LEVEL-2 PIPELINE SOFTWARE FOR ASTROSAT CADMIUM ZINC TELLURIDE IMAGER (CZTI) USER GUIDE

Dipankar Bhattacharya (IUCAA, Pune), Mithun N P S (PRL, Ahmedabad),
Ajay Vibhute (IUCAA, Pune), Rakesh Khanna A (TIFR, Mumbai),
Tanul Gupta (SAC, Ahmedabad), Preeti Tahliani (SAC, Ahmedabad),
Arvind K Singh (SAC, Ahmedabad)

February 6, 2017

Revision History

Release 1.0 03 June 2016 initial release DB
Release 1.1 05 October 2016 DB/Mithun

Contents

1 Introduction

The Cadmium Zinc Telluride Imager (CZTI) on-board ASTROSAT is a Coded Mask Imaging camera working in the energy range 15 to 150 keV. The detector is an array of a pixellated CZT detectors, and the Coded Mask is composed of a collection of pseudo noise Hadamard coded Uniformly Redundant Array (URA) patterns. The instrument is configured as four independent quadrants, each with 16 detector modules. Each detector module has 256 pixels. There are collimator slats surrounding every detector module and the Coded Aperture Mask is placed above the collimator. The instrument operates in event mode, wherein the time, energy and pixel coordinates of each photon event is recorded in the data. There is a CsI Veto detector under the CZT modules. If an event recorded in the Veto occurs within a certain small time window of any CZT event, then the latter is tagged with a Veto flag. The instrument also contains Alpha-tagged calibration sources, which generate 60 keV photons simultaneously with an Alpha particle. CZT events that fall within a defined time window of any alpha detection are marked with an Alpha tag in the recorded data.

Although the CZTI could be configured to work in a variety of modes, in normal course of operation data are acquired in only three of them. Mode M0 (Normal Mode) would contain the regular event mode data. Mode SS (Secondary Spectral Mode) operates in parallel with all other modes, and records, once every 100 seconds, accumulated spectral and housekeeping information. When the spacecraft passes through the South Atlantic Anomaly, High Voltage in the CZT and Veto detectors are switched off, and data collected in Mode M9 (SAA mode) wherein no events, but only housekeeping information is recorded once every second. Data telemetered to ground are accumulated into separate files for these different modes before delivery.

2 Overview of CZTI Data and Pipeline

The data received from the CZTI payload passes through a series of steps in a processing pipeline. Three major data levels have been designated for long-term storage. These are:

Level 0

This is the raw data received from satellite telemetry, which is segregated by instrument, along with auxiliary data. This data is archived internally and not distributed for public use.

Level 1

This is reorganized raw data, written in FITS format for Astronomical use. All auxiliary information necessary for further processing of this data are collated at this level and packed along with the respective science data. This data is released via Astrosat data archive, at first to the Principal Investigator (PI) of the corresponding observing proposal and, after a specified initial lock in period, to anyone interested in the data.

Level 2

This data contains standard science products derived from Level 1 data. Level 2 data is also in FITS format and is available for science use, with the same lock-in criteria and release mechanism as the Level 1 data.

The software to process data from Level 1 to Level 2 contains user configurable elements. While a default configuration is run at the Payload Operation Centre (POC) to create the automated Level 2 standard data products, the user has the option to generate more customized products by using the same software with other configuration settings. The Level 2 pipeline software is released publicly for general use. For Input/Output, the implementation of the pipeline software makes use of the CFITSIO and the Parameter Interface Library developed and distributed by NASA High Energy Astrophysics Science Archive Research Centre (heasarc.nasa.gov).

The functionality and usage of different components of the CZTI Level 2 pipeline software are described in this document.

3 Level-1 Data Content

The input to CZTI level-2 data reduction pipeline is the data set produced at Level 1. The original data is received by telemetry at the Indian Space Science Data Centre (ISSDC), during each visible pass over Bengaluru, India. This data is then packaged every download orbit and sent to the Payload Operations Centre (POC) for further processing. At the POC, all the data belonging to an Observation ID (ObsID) are then collected together from various orbit-wise packages and a merged Level 1 product is created for distribution and further analysis. The Level 2 pipeline analysis procedure described below starts from merged Level 1 package, which contains the following files:

1. **Science Data File**: a FITS file containing sequentially stacked 2048-byte data frames generated by the CZTI payload. This data is mode segregated, i.e. a different FITS file is generated for each distinct data mode.

2. **Time Calibration Table**(.tct): This file contains a list of time tags expressed in CZTI internal clock, Satellite On Board reference Time (OBT) and Universal Time (UT) derived from the SPS units on ASTROSAT. As all the CZTI science data is tagged with its internal clock, the TCT file is required to correlate them with other on-board events, as well as to obtain absolute timing.

3. **Orbit File**(.orb): Gives time-tagged orbital position of the satellite in terms of geocentric x, y, z as a function of time, as well as the corresponding velocity components.

4. **Attitude File**(.att): Gives time-tagged satellite attitude information, in terms of quaternions as well as the RA, Dec values of the pointing directions of the three reference axes of the satellite.

5. **LBT Housekeeping file**(.lbt): This gives a time-tagged recording of 65 different health parameters monitored by different sensors on the CZTI.

6. **MCAP file**(.mcap.xml): This xml file has information like source observed, start and end time of observation etc.

7. **Make Filter file**(.mkf): This file collects together the time series of a number of selected parameters, including health parameters, Sun angle, Earth angle, charged particle count etc, based on which the quality of data obtained can be assessed.

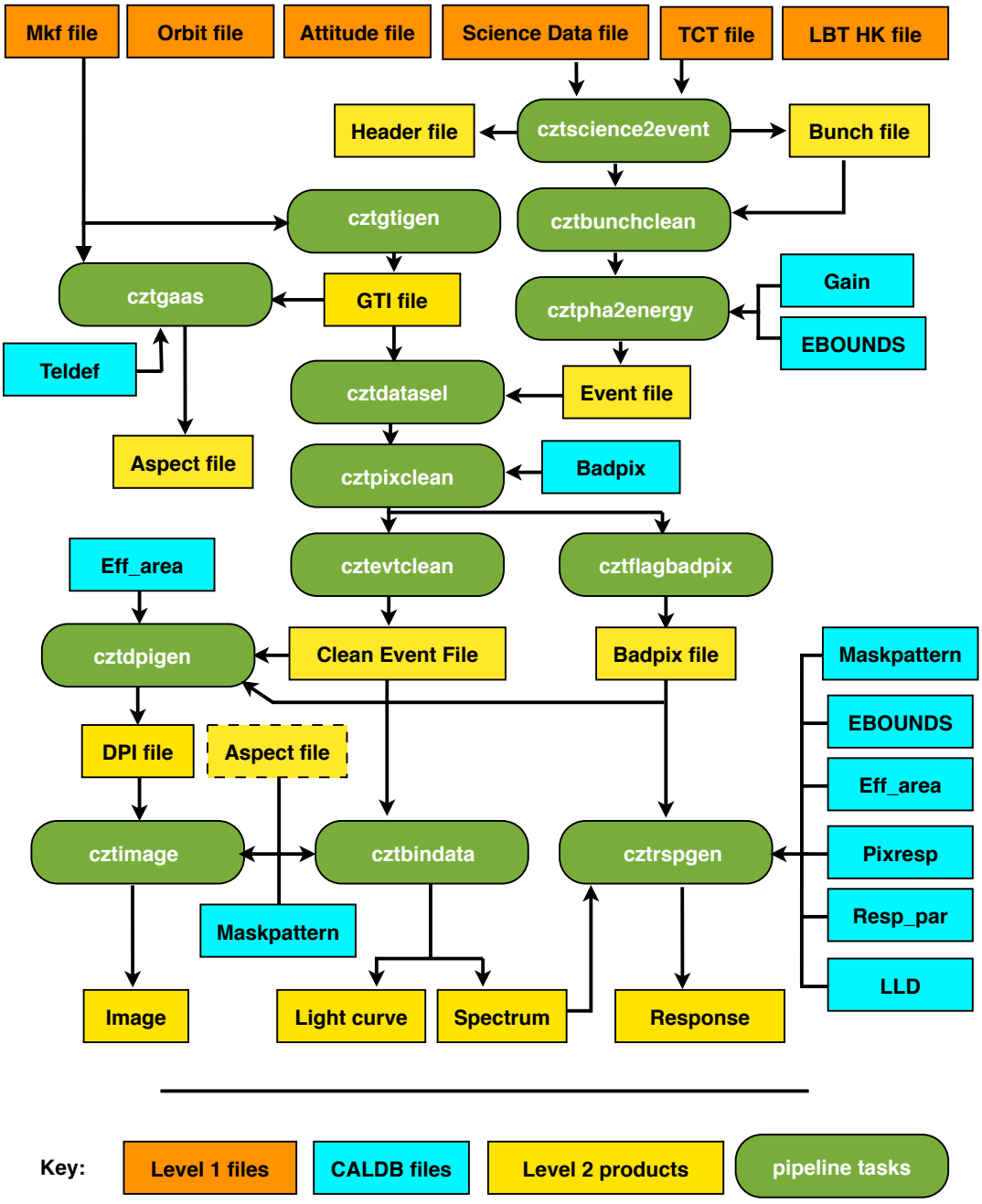

Figure 1: Level-2 pipeline work flow.

Sl No	Module name	Functionality
1	cztscience2event	a) Extraction and decoding of level-1 science data into events b) Mapping of quadid, detid and pixid to detx,dety c) Mapping of 12 bit PHA to 10 bit PHA channels d) Time calibration e) Recording of temperature history
2	cztbunchclean	Clean the event file by identifying and removing bunches
3	cztpha2energy	Convert PHA value to PI and write to event file
4	cztgtigen	Generate GTI based on current gti, mkf and user input
5	cztgaas	Compute time dependent and average aspect of CZTI
6	cztdatasel	Select events based on gti
7	cztpixclean	Identify noisy pixels and detectors and remove noisy events
8	cztflagbadpix	Combine bad pixel list from multiple sources, if required
9	cztevtclean	Select events based on veto and alpha tags
10	cztdpigen	Generate Detector Plane Image from clean event file
11	cztimage	Create image using Fast Fourier Transforms
12	cztbindata	Generate light curve/spectrum
13	cztrspgen	Generate response matrix file

Table 1: Summary of tasks in CZTI level-2 pipeline

4 Level-2 Pipeline Work flow

CZTI level-2 data reduction pipeline is organized into different modules for each specific task. In figure 1, work flow of the pipeline is shown. Table 1 summarizes the tasks performed by various software modules.

Data reduction for CZTI is performed in three stages. Each of these stages involves execution of several tasks of the pipeline.

- **Stage 1**: Generation of event file and calibration. In this stage event file is generated from level 1 data, gain offset corrections are applied and multi hit events are removed.
- **Stage 2**: Selection of data and cleaning. In this stage events are selected based on Good Time Interval and noisy pixel events are removed from the data.
- **Stage 3**: Generation of science products. In this last stage,DPI, image, spectrum, light curve and response matrix are generated.

The pipeline tasks can either be executed individually or by using the **cztpipeline** module which allows the user to run required stages of the pipeline tasks.

5 Download and Installation

CZTI data reduction pipeline is available in AstroSat Science Support Cell (SSC) website in the link given below:

http://astrosat-ssc.iucaa.in/uploads/czti/czti_pipeline_20160915_V1.1.tar

Please follow the instructions below to install the software.

Pre-Requisites

OS: Linux/Unix (CentOS-5.8+, Ubuntu 14.04+, OS X 10.10+, Scientific Linux 6.5+)
Compiler: gcc (GCC) 4.4+, Perl

Installation Instructions

1. Run the command 'bash' to change the shell to bash shell

2. Untar the package.

```
tar -xvf czti_pipeline.tar
```

This will create a directory named "czti_pipeline"

3. Set following environment variables in ~/.bashrc file

```
as1czt : Absolute path of czti folder( under "czti_pipeline/czti")
PFILES : Absolute path of paramfiles folder
PATH   : Absolute path of bin directory where all executables are placed
LD_LIBRARY_PATH : Path to shared libraries
```

Add following lines to ~/.bashrc file

```
export as1czt=<path to czti directory>
export PFILES=$as1czt/paramfiles
export PATH=$as1czt/bin:$as1czt/script:$PATH
export GLOG_log_dir=$as1czt/log
export CZTI_templates=$as1czt/templates
export PERL5LIB="$as1czt/lib/":$PERL5LIB
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$as1czt/lib
ulimit -s 65532
```

OS X users: In addition to above lines, specify DYLD_LIBRARY_PATH by adding the following line to ~/.bashrc.

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:$as1czt/lib
```

4. CALDB installation: Untar caldb_goodfiles_as1_czti*.tar.gz in a preferred location or under existing CALDB directory if caldb installation for other missions are present. It should produce directory structure data/as1/czti/, under which there would be caldb.indx file and calibration files.

Export the CALDB path by adding the below line in ~/.bashrc file

```
export CALDB=<path to CALDB directory>
```

5. Execute the following commands in bash

```
source ~/.bashrc
cd $as1czt
cd ../
./InstallLibs
cd $as1czt
make
cd scripts
chmod +x cztpipeline
```

The installation procedure is now complete.

6 Description of *cztpipeline*

cztpipeline is a perl script which executes the tasks of CZTI data analysis pipeline in sequence. The user can decide to run specific stages of data analysis and also can configure many of the parameters of individual tasks of the pipeline.

Input parameters of *cztpipeline* are described below. The default value of each of the parameters is given in square brackets. Note that many of these parameters are hidden and unless the user defines these parameters as input arguments in command line, the default values would be used. For further details of the parameters of specific tasks, please refer to the subsequent sections describing the software modules.

l1dir []

Path to level 1 data directory. This should point to the top level directory of level 1 extracted from the downloaded archive file. Please make sure not to change the name of the directory.

l2dir []

Location of level 2 data directory. *cztpipeline* will create a level 2 data directory under this location.

entrystage [1]

Entry stage for analysis. As described earlier, the data reduction has three distinct stages. This parameter defines the start stage of analysis. Minimum acceptable value is one and maximum is three.

exitstage [3]

Exit stage for analysis. Exit stage can be one, two or three, but the value should not be less than the entry stage. If user wishes to perform only one stage, say stage 3, both entry and exit stages should be set as three. Also note that execution of any stage requires products from earlier stages. By default all three stages of the pipeline would be executed.

clobber [yes]

Replace existing files. If clobber is set to yes, files existing in the location if any would be replaced. By defualt clobber is set to yes.

history [yes]

Write history to file header. This keyword controls writing of history parameters to header of the output FITS files. Users are adviced to keep it at the default.

debug [no]

Run in debug mode. This parameter controls chatter level. If set, the pipeline tasks would print debugging information.

eboundsfile [CALDB]

Ebounds file defining nominal energy range of pulse invariant channels. If set to CALDB, pipeline will pick the file from caldb distribution.

gainfile [CALDB]

Gain and offset file, which contains temperature dependant gain and offset values of detector pixels that are used to convert PHA to PI channels.If set to CALDB, pipeline will pick the file from caldb distribution.

effareafile [CALDB]

Effective area file which contains effective area of detector pixels. If set to CALDB, pipeline will pick the file from caldb distribution.

lldfile [CALDB]

Lower Level Discriminator channels of pixels, which defines the lowest responsive channel of each pixel. If set to CALDB, pipeline will pick the file from caldb distribution.

badpixfile [CALDB]

List of bad pixels, which has flags of disabled and spectroscopically bad pixels. If set to CALDB, pipeline will pick the file from caldb distribution.

maskfile [CALDB]

Coded mask pattern file, which defines the coded aperture mask pattern. If set to CALDB, pipeline will pick the file from caldb distribution.

camgeomfile [CALDB]

Camera geometry definition file, which defines the geometry of the instrument including the collimators and other structures. If set to CALDB, pipeline will pick the file from caldb distribution.

respparfile [CALDB]

Response parameter file having parameters of response model for pixels. This includes electron and hole mobility values, resolution etc. If set to CALDB, pipeline will pick the file from caldb distribution.

pixrespfile [CALDB]

Pixel wise redistribution matrix file, which stores pre-computed redistribution matrices for groups of pixels. If set to CALDB, pipeline will pick the file from caldb distribution.

Inputs of cztscience2event

sc2evt_buffersize [100000]

Buffer size for cztscience2event. The task cztscience2event reads data packets from level 1 directory and extracts events to output event file. This parameter controls the number of packets to be read at a time. User can change this parameter depending upon the memory available.

bigendian [yes]

Whether bigendian.

Inputs of cztbunchclean

bunchdeftime [30]

Bunch definition time. This parameter defines time difference between events which should be assigned as same bunch. Refer to section on cztbunchclean for more details.

bunch_length_thresh [20]

Threshold value for bunch length, which is defined as the number of events in a bunch. This parameter distinguishes short and long bunches.

skipT1 [0]

skipT1 parameter for bunchclean, which defines the time range for discarding events after every bunch.

skipT2 [0.001]

skipT2 parameter for bunchclean, which defines the time range for discarding events in the module after short bunches.

skipT3 [0.001]

skipT3 parameter for bunchclean, which defines the time range for discarding event in the quadrant after long bunches.

livetime_binsize [1]

Time binsize for livetime calculation. Multi-hits events (bunches) cause reduction in live-time. cztbunchclean computes livetime for each quadrant, and this parameter defines the size of time bin for which live time would be computed.

Inputs of cztgtigen

mkfthresholdfile [$as1czt/paramfiles/mkfThresholds.txt]

Threshold file for MKF parameters for GTI selection. This file defines thresholds on various parameters in MKF file which would be used for generation of Good Time Interval.

usergtifile [-]

Optional user GTI file. User can provide additional GTI file an input to select data in desired intervals. This file should be standard GTI FITS file with binary extension named GTI with two columns START and STOP defining start and stop times of the intervals.

Inputs of cztdatasel

gtitype [QUAD]

GTI selection type. Valid inputs are 'QUAD' and 'COMMON'. QUAD type generated GTI specific to each quadrant, whereas if COMMON is set, the output GTI would be intersection of Good Time intervals of all quadrants. In case the user wishes to add data from all four quadrants, 'COMMON' GTI type is suggested.

Inputs of cztpixclean

noisypix_nsigma [5]

Noisy pixel threshold. This parameter defines the definition of noisy pixels. Pixels with counts beyond noisypix_nsigma standard deviation from mean pixel count rate are removed.

dettbinsize [1]

Detector light curve bin size for pixclean. This defines bin size for light curves used for rejection of noisy times of detectors.

pixtbinsize [1]

Pixel light curve bin size for pixclean. This defines bin size for light curves used for rejection of noisy times of pixels.

detcount_threshold [35]

Detector count threshold. Time bins with detector counts beyond this threshold are rejected.

pixcount_threshold [2]

Pixel count threshold. Time bins with pixel counts beyond this threshold are rejected.

writedblevt [no]

Whether write double event file. If this parameter is set to yes, double event file with Compton scattered events would be written as additional output.

Inputs of cztevtclean

alphasel [0]

Alpha flag to be selected. If set to zero, cztevtclean rejects events with coincident alpha tag. Only alpha tagged events are selected if it is set to one, which would be useful for calibration only.

vetosel [0]

Veto range to be selected. If set to zero, events with coincident events in veto detector would be rejected.

Inputs of cztdpigen

dpi_badpixthreshold [0]

Threshold of badpixels in DPI generation. Pixels with flag less than or equal to this parameter would be used in the generation of Detector Plane Image.

dpi_timerange [-]

Time range for DPI. Defines the start and end time for generation of DPI.

dpi_energybins [-]

Energy range for DPI. Defines low and high energy limits for events to be used in generation of DPI.

Inputs of cztimage

image_inputtype [DPI]

Type of input for cztimage. Defines the type of input to cztimage. Vaid options are DPI, DPH, SHADOW.

Inputs of cztbindata

applymaskweight [yes]

Whether to apply maskweights. If set to yes, output spectrum and light curve will have mask weighted background subtracted counts. Counts in mask-weighted spectrum are "background subtracted counts per unit area of fully illuminated detector plane". If this parameter is set to no, spectrum and light curve would contain all events recorded.

rasrc [-]

Source Right Ascension. Right ascension of source of interest in FOV, given in decimal degrees.

decsrc [-]

Source Declination. Declination of source of interest in FOV, given in decimal degrees.

bindata_badpixthreshold [0]

Badpixel threshold for bindata. Pixels with badpix flag greater than this threshold are ignored while generating spectrum and light curve.

bindata_outputtype [both]

Output type for bindata. Valid options are spec, lc and both.

lc_timebinsize [1000.0]

Time bin size for light curve in seconds.

lc_energyrange [-]

Energy range for light curve. Low and high energy limits for events to be used for generation of spectrum and light curve.

applyLLD [yes]

Whether to apply LLD cut. If it is set to yes, events below LLD of pixels would be discarded while generating outputs.

generate_eventfile [yes]

Whether generate event file with mask weights. If set to yes, an event file identical to input event file with additional columns of mask open fraction and weight would be generated.

bindata_quadsToProcess [-]

This parameter decides which quadrants to be processed by cztbindata. Note that, in case the GTI type of input event file is 'QUAD', the spectrum and light curve would be produced for each of the quadrants separately and if it is 'COMMON' then data from all selcted quadrants will be added together.

7 Description of Software Modules

7.1 cztscience2event

This module reads the level 1 science data file and decodes the data packets into events. It creates an event file in FITS format with events for each quadrant in different extensions. Each event is assigned a calibrated time stamp which is computed using the time calibration table. Relevant housekeeping information present in headers and detector temperature information present in modeSS data are also decoded and written to output files in tabular form.

Input

- Science data file
- Tct file

Method

- Read the data packets from the Data Array column of level 1 science data file into buffer.
- Decode each of the 2048 byte data packet from the buffer as per the format specified in on-board software document.
- Read the fields of the frame header. From the values of header fields, get the frame number, mode id, quad id, the number of events in the frame and read the event data accordingly into an event buffer.

- Take the time for each event (fractional second) and the second count value from its frame header and add both to get the event time in seconds. Using the level 1 TCT file, convert the CZTI instrument time to UTC time using interpolation. Record this interpolated time as an additional column in the data.
- While decoding the PHA value read only the most significant 10 bits, to get PHA value in the range 0-1023.
- Create the output event file with 4 extensions for four quadrants with extension names as Q0, Q1, Q2 and Q3
- Create a binary table in each extension with columns time, cztseccnt, cztnticks, pha, detid, pixid, detx, dety, veto, alpha and write the decoded events to appropriate extensions.
- For each event in 4 quadrants, map the quad id, detid and pixid into (detx,dety) position on each quadrant of CZTI. Write this into corresponding columns of event file.
- Decode Veto spectrum from frame header and write it to Vetospectrum extension of event file.
- Copy all relevant keywords to event file FITS headers.
- Copy the GTI extensions from level 1 science file to the output event file.
- Decode the HK parameters and other information from frame headers and write them to hdr file as a binary table.
- In case of modeSS frame, decode the spectra and temperature information and write to SSM and TEMP extensions of event file and leave the event data extensions empty.

Output

- Level 2 event file
- Level 2 header file
- Bunch information file (see below)

7.2 cztbunchclean

Hard X-ray detectors are sensitive to charged particle interactions. Particles interacting with the detector lose energy continuously and produce tracks. Charged particles interacting with the instrument or spacecraft body produce secondary particles and X-ray photons, which can also deposit energy in the CZTI detectors.

As CZTI is composed of pixellated detectors, each pixel acts as an independent X-ray detector. So one charged particle can produce events in many pixels of CZTI at the same time. We call these multi-hit events as 'bunches', as they are bunched in time. Such events need to be identified and removed from the event file. This module of the level-2 pipeline identifies the bunched events and removes them. It also keeps track of the dead time caused by this. During the Performance Verification phase, one level of removal of bunched events has been implemented in the on-board processing in order to reduce the data volume. Relevant information of each bunch of events are encoded along with the other events.

Input

- Raw event file
- Bunch file

Method

- A bunch is defined as a collection of events occurring within $bunchdeftime$ (30 μs). Any event with a difference in time stamp less than or equal to $bunchdeftime$ from its neighbours belongs to the same bunch
- In the current version of on-board software, bunches are identified on-board and only three events of a bunch are recorded along with other relevant information of the bunch. Information about the bunches are decoded to bunch file produced by cztscience2event
- Each entry in bunch file corresponds to one bunch. Read the event file and bunch file to identify the events belonging to the bunch
- Events are assigned flag which denotes how many events are present in the bunch of the current event. Single events are flagged as one and double events are flagged as two
- Remove all events flagged as three or above, only single and double events are retained. Compute the bunch duration and subtract it from the live time for that second, which is initialized to one. Fractional exposure of all pixels of the quadrant is also modified.
- There is provision to ignore all events within $skipT1$ (0 s) time after the bunch. If it is non-zero the events are ignored and live time is modified
- For bunches with events less than or equal to $bunch\_length\_threshold$ (20), all events of the module where the bunch occurred are ignored for $skipT2$ (0.001 s) time after the end of bunch. If the events in bunch are spread across modules, then the module with the largest number of events in the bunch is considered for this. Fractional exposure of the pixels of that module and the live time are modified accordingly
- For bunches with events more than $bunch\_length\_threshold$ (20), all events are ignored for $skipT3$ (0.001 s) time after the end of bunch. Fractional exposure of all pixels of the quadrant and the live time are modified accordingly
- Note that all three time parameters are from the end time of the bunch, hence maximum ignored time is bunch duration plus maximum of the three time parameters.
- Single and double events after this cleaning are written out in the standard event file format.
- An EXPOSURE extension is added to the output event file where the fractional exposure of each pixel is stored.
- Live time for each one second interval during the observation is written to output live time file

Output

- Event file devoid of bunched events
- Live time file

7.3 cztpha2energy

Each X-ray photon incident on the CZT detector produces charge proportional to photon energy, which is measured as a voltage, and through the Analog to Digital Converters it is recorded as a Pulse Height Amplitude (PHA) channel value. This software module uses temperature dependent pixel wise calibration data to estimate the nominal energy of the incident photon from the recorded PHA, and expresses it on a Pulse Invariant and Pixel Invariant scale, called a PI channel value.

Input

- Event file
- Event file with SS Mode data
- Gain file from caldb
- Ebounds file from caldb

Method

- Copy the input event file to output event file.
- Query CIF file to get caldb Gain and Ebounds files.
- Add two columns, PI, and Energy in the output file.
- Read the PHA, detid and pixid of each event.
- Get the temperature of the detector at event time from the TEMP extension of SS Mode event file by interpolation.
- Read Gain and offset at a temperature nearest to the actual detector temperature for that event pixel from the caldb Gain file.
- Compute the energy of event as: $E = \text{gain} * \text{PHA} + \text{offset}$
- From Ebounds, find to which PI bin does the computed energy belong.
- Write the energy and PI values of the event to respective columns of the output file.
- Copy SSM and TEMP extensions from SSM event file to the output modeM0 event file.

Output

- Event file with Energy and PI columns added

7.4 cztgtigen

During the course of observation, there are time intervals where the data is not present due to SAA passage, data transmission loss etc. Also there are intervals when the source is occulted by the earth or the angular offset of the source has changed. In order to generate science products from such observations, it is important to identify such intervals and ignore the data for that duration and to properly account for the data gaps.

This task produces GTI file for each quadrant of CZTI based on thresholds on various parameters, as well as the GTI present in the event file. It also has provision to accept user defined GTIs for specific analysis cases.

Input

- Event file
- MKF file
- MKF threshold file
- Optional user GTI file

Method

- Read the mkf threshold file which defines the range of various mkf parameters for GTI
- Find time ranges when the all the mkf parameters are within the required ranges to generate GTI based on MKF
- Read the quadrant wise and common GTIs from the event file
- Find the intersection of common GTI with the MKF GTI and any other GTI provided by user
- Similarly find the intersection of each quadrant GTI with MKF and user provided GTIs
- Write the output GTIs as different extensions of GTI file

Output

- GTI file

7.5 cztgaas

In order to find the position and orientation of the CZTI field of view one needs to utilize the satellite aspect information and account for the alignment of the CZTI payload with respect to the satellite reference axes which is recorded in the telescope definition file. This module computes the time dependent pointing direction of CZTI axes as well as the average value.

Input

- Event file
- MKF file
- Teldef file from caldb
- GTI file

Method

- Read the CZTI alignment matrix elements from teldef file. The matrix elements define the transformation of a vector (DETX, DETY, DETZ) defined in detector coordinates to satellite body coordinates (SATX, SATY, SATZ)
- From the MKF file, for every recorded sample, read (RA, Dec) values of the Yaw, Roll, Pitch axes. From these, derive their Direction Cosines in the Inertial Coordinate System.

- Construct a transformation matrix from satellite body coordinates to the Inertial Coordinate System (ICS) using the above direction cosines
- CZTI z-axis is defined by vector [zDET]=(0, 0, 1) and x-axis by [xDET]=(1, 0, 0) in czti detector coordinate system. Find the components of these two unit vectors in the ICS for every sample in the mkf file and record them along with the time stamp in the output CZTI aspect array file.
- Read the quadrant wise extensions of the GTI file
- Average the components individually over the duration of good time intervals, and construct vectors out of these average components and normalise to obtain averaged unit vectors N and T respectively.
- Find the average pointing direction of the CZTI from:

$$\text{DEC} = \text{asin}(N_z);\ \text{RA} = \text{atan2}(N_y, N_x) \bmod 360 \text{ deg}$$

  and the average TWIST angle from:

$$\text{TWIST} = \text{atan2}[T_y \cos(\text{RA}) - T_x \sin(\text{RA}), T_z \cos(\text{DEC}) \sin(\text{DEC})(T_x \cos(\text{RA}) + T_y \sin(\text{RA}))]$$

- Record the normalized vectors, RA, DEC and TWIST values in the FITS header of the Aspect file

Output

- Aspect file

7.6 cztdatasel

This pipeline module filters the events in the input event file based on GTI file. As each quadrant has an independent GTI, it is possible to filter each quadrant data by the respective GTI or filter all quadrants with the common GTI. User input GTITYPE which has two possible values quad or common, determines this.

Input

- Event file
- GTI file

Method

- If the user defined GTI type is QUAD, read the quadrant wise GTI extensions of the GTI file. Else if the GTI type is COMMON, read the common GTI extension of GTI file
- Select the events with time stamps which are within the good time intervals
- Write the selected events to the output event file
- If gtitype is quad, copy the four quadrant GTIs to the output event file, else copy common GTI to the output file

Output

- GTI filtered event file

7.7 cztpixclean

CZTI is composed of 64 detectors each having 256 pixels. Some of these pixels can be noisy during the observations. In some cases, a few pixels are consistently noisy for substantial duration of the observation, and the events from these pixels need to be filtered out. Certain pixels and detectors are noisy for short durations and these pixels/detectors need to be ignored for those durations. This pipeline module filters the event file for noisy pixels/detectors.

Input

- Event file
- Livetime file from bunchclean
- Badpix file from caldb

Method

- Detector Plane Histogram is generated and noisy pixels are identified by iterative *nsigma* (5) clipping. The disabled pixel list from caldb is also taken into account.
- All events from already designated noisy and dead pixels are ignored.
- Generate detector light curve with bin size *det_tbinsize* (1 s). Ignore events from a module for a given time bin if counts in that bin is greater than *det_count_threshold* (25)
- Generate pixel light curve with bin size *pix_tbinsize* (1 s). Ignore events from a pixel for a given time bin if counts in that bin is greater than *pix_count_threshold* (2)
- Fractional exposure time of pixels and live time are modified accordingly.
- Single events and double events are written out in standard event file format separately. Modified pixel wise fractional exposures are written to the EXPOSURE extension of both the event files.
- Badpixel file with the dead and noisy pixels flagged is created.
- Modified live time values are written in the output file

Output

- Event file
- Double events file
- Live time file
- Badpixel file

7.8 cztflagbadpix

This module combines bad pixel information from multiple sources, if available. It reads a list of bad pixel files and writes out a single bad pixel file combining the information from all of them.

Input

- Badpixel file 1
- Badpixel file 2
- ...

Method

A list of all bad pixels mentioned in the input files is made, and the unique elements in the list are written out.

Output

- Badpixel file

7.9 cztevtclean

Events with simultaneous veto count or alpha particle detection are to be segregated from the rest of the events which constitute science data. Alpha tagged events have to be accumulated for calibration purpose. This module provides the functionality to select/reject alpha and veto-tagged events in various combinations according to user-supplied choices.

Input

- Event file

Method

- Create the output event file with same format as input event file.
- Read the user input alpha value (0 or 1) where value of one would select events which are alpha-tagged
- Read the user input Veto range, which defines the range of Veto PHA values to be selected. Zero value would select events which are not tagged by Veto.
- Find events which satisfy the veto and alpha criteria. Copy only those events to the output event file

Output

- Selected event file with/without veto/alpha

7.10 cztdpigen

The Detector Plane Image (DPI) gives the shadow of the coded mask recorded on the CZT detector. The pattern of total counts on each pixel gives a Detector Plane Histogram (DPH), which is then corrected for difference in effective area of different pixels to yield the DPI. This module bins the event data to generate DPH and DPI.

Input

- Clean event file
- Effective area file from caldb

Method

- Select the events based on user specified energy range
- Create detector plane histogram with counts from each pixel
- Read the effective area of the pixels from caldb file and the fractional exposure from the exposure extension of the events file
- Divide the counts in each pixel by its effective area and fractional exposure to produce the detector plane image (DPI)
- Write the DPI and DPH as image extensions in the output file

Output

- Detector plane histogram (dph)
- Detector plane image (dpi)

7.11 cztimage

Cross-correlation of mask pattern with the DPI using Fourier technique produces a crude image of the field. This module generates image using DPI by FFT method.

Input

- Detector plane image
- Mask pattern file from caldb

Method

- If the data is filtered with quad gti, each quadrant image will be obtained independently, else image is created with all four quadrant together
- Read the DPI file and the mask pattern file. Over sample DPI and mask arrays by user specified oversampling factor
- Take the Fourier Transform of oversampled mask and DPI, and multiply element by element the transform of mask and conjugate of the transform of DPI
- Take the inverse Fourier transform of the resulting array, which is the image in camera coordinates. Remove the aliased regions.
- Read the aspect file to get RA, DEC and TWIST angle
- Compute the angular scale of the image and other WCS keywords
- Write the image to output file and add the necessary header keywords

Output

- Image

7.12 cztbindata

Spectrum and light curves are generated by binning events in energy or time. As CZTI has a coded aperture mask, it is possible to generate background subtracted light curve or spectrum by weighting events from each pixel with the maskweights derived from the mask open fractions. This module generates spectrum/light curve for the field or for a given source direction by mask weighting or the total light curve and spectrum including background.

Input

- Clean event file
- Live time file
- Badpixel file from pixclean
- LLD file from caldb
- Maskpattern file from caldb
- Camera geometry file from caldb

Method

- Read the time and PI columns of the clean event file. Ignore events with PI value less than LLD of the respective pixel
- Read the user specified time bin size and energy range for light curve
- If the source coordinates are given in celestial system, compute the camera coordinates θ_x and θ_y of the source using aspect file
- For each active czti pixel i compute the open fraction (f_{ij}) for all PI channels (j). This includes the full geometry of the instrument including the mask, collimators etc
- Assign weight $w'_{ij} = 2f_{ij} - 1$ to all active pixels. Assign weight of zero to all flagged pixels in the badpix file
- Compute the renormalization offset

$$D_j = \frac{\sum_i w'_{ij} * a_{ij}}{\sum_i a_{ij}},$$

  where a_{ij} is the effective area of pixel i at energy channel j and the summation is over the active pixels alone.

- Compute the renormalized mask-weights as

$$w_{ij} = w'_{ij} - D_j$$

- An event in pixel i with PI j is assigned a weight of w_{ij}.
- To generate spectrum, add w_{ij} for an event in pixel i with PI value of j to counts in channel j of the spectrum.
- To generate light curve, compute the time bin to which a give event belongs. Add the mask weight of the event to that time bin

Output

- Spectrum
- Light curve

7.13 cztrspgen

This module generates response matrix for CZTI spectrum. Redistribution matrices for groups of pixels are precomputed and weighted addition of these with the effective area is employed to obtain multipixel response for CZTI.

Input

- Spectrum file
- Event file
- Exposure map file
- Badpixel file
- Ebounds from caldb
- Response parameter file from caldb
- Pixel response file from caldb
- LLD file from caldb
- Effective area file from caldb

Method

- Ignore the pixels that were not used while extracting the spectrum. Index i runs over the remaining good pixels.
- Compute the fraction of observation time ($t_i(T_n)$) each pixel i was at temperatures nearest to the calibration temperature T_n. As temperature is available at module level, the time fraction will be the same for all pixels in the module. This calculation takes care of GTI and any other time range used in generating the spectrum.
- For each pixel i at temperature $T_n \pm 2.5^o$, its contribution ($W_{ij}(T_n)$) to the overall response in PI channel j is given by

  $$W_i(T_n) = w_{ij}\ t_i(T_n)$$

  where w_{ij} if the mask weight and $t_i(T_n)$ is the fraction of total observation time for which the pixel was at temperature close to ($\pm 2.5^o C$) T_n.
- Multiply the redistribution matrices of each pixel with the effective area of the pixel for the given source direction which is given by

  $$A_{ik} = f_{ik}\ a_{ik}\ cos(\theta)$$

  where index i runs over active pixels an index k runs over incident photon energy channel. Here f_{ik} is the mask open fraction, a_{ik} is the effective area excluding the mask and collimators present in caldb file, θ is the source angle with detector normal

- Multiply W_{ij} for each pixel with the redistribution matrix of the pixel. These are added together to obtain the overall response.
- Write the response in standard rmf file format

Output

- Response matrix (.rsp) file