generic pipeline for gamma-ray burst (GRB) observations. The high level science products resulting from the pipeline are available in the Swift quick-look area. At present these products are not archived in the formal archive service, back-up copies for most bursts are available at the quick-look site.

1.3 Organization of this Guide

- Chapter 2 describes the BAT hardware. This information is provided for reference and is not necessary to know in detail for science analysis.

- Chapter 3 describes the BAT operating modes and data products. The BAT contains a sophisticated software system, which produces a variety of useful products.

- Chapter 4 introduces the basic concepts of coded aperture analysis, as applied to the BAT software.

- Chapter 5 provides detailed recipes of how to analyze BAT GRB data.

- Chapter 6 lists analysis issues to be aware of (both current and old).

- Appendix A contains a description of BAT data products and formats.
• Appendix B describes the FITS keywords used and produced by BAT software.
• Appendix C contains a glossary of acronyms and specialized Swift/BAT terminology.
• Appendix D contains all of the BAT software task help files for reference.

1.4 New releases and Updates

This version of the guide is written based on the Swift software released with HEASOFT 6.0.5. Swift software and calibration data are updated periodically. The latest information on new software and calibration releases are posted at:

- [http://swift.gsfc.nasa.gov/](http://swift.gsfc.nasa.gov/)

Request of additional information and bug reports can be entered in the Feedback form located at that URL.
Chapter 2

BAT Instrument

2.1 Swift Overview

2.2 Instrument Overview

The Burst Alert Telescope (BAT) is a highly sensitive, large field of view coded-aperture telescope designed to monitor a large fraction of the sky for the occurrences of gamma-ray bursts (GRBs). The BAT provides the burst trigger and the 1-3 arcmin accurate position that is then used to slew the spacecraft to point the two narrow-FOV instruments (the X-ray Telescope – XRT, and the Ultraviolet/Optical Telescope – UVOT) for follow-up observations. The BAT positions and lightcurves are transmitted through TDRSS to the ground in ~20 and 130 sec respectively, and distributed to the world through GCN. While observing bursts, BAT simultaneously and automatically accumulates an all-sky hard x-ray survey. The BAT consists of a 5200 cm$^2$ array of $4 \times 4$ mm$^2$ CdZnTe elements located 1 meter behind a $2.7$ m$^2$ coded-aperture mask of $5 \times 5$ mm$^2$ elements, with a point spread function (PSF) of 22 arcmin. The BAT coded-aperture mask, and hence its FOV, was limited by the Delta rocket faring. The BAT instrument was designed and built at Goddard Space Flight Center.

2.3 The Swift Spacecraft

The spacecraft bus was built by Spectrum Astro (Gilbert, AZ). It is a 3-axis stabilized platform. The slew rate has been enhanced beyond typical platforms because of the rapid slewing requirements for the narrow-FOV instrument (NFI) follow-up on the BAT burst positions. The platform can slew from 0 to 50° in 20-70 sec. Also, because of the rapid follow-up requirement, the Attitude Control System (ACS) has full onboard autonomy for conducting a slew maneuver. No ground mission operations decisions or commanding are needed for the spacecraft to calculate and initiate a slew maneuver. The on-board autonomy checks all spacecraft orientation constraints (sun angle, moon angle, earth limb angle, and ram vector). If the calculated slew to the new BAT burst position is determined not to violate any of the constraints, then a slew is performed. This will happen ~90% of the time. Once a new target has been acquired, the pointing stability is better than 0.1 arcsec, which is more precise than is needed for BAT and is driven by the requirements for the two NFIs.
2.4 The BAT Instrument

The BAT instrument is shown in Figure 2.1. The Burst Alert Telescope (BAT) makes the initial detection of the gamma-ray burst (GRB), calculates a position for that burst, makes an on-board decision if the burst is worth an NFI follow-up observation, and sends that position to the spacecraft attitude control system, if it is worthy. It does all this within 10-30 sec of the initial trigger of the burst. To do this for a large number of bursts (∼100 yr⁻¹), BAT has a large FOV (1.4 sr half-coded & 2.2 sr partially-coded). The only way to image such a large FOV is to use the coded-aperture technique. The following sections describe the details of the design, the function of the BAT instrument, and the data products that will be available to the world community.

2.4.1 Technical Description

The basic numbers describing the BAT instrument are listed in Table 2.1. The BAT instrument consists of a detector plane of 32,768 CZT detector elements and front-end electronics, a coded aperture mask located 1 m above the detector plane, a graded-Z fringe shield to reduce the instrumental background event rate and cosmic diffuse background, and a thermal radiator and control system to keep the detector plane at a constant temperature. The control of the BAT instrument is done through the Image Processor and it also does the on-board event processing (burst trigger detection, burst location calculations, and burst figure-of-merit calculation). While searching for bursts, BAT also accumulates a hard x-ray survey of the entire sky over the course of the mission.

The energy range of 15-150 keV in Table 2.1 describes the energy range over which the effective area is more than 50% of the peak value. The range is governed at the lower end by the electronic discriminator threshold, and at the upper end by increasing transparency of the lead tiles in the

Figure 2.1: Idealized view of the Swift optical bench, including a cut-away view of the Burst Alert Telescope (BAT). The main BAT structures are the coded aperture mask (top, shown as a randomly filled grid, and the detector array (bottom). The narrow field instruments are mounted to the side of the BAT.
TABLE 2.1: Swift BAT Instrument Parameters.

The BAT detector plane is composed of 32,768 pieces of 4.00 mm square CdZnTe (CZT), which are 2.00 mm thick. For electronic control, event data handling, and fabrications reasons, these 32K pieces are grouped into two layers of hierarchical structure. The hierarchy breaks down into the following definitions. A sub-array of 8×16 pieces of CZT are attached to a single analog signal processing ASIC (the XA1), which is known as a Detector Module (DM) Side, or Sandwich. Two of these sub-arrays with the ASIC control electronics constitutes a DM. Eight DMs are mounted in a mechanical structure called a Block. There are 16 Blocks mounted in the Detector Array Plane (DAP) in a 2-by-8 configuration. The mechanics of mounting the pieces of CZT yield a pixel pitch of 4.20 mm with gaps between the DMs and Blocks. The image reconstruction process easily handles these gaps, because the gaps have been restricted to be an integer multiple (2 or 3 pitch units) of the basic pixel pitch. This division of the 32K pixels into 128 DMs and 16 Blocks, each with its own electronics, power supplies, control parameters, and communications links, provides a natural parallelism and redundancy against failures in the BAT instrument. This, plus the forgiving nature of the coded aperture technique, means that BAT can tolerate losses of individual pixels, individual DMs, and even whole Blocks without losing the ability to detect bursts and determine locations. There is, of course, a loss in burst-detection and the survey sensitivities.

The Detector Module (DM) The CZT pixel elements have planar electrodes and are biased typically to -200 volts. This bias voltage is independently commandable for each DM (i.e. each DM of 256 pixels has a commandable HV supply of negative 0-300 volts). The anode of each CZT piece is AC-coupled to an input on the XA1 ASIC. It has 128 channels of charge sensitive pre-amps, shaping amps, and discriminators. It is a self-triggering device. When supplied with a threshold control voltage, the ASIC will recognize an event on one of its 128 input channels, block out the remaining 127 channels, and present the pulse height of the event on its output for digitization. An on-board ADC unit digitizes the pulse height to a quantization of 0.5 keV. The XA1 has a linear
CHAPTER 2. BAT INSTRUMENT

Figure 2.2: The BAT has the 32,768 pieces of CZT divided into the hierarchical structure shown here. The BAT detectors are 4 mm \( \times \) 4 mm \( \times \) 2 mm pieces of CdZnTe (upper left). A Detector Module (DM) is composed of two DM Sides, each with 128 detectors (upper right). Eight DMs are arranged into a Block (lower left in a ground test fixture). Sixteen blocks are integrated to form the full detector array (bottom part of lower right in the flight structure).

range up to \( \sim \)200 keV and extends up to \( \sim \)500 keV with decreasing linearity. The pulse height and detector number are transmitted to the Block Controller and Data Handler (BCDH; see next section). This process takes 100 \( \mu \)sec. Individual channels can be logically disabled to handle noisy detectors. There is an electronic calibration pulser circuit within each DM. When commanded, it sequentially injects the specified number of charge pulses of the specified level into each of the 128 channels. These “calibration” events are flagged to be recognizable to the event processing routines in the Image Processor. This calibration pulser allows the offset, gain, and linearity of each of the 32K channels within the instrument to be monitored. In addition to the electronic calibration, there are two Am241 “tagged sources” which provide 60 keV photons for calibrating the absolute energy scale and detector efficiency of each CZT detector inflight. Figure 2.2 (upper right) shows an actual flight DM.

The Block and the Block Controller and Data Handler The Block is the next level up in the hierarchy. The mechanical structure sits on the Detector Array Panel (DAP) and holds 8 DMs, and various interface cards, voltage regulators, and command and data handlers. Figure 2.2 (lower left) shows a flight block. The BCDH is basically a data concentrator, so that the IP need not communicate directly with the 128 DMs in the Detector Array. It receives photon and calibration event data from all 8 DMs and multiplexes this data into a single serial data stream that is transmitted to the IP. In the process, the BCDH embeds a DM identification in the event data so that the IP can determine which pixel produced the event from within the entire Detector Array. It also embeds timestamps (100 usec quantization) so that the event data can be parsed
2.4. THE BAT INSTRUMENT

Figure 2.3: Photograph of the BAT flight coded aperture mask before being integrated with the rest of the instrument. In the final flight configuration, the lead tiles are on the bottom of the mask panel, facing the detectors.

and analyzed with respect to time.

The Coded Aperture Mask

It is not technologically possible to produce an image in the gamma-ray bandpass using traditional focusing optics; especially over a large field of view. Hence, the only way to formulate an image is to use the coded-aperture method. The BAT coded aperture is composed of \( \sim 52,000 \) lead tiles located 1 meter above the CZT detector plane. The lead tiles are 5.00 mm square and 1.0 mm thick. The tiles are mounted on a low-mass, 5-cm thick composite honeycomb panel. The pattern is completely random with a 50% open / 50% closed filling factor. The Mask is 2.4 m by 1.2 m (with the corners cut off, it is 2.7 m\(^2\)), which yields a 100° by 60° FOV (half-coded). Figure 2.3 shows a picture of the BAT coded aperture mask.

The Fringe Shield

To reduce the event rate in the detector plane, a graded-Z Fringe Shield is located on the side walls between the Mask and the Detector Plane and under the Detector Plane. It reduces the isotropic cosmic diffuse flux and the anisotropic Earth albedo flux by \( \sim 95\% \). The graded-Z shield is composed of 4 layers of materials: Pb, Ta, Sn, and Cu; for a total mass of roughly 24 kg.
The Thermal Control System

The CZT and XA1 front-end ASIC require the operating temperature to be controlled. The nominal operating temperature for the CZT is 20°C with a commandable range of 0-25°C. The temporal and spatial thermal gradients are held to ±0.5°C during normal operations. There are four parts to the DAP thermal control system: 1) the heaters with adjustable set points on each DM, 2) heat pipes embedded in the honeycomb plate under the Blocks to pull the heat to the sides, 3) loop heat pipes from the two side edges of the plate to pull the heat to the radiator, and 4) the thermal radiator mounted on the front side of the instrument. The radiator is 1.4 m² and dissipates ~200 W of thermal power to space. Even with the radiator mounted in the anti-sun side of the spacecraft, the thermal environment the radiator sees varies over an orbit due to different Earth viewings. To accommodate this, the loop heat pipes are variable conductance and there are heaters attached to each DM to actively control the temperature.

The Image Processor (IP)

The Image Processor (IP) is the BAT instrument control processor. It does seven major functions: 1) collects and scans the event stream looking for rate increases (i.e. bursts), 2) calculates the sky images when there is a rate increase and scans for new point sources, 3) determines if the newly detected burst is merits a slew request, 4) accumulates the hard x-ray survey detector plane histograms, 5) controls the instrument and gathers housekeeping information, 6) formulates the BAT-portion of the telemetry stream sent to the spacecraft control computer, and 7) receive commands from the spacecraft and processes them. These seven functions are implemented in an architecture shown in Figure 5. There are two processors within the IP. The RAD6000 is the main instrument processor. It handles the event processing, trigger searching, housekeeping, and command, control, and telemetry functions. The 21020 DSP does image processing. A 256 MB DRAM board provides all the memory for a 10-minute event-by-event ringbuffer, survey mode histograms, plus other science and engineering data products. The 1553 board provides the interface between the BAT IP and the Swift spacecraft control computer.

2.4.2 Instrument Operations

The BAT instrument has two basic modes of operation: 1) scan-survey mode, and 2) burst mode. These two modes reflect the two major types of data that BAT produces: hard x-ray survey data and burst positions. Most of BAT’s time is spent waiting for a burst to occur in its FOV. It accumulates events in the detector plane looking for increases in the count rate over a range of time scales. This scanning for rate increases is the trigger algorithm. When the trigger algorithm is satisfied, it goes into burst mode. When not in burst mode and while scanning for a trigger, the instrument is accumulating spectra in each of its 32,786 detector elements every ~5 minutes. These 32K spectra are recorded and become part of the survey data. During the scan-survey mode each block periodically goes into calibration mode. Within the BAT flight software is the Figure Of Merit calculation that decides if the current burst trigger is worth performing a spacecraft slew maneuver.

Burst Detection

Finding GRBs within BAT is composed of two processes: 1) the detection of the onset of a burst by looking for increases in the event rate across the detector plan, and 2) the formation of an image of the sky using the events detected during the time interval at the beginning of the burst.
2.4. THE BAT INSTRUMENT

**Burst Trigger Algorithm** The burst trigger algorithm looks for excesses in the detector count rate above those expected from background and constant sources. The two main obstacles to GRB detection are the variation in background and the heterogeneity of GRB time profiles. In Low Earth Orbit, detector background rates can vary by more than a factor of two during a 90-minute orbit. The durations of GRBs range from milliseconds to minutes, during which they may have anywhere from one to several dozen peaks in the emission. Therefore, the triggering system must be able to extrapolate the background and compare it to the measured detector count rate over a variety of timescales and in several energy bands.

The trigger algorithms used in BAT are based on those developed for the HETE-2 GRB Explorer. The algorithm continuously applies a large number of criteria which specify the pre-burst background intervals (typically 0-100 sec), the order of the extrapolation of the background rate (constant, linear, and parabolic with time), the duration of the burst emission test interval (4 msec to 32 sec, these are the so called Rate Triggers), the region of the detector plane illuminated, and the energy range (typically 4 different bandpasses). A second burst detection method is also implemented. Every 64 sec the detector array count rate map is processed through the FFT imaging algorithm (on the DSP) and scanned for point sources (the so called Image Triggers). All sources found are compared against an on-board catalog. Any new sources will constitute a new source and initiates the burst response procedure. Any known source with an intensity above a commandable level will constitute a interesting source and initiates the interesting-source response procedure. The table of threshold levels can be adjusted after launch to balance sensitivity against the number of false triggers, and to concentrate on specific subclasses of GRBs as they are discovered.

**Burst Imaging and Location Process** Once the trigger algorithm detects a count-rate excess in the detector, the data are analyzed to discover if this is due to a GRB. The IP extracts source and background data based on the energy range and time intervals flagged by the trigger, and then is converted to a sky map using an FFT-based cross correlation algorithm. The algorithm requires about 7 seconds for the DSP to generate a 1024×512 pixel image showing the locations of transient sources. If an excess in this sky map is found, its position can be determined to within a single 17-arcmin sky pixel. If a single rate trigger fails to produce a significant image excess, the BAT will check for subsequent rate increases that produce a stronger image. The ability to use imaging to eliminate false triggers is a primary advantage of the BAT, and allows us to set the trigger thresholds to a sensitivity that would be intolerable if there were no other method of confirmation.

Once the approximate source location is known, the DSP executes a back-projection algorithm which produces an image of that region with arbitrarily small pixel size (typically 1 arcmin), giving a peak with the intrinsic 22 arcmin FWHM PSF of the instrument. Centroiding this peak gives the source location at the statistical and systematic limits of the instrumental accuracy (1-3 arcmin depending on the intensity of the burst).

**Figure of Merit Algorithm**

The Figure of Merit (FOM) algorithm is part of the spacecraft’s autonomy that decides if the burst just detected by BAT is worth requesting a slew maneuver by the spacecraft. While the FROM does reside within the BAT flight software on the BAT Image Processor, the FOM provides a mission-wide function which will not be described further here.
Chapter 3

BAT Operating Modes and Data Types

3.1 Introduction

The BAT flight software produces several data products. As mentioned in the previous chapter, the BAT is a photon counting instrument. That is, each photon interaction is recorded separately by the flight electronics and flight computer. However, like other missions there is not enough on-board storage or downlink capacity to send all of these events to the ground. Instead, the BAT sends full event data for special triggers (i.e. for gamma-ray bursts), and otherwise makes binned products.

Normally, the BAT is searching for a new GRB trigger while in survey mode. In this mode, an energy spectrum is accumulated for each detector in the array on a time scale of approximately every 5 minutes. While this data allows an observer to recover the fluxes of sources over long time scales there are significant systematic errors that must be addressed, and have not yet been accounted for in the BAT software.

After a successful trigger, several products are produced in response to a gamma-ray burst. The most significant of these is a dump of event data which brackets the trigger time, with a total duration of about 10-15 minutes.

The BAT also produces several products all the time, regardless of whether there is a GRB or not. These are typically various array rates, housekeeping values, and trigger diagnostics.

A more complete description of these products is presented in the rest of this chapter.

3.2 GRB Products and Response

Most of the time the BAT is in survey mode. Trigger criteria are evaluated on a continuous basis. When the BAT detects a rate increase (in one of several energy bands or spatial regions), then it indicates a “rate trigger,” which in turn initiates on-board image processing to confirm a new point source. Typically, there are many rate triggers per hour, each given its own trigger number, but few are confirmed by imaging. On a periodic basis, the BAT flight software also performs image analysis independent of the rate triggering system, to search for slow rising GRBs and transients.

When the BAT receives a GRB trigger the BAT and the spacecraft have a preset set of responses. The typical response process is as follows, roughly in order of the time of production:
CHAPTER 3. BAT OPERATING MODES AND DATA TYPES

Figure 3.1: BAT data products for a typical GRB with a slew.

**BAT Rate Trigger** Fast-rising GRBs lead to a rate trigger.

**BAT Image Confirmation** The image processing step takes several seconds. If a significant new excess is detected, then a GRB is declared, and a TDRSS position message is sent to the ground.

**BAT Image-only Trigger** The BAT regularly performs image analysis, on roughly 1 minute, 5 minute, and snapshot-duration accumulations, looking for new sources regardless of the rate trigger status. If a new excess is detected, it may be declared a GRB or a “transient” depending on the duration.

**Spacecraft Slew** The spacecraft may or may not slew, depending on the merit of the burst and the observing constraints. The BAT products should be the same whether or not there is a slew.

**BAT TDRSS Light Curve** The BAT produces a total rate quick-look light curve of several hundred seconds around the time of the trigger. This light curve is acceptable for looking for crude features, but cannot be trusted for sensitive analysis, since the rates are contaminated by background events. This is especially a problem during the slew, when sources pass into and out of the BAT field of view, which causes rate variations.

**BAT Event Data** This is the most relevant and useful data for GRB analysis, since it allows background to be removed via “mask weighting.” Also, the user can choose any source position, temporal or spatial binning that they desire. GRB light curves and spectra can be extracted during the spacecraft slews as well as pointings. Note that the BAT event data may overlap in time with nearby survey intervals.

The response process and products as a function of time during a GRB are illustrated in Figure 3.1. Note that even if the spacecraft does not slew, or slews to another target (not the GRB
3.2. GRB PRODUCTS AND RESPONSE

As long as the source is in the BAT field of view, it should be possible to reconstruct the GRB light curve and spectrum.

3.2.1 Important Notes About Trigger Times

There are several important things to realize about trigger times. The BAT trigger time associated with each GRB is almost arbitrary. The trigger time does not necessarily correspond to the peak flux or the start of the burst. For example, the BAT may trigger on the first peak of a multi-peak burst; or, it may trigger on the second peak after a slew. The trigger time simply corresponds to the start of the time interval which the BAT flight software used to find a significant image excess. Therefore, when a precise definition of time is needed for scientific analysis, users should exercise their scientific judgement to assign the zero point of the time scale.

Also, it is important to know that for short bursts the trigger time may be off by up to 320 milliseconds. For short triggers (trigger durations < 64 milliseconds) the BAT flight software reports a trigger time which has been rounded down (earlier) to the nearest multiple of 320 millisecond. Thus, it is important to examine the entire range of T+0 to T+320 milliseconds of light curves of short triggers.

3.2.2 Description of Standard Products

The SDC makes several standard products for each burst, summarized in Table 3.1

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Covering Entire Burst</strong></td>
<td></td>
</tr>
<tr>
<td>swNNNNNNNNNNNNbev1s.lc</td>
<td>BAT Light curve with 1-second bins and 4 energy bands (includes data taken during slew)</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevms.lc</td>
<td>BAT Light curve with 64-msec bins and 4 energy bands</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevbu.gti</td>
<td>Burst time intervals (T50,T90,T100,Tpeak, and background intervals)</td>
</tr>
<tr>
<td><strong>Covering Pre-Burst Interval</strong></td>
<td></td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevpb_dt.img</td>
<td>BAT Detector image for pre-burst interval</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevpb_sk.img</td>
<td>BAT Sky image for pre-burst interval</td>
</tr>
<tr>
<td><strong>Covering Pre-Slew Burst Interval</strong></td>
<td></td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevps_dt.img</td>
<td>BAT Detector image for pre-slew burst interval</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevps_sk.img</td>
<td>BAT Sky image for pre-slew burst interval</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevps.pha</td>
<td>BAT Spectrum for pre-slew burst interval</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevps.rsp</td>
<td>BAT Response matrix for pre-slew spectrum</td>
</tr>
<tr>
<td><strong>Covering Slew Burst Interval</strong></td>
<td></td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevsl.pha</td>
<td>BAT Spectrum for slew burst interval</td>
</tr>
<tr>
<td><strong>Covering Post-Slew Burst Interval</strong></td>
<td></td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevas_dt.img</td>
<td>BAT Detector image for post-slew burst interval</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevas_sk.img</td>
<td>BAT Sky image for post-slew burst interval</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevas.pha</td>
<td>BAT Spectrum for post-slew burst interval</td>
</tr>
<tr>
<td>swNNNNNNNNNNNbevas.rsp</td>
<td>BAT Response matrix for post-slew spectrum</td>
</tr>
</tbody>
</table>

Table 3.1: BAT GRB standard product names and descriptions.

These products are designed to be used with standard software for imaging, timing, and spectral analysis in X-ray astronomy.
Spectra and response matrices have standard extensions, and are suitable for analysis with XSPEC.

Light curves are uniformly sampled and have multiple energy channels. They can be viewed easily with the general FITS viewer ‘fv’, but also with other timing analysis software such as XRONOS (lcurve, powspec, etc.). Multiple channel light curves can be handled in XRONOS using the “feN” and “leN” energy band options.

Both detector and sky image files are produced. Detector images are processed with BAT deconvolution software (*batfftimage*) to produce sky flux images. The first four extensions of each file are the detector/sky images in each of four standard energy bands. There are also GTI and EBOUND extensions. Following that, there is a BAT_DPI_TOT/BAT_IMAGE_TOT extension which contains the full energy band. All of these images are standard FITS image extensions which can be viewed with DS9, fv, or your image viewer of choice.

The “pre-slew” sky image (swNNNNNNNNNNNNNNNNNNNbevps_sk.img) has a special BAT_CATALOG extension at the end, which was produced by *batcelldetect*. The columns RA_OBJ, DEC_OBJ of this file represent a fit to the position of the source using all pre-slew data. Because the full pre-slew burst interval usually contains more data than the BAT on-board trigger interval, this position will usually be a more sensitive measure of the source position.

The “GTI” file contains multiple time intervals of interest for a gamma-ray burst, including T50, T90, T100 (estimated total burst interval), and various background intervals. These intervals are computed by the *battblocks* task. They are standard GTIs which can be use for time filtering of Swift data.

The time intervals identified in the table are defined in the following way:

- **pre-burst** means before any significant burst emission;
- **pre-slew** means burst emission before the spacecraft slews to the burst location;
- **slew** means burst emission during the slew to the burst location;
- **post-slew** means burst emission after the slew to the burst location;

These intervals are entirely independent of the BAT trigger time.

Not all of the data files will be present for every burst. In some cases, the spacecraft does not slew to the burst, so there may not be slew or post-slew products. Also, occasionally the *battblocks* task fails to find sensible time intervals, in which case no standard products will be present (this most often occurs for faint, short bursts).

### 3.3 Non-GRB Products

The BAT also generates products that are not associated with a particular gamma-ray burst. These are typically produced continuously, or at regular intervals in the course of normal operations.

Descriptions of the formats of these products can be found in Appendix A (“BAT Data Formats”).

#### 3.3.1 Survey Data

When not responding to a GRB trigger, the BAT is typically in survey mode. In this mode, the event data from the array is collected into detector plane histograms (DPHs) by the flight software.
These histograms have an 80-bin spectrum for each detector in the array (i.e. 32k detectors × 80 bins) during the integration period. The energy bin widths are variable from one energy bin to the next, but have remained constant in time throughout the mission to date. Each survey DPH file records the energy binning used (in the FITS file, there is an EBOUNDS extension which contains this information). Figure 3.2 shows a representation of a survey DPH. It is three dimensional in the sense that there are two spatial dimension and one spectral dimension.

Survey data have a crude energy correction applied on-board, but should usually be further corrected using the task baterebin.

Typically, DPHs are integrated for five minutes, but this duration is not always the same. In some cases longer or shorter durations have been used for operational reasons (e.g., telemetry reduction or to get diagnostic information about the instrument). Also, survey intervals are always terminated when the spacecraft begins a slew to a new target or when entering the SAA. Generally speaking, the survey data cannot be used to search for time variations on time-scales shorter than 5 minutes.

Survey data is particularly difficult to analyze. It contains the longest sky integrations, and thus is potentially the most sensitive to hard X-ray sources, but also contains the greatest number of systematic error contributions. Currently the BAT team does not support general survey data analysis.

### 3.3.2 Rate Data

The BAT produces continuous streams of rate data. While these may be used for scientific analysis, they are also used by the on-board flight software for trigger searching and reporting the instrument state of health.

The rate data include:

- **one-second** rates (1 sec sampling; full energy band; full array);
- **64 msec** rates (64 msec sampling; four energy bands; full array);
- **quadrant** rates (1.6 sec sampling; four energy bands; four separate spatial quadrants);
Figure 3.3: Example of BAT 1 second rates. All variations shown are due to the space environment and not an astrophysical source.

- **mask-tagged** rates (1.6 sec sampling; four energy bands; three distinct sources);

- **maximum** rates (maximum counts recorded for multiple time scales, energy bands and spatial regions).

All rate data except for mask-tagged rate data are **not** background subtracted. This means that the hard X-ray background and particle backgrounds are included in the rates. Also, since there is no per-detector information recorded in the rate data, it is possible for noisy detectors to generate rapid spikes which may emulate a gamma-ray burst or other rapidly varying phenomenon. Thus, while rate variations may be suggestive of an astrophysical event, conclusive proof must be obtained from image analysis.

Figure 3.3 shows an example 1 second BAT rate light curve, with a typical background level and variation for a portion of the orbit. Swift spacecraft orbits which pass near the SAA can produce BAT rates as high as $10^5$ counts per second.

Mask tagged data are generated on-board by the BAT flight software, and contain a crude light curve of up to three astrophysical sources. These sources are chosen automatically by the flight software at the time of data collection. The raw mask tagged data may be converted to background subtracted light curves using the batmasktaglc software task.
3.3. NON-GRB PRODUCTS

3.3.3 Maps

The BAT flight software periodically produces housekeeping maps which are useful for BAT analysis. The most important of these are detector enable/disable maps and gain/offset maps. **Detector enable/disable** maps represent which detectors have been automatically disabled. Usually detectors are disabled because they are noisy. The number of enabled detectors must be taken into account when performing mask weighting and image analysis.

**Gain/offset** maps are produced by the automatic calibration system. They record the approximate pulse-height-to-energy calibration for each detector. This map, in combination with ground calibration files, is used to produce calibrated event lists (using bateconvert) and survey files (using baterebin).

**Mask weight maps** are associated with mask tagged rates, and are needed to produce background subtracted light curves. In the standard FITS products, mask weight maps should be attached to the mask tagged light curve files.

3.3.4 Trend Products

Various other tables and maps are generated as well. These can be broken down into several broad categories:

- detector maps used in on-board imaging ("scaled" maps);
- calibration pulser maps used to produce gain/offset maps on-board;
- trigger diagnostic tables for rate (long and short) and image trigger systems;
- the on-board source catalog;
- americium tagged-source spectra ("block spectra") and survey-type data ("Americium DPHs"), which are used to calibrate the absolute energy scale;
- science housekeeping ("DAP HK"), plus various diagnostics;
- engineering housekeeping;
- debug stream (an ASCII log of the flight software activities);

As noted above, a more detailed description of the format and content of all the products can be found in Appendix A.
Chapter 4

Introduction to BAT Analysis

4.1 Introduction

This chapter contains a discussion of the basics of coded mask analysis, differences from direct imaging systems, and useful science information about the BAT, like the coordinate system and units.

4.2 Coded Apertures: Basic Concepts

The BAT instrument has a coded aperture. The basic premise of a coded aperture is quite different from normal imaging optics, primarily because there are no focussing optics. All of the “optical” elements of a coded aperture are passive, involving the casting shadows from the aperture (mask) onto an imaging sensor.

Figure 4.1: Basic BAT imaging scheme. A gamma-ray source illuminates the coded aperture (mask), and casts a shadow onto a position sensitive detector.
CHAPTER 4. INTRODUCTION TO BAT ANALYSIS

Figure 4.2: Top view of the BAT mask. The cut-outs at the edge correspond to portions of the mask where support structures were mounted.

For example, consider Figure 4.1. This figure shows a gamma-ray source which illuminates the aperture. The portions of the aperture which are blocked by lead tiles absorb the gamma-rays, and the portions which are not blocked allow the rays to pass through. The result is a distinctive shadow pattern which is measured by the detector array below.

Coded aperture imaging is a logical extension of the *pinhole camera* concept. For a pinhole camera, there is a single small aperture. Each source at a different position on the sky casts a different narrow-beam shadow onto the detector array, and thus each source can be uniquely distinguished from the other by directly examining the intensity in the detector plane. One disadvantage to a pinhole camera is that light collection efficiency is low: nearly 100% of the aperture is blocked.

A coded aperture system improves the efficiency situation by placing more than one opening in the aperture. The disadvantage to this approach is that the shadow patterns of different sources can overlap, so the measured intensities no longer directly reflect the distribution of sky intensities. On the other hand, the aperture pattern is known, and each distinct source casts a unique shadow pattern on the detector array. Thus, it is possible to disentangle the fluxes of individual sources using special software. Figure 4.2 shows the aperture pattern of the entire BAT mask which is used in the processing.

Figure 4.3 shows the mechanical layout of the BAT, as seen from the side. The top horizontal structure supports the mask tiles, and the bottom horizontal structure contains the detector array. The diagonal structures support the mask, but leave the central volume between the two planes essentially empty. The *focal length* of the instrument – the distance between the two planes – is approximately 1 meter. This distance, coupled with the sizes of the aperture and array, determine the basic imaging properties of the instrument such as its field of view and point spread function.

Extracting light curves and spectra involves performing the coded aperture analysis in multiple time and/or spectral bins. The BAT software is designed to produce standard light curve and spectral FITS file products that can be used with other standard analysis software (although see below).
4.3 Coded Aperture Analysis for X-ray Astronomers

For an observer with experience in X-ray astronomy, coded aperture analysis with BAT may appear to be similar to their experiences, but they are not. This section gives a description of some of the major differences.

**BAT has no focussing optics.** This means that most of the familiar analysis operations of X-ray and optical astronomy do not apply.

**The BAT has two distinct spaces: detector space and sky space.** The BAT software is used to convert from detected counts to sky fluxes. Figure 4.4 compares the two kinds of images. Another example, Figure 4.5 shows the mask modulation pattern of a bright source, Sco X-1, as detected by BAT. Note however that for all but the brightest sources the mask modulation pattern will not be visible to the naked eye.
Each detector can be illuminated by many points on the sky; and each point on the sky illuminates many detectors. This entanglement means that one must use the special BAT software to do the disentangling.

You cannot “select” photons with spatial regions. The BAT software does not reconstruct where each individual photon came from. Neighboring sky image pixels are highly correlated, so it does not make sense to “select photons” within a region. For images, it is preferred to use the PSF fitting method of the batcelldetect task. For light curves and spectra, batmaskwtevt is used to select a point of interest on the sky, and batbinevt is used to extract the fluxes.

The noise is gaussian, not poissonian. BAT software provides a background-subtracted flux for a source. The deconvolution technique produces errors which are gaussian in nature, not poissonian. This means that any downstream software that you use must be able to cope with gaussian errors. It also means that you do not need to use “grouping” strategies commonly used with XSPEC in X-ray astronomy (such as GRPPHA or “setplot group”).

The BAT has a very large field of view. The complete field of view is approximately $120 \times 60$ square degrees, although sensitivity is lower at the edges. The solid angle is approximately 2 steradians. Since this is a significant fraction of the sky, it is likely that BAT will observe a given source even when Swift is not pointed at the source. The BAT typically covers $\sim 75$-$85\%$ of the sky on any given day.

The BAT is background dominated. The typical full array background count rate is about 10,000 to 12,000 counts per second. Even the Crab is background dominated (1 Crab $\sim 10\%$ of background). Thus, the most sensitive analysis will involve removing as much background contamination as possible.
4.4 How Sky Fluxes are Reconstructed

As mentioned in the previous sections, sky fluxes must be reconstructed from the detector data. This section describes how the reconstruction is done in basic terms.

The starting point is a BAT detector plane image (DPI), accumulated from the raw data using the task `batbinevt`. The raw data can be either events or survey histograms (DPHs). The result is a $286 \times 173$ counts map giving the detected BAT counts in each detector.

The reconstruction technique involves cross-correlation — or matched filtering — of the detected counts with a the mask aperture pattern. At its most basic level, this process involves ray-tracing all possible shadow patterns from all possible locations on the sky onto the detector plane. For each possible pattern, the shadow pattern is multiplied by the detected counts, and the result is summed. When a source is present at a given sky position, there will be an excess of counts in detectors where the mask was open, and a deficit of counts where the mask was blocked. The ray-traced shadow pattern for the same position should have exactly the same pattern of excesses as the measured counts excesses, so the multiply-and-sum operation will produce an enhanced correlation for that position. For other sky positions where there is no source, the mask shadow pattern is uncorrelated with the detected counts, and so a small correlation sum is produced. The result is a background subtracted mask weighted flux for each position on the sky. The normalization of BAT mask weighted fluxes is discussed later in this chapter.

For image analysis, all possible locations on the sky are checked, and set of all correlations is flux map. Positions with large correlation correspond to a point source, and positions with noisy correlations can be considered to be blank. The `batfftimage` task creates images, and `batcelldetect` detects and extracts fluxes for individual sources in those images.

For light curve and spectral analysis of a previously known sources, it is possible to bypass most of the machinery of batfftimage and batcelldetect. A technique known as mask-weighting allows one to assign a ray-traced shadow value for each individual event (using the `batmaskwtevt` task). Once this has been done, it is possible to compute the correlation with any time or spectral binning desired (using the `batbinevt` task).

Accurate image reconstruction of BAT data is still an area of active investigation by the BAT team. For long exposures, the images are dominated by non-statistical spatial variations which spoil the cross-correlation algorithm. For observations longer than a few hundred seconds, systematic errors start to become important.

4.5 BAT Field of View and Partial Coding

The BAT field of view can be computed by determining which parts of the sky could illuminate the detector array. This computation involves knowing the size and shape of the mask, the detector array, the focal length of the instrument, and whether any detectors have been disabled.

Figure [4.6] shows the shape of the BAT field of view in sky coordinates.

An important concept to understand is partial coding. When a source is on-axis, its aperture shadow fully illuminates the detector array, and so it is fully coded. BAT has a small region near
the center of its field of view which is 100% coded. However, as a source moves farther off axis, only a portion of the aperture shadow will illuminate the detector array, and the rest are blocked by the shield which surrounds the BAT instrument. The partial coding effect is analogous to the off-axis vignetting effect for classical telescopes. The fractional illumination is called the partial coding fraction. For example, the source in Figure 4.5 is about 98% partially coded, since about 2% of the detectors in the lower right corner are not illuminated by the source. Eventually the source is so far off axis that it can illuminate no detectors, and so the source becomes completely undetectable using coded aperture techniques. Figure 4.6 shows various partial coding contours in sky coordinates.

The BAT field of view which contains at least 10% partial coding is about 2.2 steradians (>50%, 1.5 steradians; >90%, 0.5 steradians). Of course, as the partial coding decreases, the instrument sensitivity also decreases. This is why gamma-ray burst light curves are typically noisier at their start, when the source is detected off-axis; and less noisy by the end, when the spacecraft has slewed to put the source in the fully coded region.

The BAT software automatically corrects for partial coding effects by default.

4.6 BAT Coordinates

Various coordinate systems are used in analyzing BAT data. More detailed information can be found in the document "BAT Coordinates Definition and BAT Detector Layout," available from the Swift CALDB documentation area. Here is a short summary.

BAT detector plane images and detector plane histograms (DPIs and DPHs) have spatial dimensions DETX and DETY. These refer to the individual pixel positions on the detector array (DET X is numbered from 0-285; DET Y from 0-172). Positions of detector gaps are filled with zeroes.

Most users will be interested in images with celestial coordinates. By default, a standard WCS celestial coordinate system is be attached to each BAT sky image, which should be readable by any standard FITS image viewer.

BAT sky images are most naturally expressed in tangent plane coordinates. This is because the coded mask image analysis translates directly into a tangent plane projection of the sky. All BAT pixels have the same spacing and size in these coordinates; in any other projection BAT pixels will
appears distorted. In terms of true solid angle however, BAT sky pixels far off-axis are significantly smaller than the on-axis ones (by as much as a factor of two). While in principle this allows one to determine source positions more accurately, the sensitivity is also much less far off-axis due to partial coding effects.

All BAT images contain a secondary tangent plane coordinate system (WCS-T), which is labeled with the axes IMX and IMY. These coordinates are the tangent of the off-axis angle in the X and Y directions (the boresight is centered at 0,0).

The BAT also has an instrumental longitude/co-latitude system. The co-latitude component, “theta,” measures the off-axis angle in degrees. The other component, “phi,” measures the longitude of a source, starting at the instrument X axis. Both of these values are reported in BAT GRB products.

4.7 Definition of BAT Count Units and Corrections

This section describes the “mask weighted” counts and rates that are computed by the procedures described above. Generally speaking, if the default corrections are applied, light curves and spectra will be adjusted to be comparable to an on-axis source, independent of the number of illuminated or active detectors. However, if the user chooses non-default corrections, then this property may no longer hold true. Each of the corrections is discussed below.

4.7.1 Definition of BAT Mask-Weighted Counts

BAT counts derived from mask weighting procedures or sky imaging – batmaskwtevt, batmaskwting or batfftimage – and using the default corrections are defined in the following way:

Background subtracted counts per fully illuminated detector
for an equivalent on-axis source

Rate is defined similarly, per unit time.

Let us examine what each of these words means. Next to each phrase is the corresponding correction that can be used in the analysis tasks. These corrections apply to both the mask weighting programs and the sky imaging programs (which fundamentally apply the same technique).

Background subtracted counts...

The units are counts, not photons, which customary for X- and gamma-ray instruments. To convert to photon fluxes, one must generally fit the spectrum to an a priori model. However, the mask weighting technique provides automatic background subtraction.

Also the ‘background’ that is subtracted includes the effects of flux that passes through the lead mask cells without scattering. Both foreground and background include counts from photons that scatter off the lead mask cells and other parts of the BAT and Swift structure.

... per ... detector (correction ‘ndets’)

The counts are normalized to the number of enabled BAT detectors. This compensates for the possibility that some, or many, BAT detectors are disabled for a given observation.
CHAPTER 4. INTRODUCTION TO BAT ANALYSIS

... per ... *illuminated* detector ... (correction 'pcode')

The ‘pcode’ correction normalizes the counts by the number of illuminated detectors. Since the BAT is a wide-field instrument it is possible, indeed likely, that sources will be at the edge of the field of view. This means that only a fraction of the detector array is illuminated by through mask, i.e. the source is partially coded, and thus will produce fewer counts. This correction normalizes the partial coding fraction to unity for all sources.

... per *fully illuminated* detector ... (correction 'maskwt')

Even when detectors are illuminated by a source, most detectors are not 100% illuminated. In general, there is a distribution of detector illumination fractions, ranging from 0% (fully shadowed by the mask) to 100% (fully illuminated by the mask). The mask weighting technique used by the BAT software combines all events to produce background-subtracted counts in a “mean” fractionally illuminated detector. The ‘maskwt’ correction renormalizes the counts to be those received in a fully illuminated detector. This accounts for both the open fraction of the mask (50% for the BAT), plus the per-detector partial illumination.

The ‘maskwt’ correction should be distinguished from the partial coding correction. The partial coding correction adjusts for the fractional number of detectors that a source could illuminate through the mask, regardless of which mask cells are blocked or open. The ‘maskwt’ correction adjusts for the partial illumination of each individual detector.

... for an equivalent on-axis source* (correction 'flatfield')

Off-axis sources have a different count rate because the projected area of the BAT detectors is changed by projection effects. Generally, the effective area is reduced off-axis because of the “cosine effect.” However, the effective area increases slightly for mild off-axis angles because detector sides are illuminated. The ‘flatfield’ correction adjusts for these effects, to create a count rate for an effective on-axis source. In this manner, it will be more practical to compare detections of the same source at different positions in the field of view.

It should be noted that this correction is primarily due to geometric effects. It is also understood that off-axis illumination changes the detector response in non-subtle ways, which is not corrected for by the ‘flatfield’ correction.

4.7.2 Converting to a counts per area

The area of a single BAT detector is 0.16 cm$^2$. Thus, the conversion between “per unit area” and ”per fully illuminated detector” is:

$$counts_{\text{cm}^2} = \frac{counts_{\text{fully illuminated det}}}{0.16}$$
Chapter 5

BAT Analysis Procedures

5.1 Introduction

This chapter provides information and recipes regarding the analysis of BAT data. The primary focus of the information is extracting light curves, spectra and images from the event data associated with gamma-ray bursts.

In response to a gamma-ray burst, the BAT instrument sends raw event data to the ground. This event data includes all events from particle background, noisy detectors, and non-imaged X-rays. The user must take advantage of the coded mask analysis in order to remove the effects of these nuisance components. The technique described here is known as “mask-weighting,” which is equivalent to a specialized form of ray-tracing.

It should be noted that event data allows maximum flexibility to analyze the data. For example, the user can select which time intervals they are interested in, and the desired spectral binning. (Contrast this with survey data, which has a fixed integration time and energy binning scheme.) The BAT software and hardware has also been designed to allow the user to extract light curves and spectra while the spacecraft is slewing. This is important because the Swift spacecraft automatically slews to most bursts, and a significant fraction of the bursts have prompt emission during the slew.

For most GRB analysis, users should begin with the ‘batgrbproduct’ script. This script, described in the following sections, performs a standard analysis and makes products that will satisfy many users.

However, this chapter also describes some of the more fundamental building-block analysis procedures, including:

- Energy calibration of events
- Making a detector quality map
- Mask Weighting (Ray Tracing) for a known source position
- Extracting a light curve
- Extracting a spectrum
- Correcting the spectrum
- Generating a response matrix
- Making sky images
5.2 Complete GRB Processing Script

This section describes how to run automatic GRB processing script, batgrbproduct. This script produces light curves, spectra and summary reports that many common users would want. It also produces a detailed log of commands executed, which users can cut and paste to make their own products.

5.2.1 Introduction

In late 2006, a new version of the BAT software became available which includes the task batgrbproduct. This task generates standard products for a GRB observation, using conservative data filtering and sensible defaults. The BAT team recommends that most users start with this script for their analysis of a GRB. The BAT team itself uses this script to produce the results posted on GCN circulars. The batgrbproduct script performs the mask weighting operation (ray-tracing), creates a quality map, and generates light curves spectra and response matrices with various binnings. Sky images are produced, and if possible, a refined GRB position is derived from these positions.

The task also produces a detailed log file which users can consult when they want to make their own custom light curves or spectra.

5.2.2 Prerequisites

- A complete Swift observation containing GRB event data (usually this will be in observation segment 0, labeled NNNNNNNNN000 with three final zeroes);
- a recent version of HEASoft tools (version 6.1.2 or later);
- a recent version of CALDB.

It is assumed that the correct energy calibration has been run on the event file. Please see the section on applying an energy calibration to BAT event data for more information.

5.2.3 Change to Data Directory

Change to the directory containing the observation, but not into the observation directory itself. Extract the observation data if needed. For example, if the observation directory was /data/grbs/00145675000, then issue the following command,

```
    cd /data/grbs
```

5.2.4 Creating GRB Products Automatically

To run the script, issue the following command,

```
    batgrbproduct indir=./00145675000 outdir=./00145675000-results
```

where
5.2. COMPLETE GRB PROCESSING SCRIPT

- \texttt{indir} is the observation directory
- \texttt{outdir} is the output results directory

The script will print diagnostic output to the console as it runs, and may take tens of minutes to an hour to complete, depending on the speed of your computer and hard drive. When the script finishes, it should print an inventory of the results.

5.2.5 Results

The results are stored in the ‘\texttt{outdir}’ directory you specified on the command line. A summary of the results of the analysis is stored in the file \texttt{report.txt} in that directory.

Light curves, spectra and images are stored in the \texttt{lc}, \texttt{pha} and \texttt{img} subdirectories, respectively. Response matrices are generated for the pre-slew and post-slew time intervals. Burst duration measures are reported in \texttt{report.txt}, and also are contained in the \texttt{gti} subdirectory.

More information about the results can be found in the documentation for the \texttt{batgrbproduct}.

5.2.6 When Things Go Wrong

There are several situations where \texttt{batgrbproduct} may not be able to perform a totally automatic analysis.

If the burst is very faint, especially a short hard burst, the automatic software is not always able to select the right time interval. If that is the case, the script defaults to the time interval reported in the TDRSS position message, which may not always be appropriate. Users may always override the default time interval by hand, using the \texttt{trigtime/trigstop/backstrt/backstop} parameters. The user can also override the default position by using the ‘\texttt{ra}’ and ‘\texttt{dec}’ parameters. By default, \texttt{batgrbproduct} uses the reported \texttt{BAT} position, and not the XRT position.

The Swift spacecraft does not always respond in the same way to a GRB. For example, a slew does not always occur to the burst. In that case, there will be no post-slew images or spectra. There will also be no post-slew spectrum if the burst emission does not extend past the slew to the source. Warning messages will be produced, but they can be safely ignored.

Finally, it is worth mentioning that a burst occasionally is observed already emitting when it enters the BAT field of view. In that case, the automatic duration measures will of course be unreliable. To improve the operation of the script, you should set \texttt{tbkgsub=NO} for those kinds of bursts.

5.2.7 Making Your Own Custom Products

\texttt{batgrbproduct} makes a standard set of light curves and spectra which will satisfy many users. However, some users need custom products that are not made automatically. These might be light curves with different binning, or spectra with different time cuts. \texttt{batgrbproduct} does not directly support custom options, but it does produce a detailed log of each command called \texttt{process.log}. This file also contains the result of the command.

You should scan through \texttt{process.log} using your favorite text editor, and locate the light curve or spectrum which most closely matches your need. Then copy the command from the file to the command line and modify it as needed.
5.2.8 Spectral Fitting

For more tips, warnings and other information on spectral analysis with BAT, please see the BAT Spectrum and Response section.

5.3 Energy Calibration of Events

BAT events telemetered to the ground have raw pulse heights. Individual detectors have different energy scales which cannot be combined. Energy calibration of the events is required for any useful analysis. Typically, the SDC performs the energy calibration in their automatic pipeline. This section provides information on how to check for the appropriate calibration procedure and to re-run the calibration when necessary.

The energy scale is based on both an on-board pulser (electronics) calibration, and ground analysis of the instrument performance (both pre- and post-launch).

5.3.1 Reasons to Re-run the Energy Calibration

- If it has not been run before (i.e. missing gain/offset file or something like that).
- To capture improvements in the newer software/ground calibration (see below).

Checking the event file for the correct ground calibration files

You can check the event file to see if its energy scale was correctly calibrated.

Run the command:

```
fkeyprint sw00145675000bevshsp_uf.evt.gz GAIN
```

You should see the following results (among other entries):

```
GAINAPP = T / Gain correction has been applied
GAINMETH= 'FIXEDDAC' / Cubic ground gain/offset correction using DAC-b
```

If these keywords don’t exist, or if GAINAPP is false or GAINMETH is not ‘FIXEDDAC’, then you should run the energy calibration tool.

5.3.2 Prerequisites

- BAT event data
  - either original (in obsid/bat/event/swNNNNNNNNNNbevshsp_uf.evt.gz); or
  - cleaned event list (See “Producing a Cleaned Event List”);
- BAT gain/offset file (in obsid/bat/hk/swNNNNNNNNNNbcb*.fits.gz);
5.3. ENERGY CALIBRATION OF EVENTS

The SDC should be producing a gain/offset file for every observation. If it is not present, then you must locate the nearest one in time in the trend data in the archive. (see the Swift FTP site in /swift/data/trend/YYYY_MM/bat/bgainoffs).

In this case, let us recompute the energy scale for GRB 050713A (obsid 00145675 000) using the original event file.

5.3.3 Initial Steps

Change directories to the location of the event data.

cd 00145675000/bat/event

If the event file is compressed, unzip it. The energy calibration task modifies the event file in place, and gzipped files cannot be modified.

gunzip sw*.evt.gz

5.3.4 Compute Energy Calibration for BAT Events

The energy calibration task for events is called ‘bateconvert’. Here is an example run,

bateconvert infile=sw00145675000bevshsp_uf.evt \
   calfile=./hk/sw00145675000bcb01deg00ab.fits.gz \
   residfile=CALDB pulserfile=CALDB fltpulserfile=CALDB \
   outfile=NONE calmode=INDEF

- infile is the event file to be calibrated
- calfile is the BAT gain/offset file
- residfile, pulserfile and fltpulserfile are ground-derived calibration files that contain the BAT energy-to-channel conversion
- outfile=NONE indicates that the operation should be done on the file in-place
- calmode=INDEF means to use the default calibration method

5.3.5 What to do if no gain/offset map is present

Gain/offset maps are produced by the BAT flight software periodically to record the pulse height scale of the electronics. While the BAT electronics appear to be quite stable, these regular calibration activities are intended to account for any systematic changes over time. Gain/offset maps are used to compute calibrated event energies from raw pulse heights during ground processing, as shown above.

It is important to understand that the calibration activities are not synchronized with the observing plan. The gain/offset map may have been produced during a previous observation.

Normally the SDC retrieves the proper gain/offset map for GRB data and includes it with the data set. If there are processing errors, or for non-GRB observations, the gain/offset maps may be missing. If that is the case, the gain/offset map must be found manually from the trend data.
Search in the BAT trend data by month. See `/swift/data/trend/YYYY/MM/bat/bgainoffs` on the HEASARC FTP site.

You want the next earliest gain/offset map produced before the event data in question. Since the trend data are not named by time but by observation, this is especially inconvenient. Rather than searching all the files in the given month, you can narrow your search. In the event file, look for the LDPGAIN and LDPOFFST keywords,

```
ftlist sw00145675000bevshsp_uf.evt.gz K | grep LDP | grep Index
```

which produces the following

```
LDPGAIN = 171. / LDP Gain Index
LDPOFFST = 478. / LDP Offset Index
```

These are semi-unique identifiers of which map were used. If you convert these two index values to hexadecimal, you get gain=0x00ab, offset=0x01de. This tells you which kind of filenames to look for in the trend data. You would use a wildcard mask like this:

```
ls sw*bcbo01deg00ab.fits*
```

Note how the offset index (01de) and gain index (00ab) have been incorporated into the filename. This command produces the following three files:

```
sw00067672001bcbo01deg00ab.fits.gz  sw001455675000bcbo01deg00ab.fits.gz
sw00145581002bcbo01deg00ab.fits.gz
```

All of these files should have identical gain/offset contents. However, it is prudent to download each of them and use the one closest in time to the event data of interest according to its TSTART keyword.

### 5.4 Making a Detector Quality Map

Once a BAT event list has been cleaned of diagnostic events and the energy calibration has been applied, the full detector array should be scanned for anomalous detectors. These are detectors that have too high or too low counts, compared to the expected statistical distribution. This section describes how to make a detector quality map, which is used in all following steps, including performing mask weighting (ray tracing), extracting spectra, light curves, and images.

#### 5.4.1 Reasons to Create a BAT Quality Map

- If it has not been run before (i.e. missing files).
- To capture improvements in the newer software/ground calibration.
- To focus on a narrow energy or time range where some detectors became noisy.
5.4. MAKING A DETECTOR QUALITY MAP

5.4.2 Prerequisites

- BAT detector enable/disable map (in obsid/bat/hk/swNNNNNNNNNNbdecb.hk.gz);
- BAT event data (in obsid/bat/event/swNNNNNNNNNNNbevsh*.evt.gz);
- Access to Calibration Database (CALDB).

The SDC should be producing a detector enable/disable file for every observation. If it is not present, then you must locate the nearest one in time in the trend data in the archive (see below).

The process involves three main steps. First, a detector plane image of the event data is created. Second, using the observation time of the image, a master quality map is retrieved from CALDB. Finally, the detector plane image is searched for noisy detectors which may have flared up during the observation, using the task `bathotpix`.

In this case, let us create a quality map for GRB 050713A (obsid 00145675000).

5.4.3 Initial Steps

Change directories to the location of the event data.

```
cd 00145675000/bat/event
```

5.4.4 Create a Detector Plane Image

Create a detector plane image for the time or energy range of interest. This is a temporary file which will be checked for noisy detectors.

For our example, we will simply sum all the event data, including all times and all energy ranges. This should normally be acceptable, since summing over a long period of time should produce the most conservative number of detectors to exclude.

```
batbinevt infile=sw00145675000bevshsp_uf.evt.gz outfile=total.dpi \ 
  outtype=DPI timedel=0 timebinalg=u energybins=- \ 
  weighted=NO outunits=COUNTS clobber=YES
```

- `infile` is the event file accumulated;
- `outfile` is the temporary detector plane image;
- `outtype=DPI` indicates that a detector plane image will be created;
- `timedel=0` indicates to sum all the data into one map. It would be possible to set `tstart=` and `tstop=` to narrow the range of time if desired;
- `energybins=-` indicates to sum all energies. A narrower energy range can be given as `ELOW-EHIGH`;
- `weighted=NO` indicates to make an unweighted detector map
- `outunits=COUNTS` indicates that the map should be in units of counts instead of rate;
5.4.5 Retrieve Known Problematic Detectors

The BAT team maintains a list of known bad detectors, which is stored in the Calibration Database (CALDB). These maps only contain detectors (usually detector modules) which were known for operational reasons to have issues which affect science analysis. The user is still responsible for filtering the data for noisy detectors (see the next section) or other obvious problems. The ‘batdetmask’ task is used to retrieve the appropriate detector quality map from CALDB.

```
batdetmask date=total.dpi outfile=master.detmask clobber=YES \
  detmask=../hk/swt0142921337bdecb.fits.gz
```

- date tells batdetmask to retrieve the observation date from the detector plane image you just created
- outfile is the output master quality map for the observation
- detmask is the detector enable/disable map (usually in the bat/hk directory)

We use the detmask parameter to combine the existing flight enable/disable map with the quality map based on ground analysis.

5.4.6 Find Noisy Detectors

Noisy detectors are found with the bathotpix task.

```
bathotpix infile=total.dpi outfile=total.qmap \
  detmask=master.detmask clobber=YES
```

- infile is the detector plane image you just created
- outfile is the resulting quality map
- detmask is the quality map we made in the previous step

5.4.7 Where to Use the Quality Map

The quality map should be used in all subsequent analysis. In particular, it should be used during the mask weighting process (using the batmaskwtevt task), and in the generation of all light curves and spectra (using the batbinevt task) and images (using the batfftimage task). All of these tasks have an input parameter called detmask= which accept a detector quality map. This information is used to exclude noisy detectors, and to properly normalize the resulting output fluxes.

5.4.8 The Difference Between the Enable/Disable Map and the Quality Map

There is a difference between the enable/disable map and the quality map we just produced. The enable/disable map is produced by the flight software, and records which detectors were disabled at the time. Thus, this map indicates which detectors were noisy in the past, were disabled, and now have zero counts. The quality map contains that information, but also excludes any currently noisy detectors, i.e. detectors with too many counts. The BAT flight software does not catch all noisy detectors. Some detector noise is sporadic, or below the count rate threshold for disabling. Thus, the best quality map is a combination of both the disabled detectors and the currently noisy ones.
5.4.9 What to do if no detector enable/disable map is present

Detector enable/disable maps are periodically produced by the BAT flight software to indicate which detectors have been disabled in flight. When noisy detectors appear and are detected by the flight software, they are automatically disabled to prevent them from interfering with the BAT triggering process.

It is important to understand that the detector enable/disable map production is not synchronized with the observing plan. The map may have been produced during a previous observation.

Normally the SDC retrieves the proper map for GRB data and includes it with the data set. If there are processing errors, or for non-GRB observations, the detector enable/disable maps may be missing. If that is the case, then the map must be found manually from the trend data.

Search in the BAT trend data by month. See /swift/data/trend/YYYY_MM/bat/bdetflag on the HEASARC FTP site.

You want the next earliest detector enable/disable map produced before the event data in question. The files are listed by time (expressed as mission elapsed time, or MET), so one needs to scan the list until the right time is found.

First, determine the time of the data you are interested in. Usually you can look at the TSTART keyword,

\[
\text{ftlist} \text{ sw00145675000bevshsp_uf.evt.gz K | grep TSTART}
\]

which produces the following

\[
\text{TSTART} = 142921443. / \text{Start time}
\]

A quick scan of the trend directory in /swift/data/trend/2005_07/bat/bdetflag shows that the next earliest detector enable/disable map is swt0142921337bdcb.fits.gz. In this case the time listed in the file name is about 106 seconds before the start of the event data.

It is not vitally important to get exactly the right detector map. Having a few detectors incorrectly labeled as enabled or disabled will not substantially alter the analysis.

5.5 Mask Weighting (Ray Tracing)

Mask weighting (also known as ray-tracing) is the fundamental operation required to produce background-subtracted light curves and spectrum from the BAT. This section shows you how to check the event file for how it was ray-traced; how to make a new set of mask weights; how to make an auxiliary ray tracing file. It is important to stress that the mask weighting approach works for one source at a time. If you are interested in several sources in the same observation, then you should make one copy of the event file for each source, and perform the mask weighting calculation for each separately.

5.5.1 Introduction

Mask weighting is required for all BAT light curve and spectral analysis. This is the first operation needed before background-subtracted light curves or spectra can be analyzed.
CHAPTER 5. BAT ANALYSIS PROCEDURES

Figure 5.1: Schematic diagram of mask weighting operation.

The BAT has a coded-aperture imaging system, which means that does not contain any focusing optics, and point sources on the sky are not imaged to a single point on the BAT detector array. BAT imaging and spectroscopy involves cross-correlating the detected shadow pattern with the expected shadow pattern. The first step in this process for GRB analysis is to determine the degree of shadowing for each event recorded during the GRB via ray tracing, and convert shadowing fraction into a weight. At a later step, these weights can be summed to produce a light curve or spectrum.

BAT mask weights are only valid for a given position on the sky! If you are interested in a different (or revised) position, then you must re-run the mask weighting.

5.5.2 Reasons to Re-run the Mask Weighting

The Swift Data Center runs the mask weighting tool in the automated pipeline. In principle, you will not need to re-run it yourself. However, in practice, there are several reasons you may want to.

- You have a refined position for the GRB. Mask weighting is only valid for one position on the sky, so if you have a new position, you need to re-calculate the weights.
- You are interested in another source than the GRB.
- To capture improvements in the newer software (see below).
- To make an auxiliary raytracing file.

Checking the event file for the position

You can check the event file to see if it was mask-weighted with the proper position of the GRB.

Run the command:

fkeyprint sw00145675000bevshsp_uf.evt.gz BAT_

You should see the following results (among other entries):

BAT_RA = 320.5402 / [deg] Right ascension of source
BAT_DEC = 77.0751 / [deg] Declination of source

If these lines do not exist, or they contain the incorrect position, then you will need to re-run the mask-weighting task.
5.5. **MASK WEIGHTING (RAY TRACING)**

Checking for old software versions

The SDC has used several versions of the software throughout the mission life-time, and they are not always the best versions. Versions of the software before HEASOFT 6.0.3 contained an error which affects the flux determination of far off-axis bursts. HEASOFT 6.1.2 introduced some new filtering techniques which should slightly improve sensitivity. You can check which version of the software was used with the following command:

```
fkeyprint sw00145675000bevshsp_uf.evt.gz BATCREAT
```

You should see the following result:

```
# EXTENSION: 1
BATCREAT= 'batmaskwtevt 1.11' / BAT Program that modified this FITS file
```

If you see a version earlier than 1.11, then you should definitely re-run the mask-weighting task. If you see a version earlier than 1.16, then you might consider re-running the mask weighting task, which could gain a slight improvement in sensitivity.

**What is an auxiliary raytracing file?**

An auxiliary raytracing file contains information about the position of the GRB and the mask geometry as a function of time within the BAT field of view. This information is needed to make an accurate spectral response before, during and after the spacecraft is slewing to the burst. As of late 2005, the SDC has started to make an auxiliary raytracing file for each observation. Before that time, the file will not be present, and you will need to re-run the mask-weighting tool in order to produce one.

### 5.5.3 Prerequisites

We will practice the mask weighting on GRB 050713A (obsid 00145675 000).

Here is what is needed:

- Known position of the GRB (RA and Dec)
- Attitude file (obsid/aux/swNNNNNNNNNNNsat.fits.gz)
- BAT event data (in obsid/bat/event/swNNNNNNNNNNNbevshsp_uf.evt.gz);
- BAT quality map (in obsid/bat/hk/swNNNNNNNNNNNbcbdq.hk.gz)

In this case, let us enter the revised position as reported by Castro-Tirado (GCN 3584): RA=320.5397083, Dec=+77.074861, and apply the mask weighting operation to the original event list.

### 5.5.4 Initial Steps

Change directories to the location of the event data.
cd 00145675000/bat/event

If the event file is compressed, unzip it. The mask-weighting task modifies the event file in place, and gzipped files cannot be modified.

`gunzip sw*.evt.gz`

### 5.5.5 Compute Mask Weights for BAT Events

The mask-weighting task for events is called `batmaskwtevt`. Here is an example run,

```
batmaskwtevt infile=sw00145675000bevshsp_uf.evt \\
    attitude=../../auxil/sw00145675000sat.fits.gz \\
    ra=320.5397083 dec=+77.074861 \\
    detmask=../hk/sw00145675000bcbdq.hk.gz \\
    rebalance=YES corrections=default \\
    auxfile=/local/data/gcn5b/craigm/sw00145675000bevtr.fits clobber=YES
```

- `infile` gives the event file name
- `attitude` gives the attitude file name
- `ra` and `dec` give the revised sky position of the GRB (see above)
- `detmask` is a quality filter which excludes disabled and noisy detectors; this is important so than an accurate accounting of the number of illuminated detectors can be computed
- `rebalance=YES` means that the resulting light curves will be background-subtracted
- `corrections=default` applies the default corrections to the mask weights
- `auxfile` gives the output auxiliary raytracing file, to be used for spectral analysis

This analysis may take some time (several minutes), as each event needs to be ray-traced. The speed of the task depends on the number of slews that occur during the observation (i.e. more slews require more intensive calculations).

### 5.6 Light Curve from Event Data

This section describes how to generate a light curve from BAT event data. Later sub-sections describe how to make a light curve with custom time binning.

Figure 5.2 shows schematically how light curves are created using the batbinevt task. The same task is used to create spectra and detector plane images and detector plane histograms.

#### 5.6.1 Prerequisites

- BAT event data (in obsid/bat/event/swNNNNNNNNNNNbevshsp_uf.evt.gz);
- BAT quality map (in obsid/bat/hk/swNNNNNNNNNNNbcbdq.hk.gz).
5.6. **LIGHT CURVE FROM EVENT DATA**

---

**Figure 5.2:** Schematic diagram of binning event data using the `batbinevt` task. It can be used to create light curves, as well as spectra and images.

We will use the original event list of GRB 050713A as an example (obsid 00145675 000). The event data should have been ray-traced by the SDC using the position of the GRB. Energy calibrations should also have been applied. See the Mask-Weighting and Energy Correction threads for more information.

### 5.6.2 Change into Event Directory

Change directories to the location of the event data.

```
cd 00145675000/bat/event
```

### 5.6.3 Make the light curve

BAT light curves are made with the `batbinevt` task. Here is an example:

```
batbinevt infile=sw00145675000bevshsp_uf.evt.gz outfile=onesec.lc outtype=LC \ 
timedel=1.0 timebinalg=u energybins=15-150 \ 
detmask=../hk/sw00145675000bcbdq.hk.gz clobber=YES
```

- the `infile` is the input events file
- the `outfile` is the output light curve
- the `outtype=LC` means make a “light curve”
- the `timedel=1.0` means make 1 second time bins
- the `timebinalg=u` means make them uniform bins (instead of variable sized bins)
- the `energybins=15-150` means make a 1-channel light curve from 15-150 keV
- the `detmask` is required to filter out noisy detectors
5.6.4  Plot the Light Curve

You can plot the light curve with ‘fplot’ or ‘fv’. Example:

```plaintext
fplot onsec.lc offset=yes
   X column = TIME
   Y column = RATE[ERROR]
```

you will have to adjust the X range to see the burst since there are large amounts of extraneous data included by the SDC. For example: `r x -100 3000` zooms on the first 3000 seconds.

If the trigger time (`TRIGTIME`) keyword is present in the light curve file, then it is convenient to plot the light curve after subtracting this time as an offset. It is easiest to do this with the ‘fv’ task. Run ‘fv’ on the light curve file and select ‘Plot’. For the X value, type `TIME - TRIGTIME` and the trigger time will be placed at the origin of the plot.

![Figure 5.3: Result of plotting the one second light curve of GRB 050713A in fv.](image)

5.6.5  Changing the Energy Binning

You can choose different energy bins using the ‘energybins’ parameter to the task. Give any energy range `ELOW` to `EHIGH` as “ELOW-EHIGH”. Be sure to use an energy range appropriate to the BAT (within 14-200 keV).
An easy way to make light-curves in multiple bands is to re-run the task with different energy ranges. For example, if you want to make light curves in the 15-50 and 50-150 keV bands, you can run the task twice:

```
batbinevt ... energybins=15-50 ...
batbinevt ... energybins=50-150 ...
```

This will produce two light curve files, each with the different energy ranges.

Another way to make multi-channel light curves is to specify both energybins in one command. In the example above of two bands (15-50 and 50-150 keV), the command would be:

```
batbinevt ... energybins=15-50,50-150 ...
```

This will produce one light curve file with a vector RATE column. You can plot these within `fv` using `RATE[1]` and `RATE[2]` as the Y column names.

### 5.6.6 Changing the time binning

The task `batbinevt` has several options for time binning of light curves.

- **uniform time binning** (timebinalg='u') - you choose the time bin size with the timedel parameter, or set it to zero to sum all the data into one bin;
- **constant signal to noise** (timebinalg='s') - the time bin size is variable and determined by the signal to noise ratio of the data (using the snrthresh parameter);
- **user-specified binning** (timebinalg='g') - you explicitly set the time bin edges using the gtifile parameter.

The user-specified binning gives you the freedom to choose precisely the time bin edges that you wish (expressed in mission elapsed time). You specify these bin edges with a good time interval file (GTI file). The task `battblocks` can produce a set of “interesting” time bins automatically using the Bayesian block algorithm. You can also construct a GTI file by hand (see below).

### 5.6.7 Optional: Making a GTI file

This example shows how to make a GTI file with user-specified time bins. We will assume the time bins of interest are already known relative to the trigger time. The goal will be to make a GTI file with absolute mission elapsed times.

For example, enter the time bin edges in the text file `gti.txt`. These are in seconds relative to the trigger time. Of course you can choose any relative times that you wish.

```
# START STOP
-500  -400
-400  -300
-300  -250
-250  -200
```
Describe the good time interval column names in the file gti.col.

```
START 1D s
STOP 1D s
```

Enter required keywords into the file gti.head

```
MJDREFI = 51910 / Swift reference epoch: days
MJDREFF = 0.00074287037 / Swift reference epoch: fractional days
TIMEZERO = 0.0 / Time offset value
TRIGTIME = 142921742.016 / Trigger time: ** Change for your burst! **
```

And create the relative time good time interval.

```
ftcreate gti.col gti.txt offset.gti headfile=gti.head extname=GTI
```

Finally, add the trigger mission elapsed time (TRIGTIME=142921742.016 for this particular GRB) to make an absolute-time good time interval file.

```
ftcopy offset.gti'[1][col START=START+TRIGTIME;STOP=STOP+TRIGTIME]' absolute.gti
```

You can use this good time interval file to extract the light curve values in your desired time bin ranges:

```
batbinevt infile=sw00145675000bevshsp_uf.evt.gz outfile=mybins.lc outtype=LC \
    timedel=1.0 timebinalg=g energybins=15-150 gtifile=absolute.gti \
    detmask=../hk/sw00145675000bcbdq.hk.gz clobber=YES
```
5.7 Spectrum and Response Matrix

This section describes how to make a spectrum from BAT event data using the task ‘batbinevt’. After creating a raw spectrum, certain corrections must be applied, and then a response matrix can be created.

Figure 5.2 shows that spectra are created with the ‘batbinevt’ task. Figure 5.4 shows the steps that must be taken after the spectrum has been created.

5.7.1 Introduction

Making a spectrum involves selecting the time of interest, and the desired energy binning.

Let’s assume we are interested in the burst GRB 050713A, in particular a flare from time 142921843 (MET) to 142921868 (MET).

5.7.2 Prerequisites

- BAT event data (in obsid/bat/event/swNNNNNNNNNNNbevshsp uf.evt.gz);
- BAT quality map (in obsid/bat/hk/swNNNNNNNNNNNbcbdq.hk.gz).
- BAT event auxiliary raytracing file (in obsid/bat/event/swNNNNNNNNNNNbevtr.fits.gz);

We will use the original event list of GRB 050713A as an example (obsid 00145675 000). The event data should have been ray-traced by the SDC using the position of the GRB. Energy calibrations should also have been applied. See the Mask-Weighting and Energy Correction threads for more information.

If the auxiliary raytracing file is not available (it was not archived by the SDC until late 2005), then it must be recreated using the Mask-Weighting thread.
5.7.3 Change into Event Directory

Change directories to the location of the event data.

\texttt{cd 00145675000/bat/event}

5.7.4 Make the Spectrum

BAT spectra are made with the `batbinevt` task. Here is an example for the flare discussed above:

\begin{verbatim}
batbinevt infile=sw00145675000bevshsp_uf.evt.gz outfile=flare.pha outtype=PHA \
timedel=0.0 timebinalg=u tstart=142921843 tstop=142921868 \
energybins=CALDB:80 outunits=RATE \
detmask=../hk/sw00145675000bcbdq.hk.gz clobber=YES
\end{verbatim}

- the `outtype=PHA` means make a "spectrum" (PHA = pulse height analyzer)
- the `timedel=0.00` means accumulate all the data in the specified time range into one spectrum
- the `timebinalg=u` means make them uniform bins (really "don’t care" in this case)
- the `tstart` and `tstop` give the start and stop time of the spectrum as discussed above
- the `energybins=CALDB:80` means make a default 80-channel spectrum
- the `outunits=RATE` makes a rate spectrum, which is usually required since BAT spectra have fractional counts, which XSPEC does not accept
- the `detmask` is required to filter out noisy detectors

5.7.5 Apply Corrections to the Spectrum

There are several corrections that need to be applied to a spectrum before it can be fit by XSPEC. The first correction involves inserting the proper ray-tracing keywords into the file.

Several geometric parameters are required to make an accurate response matrix, such as the position of the burst in instrument coordinates. Because the spacecraft is often slewing during the burst, it is not possible to know for sure what those geometric parameters should be until the desired spectrum has been created. The `batupdatephakw` task does this for you, using the auxiliary ray-tracing file described above. If the auxiliary raytracing file does not exist in your observation, then you will have to create it using the Mask-Weighting thread. If you do not apply this correction, then fluxes may be in error by up to 50%, and power law indices may be off by up to 0.5.

\texttt{batupdatephakw flare.pha sw00145675000bevtr.fits.gz}

Next, you should apply the BAT-recommended systematic error to your spectrum. The `batphaysyserr` task is used to do this, and the error vector is stored in the calibration database.

\texttt{batphaysyserr flare.pha CALDB}
5.7.6 Generate a Response Matrix

A response matrix should be created for each spectrum where the GRB is at a different position in the BAT field of view. For example, one response matrix should be made for data taken before any slew, and one after any slew. Any spectra made during the slew should have their own response matrix.

batdrmgen flare.pha flare.rsp NONE

5.7.7 Important: Spectrum of a Source During a Slew

It is often the case that a gamma-ray burst source is moving in the BAT field of view. This is because the spacecraft often slews to the position of the GRB before the burst emission is has ended. The BAT spectral response changes with position, so you should not use one response matrix for the entire burst. For a source with a power law spectrum, a position error of 25 degrees introduces an error of about 0.1 in the photon index, and about 10% in total flux. The spectral distortion effect will be more pronounced if the slew is a longer distance, or if the GRB has strong emission peaks separated by a large time and/or distance. The distortion will be less if the GRB emission occurs mostly during a pointing, and of course there will be no distortion if the burst emission is completely confined to a pointing.

You should break long bursts which overlap slews into smaller segments corresponding to whether the spacecraft is moving or not. Generally speaking, you should make a response matrix whenever the position of the source changes by more than 15 degrees (or smaller increments if you wish). One straightforward way to do this is to use the position columns in the auxiliary ray tracing file. Look for the columns named IMX and IMY, which contain the source position in the BAT field of view, relative to the spacecraft pointing direction. The values are tangent plane coordinates, so tan(15 deg) ~ 0.27. Thus, you should break a spectrum into segments where the source moves no more than 0.25 in the IMX or IMY coordinate.

The SDC should make “pre-slew” and “post-slew” response matrices. These generally should correspond to the pointings during the BAT trigger, and the following pointing after the slew, respectively. These matrices will be useful if the burst primarily overlaps either the pre-slew or post-slew pointings.

5.7.8 Analyze Spectrum in XSPEC

You can now load this spectrum within the spectral fitting program XSPEC. Please be aware that you should ignore data below 14 keV and above 195 keV. Since the BAT mask becomes transparent around 150 keV, the BAT team recommends fitting spectra in the 15-150 keV range.

Common Spectral Models

The following models are commonly used in fitting of BAT spectra.

**powerlaw/pegpwrlw** The simplest model. Powerlaw is normalized by its differential spectrum, and pegpwrlw is normalized by total flux. NOTE: since the powerlaw model is normalized at 1 keV, it can lead to spurious statistical correlations between parameters in the BAT 15-150 keV band; pegpwrlw is preferred.
**cutoffpl** Power law with an exponential cut-off at high energies. As with the simple power law model, cutoffpl is normalized at 1 keV.

**grbm** “Band” gamma-ray burst model. Typically BAT spectra do not cover a large enough energy range to uniquely determine all Band model parameters. NOTE: “tem” parameter in “grbm” model is a cutoff energy instead of Epeak. Epeak is calculated with the following way; \( E_{\text{peak}} = (2+\alpha) \times \text{tem} \).

The BAT team uses the following custom cut-off power law model in XSPEC:

\[
m\text{define cutep50 } (E/50.0)^{(-a)} \times \exp(-E(2.0-a)/b)
\]

where the parameters are:

- **a** photon index
- **b** Epeak [keV]

the advantages to this model are that the photon index is normalized at 50 keV, in the middle of the BAT energy band, and the Epeak parameter is the well known quantity associated with gamma-ray bursts, which can also be derived from the Band model (see above).

In comparison to the standard cut-off power law model (cutoffpl), the photon index is identical, and the relation between Epeak and HighECut is,

\[
E_{\text{peak}} = (2-\text{PhoIndex}) \times \text{HighECut}
\]

where HighECut is the exponential cut-off energy of XSPEC.

Typically, the BAT team prefers the custom cut-off power law fit if the chi-squared value improves by more than 6 over the pure power law fit.

### 5.7.9 Choosing Different Energy Binning

You are free to choose different energy binning than the default 80-bin spectrum (the default is selected with `energybins=CALDB:80`).

The batbinevt task accepts either a file with an EBOUNDS extension containing bin edges, or a comma-separate list of energy bin edges.

### 5.7.10 Do Not Use GRPPHA

Do not group spectral bins of BAT spectra with programs like GRPPHA. The purpose of doing so for photon counting instruments is to collect enough photons in each bin so that gaussian statistics apply. However, BAT is not a photon counting instrument. By virtue of the mask weighting technique, BAT spectra already have gaussian statistics. Thus, there is no need to performing grouping. Spectral fits with non-grouped spectra should be correct, even if the count rate in a given bin is consistent with zero.

For low-statistics data, it may appropriate to use XSPEC’s `setplot group` command to make a more pleasing plot, but be aware that this command makes some statistical assumptions about “counts” in a spectrum which are not necessarily applicable to BAT spectra. It is more correct to make a new spectrum with coarser energy binning.
5.8. Making BAT Sky Images

5.8.1 Introduction

While the BAT does not have focussing optics, it is possible to create sky images using the task `batfftimage`. This task combines a detector plane image and information about the coded aperture pattern to reconstruct the sky flux at each point in the field of view. Sky images have the same flux units as all other mask weighted counts values.

The steps of this procedure are: (1) make a source detector plane image using `batbinevt`; (2) optionally make a background detector plane image; and finally (3) to make a sky image using `batfftimage`.

5.8.2 Prerequisites

- BAT event data
  - for example, (in obsid/bat/event/swNNNNNNNNNNbevshsp_ne.fvt.gz);
  - BAT quality map,
– a basic map can be found in obsid/bat/hk/swNNNNNNNNbcbdq.hk.gz;
– one can be created by using the “Creating a Quality Map” procedure;

Swift attitude information (in obsid/auxil/swNNNNNNNNNsat.hk.gz).

For example, let’s assume we are interested in the burst GRB 050713A, in particular to produce an image of the “flare” from time 142921843 (MET) to 142921868 (MET).

5.8.3 Change into Event Directory

Change directories to the location of the event data.

cd 00145675000/bat/event

5.8.4 Make a Detector Plane Image

A Detector Plane Image (DPI) is a map of the counts (or count rate) in each detector as it is laid out geographically in the detector plane. This map is the input to the deconvolution software.

Issue the following command,

```
batbinevt infile=sw00145675000bevshsp_uf.evt.gz outfile=flare.dpi outtype=DPI \ timedel=0.0 timebinalg=u tstart=142921843 tstop=142921868 \ energybins=14-150 outunits=RATE weighted=NO ecol=ENERGY \ detmask=../hk/sw00145675000bcbdq.hk.gz clobber=YES
```

where

- outtype=DPI means to make a DPI
- timedel=0.0 means accumulate all the data into one single DPI
- timebinalg=u means make them uniform time bins (really “don’t care” in this case)
- the tstart and tstop give the start and stop time of the spectrum as discussed above
- the energybins=14-150 means make a single image in the 14-150 keV range
- the ecol=ENERGY means use the ENERGY column for energy selections
- the outunits=RATE makes a rate image
- the weighted=NO means do not perform background subtraction
- the detmask is required to filter out noisy detectors

The result should be a single detector plane image. It is also possible to select multiple time ranges or multiple energy bands. batfftimage will automatically process these to produce sky images in multiple slices.
5.8. MAKING BAT SKY IMAGES

5.8.5 Making a Background Detector Plane Image

Making a background detector plane image is optional, but it will increase the sensitivity of the image when bright sources are present. If possible, the background interval should be made from data with similar background levels and for a longer duration than the source image.

Use the same procedure as above to create a DPI file during a time range when you know there is no source emission. Let us call that file “bkg.dpi”.

NOTE: if the background or bright sources are known to have varied significantly between the time of the source and background DPIs, there may be little or no benefit to making a background file.

Advanced usage: if bright sources are known to be present in the image, it is possible to use the batclean task to create a background map which removes many systematic artifacts. This is usually not necessary for images with exposures less than 100 seconds.

5.8.6 Making a Sky Image

A sky image is created from a DPI using the ‘batfftimage’ task. Issue the following command,

```
batfftimage infile=flare.dpi outfile=flare.img attitude=../auxil/sw00145675000sat.hk \\ bkgfile=bkg.dpi detmask=../hk/sw00145675000bcbdq.hk.gz clobber=YES
```

where

- `infile` is the DPI file name
- `bkgfile` is the background DPI file name (‘NONE’ can also be used if no background file is available)
- `outfile` is the output sky image file name
- `attitude` is the spacecraft attitude file name
- `detmask` is the detector quality map

The result is a BAT sky image. If valid spacecraft attitude data was used, then the sky image should automatically have celestial coordinates (WCS keywords) attached.

You may use your favorite FITS image viewer to examine the image (for example FV or DS9). Figure 5.6 shows the appearance of a typical bright point source in a BAT sky image.

5.8.7 Caveats

The spacecraft attitude may have changed during the accumulation interval of your DPI. You must carefully filter out spacecraft slew data and use only steady pointed attitude information, or the celestial coordinate system will be erroneous.
5.8.8 Making a Partial Coding Image

As described elsewhere, the BAT has a large field of view with a large variation in sensitivity. The partial coding map describes the fraction of the detector array which is illuminated by each point on the sky. To make a partial coding map, issue the following command,

```
batfftimage infile=flare.dpi outfile=flare.pcodeimg attitude=../auxil/sw00145675000sat.hk \
bkgfile=bkg.dpi detmask=../hk/sw00145675000bcbdq.hk.gz clobber=YES pcodemap=YES
```

Note that the only change from the previous call is to use the parameter “pcodemap=YES”, and to change the output file name.

Figure 5.6: Bright point source in a BAT sky image.
Chapter 6

BAT Analysis Issues

6.1 Introduction

This chapter contains various science analysis issues that may come up during analysis. In some cases, the issues represent “bugs” in the software, which are usually corrected in later software releases. In other cases, the issue may reflect a “gotcha” or other subtle problem that needs special attention.

The chapter is organized into two sections. The first section contains “current” issues that may affect analysis of data in the archive today, using recently available software. Users should definitely consult this first section for any issues that might affect their data analysis.

The second section contains older issues that are not likely to affect present-day analysis. This involves software bugs that have been fixed more than a year ago, Swift Data Center problems that have been fixed, and other issues that have since been resolved.

Each issue contains vital information, such as the versions of the software and which tasks it affects, the status of the issue, and a more complete narrative about the problem.

6.2 Current Issues

6.2.1 Analysis: filtering out non-gamma-ray events may improve sensitivity

<table>
<thead>
<tr>
<th>Task:</th>
<th>batmaskwtevt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>1.15 and earlier</td>
</tr>
<tr>
<td>What builds:</td>
<td>HEASoft 6.1.1 and earlier</td>
</tr>
<tr>
<td>Problem:</td>
<td>Slight reduction in sensitivity due to non gamma-ray events</td>
</tr>
<tr>
<td>Status:</td>
<td>Closed (HEASoft 6.1.2)</td>
</tr>
<tr>
<td>Updated:</td>
<td>08 Jan 2007</td>
</tr>
</tbody>
</table>

BAT count rates are typically background dominated. Any reduction in the background can improve the sensitivity to a source signal.

BAT event data includes many kinds of events, not just gamma-ray events. Of course many events are due to cosmic diffuse and particle backgrounds, but they cannot be easily identified as such on an event-by-event basis. On the other hand, there are other events which are known to be non-gamma-ray events based on their electronic signature. These events are flagged by the electronics and the flight software. In previous versions of the ground software, all events, including the flagged non-gamma-ray events, were assigned weights by batmaskwtevt. However, these events
should be excluded, which will marginally reduce the background and improve sensitivity.

The new version of batmaskwtevt, version 1.16 (released with HEASoft 6.1.2), excludes non-gamma-ray events from further processing. The BAT team recommends that users re-run the batmaskwtevt with version 1.16 or better to incorporate these changes. Further information can be found in the recipe for running mask weighting for a given event file.

It is worth mentioning that the improvement in sensitivity may be so marginal that in some cases the significance of any one observation may actually decrease very slightly. On the other hand, the average of a large ensemble of observations should improve slightly.

### 6.2.2 Analysis: Passive materials distort the off-axis counts/rates

<table>
<thead>
<tr>
<th>Task:</th>
<th>All flux extraction (batfftimage/batmaskwtevt/batmaskwtimg/batbinevt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>All versions</td>
</tr>
<tr>
<td>What builds:</td>
<td>All builds</td>
</tr>
<tr>
<td>Problem:</td>
<td>Passive materials introduce errors in mask-weighted counts/rates</td>
</tr>
<tr>
<td>Status:</td>
<td>Corrective Procedure (HEASoft 6.0.3)</td>
</tr>
<tr>
<td>Updated:</td>
<td>15 Oct 2005</td>
</tr>
</tbody>
</table>

The BAT image system works by forming shadow patterns cast by the mask onto the detector array. The mask is made from lead tiles. However, the mask support structure also contains significant absorbing materials. The edge of the support structure is a particular problem because (a) extra absorbing materials (e.g. epoxy) were applied, and (b) this material rises up above the plane of the mask. The net result is that for sensitive imaging and spectroscopy of off-axis sources, the full mask aperture cannot be used.

The BAT team has provided a new set of aperture files in CALDB, tied to the HEASoft 6.0.3 release. These apertures have been reduced in size to appropriately block out the shadows of most of the absorbing material. However, this does reduce solid angle sky coverage by 5-10%.

The aperture files are now divided into two classes:

- **FLUX** - reduced aperture for sensitive flux measurements (DEFAULT)
- **DETECTION** - full aperture for largest solid angle and most illuminated detectors, but reduced flux accuracy

We should note that the differences between these apertures are small. The “old” DETECTION apertures can still be used if desired, but the FLUX apertures are likely to be what most users want, and are thus the default. The aperture type is selectable in the raytracing tasks using aperture=\texttt{CALDB:FLUX} or aperture=\texttt{CALDB:DETECTION}.

### 6.2.3 Analysis: Users must apply systematic error vector for spectral fitting

<table>
<thead>
<tr>
<th>Task:</th>
<th>Spectral Analysis (batdrmgen and XSPEC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>All versions</td>
</tr>
<tr>
<td>What builds:</td>
<td>All builds</td>
</tr>
<tr>
<td>Problem:</td>
<td>Response matrix contains significant systematic errors</td>
</tr>
<tr>
<td>Status:</td>
<td>Corrective Procedure (HEASoft 6.0.3)</td>
</tr>
<tr>
<td>Updated:</td>
<td>15 Oct 2005</td>
</tr>
</tbody>
</table>

The BAT detector response matrix has been significantly improved in the HEASoft 6.0.3 release. However, significant systematic residuals remain. Users should consult the BAT Calibration Status.
section for more discussion of this. To account for the residuals, the BAT team has prepared a systematic error vector which should be attached to all spectra before analysis in XSPEC.

The BAT team has released a task called **batphasyserr** which can apply the systematic error vector to BAT spectra. It works for both type I and type II spectral files, and works for any type of energy binning.

Most commonly, the task will be invoked in the following way:

```
batphasyserr spectrum.pha CALDB
```

where `spectrum.pha` is the spectrum to be modified (it is modified in place). After using this task, `spectrum.pha` should have a new column, **SYS_ERR**, which XSPEC uses during spectral fitting.

Normally, users should use the BAT-team estimated systematic error vector provided with calibration database (CALDB). The most up to date version of this file can be found by querying the CALDB with this command:

```
quzcif Swift BAT -- SYS_ERR yyyy-mm-dd 00:00:00 -
```

Where `yyyy-mm-dd` is the date of the observation (**now** can also be used). As of this writing, the systematic error vector is named `swbsyserr20030101v002.fits`.

NOTE: Because the systematic errors are significant, it is likely that for bright sources the **reduced chi-squared value will be less than 1.0**. When this occurs, at least a portion of the spectrum is systematics-dominated instead of statistics-dominated. The formal parameter errors of the XSPEC fit will be appropriately larger than the corresponding statistical errors.

Updated (15 Oct 2005) - to discuss the new batphasyserr task

6.2.4 batfftimage and batmaskwtimg: potentially incorrect derived attitude

<table>
<thead>
<tr>
<th>Task:</th>
<th>Imaging tasks (batfftimage and batmaskwtimg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>All versions</td>
</tr>
<tr>
<td>What Builds:</td>
<td>All builds</td>
</tr>
<tr>
<td>Problem:</td>
<td>Attitude may be incorrect for observation with many snapshots</td>
</tr>
<tr>
<td>Status:</td>
<td>Corrective Procedure</td>
</tr>
<tr>
<td>Updated:</td>
<td>15 Oct 2005</td>
</tr>
</tbody>
</table>

Two imaging tasks assume that the spacecraft attitude is fixed during an observation: batfftimage (to make sky images) and batmaskwtimg (to make mask weighting maps for flux extraction). Both tasks take the time at the midpoint of the observation.

If there are gaps in the observation, i.e. multiple snapshots, then it is possible, even likely, that the midpoint time will fall within a gap. When this happens, the attitude may be erroneously interpolated.

The recommended solution is to use the ‘aspect’ tool to generate a revised attitude file. As of HEASoft 6.0, ‘aspect’ can create a new attitude file based on the median pointing direction during the observation good times only. You will also need to supply a good time interval extension to ‘aspect’, which should be available in the detector or sky images.

The ‘aspect’ command is:

```
aspect swNNNNNNNNNNNsat.fits none detimage.dpi'[STDGTI]' newattfile=detimage.att
```
where \texttt{swNNNNNNNNNNNNsat.fits} is the spacecraft attitude file, \texttt{detimage.dpi} is the detector plane image (which should contain a GTI extension named STDGTI), and \texttt{detimage.att} is the new attitude file valid for the \texttt{detimage.dpi} only.

In the future the BAT team will investigate how feasible it is to incorporate ‘aspect’-like functionality into the BAT imaging tools.

Updated (19 May 2005) - to recommend only the ‘aspect’ solution.
Updated (15 Oct 2005) - both batfftimage and batmaskwtimg now analyze the midpoint of the time interval.

### 6.2.5 Analysis: Earth and Moon Occultation

<table>
<thead>
<tr>
<th>Task:</th>
<th>Flux extraction tasks (batmaskwtevt,batmaskwtimg,batfftimage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>All versions</td>
</tr>
<tr>
<td>What Builds:</td>
<td>All builds</td>
</tr>
<tr>
<td>Problem:</td>
<td>Earth and moon may block parts of the BAT field of view</td>
</tr>
<tr>
<td>Status:</td>
<td>Corrective procedure</td>
</tr>
<tr>
<td>Updated:</td>
<td>15 Oct 2005</td>
</tr>
</tbody>
</table>

The BAT field of view is large, approximately 120 x 60 degrees fully coded. The current spacecraft constraint excludes the sun with a 45 deg constraint cone, and the earth limb and moon with 30 degree constraints each.

Even so, the Earth and Moon may enter the BAT field of view. This will most commonly occur at edges of the “long” BAT axis (i.e. large IMX in the image plane). The effect will be to occult the flux of sources in that part of the sky. Since the Moon and (primarily) the Earth move as a function of time, the blockage may have the effect of reducing, but not totally eliminating the source on-time.

Example: in a 2000 second image, Sco X-1 might be blocked during the final 300 seconds.

The BAT team has released two tools to aid in correcting for occultation.

- **batoccultmap** produces a fractional exposure map for a full sky image. Users provide the sky image and the prefilter attitude file ("SAO" file), and batoccultmap computes the fractional sky exposure in each pixel. This task should be used when users are interested many sources at one time.

  NOTE: we suggest to only correct for earth occultation. Correcting for sun/moon occultation typically does not help, since it changes the image statistics in a non-smooth way.

- **batoccultgti** produces a good time interval file (GTI file) for a known source. Good times are selected based on being unocculted and/or in a certain position in the field of view. This task should be used when users are interested in a particular source (i.e. not imaging), and should be used to filter the input data by time before further analysis.

Updated (15 Oct 2005) - to discuss the new occultation tasks.

### 6.2.6 batmaskwtevt: Slewing Raytracing Keywords Potentially Incorrect

<table>
<thead>
<tr>
<th>Task:</th>
<th>batmaskwtevt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>ALL</td>
</tr>
<tr>
<td>What Builds:</td>
<td>All builds</td>
</tr>
<tr>
<td>Problem:</td>
<td>Improper ray-tracing keywords are written to the event file during slews</td>
</tr>
<tr>
<td>Status:</td>
<td>Corrective Procedure</td>
</tr>
</tbody>
</table>
6.3. OLD ISSUES

Discussion: During slews, the source of interest (such as a GRB) moves in the BAT field of view. Users can still make light curves and spectra from moving sources by using the batmaskwtevt task. This task computes a ray-trace for each event, and dynamically updates the ray-trace as the spacecraft slews.

The response matrix generator, which is downstream from the event processing, relies on several keywords to construct a proper response matrix. These special keywords are written by batmaskwtevt. However, batmaskwtevt can only write a single set of keywords to cover the whole slew. It uses the final values, corresponding to the end of the slew. This is usually the wrong thing to do, since GRB transients are typically before the slew, not after.

Nominally, light curves and spectra are corrected crudely for off-axis effects by the mask weighting procedure (and the default correction settings). HOWEVER, this is not the whole story. The shape of the response is also a function of angle, and this shape change will alter the spectrum in subtle ways. As a source with the same intrinsic spectrum moves farther off-axis, the measured spectral index will appear to become flatter and flatter, unless the ray-tracing keywords are properly set. This also changes the measured flux as well, of course.

The solution is that users must rewrite the keywords for the time of interest. The software task batupdatephakw, released in Swift 2.0, makes it more easy to do this.

The steps to use this task are:

1. Users should run batmaskwtevt with the auxfile= parameter. This produces an auxiliary file that contains the raytracing values as columns, and as a function of time.

2. For each spectral file spectrum.pha, run this command:

```
batupdatephakw spectrum.pha raytrace.aux
```

where raytrace.aux is the auxiliary file created in step 1. The script will determine the center time of each spectrum in the input file, and transfer the raytracing values for the nearest time from the auxiliary file to the spectrum. The spectrum is modified in place.

The required column/keywords are: BAT_XOBJ, BAT_YOBJ, BAT_ZOBJ, PCODEFR, NGOOD-PIX, and MSKWTSQF.

6.3 Old Issues

6.3.1 SDC: BAT data from SDC pipeline has incorrect DATE-OBS/END keywords

<table>
<thead>
<tr>
<th>Task:</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>N/A</td>
</tr>
<tr>
<td>What Builds:</td>
<td>All (including Swift 2.0)</td>
</tr>
<tr>
<td>Problem:</td>
<td>Erroneous DATE-OBS/END keywords in SDC data before May 4 2005</td>
</tr>
<tr>
<td>Status:</td>
<td>Fixed</td>
</tr>
<tr>
<td>Updated:</td>
<td>08 Jan 2007</td>
</tr>
</tbody>
</table>

**Update:** The SDC has now reprocessed all observations which were affected by this problem. Users should no longer have to apply this workaround. With Swift 2.1, more descriptive error messages are produced.

The BAT software tasks depend on the proper DATE-OBS and DATE-END keywords in order to access the CALDB system. If these keywords are incorrect, then the tasks will attempt to find the wrong files in CALDB.
Before the date of XXX the Swift Data Center (SDC) produced some files with the date 2001-01-01, which causes this CALDB problem. You can check for the problem by running most tasks with chatter=5. If you see “strtdate=2001-01-01” before the task quits, then you have this problem. Also, you can run `fkeyprint filename DATE-OBS` and check for ‘2001-01-01’.

A workaround is to correct the DATE-OBS and DATE-END keywords. You can do this with the following commands:

```
fthedit myfile.fits DATE-OBS add value="YYYY-MM-DD"
fthedit myfile.fits DATE-END add value="YYYY-MM-DD"
```

where YYYY-MM-DD is the proper date of the observation.

This bug was fixed on May 4, 2005 (pipeline processing version 2.1.6). However, the workaround will still be needed for old data before this date, until the SDC reprocesses it.

### 6.3.2 batoccultmap: incorrect exposure correction

<table>
<thead>
<tr>
<th>Task:</th>
<th>batoccultmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>v4.1 and earlier</td>
</tr>
<tr>
<td>What builds:</td>
<td>HEADAS 6.0.5 and earlier</td>
</tr>
<tr>
<td>Problem:</td>
<td>Incorrect exposure over entire image</td>
</tr>
<tr>
<td>Status:</td>
<td>Fixed in HEASoft 6.1</td>
</tr>
<tr>
<td>Updated:</td>
<td>04 Aug 2006</td>
</tr>
</tbody>
</table>

batoccultmap is used to compute the exposure correction due to occultation by the earth or other celestial bodies. Typically earth occultation only affects the edges of the image. However, a bug was found in this task which incorrectly computes the exposure correction over the entire image.

The error does not always occur. The visible portion of the earth’s limb must be entirely in the southern celestial hemisphere, the earth must be at the very edge of the image, and the CONTOUR algorithm must be used. Even then, the behavior is not always predictable.

The result is that the full image has a depressed fractional exposure level. Typically the loss is a few percent. Based on analysis of BAT data from 2005, 93% of results will be correct, 98% will be correct to within 2%, and 99.5% will be correct to within 5%. For comparison, relative BAT fluxes are calibrated to within a few percent at best.

Workarounds:
- Use the IMAGE algorithm of batoccultmap instead of the CONTOUR algorithm
- Take the exposure result, and divide the full image by the central pixel value

This bug was been fixed in HEASoft 6.1 (Swift Build 19).

### 6.3.3 batbinevt: light curves have slightly incorrect energy bins

<table>
<thead>
<tr>
<th>Task:</th>
<th>batbinevt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>v1.28 and earlier</td>
</tr>
<tr>
<td>What builds:</td>
<td>HEADAS 6.0.4 and earlier</td>
</tr>
<tr>
<td>Problem:</td>
<td>Incorrect energy bin assignment for light curves</td>
</tr>
<tr>
<td>Status:</td>
<td>Closed (HEADAS 6.0.5)</td>
</tr>
<tr>
<td>Updated:</td>
<td>29 May 2006</td>
</tr>
</tbody>
</table>
The task batbinevt is used to make spectra and light curves from BAT data. A bug in the way that energy bin assignments are made will lead to slightly incorrect reported count rates.

The easiest way to explain it is by example: if you ask for a light curve from 15-25 keV, then you actually get a light curve from 15-25.1 keV (not the extra 0.1 keV). The upper bin edge will always be too large by 0.1 keV.

If you are making a vector light curve (multiple energy bands), or a spectrum, then only the last energy bin is affected.

Thus, this is typically a small effect: spectra are basically unaffected (since the top bin is usually ignored). Light curves made in individual bands are affected. However, for wide energy bins, the light curves will be off by only a small amount (<1% for the 15-25 keV case described above). The only place where it is a strong effect is when you make many individual light curves in narrow energy bands.

This bug has been fixed for HEADAS 6.0.5 (Swift Build 18).

### 6.3.4 Mask tagged light curve systematic flux errors

<table>
<thead>
<tr>
<th>Task:</th>
<th>batmasktaglc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>All (bug is in BAT flight software)</td>
</tr>
<tr>
<td>What Builds:</td>
<td>All</td>
</tr>
<tr>
<td>Problem:</td>
<td>There are systematic flux errors in mask tagged light curves</td>
</tr>
<tr>
<td>Status:</td>
<td>Closed (HEASoft 6.0.3)</td>
</tr>
</tbody>
</table>

Discussion: BAT mask tagged light curves are generated on board by the BAT flight software. The on-board process involves generating a mask weight map via ray tracing (similar to the ground task batmaskwtimg). On the ground, the raw light curves require further processing before they are scientifically meaningful.

The “mask tagging” process requires that the source position be known in advance, in instrument coordinates. This transformation requires knowledge of the spacecraft attitude, the instrument-to-spacecraft alignment, and the source celestial coordinates. A bug has been found in the BAT flight software which makes the incorrect transformation. The ray-traced position used for generating the mask weight map is several arcminutes off of the true position.

The point spread function is thus sampled significantly off-peak; this irretrievably reduces the signal to noise of the mask weighted light curve fluxes. Also, the flux itself is underreported compared to its true value.

The task batmasktaglc now has a “fudge” parameter which corrects the light curve values to be comparable to mask weighted light curves derived from event data. The default value for the task reflects the BAT team’s knowledge of the error introduced by the flight software.

Updated (15 Oct 2005) - Closed by HEASoft 6.0.3 release.

### 6.3.5 XSPEC crashes with segmentation fault with BAT spectra

<table>
<thead>
<tr>
<th>Task:</th>
<th>XSPEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>11.3.2</td>
</tr>
<tr>
<td>What Builds:</td>
<td>HEASOFT 6.0 (including Swift 2.0)</td>
</tr>
<tr>
<td>Problem:</td>
<td>XSPEC crashes with Type II spectral data</td>
</tr>
<tr>
<td>Status:</td>
<td>Closed Swift 2.1</td>
</tr>
</tbody>
</table>

This problem has been solved in the Swift 2.1 release.
It was recently discovered that XSPEC version 11.3.2 crashes when reading BAT data with a segmentation fault error. This bug occurs for any “Type II” spectral file, but only on certain platforms (e.g. Linux and Solaris). Version 11.3.2 is the version released in HEASOFT 6.0. XSPEC version 12 is not affected by this bug.

Currently the best solution is to obtain a software patch from [http://heasarc.gsfc.nasa.gov/docs/xanadu/xspec/](http://heasarc.gsfc.nasa.gov/docs/xanadu/xspec/) and recompile XSPEC.

Another workaround is to add the XFLTNNNN keywords where NNNN ranges from 1 to 10. This can be done with the following C-shell commands:

```bash
foreach num (01 02 03 04 05 06 07 08 09 10)
    fthedit myfile.pha XFLT00$num add value=${num}
end
```

### 6.3.6 `baterebin`: Possible extra rows in EBOUNDS extension of output DPH

<table>
<thead>
<tr>
<th>Task:</th>
<th>baterebin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>Previous to 6.0</td>
</tr>
<tr>
<td>What Builds:</td>
<td>Swift 2.0 and earlier</td>
</tr>
<tr>
<td>Problem:</td>
<td>Bug leaving extra rows in EBOUNDS extension in some circumstances</td>
</tr>
<tr>
<td>Status:</td>
<td>Close Swift 2.1</td>
</tr>
<tr>
<td>Updated:</td>
<td>20 Jul 2005</td>
</tr>
</tbody>
</table>

`baterebin` is used to reset the energy scale for survey histograms accumulated on-board. This can improve the survey performance since typically the ground knowledge of the energy scale is better than the flight system provides. `baterebin` can “rebin” the output survey DPH to into a smaller number of bins.

If the number of energy bins in the output DPH is less than the number of rows (time bins) in the input DPH, the EBOUNDS extension of the output DPH will have extra rows with spurious energy bounds (left over from a copy of the input DPH).

`batbinevt` will crash with erroneous energy bounds.

The workaround is to delete the extra rows in the EBOUNDS extension before running `batbinevt` or other processing.

### 6.3.7 `batbinevt`: Incorrect exposure for survey analysis

<table>
<thead>
<tr>
<th>Task:</th>
<th>batbinevt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>1.13 and earlier</td>
</tr>
<tr>
<td>What Builds:</td>
<td>Swift 2.0 and earlier</td>
</tr>
<tr>
<td>Problem:</td>
<td>EXPOSURE keyword can be incorrect for survey analysis</td>
</tr>
<tr>
<td>Status:</td>
<td>Closed Swift 2.1</td>
</tr>
<tr>
<td>Updated:</td>
<td>20 Jul 2005</td>
</tr>
</tbody>
</table>

`batbinevt` can apply a user-specified good time interval (GTI) file, or by specifying tstart/tstop on the command line.

If either of these time selections cuts through the middle of a survey histogram (DPH), then what should happen is that DPH is excluded from analysis. What actually happens is somewhat complicated, and may result in an incorrect EXPOSURE keyword value being computed (the same is true for the GTI extension).

This bug has been fixed for Swift 2.1.
6.3.8 CALDB: Incorrect Spectral Response < 20 keV

<table>
<thead>
<tr>
<th>Task:</th>
<th>CALDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>20030101 version 2 and 20041006 version 1</td>
</tr>
<tr>
<td>What Builds:</td>
<td>Swift 1.3 and earlier</td>
</tr>
<tr>
<td>Problem:</td>
<td>Pre-launch response over-estimates effective area below 20 keV</td>
</tr>
<tr>
<td>Status:</td>
<td>Closed Swift 2.0</td>
</tr>
</tbody>
</table>

The pre-launch BAT spectral response file has significant systematic errors below 20 keV. This are caused primarily by the model of passive absorbing material in the BAT field of view. The errors were 30-50% in the 15-20 keV band.

The response matrix has been improved significantly for builds 12 and 14. Users should retrieve the newest set of CALDB files at the same time as downloading new software (at least Swift 2.0 and the corresponding CALDB release).

Users should be aware that significant systematic errors remain even after these adjustments, so they should apply a systematic error vector to all spectra.

6.3.9 CALDB: Incorrect Alignment Matrix

<table>
<thead>
<tr>
<th>Task:</th>
<th>CALDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>20041013 and earlier</td>
</tr>
<tr>
<td>What Builds:</td>
<td>Swift 1.0 and earlier</td>
</tr>
<tr>
<td>Problem:</td>
<td>Pre-launch alignment matrix leads to incorrect positions and fluxes</td>
</tr>
<tr>
<td>Status:</td>
<td>Fixed in Swift 1.1</td>
</tr>
</tbody>
</table>

The BAT alignment matrix components are composed of two files: the “TELDEF” file which describes the alignment of the BAT boresite with respect to the spacecraft, and the “aperture” file which describes the alignment of the mask with respect to the detector array.

These files were checked into CALDB with their pre-launch best guess values. They were further refined after launch according to the calibration plan, and using serendipitously detected sources with known positions.

Users of the pre-launch matrix will find the following problems:

1. Sources will appear to be at the wrong locations;
2. Mask weighted fluxes for known sources will be lower, or zero,
   - since the known celestial position will translate to the incorrect detector position. This causes the mask weighting function to sample the PSF off-center, and thus reduces the response.

Users with pre-launch files should download a revised set of CALDB files. (20041120 version 1 or later).

6.3.10 batbinevt: Incorrect EXPOSURE Keyword

<table>
<thead>
<tr>
<th>Task:</th>
<th>batbinevt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>1.7 and earlier</td>
</tr>
<tr>
<td>What Builds:</td>
<td>Swift 1.1 and earlier (Build 11 and earlier)</td>
</tr>
<tr>
<td>Problem:</td>
<td>Incorrect EXPOSURE keyword written to spectral files</td>
</tr>
<tr>
<td>Status:</td>
<td>Closed (Swift 1.2)</td>
</tr>
</tbody>
</table>
The batbinevt task makes spectra. It writes “Type II” files which can have multiple spectra per file. It writes an \texttt{EXPOSURE} column to record the exposure for each spectrum.

However, the input event or survey file also has an \texttt{EXPOSURE} keyword, which was not being overwritten. Thus it was possible to have conflicting exposure values. Unfortunately, XPSEC uses the keyword (incorrect) value instead of the column (correct) value.

This bug is fixed in Swift 1.2.

**Workaround.** Remove the \texttt{EXPOSURE} keyword from the spectral file. Since the \texttt{EXPOSURE} column is still present, XSPEC will still work, and should report the correct fluxes and fluences.

Commands to remove the \texttt{EXPOSURE} keyword from myfile.pha:

\begin{verbatim}
 fthedit myfile.pha EXPOSURE delete
\end{verbatim}

### 6.3.11 batcelldetect: Incorrect Error Bars

<table>
<thead>
<tr>
<th>Task</th>
<th>batcelldetect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.09 and earlier</td>
</tr>
<tr>
<td>What Builds</td>
<td>Swift 1.1 and earlier</td>
</tr>
<tr>
<td>Problem</td>
<td>Error bars reported by batcelldetect are too small by a factor of (\sim 4)</td>
</tr>
<tr>
<td>Status</td>
<td>Corrected for Swift 1.2</td>
</tr>
</tbody>
</table>

Discussion: batcelldetect performs a PSF fit to detected sources in an input image. This fit is done with a least-squares routine. Errors are estimated by using the formal statistical error (i.e. covariance matrix). The problem is that the error bars were too small by a factor of \(\sim 4\) compared to the “true” values based on population analysis. This turned out to be because the images were oversampled by a factor of 2 in each dimension, which affects the statistics. batcelldetect attempted to account for the oversampling, but did it incorrectly in versions 1.09 and earlier.

### 6.3.12 batbinevt: Incorrect Handling of float point DPHs

<table>
<thead>
<tr>
<th>Task</th>
<th>batbinevt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.8 and earlier</td>
</tr>
<tr>
<td>What Builds</td>
<td>Swift 1.1 and earlier</td>
</tr>
<tr>
<td>Problem</td>
<td>Floating point DPHs were miscounted</td>
</tr>
<tr>
<td>Status</td>
<td>Corrected for Swift 1.2</td>
</tr>
</tbody>
</table>

Discussion: batbinevt reads detector plane histograms (DPHs). DPHs are commonly stored as integer counts, but can be stored as floating point (fractional) counts. A floating point DPH can be created by downstream processing, such as by baterebin, which reassigns fractional counts to each energy bin.

batbinevt was reading all DPHs as integers. Floating point counts were truncated at the integer level, resulting in (sometimes severe) undercounts.

This has been fixed for build 12.
6.3.13  *batmasktaglc*: light curve error bars too small

<table>
<thead>
<tr>
<th>Task</th>
<th>batmasktaglc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>2.2 and earlier</td>
</tr>
<tr>
<td>What Builds</td>
<td>Swift 1.1 and earlier</td>
</tr>
<tr>
<td>Problem</td>
<td>Light curve error bars too small</td>
</tr>
<tr>
<td>Status</td>
<td>Corrected for Swift 1.2</td>
</tr>
</tbody>
</table>

Discussion: BAT mask tagged light curves are generated on board. These raw light curves require further processing before they are meaningful. More specifically, the raw light curves are generated using **unbalanced** mask weights. The task batmasktaglc accounts for this, and generates background-subtracted (balanced) light curves.

Versions 2.2 and earlier of the task incorrectly calculated the light curve error bars. This has been corrected for build 12.

6.3.14  *batmaskwtevt/batmaskwtimg*: High-Energy Background Contamination

<table>
<thead>
<tr>
<th>Task</th>
<th>batmaskwtevt and batmaskwtimg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.5 and earlier</td>
</tr>
<tr>
<td>What Builds</td>
<td>Swift 1.1 and earlier (Build 11 and earlier)</td>
</tr>
<tr>
<td>Problem</td>
<td>Background contaminates spectrum at high energies</td>
</tr>
<tr>
<td>Status</td>
<td>Closed (Swift 1.2)</td>
</tr>
</tbody>
</table>

The mask weighting approach to extracting flux requirees that the weights be **balanced**. That is, the sum of the weights must be zero. If they are not, then it is possible for the background to contaminate a weighted spectrum or light curve. This would be especially apparent at high energies where the intrinsic source flux is typically low.

Versions 1.5 of batmaskwtimg and batmaskwtevt did not have a perfectly balanced set of weights, which led to such contamination. This was a bug, and was fixed for Swift build 12.

In BAT Build 10 (Swift 1.0) the response matrix task, batdrmgen, also had problems at high energies. These problems were caused by the incident energy scale not extending to high enough energies. This problem was fixed in BAT build 11 (Swift 1.1).
Appendix A

BAT Data Formats

A.1 Introduction

This Appendix provides a detailed account of the formats of BAT data. A discussion of file naming provided; followed by descriptions of BAT science products, including event, survey and rate data; and finishing with descriptions of housekeeping and trend data products.

A.2 File Naming Overview

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):

\[
\text{sw[Observation ID][Segment Number]}b[\text{code}].[\text{suffix}]
\]

or

\[
\text{sw[Time stamp]}b[\text{code}].[\text{suffix}]
\]

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.”

A.3 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 (\text{EXTNAME} = ‘EVENTS’):
### APPENDIX A. BAT DATA FORMATS

| Name          | Format | Units | Range      | Comment                                           |
|---------------|--------|-------|------------|                                                  |
| TIME          | 1D     | s     | 0:32767    | TIME associated with event                       |
| DET_ID        | 1I     |       | (0:32767)  | Detector Channel, DM, Side and Block             |
| EVENT_FLAGS   | 1B     |       |            | DM Event Flags                                   |
| PHA           | 1I     | chan  | 0:4095     | Raw Pulse Height                                 |
| PI            | 1I     | keV   |            | Pulse Height Invariant (filled by bate-convert)  |
| MASK_WEIGHT   | 1E     |       |            | Mask weight value for source of interest (filled by batmaskwtevt) |
| DETX          | 1I     |       | 0:285      | BAT_X detector pixel number                      |
| DETY          | 1I     |       | 0:172      | BAT_Y detector pixel number                      |

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

The coding is: 
\[ (2048 \times \text{Block}) + (256 \times \text{DM}) + (128 \times \text{Side}) + \text{Channel} \].

\[ 0 \leq \text{Block} \leq 15; 0 \leq \text{DM} \leq 7; 0 \leq \text{Side} \leq 1; 0 \leq \text{Channel} \leq 127. \]

EVENT_FLAGS is a bitmask of event flags, reported in the DM Event Flag DWord, which is an optional continuation word which follows the Normal Event DWord. The following bit assignments be made:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Bit Mask</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-7</td>
<td>b00000000</td>
<td>Reserved (zero)</td>
</tr>
<tr>
<td>4</td>
<td>b00010000</td>
<td>Long format event converted to Short format (derived by bat2fits)</td>
</tr>
<tr>
<td>3</td>
<td>b00001000</td>
<td>ADC Error Flag</td>
</tr>
<tr>
<td>2</td>
<td>b00000100</td>
<td>Front Tagged Source Coincidence Flag</td>
</tr>
<tr>
<td>1</td>
<td>b00000010</td>
<td>Rear Tagged Source Coincidence Flag</td>
</tr>
<tr>
<td>0</td>
<td>b00000001</td>
<td>Multiple Hit Flag</td>
</tr>
<tr>
<td>-</td>
<td>b000X0000</td>
<td>Good event</td>
</tr>
</tbody>
</table>

When the Event Flag bitmap is not present in the telemetry stream for a given event, the respective bits will be set to zero. A valid X-ray may have bit 4 set, but for “normal short” events, the Event Flag normally be zero.

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.

### A.4 Survey 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 (EXTNAME='BAT_DPH'): 
A.5. RATE FILES

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>[Units]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>[s]</td>
<td>TIME associated with DPH</td>
</tr>
<tr>
<td>EXPOSURE</td>
<td>1D</td>
<td>[s]</td>
<td>Exposure of the histogram</td>
</tr>
<tr>
<td>DPH_COUNTS</td>
<td>(80,286,173)D</td>
<td>[count]</td>
<td>Detector plane histogram</td>
</tr>
<tr>
<td>DATA_FLAGS</td>
<td>1I</td>
<td></td>
<td>Data Quality 0=OK, 1=Problem</td>
</tr>
<tr>
<td>GAIN_INDEX</td>
<td>1J</td>
<td></td>
<td>Gain used in DPH</td>
</tr>
<tr>
<td>OFFSET_INDEX</td>
<td>1J</td>
<td></td>
<td>Offset used in DPH</td>
</tr>
<tr>
<td>LDPNAME</td>
<td>240A15</td>
<td></td>
<td>BAT LDP File Name</td>
</tr>
<tr>
<td>BLOCK_MAP</td>
<td>1I</td>
<td></td>
<td>Block Bit Mask</td>
</tr>
<tr>
<td>NUM_DETS</td>
<td>1J</td>
<td></td>
<td>Number of detectors</td>
</tr>
<tr>
<td>APID</td>
<td>1I</td>
<td></td>
<td>Application ID</td>
</tr>
<tr>
<td>LDP</td>
<td>1I</td>
<td></td>
<td>LDP product number</td>
</tr>
</tbody>
</table>

DPH_COUNTS is a three-dimensional data cube with $80 \times 286 \times 173$ 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).

The remaining columns 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. For science analysis and visualization, batbinevt should be used.

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

A.5 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”.

A.5.1 One-second Rates

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 (not background subtracted). 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 regardless of energy.
A.5.2 Quadrant Rates

Quadrant rates (code “rtqd”) contain four channel light curves for each of four quadrants of the BAT array (not background subtracted). 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Units</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>s</td>
<td>TIME of Rates</td>
</tr>
<tr>
<td>QUAD0_COUNTS</td>
<td>4I</td>
<td>count</td>
<td>Quadrant 0 counts per 1600 ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>QUAD1_COUNTS</td>
<td>4I</td>
<td>count</td>
<td>Quadrant 1 counts per 1600 ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>QUAD2_COUNTS</td>
<td>4I</td>
<td>count</td>
<td>Quadrant 2 counts per 1600 ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>QUAD3_COUNTS</td>
<td>4I</td>
<td>count</td>
<td>Quadrant 3 counts per 1600 ms time bin in 4 energy bins</td>
</tr>
</tbody>
</table>

A.5.3 64-Millisecond Rates

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Units</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>s</td>
<td>TIME of Rates</td>
</tr>
<tr>
<td>COUNTS</td>
<td>1J</td>
<td>count</td>
<td>Array counts per 1 s bin in 4 energy bins</td>
</tr>
</tbody>
</table>

A.5.4 Maximum Rates

Max rates (code “rtmc”) represent the maximum rates detected by the BAT over multiple time scales (not background subtracted). During each 8 second interval, the flight software checks each 4 ms, 8 ms, 16 ms, 32 ms and 64 ms time bin, in four energy bands, and using multiple spatial selections. There are five HDU extensions, corresponding to the five time bin sizes. Each extension contains nine four-channel series, sampled every eight second. 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 MAX_COUNTS_DTMS, where DT is one of (04, 08, 16, 32 or 64).

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 used for on-board triggering of short bursts. On the ground, it is mostly used for diagnostics.

**Layout.** Max counts for the XX millisecond time bins. \[ \text{EXTNAME} = \text{`MAX\_COUNTS\_XXMS'} \]
A.5. RATE FILES

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>[Units]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>[s]</td>
<td>TIME of Rates</td>
</tr>
<tr>
<td>QUAD0CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Quadrant 0 counts in highest XX ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>QUAD1CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Quadrant 1 counts in highest XX ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>QUAD2CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Quadrant 2 counts in highest XX ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>QUAD3CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Quadrant 3 counts in highest XX ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>Q0_1_CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Quadrant 0+1 counts in highest XX ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>Q0_2_CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Quadrant 0+2 counts in highest XX ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>Q1_3_CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Quadrant 1+3 counts in highest XX ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>Q2_3_CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Quadrant 2+3 counts in highest XX ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>ARRAY_CNTS</td>
<td>4I</td>
<td>[count]</td>
<td>Array counts in highest XX ms time bin in 4 energy bins</td>
</tr>
</tbody>
</table>

A.5.5 Mask-tagged Light Curves

Mask-tagged light curves (code “mt”) are produced by the BAT flight software, typically for three sources in the BAT field of view. In most cases, one source will be the most recent gamma-ray burst, 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, and bins the weighted events into light curves with 1.6 second time binning and four channel energy binning. The raw light curves are not background subtracted (the on-board algorithm introduces biases and scale factors that must be accounted for).

Ground processing must be performed to convert the raw weighted counts into background-subtracted rates. The `batmasktaglc` accepts the raw mask tagged rate file, the mask weight map, and the quadrant rate file (which must cover the same time period), and produces a “processed” light curve, which is background subtracted.

The raw light curve has an extension with the time and weighted counts columns (in four channels). Mask weight maps are attached in a separate extension (see “Mask Weight Maps” below).

The processed light curve has estimated rate, error and background columns. An EBOUNDS and GTI extension are also appended.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>sw*bmtNNNNNNNNN_rw.lc</td>
<td>Raw weighted counts for catalog source NNNNNNNNN</td>
</tr>
<tr>
<td>sw*bmtNNNNNNNNN.lc</td>
<td>Processed mask weighted counts for catalog source NNNNNNNN</td>
</tr>
</tbody>
</table>

**Layout.** Processed mask weighted counts. `EXTNAME = ‘RATE’`

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>[Units]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>[s]</td>
<td>TIME of Rates</td>
</tr>
<tr>
<td>RATE</td>
<td>4E</td>
<td>[count/s]</td>
<td>Processed mask weighted rate per 1600 ms time bin in 4 energy bins</td>
</tr>
<tr>
<td>ERROR</td>
<td>4E</td>
<td>[count/s]</td>
<td>Estimated statistical uncertainty in RATE column in 4 energy bins</td>
</tr>
<tr>
<td>BACKV</td>
<td>4E</td>
<td>[count/s]</td>
<td>Estimated background level in 4 energy bins</td>
</tr>
</tbody>
</table>

**Layout.** Raw mask weighted counts. `EXTNAME = ‘MASK_TAG_RATES’`

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>[Units]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>[s]</td>
<td>TIME of Rates</td>
</tr>
<tr>
<td>WEIGHTED_COUNTS</td>
<td>1J</td>
<td>[count]</td>
<td>Raw mask weighted counts per 1600 ms time bin in 4 energy bins</td>
</tr>
</tbody>
</table>
A.5.6 TDRSS GRB Light Curves

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.

**Layout.** Initial BAT TDRSS light curve EXTNAME = ‘TDRSS_LC’

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Units</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>s</td>
<td>TIME of rates</td>
</tr>
<tr>
<td>RELTIME</td>
<td>1D</td>
<td>s</td>
<td>Bin time relative to trigger time</td>
</tr>
<tr>
<td>TIMEDEL</td>
<td>1D</td>
<td>s</td>
<td>Width of time bin</td>
</tr>
<tr>
<td>RAW_COUNTS</td>
<td>4J</td>
<td>count</td>
<td>Counts per time bin in 4 energy bins</td>
</tr>
<tr>
<td>RATE</td>
<td>4E</td>
<td>count/s</td>
<td>Rate in 4 energy bins</td>
</tr>
<tr>
<td>ERROR</td>
<td>4E</td>
<td>count/s</td>
<td>Statistical uncertainty in RATE</td>
</tr>
<tr>
<td>TOT_COUNTS</td>
<td>1J</td>
<td>count</td>
<td>Total array counts in all energy bins</td>
</tr>
<tr>
<td>TOT_RATE</td>
<td>1E</td>
<td>count/s</td>
<td>Total array rate in all energy bins</td>
</tr>
<tr>
<td>TOT_RATE_ERR</td>
<td>1E</td>
<td>count/s</td>
<td>Statistical uncertainty in TOT_RATE</td>
</tr>
</tbody>
</table>

A.6 Maps

A.6.1 Detector Enable/Disable Map

Detector enable/disable maps (code “decb”). The BAT on-board software automatically maintains a map of enabled and disabled detectors. Detectors may be automatically disabled if they are noisy. Detectors may also be enabled or disabled based on ground command. When the enable/disable state changes, a new map is set to the ground. NOTE: per OGIP convention, a 0 value indicates an enabled detector, while a non-zero value indicates a disabled detector.

Detector enable/disable maps are not always produced during every observation, but the Swift Data Center should collect the nearest map for each snapshot, and should include these maps in the “decb” file.

**Layout.** Detector Enable/Disable Map EXTNAME = ‘Det_Enable_Map’

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Units</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>s</td>
<td>TIME associated with Detector Enable</td>
</tr>
<tr>
<td>FLAG</td>
<td>(286,173)B</td>
<td></td>
<td>Detector logical flag 0=enabled, 1=disabled</td>
</tr>
<tr>
<td>DATA_FLAGS</td>
<td>1I</td>
<td></td>
<td>Data Quality 0=OK, 1=Problem</td>
</tr>
<tr>
<td>LDPNAME</td>
<td>240A15</td>
<td></td>
<td>BAT File Name</td>
</tr>
<tr>
<td>BLOCK_MAP</td>
<td>1I</td>
<td></td>
<td>Block Bit Mask</td>
</tr>
<tr>
<td>NUM_DETS</td>
<td>1J</td>
<td></td>
<td>Number of detectors</td>
</tr>
<tr>
<td>APID</td>
<td>1I</td>
<td></td>
<td>Application Process Identifier</td>
</tr>
<tr>
<td>LDP</td>
<td>1J</td>
<td></td>
<td>LDP Product Number</td>
</tr>
</tbody>
</table>
A.6.2 Detector Quality Map

A detector quality map file (code “dqcb”) is a ground-derived product. The primary extension contains a quality map indicating which detectors should be used for science analysis and which should not (0=use; 1=discard). This file is typically the output of the bathotpix task, which combines the detector enable/disable map (see “Enable/Disable Map” above) and counts map which has been screened for noisy detectors.

**Layout.** Detector quality file

<table>
<thead>
<tr>
<th>Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>286×173 Map; Detector quality map (0=good; 1=bad)</td>
</tr>
<tr>
<td>BADPIX</td>
<td>List of bad pixels</td>
</tr>
</tbody>
</table>

**Layout.** List of bad pixels. EXTNAME = ‘BADPIX’

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Units/Range</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>[s]</td>
<td>Start time of hot pixel counts map</td>
</tr>
<tr>
<td>EXPOSURE</td>
<td>1D</td>
<td>[s]</td>
<td>Exposure of hot pixel counts map</td>
</tr>
<tr>
<td>DET_ID</td>
<td>1J</td>
<td>(0:32767)</td>
<td>Detector identification number</td>
</tr>
<tr>
<td>DETX</td>
<td>1I</td>
<td><a href="">pixel</a></td>
<td>BAT_X detector pixel number</td>
</tr>
<tr>
<td>DETY</td>
<td>1I</td>
<td><a href="">pixel</a></td>
<td>BAT_Y detector pixel number</td>
</tr>
<tr>
<td>QUALITY</td>
<td>1I</td>
<td></td>
<td>Detector quality flag</td>
</tr>
<tr>
<td>COUNTS</td>
<td>1J</td>
<td>[count]</td>
<td>Number of counts in flagged detector</td>
</tr>
</tbody>
</table>

The QUALITY flag can take the following values.

<table>
<thead>
<tr>
<th>QUALITY</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Good</td>
</tr>
<tr>
<td>1</td>
<td>disabled detector</td>
</tr>
<tr>
<td>2</td>
<td>cold or undersensitive detector</td>
</tr>
<tr>
<td>3</td>
<td>hot or noisy detector</td>
</tr>
</tbody>
</table>

A.6.3 Gain / Offset Map

Gain/offset maps (code “gocb”) are produced by the BAT on-board automatic calibration system, and provide the linear transformation from pulse height to keV. This map is used with other calibration files to produced calibrated energy values for event and survey data. While other non-linear corrections should be applied, the basic calculation for a given detector is, \( PI \ [\text{keV}] = PHA [\text{chan}] \times GAIN + OFFSET. \)

Gain/offset maps are not always produced during every observation, but the Swift Data Center should collect the nearest map for each snapshot, and should include these maps in the “decb” file.

**Layout.** BAT gain/offset map. EXTNAME = ‘Gain_Offset_Map’
### APPENDIX A. BAT DATA FORMATS

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Units</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>s</td>
<td>TIME associated with Gain Offsets</td>
</tr>
<tr>
<td>EXPOSURE</td>
<td>1D</td>
<td>s</td>
<td>Net TIME associated with Gain Offsets</td>
</tr>
<tr>
<td>GAIN (286,173)E</td>
<td>keV/chan</td>
<td></td>
<td>Detector Gain Values</td>
</tr>
<tr>
<td>OFFSET (286,173)E</td>
<td>chan</td>
<td></td>
<td>Detector Offset Values</td>
</tr>
<tr>
<td>DATA_FLAGS</td>
<td>I</td>
<td></td>
<td>Data Quality 0=OK, 1=Problem</td>
</tr>
<tr>
<td>LDPNAME</td>
<td>240A15</td>
<td></td>
<td>BAT File Name</td>
</tr>
<tr>
<td>BLOCK_MAP</td>
<td>I</td>
<td></td>
<td>Block Bit Mask</td>
</tr>
<tr>
<td>NUM_DETS</td>
<td>J</td>
<td></td>
<td>Number of detectors</td>
</tr>
<tr>
<td>APID</td>
<td>I</td>
<td></td>
<td>Application process Identifier</td>
</tr>
<tr>
<td>LDP</td>
<td>J</td>
<td></td>
<td>LDP Product Number</td>
</tr>
</tbody>
</table>

#### A.6.4 Mask Weight Map

Mask weight maps are produced every time the spacecraft slews to a new pointing position. These are used in combination with the raw mask tagged light curves to produce a processed mask tagged light curve (see “Mask Tagged Rates” above). The weight map is a detector plane map which contains the mask weights for a given catalog source, as derived by ray tracing. The on-board weight maps are offset and scaled such that the mean value is 7 and the range is 0-14.

**Layout.** Mask weight map (attached to each raw mask tagged light curve).  

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Units</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>s</td>
<td>TIME associated with Mask Weights</td>
</tr>
<tr>
<td>EXPOSURE</td>
<td>1D</td>
<td>s</td>
<td>Exposure of the Weight</td>
</tr>
<tr>
<td>MASK_WEIGHT</td>
<td>(286,173)E</td>
<td></td>
<td>Mask tagged weights for source</td>
</tr>
<tr>
<td>MAKSTAG</td>
<td>I</td>
<td></td>
<td>Mask Tagging enabled? (1=YES; 0=NO)</td>
</tr>
<tr>
<td>THETA</td>
<td>1E</td>
<td>deg</td>
<td>Source co-latitude in BAT field of view</td>
</tr>
<tr>
<td>PHI</td>
<td>1E</td>
<td>deg</td>
<td>Source longitude in BAT field of view</td>
</tr>
<tr>
<td>INVDIST</td>
<td>1E</td>
<td>m**{-1}</td>
<td>Inverse distance in 1/meters</td>
</tr>
<tr>
<td>BITSHIFT</td>
<td>1J</td>
<td></td>
<td>Number of bits to shift</td>
</tr>
<tr>
<td>BITMASK</td>
<td>1I</td>
<td></td>
<td>Source Bit Mask</td>
</tr>
<tr>
<td>SUM</td>
<td>4J</td>
<td></td>
<td>Sum by quadrant</td>
</tr>
<tr>
<td>SUMSQ</td>
<td>4J</td>
<td></td>
<td>Sum squared by quadrant</td>
</tr>
<tr>
<td>SUMSQMK</td>
<td>4J</td>
<td></td>
<td>Sum squared masked by quadrant</td>
</tr>
<tr>
<td>MASK</td>
<td>4J</td>
<td></td>
<td>Masked quadrant</td>
</tr>
<tr>
<td>SHIELD</td>
<td>4J</td>
<td></td>
<td>Shielded quadrant</td>
</tr>
<tr>
<td>DATA_FLAGS</td>
<td>I</td>
<td></td>
<td>Data Quality 0=OK, 1=Problem</td>
</tr>
<tr>
<td>LDPNAME</td>
<td>240A15</td>
<td></td>
<td>BAT File Name</td>
</tr>
<tr>
<td>BLOCK_MAP</td>
<td>I</td>
<td></td>
<td>Block Bit Mask</td>
</tr>
<tr>
<td>NUM_DETS</td>
<td>J</td>
<td></td>
<td>Number of detectors</td>
</tr>
<tr>
<td>APID</td>
<td>I</td>
<td></td>
<td>Application Process Identifier</td>
</tr>
<tr>
<td>LDP</td>
<td>J</td>
<td></td>
<td>LDP Product Number</td>
</tr>
</tbody>
</table>

#### A.7 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.7. TDRSS MESSAGES

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.

A.7.1 BAT GRB Alert

This provides the first evidence that the BAT has detected a potential GRB. The BAT monitors several quantities that characterize the gamma-ray sky. Each of these quantities defines an index into a table. The BAT triggers when one or more of these quantities exceeds a pre-defined threshold. The BAT GRB Alert message information includes:

- the time when the trigger was detected,
- the table index of the trigger with the highest value, and
- the trigger significance.

A description of possible quantities that can initiate a BAT trigger is contained in Fenimore, et al., "The Trigger Algorithm for the Burst Alert Telescope on Swift" (astro-ph/0408514; http://content.aip.org/APCPCS/v662/i1/491_1.html).

A.7.2 GRB Position Message (ACK or NACK)

There are two possible BAT position message types: the “BAT ACK” in which a point source can be imaged and the “BAT NACK” in which no point source is detected in the BAT image. Both message types are distributed through the GCN.

After the GRB Alert Message, the FFT-based image reconstruction from the BAT will confirm (ACK) or not confirm (NACK) that the trigger is from a point source. The ACK message contains the position of the source, intensity information and the length of the best trigger at the time of the alert. This can distinguish between fast and slow bursts but doesn’t give the full-burst fluence. The significance of the detection is included, as well as a flag where each bit gives more detailed information on the detection. The BAT position message will have an accuracy of 1–4 arcminutes, depending on the source brightness.

The NACK message is similar in content to the ACK message, but a flag is set to indicate a point source solution was not found. If the NACK message is received, (a non-confirmed trigger), no other GCN messages follow, except for the diagnostic scaled map from the BAT.

A.7.3 FoM and S/C Will/Will Not Observe

Following an ACK message for the BAT position, the BAT sends the GRB position to the onboard FoM. Two messages follow: FoM will/will not observe and S/C will/will not slew. The first reports whether the new BAT position had sufficient merit to become the next Automated Target (AT) and to request an autonomous slew. The second gives information about whether the S/C can or
will slew. In some cases, while the new BAT position may have a high merit, spacecraft pointing constraints might be such that it is impossible to slew immediately.

A.7.4 BAT Light Curves

The purpose of the BAT TDRSS light curve messages is to provide a quick indication of what the burst looks like and to allow it to be classified as long or short. For the lightcurve messages, all counts detected by the entire BAT are telemetered. **Note that BAT TDRSS light curves are not background subtracted.** This is made especially complicated during a slew, as other astrophysical sources enter and leave the BAT field of view. More complete information to do a faithful background-subtraction on BAT light curves is available from the event data.

A set of BAT light curve messages is sent if there is a position for the GRB (i.e., the BAT position message is ACK). All light curves are produced in four (adjustable) energy ranges so that some information about the hardness of the GRB may be obtained quickly; the current defaults are 15–25 keV, 25–50 keV, 50–100 keV, and 100–350 keV.

To insure the very important early-time energy and time evolution information from the GRB is received on the ground, Swift transmits three light curve messages containing overlapping data via TDRSS, each containing the four band-limited light curves. These are:

**First lightcurve message** Up to 32 seconds long, spanning $T - 24$ seconds through $T + 8$ seconds, where $T$ is the BAT rate trigger time. This message is typically cut off by the slew and might also be truncated by the data available when it is produced (i.e. if there are only 5 seconds of data past the trigger at production time, the last three seconds the first light curve will contain “UNKNOWN” values). Typically sent within 20 seconds of the BAT rate trigger.

**Updated first lightcurve message** This is a complete 32-second light curve message. To generate it, the BAT software waits about 40 seconds, then continues to wait until Swift is not slewing; it then computes the 32-second light curve spanning the $T - 24$ seconds through $T + 8$ seconds, then transmits the TDRSS message. This is essentially a copy of the first light curve message, but it is guaranteed to contain the entire 32 seconds around the GRB trigger time. This message will be useful for identifying bursts that re-brighten within a couple of minutes after the main trigger.

**Second lightcurve message** This is 128 seconds long, spanning $T - 24$ seconds through $T + 104$ seconds. This message is sent within a second of the end of the 128-second light curves, and about 200 seconds after the BAT rate trigger.

**UPDATE** (as of Mar 1, 2006) the BAT produces an additional set of light curve information, which covers more time, pre- and post-trigger. Also, finer time sampling is available around the trigger time (to enhance the sensitivy to short bursts). The original light curve is preserved unchanged; the new light curve segments are placed in another extension (EXTNAME = ‘TDRSS_LC2’).

The final BAT TDRSS message, the scaled map, is for verification of sources found and/or refined analysis of sources not found.
A.8. **HOUSEKEEPING FILES**

# A.8 Housekeeping Files

## A.8.1 Americium Survey DPHs

Americium tagged source surveys (code “amcb,” 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.

## A.8.2 Americium Block Spectra

Americium block spectra (code “amcb,” 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).

## A.8.3 Calibration Pulser Maps

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 nonlinearity of the energy calibration.

## A.8.4 Detector Array Panel (DAP) Housekeeping

The DAP housekeeping file (code “dp”) contains high level housekeeping values that pertain to the entire array.

layout. DAP housekeeping extname = ‘DAP_HK’

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>[Units]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>1D</td>
<td>[s]</td>
<td>Mission TIME in seconds</td>
</tr>
<tr>
<td>BStatus_DMRcvrOvr</td>
<td>(16,16)X</td>
<td></td>
<td>DM Receiver Overflow Flags</td>
</tr>
<tr>
<td>BStatus_CtrlSigSt</td>
<td>(16,16)X</td>
<td></td>
<td>Control Signal Status</td>
</tr>
<tr>
<td>BStatus_DMRDataE</td>
<td>(16,16)X</td>
<td></td>
<td>DM Receiver Parity/Data Code Error</td>
</tr>
<tr>
<td>BStatus_DataStat</td>
<td>(16,16)X</td>
<td></td>
<td>ADC Latchup / Overflow / FIFO Status</td>
</tr>
<tr>
<td>BStatus_DMTimOut</td>
<td>(16,16)X</td>
<td></td>
<td>DM Timeout/Invalid Command Flags</td>
</tr>
<tr>
<td>BStatus_SPWStatA</td>
<td>(16,16)X</td>
<td></td>
<td>SpaceWire Status Register A</td>
</tr>
<tr>
<td>BStatus_SPWOvrErr</td>
<td>16B</td>
<td></td>
<td>SpaceWire Overflow Error Count</td>
</tr>
<tr>
<td>BStatus_SPWEOP2Err</td>
<td>16B</td>
<td></td>
<td>SpaceWire EOP2 Error Count</td>
</tr>
<tr>
<td>BStatus_SPWFrmgErr</td>
<td>16B</td>
<td></td>
<td>SpaceWire Framing Error Count</td>
</tr>
</tbody>
</table>

<p>| BStatus_SPWStatB   | (16,16)X |         | SpaceWire Status Register B      |
| BStatus_SPWCrlRErr | 16B     |         | SpaceWire Cleared Receiver Error Ct |
| BStatus_SPWCrlPErr | 16B     |         | SpaceWire Clear Partial Pkt Error Ct |
| BBiasVolt_IbPre    | 16I     | [uA]    | IbPre Bias Voltage ADC Value     |
| BBiasVolt_IbSha    | 16I     | [uA]    | IbSha Bias Voltage ADC Value     |
| BBiasVolt_IbOTA    | 16I     | [uA]    | IbOTA Bias Voltage ADC Value     |
| BBiasVolt_IbDAC    | 16I     | [uA]    | IbDAC Bias Voltage ADC Value     |
| BBiasVolt_IbTDB    | 16I     | [uA]    | IbTDB Bias Voltage ADC Value     |
| BBiasVolt_IbRWB    | 16I     | [uA]    | IbRWB Bias Voltage ADC Value     |</p>
<table>
<thead>
<tr>
<th>Description</th>
<th>Field Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBiasVolt_DMLatch</td>
<td>16I</td>
<td>V</td>
<td>DM ADC Latchup Threshold Value</td>
</tr>
<tr>
<td>BBiasVolt_XAPLatch</td>
<td>16I</td>
<td>V</td>
<td>XA1 +2.15 V Latchup Threshold Value</td>
</tr>
<tr>
<td>BBiasVolt_BCLatch</td>
<td>16I</td>
<td>V</td>
<td>DMTDB Bias Voltage ADC Value</td>
</tr>
<tr>
<td>BBiasVolt_XAMLatch</td>
<td>16I</td>
<td>V</td>
<td>XA1 -2.15 V Latchup Threshold Value</td>
</tr>
<tr>
<td>BHVCtrl_Level</td>
<td>(8,16)I</td>
<td>V</td>
<td>High Voltage Control Levels</td>
</tr>
<tr>
<td>BDMPwr_XA1P25APwr</td>
<td>16I</td>
<td>V</td>
<td>XA1VR +2.5 V analog power level</td>
</tr>
<tr>
<td>BDMPwr_XA1M25APwr</td>
<td>16I</td>
<td>V</td>
<td>XA1VR -2.5 V analog power level</td>
</tr>
<tr>
<td>BDMPwr_DMA33APwr</td>
<td>16I</td>
<td>V</td>
<td>DM +3.3 V analog power level</td>
</tr>
<tr>
<td>BDMPwr_DM33APwr</td>
<td>16I</td>
<td>V</td>
<td>DM -3.3 V analog power level</td>
</tr>
<tr>
<td>BDMPwr_DMA33DPwr</td>
<td>16I</td>
<td>V</td>
<td>DM +3.3 V digital power level</td>
</tr>
<tr>
<td>BBCPwr_P33AVolt</td>
<td>16I</td>
<td>V</td>
<td>BCDH +3.3 V analog voltage level</td>
</tr>
<tr>
<td>BBCPwr_M33AVolt</td>
<td>16I</td>
<td>V</td>
<td>BCDH -3.3 V analog voltage level</td>
</tr>
<tr>
<td>BBCPwr_M33ACurr</td>
<td>16I</td>
<td>mA</td>
<td>BCDH -3.3 V analog current level</td>
</tr>
<tr>
<td>BBCPwr_P75AVolt</td>
<td>16I</td>
<td>V</td>
<td>BCDH +7.5 V analog voltage level</td>
</tr>
<tr>
<td>BBCPwr_P75ACurr</td>
<td>16I</td>
<td>mA</td>
<td>BCDH -7.5 V analog current level</td>
</tr>
<tr>
<td>BBCPwr_M75AVolt</td>
<td>16I</td>
<td>V</td>
<td>BCDH +7.5 V analog voltage level</td>
</tr>
<tr>
<td>BBCPwr_M75ACurr</td>
<td>16I</td>
<td>mA</td>
<td>BCDH -7.5 V analog current level</td>
</tr>
<tr>
<td>BBCPwr_P33DVolt</td>
<td>16I</td>
<td>V</td>
<td>BCDH +3.3 V digital voltage level</td>
</tr>
<tr>
<td>BBCPwr_P33DCurr</td>
<td>16I</td>
<td>mA</td>
<td>BCDH +3.3 V digital current level</td>
</tr>
<tr>
<td>BBCPwr_P55DVolt</td>
<td>16I</td>
<td>V</td>
<td>BCDH +5.0 V digital voltage level</td>
</tr>
<tr>
<td>BBCPwr_P55DCurr</td>
<td>16I</td>
<td>mA</td>
<td>BCDH +5.0 V digital current level</td>
</tr>
<tr>
<td>BBCPwr_P55DVolt</td>
<td>16I</td>
<td>V</td>
<td>BCDH +5.0 V digital voltage level</td>
</tr>
<tr>
<td>BBCPwr_P55DVolt</td>
<td>16I</td>
<td>V</td>
<td>BCDH +5.0 V digital voltage level</td>
</tr>
<tr>
<td>BBCC_Acc JAVA2</td>
<td>16I</td>
<td>V</td>
<td>BCDH Analog Ground</td>
</tr>
<tr>
<td>BHVCtrl_Level</td>
<td>(8,16)I</td>
<td>V</td>
<td>High Voltage Monitor Levels</td>
</tr>
<tr>
<td>DM_Status_Flags</td>
<td>(16,16,16)X</td>
<td></td>
<td>DM Status Flags</td>
</tr>
<tr>
<td>DM_Status_XA1Ver</td>
<td>(16,16)B</td>
<td></td>
<td>DM XA1 Version</td>
</tr>
<tr>
<td>DM_Status_ErrFlag</td>
<td>(8,16)X</td>
<td></td>
<td>DM Error Flags</td>
</tr>
<tr>
<td>DM_Status_SuspectCmd</td>
<td>(16,16)B</td>
<td></td>
<td>DM SuspectCmdCode</td>
</tr>
<tr>
<td>DM_Status_ErrIDCCnt</td>
<td>(16,16)B</td>
<td></td>
<td>DM ErrorIDCCNT</td>
</tr>
</tbody>
</table>
### A.8. HOUSEKEEPING FILES

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM_Status_FilterMask</td>
<td>(16,16,16)X</td>
<td>DM FilterMask</td>
</tr>
<tr>
<td>DM_TagSrc_FrCnt</td>
<td>(8,16)J</td>
<td>DM Front Tagged Source Count</td>
</tr>
<tr>
<td>DM_TagSrc_RrCnt</td>
<td>(8,16)J</td>
<td>DM Rear Tagged Source Count</td>
</tr>
<tr>
<td>DM_TagSrc_100usec</td>
<td>(16,16)B</td>
<td>100 usec Relative Timestamp</td>
</tr>
<tr>
<td>DM_HK_Temp</td>
<td>(16,16)I</td>
<td>DM Side temperature</td>
</tr>
<tr>
<td>DM_HK_HVLkCurr</td>
<td>(16,16)I</td>
<td>High Voltage Leakage Current</td>
</tr>
<tr>
<td>DM_HK_Vthr</td>
<td>(16,16)I</td>
<td>XA1 Discriminator Threshold Voltage</td>
</tr>
<tr>
<td>DM_HK_Vcaj</td>
<td>(16,16)I</td>
<td>XA1 output buffer offset adjust</td>
</tr>
<tr>
<td>DM_HK_Vfs</td>
<td>(8,16)I</td>
<td>Cal pulser voltage (Side 0 ONLY)</td>
</tr>
<tr>
<td>DM_HK_Vfs</td>
<td>(8,16)I</td>
<td>XA1 Feedback shaper (Side 1 ONLY)</td>
</tr>
<tr>
<td>DM_Cnts_LLD</td>
<td>(16,16)J</td>
<td>LLD Count</td>
</tr>
<tr>
<td>DM_CntsEvt</td>
<td>(16,16)J</td>
<td>Event Count</td>
</tr>
<tr>
<td>DM_Cnts_100usec</td>
<td>(16,16)B</td>
<td>100 usec Timestamp count</td>
</tr>
<tr>
<td>DM_Cnts_MLD</td>
<td>(16,16)I</td>
<td>Multi channel hit Count</td>
</tr>
<tr>
<td>DM_Cnts_Lost</td>
<td>(16,16)I</td>
<td>Lost Event Count</td>
</tr>
<tr>
<td>DM_Coef_SegGain</td>
<td>(16,16)I</td>
<td>Segment Gain Coefficient</td>
</tr>
<tr>
<td>DM_Coef_SegOffset</td>
<td>(16,16)I</td>
<td>Segment Offset Coefficient</td>
</tr>
<tr>
<td>DM_Coef_STRIPGain</td>
<td>(16,16)I</td>
<td>Strip Gain Coefficient</td>
</tr>
<tr>
<td>DM_Coef_STRIPOffset</td>
<td>(16,16)I</td>
<td>Segment Offset Coefficient</td>
</tr>
<tr>
<td>DMParms_CalPeriod</td>
<td>(16,16)I</td>
<td>Calibration Pulser Period</td>
</tr>
<tr>
<td>DM_Parms_DMCVersion</td>
<td>(16,16)B</td>
<td>DMC FPGA Version Number</td>
</tr>
<tr>
<td>DM_Parms_TagSrcEn</td>
<td>(8,16,16)X</td>
<td>Tagged Source Front / Rear Enable</td>
</tr>
<tr>
<td>DM_Parms_TagSrcFrDel</td>
<td>(16,16)I</td>
<td>Tagged Source Front Delay</td>
</tr>
<tr>
<td>DM_Parms_TagSrcRrDel</td>
<td>(16,16)I</td>
<td>Tagged Source Rear Delay</td>
</tr>
<tr>
<td>DM_Parms_TestMuxAddr</td>
<td>(16,16)I</td>
<td>Testport Multiplexer Address Stng</td>
</tr>
<tr>
<td>DM_Enable</td>
<td>(128,16,16)X</td>
<td>DM Channel Control Enable Bits</td>
</tr>
<tr>
<td>DM_DSR1_XA1_VFP_A</td>
<td>(16,16)I</td>
<td>XA1 Feedback preamp Bias Voltage (A)</td>
</tr>
<tr>
<td>DM_DSR1_XA1_VTHR_A</td>
<td>(16,16)I</td>
<td>XA1 Discriminator Threshold Volt. (A)</td>
</tr>
<tr>
<td>DM_DSR1_XA1_VCADJ_A</td>
<td>(16,16)I</td>
<td>XA1 Output Buffer Offset Adjust (A)</td>
</tr>
<tr>
<td>DM_DSR1_XA1_VCAL_A</td>
<td>(16,16)I</td>
<td>Calibration Pulser Voltage</td>
</tr>
<tr>
<td>DM_DSR2_XA1_VFP_B</td>
<td>(16,16)I</td>
<td>XA1 Feedback preamp Bias Voltage (B)</td>
</tr>
<tr>
<td>DM_DSR2_XA1_VTHR_B</td>
<td>(16,16)I</td>
<td>XA1 Discriminator Threshold Volt. (B)</td>
</tr>
<tr>
<td>DM_DSR2_XA1_VCADJ_B</td>
<td>(16,16)I</td>
<td>XA1 Output Buffer Offset Adjust (B)</td>
</tr>
<tr>
<td>DM_DSR2_XA1_VFS</td>
<td>(16,16)I</td>
<td>XA1 feedback shaper bias voltage</td>
</tr>
<tr>
<td>DM_ChannelTest</td>
<td>(128,16,16)X</td>
<td>XA1 Shadow Register Channel Test</td>
</tr>
<tr>
<td>DM_ChannelMask</td>
<td>(128,16,16)X</td>
<td>XA1 Shadow Register Channel Mask</td>
</tr>
<tr>
<td>DM_XSR4_XA1_Addr</td>
<td>(16,16)B</td>
<td>XA1 Chip Address</td>
</tr>
<tr>
<td>DM_XSR4_Cmd</td>
<td>(3,16,16)X</td>
<td>XA1 Mode Control Bits</td>
</tr>
<tr>
<td>APID</td>
<td>II</td>
<td>Application Process Identifier</td>
</tr>
<tr>
<td>LDP</td>
<td>1J</td>
<td>LDP Product Number</td>
</tr>
<tr>
<td>DATA_FLAGS</td>
<td>II</td>
<td>Data Quality 0=OK, 1=Problem</td>
</tr>
</tbody>
</table>

Several products have different dimensions. The array dimensions refer to data values stored on a per-block, per-DM or per-side basis, as follows.
### APPENDIX A. BAT DATA FORMATS

<table>
<thead>
<tr>
<th>Array dimension</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Per block values</td>
</tr>
<tr>
<td>(8,16)</td>
<td>Per DM values (8 DMs x 16 blocks)</td>
</tr>
<tr>
<td>(16,16)</td>
<td>Per side values (16 sides x 16 blocks)</td>
</tr>
<tr>
<td>(8,16,16)</td>
<td>Per side flags (8 flag bits x 16 sides x 16 blocks)</td>
</tr>
<tr>
<td>(16,16,16)</td>
<td>Per side flags (16 flag bits x 16 sides x 16 blocks)</td>
</tr>
<tr>
<td>(128,16,16)</td>
<td>Per detector flags (128 detectors x 16 sides x 16 blocks)</td>
</tr>
<tr>
<td>(3,16,16)</td>
<td>Control flags (3 flag bits x 16 sides x 16 blocks)</td>
</tr>
</tbody>
</table>

#### A.8.5 BAT Engineering Housekeeping

The BAT produces extensive engineering housekeeping data (code “en”), which is not normally meant for scientific use. The contents of each housekeeping file are too numerous to discuss here. Instead, we give a short description of each application ID (“APID”). The APID number is given first in decimal and then in hexadecimal.

The extension names in the engineering housekeeping file are hkNNNx001, where NNN is the hexadecimal APID.

<table>
<thead>
<tr>
<th>APID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>288</td>
<td>0x0120 BAT Minimum APID - SYS_RT_TLM_MIN_AID</td>
</tr>
<tr>
<td>289</td>
<td>0x0121 BAT Comp Hub Task Execution Profiler Telemetry Packet (part 1)</td>
</tr>
<tr>
<td>290</td>
<td>0x0122 BAT Comp Hub Task Execution Profiler Telemetry Packet (part 2)</td>
</tr>
<tr>
<td>291</td>
<td>0x0123 BAT SO task housekeeping telemetry - HK_CDH_SW_EVENT_AID</td>
</tr>
<tr>
<td>293</td>
<td>0x0125 BAT SM task non-generic housekeeping telemetry</td>
</tr>
<tr>
<td>294</td>
<td>0x0126 BAT CDH Memory Dump packet</td>
</tr>
<tr>
<td>298</td>
<td>0x012a BAT Data types packet</td>
</tr>
<tr>
<td>300</td>
<td>0x012c BAT CDH File Information Packet</td>
</tr>
<tr>
<td>301</td>
<td>0x012d BAT SC task (part 2) non-generic housekeeping telemetry</td>
</tr>
<tr>
<td>304</td>
<td>0x0130 BAT Combined Housekeeping (HK &amp; PD Tasks) - HK_ENG_PKT_01_AID</td>
</tr>
<tr>
<td>305</td>
<td>0x0131 BAT Combined Housekeeping (SM, SH, SB &amp; XR Tasks) - HK_ENG_PKT_02_AID</td>
</tr>
<tr>
<td>306</td>
<td>0x0132 BAT Combined Housekeeping (TM &amp; CI Tasks) - HK_ENG_PKT_03_AID</td>
</tr>
<tr>
<td>307</td>
<td>0x0133 BAT Combined Housekeeping (PW, FO, ST &amp; DM Tasks) - HK_ENG_PKT_04_AID</td>
</tr>
<tr>
<td>308</td>
<td>0x0134 BAT Combined Housekeeping (SC Task) - HK_ENG_PKT_05_AID</td>
</tr>
<tr>
<td>309</td>
<td>0x0135 BAT Combined Housekeeping (FM &amp; DSP Tasks) - HK_ENG_PKT_06_AID</td>
</tr>
<tr>
<td>310</td>
<td>0x0136 BAT Combined Housekeeping (IH &amp; SC Tasks) - HK_ENG_PKT_07_AID</td>
</tr>
<tr>
<td>311</td>
<td>0x0137 BAT Combined Housekeeping (TO &amp; MS Tasks) - HK_ENG_PKT_08_AID</td>
</tr>
<tr>
<td>312</td>
<td>0x0138 BAT Combined Housekeeping (BR &amp; CA Tasks) - HK_SCI_PKT_01_AID</td>
</tr>
<tr>
<td>313</td>
<td>0x0139 BAT Combined Housekeeping (DI &amp; TG Tasks) - HK_SCI_PKT_02_AID</td>
</tr>
<tr>
<td>314</td>
<td>0x013a BAT Combined Housekeeping (SP Task) - HK_SCI_PKT_03_AID</td>
</tr>
<tr>
<td>315</td>
<td>0x013b BAT Task Command Request/Error/Execution Counts - HK_GENERIC_PKT_AID</td>
</tr>
<tr>
<td>320</td>
<td>0x0140 BAT Power Box Housekeeping Packet (part #1) [RealTime]</td>
</tr>
<tr>
<td>321</td>
<td>0x0141 BAT Power Box Housekeeping Packet (part #2) [RealTime]</td>
</tr>
<tr>
<td>322</td>
<td>0x0142 BAT Power Box Housekeeping Packet (part #3) [RealTime]</td>
</tr>
<tr>
<td>323</td>
<td>0x0143 BAT MIC Hardware Status Packet - MIC_HW_STATUS_AID</td>
</tr>
<tr>
<td>324</td>
<td>0x0144 BAT DSP Diagnostic Data Packet - DSP_HW_STATUS_AID</td>
</tr>
<tr>
<td>325</td>
<td>0x0145 BAT Power Box Status</td>
</tr>
<tr>
<td>326</td>
<td>0x0146 BAT DSP Test packet - DSP_AVER_RSP_PKT_AID</td>
</tr>
<tr>
<td>327</td>
<td>0x0147 BAT DSP Test packet - DSP_SUM_RSP_PKT_AID</td>
</tr>
<tr>
<td>Hex Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x0153</td>
<td>BAT TG Task Status Telemetry, Real-time channel</td>
</tr>
<tr>
<td>0x0154</td>
<td>BAT DI Task parameters (Realtime)</td>
</tr>
<tr>
<td>0x015f</td>
<td>BAT TG Task Criterion Dump Telemetry, Real-time channel</td>
</tr>
<tr>
<td>0x0160</td>
<td>BAT BR Science Clean Strategy</td>
</tr>
<tr>
<td>0x0161</td>
<td>BAT BR Rest of DSP State</td>
</tr>
<tr>
<td>0x0170</td>
<td>BAT RealTime Science Files Packet</td>
</tr>
<tr>
<td>0x0171</td>
<td>BAT Sparse DAP HK - HK_MEM_PKT_01_AID</td>
</tr>
<tr>
<td>0x0172</td>
<td>BAT Unknown - HK_MEM_PKT_02_AID</td>
</tr>
<tr>
<td>0x0173</td>
<td>BAT Realtime Leakage Currents Packet #1 - HK_MEM_PKT_03_AID</td>
</tr>
<tr>
<td>0x0174</td>
<td>BAT Realtime Leakage Currents Packet #2 - HK_MEM_PKT_04_AID</td>
</tr>
<tr>
<td>0x0175</td>
<td>BAT Realtime Leakage Currents Packet #3 - HK_MEM_PKT_05_AID</td>
</tr>
<tr>
<td>0x0176</td>
<td>BAT Realtime XA1 Temperature Packet #1 - HK_MEM_PKT_06_AID</td>
</tr>
<tr>
<td>0x0177</td>
<td>BAT Realtime XA1 Temperature Packet #2 - HK_MEM_PKT_07_AID</td>
</tr>
<tr>
<td>0x0178</td>
<td>BAT Realtime XA1 Temperature Packet #3 - HK_MEM_PKT_08_AID</td>
</tr>
<tr>
<td>0x0179</td>
<td>BAT CDH Housekeeping -- HK_MEM_PKT_09_AID</td>
</tr>
<tr>
<td>0x017a</td>
<td>BAT CDH Housekeeping -- HK_MEM_PKT_0A_AID</td>
</tr>
<tr>
<td>0x018a</td>
<td>BAT MIC Hardware Status Packet - TDRS_MIC_HI_STATUS_AID</td>
</tr>
<tr>
<td>0x018b</td>
<td>BAT DI task parameters (TDRSS)</td>
</tr>
<tr>
<td>0x018c</td>
<td>BAT Power Box Housekeeping Packet; TDRSS piece #1</td>
</tr>
<tr>
<td>0x018d</td>
<td>BAT Power Box Housekeeping Packet; TDRSS piece #2</td>
</tr>
<tr>
<td>0x018e</td>
<td>BAT Power Box Housekeeping Packet; TDRSS piece #3</td>
</tr>
<tr>
<td>0x018f</td>
<td>BAT TDRSS mixed HK packet</td>
</tr>
<tr>
<td>0x0190</td>
<td>FOM 2 Observe AT Target msg PKT - TDRS_FOM_OBS_AT_AID</td>
</tr>
<tr>
<td>0x0191</td>
<td>FOM FO SC 2 Observe AT Target msg PKT - TDRS_SC_OBS_AT_AID</td>
</tr>
<tr>
<td>0x0192</td>
<td>FOM FO SC 2 Observe PPT msg PKT - TDRS_SC_OBS_PPT_AID</td>
</tr>
<tr>
<td>0x0193</td>
<td>FOM FO SC 2 Safe Point msg PKT - TDRS_SC_SAFE_PT_AID</td>
</tr>
<tr>
<td>0x0194</td>
<td>FOM FO SC Slew Abort msg PKT - TDRS_SC_SLEW_ABORT_AID</td>
</tr>
<tr>
<td>0x0195</td>
<td>FOM TDRSS FO DIAG TLM PKT - FO_DIAG_PKT_AID</td>
</tr>
<tr>
<td>0x0196</td>
<td>BAT CDH Memory Dump packet via TDRSS</td>
</tr>
<tr>
<td>0x0197</td>
<td>BAT Engineering HK - TDRS_BAT_HK03_AID</td>
</tr>
<tr>
<td>0x0198</td>
<td>BAT Engineering HK - TDRS_BAT_HK04_AID</td>
</tr>
<tr>
<td>0x0199</td>
<td>BAT Engineering HK - TDRS_BAT_HK05_AID</td>
</tr>
<tr>
<td>0x019a</td>
<td>BAT Engineering HK - TDRS_BAT_HK06_AID</td>
</tr>
<tr>
<td>0x01e0</td>
<td>BAT Power Box Diagnostic Packet</td>
</tr>
<tr>
<td>0x01e2</td>
<td>FOM Diagnostic TLM PKT - FO_DIAG_PKT_AID</td>
</tr>
<tr>
<td>0x01e3</td>
<td>BAT CDH Memory Dwell packet</td>
</tr>
<tr>
<td>0x01e5</td>
<td>BAT Spacecraft ACS echo</td>
</tr>
<tr>
<td>0x01ea</td>
<td>BAT TG Task Status Telemetry, High-priority SSR channel</td>
</tr>
<tr>
<td>0x01ec</td>
<td>BAT DI task parameters (Hi-priority)</td>
</tr>
<tr>
<td>0x01ef</td>
<td>BAT SCI_RATE_SPARSE2_H_AID (?)</td>
</tr>
<tr>
<td>0x01f0</td>
<td>BAT TG Task Criterion Dump Telemetry, High-priority SSR channel</td>
</tr>
<tr>
<td>0x01f1</td>
<td>BAT BR GRB flux info</td>
</tr>
<tr>
<td>0x0214</td>
<td>BAT Bumped Files</td>
</tr>
<tr>
<td>0x0215</td>
<td>BAT Imaging Result Telemetry Pkt (Burst Response Task), Lo-priority</td>
</tr>
<tr>
<td>0x0216</td>
<td>BAT DSP source catalog result</td>
</tr>
</tbody>
</table>
APPENDIX A. BAT DATA FORMATS

535 0x0217 BAT DSP Update detector enabled map
536 0x0218 BAT DSP Debug mask result
537 0x0219 BAT SCI_DSP_IMAGE_TEST_RESULT_AID
538 0x021a BAT DSP Science response packet to TG
539 0x021b BAT DSP Clean Strategy result mask
540 0x021c BAT DSP Science response packet
541 0x021d BAT DSP Image init done
542 0x021e BAT DSP Update pointing done
544 0x0220 BAT Fault Count Packet
545 0x0221 BAT BCDH/DM cmd verifies and exceptions

A.8.6 DI Commandables

The DI commandables file contains the command settings of the “Data Ingest” task. This task performs the basic computations on raw event data from the array. The events are transformed into light curves on various time scales, survey DPHs, and other products. While a detailed description of every field in this file is beyond the scope of this document, the major categories are:

- Trigger thresholds (TRIGEN, THCERT, THLONG, THPOSS, THPROJ, THSHRT);
- Noisy detector management (NHTOKL, NHTO TL);
- Survey and calibration intervals (SURDUR, AM2DUR);
- Survey and light curve energy bin edges;
- Automatic script names;
- DM calibration coefficients;

A.8.7 Debug Stream

The BAT produces continuous logs of its activities. This is primarily used for debugging purposes. There are actually two separate streams, the “debug” stream, which contains the BAT flight software debugging information, and the “shell” stream, which is a log of the VxWorks command shell. The files contain ASCII substrings of variable length. The LENGTH column determines the length of the substring, and the ASCII column contains the actual character data. Note that this log information is not meant to be used for any scientific purposes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>[Units]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGSEC</td>
<td>1J</td>
<td>[s]</td>
<td>Spacecraft MET of log entry</td>
</tr>
<tr>
<td>LOGSUBSEC</td>
<td>1I</td>
<td>[s]</td>
<td>Spacecraft MET subseconds of log entry</td>
</tr>
<tr>
<td>APID</td>
<td>1I</td>
<td></td>
<td>Application ID data originally from</td>
</tr>
<tr>
<td>LENGTH</td>
<td>1I</td>
<td></td>
<td>Log entry length in characters</td>
</tr>
<tr>
<td>ASCII</td>
<td>215A</td>
<td></td>
<td>Log entry text</td>
</tr>
</tbody>
</table>

A.8.8 Trigger Diagnostic Tables

The trigger system of the BAT flight software produces periodic table dumps of the on-board trigger tables, and are stored in trend data on the ground. The trigger tables are: “long”, “short”, “rate”, and “image”.
Appendix B

BAT FITS Keywords

B.1 Introduction

The BAT ground software operates on standard FITS data files. However, the software produces — and requires — certain FITS keywords in order to operate properly. This appendix describes the keywords used. It also describes the HDUCLASn keywords used to identify the contents of data files.

B.1.1 Keywords Used

The BAT software uses FITS keywords to both document the analysis procedures used, and to pass information to downstream tasks. This section lists those keywords, how they are created and used, and by what tasks.

- **ACOLAPP** - a boolean keyword indicating whether the autocollimation correction has been applied to the image plate scale. Set by: batfftimage. Used by: none.

- **BACKAPP** - a boolean keyword which indicates whether the BAT background has been subtracted. This can mean some form of explicit subtraction, or automatic subtraction via the mask-weighting/imaging approaches. Set by: batfftimage, batbinevt. Used by: batclean.

- **BALAPP** - a boolean keyword indicating whether fine-level balancing has been performed on a detector image. Set by: batclean. Used by: NONE

- **BAT\{X,Y,Z\}OBJ** - floating point keywords which indicate where in the BAT field of view the source is located. The coordinate system is the BAT coordinate system, and the units are centimeters. Objects at at infinity are placed at 10 km. Set by: batmaskwtevt, batmaskwting, batupdatephakw. Used by: batcelldetect, batdrmgen.

- **BAT_RA, BAT_DEC** - floating point keywords which indicate where on the sky the source is located (celestial RA and Dec). Set by: batmaskwtevt, batmaskwting. Used by: NONE.

- **CLEANLEV** - an integer keyword indicating the number of iterations of cleaning have been performed. Set by: batclean. Used by: batclean.

- **DISTAPP** - a boolean keyword indicating whether the image, mask weights, or source detection results have been adjusted for non-linear image distortion effects. Set by: batcelldetect, batwarping, batmaskwtevt, batmaskwting. Used by: batwarping, batcelldetect.
APPENDIX B. BAT FITS KEYWORDS

- **FFAPP** - a boolean keyword indicating whether the image, mask weights or fluxes have been corrected for geometric projection effects. Set by: batfftimage, batmaskwtimg, batmaskwtevt. Used by: NONE.

- **FLUXMETH** - a string keyword which indicates what kind of flux extraction method was used. Allowed values are ‘RAW’ (pure unsubtracted counts); or ‘WEIGHTED’ (mask-weighted light curve or image = background-subtracted). Set by: batfftimage, batbinevt. Used by: batdrmgen.

- **GAINAPP** - a boolean keyword indicating whether BAT energy calibration has been applied. Set by: bateconvert, baterebin. Used by: bateconvert, batdrmgen.

- **GAINMETH** - a string keyword indicating what kind of BAT energy calibration was performed. Allowed values are: ‘LINEAR’ (linear ADU to energy conversion); QUADRATIC (quadratic ADU to energy conversion); ‘CUBIC’ (cubic ADU to energy conversion); ‘DIRECTCUBIC’ (obsolete); ‘FIXEDDAC’ (cubic ADU to energy conversion at fixed DAC level). Set by: bateconvert, baterebin. Used by: batdrmgen, baterebin.

- **IMATYPE** - a string indicating the type of image produced. Allowed values are: ‘INTENSITY’ (rate map); ‘EXPOSURE’ (partial coding / vignetting map); ‘SIGNIFICANCE’ (signal-to-noise map); ‘BACKGROUND’ (mean background map); ‘VARIANCE’ (variance map); ‘ERROR’ (standard deviation map). Set by: batfftimage, batcelldetect. Used by: NONE.

- **MSKWTAPP** - a boolean keyword indicating whether the image, mask weights or fluxes have been normalized by the efficiency of the mask weighting algorithm. Set by: batfftimage, batmaskwtimg, batmaskwtevt. Used by: NONE.

- **MSKWTSQF** - a floating point keyword recording the half-variance of the mask weights. Set by: batfftimage, batmaskwtimg, batmaskwtevt, batupdatephakw. Used by: batdrmgen.

- **NGOODPIX** - an integer keyword recording the number of enabled detectors (pixels). Set by: batfftimage, batmaskwtimg, batmaskwtevt, batupdatephakw. Used by: NONE.

- **NGPIXAPP** - a boolean keyword indicating whether the image, mask weights or fluxes have been normalized by the number of enabled detectors. Set by: batfftimage, batmaskwtimg, batmaskwtevt. Used by: NONE.

- **OVERSMP\{X,Y\}** - an integer keyword recording the oversampling factor used in the image in the BAT\_X and BAT\_Y directions. Set by: batfftimage. Used by: batcelldetect.

- **OCCAPP** - a boolean keyword indicating whether the image has been corrected for occultation by the earth, moon or sun. Set by: batoccultmap. Used by: batoccultmap.

- **OCCCTYPE** - a string keyword indicating how the image was corrected for occultation. Allowed values: ‘DIVIDE’ if the image was divided by the fractional exposure; ‘MULTIPLY’ if the image was multiplied by the fractional exposure. Set by: batoccultmap. Used by: NONE.

- **PCODEAPP** - a boolean keyword indicating whether the image, mask weights, or fluxes have been corrected for partial coding. Set by: batfftimage, batmaskwting, batmaskwtevt. Used by: NONE.

- **PCODEFR** - a floating point keyword recording the partial coding fraction for a given source in the BAT field of view (a number between zero and one). Set by: batmaskwtevt, batmaskwting, batupdatephakw. Used by: NONE
In addition to the tasks listed above, it is also common for FTOOLs tasks to transfer existing keywords from the input file to the output file, so it is possible for the keywords to appear in other outputs than described.

**B.1.2 Data Types**

BAT software uses and creates various data types.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>HDUCLAS1</th>
<th>HDUCLAS2</th>
<th>HDUCLAS3</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light curve</td>
<td>‘LIGHTCURVE’</td>
<td>‘NET’ or ‘TOTAL’</td>
<td>‘RATE’ or ‘COUNT’</td>
<td>Original, batbinevt, batmasktaglc</td>
</tr>
<tr>
<td>Spectrum</td>
<td>‘SPECTRUM’</td>
<td>‘NET’ or ‘TOTAL’</td>
<td>‘RATE’ or ‘COUNT’</td>
<td>batbinevt</td>
</tr>
<tr>
<td>Survey Histogram (DPH)</td>
<td>‘ARRAY’</td>
<td>‘TOTAL’</td>
<td>–</td>
<td>Original, batbinevt</td>
</tr>
<tr>
<td>Detector Plane Image (DPI)</td>
<td>‘ARRAY’</td>
<td>‘TOTAL’</td>
<td>–</td>
<td>Original, batbinevt</td>
</tr>
<tr>
<td>Cleaned DPI or DPH</td>
<td>‘ARRAY’</td>
<td>‘RESIDUAL’</td>
<td>–</td>
<td>batclean, batbinevt, batfftimage</td>
</tr>
<tr>
<td>Model background DPI or DPH</td>
<td>‘ARRAY’</td>
<td>‘PREDICTED’</td>
<td>–</td>
<td>batclean, batbinevt, batfftimage</td>
</tr>
<tr>
<td>Energy Bins</td>
<td>‘RESPONSE’</td>
<td>‘EBOUNDS’</td>
<td>–</td>
<td>batbinevt, batdrrmgen</td>
</tr>
<tr>
<td>Flux Image</td>
<td>‘IMAGE’</td>
<td>‘NET’</td>
<td>–</td>
<td>batfftimage, batcelldetect</td>
</tr>
<tr>
<td>Significance Image</td>
<td>‘IMAGE’</td>
<td>‘SIGNIFICANCE’</td>
<td>–</td>
<td>batfftimage, batcelldetect</td>
</tr>
<tr>
<td>Mean Background Image</td>
<td>‘IMAGE’</td>
<td>‘BKG’</td>
<td>–</td>
<td>batfftimage, batcelldetect</td>
</tr>
<tr>
<td>Background Variation Image</td>
<td>‘IMAGE’</td>
<td>‘BKG_STDDEV’</td>
<td>‘MEASURED’ or ‘PREDICTED’</td>
<td>batfftimage, batcelldetect</td>
</tr>
<tr>
<td>Background Variance Image</td>
<td>‘IMAGE’</td>
<td>‘BKG_VARIANCE’</td>
<td>‘MEASURED’ or ‘PREDICTED’</td>
<td>batfftimage, batcelldetect</td>
</tr>
<tr>
<td>Partial Coding Image</td>
<td>‘IMAGE’</td>
<td>‘VIGNETTING’</td>
<td>–</td>
<td>batfftimage, batcelldetect</td>
</tr>
<tr>
<td>Occultation Image</td>
<td>‘IMAGE’</td>
<td>‘VIGNETTING’</td>
<td>‘OCCULTATION’</td>
<td>batfftimage, batcelldetect</td>
</tr>
<tr>
<td>Response Matrix</td>
<td>‘RESPONSE’</td>
<td>‘RSP_MATRIX’</td>
<td>‘FULL’</td>
<td>batdrrmgen</td>
</tr>
</tbody>
</table>
APPENDIX B. BAT FITS KEYWORDS
Appendix C

BAT Acronym and Terminology
Glossary

This Appendix contains a glossary of special terms and acronyms related to the BAT.

1553 Bus Communication bus connecting the BAT to the spacecraft systems.

Aperture pattern of open and blocked cells used in the BAT mask. Also, the aperture data file records this pattern for use in image analysis.

BAT Burst Alert Telescope

Block A BAT block contains 8 DMs and 2048 detectors. There are 16 blocks in the detector array.

CALDB Calibration Database, used to store calibration and auxiliary files for science analysis. Available from the Swift web page.

CdZnTe Cadmium Zinc Teluride, the BAT detector technology.

CZT same as CdZnTe.

Coded Aperture A type of non-imaging optics, consisting of absorbing elements arranged in a specific pattern above a detector plane.

DAP Detector Array Plane, a term which refers to the entire detector array.

Disable/Enable Map Map produced on-board which contains the individual detectors which were enabled or disabled. A disable/enable map is produced whenever the instrument configuration changes on board, and should be present in each observation data set.

DM Detector Module, a component that contains two DM Sides and 256 detectors.

DM Side Detector Module Side, containing 128 detectors. Two sides are arranged into a single Detector Module.

DPH Detector Plane Histogram, a histogram accumulated for each detector in several energy bands.

DPI Detector Plane Histogram, an image (histogram) accumulated for each detector in a single energy band.

DRM Detector Response Matrix, the response function used in spectral analysis.
**FFT** Fast Fourier Transform, used during image reconstruction.

**Focal Length** In the context of the BAT, “focal length” refers to the distance between the mask and detector arrays.

**FOM** Figure of Merit.

**FOV** Field of View, the observable instrumental solid angle.

**FTOOLS** Software distribution (also known as HEASOFT) which contains the BAT software tasks.

**Gain/Offset Map** Detector map produced on-board which contains the coefficients for a linear pulse-height to energy conversion. A gain/offset map is produced after each calibration pulser map is constructed, and should be present in each observation data set.

**GRB** Gamma-Ray Burst.

**HK** Housekeeping data.

**IMX,IMY** BAT tangent plane coordinate labels.

**IP** Image Processor, the flight computer of the BAT.

**Malindi** Name of the primary ground station which supports Swift.

**Malindi Gap** Approximate five hour period of each day when there are no possible Malindi ground station contacts.

**Mask** The BAT coded aperture consisting of approximately 52,000 lead tiles.

**Mask tagged light curves** set of light curves generated by the BAT flight software for up to three sources.

**Mask weighting** Procedure for generating mathematical weights for each event, to be used to generated background subtracted light curves, spectra or images. The mask weighting procedure involves ray tracing.

**Maximum rates** A special kind of rate history which gives the maximum rate detected on various timescales. This time series is used in real-time by the flight software for triggering. (also “max rates”)

**Multi-hit** A type of diagnostic event which indicates that multiple detectors in a DM Sandwich were triggered simultaneously, indicating a cosmic ray shower.

**Partial coding** Indicates the fractional illumination of the detector array by an off-axis source.

**Phi** BAT longitude coordinate.

**PSF** Point spread function, the spatial response of the instrument to an incident point source.

**Pulser** Electronic device used to calibrate the BAT electronic pulse height scale. Periodic pulser calibration cycles are used to produce a calibration pulser map (trend product). These maps are used to update the gain/offset map.

**Quadrant Rates** A type of light curve produced by the BAT, which contains the count rates in each of four geographic quadrants of the detector array (also “quad rates”).
Quality Map  Detector map produced on the ground, which combines information about disabled/enabled detectors, and noisy detectors discovered in post-processing.

SAA  South Atlantic Anomaly, region of high ionizing radiation in low earth orbit. The BAT does not typically produce science data during SAA passes.

Sandwich  same as DM Side.

Spacewire  communication bus connecting BAT blocks to the Image Processor.

SSR  Solid State Recorder, the spacecraft on-board storage system.

Survey  image data collected while waiting for the next GRB (in the form of DPHs).

Tagged Source  Radioactive $^{241}$Am calibration source used for on-board BAT calibration.

Theta  BAT off-axis angle.

UVOT  Ultra-Violet Optical Telescope, the UV/Optical instrument aboard Swift.

XRT  X-Ray Telescope, the X-ray instrument aboard Swift.
Appendix D

BAT Software Tool Reference

This appendix is for reference. It contains documentation files for all BAT software tasks.
D.1 BATBINEVT

D.1.1 NAME

batbinevt - accumulate BAT event and DPH data into spectra, lightcurves or images

D.1.2 USAGE

batbinevt infile outfile outtype timedel timebinalg energybins

D.1.3 DESCRIPTION

batbinevt computes mask weighted light curves and spectra for BAT event and detector plane histogram data (DPHs). This tool can use the weights generated by batmaskwtevt or batmaskwtimg to make binned output light curves and spectra. batbinevt can also be used to make raw detector plane images and histograms, and to rebin raw detector plane histograms in time or energy.

The input event file must normally have been processed first by batmaskwtevt for a particular source position. Thus, batbinevt will only extract weighted products for one source at a time, although it can compute many time or energy samples.

INPUT FILE TYPES

batbinevt operates on either event data or BAT DPH data. 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.

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.

OUTPUT FILE TYPES

batbinevt can output several file types. The most common will be

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

PHA  standard OGIP spectrum (type I or II; weighted=yes, outunits=RATE);

Output format is determined automatically; if there is one output spectrum then the output file is a type I spectrum; otherwise it is type II.

PHA1 standard OGIP spectrum (type I; default: weighted=yes, outunits=RATE);

PHA2 standard OGIP spectrum (type II; default: weighted=yes, outunits=RATE);
D.1. BATBINEVT

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. batbinevt can rebin an existing DPH in time or energy, or generate a new DPH from event data. 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 concatenated FITS extensions. batbinevt can either “flatten” existing DPHs into DPIs, or make new DPIs from event data. (default: weighted=no, outunits=COUNTS);

**DPITAB** detector plane image. Images are stored in rows of a FITS binary table (multiple images in one extension). (default: weighted=no, outunits=COUNTS).

**WEIGHTED or UNWEIGHTED?**

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

The default of the ‘weighted’ parameter depends on the output type.

- 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**

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 the 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 desired 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. Note that the ‘minfracexp’ parameter is always honored. If it is important to preserve all requested bins, even if they have zero exposure, then set minfracexp=0.

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.

The “matchlc” binning method will bin the input data to match an existing light curve. The light curve, which is passed in the ‘gtifile’ parameter, must be a standard OGIP light curve file with enough information to determine the time binning. If the light curve file has a GTI extension, it is ignored. Note that the ‘minfracexp’ parameter is always honored. If it is important to exactly match the template light curve time bins, even if they have zero exposure, then set minfracexp=0.

For output light curves, the TIME column may refer to the start or the center of the bin, depending on the setting of the ‘timepixr’ parameter. Output lightcurves also contain the keyword TIMEPIXR which indicates the reference point for light curve time bins. For all other output types, TIME refers to the start of the accumulation time. In some cases, a separate keyword or column named TSTOP or TIME_STOP is used to indicate the stop of the accumulation time.

**TIME SELECTION**

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. For input DPH (survey) data, the time interval of each input DPH must overlap with the total good time intervals by a large enough amount, and must never straddle more than one output bin.

The DPH overlap time required is determined by the min_dph_frac_overlap, min_dph_time_overlap and max_dph_time_nonoverlap parameters. Care should be taken when altering the overlap parameters, especially when the properties of the source and the background are not constant with time. For example, if a source becomes earth-occulted during the DPH, and a corresponding occultation GTI was input to batbinevt, then the background will have the full exposure, but the source will only have a fractional exposure, i.e. the exposure is ambiguous. For exposure purposes, batbinevt records the full exposure of all DPHs that pass the overlap tests, including any non-overlap intervals. The output GTI, however, retains the user-requested good time intervals.

**FLUX UNCERTAINTIES**

batbinevt assumes that detector counts are Poisson-distributed, and propagates the errors to the output. However, if the input is a modeled background (e.g. from batclean), then the output uncertainties will be set to zero. This behavior is governed by the HDUCLAS2 keyword in the input DPH file: modeled background maps should have HDUCLAS2 = ‘PREDICTED’.

**D.1.4 PARAMETERS**

infile [filename]

Input file name or names, containing BAT event or DPH data. This may be a comma-delimited list of names, or the name of a text file containing a list of file names, one per line, preceded by an ‘@’ character.
D.1. BATBINEVT

outfile [filename ]

Output light curve or spectrum file name.

outtype [string ]

Chooses output format of either light curve (outtype=“LC”), spectrum (outtype=“PHA”, “PHA1”, “PHA2”), detector plane histogram (outtype=“DPH”), detector plane image (outtype=“DPI”), or detector plane image table (outtype=“DPITAB”). The difference between DPI and DPITAB is that DPI output has one detector image per FITS image extension and the later has one image per FITS row in a single extension.

timedel = 1.0 [real ]

Select a time bin size, in seconds. If time binning is uniform, then a value of timedel=0.0 causes the tool to accumulate all data into a single time bin. If time binning algorithm is “snr”, then a non-zero bin size indicates the maximum bin size regardless of the signal to noise ratio. A zero value of timedel indicates no maximum bin size.

timebinalg = uniform [string ]

One of “uniform”, for uniform binning, “snr”, for constant signal to noise ratio, “gti” for time binning according to the intervals in the GTI file, “infile” to mimic the input file binning (binned input data only), or “matchlc” to match an existing light curve. Only the first character of the algorithm type needs to be specified.

energybins = “-“ [string ]

Energy bin ranges, expressed as a comma-separated list of floating point number ranges, a file name containing energy bin ranges, INFILE, FILEBINS or CALDB.

Comma-separated list Each bin in the comma-separated list is specified as EMIN1-EMAX1,EMIN2-EMAX2,... (in units of keV). Pulse heights which satisfy EMIN[n] ≤ E < EMAX[n] are considered to be in bin n. If EMIN[n] is missing, then it is assumed to be 0. If EMAX[n] is missing it is assumed to be the maximum energy. A value of “-“ indicates single all-inclusive energy bin. At least one bin must be given to make even a light curve, and bins must not overlap.

File name The name of a FITS file with an EBOUNDS extension (such as a response matrix), containing columns E_MIN and E_MAX (in keV), or an ASCII file containing a comma-separated energy bin list as described above.

FILEBINS or INFILE FILEBINS or INFILE indicates that the energy binning of the first input file is used. (this only applies when the inputs are DPHs, and the DPH must have an EBOUNDS extension).

CALDB If CALDB is specified, then the CALDB database is consulted for energy bins. A standard 4-channel and 80-channel set of binnings is chosen depending on whether the outtype is LC or PHA, respectively. By using CALDB:n, a specific n-channel binning can be selected (provided it exists in CALDB).

(gtifile = NONE) [string ]

Name of goodtime interval file or time bin file. If the time binning algorithm is “gti” or “matchlc” then this file is used to establish the time bins. Normally, this file is a standard GTI. However, for time binning algorithm “matchlc”, this file should be standard OGIP light curve. The user-supplied good times are 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.
(ecol = PI) [string ]

Column name to use for the energy value. The default is the energy computed in the “PI” column, but it is possible to choose another column name such as “PHA”.

(weighted = INDEF) [string ]

Set to boolean yes or no (or INDEF), depending on whether mask weighting should be applied to the binning operation. A value of “no” indicates that all counts should be unweighted (unity weight). A value of INDEF depends on outype. If outtype is LC or PHA, then INDEF signifies that “yes,” weighting should be applied; for output types DPH, DPI or DPITAB, then INDEF means “no” weighting should be applied.

(outunits = INDEF) [string ]

Set to RATE, COUNTS or INDEF, depending on whether the output histogram units should be “count/s”, “count”, or automatically determined (INDEF). When INDEF is specified, outunit depends on outype. If outtype is LC or PHA, then INDEF signifies “RATE;” for output types DPH, DPI or DPITAB, then INDEF means “COUNTS.” Note that not all combinations are meaningful.

(timepixr = -1.0) [real ]

Determines the reference point of each time bin for light curves only. A value of 0.0 (0.5) indicates that the TIME column refers to the start (center) of each time bin. A special value of -1 (the default) specifies that timepixr=0.0 for uniformly binned light curves and timepixr=0.5 for non-uniformly binned light curves. This parameter does not have any affect for spectra, images or DPHs.

(maskwt = NONE) [string ]

Mask weight map file. This file should contain a mask weight image for the detector plane, and if given, overrides any mask weighting found in the infile.

(tstart = INDEF) [string ]

Start time of the accumulation, in seconds, expressed in Mission Elapsed Time. Default value (INDEF) implies using the TSTART/TSTOP keywords from the input file.

(tstop = INDEF) [string ]

Stop time of the accumulation, in seconds, expressed in Mission Elapsed Time. Default value (INDEF) implies using the TSTART/TSTOP keywords from the input file.

(snrthresh = 6.0) [real ]

For the constsnr binning method, the signal to noise ratio to require for each time bin. If there is more than one energy bin selected, the threshold refers to the signal to noise ratio for the sum of all energy bins.

(detmask = “NONE”) [string ]

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.

(tcol = “TIME”) [string ]

Name of TIME column in input file.
(countscol = “DPH_COUNTS”) [string]

Name of counts column column in input file (column ‘DPH_COUNTS’ containing detector counts histogram when reading DPH data).

(xcol = “DETX”) [string]

Name of X column in input file (event data).

(ycol = “DETY”) [string]

Name of Y column in input file (event data).

(maskwtcol = “MASK_WEIGHT”) [string]

Name of MASK_WEIGHT column in input file (event data).

(ebinquant = 0.1) [real]

Default energy bin quantization, in keV, if none is specified in the input event file. Most BAT input events are quantized with 0.1 keV step sizes using the TSCALn keyword. If no such keywords are found, then ebinquant is used. This value should represent the minimum possible energy bin size of the instrument.

(delzeroes = “NO”) [boolean]

Delete time bins with zero flux? If yes, then any bin which has no counts, or all counts with zero weights, will be removed from the output. For multi-channel output data types, the time bin will only be deleted if all spatial/spectral bins contain zero. If delzeroes=“NO”, then the output may contain bins with zero flux. The error bars will be zero too for those cases, which may cause subsequent processing to fail.

(minfracexp = 0.1) [real]

The minimum fractional exposure. Time bins with smaller fractional exposure are deleted before creating the output. To preserve all time bins, even bins with zero exposure, set minfracexp=0. Bins with no exposure will have null COUNTS / RATE.

(min_dph_frac_overlap = 0.999) [real]

The minimum fractional exposure per individual DPH in the input file. The combined set of all good time intervals must fractionally overlap with a DPH by at least mindphfracoverlap, or else the DPH row is rejected. The default value of 0.999 allows for a small non-overlap as well as numerical round-off issues.

(min_dph_time_overlap = 0) [real]

The minimum time overlap, in seconds, per individual DPH in the input file. The combined set of all good time intervals must overlap with the DPH by at least mindphtimeoverlap duration, or else the DPH row is rejected. This is also equivalent to a filter on the minimum exposure time per individual DPH. The default value of 0 indicates that any number of seconds is allowed – but the fractional overlap test (mindphfracoverlap) must also pass.

(max_dph_time_nonoverlap = 0.5) [real]

The maximum non-overlap time, in seconds, allowed per individual DPH in the input file. After intersecting the combined set of all good times and the individual DPH time interval, the remaining non-overlapping time must be no more than maxdphtimeonoverlap seconds, or else the DPH row is rejected. A larger number is a more liberal acceptance criterium. The default value of 0.5 seconds, allows for a small amount of non-overlap between the good time intervals and the survey DPH.
(buffersize = 32768) [integer ]
Size of internal event buffer for processing.

(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. Setting chatter = 1 produces a basic summary of the task actions; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints debugging information.

(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.

D.1.5 EXAMPLES

MAKING LIGHT CURVES from EVENT DATA

1. Uniform binning of 0.1 seconds in the 15-195 keV energy range (light curve is weighted i.e. background subtracted)

   batbinevt infile.evt outfile.lc LC 0.1 uniform 15-195

   (“uniform” binning means uniformly sampled (=0.1s) bins)

2. Two-band light light curve for 15-50 and 50-195 keV (otherwise same as above)

   batbinevt infile.evt outfile.lc LC 0.1 uniform 15-50,50-195

   (the energy bins can be specified as either an ASCII list or a file name with the energy bins)

3. Binning by constant signal to noise (threshold = 6 sigma)

   batbinevt infile.evt outfile.lc LC 0 snr 15-195 snrthresh=6.0

   (When the “snr” time binning method is specified, a zero time binning is ignored.)

4. Binning by constant signal to noise (threshold = 6 sigma, maximum bin size of 10 seconds)

   batbinevt infile.evt outfile.lc LC 10 snr 15-195 snrthresh=6.0

   (When the “snr” time binning method is specified, a non-zero time binning indicates the maximum bin size that is imposed.)

5. Selecting the times by hand (0.1 second uniform bins, 15-195 keV)
batbinevt infile.evt outfile.lc LC 0.1 uniform 15-195 tstart=$START tstop=$STOP

(only include data from $START to $STOP)

6. Applying your own binning (0.1 second uniform bins, 15-195 keV) (must make a Good Time Interval (GTI) named mygti.gti)

batbinevt infile.evt outfile.lc LC 0.1 gti 15-195 gtifile=mygti.gti

(The “gti” time binning means that the bins are taken from the GTI file. When the time binning is not “gti” the GTI file is still useful to select data by time.)

7. Unweighted light curve (with no background subtraction)

batbinevt infile.evt outfile.lc LC 0.1 uniform 15-195 weighted=no

(The weighted parameter decides whether mask weighting is applied or not.)

8. Change output units to counts instead of counts/s.

batbinevt infile.evt outfile.lc LC 0.1 uniform 15-195 outunits=count

(The outunits parameter can be either COUNT or RATE.)

MAKING SPECTRA from EVENT DATA

9. Total spectrum, in user-specified energy bins

batbinevt infile.evt outfile.pha PHA 0 uniform 15-25,25-50,50-100,100-195

(A time bin size of 0 means that the entire data set should be accumulated into one total spectrum.

The energy bins can be specified as either an ASCII list or a file name with the energy bins. With a few bins it is easier to give an ASCII list.)

10. Total spectrum, with energy bins given by EBOUNDS extension in myebounds.fits

batbinevt infile.evt outfile.pha PHA 0 uniform myebounds.fits

(Here the energy bins are given in a file with an EBOUNDS extension.)

11. Spectrum every 100 seconds, using EBOUNDS extension

batbinevt infile.evt outfile.pha PHA 100.0 uniform myebounds.fits
12. Binning by constant signal to noise (threshold = 6 sigma)

   batbinevt infile.evt outfile.pha PHA 0 snr myebounds.fits snrthresh=6.0

   (The threshold applies to total spectrum, not any one bin.)

13. Selecting the times by hand

   batbinevt infile.evt outfile.pha PHA 0 uniform myebounds.fits tstart=$START tstop=$STOP

   (Only includes data from $START to $STOP)

14. Applying your own binning, say from Bayesian Blocks analysis (must make a Good Time
    Interval (GTI) named mygti.gti)

   batbinevt infile.evt outfile.pha PHA 0 gti myebounds.fits gtifile=mygti.gti

   (The “gti” time binning means that the bins are taken from the GTI file. When the time
   binning is not “gti” the GTI file is still useful to select data by time.)

15. Unweighted spectrum (with no background subtraction)

   batbinevt infile.evt outfile.pha PHA 0 uniform myebounds.fits weighted=no

   (The weighted parameter decides whether mask weighting is applied or not.)

16. Change output units to counts instead of counts/s.

   batbinevt infile.evt outfile.pha PHA 0 uniform myebounds.fits outunits=count

   (The outunits parameter can be either COUNT or RATE.)

**MAKING LIGHT CURVES and SPECTRA from DPHs**

17. Accumulate mask weighted light curve from DPH (must have generated mask weight image
    maskwt.img for source position of interest)

   batbinevt infile.dph outfile.lc LC 0 infile 14.0-194.9 maskwt=maskwt.img

   (Note that energy bins must be exact to within 0.1 keV. The “infile” for the time binning
   algorithm indicates that the input file binning should be replicated in the output.)

18. Accumulate total mask weighted spectrum from DPH

   batbinevt infile.dph outfile.pha PHA 0 uniform infile.dph maskwt=maskwt.img
D.1. BATBINEVT

(Note that EBOUNDS extension from infile is used to generate energy bin edges.)

19. Accumulate total mask weighted spectrum with user-designed bins

```
batbinevt infile.dph outfile.pha PHA 0 gti infile.dph maskwt=maskwt.img
```

20. Accumulate raw light curve from DPH (no mask weighting)

```
batbinevt infile.dph outfile.lc LC 0 infile 14.0-194.9 weighted=no
```

21. Accumulate raw spectrum from DPH (no mask weighting)

```
batbinevt infile.dph outfile.pha PHA 0 uniform infile.dph weighted=no
```

MAKING DPHs and DPIs from EVENT DATA

22. Make total DPH from event data

```
batbinevt infile.evt outfile.dph DPH 0 uniform myebounds.fits
```

(myebounds.fits should contain the desired energy binning. A time bin size of 0 indicates that
the total DPH should be constructed.)

23. Make total DPI from event data (15-195 keV)

```
batbinevt infile.evt outfile.dpi DPI 0 uniform 15-195
```

24. Make many DPIs from event data (100 second binning)

```
batbinevt infile.evt outfile.dpi DPI 100 uniform 15-195
```

(Output file is stored as one DPI per image extension.) One can also use:

```
batbinevt infile.evt outfile.dpi DPITAB 100 uniform 15-195
```

(which will put many DPIs into one extension as a table.)

25. Making DPHs from event data is accomplished with this command

```
batbinevt infile.evt outfile.dph DPH 0 uniform myebounds.fits
```

(making more than one or two DPHs in this way is not recommended, since it requires a large
amount of computer memory).
**APPENDIX D. BAT SOFTWARE TOOL REFERENCE**

**MANIPULATING and FLATTENING DPHs**

`batbinevt` can be used to “flatten” detector plane histograms in various ways.

26. Make total DPH (add all DPH rows into a single DPH row).

```
batbinevt infile.dph outfile.dph DPH 0 uniform infile.dph
```

(The energy binning of `infile.dph` means that the energy binning of the input file is preserved in the output).

27. Make total DPI (add all DPH rows into a single detector image).

```
batbinevt infile.dph outfile.dpi DPI 0 uniform 14.0-194.9
```

(Only energies from 14.0-194.9 are kept in the detector image.)

28. Make one DPI for each DPH (flatten by energy)

```
batbinevt infile.dph outfile.dpi DPITAB 0 infile 14.0-194.9
```

(“`infile`” time binning indicates to preserve input file time bins. The output file is a table of DPIs, one per FITS row). This command creates one DPI per image extension.

```
batbinevt infile.dph outfile.dpi DPI 0 infile 14.0-194.9
```

(either is compatible with the `batfftimage` imaging software)

29. Make new DPHs by collapsing some energy bins. This command makes approximate bins from 15-25, 25-50, 50-100 and 100-200 keV.

```
batbinevt infile.dph outfile.dph DPH 0 infile 14-26,26-51.1,51.1-101.2,101.2-194.9
```

(Note energy bins must agree to within 0.1 keV of input binning)

It is not advisable to make more than one full-sized (>80 energy channels) DPH because the memory storage requirements are prohibitive.

**D.1.6 SEE ALSO**

`extractor`, `batmaskwtevt`, `batmaskwting`

**D.1.7 LAST MODIFIED**

Nov 2006
D.2 BATCELLEDTECT

D.2.1 NAME

batcelldetect - perform source detection using the sliding cell method

D.2.2 USAGE

batcelldetect infile outfile snrthresh

D.2.3 DESCRIPTION

batcelldetect performs source detection on a sky image. The sliding cell method is used to locate regions of the image which are significantly different from the background.

This tool is more 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 “nullborder” and “bkgpcodethresh” parameters can also be used to exclude false positives near the edge of the image.

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 point spread function to regions of the image where sources are detected. For new sources, 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 output flux column is either COUNTS or RATE, depending on the input image units. The tool can optionally output the background map, the background fluctuation map, and a significance map.

IMPORTANT NOTE: if the keepbadsources parameter is ‘YES’, then the output catalog may contain some sources which failed processing for one reason or another. Information on the success or failure on a source-by-source basis is recorded in the DETECT_STATUS column.

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 during
the PSF-fitting stage for intensity only (unless posfit=NO). Additional detected sources are also written to the output catalog.

This task can operate on multiple images, as produced by batfftimage. If the images vary only in their energy range, and are otherwise identical in their coordinate systems and exposures, then the ‘vectorflux’ parameter can be set to YES. When vectorflux=YES, each source has only one catalog row, and multiple fluxes are recorded in a vector table cell. When vectorflux=NO, a source will appear multiple times in the catalog, with a single flux in each entry. Note that if the input images do not match exactly their coordinate system or exposures, then you must specify vectorflux=NO.

To avoid spurious detections at the edge of the field of view, a partial coding threshold can be specified (see ‘pcodefile’ and ‘pcodethresh’). If a partial coding map is provided, then by default sources must be at least 1% coded.

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.

**Fitting For Position and Flux**

batcelldetect has several options to fit the positions and fluxes of sources. Newly detected and previously known sources are treated differently. By default, the positions of newly detected sources are allowed to vary and the those of previously known sources are held fixed. This behavior depends on the input parameters and the input catalog.

The following behaviors are possible:

- **srcfit=NO** Perform source detection only. The positions and fluxes for all sources are not fit. For new sources, the positions are taken from the source detection stage. For previously known sources, the values are transferred from the input catalog completely unchanged.

- **srcfit=YES; posfit=NO** (DEFAULT) Hold the position of known sources fixed, but fit for the positions of new sources. The fluxes of all sources are fitted.

- **srcfit=YES; posfit=YES; posfitwindow=0.0** Fit for the positions of all sources. The positions of known sources are allowed to vary within a window which is 3 times the ERR_RAD value from the input catalog. Thus, if ERR_RAD = 0, the position is held fixed. Special cases: if an individual value of ERR_RAD is null, the position of that source is allowed to vary. If the ERR_RAD column is missing entirely, the positions are assumed to be fixed. The fluxes of all sources are fitted.

- **srcfit=YES; posfit=YES; posfitwindow > 0** As above, except the positions of known sources are allowed to vary by ± 3*ERR_RAD or ± posfitwindow, whichever is larger.

**D.2.4 CATALOG OUTPUT**

The output catalog contains the following columns. Other input columns are copied to the output unchanged. For new sources entries, the row is filled with null value before writing batcelldetect-derived quantities.

Note that while the input catalog is only required to have the RA_OBJ and DEC_OBJ columns, they must have the units listed below. Any input catalog columns that match the list below, must also have the specified units.
The data type should be either “D” (double) or “J” (integer); the NAME column can be up to a 30-character string. The word “Vector” indicates which output catalog columns will be a vector if vectflux=YES and multiple images are analyzed.

**TIME**  Start time of image (Mission Elapsed Time). Type=D, Unit=s.

**TIME_STOP**  Stop time of image (Mission Elapsed Time). Type=D, Unit=s.

**EXPOSURE**  EXPOSURE time of image. Type=D, Unit=s.

**CATNUM**  Catalog identification number (copied from input catalog). Type=J.

**NAME**  Source name (copied from input catalog). Type=30A (string).

**IMX**  IMX Tangent plane coordinate (instrument X). Type=D.

**IMY**  IMY Tangent plane coordinate (instrument Y). Type=D.

**IMX_ERR**  IMX formal uncertainty. Type=D.

**IMY_ERR**  IMY formal uncertainty. Type=D.

**IMXPIX**  IMX pixel coordinate (instrument X). Type=D.

**IMYPix**  IMY pixel coordinate (instrument Y). Type=D.

**RA_OBJ / GLON_OBJ**  Source celestial longitude (equatorial or galactic). Type=D, Unit=deg.

**DEC_OBJ / GLAT_OBJ**  Source celestial latitude (equatorial or galactic). Type=D, Unit=deg.

**RA_OBJ_ERR / GLON_OBJ_ERR**  Source celestial longitude formal uncertainty (1 sigma). Type=D, Unit=deg.

**DEC_OBJ_ERR / GLAT_OBJ_ERR**  Source celestial latitude formal uncertainty (1 sigma). Type=D, Unit=deg.

**ERR_RAD**  Estimated error radius (formal; 1 sigma). Type=D, Unit=deg.

**IMX_FWHM**  Instrumental PSF X full width at half-max (tangent plane coords). Type=D.

**IMY_FWHM**  Instrumental PSF Y full width at half-max (tangent plane coords). Type=D.

**IMX_FWHM_ERR**  Instrumental PSF X FWHM formal uncertainty (1 sigma). Type=D.

**IMY_FWHM_ERR**  Instrumental PSF Y FWHM formal uncertainty (1 sigma). Type=D.

**THETA**  Source angle from instrumental boresight. Type=D, Unit=deg.

**PHI**  Source instrumental longitude. Type=D, Unit=deg.

**GRMCLON**  BAT simulator longitude. Type=D, Unit=deg.

**GRMCLAT**  BAT simulator latitude. Type=D, Unit=deg.

**RATE**  Source count rate (units from image)

  RATE is used if the image has rate-like units. Type=D, Vector.

**RATE_ERR**  Source count rate formal uncertainty (1 sigma; units from image). Type=D, Vector.
COUNTS Source counts (units from image)
    COUNTS is used if the image does not have rate-like units. Type=D, Vector.

COUNTS_ERR Source counts formal uncertainty (1 sigma; units from image). Type=D, Vector.

BKG Local background intensity (units from image). Type=D, Vector.

BKG_ERR Local background intensity formal uncertainty (1 sigma; units from image). Type=D, Vector.

BKG_VAR Local background standard deviation (units from image). Type=D, Vector.

SNR Source signal to noise ratio (defined as RATE/BKG_VAR). Type=D, Vector.

NPIXSOU Number of valid pixels in the source window. Type=J, Unit=pixel.

NPIXBKG Number of valid pixels in the background window. Type=J, Unit=pixel.

DETECT_METHOD Method used for source detection:
    1=Cell detection
    2=PSF fit.
    Type=J.

CHI2 Chi^2 value in PSF fit. Type=D, Vector.

DOF Number of degrees of freedom in PSF fit. Type=D, Vector.

CHI2 NU Reduced Chi^2 value in PSF fit. Type=D, Vector.

DETECT_STATUS Source detection status code; entries with non-zero DETECT_STATUS are deleted when keepbadsources='NO'; One of:
    0=OK;
    -1=Too many pixels detected;
    -2=No pixels detected;
    -3=Source touches border of image;
    -4=Source error radius did not fit in source window.
    -5=PSF fit failed (source was likely out of bounds)
    Type=J.

CONSTRAINT FLAG PSF fitting constraint flag; Describes which parameters of the PSF model were fixed; Sum of:
    1=IMX fixed;
    2=IMY fixed;
    4=COUNTS/RATE fixed;
    8=PSF width fixed.
    Type=J.

SRC WINDOW RAD Source window radius . Type=J, Unit=pixel.

BKG WINDOW RAD Background window radius . Type=J, Unit=pixel.

NGOODPIX Number of good detectors (copied from input image). Type=J, Unit=pixel.

BAT ZOBJ Near-field source height (ground testing only; coped from input image). Type=D, Unit=cm.

PCODEFR Partial coding fraction of source. Type=D.
D.2. BATCELLEDTECT

\textbf{x\_PER\_IMX} Derivative of \textit{x} position with respect to IMX. Type=D, Unit=deg.

\textbf{x\_PER\_IMY} Derivative of \textit{x} position with respect to IMY. Type=D, Unit=deg.

\textbf{keyword} Keywords copied from input image via the ‘keepkeywords’ parameter.

\subsection*{D.2.5 DISTORTION CORRECTION}

Batcelldetect can correct for systematic non-linear centroid shifts in the image. It is assumed that all catalog celestial positions (input and output) are in “true” undistorted coordinates. All distortion corrections occur at the pixel level within batcelldetect. The distortion correction occurs if the ‘distfile’ parameter points to a correction map.

\subsection*{D.2.6 PARAMETERS}

\textbf{infile} [filename]

Input file name containing one or more sky maps. Each sky map should be stored as one image extension. (Avoid using the CFITSIO syntax to select image extensions; individual images extensions can be selected from the file using the ‘rows’ parameter).

\textbf{outfile} [filename]

Output source list.

\textbf{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 considered a detection. Here 1 sigma is the standard deviation of the background. Note: for performance reasons users should choose \textbf{snrthresh} > 3.5. Also, if \textbf{nadjpix} > 1, then the effective threshold will be higher than \textbf{snrthresh}.

\textbf{(incatalog=NONE)} [string]

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 (however see above about required units). A value of “NONE” indicates that no a priori catalog should be used. Specifying this incatalog=NONE by itself does not prevent the cell-detection based source detection algorithm from being applied.

\textbf{(pcodefile=NONE)} [string]

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 pcode file is “NONE”, then the entire image is searched.

\textbf{(pcodethresh=0.01)} [real]

Partial coding threshold for source detection. This threshold is applied to the map specified in pcodefile. Any partial coding value greater than pcodethresh is considered active for the purposes of source detection.
Partial coding threshold for background analysis. This threshold is applied to the map specified in pcodefile. Any partial coding value greater than bkgpcodethresh is considered active for the purposes of measuring the background and its variance. A value of -1 indicates that bkgpcodethresh should be set to pcodethresh. The “source” and “background” thresholds are kept separately so that background analysis can be done over a wider area than the source acceptance region.

Constrain newly detected peaks to be positive. If NO, then significant positive or negative deviations will be searched for. Known cataloged sources are permitted to have a negative fitted flux regardless of pospeaks.

If YES, then output flux values for a given source as a vector, one element for each input image. If NO, then output one catalog rows for each source in each input image. Note, for vectorflux=YES, the input images must all match their coordinate systems, exposure, and instrument configuration exactly.

An index list of images to operate on. This is a standard comma-separated list of image ranges, starting with image number 1. Image number 1 corresponds to the primary extension of the input FITS file, 2 to the next extension, and so on. A range of “-” indicates all images should be processed. Thus, this parameter really selects image extension numbers to operate on rather than binary table rows.

Number of iterations of source detection to perform. niter=0 is allowed; in that case the sources in the input catalog (if any) which are in the field of view are transferred to the output catalog.

For the analysis of multiple images. If YES, then carry over the sources detected in the analysis of earlier images to the analysis of later images. Positions of previously detected new sources are held fixed as described above. If NO, then the analysis of each image begins from scratch, excluding any previously detected sources. Note that carryover=NO and vectorflux=YES are contradictory.

Minimum number of adjacent pixels required for source detection. This serves to decrease the number of false detections. This quantity is somewhat dependent on the image oversampling used.

Allow sources which fall on the border of the image? If no, then sources whose “source window” overlaps with the edge of the image are ignored. The edge of the image is defined by the partial coding threshold. These sources may still appear in the output catalog with a DETECT_STATUS value of -3.

Background window type, one of “circle” or “square”.
D.2. BATCELLDETECT

(bkgradius=30) [integer]
Background window radius in pixels. For a window of “circle”, the bkgradius parameter is
the radius of the circle. For a “square” window, the bkgradius is the half-width of the square.
Actual diameter of circle or square is 2*bkgradius+1.

(regionfile=NONE) [string]
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.

(srcradius=6) [integer]
Source radius in pixels. If 0, then the full background window is used to estimate the back-
ground and its standard deviation. Actual diameter of circle or square is 2*srcradius+1.

(srcdetect=YES) [boolean]
If srcdetect is set to “YES”, then batcelldetect will attempt to detect new sources. If “NO”,
then source detection is disabled, but source fitting of previously cataloged sources may still
be performed (depending on the value of ‘srcfit’).

(srcfit=YES) [boolean]
If srcfit is set to “YES”, then batcelldetect will fit newly detected and previously cataloged
sources (if any) as described above. If “NO”, then previously determined positions and
intensities are output. See “Fitting For Position and Flux” above.

(posfit=NO) [boolean]
Posfit specifies whether batcelldetect fits the position of already known sources. See “Fitting
For Position and Flux” above.

(bkgfit=NO) [boolean]
Fit the background for each source individually? If yes, then during the PSF fitting stage, each
source has an independent background level. If no, a fixed background is subtracted. NOTE,
that bkgfit=YES psfshape=GAUSSIAN may be a bad combination, since the gaussian shape
is not a particularly good match to the Swift/BAT point spread function.

(psfshape=“PYRAMID”) [string]
Shape of the point spread function to use. It should be one of “GAUSSIAN”, “PYRAMID”
or “TRUNCONE”. “GAUSSIAN” is a two-dimensional gaussian function; “PYRAMID” is
a two-dimensional pyramidal frustum with a square base (which more closely matches the
Swift/BAT point spread function in a single image); “TRUNCONE” is a truncated cone
(which most closely resembles the BAT PSF after mosaicking many images).

(psffwhm=“0.37413”) [string]
Full-width at half maximum of the point spread function, in units of degrees. By default,
this is held fixed, but if “.FIT” is appended, then the width is fitted. The default value listed
above applies to the BAT point spread function.

(psftopwidth=“0.04584”) [string]
For the pyramidal point spread function, the full width of the top of the truncated pyramid,
in units of degrees. By default, this is held fixed, but if “.FIT” is appended, then the width
is fitted. The default value listed above applies to the BAT point spread function.
(newsrcname="UNKNOWN") [string ]

Name to be used for newly detected sources. This can be a C format string, whose interpretation depends on the value of the newsrcind parameter. If newsrcind is greater than or equal to zero, then newsrcname may contain a single "%d"-style C format code. Each new source is given a unique source number, starting with index newsrcind. If newsrcind is -1, then newsrcname may contain two "%f" C format codes for the RA and DEC position, in degrees.

(newsrcind=1) [integer ]

Starting index number to be used to label new sources (see newsrcname). If -1, then sources are labeled by their position on the sky.

(signifmap=NONE) [string ]

Optional output file name for the significance map.

(bkgmap=NONE) [string ]

Optional output file name for the mean background map.

(bkgvarmap=NONE) [string ]

Optional output file name for the background fluctuation map.

(inbkgmap=NONE) [string ]

Optional background map which overrides the internally-calculated background. The map or maps must have exactly the same structure and number of images as the input image file. If inbkgmap=ZERO, then the background is explicitly set to zero in the source detection stage.

(inbkgvarmap=NONE) [string ]

Optional background variation map which overrides the internally-calculated variation map. The map or maps must have exactly the same structure and number of images as the input image file.

(distfile=CALDB) [string ]

The ‘distfile’ parameter should point to a FITS file containing an MxNx2 image (or CALDB to use the default instrument correction in the calibration database). The two planes of the image are the non-linear distortion of the BAT “plate scale” as a function of position in the sky image. The values are offsets in tangent plane coordinates (IMX,IMY), and therefore unitless. The first plane of the image cube is the IMX offset, and the second plane is the IMY offset. The sense of the offset is (TRUE-APPARENT), i.e. for a measured position in tangent plane coordinates, the offset values should be added to arrive at the true position in tangent plane coordinates. The images are low-resolution versions of the BAT field of view in instrumental sky coordinates, and are meant to be interpolated to the desired sampling. The WCS keywords must specify the image coordinate systems.

(keepbits = “ALL”) [string ]

Integer number of bits of precision to keep in the output images. A value of ALL means keep all bits. A positive value indicates that ‘keepbits’ bits of precision should be preserved, and the other bits should be set to zero. Fewer kept bits means the output images are more compressible, at the expense of lost precision. For background and variance maps, a straight number of bits are kept (thus preserving the same relative precision across the map). For significance maps, ‘keepbits’ prescribes the number of noise bits to keep; more bits may be
kept if a pixel is significantly above the noise (thus preserving the same noise quantization error across the map).

\textbf{(npixthresh=20) [integer ]}

Minimum number of pixels in the background window to enable source detection. Setting this parameter to a high number will prevent detections based on only a few background pixels.

\textbf{(posfitwindow=0) [real ]}

The position ‘window’ radius, in degrees, used for all position fits, including known sources. The window is the allowed region over which the centroid position is allowed to vary during the position fit, regardless of the error radius specified for the source. Typically, this parameter will only be used if the attitude solution is suspected to be incorrect. A value of zero indicates that the window size is determined by the error radius specified in the catalog, if any. This parameter is basically an override for the ERR_RAD column. See “Fitting For Position and Flux” above.

\textbf{(possyserr=0) [real ]}

Systematic position error, in degrees, to be added in quadrature to each position error radius reported in the output catalog. This parameter allows one to specify a minimum systematic position error for every reported detection (which is separate from posfitwindow, which controls the position determination process itself).

\textbf{(keepkeywords=“*APP,CLEANLEV”) [string ]}

A comma separated list of keywords or keyword wildcard patterns. The matching keywords are transferred to the output catalog, under a column of the same name. This is a way to automatically keep track of corrections that have been applied to the input image. The CFITSIO wildcard patterns are: ‘?’ will match any single character at that position in the keyword name; ‘*’ will match any length (including zero) string of characters; and ‘#’ will match any consecutive string of decimal digits (0 - 9).

\textbf{(hduclasses=“-VIGNETTING,-BKG_VARIANCE,-BKG_STDDEV,-SIGNIFICANCE”) [string ]}

List of comma-separated image type filters, used to accept or reject images based on their HDUCLASn keywords. This is useful for files that contain different kinds of image extensions. A minus sign “-” indicates exclusion. Each image is checked for each of the HDUCLASn keywords. If any of the keyword values matches an excluded list member, then that image is rejected for analysis. List members without a minus sign must be present. A value of NONE indicates no filtering (i.e. all images are accepted).

\textbf{(sortcolumns=“-KNOWN,-SNR/UNKNOWN,RA_OBJ”) [string ]}

Controls the sort order of the table displayed on the console. Note that this does not affect the FITS table, which can be sorted using ftsort or fsort. The string should be a comma-separated list of sort keys, or “NONE” for arbitrary printed order. The first key listed is the primary sort key, the next is the secondary and so on. By default, values are printed from lowest to highest, but a preceding ‘-’ sign indicates highest to lowest. The following sort keys are available: NAME; RA_OBJ; DEC_OBJ; SNR; RATE; COUNTS; FLUX (all three of these keys are synonymous); KNOWN (if the sources appears in the input catalog or not); IMX; IMY; PCODEFR; CATNUM; THETA; PHI; SNR/UNKNOWN. Most sort keys are identical to their corresponding catalog column, described above. “SNR/UNKNOWN” is a special sort key, indicating to sort only previously unknown columns by SNR. The default
sortcolumns indicate to arrange the table by printing the known sources first, sorted by RA, and then the unknown sources, arranged in descending order by signal to noise ratio.

(keepbadsources = NO) [boolean ]

If yes, then individual sources are kept in the output catalog, even if DETECT_STATUS is non-zero, indicating a failure of some kind. If no, then any output catalog entry with DETECT_STATUS non-zero is removed.

(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. Setting chatter = 1 produces a basic summary, including a list of detected sources; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints debugging information.

(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.

D.2.7 EXAMPLES

1. Performs source detection on skyimage.img, finding those sources which exceed the background by 6 sigma. The output is written to srclist.fits. The default circular background window is used.

   batcelldetect skyimage.img srclist.fits 6.0

2. Same as example 1, but use a square background window of half-width 10 pixels.

   batcelldetect skyimage.img srclist.fits 6.0 bkgwindowtype=square bkgradius=10

3. Same as example 1, but outputs the significance image to signif.img as well.

   batcelldetect skyimage.img srclist.fits 6.0 signifmap=signif.img

D.2.8 SEE ALSO

batfftimage

D.2.9 LAST MODIFIED

Sep 2006
D.3 BATCLEAN

D.3.1 NAME

batclean - read in a DPI and clean out a fit to the diffuse background

D.3.2 USAGE

batclean infile outfile incatalog [detmask] [options]

D.3.3 DESCRIPTION

This tool fits a simple background or source plus background model to a detector plane image. The code contains a default background model which is described below. The user can also specify one or more background models to be fit using the bkgmodel parameter. The output file can be either the cleaned focal plane (default), the fit to the detector plane or the original input file.

The tool can also, optionally, balance the detector plane either before or after or instead of cleaning. See below under the “balance” parameter for more details.

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.

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: constant,

• 10 = Right side detectors: proportional to X,

• 11 = Detectors on front (-Y) side of sandwiches: constant,

• 12 = Front side detectors: proportional to Y,

• 13 = Detectors on back (+Y) side of sandwiches: constant,

• 14 = Back side detectors: proportional to Y,

The code creates a background exposure map based on the model above (which can be optionally output using the `backexp` parameter). In addition the user can supply his own background model to be fit in addition to or in place of the standard background model.

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. The code writes keywords giving the fit parameter for each background parameter (simple plus user supplied) and source parameter that is fit. These keywords are BGPARnn for background and SRPARnn for sources. The number nn goes from 0 to the total number of parameters, starting with the hard-coded background parameters.

D.3.4 PARAMETERS

infile [filename ]

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). If there are multiple BAT_DPI image extensions then all are cleaned and multiple cleaned images are written to the output file.

outfile [filename ]

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).

incatalog [filename ]

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,” “CATNUM,” (or “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”). The SRPARnn keywords giving the fit parameter for each source contain the source name in the comments field. Allowed extension names for `incatalog` are “BAT_CATALOG” (default of `batcelldetect`), “TRIG_SOURCE” (BAT on-board catalogs) and “INPUT_CATALOG.”
(aperture) [filename ]
Aperture file. The default is the aperture file in the CALDB area. This parameter may be omitted or set to “NONE” if only background cleaning is required (i.e., the srcclean parameter set to “NO”). If several aperture types are available in the CALDB, the user can specify CALDB:apertype. Valid values of apertype are currently: FLUX or DETECTION, for apertures optimized for flux reproducibility or detection sensitivity, respectively.

(detmask) = “NONE” [filename ]
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) = “NONE” [filename ]
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.

(wtmapout) = “NONE” [filename ]
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) = “NONE” [filename ]
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.

(bat_z) = 0 [real ]
This is the z-component of the distance to the source in BAT,Z coordinates (the vertical distance of the source from the origin of the BAT coordinate system). This parameter is ignored if srcclean = “NO.” If the incatalog contains a BAT,Z keyword, then the value of the keyword is used and the value of the bat_z parameter is not used. Note that the default value of 0 corresponds to a source at infinity.

(bkgmodel) = “simple” [string ]
This optional parameter indicates the background model to be fit. Currently only two models are hard coded: the simple model which is the 14 parameter fit described above and an even simpler test model which was used in testing the fit procedures. As more sophisticated background models are developed, they will be incorporated into the code.
• The user can also use this parameter to supply a file or list of files containing user-developed background models to be fit. This can be input as either a comma-separated list of model files or a text file (preceded by the ‘@’ symbol) containing a list of input model files, one per line. These model files must be image files in the same format as a BAT DPI (286 columns by 173 rows). There are no required keywords. If a model file is not found or is not an image file, the code will ignore it. The comment fields of the BGPARnn keywords contain the model names. If an optional MODNAME is supplied, then its value becomes the model name in the comment field. If this keyword is not present, then the model name is the name of the file (not including the path). The default behavior of the code is to fit the simple model in addition to any user supplied models. The user does not need to also include the string “simple” in the bkgmodel parameter list. If, however, the user wishes to fit ONLY the user-supplied models and not the simple model, then he can request this by adding the string “useronly” to the bkgmodel parameter list. See examples below. Finally, it is possible not to fit any model at all, but simply to balance the array (using the balance parameter below. To do this set bkgmodel=none. Note that if the user selects outversion=“fit” or “bkgfit”, then no balancing is performed regardless of the value of the balance parameter.

(srcclean) = “YES” [boolean ]
Determine whether sources are cleaned from the input DPI or only background. If this parameter is “YES,” then an incatalog and aperture file must be supplied.

(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. If no sources have a signal-to-noise ratio above the cleansnr value, then only the background will be cleaned.

(snrcol = “SNR”) [string ]
Name of catalog column to be used with ‘cleansnr’ for signal to noise thresholding. By default it is “SNR”, but any column can be used to trigger cleaning.

(ignore) = “UNKNOWN” [string ]
Unless this parameter is set to “NONE” or “none,” the cleaning code will not clean any source in incatalog whose NAME matches the ignore string. The user can either specify a particular source name, the strings “IGNORE” or “NONE”, or the name of a file (preceded by “@”). If a file name is supplied, then the file must contain a list of sources to be ignored, once source per line. Note that source names to be ignored must match exactly (including spaces, capitalization, etc.) the names in the catalog.

(balance) = “none” [string ]
Unless this parameter is set to “NONE” or “none,” the cleaning code will, after cleaning, also balance the entire array or subsets of the array as specified below. Balancing means subtracting the mean for a set of detectors from the values for that set of detectors. In other words, it sets the average for the detectors to zero. If a detector mask (detmask) is given, then only non-masked detectors are included in the balancing.

• The balancing that is requested is given by supplied a comma-separated list of any or all of the following values. The balance is performed in the order in which the values are listed. Balancing is not commutative, so order is significant. Allowed balance values:
  – none: No balancing.
D.3. BATCLEAN

- All: The entire array is balanced (1 balancing of 32768 detectors).
- Module: Each of the 256 detector modules is separately balanced.
- InOut: Each module has its inner detectors balanced and in separate operations, its outer detectors balanced.
- Edges: Each module has only its outer (or edge) detectors balanced.
- LongEdges: There are 32 balancings of the long edges of all modules in a row across the array.
- ShortEdges: There are 32 balancings of the short edges of all modules in a column across the array.
- flight: Reproduce the flight code balancing, by doing first the Edges, then the InOut balancing.

There is further illumination of the balancing in the comments in the code.

(balfirst) = “no” [boolean]

Normally, balancing is done after cleaning. However, by setting this parameter to “yes” then balancing can be done before cleaning.

(maskfit) = “no” [boolean]

If this parameter is set to “YES” and outversion=“fit” then the output file contains the fit parameters for all detectors even those masked using a detmask. The default behavior is that the fit is output only for unmasked detectors. Detectors will still masked for cleaning and balancing when this option is selected.

(leadedge) = “NO” [boolean]

If this parameter is set to “YES,” then the leading edge detectors (those on the two edges of detector module sides facing toward the source) are fit separately from the rest of the detectors. Since the current forward projection code does not correctly handle these edge detectors, setting this parameter does not significantly affect the overall clean.

(eff_edge) = 0.2 [float]

The depth of edge (cm) to consider effective. Range from 0.03 to 0.2

(corrections = “none”) [string]

Comma-separated list of corrections to apply to the mask weighting, or “none” if no corrections are to be applied. Note that corrections are only applied to the forward projection for sources. This parameter has no effect if only the background is to be cleaned.

- Possible correction types are: “flatfield” (basic flat fielding correction accounting for both cosine-type and r-squared-type effects); “nearfield” (for ground testing results where the source is at a finite distance – requires bat.z != 0); “cosine” (cosine effects of off-axis illumination only); “rsquare” (r-squared effects only); “opaque” (activates a new algorithm for calculating the mask transmission function); “sides” (effects of sides of detectors XXX not implemented).

(outversion) = “cleaned” [string]

This optional parameter indicates what detector plane image is written to the outfile. The default is to write the cleaned DPI (input minus fit). Other options are:
• original: Write the input DPI filtered by the detector mask.
• fit: Write the fit DPI.
• bkgcleaned: Write the background cleaned DPI (input minus fit to background only).
• bkgfit: Write the fit to the background only.

• Choice of \textit{outversion} parameter dictates what value is written to the HDUCLAS2 keyword. Possible values are:
  – original: HDUCLAS2 is unchanged
  – cleaned or bkgcleaned: HDUCLAS2 = RESIDUAL
  – fit or bkgfit: HDUCLAS2 = PREDICTED

\textbf{(clobber = NO)} \textbf{[boolean ]}

If the output file already exists, then setting “clobber = yes” will cause it to be overwritten.

\textbf{(chatter = 1)} \textbf{[integer, 0 - 5 ]}

Controls the amount of informative text written to standard output. Setting chatter = 4 or higher will produce detailed diagnostic output; chatter = 2 prints out a basic diagnostic message. The default is to produce no printed output.

\textbf{(history = NO)} \textbf{[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 ftcopy task parameters that were used to produce the output file.

\subsection*{D.3.5 EXAMPLES}

Note that when commands are issued on the Unix command line, strings containing special characters such as `[' or `]' must be enclosed in single or double quotes. \textit{The following examples illustrate ways in which batclean can be used.}

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

   \texttt{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.

   \texttt{batclean input.dpi fit.dpi catalog.tbl detmask=bat.mask outversion="fit" corrections="flatfield"}

3. Fit two user-supplied background models in addition to the hard-coded model.
batclean input.dpi cleaned.dpi detmask=bat.mask bkgmodel=model1.dpi,model2.dpi

4. Same example, except the user gives the two model files as lines in the text file “models.lst.”

batclean input.dpi cleaned.dpi detmask=bat.mask bkgmodel=@models.lst

5. Balancing is performed after cleaning, applying the Module and InOut balancings.

batclean input.dpi cleaned.dpi detmask=bat.mask balance=Module,InOut

D.3.6 BUGS

D.3.7 SEE ALSO
batcelldetect, batmaskwtimg

D.3.8 REFERENCES

- Based on Netlib code [http://www.netlib.org/c/numcomp-free-c](http://www.netlib.org/c/numcomp-free-c)

Name : svd.c.Z (8704 bytes)
Where : in pub/C-numanal on ftp.usc.edu
Description : SVD based on pascal from J. C. Nash book
Author : Bryant Marks (bryant@sioux.stanford.edu)
         Brian Collett (bcollett@hamilton.edu)
Version : 14 April 1993


D.3.9 LAST MODIFIED

Aug 2006
APPENDIX D. BAT SOFTWARE TOOL REFERENCE

D.4 BATDETMASK

D.4.1 NAME

batdetmask - retrieve BAT gross detector quality map from CALDB

D.4.2 USAGE

batdetmask date outfile

D.4.3 DESCRIPTION

batdetmask retrieves a gross detector quality map (bad pixel map) from CALDB for the time of a given observation. This quality map is meant to reflect any operational issues that affected the BAT science data quality which are not easily extractable from the filter file or other telemetry. The user is not excused from doing other routine quality filtering, for example for SAA or noisy detectors. Thus, this task is most useful for doing gross filtering of data at the observation level to remove detectors with known bad data.

The master quality maps are stored in CALDB. This task is merely an interface to retrieve the correct map for the right time. The user can specify the time explicitly, or an FITS file which contains the time in a header keyword. The retrieved quality map can also be combined with an existing user quality map using the detmask parameter.

The result is a detector quality map suitable for input to any BAT task which accepts the detmask parameter.

D.4.4 PARAMETERS

date [string ]

The date of observation. This can be either the date of the exact form "YYYY-MM-mm:ss"; OR the Swift mission elapsed time in seconds as a real number; OR a file name which contains the observation time in a header keyword.

outfile [filename ]

Name of the output detector quality map file.

(keyword = "TSTART") [string ]

If the “date” parameter specifies a file, then the file is searched for a header keyword of this name. The keyword value may be of either of the time formats listed for the “date” parameter.

(detmask = "NONE") [string ]

The name of a user quality map which should be combined with the one from CALDB, or NONE if only the CALDB map is desired.
Upon return, this parameter is set to the global quality map and extension found from CALDB which matches the time of the observation.

Upon return, this parameter is set to the detector mask which was selected (including file name and extension), based on the time of the observation.

If the output file already exists, then setting “clobber = yes” will cause it to be overwritten.

Controls the amount of informative text written to standard output.

1. Retrieve the detector quality map for June 5, 2005,

   batdetmask 2005-06-05T00:00:00 master.detmask

2. Retrieve the detector quality map based on the TSTART keyword in file myfile.dpi.

   batdetmask myfile.dpi master.detmask keyword=TSTART

SEE ALSO

batglobalgti

LAST MODIFIED

Nov 2006
D.5 BATDRMGEN

D.5.1 NAME

batdrmgen - computes BAT detector response matrix (RSP) for a known source position

D.5.2 USAGE

batdrmgen infilename outfilename hkfile

D.5.3 DESCRIPTION

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 \( N \times M \) 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
D.5. BATDRMGEN

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, “MEAN”, accesses calibration parameters that are averaged over the entire BAT FOV. The alternative method is “TABLE”, in which batdrmgen accesses a 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.

D.5.4 PARAMETERS

infile [filename]

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.

outfile [filename]

Name of the output FITS response file (*.rsp) that will contain the N × M OGIP-compatible response matrix, usable with XSPEC.

hkfile [filename]

Name of the DAP Housekeeping file that contains detector bias voltage (column: BHV-Mon_Level) and electronic threshold (column: DM_HK_Vthr) values for the time interval of interest. If “NONE” is specified, batdrmgen uses the values specified by the hv_def and vthr_def parameters.

- At present, batdrmgen uses this file simply to check the hv values, make certain they are nominally 200V, and warn the user if they are not. Therefore, the default value of “NONE” is perfectly acceptable.

(row = 1) [integer]

The row within the PHA file to which the response matrix applies

(calfi le = CALDB) [filename]

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 = CALDB) [filename]

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.
(escale = CALDB) [string]
Desired form of the incident photon energy bin edges. Possible responses include:

- **FILE** the program reads in a FITS file containing the bin edges (see description of the efile parameter below);
- **LIN** linear binning
- **LOG** logarithmic binning.

- Both **LIN** and **LOG** allow the user to specify the number of bins (see nphoton_bins below) and the energy range (see elimit_lo and elimit_hi listed below).

(detmask = NONE) [string]
Name of a “mask weight” map (with dimensions of 286x173) which indicates which detectors were used to accumulate the spectrum. At present, this parameter is not used.

(method = MEAN) [string]
Computation method for DRM. The default method (“MEAN”) accesses calibration parameters that are averaged over the entire BAT FOV. The alternative method is “TABLE”, in which batdrmgen accesses a 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.

(flight_data = YES) [boolean]
Flag indicating whether the PHA file contains flight data as opposed to ground calibration data.

(nphoton_bins = 200) [integer]
The number of bins desired in the incident photon energy scale (used for escale = LIN or LOG as discussed above).

(elimit_lo = 10.0) [real]
The desired lower energy limit for the incident photon energy scale (used for escale = LIN or LOG as discussed above).

(elimit_hi = 500.0) [real]
The desired upper energy limit for the incident photon energy scale (used for escale = LIN or LOG as discussed above).

(efile = CALDB) [string]
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 bin edges will be used. This default bin edges have the energy range of 10 keV to 1000 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.
D.5. BATDRMGEN

\[ (hv_{\text{def}} = 200.0) \text{ [real]} \]
The default detector bias voltage setting, in volts. This value is used if no hkfile is specified. If an hkfile is specified and some of the values contained in it are NULL, this value is substituted in place of the null values.

\[ (vthr_{\text{def}} = 12.0) \text{ [real]} \]
The default XA1 threshold setting, in millivolts. This value is used if no hkfile is specified. If an hkfile is specified and some of the values contained in it are NULL, this value is substituted in place of the null values.

\[ (\text{fudge} = \text{INDEF}) \text{ [string]} \]
A string specifying which correction should be applied to the response matrix calculation. Possible values are:

- **INDEF** Apply default correction (equivalent to CRAB)
- **CRAB** Apply correction derived by fitting the residuals of the uncorrected matrix to an on-axis crab spectrum
- **NONE** Do not apply a correction

\[ (\text{clobber} = \text{NO}) \text{ [boolean]} \]
If the output file already exists, then setting “clobber = yes” will cause it to be overwritten.

\[ (\text{chatter} = 2) \text{ [integer, 0 - 5]} \]
Controls the amount of informative text written to standard output. Setting chatter = 1 produces a basic summary of the task actions; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints detailed debugging information.

\[ (\text{history} = \text{YES}) \text{ [boolean]} \]
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.

D.5.5 EXAMPLES

```
batdrmgen sw00050100004bsvo01f0g01f0.pha sw00050100004bsvo01f0g01f0.rsp NONE
```

D.5.6 SEE ALSO

batbinevt

D.5.7 LAST MODIFIED

Mar 2006
D.6 BATECONVERT

D.6.1 NAME
bateconvert - read in an event file and apply energy conversions.

D.6.2 USAGE
bateconvert infile calfile residfile pulserfile [outfile] [calmode] [zeroit]

D.6.3 DESCRIPTION
This task 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.

D.6.4 PARAMETERS

infile [filename ]
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 [filename ]
Name of the calibration file. This is a required parameter. 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. If zeroit==YES then the calfile need not exist and calfile can be set to NONE (or left off the command line).

residfile [filename ]
Name of the file containing the residuals between a quadratic pulser gain fitting and a linear fitting. This is a required parameter. This file is typically derived from ground processing. This file is used for all but the LINEAR 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 and had the appropriate value of the GAINMETH keyword. If calmodes CUBIC, DIRECTCUBIC, or FIXEDDAC are used, the file must also have a GAIN3 column. If DIRECTCUBIC or FIXEDDAC are used, the file must have
LIN_GAIN and LIN_OFFSET columns, contain the cubic fit itself rather than the residuals of the cubic from the linear fit, and the GAINMETH keyword is FULLCUBIC. If zeroit==YES or CALMODE==LINEAR then the residfile need not exist and residfile can be set to NONE (or left off the command line).

**pulserfile="CALDB"** [string]
Name of the file that contains the pulser DAC to energy calibration coefficients, or “CALDB” to search CALDB. These are the best known coefficients derived from ground calibration efforts. Only used for QUADRATIC correction; pulsercal=NONE is allowed for the LINEAR correction case.

**fltpulserfile="CALDB"** [string]
Name of the file that contains the pulser DAC to energy calibration coefficients, or CALDB; this should be the as-flown coefficients at the time of the observation. Only used for QUADRATIC correction; pulsercal=NONE is allowed for the LINEAR correction case.

**outfile** [filename]
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).

**calmode=INDEF** [string]
The calibration mode. The currently supported options are LINEAR, QUADRATIC, CUBIC, DIRECTCUBIC, and FIXEDDAC. If LINEAR is chosen, no residfile is needed, otherwise, the keyword GAINMETH in residfile must correspond to the calmode. If calmode == INDEF, the mode will be chosen based on the keyword in the residfile. FIXEDDAC is the preferred mode for accurate energy correction. LINEAR is provided to do the same correction as is done by the flight software before the events are binned in a DPH. A linear fit applies two parameters, gain and offset using the formula: ENERGY=GAIN*(OFFSET-PHA). The other fits perform the linear fit and then adds in a residual to correct for the deviation of the true fit from linearity, except for DIRECTCUBIC. DIRECTCUBIC computes the energy from the cubic pre-flight DAC to ADU fit and the values in pulserfile alone, ignoring the changes indicated by changes in the peaks of the pulser as a function of DAC value in flight.

**zeroit=NO** [boolean]
An optional parameter requesting bateconvert to set all values in the PI and ENERGY columns to zero instead of calculating the energy conversion. If zeroit==YES, then calfile can equal NONE.

**scaled_energy=YES** [boolean]
If YES (default), then the ENERGY column is written as 16-bit integers with a TSCALE scale factor which converts to keV. If NO, 32-bit floating point values are written directly. The default is YES to conserve disk space.

**clobber = NO** [boolean]
If the output file already exists, then setting “clobber = yes” will cause it to be overwritten.
(chatter = 1) [integer, 0 - 5 ]

Controls the amount of informative text written to standard output. Setting chatter = 4 or higher will produce detailed diagnostic output; chatter = 1 prints out a basic diagnostic message. The default is to produce a brief summary on 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 bateconvert task parameters that were used to produce the output file.

D.6.5 EXAMPLES

Note that when commands are issued on the Unix command line, strings containing special characters such as '[' or ']' must be enclosed in single or double quotes. The following examples illustrate ways in which bateconvert can be used.

1. Apply a linear conversion to convert from PHA (ADUs) to ENERGY (keV). The input file is updated with the PI and ENERGY columns filled in.

```
bateconvert infile='events.fits' calfile='cal.fits' residfile=NONE
     pulserfile=NONE calmode=LINEAR
-or- bateconvert events.fits cal.fits NONE NONE calmode=LINEAR
```

2. Apply the same linear conversion from PHA(ADUs) to ENERGY (keV). Copy the input file to a new file with the PI and ENERGY columns filled in. The input file is untouched.

```
bateconvert infile='events.fits' calfile='cal.fits' outfile='out.fits'
     residfile=NONE pulserfile=NONE calmode=LINEAR
-or- bateconvert events.fits cal.fits NONE NONE out.fits calmode=LINEAR
```

Overwrite an existing version of ‘out.fits’

```
bateconvert events.fits cal.fits NONE NONE \!out.fits calmode=LINEAR
-or- bateconvert events.fits cal.fits out.fits NONE NONE calmode=LINEAR clobber=YES
```

3. Apply a quadratic conversion instead of a linear conversion. This requires valid residfile and pulserfile.

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

4. Set all values in the PI and ENERGY columns of the existing file to zeros.

```
bateconvert events.fits calfile='none' residfile=NONE
     pulserfile=NONE zeroit=YES
```
D.6. SEE ALSO

baterebin

D.6.7 LAST MODIFIED

Aug 2006
APPENDIX D. BAT SOFTWARE TOOL REFERENCE

D.7 BATEREBIN

D.7.1 NAME

baterebin - read in a survey DPH and rebin the energy spectra using non-linear corrections

D.7.2 USAGE

baterebin infile outfile calfile [residfile pulserfile fltpulserfile outmap lowecare highecare detmask ebins outdatatype wholecount rows

D.7.3 DESCRIPTION

This code reads in a BAT Detector Plane Histogram (DPH), which is organized as a single three-dimensional binary table with spectral information for each BAT detector arranged by geometric location in the focal plane. The tool extracts the individual detector spectra and reanalyzes them using cubic corrections from a ground cal file. A new DPH is output with the new spectra.

D.7.4 PARAMETERS

infile="" [string ]
    Input file name, which must contain a detector plane histogram (extension names BAT_DPH and EBOUNDS must be in the file).

outfile="" [filename ]
    Name of the output file. The output file is also a DPH file with a new DPH_COUNTS column that has been corrected.

calfile="gainoffset.fits" [filename ]
    Name of the file that contains the “gain/offset” maps. These are pseudo-linear calibration constants for conversion of ADU to energy units.

(residfile="CALDB") [string ]
    Name of the file that contains the quadratic or cubic residuals to the ADU to DAC relation, or CALDB. The keyword GAINMETH in this file will control whether the quadratic or cubic correction is performed.

(pulserfile="CALDB") [string ]
    Name of the file that contains the pulser DAC to energy calibration coefficients, or “CALDB” to search CALDB. These are the best known coefficients derived from ground calibration efforts.
(fltpulserfile="CALDB") [string ]

Name of the file that contains the pulser DAC to energy calibration coefficients, or CALDB; this should be the as-flown coefficients at the time of the observation.

(detmask=“NONE”) [string ]

Name of the file that contains a detmask image or table containing a detmask image. A detmask is a DPI. Pixels containing values other than the value in the goodval keyword are zeroed out in the output DPH.

(outmap=“NONE”) [string ]

Name of the file to be written that contains a detmask image. The values in the image are:

- **bit 0** indicates high edge of input integral bin is more than the low edge of the lowest energy non integral output bin that you care about (default 14.0 keV)
- **bit 1** indicates low edge of input integral bin is less than the high edge of the highest non integral output bin that you care about (default 151.4 keV)
- **bit 2** (value=4) indicates correction > 50 keV in magnitude
- **bit 4** (value=8) indicates wacked gain/offset values
- **bit 5** (value=16) indicates no V -> energy conversion value available
- **bit 6** (value=32) indicates detector masked by input detmask

(lowecare=14.0) [real ]

Lowest energy you care about. This value determines only if bit 1 is set in outmap when the integral bin runs into a bin that you care about.

(highecare=151.4.0) [real ]

Highest energy you care about. This value determines only if bit 2 is set in outmap when the integral bin runs into a bin that you care about.

(ebins=“FILEBINS”) [string ]

If FILEBINS, the output has the same energy bins as the input (but the counts will be redistributed into the correct energy bins). Energy bin ranges, expressed as a comma-separated list of floating point number ranges, a file name containing energy bin ranges, or CALDB. Each bin in the comma-separated list is specified as E_MIN1-E_MAX1,E_MIN2-E_MAX2,... (in units of keV). 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. A standard 80-channel set of binnings is used. By using CALDB:n, a specific n-channel binning can be selected (provided it exists in CALDB). If the first energy bin starts at 0, the first bin will include all the counts from the first bin from the input (which has all counts less than a nominal linear uncorrected 10 keV). If the highest energy bin stops at 6553.6, the highest bin will include all the counts from the highest input bin (which has all counts greater than a nominal linear uncorrected 191.4 keV). Otherwise integral bin counts will be discarded.

(wholecount = no) [boolean ]

Selects whether the output contains only whole (wholecount=yes) or possibly fractional (wholecount=no) counts. Automatically set to yes if the output data type is an integer type. Either way, the output should contain the same total counts per detector as the input.
APPENDIX D. BAT SOFTWARE TOOL REFERENCE

(rows="-" [string ]

The row numbers in the input DPH which are to be included in the output. Row numbers in a DPH correspond to different time intervals. The rows are expressed as a series of ranges separated by commas, e.g. “1, 3-5, 9-10”.

(outdatatype="FLOAT")) [string ]

Data type of the output histograms. One of “SHORT”, “INT”, “FLOAT”, or “DOUBLE”. Choosing an integer type will result in the wholecount parameter to be set to YES.

(clobber="no") [boolean ]

If the output file already exists, then setting “clobber = yes” will cause it to be overwritten.

(chatter = 1) [integer, 0 - 5 ]

Controls the amount of informative text written to standard output. Setting chatter = 4 or higher will produce detailed diagnostic output; chatter = 1 prints out a basic diagnostic message. The default is to produce a brief summary on 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 bateconvert task parameters that were used to produce the output file.

D.7.5 EXAMPLES

Note that when commands are issued on the Unix command line, strings containing special characters such as ‘[‘ or ‘]’ must be enclosed in single or double quotes. The following examples illustrate ways in which baterebin can be used.

1. Rebin all rows in a DPH file choosing boundaries to the energy bins different from the nominal boundaries in the input file.

   baterebin {infile} {outfile} {calfile} {residfile} {pulserfile} outefile={outefile} rows=-

2. Rebin the 2nd to 10th rows in a DPH file assigning fractional counts to bins as appropriate.

   baterebin {infile} {outfile} {calfile} {residfile} {pulserfile} rows=2-10

3. Rebin the third row in a DPH file moving only whole numbers of events from a bin.

   baterebin {infile} {outfile} {calfile} {residfile} {pulserfile} wholecount=yes rows=3

D.7.6 HISTORY

/**************************** HISTORY **********************************
Version 3.0 --- Jay Cummings October 13, 2004
* Now does more than one row at a time
* Added ebins parameter: Filename of text file containing number of energy
  bins and a list of the lower edges of the bins in the output file.
* Does things in image order instead of detector number order
Version 4.0 --- Jay Cummings February 18, 2005
* Added a cubic correction, now the default.
Version 5.0 --- Jay Cummings March 11, 2005
* Added fixed-DAC method, now the default.
Version 6.0 --- Jay Cummings March 30, 2005
* Now copies the GTI extension of the input DPH to the output DPH
* Moved the bin edge energy calculations to outside the row loop. This should
  slightly speed up processing of multi-row DPHs.
* Changed which keywords are checked in the calfile to determine what the
  gain and offset indices are. In some cases GAININD != LDPGAIN and/or
  OFFINDEX != LDPOFFST, and LDPGAIN or LDPOFFST is not equal to the
  numbers in the name of the file. This should make the warning about
  the indices not matching appear less often.
* Changed the wording of warnings to make them seem less dire.
* Now uses the row of the calfile that is closest in time to the
  first DPH row instead of the first calfile row
* 
Version 6.1 --- Jay Cummings April 28, 2005
* Fixed a bug that would delete the incorrect number of rows from the
  output EBOUNDS extension compared to the input EBOUNDS extension.
* Previously the number of input energy bins was figured from the
  NAXIS2 keyword of the input DPH extension rather than the input EBOUNDS
  extension. If the number of output energy bins was different, there
  would be extra rows in the EBOUNDS extension filled with the bounds
  from the input file (since this extension was originally copied from
  the input file, then the new values were written in). This error
  was seen when the number of output energy bins was less than the
  number of DPH rows. Apparently somehow, I don’t know how, the right
  number of rows were written when the number of input DPH rows was less
  than the number of output energy bins.
* Used to compare the low end of the highest input bin to the high end of
  the highest output bin in order to determine whether to print a warning
  message that they were different. Now compares the high end of both
  bins. This change has no effect on the function of the program, just
  whether the warning message is printed.
* 
Version 6.2 -- C. Markwardt 24 Jun 2005
* Make gain_offset structure static so that it doesn’t clobber the
* stack and confuse a lot of architectures. (it was ~2.5 MB)

* Also, free the large memory chunks at the end of the task.

Version 7.0 -- Jay Cummings July 2005
* Added the ability to include an input detmask
* Reduced the number of error messages
* Reduced the sound of urgency of error messages
* Added more information on in what way and how much pixels are not good

Version 7.1 -- Jay Cummings Aug 2005
* Counts from the integral bins will be placed into the end output bins
  if and only if the extreme energy of the corresponding output bin
  is the minimum energy (0) or maximum energy (6553.6)
* Counts from the integral bins were not placed in other than the
  end output bins in previous versions and will not be in this version.

D.7.7 SEE ALSO
bateconvert – energy calculation for event files

D.7.8 LAST MODIFIED
Aug 2006
D.8 BATFFTIMAGE

D.8.1 NAME

batfftimage - make sky image from detector plane image by FFT deconvolution

D.8.2 USAGE

batfftimage infile outfile

D.8.3 DESCRIPTION

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 batbkgclean, 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 use the pcodemap parameter. Users must also provide an input map. Although the contents of the map are ignored, the FITS keywords are used to select the proper calibration files from CALDB. 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 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. Output images have extension names of BAT_IMAGE_n and BATPCODE_n for sky and partial coding maps, respectively.

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.

batfftimage can also calculate theoretical Poisson variance and significance maps. Note that the input file must not be cleaned for these maps to be correct.

By default, batfftimage will quantize the image into discrete intensity levels. For sky images, the quantization steps are adaptive, depending on the local noise level. Quantization usually reduces compressed image sizes significantly. The degree of quantization is governed by the ‘keepbits’
D.8.4 PARAMETERS

infile [filename]

Input file name containing BAT detector plane image. If pcodepa is ‘YES’, then the contents of the map are ignored, but the keywords from the map file are used to select the calibration files from CALDB. Thus, using a detector enable map or a quality map is appropriate in that case. Avoid using the CFITSIO syntax to select image extensions; use the ‘rows’ parameter instead.

outfile [filename]

Output sky image or partial coding map file name.

attitude = “NONE” [string]

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

(aperture = “CALDB:FLUX”)[filename]

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. If several aperture types are available in the CALDB, the user can specify CALDB:apertype. Valid values of apertype are currently: FLUX or DETECTION, for apertures optimized for flux reproducibility or detection sensitivity, respectively.

(detmask = “NONE”) [string]

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 = “NONE”) [string]

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

(bkgvarmap=“NONE”) [string]

Optional output file name for the theoretical Poisson background fluctuation map. This map will be either the standard deviation map or the variance map depending on the value of bkgvartype. NOTE: The input map must not be cleaned for the theoretical map to be computed properly. A value of NONE indicates no fluctuation maps should be created. A value of APPEND indicates that each fluctuation map should be appended to the output file given by ‘outfile’.

(bkgvartype=“STDDEV”) [string]

Type of output fluctuation map. Valid values are: STDDEV, for a standard deviation map; or VARIANCE, for a variance map.
(signifmap="NONE") [string ]
Optional output file name for the significance map based on the theoretical Poisson variance map (see ‘bkgvarmap’ for restrictions). A value of NONE indicates no significance maps should be created. A value of APPEND indicates that each significance map should be appended to the output file given by ‘outfile’.

(oversampx = 2) [integer ]
Oversampling factor of image in X direction. Finer sampling may allow peaks which straddle pixel boundaries to be detected, but of course will result in increased compute time and disk storage.

(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 coordinate from its nominal design position.

(maskoffy = 0.0) [real ]
Translational offset to apply to the mask along the BAT_Y coordinate from its nominal design position.

(maskoffz = 0.0) [real ]
Translational offset to apply to the mask along the BAT_Z coordinate 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. If NO, then, the maps will be used unadjusted. (NOTE: pcodemap implies rebalance=NO)

(pcodemap = "NO") [string ]
Partial coding map output setting. The possible values are:

"NO"  No partial coding maps are output (although they may be computed internally, depending on the required corrections).
"YES"  Only the partial coding map is output.
"APPEND_ALL"  Append each partial coding map to the output.
"APPEND_LAST"  Append only the last partial coding map to the output.

• Note that if the spacecraft attitude changes between sequential input images, then the coordinate system attached to the partial coding map will also change. Use APPEND_ALL in this case. If the spacecraft attitude is known to be fixed (for example, if each image is taken at the same time, but in different energy bands), then APPEND_LAST will append only one partial coding map, and thus save space.

(pcodethresh = 0.01) [real ]
Minimum allowable partial coding value in the partial coding map. This is a fraction between 0.0 (no partial coding) and 1.0 (fully coded). The default of 0.01 implies that at least 1% of the detectors must be illuminated. Values below the threshold will be set to zero in the partial coding map.
(bat_z = 0) [real ]
For testing with a near-field source. The position of the source in centimeters along the
BAT_Z axis. A value of 0 indicates infinity.

(origin_z = 0) [real ]
For ground testing with a near-field source. The position of the origin of coordinates (in cm),
used for the angular measures attached to the output image.

(corrections = “default”) [string ]
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 cosine and edge projection effects
    maskwt Apply corrections for FFT technique
    ndets Normalize by number of exposed detectors
    pcode Apply partial coding corrections

(keepbits = “7”) [string ]
Integer number of bits of precision to keep in the output images. A value of ALL means
keep all bits. A positive value indicates that ‘keepbits’ bits of precision should be preserved,
and the other bits should be set to zero. Fewer kept bits means the output images are more
compressible, at the expense of lost precision. For partial coding maps and variance maps, a
straight number of bits are kept (thus preserving the same relative precision across the map).
For sky maps and significance maps, ‘keepbits’ prescribes the number of noise
bits to keep; more bits may be kept if a pixel is significantly above the noise (thus preserving the same
noise quantization error across the map). NOTE: it must be possible to compute a variance
map in order for most ‘keepbits’ operations to work. (see ‘bkgvarmap’ for restrictions).

(handedness = “left”) [string ]
The handedness of the image. For an image where north is up, a left-handed image has
East=left, and a right-handed image has East=right. “left” is conventional for astronomical
images.

( rows = “-” ) [string ]
An index list of images to operate on. This is a standard comma-separated list of row ranges,
starting with image number 1. A range of “-” indicates all images should be processed.

(countscol = “INDEF”) [string ]
If the input image is a FITS table, the column to analyze. If countscol is INDEF, then the
first 2 dimensional column to be found in the table is used.

(teldef = “CALDB”) [string ]
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.
D.8. BATFFTIMAGE

(time = 0) [real ]

Command line override for the image time. This MET time is used to interpolate the attitude file. By default the image midpoint time is used (= (TSTART+TSTOP)/2).

(maskwtswgain = 0.03) [real ]

Mask weight technique software gain correction factor. When corrections=maskwt, the image is divided by (1+maskwtswgain). This is correction is software algorithm dependent and not intended to be changed by the user.

(copyall = YES) [boolean ]

If set, then batfftimage will copy any extra HDUs to the output file. If not set, then only the sky images are written. Extra extensions are any HDUs that come after the image extension(s).

(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. Setting chatter = 1 produces a basic summary of the task actions; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints debugging information.

(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.

D.8.5 EXAMPLES

1. Reconstructs the sky image from the detector plane image focalplane.img. The aperture map, aperture.img, is in the current directory.

   batfftimage focalplane.img skyimage.img

2. Make a partial coding map for the given detector configuration given in detmask.img. The output file is the image pcodemap.img. If the detmask parameter were not given, then a default one would be used, which assumed that all detectors were enabled.

   batfftimage NONE pcodemap.img detmask=detmask.img pcodemap=YES

D.8.6 REFERENCES

Fast Fourier Transform routines by T. Ooura, used with permission. [http://momonga.t.u-tokyo.ac.jp/~ooura/]

D.8.7 SEE ALSO

batbkgclean, batcelldetect
D.8.8 LAST MODIFIED

Sep 2006
D.9 BATGLOBALGTI

D.9.1 NAME
batglobalgti - retrieve BAT global science analysis times

D.9.2 USAGE
batglobalgti infile outfile

D.9.3 DESCRIPTION
batglobalgti retrieves global good or bad times for BAT science analysis. The BAT CALDB has a GTI file which contains a listing of all known good and bad science analysis times, include a quality indicator and a description of any issues. This GTI is meant to reflect any operational issues that affected the BAT science data quality which are not easily extractable from the filter file or other telemetry. The user is not excused from doing other routine quality filtering, for example for SAA or noisy detectors. Thus, this task is most useful for doing gross filtering of data at the observation level to remove times of known bad data.

The time intervals stored in the GTI file have QUALITY values assigned according to the following scale:

0 No known issues; data OK
1 Something non-routine occurred, but it had no known impact;
2 Bad data, with some possible good data; user must proceed with extreme caution;
3 Bad data, with little to no good data.

The user can choose which quality level to accept with the ‘expr’ parameter. The default value is 0 or 1.

The result is a standard GTI file that can be merged with other GTI files, or otherwise used to time filter time-ordered Swift data. This GTI file applies to the entire mission duration.

Please note: The time intervals in the output file are not necessarily “good.” That depends on the quality level chosen (i.e. if one chooses expr="QUALITY >= 2" then only the bad time intervals will be selected).

D.9.4 PARAMETERS

infile [filename ]

Global good/bad science time interval file. A value of “CALDB” is allowed (and in fact recommended), indicating to select data from the Calibration database.
outfile [filename ]

Name of the output GTI file.

(expr = “QUALITY < 1”) [string ]

Quality level selection filter. Any row filter expression is allowed, involving the columns START, STOP and QUALITY. “Good” data will typically be chosen with ”QUALITY < 1“ and ”bad“ data will have ”QUALITY >= 2“.

(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.

D.9.5 EXAMPLES

1. Create a “good” time interval file from CALDB.

    batglobalgti CALDB good.gti

D.9.6 SEE ALSO

maketime, mgtime

D.9.7 LAST MODIFIED

Sep 2006
D.10 BATGRBPRODUCT

D.10.1 NAME

batgrbproduct - standard BAT GRB processing

D.10.2 USAGE

batgrbproduct indir outdir

D.10.3 DESCRIPTION

batgrbproduct performs standard processing for gamma-ray bursts detected by the Swift BAT instrument. It performs the following activities:

- generation of a master quality map
- generation of good time interval files
- extraction of events near the time of the GRB
- computing mask weights for all events for the GRB position
- finding an optimal time interval based on Bayesian blocks analysis
- generation of sky images
- generation of a partial coding map
- refinement of the BAT position (if possible)
- generation of energy spectra for different time intervals, including
  - pre-slew
  - during-slew
  - post-slew
- generation of detector response matrices
- generation of light curves with different samplings
- computation of counts fluences
- creation of a summary report
- creation of a detailed execution log file
NOTE: the scripts batgrbproduct and bat-burst-advocate are identical. The first name has been preserved for historical reasons, and the second name for consistency with the product scripts of the UVOT and XRT.

By default it is assumed that the TDRSS position message is present. This message provides information about the position of the burst and the initial time interval of the burst. However, the user can override these values using command line parameters. If the RA and Dec values are set to BLIND, then sky images are made, but no further analysis is done, and it is up to the user to search the image to find new sources.

PREREQUISITES - the user must have the standard FTOOLS (6.0.5 or later) installed and set up. Also, the CALDB must be installed and set up. It is assumed that the input directory structure is the same as that produced by the Swift Data Center (SDC). The event by event data should already be energy-calibrated by either the SDC or running bateconvert manually (i.e. the PI and ENERGY columns should be filled).

D.10.4 SUMMARY OUTPUTS

Upon finishing, the output directory contains a text file called process.log. This file contains all the commands executed and detailed output transcripts. Users can check this file to: (a) diagnose problems; and (b) to get ideas on how to use the software to make their own custom science products.

The output directory also contains report.txt, which is text file with a summary report giving basic statistics and factual information about the trigger that was analyzed.

The task concludes by printing to the console a detailed inventory of files it created, including the name and a brief description for each file.

D.10.5 OUTPUT DIRECTORIES

The output files are arranged by type into several directories. The following list describes each directory. Each file name contains code phrases which are also described.

outdir/lc  Light curves. File names with “1chan” and “4chan” have 1 and 4 energy bins respectively. Time binnings can be 1 second (“1s”), 64 millisecond (“64ms”) and 4 millisecond (“4ms”). The light curves cover the full range of event data.

outdir/pha  Spectra and response matrices. Spectra are computed for the total burst interval, the T50/T90 intervals, the peak flux interval (1 second), and also for the pre-slew, slew, and post-slew intervals. Response matrices are computed for the pre-slew and post-slew intervals.

outdir/dpi  Detector images. Detector images are made in the pre-burst, pre-slew and post-slew time intervals. Pre-burst means an image created before any detectable emission. Images with 1 and 4 energy bins are created.

outdir/img  Sky and partial coding images. Sky images are made in the same time intervals as detector images, but only for the 1-channel images. Small “postage-stamp” images are also created (suffix “.postimg”), centered on the burst position. Partial coding maps are made for the pre-slew and post-slew intervals.

outdir/gti  Duration information. Various good time interval files are generated. Files with “acs” are spacecraft-attitude related (“acs_10am”, “acs_sett”, “acs_slew” corresponding to the 10-arcmin, settled, and slewing flags, respectively). Files with “grb” select events near the GRB
(within 1500 seconds), including the pre-slew, slew and post-slew intervals. The “tdrss” file contains time intervals transcribed from the TDRSS position message. Files with “bb” are Bayesian blocks produced with the battblocks task on light curves with different time binnings.

`outdir/auxil` Auxiliary information. The file with “qmap” is a detector quality map. The file with “output.cat” contains a revised position based on a fit to all pre-slew data. The “output.reg” file is a region file suitable for use within DS9 or FV. The “evaux” file is an auxiliary ray-tracing file.

`outdir/events` Event files. Intermediate event files are stored here. Since they are re-created each time the task is run, the files in this directory can be deleted if desired.

### D.10.6 ERRORS

Most error conditions are reported on the console. Please note that not all bursts are the same, and the Swift spacecraft response is not the same. For example, there may not be a slew to the burst, in which case, there won’t be “slew” and “post-slew” spectra or images. Also, if the burst does not extend past the end of the slew (if there is one), there will not be a post-slew spectrum. The task will report these failures to produce spectra and images as errors, but they are benign.

For certain kinds of bursts – namely short hard bursts – the task is not always able to find the time interval of the burst. In those cases, the user must manually determine the time interval of the burst and use the `trigtime` and `trigstop` keywords to the task.

### D.10.7 PARAMETERS

`indir [string ]`

Input directory containing Swift data. This directory must have the standard Swift archive layout. Files in the input directory will not be modified.

`outdir [string ]`

Output directory which will contain the processing results. The output files are organized by file type: events, pha (spectra), lc (light curves), gti (good time intervals), auxil (miscellaneous). Also a summary report is created (report.txt), and a detailed processing log (process.log).

`ra = “BAT” [string ]`

RA of the source (degrees). A value of “BAT” or “XRT” means that the BAT or XRT TDRSS position should be used. A value of “BLIND” means to do the analysis blind, without concentrating on any one position in the field of view. When “BLIND”, then the `trigtime/trigstop/backstrt/backstop` parameters must be specified.

`dec = “BAT” [string ]`

Declination of the source (degrees). See “ra”.

`trigtime = 0.0 [real ]`

Start time of the foreground interval containing the burst in mission elapsed time. A value of 0 indicates to use the times from the TDRSS position message, and a non-zero value overrides the TDRSS message.
trigstop = 0.0 [real ]
Stop time of the foreground interval containing the burst in mission elapsed time. A value of 0 indicates to use the times from the TDRSS position message, and a non-zero value overrides the TDRSS message.

backstrt = 0.0 [real ]
Start time of the background interval before the burst in mission elapsed time. A value of 0 indicates to use the times from the TDRSS position message, and a non-zero value overrides the TDRSS message.

backstop = 0.0 [real ]
Stop time of the background interval before the burst in mission elapsed time. A value of 0 indicates to use the times from the TDRSS position message, and a non-zero value overrides the TDRSS message.

date_obs = “INDEF” [string ]
User override of the DATE-OBS keyword in event files. This parameter is useful if the input data has an invalid date, like ‘2001-01-01’ or ‘TBD’. In that case, set date_obs=’YYYY-MM-DD’ where YYYY-MM-DD is the date of the observation. A value of “INDEF” indicates that the DATE-OBS keyword should not be changed (although it will still be checked for validity).

(tbkgsbu = YES) [boolean ]
If set, then the task instructs battblocks to perform background subtraction during the “background” intervals before and after the GRB before estimating the T50/T90 durations. If not set, then it is assumed that the light curves are already background subtracted (via mask weighting). Caveats: setting tbkgsbu=YES may cause problems if the light curve doesn’t have a true quiescent background interval; setting tbkgsbu=NO may cause problems if the light curve is not fully background subtracted.

(pcodethresh = 0.0) [real ]
Minimum partial coding threshold. Set this value to a number greater than 0 in order to exclude noisy data when a GRB passes out of the BAT field of view. A value of 0.0 includes all data; A value of 1.0 includes only data when the source is in the (small) BAT fully coded field of view.

(chatter = 1) [integer, 0 - 5 ]
Currently ignored.

(history = YES) [boolean ]
Currently ignored.

D.10.8 EXAMPLES

1. Run the BAT GRB product script on BAT trigger 154112 (GRB 050908). The BAT GRB data are archived in a directory called 00154112000 (the “000” refers to segment number 0). batgrbproduct will perform a standard analysis and place the results in the directory 00154112000-results.

batgrbproduct 00154112000 00154112000-results
D.10. BATGRBPRODUCT

D.10.9 SEE ALSO

batmaskwtevt, batbinevt, batdetmask, battblocks, batfftimage, batcelldetect, batgrbproduct

D.10.10 LAST MODIFIED

Nov 2006
D.11 BATHOTPIX

D.11.1 NAME

bathotpix - locate and mask out hot pixels in a BAT detector image

D.11.2 USAGE

bathotpix infile outfile

D.11.3 DESCRIPTION

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.

D.11.4 PARAMETERS

infile [filename ]

Input file name containing a detector plane image.

outfile [filename ]

Output detector mask image.

(detmask = “NONE”) [string ]

Input detector mask image. Pixels with bad quality are ignored in the bathotpix analysis.

(row = 1) [integer ]

Image number to process, where the first image is number 1. This applies both to files with multiple image extensions (where ‘row’ selects the extension number) and to files with one binary table extension (where ‘row’ selects the table row).
(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. A value of 0.25 indicates an enlargement of 0.25 x the median on both sides of the distribution.

(guardval = 5.0) [real ]

Further amount to enlarge the selection window, this time in counts. This keyword is present for the case of an image with very small dynamic range, where guardfract may be too small to be effective.

(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. If no, then only the pixels adjusted within this run of bathotpix are written.

(zerothresh = 20.0) [real ]

Threshold value. If the median is above zerothresh, then pixels with zero are considered to be COLD. If the median is below zerothresh, i.e. the total number of counts is very low, then pixels with zero are considered to be okay.

(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.

(clobber = NO) [boolean ]

If the output file already exists, then setting “clobber = yes” will cause it to be overwritten.

(chatter = 1) [integer, 0 - 5 ]

Controls the amount of informative text written to standard output. Setting chatter = 1 produces a basic summary and image statistics; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 3 prints detailed information on which hot and cold detectors; chatter = 5 prints debugging information.

(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.
D.11.5 EXAMPLES

1. Locate hot pixels in a detector plane image

    bathotpix image.dpi image.mask

2. Same as example 1, but an input mask is also supplied

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

D.11.6 SEE ALSO

batgse2dpi

D.11.7 LAST MODIFIED

Aug 2006
D.12 BATID2XY

D.12.1 NAME

batid2xy - read in the detector number (either block, dm, side, detector numbers or single detector ID) and output the x/y position in the BAT array –OR– input the x/y position in the BAT array and output the detector identification.

D.12.2 USAGE

batid2xy infile [block] [dm] [side] [det] [detid] [detx] [dety] [outfile]

D.12.3 DESCRIPTION

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 batidconvert to figure out the X (column) and Y (row) numbers associated with the particular detector or does 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, 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.

D.12.4 PARAMETERS

infile [filename ]
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.

(block) [integer, 0 - 15 ]

Specify the block number. Ignored if a valid input file is specified or if any of the detid, detx, or dety parameters are greater than -1.

(dm) [integer, 0 - 7 ]

Specify the detector module number. Ignored if a valid input file is specified or if any of the detid, detx, or dety parameters are greater than -1.

(side) [integer, 0 - 1 ]

Specify the detector module side. Ignored if a valid input file is specified or if any of the detid, detx, or dety parameters are greater than -1. 0 is equivalent to A and 1 to B.

(det) [integer, 0 - 127 ]

Specify the detector number. Ignored if a valid input file is specified or if any of the detid, detx, or dety parameters are greater than -1.

(detid) [integer, -1 to 32767 ]

Specify the detector id (block*2048 + dm*256 + side*128 + detector). If a different identification of detectors is desired, then this parameter should be set to -1. Ignored if a valid input file is specified.

(detx) [integer, -1 to 285 ]

Specify the detector X location in detector units, starting from 0. If a different identification of detectors is desired, then this parameter should be set to -1. Ignored if a valid input file is specified. If the value provided corresponds to a gap in the detector array, then the code returns a TNULL value for each detector identification.

(dety) [integer, -1 to 1273 ]

Specify the detector Y location in detector units, starting from 0. If a different identification of detectors is desired, then this parameter should be set to -1. Ignored if a valid input file is specified. If the value provided corresponds to a gap in the detector array, then the code returns a TNULL value for each detector identification.

(outfile) [filename ]

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

(clobber = NO) [boolean ]

If the output file already exists, then setting “clobber = yes” will cause it to be overwritten.

(chatter = 1) [integer, 0 - 5 ]

Controls the amount of informative text written to standard output. Setting chatter = 5 will produce detailed diagnostic output, otherwise this task normally does not write any output.
D.12. **BATID2XY**

(history = NO) [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 ftcopy task parameters that were used to produce the output file.

D.12.5 EXAMPLES

Note that when commands are issued on the Unix command line, strings containing special characters such as '[' or ']' must be enclosed in single or double quotes. The following examples illustrate ways in which batid2xy can be used.

1. With an input file (“BDSD” mode).

   batid2xy infile='detids.fits'
   -or- batid2xy detids.fits

   An input file can be created from an ascii table containing the block, dm, side, and detector values in columns 1-4, respectively. A template to create the proper column names can be found in the batid2xy directory as the file bdsd_mode.lis. A header template is bdsd_mode.hdr.

   If the data is in a file called data.lis, then the input file detids.fits is created using one of the following commands. Both of these assume that the template files are in the current directory.

   ```
   create_input.pl infile="data.lis" outfile="detids.fits"
   -or-
   fcreate bdsd_mode.lis data.lis detids.fits extname="DETID"
   headfile=bdsd_mode.hdr clobber=yes
   ```

2. With no input file (“BDSD” mode).

   batid2xy NONE 7 3 1 67


   batid2xy infile='detids.fits'
   -or- batid2xy detids.fits

   An input file can be created from an ascii table containing the detector ID values in column 1. A template to create the proper column names can be found in the batid2xy directory as the file detid_mode.lis. A header template is detid_mode.hdr.

   If the data is in a file called data.lis, then the input file detids.fits is created using one of the following commands. Both of these assume that the template files are in the current directory.
APPENDIX D. BAT SOFTWARE TOOL REFERENCE

create_input.pl infile="data.lis" outfile="detids.fits" batidmod="detid"
-or-
   fcreate detid_mode.lis data.lis detids.fits extname="DETID"
   headfile=detid_mode.hdr clobber=yes

4. With no input file ("DETID" mode).

   batid2xy NONE detid=15299

5. With an input file ("DETXY" mode).

   batid2xy infile='detids.fits'
   -or- batid2xy detids.fits

   An input file can be created from an ascii table containing the detector X and Y values in columns 1 and 2, respectively. A template to create the proper column names can be found in the batid2xy directory as the file detxy_mode.lis. A header template is detxy_mode.hdr.

   If the data is in a file called data.lis, then the input file detids.fits is created using one of the following commands. Both of these assume that the template files are in the current directory.

   create_input.pl infile="data.lis" outfile="detids.fits" batidmod="detyxy"
   -or-
   fcreate detyxy_mode.lis data.lis detids.fits extname="DETID"
   headfile=detyxy_mode.hdr clobber=yes

6. With no input file ("DETXY" mode).

   batid2xy NONE detx=259 dety=102

D.12.6 SEE ALSO

fcreate

D.12.7 LAST MODIFIED

Aug 2006
D.13 BATMASKTAGLC

D.13.1 NAME

batmasktaglc - Compute background subtracted light curves from raw BAT mask tagged counts

D.13.2 USAGE

batmasktaglc infile quadfile maskwt outfile [detmask] [ebounds]

D.13.3 DESCRIPTION

batmasktaglc converts a raw “mask tagged” light curve, produced by the BAT flight software, into a background subtracted mask weighted light curve. The BAT flight software by itself produces “unbalanced” mask weighted sums, which introduces a bias. This task must be used to remove the bias.

The three basic inputs to the task are the raw mask tagged light curve; the “quadrant rates” light curve; and the mask weight map used by the flight software to create the raw mask tagged light curves.

Generally speaking, raw mask tagged light curves will be stored with file names like: swNNNNNNNNNNNNbmtCCCCrrw.lc.gz, where NNNNNNNNNNN is the observation and segment numbers, and CCCCCCCCC is the bat catalog number for the source. A given observation may contain multiple mask tagged sources. The BAT can produce mask tagged light curves for up to three sources simultaneously. In addition, the sources chosen for mask tagging may vary from pointing to pointing in the same observation. Gamma-ray burst mask tagged light curves will have a catalog number of the form, 0002CCCC, where CCCC matches the last four digits of the GRB trigger number in decimal.

The quadrant rates file is typically in the obsid/bat/rate directory, with a file name, swNNNNNNNNNNNbrtqd.lc. It contains the total rates in each quadrant of the instrument, in several energy bands.

A mask weight map is produced by the BAT every time the mask tagged sources change (typically every pointing), and describes the weighting each detector had in computing the mask tagged sums. Typically, the maps should be attached to the raw mask tagged light curve file as follow-on extensions. Thus, users should normally just set “maskwt=INFILE”. The extension name of the weight map should be “MASK_TAG_WEIGHT”.

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. 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.
D.13.4 IMPORTANT NOTES

The mask tagged light curves are invalid during spacecraft slews. Mask tagged data during slews should be ignored.

The generation of the output file will fail if the file is too large to fit the available memory.

D.13.5 PARAMETERS

infile [filename ]
Name of the raw mask tagged rate file, as described above.

quadfile [filename ]
Name of the input quadrant rate file.

maskwt [string ]
Name of mask weight map file. Normally, users should set the value to “INFILE” or “FILE”, to indicate that the mask weight map is attached to the raw mask tagged light curve.

outfile [filename ]
Output light curve file name.

(detmask = “none”) [string ]
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. Typically, these files are found in files named like obsid/bat/hk/swNNNNNNNNNNbdecb.fits.

(ebounds = “INFILE”) 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 (use “INFILE” or “FILE”). One can also specify “CALDB” if the EBOUNDS extension is not present.

(corrections = “default”) [string ]
Comma-separated list of corrections to apply to the mask weighting, or “none” if no corrections are to be applied. Other possible correction types are: “flatfield” (basic flat fielding correction accounting for both cosine-type and r-squared-type effects); “cosine” (cosine effects of off-axis illumination only); “pcode” (partial coding effect only); “ndets” (correction for disabled detectors). The “default” correction is pcode,flatfield,ndets.

(detdiv = “YES”) [string ]
Do you want the rate to be per detector (detdiv=“YES”) or for the whole array (detdiv=“NO”). The division is only applied if the corrections parameter includes “ndets”. Note that if no detector mask is supplied, then the rate will be divided by the total number of detectors (32768).

(scale = 1.3) [real ]
The user is allowed to supply a scale factor to correct for any known errors in how the raw mask-tagged light curve is calculated. The scale factor is multiplicative and scales all output counts by the same factor. The default value of this parameter was determined empirically.
for the average case. The user can override the default with a more precisely determined scale
factor.

(clobber = NO) [boolean ]

If the output file already exists setting clobber=Yes causes the existing output file to be
overwritten.

(chatter = 1) [integer 0 - 5 ]

The chatter value controls the amount of output written to standard output. Setting chatter
= 5 causes detailed diagnostic output to be 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.

D.13.6 EXAMPLES

- Note that when commands are issued on the Unix command line, strings containing special
characters such as ‘[’ or ‘]’ must be enclosed in single or double quotes.

1. Generate a mask weighted light curve for GRB 050922B (trigger 156434).

```
batmasktaglc infile=sw00156434000bmt00026434_rw.lc.gz \
quadfile=sw00156434000brtqd.lc.gz maskwt=FILE \
outfile=grb050922b.lc ebounds=CALDB clobber=YES
```

Note that mask weighted data during slews should be ignored.

D.13.7 LAST MODIFIED

Nov 2006
**D.14 BATMASKWTEVT**

**D.14.1 NAME**

batmaskwtevt - compute mask weights for an event file

**D.14.2 USAGE**

batmaskwtevt infile attitude ra dec

**D.14.3 DESCRIPTION**

batmaskwtevt applies mask weighting, for a particular source position, to an event file. This is the first step to creating a light curve or spectrum for a source from event data.

By default, the coordinates are right ascension and declination. Users must also provide the instrument aperture and teldef files (or use CALDB). Instrument coordinates can also be used, as described in ADVANCED COORDINATE SYSTEMS below.

**D.14.4 ADVANCED COORDINATE SYSTEMS**

Users may specify a source position using a coordinate system other than sky position. Depending on the value of the coord_type parameter, the user may give the cartesian or angular position of the source in instrument coordinates. When coord_type is changed from “sky” to any of the other coordinate systems, then the meaning of the ra and dec parameters changes according to the following table:

<table>
<thead>
<tr>
<th>COORD_TYPE</th>
<th>RA parameter</th>
<th>DEC parameter</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>sky</td>
<td>Right Ascension</td>
<td>Declination</td>
<td>deg</td>
</tr>
<tr>
<td>tanxy</td>
<td>Tan(theta_x)</td>
<td>Tan(theta_y)</td>
<td>none</td>
</tr>
<tr>
<td>cartesian</td>
<td>BAT_X position</td>
<td>BAT_Y position</td>
<td>cm</td>
</tr>
<tr>
<td>unit</td>
<td>BAT_X unit vector</td>
<td>BAT_Y unit vector</td>
<td>none</td>
</tr>
<tr>
<td>fswlonlat</td>
<td>Phi (flt. software)</td>
<td>Theta</td>
<td>deg</td>
</tr>
<tr>
<td>grmclonlat</td>
<td>Longitude (GRMC)</td>
<td>Latitude</td>
<td>deg</td>
</tr>
</tbody>
</table>

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).
D.14.5 DISTORTION CORRECTION

Batmaskwtevt can correct for systematic non-linear centroid shifts in the BAT imaging system. It is assumed that the input position (either celestial or instrumental) is in “true” undistorted coordinates. All distortion corrections occur internally within the task. The distortion correction occurs if the ‘distfile’ parameter points to a correction map.

D.14.6 PARAMETERS

infile [filename ]
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.

attitude [string ]
File name of Swift attitude history, or NONE if none is used.

ra [real ]
Right ascension of source in decimal degrees, or other source position coordinate, depending on coord_type.

dec [real ]
Declination of source in decimal degrees, or other source position coordinate, depending on coord_type.

(aperture = “CALDB”) [filename ]
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. If several aperture types are available in the CALDB, the user can specify CALDB:apertype. Valid values of apertype are currently: FLUX or DETECTION, for apertures optimized for flux reproducibility or detection sensitivity, respectively.

(coord_type = “sky”) [string ]
Method of specifying the source coordinates. It should be one of: “sky”; “tanxy” (coordinates expressed in tangent-plane coordinates); “cartesian” (source at cartesian position in BAT coordinates); “unit” (unit vector direction in BAT coordinates); “fswlonlat” (flight software phi and theta); “grmclonlat” (source at simulator longitude and latitude). All coordinate types but the first are specified in the instrument coordinate system.

(rebalance = YES) [boolean ]
If YES, then the mask weight map will be adjusted so that its additive mean is zero. If NO, then, the map will be written unadjusted.

(corrections = “default”) [string ]
Comma-separated list of corrections to apply to the mask weighting, or “none” if no corrections are to be applied. The possible corrections are:

  default Default corrections, which is shorthand for: “flatfield,ndets,pcode,maskwt”
  flatfield Apply corrections for cosine and edge projection effects
**maskwt**  Apply corrections for mask weighting technique

**pcode**  Apply partial coding corrections

**ndets**  Normalize by number of exposed detectors

**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.

**(pcodethresh = 0.00) [real ]**

Minimum allowable partial coding value allowed for the source position. This is a fraction between 0.0 (no partial coding) and 1.0 (fully coded). For example, a value of 0.01 implies that at least 1% of the detectors must be illuminated. Mask weights for events whose partial coding is below the threshold will be set to zero.

**(detmask = “NONE”) [string ]**

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.

**(auxfile = “NONE”) [string ]**

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.

**(filtexpr = ”(EVENT_FLAGS == 0) —— (EVENT_FLAGS == 16)” ) [string ]**

Event filtering expression to be applied to the data. Any column filtering expression using the columns or keywords in the event file is valid (see the help for the “colfilter” task for more information). Events that fail the expression will have their mask weight set to zero. The default expression rejects calibration and multi-hit events, which are never due to cosmic sources. NOTE: users should not use spatial filters which exclude certain detectors, since batmaskwtevt will be unable to properly normalize the weights. For spatial filtering, use a detmask.

**(bat.z = 0) [real ]**

Position of the source in BAT.Z coordinates, if coord_type is “cartesian” (in units of cm).

**(maskwtcol = “MASK_WEIGHT”) [string ]**

Name of mask weight column.
(distance = 1.0e7) [real ]
Default bat_z position if bat_z is zero (centimeters).

(maskoffx = 0.0) [real ]
Translational offset to apply to the mask along the BAT_X coordinate from its nominal design position.

(maskoffy = 0.0) [real ]
Translational offset to apply to the mask along the BAT_Y coordinate from its nominal design position.

(maskoffz = 0.0) [real ]
Translational offset to apply to the mask along the BAT_Z coordinate from its nominal design position.

(maskthickness = INDEF) [string ]
Thickness of the lead mask tiles in centimeters. Normally this will be set to INDEF, which reads the mask thickness from the aperture file. A thickness of 0 cm is permitted.

(distfile=CALDB) [string ]
The ‘distfile’ parameter should point to a FITS file containing an MxNx2 image (or CALDB to use the default instrument correction in the calibration database). The two planes of the image are the non-linear distortion of the BAT “plate scale” as a function of position in the sky image. The values are offsets in tangent plane coordinates (IMX,IMY), and therefore unitless. The first plane of the image cube is the IMX offset, and the second plane is the IMY offset. The sense of the offset is (TRUE-APPARENT), i.e. for a measured position in tangent plane coordinates, the offset values should be *added* to arrive at the true position in tangent plane coordinates. The images are low-resolution versions of the BAT field of view in instrumental sky coordinates, and are meant to be interpolated to the desired sampling. The WCS keywords must specify the image coordinate systems.

(buffersize = 32768) [integer ]
Size of internal event buffer for processing.

(subpixsize = 0.02) [real ]
Size of detector subpixels used in computing mask weighting, in centimeters. Can be no larger than 0.02 cm.

(teldef = “CALDB”) [string ]
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.

(origin_z = 0) [real ]
For ground testing with a near-field source. The position of the origin of coordinates (in cm), to which the source position is referred.

(maskwtswgain = 0.04) [real ]
Mask weight technique software gain correction factor. When corrections=maskwt, the weights are divided by (1+maskwtswgain). This is correction is software algorithm dependent and not intended to be changed by the user.
(chatter = 2) [integer, 0 - 5]

Controls the amount of informative text written to standard output. Setting chatter = 1 produces a basic summary of the task actions; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints debugging information.

(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.

**D.14.7 EXAMPLES**

1. Computes the mask weights for a source at (RA,DEC) = (257.5500,-28.118) degrees. The event file sw00000001000bnone028.unf has its MASK_WEIGHT column overwritten.

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

2. Computes mask weights for a source which is fixed in detector coordinates. The tangent-plane coordinates are provided (tan(theta_x) and tan(theta_y)), with the source at a distance of 10^7 cm = 100 km. Because no attitude file is needed, NONE is specified.

```bash
batmaskwtevt sw00000001000bnone028.unf NONE 0.348 0.112
coord_type=tanxy distance=1e7
```

3. Computes mask weights for a source which is fixed in detector coordinates. The X and Y components of the unit vector are given; the Z unit vector is computed implicitly by batmaskwtevt, so that the full unit vector is (0.348, 0.112, 0.931). The source is placed at a distance of 10^7 cm = 100 km.

```bash
batmaskwtevt sw00000001000bnone028.unf NONE 0.348 0.112
coord_type=unit distance=1e7
```

**D.14.8 SEE ALSO**

batmaskwting, colfilter

**D.14.9 LAST MODIFIED**

Sep 2006
D.15 BATMASKWTIMG

D.15.1 NAME

batmaskwtimg - compute mask weights for the entire detector BAT plane

D.15.2 USAGE

batmaskwtimg outfile attitude ra dec

D.15.3 DESCRIPTION

batmaskwtimg constructs a mask weighting map, for a particular source position, for the entire BAT detector plane. This is the first step to computing a flux or spectrum for a source from survey type or image type data. The output is an image of the detector plane with one mask weight for each detector.

Users can also compute a “forward” ray tracing map, which gives the expected counts in each detector based on a unit intensity source, accounting for various geometric optics effects. Users should use "corrections=forward,unbalanced,flatfield" and "rebalance=no" in order to simulate such a map.

Users can also request analysis for a batch of sources in a catalog, using the ‘incatalog’ parameter. The catalog must provide at least the source position (and optionally the name of the source and an identifying number). By default, the output will contain multiple extensions, one for each source. However, individual catalog maps can be combined into one map with the combmeth parameter. **NOTE:** If the input catalog is empty, then no output catalog will be created. If ‘clobber=yes’ is set and the input catalog is empty, then any pre-existing output file will be erased.

The actual output map type is controlled by the ‘outtype’ parameter. By default it is WEIGHTS, indicating to output the ray traced mask weights. If outtype=ZERO or outtype=NONZERO is chosen, then a “footprint” mask is made instead, which shows which parts of the aperture (or non-aperture) illuminates the array, regardless of the aperture pattern. This may be especially useful in combination with combmeth="MIN" or “MAX”, to mask out portions of the array illuminated by particular sources.

By default, the coordinates are right ascension and declination. Users must also provide the instrument aperture and teldef files (or use CALDB). Instrument coordinates can also be used, as described in ADVANCED COORDINATE SYSTEMS below.

D.15.4 ADVANCED COORDINATE SYSTEMS

Users may specify a source position using a coordinate system other than sky position. Depending on the value of the coord_type parameter, the user may give the cartesian or angular position of the source in instrument coordinates. When coord_type is changed from “sky” to any of the
other coordinate systems, then the meaning of the ra and dec parameters changes according to the following table:

<table>
<thead>
<tr>
<th>COORD_TYPE</th>
<th>RA parameter</th>
<th>DEC parameter</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>sky</td>
<td>Right Ascension</td>
<td>Declination</td>
<td>deg</td>
</tr>
<tr>
<td>tanxy</td>
<td>Tan(theta_x)</td>
<td>Tan(theta_y)</td>
<td>none</td>
</tr>
<tr>
<td>cartesian</td>
<td>BAT_X position</td>
<td>BAT_Y position</td>
<td>cm</td>
</tr>
<tr>
<td>unit</td>
<td>BAT_X unit vector</td>
<td>BAT_Y unit vector</td>
<td>none</td>
</tr>
<tr>
<td>fswlonlat</td>
<td>Phi (flt. software)</td>
<td>Theta</td>
<td>deg</td>
</tr>
<tr>
<td>grmclonlat</td>
<td>Longitude (GRMC)</td>
<td>Latitude</td>
<td>deg</td>
</tr>
</tbody>
</table>

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).

### D.15.5 DISTORTION CORRECTION

Batmaskwtevt can correct for systematic non-linear centroid shifts in the BAT imaging system. It is assumed that the input position (either celestial or instrumental) is in “true” undistorted coordinates. All distortion corrections occur internally within the task. The distortion correction occurs if the ‘distfile’ parameter points to a correction map.

### D.15.6 PARAMETERS

**outfile [filename ]**

Output image file name. The mask weight map will be written to the primary extension of this file.

**attitude [string ]**

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

**ra [real ]**

Right ascension of source in decimal degrees, or other source position coordinate, depending on coord_type. This value is ignored if incatalog is specified.

**dec [real ]**

Declination of source in decimal degrees, or other source position coordinate, depending on coord_type. This value is ignored if incatalog is specified.

**(infile = “NONE”) [string ]**

The name of an input file containing valid Swift/BAT time and instrument keywords. For example, an existing detector plane image. The time and instrument keywords are used to select the proper CALDB files, and also to determine the TSTART/TSTOP time for the output file. The keywords in infile are also cloned to the output file. If given, the ‘time’ parameter overrides TSTART/TSTOP in infile. By default the midpoint time is used to compute the spacecraft attitude.
(aperture = “CALDB”) [filename ]

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. If several aperture types are available in the CALDB, the user can specify CALDB: apertype. Valid values of apertype are currently: FLUX, DETECTION, or MASK_EDGES; for apertures optimized for flux reproducibility, detection sensitivity, or for the shadow of the edge of the mask, respectively.

(coord_type = “sky”) [string ]

Method of specifying the source coordinates. It should be one of: “sky”; “tanxy” (coordinates expressed in tangent-plane coordinates); “cartesian” (source at cartesian position in BAT coordinates); “unit” (unit vector direction in BAT coordinates); “fswlonlat” (flight software phi and theta); “grmclonlat” (source at simulator longitude and latitude). All coordinate types but the first are specified in the instrument coordinate system.

(time = 0) [real ]

Time associated with the mask weighted image. When coordinates are used, it is also possible for the attitude to change as a function of time. Specify the mission elapsed time of interest with this parameter. A value of 0 (the default) indicates to use the first row of the attitude file, or the time in ‘infile’ if it is present. Ignored for non-sky coordinate systems.

(rebalance = YES) [boolean ]

If YES, then the mask weight map will be adjusted so that its additive mean is zero. If NO, then, the map will be written unadjusted.

(bat_z = 0) [real ]

Position of the source in BAT_Z coordinates, if coord_type is “cartesian” (in units of cm).

(incatalog = “NONE”) [string ]

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.

(racol = “RA_OBJ”) [string ]

Column name of the first coordinate value in the input catalog, typically right ascension.

(deccol = “DEC_OBJ”) [string ]

Column name of the second coordinate value in the input catalog, typically declination.

(namecol = “NAME”) [string ]

Column name of the source name in the input catalog. This value is written to the output keyword named ‘OBJECT’. If NONE is specified, then this column is ignored.

(catnumcol = “CATNUM”) [string ]

Column name of the source catalog number in the input catalog. This value is written to the output keyword named ‘CATNUM’. If NONE is specified, then this column is ignored.

(outtype = “WEIGHTS”) [string ]

Type of output map (one of WEIGHTS, ZERO or NONZERO). For ZERO, the test (weight == 0) is performed on each pixel; for NONZERO, the test (weight != 0) is performed. (1=true; 0=false).
**APPENDIX D. BAT SOFTWARE TOOL REFERENCE**

(combmeth = "NONE") [string ]

Method of combining multiple maps (one of NONE, MIN, MAX or SUM). If NONE, then multiple maps are output individually. If MIN/MAX/SUM, the minimum/maximum/total value is output for each pixel.

(distance = 1.0e7) [real ]

Default bat.z position if bat.z is zero (centimeters).

(maskoffx = 0.0) [real ]

Translational offset to apply to the mask along the BAT.X coordinate from its nominal design position.

(maskoffy = 0.0) [real ]

Translational offset to apply to the mask along the BAT.Y coordinate from its nominal design position.

(maskoffz = 0.0) [real ]

Translational offset to apply to the mask along the BAT.Z coordinate from its nominal design position.

(maskthickness = INDEF) [string ]

Thickness of the lead mask tiles in centimeters. Normally this will be set to INDEF, which reads the mask thickness from the aperture file. A thickness of 0 cm is permitted.

(distfile=CALDB) [string ]

The ‘distfile’ parameter should point to a FITS file containing an MxNx2 image (or CALDB to use the default instrument correction in the calibration database). The two planes of the image are the non-linear distortion of the BAT “plate scale” as a function of position in the sky image. The values are offsets in tangent plane coordinates (IMX,IMY), and therefore unitless. The first plane of the image cube is the IMX offset, and the second plane is the IMY offset. The sense of the offset is (TRUE-APPARENT), i.e. for a measured position in tangent plane coordinates, the offset values should be *added* to arrive at the true position in tangent plane coordinates. The images are low-resolution versions of the BAT field of view in instrumental sky coordinates, and are meant to be interpolated to the desired sampling. The WCS keywords must specify the image coordinate systems.

(corrections = “default”) [string ]

Comma-separated list of corrections to apply to the mask weighting, or “none” if no corrections are to be applied. The possible corrections are:

- **default** Default corrections, which is shorthand for: “flatfield,ndets,pcode,maskwt”
- **flatfield** Apply corrections for cosine and edge projection effects
- **maskwt** Apply corrections for mask weighting technique
- **pcode** Apply partial coding corrections
- **ndets** Normalize by number of exposed detectors
- **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.

(detmask = “NONE”) [string]

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 of zero for all bad quality pixels.

detmask.

(subpixsize = 0.02) [real]

Size of detector subpixels used in computing mask weighting, in centimeters. Can be no larger than 0.02 cm.

(teldef = “CALDB”) [string]

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.

(gapval = 0.0) [real]

Value to place in image cells where detector gaps are located.

(origin_z = 0) [real]

For ground testing with a near-field source. The position of the origin of coordinates (in cm), to which the source position is referred.

(maskwtswgain = 0.04) [real]

Mask weight technique software gain correction factor. When corrections=maskwt, the weights are divided by (1+maskwtswgain). This is correction is software algorithm dependent and not intended to be changed by the user.

(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. Setting chatter = 1 produces a basic summary of the task actions; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints debugging information.

(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.
D.15.7 EXAMPLES

1. Computes the mask weights for a source at (RA, DEC) = (257.5500, -28.118) degrees. The output is sent to the image maskwt.img.

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

2. Computes mask weights for a source which is fixed in detector coordinates. The tangent-plane coordinates are provided (tan(theta_x) and tan(theta_y)), with the source at a distance of $10^7$ cm = 100 km. Because no attitude file is needed, NONE is specified.

   `batmaskwtimg maskwt.img NONE 0.348 0.112 coord_type=tanxy distance=1e7`

3. Computes mask weights for a source which is fixed in detector coordinates. The X and Y components of the unit vector are given; the Z unit vector is computed implicitly by batmaskwtimg, so that the full unit vector is (0.348, 0.112, 0.931). The source is placed at a distance of $10^7$ cm = 100 km.

   `batmaskwtimg maskwt.img NONE 0.348 0.112 coord_type=unit distance=1e7`

D.15.8 SEE ALSO

batmaskwtevt

D.15.9 LAST MODIFIED

Oct 2006
D.16 BATOCCULTGTI

D.16.1 NAME

batoccultgti - compute BAT good time intervals due to occultation

D.16.2 USAGE

batoccultgti infile outfile ra dec

D.16.3 DESCRIPTION

batoccultgti computes the occultation good time intervals for a particular source during a BAT observation. Occultation of the earth, moon and/or sun can be considered. The BAT field of view is large enough that any of these bodies may occult cosmic sources. Note that this task is used for a single source, usually for data selection. If operating on a sky image, the batoccultmap task should be used.

Users can also check whether a source is within the X% partially coded field of view of the BAT using the “FOV” constraint. Users need to simply specify a partial coding threshold. However, users should be warned that the coordinate transformations and partial coding levels are not exact (to within a few degrees).

In addition to the partial coding check, users can also constrain the target to be in a certain position in the field of view, either in tangent plane coordinates (using the im\{x,y\}\{min,max\} parameters), or in polar coordinates (theta,phi).

Normally, the user must supply the “SAO” file created by prefilter. This file contains the positions of the earth, moon and sun as a function of time. The “sizes” of the earth, moon and sun are configurable in this task. If the user is only performing a field of view check (i.e. not the earth, moon or sun), then it is possible to use a Swift attitude file (“sw*sat*.fits”), along with the att_type=”ATT” parameter.

This task calls the FTOOLS ‘maketime’ and ‘ftcopy’, which must be present.

D.16.4 PARAMETERS

infile [filename ]

The name of the “SAO” file produced by prefilter. This file should contain the columns TIME, RA, DEC, ROLL, SAT_ALT, MOON_RA, MOON_DEC, SUN_RA, SUN_DEC, EARTH_RA, EARTH_DEC. The file may also be a Swift attitude file if only field of view constraints are being checked, and if att_type=“ATT”.

outfile [filename ]

The name of the output good time interval file. This is a standard OGIP-compatible GTI file.
ra [real ]

The right ascension of the target in degrees.

dec [real ]

The declination of the target in degrees.

(atmdepth = 5.0) [real ]

Thickness of the earth “atmosphere” in kilometers. Points within this thickness are assumed to have zero exposure.

(rearth = 6378.1) [real ]

The effective radius of the earth in kilometers. The default value is the mean equatorial radius (a conservative number).

(rmoon = 16.0) [real ]

The effective radius of the moon in arcminutes. The default value is conservatively set at 0.5 arcminutes larger than the true value.

(rsun = 16.5) [real ]

The effective radius of the sun in arcminutes. The default value is conservatively set at 0.5 arcminutes larger than the true value.

(constraints = “EARTH,MOON,SUN”) [string ]

A comma-separated list of the constraints to check (any of EARTH, MOON, SUN and FOV). “FOV” refers to a BAT field of view check; if it is specified then tempregionfile and tempex-prfile must also be given.

imxmin = “INDEF” [string ]

Minimum value for the image (tangent plane) IMX coordinate for the target. INDEF indicates no selection.

imxmax = “INDEF” [string ]

Maximum value for the image (tangent plane) IMX coordinate for the target. INDEF indicates no selection.

imymin = “INDEF” [string ]

Minimum value for the image (tangent plane) IMY coordinate for the target. INDEF indicates no selection.

imymax = “INDEF” [string ]

Maximum value for the image (tangent plane) IMY coordinate for the target. INDEF indicates no selection.

thetamin = “INDEF” [string ]

Minimum value for the image THETA angle (angle of the target from the boresite). INDEF indicates no selection.

thetamax = “INDEF” [string ]

Maximum value for the image THETA angle (angle of the target from the boresite). INDEF indicates no selection.
Enter “YES” to find the unocculted intervals. In the unlikely event that you are interested in the occulted intervals, enter “NO”.

Name of a temporary file used when performing the field of view (FOV) check. A region file is written out containing the shape of the BAT field of view in tangent plane coordinates at the desired partial coding threshold. A value of NONE indicates that no file is required. A value of INDEF indicates that a scratch file should automatically be created and deleted as needed for this purpose. If an explicit file name is given, the user is responsible for deleting the file.

Name of a temporary file used for making intermediate coordinate transformation calculations for the field of view (FOV) check. A value of NONE indicates that no file is required. A value of INDEF indicates that a scratch file should automatically be created and deleted as needed for this purpose. If an explicit file name is given, the user is responsible for deleting the file.

Partial coding threshold, as a fraction of unity. If the FOV check is enabled, then times are considered good only if the target is partially coded at the level of pcdethresh or higher.

Name of file to be used for intermediate calculations in the FOV check. Various columns, containing coordinate transformation and BAT tangent plane coordinates, are appended to the SAO file. A value of “NONE” indicates that an intermediate file should not be created.

Set to “SAO” if the input file is a prefilter (SAO) file, or “ATT” if the input file is a Swift attitude file.

Pre-Time Interval factor passed to the ‘maketime’ task.

Post-Time Interval factor passed to the ‘maketime’ task.

If the output file already exists, then setting “clobber = yes” will cause it to be overwritten.

Controls the amount of informative text written to standard output. Setting chatter = 1 produces a basic summary of the task actions; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints debugging information.

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.
D.16.5 EXAMPLES

1. Computes a good time interval for the Crab.

    batoccultgti sw00111622000sao.fits.gz crab.gti 83.6332 22.0145

D.16.6 SEE ALSO

batoccultmap, maketime

D.16.7 LAST MODIFIED

Sep 2006
D.17 BATOCCLUDMAP

D.17.1 NAME
batoccultmap - compute BAT exposure mask due to occultation

D.17.2 USAGE
batoccultmap infile outfile saofile

D.17.3 DESCRIPTION
batoccultmap computes the fractional exposure in the BAT field of view due to occultation by the earth, moon and/or sun. The BAT field of view is large enough that any of these bodies may occult cosmic sources. Since the bodies move with time, the task computes the exposure map as a function of time and merges the results. This task is used for sky images; if you want to data selection for the visibility of a single source, use batoccultgti.

The task can be run in two modes. If occultation=“any”, then any portion of the image which is blocked at any time is masked in the output. If occultation=“fraction”, then the fractional exposure is computed (a value between 0 and 1).

The user must supply the “SAO” file created by prefilter. This file contains the positions of the earth, moon and sun as a function of time. The sizes of the earth, moon and sun are configurable.

D.17.4 PARAMETERS

infile [filename ]
The name (and optional extension) of the input image file name.

outfile [filename ]
The name of the output map file. For occultation=“fraction”, the map will range continuously from 0.0 to 1.0, corresponding to the partial exposure. For occultation=“any”, the map will be either 0.0 (occulted at some time) or 1.0 (never occulted).

saofile [filename ]
The name of the “SAO” file produced by prefilter. This file should contain the columns TIME, RA, DEC, ROLL, SAT_ALT, MOON_RA, MOON_DEC, SUN_RA, SUN_DEC, EARTH_RA, EARTH_DEC.

atmdepth = 5.0 [real ]
Thickness of the earth “atmosphere” in kilometers. Points within this thickness are assumed to have zero exposure.
rearth = 6378.1 [real ]
   The effective radius of the earth in kilometers. The default value is the mean equatorial
   radius (a conservative number).

rmoon = 16.0 [real ]
   The effective radius of the moon in arcminutes. The default value is conservatively set at 0.5
   arcminutes larger than the true value.

rsun = 16.5 [real ]
   The effective radius of the sun in arcminutes. The default value is conservatively set at 0.5
   arcminutes larger than the true value.

method = “position” [string ]
   Time segmentation method, used for subdividing good time intervals. For method=“time”,
   good time intervals are divided into subintervals no larger than timesegtol seconds. For
   method=“position”, good time intervals are divided so that the spacecraft and earth/moon/sun
   move no more than timesegtol arcminutes.

timesegtol = 0.0 [real ]
   Time segmentation parameter (see “method” above). Generally smaller values will provide
   higher fidelity, but take longer to compute. A value of zero indicates the defaults of 3 arcmin
   / 10 seconds should be used.

occultation [string ]
   Either “any” or “fraction”. See “outfile” above.

constraints = “EARTH” [string ]
   A comma-separated list of the constraints to check (EARTH, MOON and/or SUN). Note
   that automatic inclusion of the MOON and SUN is not recommended, because both are
   comparable to the size of a BAT point source and may confuse the source detection software.

gtifile = “INFILE” [string ]
   Name of file containing the good time interval extension. The file is searched for the first
   extension matching “*GTI*”. A value of “INFILE” indicates to search the input file. A value
   of “NONE” indicates to use the TSTART/TSTOP image header keywords.

(multfiles = “NONE”) [string ]
   Comma-separated list of image filenames to be corrected using the calculated correction image
   (or “@filename.lis” to supply a list of files in filename.lis). Images are corrected IN PLACE
   (i.e. no back-up is made) by MULTIPLYING the original image values by the correction.

(divfiles = “NONE”) [string ]
   Comma-separated list of image filenames to be corrected using the calculated correction image
   (or “@filename.lis” to supply a list of files in filename.lis). Images are corrected IN PLACE
   (i.e. no back-up is made) by DIVIDING the original image values by the correction.

(algorithm = “CONTOUR”) [string ]
   Name of algorithm to use, either IMAGE or CONTOUR. The IMAGE method computes
   visibility for each pixel; the CONTOUR method computes the contours of each object in pixel
   coordinates. By virtue of the algorithms, the two results are slightly different (CONTOUR
   produces slightly larger occultation regions).
(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. Setting chatter = 1 produces a basic summary of the task actions; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints debugging information.

(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.

D.17.5 EXAMPLES

1. Computes the fractional exposure map for the primary HDU of sky.img. Good time intervals are subdivided so that the spacecraft attitude/earth/moon/sun do not move more than 3 arcmin (the default).

   batoccultmap sky.img occult.img sw00111622000sao.fits.gz \
   method=position occultation=fraction

D.17.6 SEE ALSO

batoccultgti

D.17.7 LAST MODIFIED

Jun 2006
D.18 BATPHASYSERR

D.18.1 NAME

batphasyserr - apply BAT spectral systematic error vector

D.18.2 USAGE

batphasyserr infile syserrfile

D.18.3 DESCRIPTION

batphasyserr adds the BAT systematic error vector to a BAT spectral file. The systematic error vector contains an estimate of the fractional systematic error in each BAT spectral channel. Normally this information comes from the BAT calibration database (CALDB).

The task handles the differences between type I and type II spectral files transparently. It also resamples the systematic error template to the energy bin edges of the input spectrum (this is done by pure averaging).

D.18.4 PARAMETERS

infile [filename]

Input/output spectral file. The file should be an OGIP-standard type I or type II spectral file. The DATE-OBS keyword is used to query the CALDB. The file is modified in place. If necessary, the SYS_ERR column is created, and the SYS_ERR keyword is deleted.

syserrfile [filename]

Name of the systematic error vector file to use. Normally this should be set to “CALDB” so that the current best estimate of the BAT spectral systematic error is used for the date of the given infile.

(chatter = 2) [integer, 0 - 5]

Controls the amount of informative text written to standard output.

D.18.5 EXAMPLES

1. Create the SYS_ERR column in a spectral file named myspectrum.pha.

   batphasyserr myspectrum.pha CALDB

   The systematic error vector is retrieved from the calibration database.
D.18. **BATPHASYSERR**

D.18.6 **SEE ALSO**

batupdatephakw

D.18.7 **LAST MODIFIED**

Apr 2006
D.19  BATTBLOCKS

D.19.1  NAME

battblocks - estimate interesting GTIs for time-variable data

D.19.2  USAGE

battblocks infile outfile

D.19.3  DESCRIPTION

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)

The user provides event data or a single-channel binned light curve. 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 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 can also choose their own percentage fluence with the ‘txx’ keyword; the result is
stored in the GTI_TXX extension.

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 ‘spill’ parameter to decimate the events, at the expense of decreasing the time resolution proportionately.

D.19.4 PARAMETERS

infile [filename]
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.

outfile [filename]
Name of output good time interval file.

(durfile = ‘NONE’) [filename]
Name of output duration measures. If durfile is ‘NONE’ then the duration measures are not computed or written. GTI tables with extensions ‘GTI_T90’, ‘GTI_T50’ and ‘GTI_PEAK’ are created.

(nspill = 128) [integer]
Number of events to skip per analysis cell. ‘nspill’ events are grouped into one cell before Bayesian analysis. Ignored for binned data.

(gaussian = INDEF) [string]
Boolean which determines whether the Bayesian blocks computation is performed using Gaussian statistics or not. A value of INDEF causes battblocks to use Gaussian statistics for background-subtracted light curves, and Poissonian statistics otherwise. If “yes”, then the ERROR column must exist in the input file. Ignored for event data.

(txx = 0.0) [real]
User-specified percentage of burst to estimate the duration for. By default 90% (GTI_T90) and 50% (GTI_T50) fluence intervals are computed, but users can choose another percentage point as well (stored as GTI_TXX). No interval is estimated of txx is zero.

(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. Smaller values will run faster, at the expense of generating more blocks (no block will be larger than tlookback). The lookback time is converted to a number of samples by examining the mean sample rate. A value of 0.0 indicates the global bayesian block optimization should be performed.
(timedel = 100e-6) [real ]

Internal time quantization size. Time resolutions more finely sampled than timedel are lost.

(ncp_prior = 6.0) [real ]

Log parameter prior for the number of “change points” between Bayesian blocks. This parameter acts as a smoothing parameter which weights against separate blocks. A smaller number permits less significant changes to be “detected,” while a larger number requires more significant changes to be “detected.”

(bkgsnb = NO) [boolean ]

Automatic background subtraction, for computation of duration measures and fluence estimates.

(coalescefrac = 0.05) [real ]

The background subtraction and T50/T90 algorithms depend on the first and last Bayesian blocks being representative of the true local background. If there is an outlier block, this may heavily skew the background and duration calculations. If this parameter is non-zero, and if the first or last block has a smaller duration than coalescefrac times its neighbor’s duration, then the two blocks are coalesced into one. Note that this will erroneously ignore real variations if they fall at the extremes of the time series. Also, the coalescence is not iterative; the check is done only once.

(timecol = “TIME”) [string ]

Name of the light curve time column.

(countscol = “INDEF”) [string ]

Name of the light curve rate or counts column. For a value of INDEF, battblocks will search for RATE and COUNTS columns, in that order.

(errcol = “ERROR”) [string ]

Name of the light curve gaussian error column. Not used for event or Poisson light curves.

(expocol = “INDEF”) [string ]

Name of the light curve per-bin exposure column. For a value of INDEF, battblocks searches for the TIMEDEL and EXPOSURE columns, in that order. If expocol/TIMEDEL/EXPOSURE is not found, then the file is searched for a TIMEDEL or EXPOSURE keyword.

(hduclas3 = “INDEF”) [string ]

Default value of the HDUCLAS3 keyword, if one is not present in the file. If both hduclas3 and countscol are “INDEF”, then battblocks will search for a RATE or COUNTS column and set hduclas3 according to which column is found.

(diagfile = NONE) [string ]

Diagnostic output file, used for internal debugging purposes. NONE indicates no diagnostic output should be made.

(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. Setting chatter = 1 produces terse output; chatter = 2 (default) additionally prints a summary of input parameters; chatter = 5 prints debugging information.

(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.

D.19.5 EXAMPLES

1. Partition data into Bayesian Blocks

   battblocks burst.evt burst_bb.gti

2. Same as example 1, but with further event decimation

   battblocks burst.evt burst_bb.gti nspill=256

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

D.19.6 REFERENCES


D.19.7 LAST MODIFIED

Mar 2006
D.20 BATUPDATEPHAKW

D.20.1 NAME

batupdatephakw - update BAT ray tracing columns in spectral files

D.20.2 USAGE

batupdatephakw infile auxfile

D.20.3 DESCRIPTION

batupdatephakw updates the BAT ray tracing columns in a mask weighted spectrum. This is important especially for gamma-ray bursts or other events where the spacecraft has slewed. Without the update, the keywords will reflect the ray tracing status at the end of the event data set, which will lead to an erroneous detector response matrix.

The ray tracing quantities that are updated are: MSKWTSQF, NGOODPIX, PCODEFR, BAT_XOBJ, BAT_YOBJ, BAT_ZOBJ.

This script modifies the spectral file in place. Thus, the spectrum must be writeable and not compressed. New columns are created if they are not already present. If keywords with the same name are present, then they are deleted (to prevent confusion with the columns).

This task requires the “auxiliary” file produce by batmaskwtevt, which contains the above quantities as a function of time.

D.20.4 PARAMETERS

infile [filename]

Input/output spectral file. The file should be a batbinevt-derived OGIP standard Type I or Type II spectral file. The file should contain at least the TIME and EXPOSURE keywords. The file is modified in place.

auxfile [filename]

Name of the “auxiliary” ray trace file produced by batmaskwtevt. The times in the auxiliary file should overlap with the times in the spectral file.

(chatter = 2) [integer, 0 - 5]

Controls the amount of informative text written to standard output.

D.20.5 EXAMPLES

1. Update the ray tracing columns in a spectral file named myspectrum.pha.
batupdatephakw myspectrump.ha auxfile.fits

auxfile.fits is the “auxiliary” file produced by batmaskwtevt.

D.20.6 SEE ALSO

batmaskwtevt

D.20.7 LAST MODIFIED

Aug 2006
D.21 BATWARPIMG

D.21.1 NAME
batwarpimg - Correct BAT sky image for known imaging distortions

D.21.2 USAGE
batwarpimg infile outfile distfile=CALDB

D.21.3 DESCRIPTION
batwarpimg corrects BAT sky images for known imaging distortions. The distortion map is usually stored in the calibration database (CALDB), and describes the image shift in instrument coordinates. The input sky map is resampled with these slight shifts applied, so that sources appear at their “true” position.

Note that the interpolation process inherently involves some smoothing of the sky image, so a small amount of information is lost.

D.21.4 PARAMETERS

infile [filename ]
The name (and optional extension) of the input image file name. If an explicit image extension name is supplied, and rows='-', then only that extension will be processed.

outfile [filename ]
The name of the output image.

(rows = “-“) [string ]
List of images to be processed from the input file. The list must be comma-separated, and can have single elements or ranges. For example, “1,3-5,6”. The first image in the file is number 1.

(distfile = “CALDB”) [string ]
Name of the distortion map, or CALDB if users have access to the BAT calibration database. This must be an N × M × 3 image extension with proper WCS keywords, including the alternate “T” coordinate system in tangent plane coordinates. The first image plane should be the(true-apparent) position offset in the IMX direction; the second plane is the same for the IMY direction.

(outcoord=“TRUE”) [string ]
The output coordinate system. TRUE indicates to convert to “true” or corrected coordinates, while APPARENT indicates the reverse conversion. Note that because the conversion is lossy, a round trip pair of conversions may not equal the original.
(copyall="YES") [boolean ]
Indicates whether to copy all other extensions. If yes, then the unprocessed extensions are
copied untouched. If no, then only the processed extensions are saved.

(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. Setting chatter = 1
produces a basic summary of the task actions; chatter = 2 (default) additionally prints a
summary of input parameters; chatter = 5 prints debugging information.

(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.

D.21.5 EXAMPLES

1. Correct an image for distortions.

   batwarpimg input.fits output.fits distfile=CALDB

D.21.6 SEE ALSO

batwarpcoord

D.21.7 LAST MODIFIED

Apr 2006