LEVEL-2 SOFTWARE PIPELINE FOR CADMIUM ZINC TELLURIDE IMAGER (CZTI) ON-BOARD ASTROSAT

April 21, 2016

Revision History

Contents

1 Introduction **4**

2 Overview of CZTI Data and Pipeline **4**

3 Level-1 Data Content **5**

4 Level-2 Pipeline Work flow **5**

5 Description of Software Modules **5**

5.1 cztmergeaux 5
5.2 cztattcorrect 8
5.3 cztorbcorrect 8
5.4 cztmergescience 9
5.5 cztscience2event 10
5.6 cztpha2energy 11
5.7 cztmkfgen 12
5.8 cztbunchclean 13
5.9 cztgtigen 14
5.10 cztgaas 15
5.11 cztdatasel 17
5.12 cztpixclean 17
5.13 cztevtclean 18
5.14 cztdpigen 18
5.15 cztimage 19
5.16 cztbindata 20
5.17 cztrspgen 21

6 Description of the pipeline script **22**

7 Format of level-2 FITS files **22**

1 Introduction

Introduction may be rewritten in the context of the scope of this document

Cadmium Zinc Telluride Imager (CZTI) on-board ASTROSAT employs an array of pixellated CZT detectors behind a Coded Aperture Mask (CAM) to detect and study hard X-ray sources. 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 a Coded Aperture Mask is placed above the collimator, about 48 cm above the detector plane. The combined geometric area of the all the CZT detectors in this payload is 976 cm^2. The primary energy range of operation of the CZTI would be about 15 to 150 keV. The CAM is made up of independent 256-element patterns for each detector module, derived from pseudo-noise Hadamard sets.

2 Overview of CZTI Data and Pipeline

The data received from the CZTI payload will pass 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 will be archived internally and will not be available 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 will be collated at this level and packed along with the respective science data. This data will be released via the web for science use, 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 will contain standard science products obtained from Level 1 data. Level 2 data would also be in FITS format and will be 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 will contain user configurable elements. While a default configuration will be run at the Payload Operation Centre (POC) to create the automated Level 2 data products, the user would have the option to generate more customized products by using the same software with other configuration settings. The Level 2 pipeline software will be made available for download. 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 of the CZTI Level 2 pipeline software is 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. This data consists of

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.
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**: This xml file has information like source observed, start and end time of observation etc.

Some description of orbit wise and merged files should be here.

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 4 summarizes the tasks performed by various software modules.

5 Description of Software Modules

5.1 cztmergeaux

Data is downloaded from Astrosat once in every orbit. A typical science observation spans over several orbits. Hence in order to produce data sets for one complete observation id, it is required to combine the instrument data and the auxillary data which is downloaded in multiple orbits. To avoid lose of data, successive orbit downloads would be having some amount of data overlap.

This task removes the overlaps between adjacent orbit auxillary data sets and merges them together to produce observation id level auxillary files.

Input

- Level 1 TCT file(s)
- Level 1 attitude file(s)
- Level 1 orbit file(s)
- Level 1 LBT HK file(s)

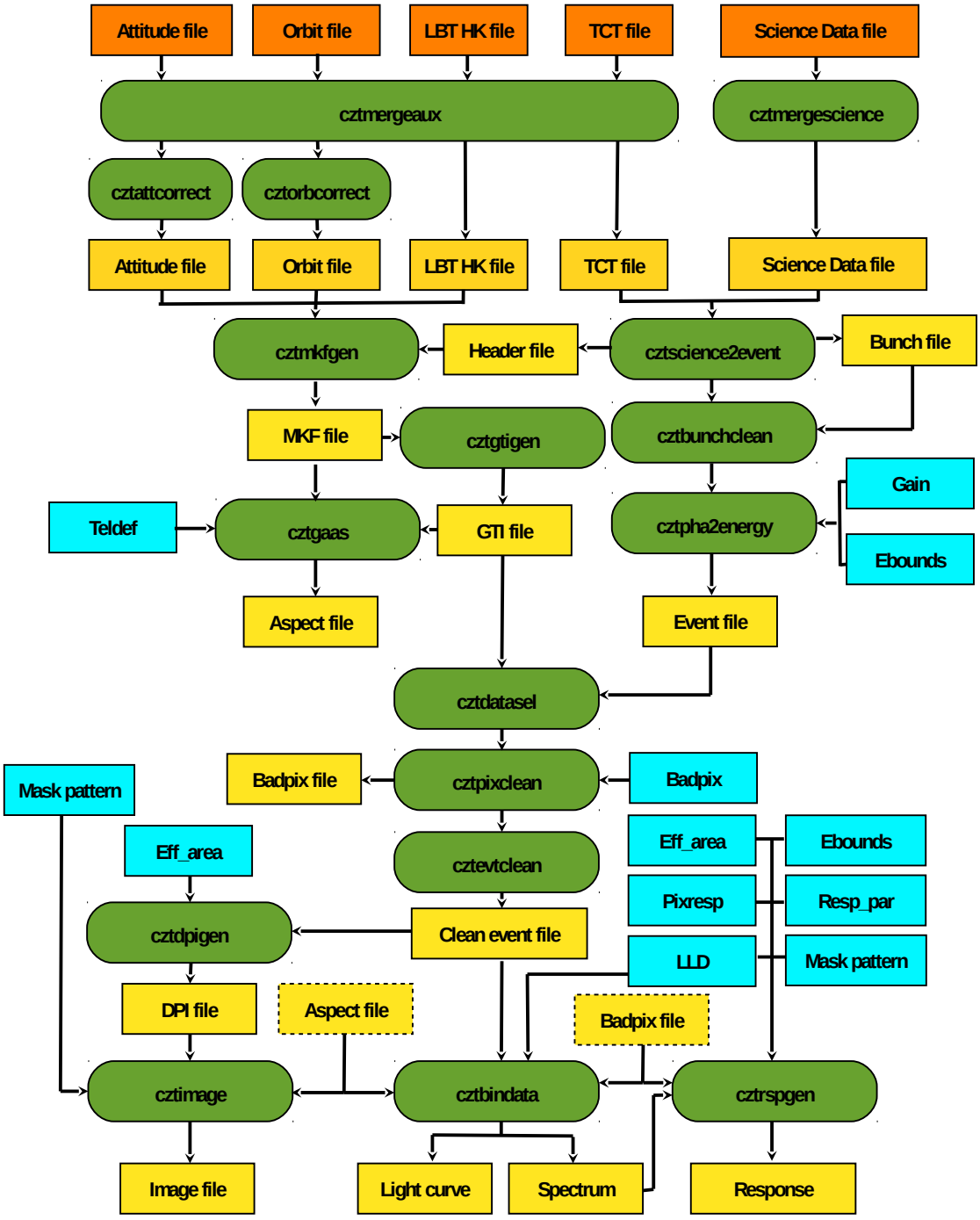

Figure 1: Level-2 pipeline work flow.

Sl No	Module name	Functionality
1	cztmergeaux	Merge auxillary files of the same obsid
2	cztattcorrect	Resample attitude file to one second
3	cztorbcorrect	Resample orbit file to one second
4	cztmergescience	a) Remove corrupted/incomplete frames b) Merge orbit-wise science data files of same obsid without overlapping frames c) Create GTI
5	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
6	cztpha2energy	Convert PHA value to PI and write to event file
7	cztmkfgen	Generate mkf using attitude, orbit and hbt header files
8	cztbunchclean	Clean the event file by identifying and removing bunches
9	cztgtigen	Generate GTI based on current gti, mkf and user input
10	cztgaas	Compute time dependant and average aspect of CZTI
11	cztdatasel	Select events based on gti
12	cztpixclean	Identify noisy pixels and detectors and remove noisy events
13	cztevtclean	Select events based on veto and alpha tags
14	cztdpigen	Generate Detector Plane Image from clean event file
15	cztimage	Create image by fft method
16	cztbindata	Generate light curve/spectrum
17	cztrspgen	Generate response matrix file

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

Method

- This module is common for all level 1 auxillary files
- Read the time column of the input auxillary files. Note the last time stamp present in one file and ignore all records with time stamps before this for the next file
- Write the unique time stamp samples to the output file
- Write or modify the necessary header keywords as given in the file format
- Write the algorithm to remove junk tct values, att values, orb values, hk values (if begin done here)

Output

- Level 1 TCT file
- Level 1 attitude file
- Level 1 orbit file
- Level 1 LBT HK file

5.2 cztattcorrect

This task of czti pipeline resamples the attitude file to uniform one second intervals.

- Attitude file

Method

- Read the time and other columns of input attitude file
- The output file should have attitude information at integer seconds spanning the full observation duration
- For each integer second time stamp, find two records before and two records after in the original file.
- Compute the quaternions, ra and dec by Lagrangian interpolation
- Write the algorithm
- Write the interpolated records to the output attitude file

Output

- Resampled attitude file

5.3 cztorbcorrect

This task of czti pipeline resamples the orbit file to uniform one second intervals.

- Orbit file

Method

- Read the time and other columns of input orbit file
- The output file should have orbit information at integer seconds spanning the full observation duration
- For each integer second time stamp, find two records before and two records after in the original file.
- Compute the orbit parameter values by Lagrangian interpolation, similar to cztattcorrect
- Write the interpolated records to the output attitude file

Output

- Resampled orbit file

5.4 cztmergescience

This module removes the czti data frames which are either corrupted or not having expected number of packets. Another major functionality of this module is merging science data files of the same observation id. As mentioned earlier, data from each orbit is received and it can have overlaps with adjacent orbit data. Such overlaps are removed and unique good data frames are written out to merged science data file. Good Time Intervals where each quadrant data was present is also written as extensions to the output science file.

Input

- Level 1 science data file(s)
- Level 1 tct file
- Level 1 mcap file

Method

- Read the DataArray and DecodingStatusFlag columns of the input science file(s).
- Mark all packets with DecodingStatusFlag not equal to zero or wrong sync words as corrupted.
- Decode frame header of first good frame to get Double words of detector data read (dcnt).
- Compute the expected number of packets as ceil(((dcnt*2-1024)/1020.))+1
- Consider the subsequent uncorrupted packets before next frame and verify whether the packet number is incrementing properly without repetition and whether number of packets present is equal to expected number of packets. If yes mark the frame as good else flag all packets of the frame as bad.
- Continue the same steps for next frame if the frame number is greater than the last good frame else skip till a frame satisfies this.
- Write out the good frames to output science data file.

- For modeM0 data, mark start and stop of good time intervals for each quadrant based on the following:
  - Data gaps due to missing/corrupted frames or change to other modes of operation.
  - If number of events is equal to maximum limit for FEB (3072), time stamp of last event is the end of the current good time interval as no data is recorded from that time to the start of next second.
  - At every hundredth second of czt clock data is not present with correct time stamps for 400ms.
  - There is no event data when FEB is in command mode.
- Convert start and stop times of GTI from czt time to UT using tct file. Actual time of events in a frame is czttime in header minus two.
- Write GTIs of each quadrant and common GTI for all quadrants as extensions to science data file.
- Write correct header keywords in all extensions of output science file.

Output

- Science data file with GTI extensions

5.5 cztscience2event

This module reads the level 1 science data files 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 with the help of time calibration table. Relevant housekeeping information present in headers and detector temperature information present in modeSS data is 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 czt 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 (if needed)

5.6 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 dependant pixel wise calibration data to estimate the nominal energy of the incident photon from the recorded PHA, and express it on a Pulse Invariant and Pixel Invariant scale, called a PI channel value.

Input

- Event file
- Event file with SSM 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, Energy and 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: $E = gain * PHA + offset$
- From Ebounds, find which PI bin to which the computed energy belongs.
- 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

5.7 cztmkfgen

In order to create good time interval (GTI), it is necessary to have house keeping parameters of CZTI, attitude information, orbit information, earth elevation, sun angle etc as a function of time. Some of this information is present in various auxillary files and some is present in the level 2 header files and some needs to be computed based on the available data. This software module generates a Make Filter File (MKF) which includes all relevant auxillary data for generation of good time intervals.

Input

- Orbit file
- Attitude file
- Level 2 hdr file
- LBT HK file

Method

- Create empty mkf file as per the template
- Read attitude and orbit file with integer second time stamp data
- Copy attitude and orbit information from respective files to the newly created MKF file
- Compute the coordinates of roll,pitch and yaw axes from the quaternions and write to mkf file

- Compute parameters like angular offset, sun angle, elevation, moon angle, sun elevation and write to respective columns of mkf file
- Read cpm rate from the lbt hk file. For each time stamp in the mkf file, add the latest available cpm rate from lbt file, to respective column of the mkf file.
- For other house keeping parameters, read level-2 header file, and write the latest available HK value for each time stamp in mkf file
- Modify the necessary header keywords

Output

- Level 2 MKF file

5.8 cztbunchclean

Hard X-ray detectors are sensitive to charged particle interactions. Particles interacting with the detector loses energy continuously and produce tracks. Charged particles interacting with the instrument or spacecraft body produces 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 is 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. Hence some part of the algorithm is different for two versions of on-board Processing Electronics software.

Input

- Raw event file
- Bunch file

Method

- A bunch is defined as events occurring within $bunchdeftime$ (30 μs). Any events with difference in time stamps less than or equal to $bunchdeftime$ belongs to the same bunch
- In the new version of on-board software, bunches are identified on-board and only three events of bunch are recorded along with other relevant information of the bunch. Information about the bunches are decoded to bunch file produced by cztscience2event
- In case of old version of on-board software (boot page 0), the bunch file will be empty. In that case read the events from the event file and identify the bunches according to the above definition
- In case of new version of on-board software (boot page 2), 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 $skipT2$ ($1e-3$ s) time after the end of bunch. If the events in bunch are spread across modules, module with 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 $skipT3$ ($1e-3$ 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

5.9 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 earth or the angular offset of the source has changed. In order to have 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 events 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

5.10 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 dependant 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) as:

$$SATX = ALIGNM11 * DETX + ALIGNM12 * DETY + ALIGNM13 * DETZ$$
$$SATY = ALIGNM21 * DETX + ALIGNM22 * DETY + ALIGNM23 * DETZ$$
$$SATZ = ALIGNM31 * DETX + ALIGNM32 * DETY + ALIGNM33 * DETZ$$

$$SAT = [ALIGNM] * DET$$

- 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.

$$L_y = cos(RA_y)cos(DEC_y); M_y = sin(RA_y)cos(DEC_y); N_y = sin(DEC_y)$$
$$L_r = cos(RA_r)cos(DEC_r); M_r = sin(RA_r)cos(DEC_r); N_r = sin(DEC_r)$$
$$L_p = cos(RA_p)cos(DEC_p); M_p = sin(RA_p)cos(DEC_p); N_p = sin(DEC_p)$$

- Construct a transformation matrix from satellite body coordinates to the Inertial Coordinate System (ICS):

$$[M] = \begin{pmatrix} L_y & L_r & L_p \\ M_y & M_r & M_p \\ N_y & N_r & N_p \end{pmatrix}$$

- 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 vector in the ICS as:

$$[N] = (N_x, N_y, N_z) = [M] * [ALIGNM] * [zDET]$$
$$[T] = (T_x, T_y, T_z) = [M] * [ALIGNM] * [xDET]$$

  Record (N_x, N_y, N_z) and (T_x, T_y, T_z) for every sample in mkf file 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 to obtain vectors $(< N_x >, < N_y >, < N_z >)$ and$(< T_x >, < T_y >, < T_z >)$. Normalize the vectors as:

$$(\hat{N_x} =< N_x > /R_n, \hat{N_y} =< N_y > /R_n, \hat{N_z} =< N_z > /R_n)$$

  where $R_n = (< N_x >^2 + < N_y >^2 + < N_z >^2)^{(1/2)}$

$$(\hat{T_x} =< T_x > /R_n, \hat{T_y} =< T_y > /R_n, \hat{T_z} =< T_z > /R_n)$$

  where $R_n = (< T_x >^2 + < T_y >^2 + < T_z >^2)^{(1/2)}$

- Find the average pointing direction of the CZTI:

$$DEC = asin(\hat{N_z}); RA = atan2(Ay, Az) mod 360 deg$$

  The average TWIST angle is computed as:

$$TWIST = atan2[Tycos(RA) - Txsin(RA), Tzcos(DEC)sin(DEC)(Txcos(RA) + Tysin(RA))]$$

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

Output

- Aspect file

5.11 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

5.12 cztpixclean

CZTI is composed of 64 detectors each having 256 pixels. Some of these pixels can be noisy during the observations. In some cases, few pixels are consistently noisy for substantial duration of the observation, and the events from these pixels needs to be filtered out. Certain pixels and detectors are noisy for short durations and these pixels/detectors need to be ignored for those duration. 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 noisy and dead pixels are ignored.
- Generate detector light curve with bin size $det\_tbinsize$ (1 s). Ignore events from 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 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

5.13 cztevtclean

Events with simultaneous veto count or alpha particle detection are to be segregated from 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

5.14 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)

5.15 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 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 the elements one by one
- Take the inverse Fourier transform of the resulting array, which is the image in camera coordinates
- Read the aspect file to get RA, DEC and TWIST angle
- Compute the angular scale of the image and other required keywords write the actual algorithm with correct equations
- Write the image to output file and add the necessary header keywords

Output

- Image

5.16 cztbindata

Spectrum and light curves are generated by binning events in energy or time space. As CZTI has 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

5.17 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 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 ($\pm 2.5^oC$) 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

6 Description of the pipeline script

Not the POC wrappers, but the user level wrappers which needs to be written

7 Format of level-2 FITS files

Shall we put the format of fits files here? Something like what was present in earlier ICD