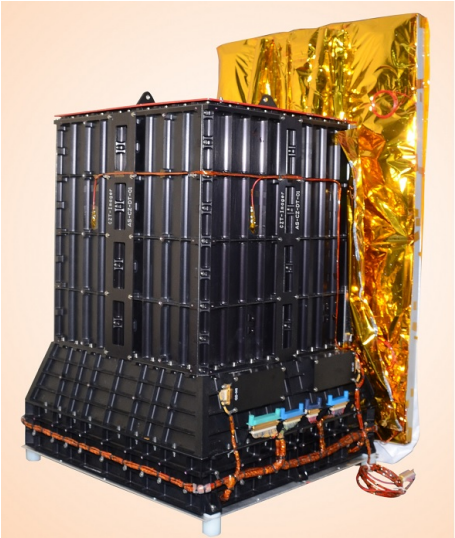
AstroSat
Cadmium Zinc Telluride Imager (CZTI)



Data Analysis Pipeline
User Guide

Version 3.0

December 2022

Mithun N P S[1], Dipankar Bhattacharya[2], Ajay Vibhute[2], Rakesh Khanna A[3], Ajay Ratheesh[3,4], Rahul Gopalakrishnan[2], Tanul Gupta[5], Preeti Tahliani[5], Arvind K Singh[5]

[1]PRL, Ahmedabad; [2]IUCAA, Pune; [3]TIFR, Mumbai; [4]INAF - IAPS, Rome, Italy ;[5]SAC, Ahmedabad

Document Revision History

Release 1.0 03 June 2016 initial release DB
Release 1.1 05 October 2016 DB/Mithun
Release 2.0 06 October 2017 DB/Ajay

- cztimage: Applied corrections for the detector and mask misalignment
- cztbindata: Added functionality to compute livetime for fractional binsize
- cztpha2energy: Added functionality to process events in chunk
- cztdatasel: Minor changes to header keywords
- other: Minor bug fixes

Release 2.1 08 March 2018 DB/Ajay/Mayuri

- cztbindata: Bug fixes in fractional livetime correction
- cztdatasel: Fixed a bug in calculation of exposure time
- cztpipeline: Added functionality to run pipeline in stages
- other: Added a version number to the header of the output file.

Release 3.0 09 December 2022 Mithun/Ajay V/Ajay R/DB

- Several major updates to various pipeline modules and inclusion of new modules based on updated calibration

Contents

1 Introduction

The Cadmium Zinc Telluride Imager (CZTI) on-board ASTROSAT is a Coded Mask Imaging camera working in the energy range 20 to 200 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.

For a more detailed description of the instrument, see Bhalerao et al. (2017).

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.

Details on the files in level-1 and level-2 data archive are given in Section 2.1.

2.1 Overview of Data Analysis Pipeline

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).

CZTI level-2 data reduction pipeline is organized into different modules for each specific task. In figure 2.1, work flow of the pipeline is shown. Table 2.1 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. It may be noted that two options of event selection procedures are included in the pipeline: a default event selection algorithm suited for the sources observed within the FOV and a generalized one which is best suited for bright transients such as Gamma Ray Bursts.
- **Stage 3**: Generation of science products. In this last stage, Detector Plane Image (DPI), sky plane image, spectrum, light curve and response matrix are generated.

The pipeline tasks can either be executed individually or by using the **cztpipeline** module, a python script, which allows the user to run required stages of the pipeline tasks together. It is advised that users use the **cztpipeline** module and use the individual modules only if it is required for specific analysis cases.

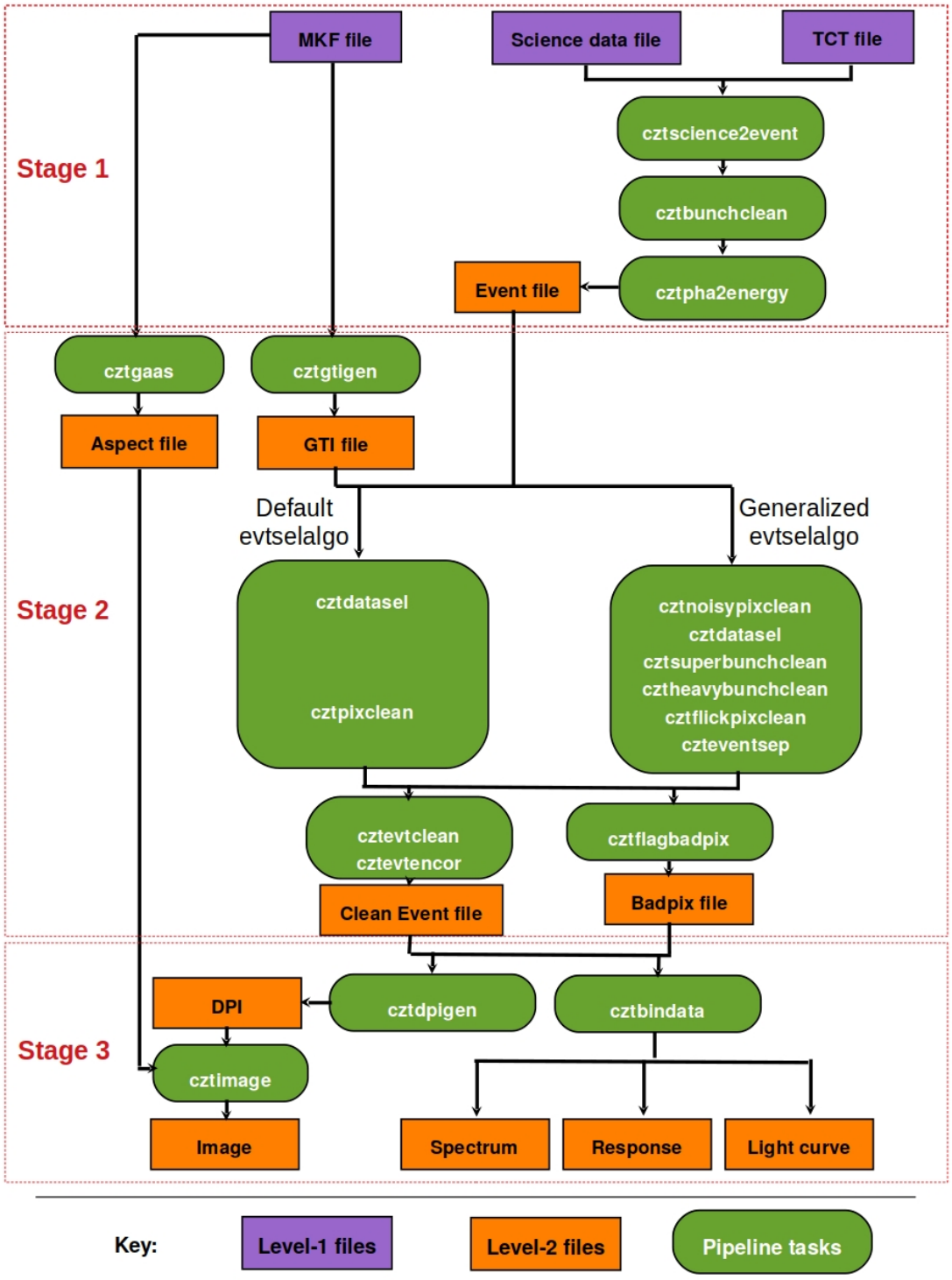


Figure 2.1: Overview of CZTI data analysis pipeline work flow. Dotted line boxes show pipeline modules in each of the three stages.

Sl No	Module name	Stage	Functionality
1	cztpipeline	NA	Python script to run all modules of the pipeline
2	cztscience2event	1	Generate raw event files from level-1 science data
3	cztbunchclean	1	Removes multi-hit bunched events from the event list
4	cztpha2energy	1	Applies gain calibration and add PI channel for each event
5	cztgtigen	2	Generates good time intervals based on user specified rangs of MKF parameters as well as optional user GTIs
6	cztgaas	2	Generates time-dependent and average aspect of CZTI detectors
7	cztdatasel	2	Filters the event list with the GTI provided
8	cztpixclean	2	Removes events from noisy and flickering pixels from the event list
9	cztflagbadpix	2	Provides option to combine multiple badpixel files
10	cztevtencor	2	Apply additional gain correction on event energies based on in-flight calibration.
11	cztevtclean	2	Filters out (or in) alpha and veto tagged events from the event list and creates final clean event file
12	cztdpigen	3	Generate detector plane image from clean event file
13	cztimage	3	Generate sky image using the DPI and mask pattern
14	cztbindata	3	Generate background substracted source spectrum (with response) and light curve by mask-weighting
15	cztaddspec	NA	Provide option to add multiple spectra together.

In addition to the default pipeline modules, following modules that implement a generalized event selection algorithm instead of that used in cztpixclean module are also provided. These modules are to be used instead of cztpixclean module of the default pipeline. Recommended way to use these modules is by using the cztpipeline python script with option eventselalgo = generalized.

16	cztnoisypixclean	2	Removes gross noisy pixel events
17	cztsuperbunchclean	2	Removes post-bunch events for bunches with large number of events
18	cztheavybunchclean	2	Further removal of post-bunch noise events
19	cztflickpixclean	2	Identied flickering pixels and removes events from those pixels
20	czteventsep	2	Separates single and double events

Table 2.1: Summary of modules in CZTI level-2 pipeline and their functionality

3 Downloading Data and File Content

AstroSat data is available for download from AstroSat data archive at ISSDC. Level-1 and standard level-2 data for each observation ID that are public are available on this website for download. It may be noted that there would be only one level1 zip file and one level2 zip file for each observation ID which includes the merged data from multiple orbits.

In most cases, it may be sufficient to download the level-2 zip file and then excute stages 2 aand 3 of the pipeline by user. However, there are some cases where the user may want to start from the level-1 file as discussed in examples in Section 3.2.

Content of the level-1 and level-2 data files are briefly described here.

3.1 Level-1 Data Files

The data from spacecraft 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. Content of the a CZTI level1 zip file is shown below as an example and the details of each of files is listed after that.

```
unzip LEVEL1AS1CZT20181014C04_007T02_9000002432.zip

20181014_C04_007T02_9000002432_level1/
    czti/
        AS1C04_007T02_9000002432czt_level2.mkf
        modeM0/
            AS1C04_007T02_9000002432cztM0_level1.fits
        modeSS/
            AS1C04_007T02_9000002432cztSS_level1.fits
        aux/
            AS1C04_007T02_9000002432czt_level1.tct
            aux1/
                AS1C04_007T02_9000002432czt_level1.att
                AS1C04_007T02_9000002432czt_level1.orb
            aux2/
                AS1C04_007T02_9000002432czt_level1.lbt
            aux3/
                AS1C04_007T02_9000002432czt_level1.orb
        AS1C04_007T02_9000002432czt_level1_mcap.xml
        LEVL1AS1CZT20181014C04_007T02_9000002432_dqr_V1.2.xml
```

1. **Science Data File**(level1.fits): 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, usually modeM0 and modeSS.

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. **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.

4. **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.

5. **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.

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

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

8. **DQR file**(.dqr.xml): This xml file has information on data quality.

3.2 Level-2 Data Files

Level-2 data archive contains event files and final products such as image, light curve, and spectra generated by default processing at CZTI POC. It is possible that the data archive includes the default products from previous versions of the pipeline. In case the level-2 files do not have a FITS header keword 'PIPEVER' or its value is less than the latest pipeline version, it is required to reprocess the level-2 data from stage 2 of the pipeline and use the products generated.

Content of the a CZTI level2 zip file is shown below as an example.

```
unzip LEVEL2AS1CZT20181014C04_007T02_9000002432.zip

20181014_C04_007T02_9000002432_level2/
    czti/
        modeM0/
            AS1C04_007T02_9000002432cztM0_level2.hdr
            AS1C04_007T02_9000002432cztM0_level2_bc.evt
            AS1C04_007T02_9000002432cztM0_level2_bc_livetime.fits
            AS1C04_007T02_9000002432cztM0_level2.gti
```

```
    AS1C04_007T02_9000002432cztM0_level2.aspect_Q3
    AS1C04_007T02_9000002432cztM0_level2.aspect_Q2
    AS1C04_007T02_9000002432cztM0_level2.aspect_Q1
    AS1C04_007T02_9000002432cztM0_level2.aspect_Q0
    AS1C04_007T02_9000002432cztM0_level2_quad_clean.evt
    AS1C04_007T02_9000002432cztM0_level2_quad_livetime.fits
    AS1C04_007T02_9000002432cztM0_level2_quad_badpix.fits
    AS1C04_007T02_9000002432cztM0_level2_quad_clean.dpi
    AS1C04_007T02_9000002432cztM0_level2_quad_clean.img_Q0
    AS1C04_007T02_9000002432cztM0_level2_quad_clean.img_Q1
    AS1C04_007T02_9000002432cztM0_level2_quad_clean.img_Q2
    AS1C04_007T02_9000002432cztM0_level2_quad_clean.img_Q3
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q0.pha
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q1.pha
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q2.pha
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q3.pha
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q0.rsp
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q1.rsp
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q2.rsp
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q3.rsp
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q0.lc
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q1.lc
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q2.lc
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q3.lc
    AS1C04_007T02_9000002432cztM0_level2_quad_clean_weight.evt
modeSS/
    AS1C04_007T02_9000002432cztSS_level2.fits
aux/
    AS1C04_007T02_9000002432czt_level1.tct
    aux1/
        AS1C04_007T02_9000002432czt_level1.att
        AS1C04_007T02_9000002432czt_level1.orb
    aux2/
        AS1C04_007T02_9000002432czt_level1.lbt
    aux3/
        AS1C04_007T02_9000002432czt_level1.orb
AS1C04_007T02_9000002432czt_level2.mkf
AS1C04_007T02_9000002432czt_level1_mcap.xml
LEVL1AS1CZT20181014C04_007T02_9000002432_dqr_V1.2.xml
```

It may be noted that several files in level-2 package are same as that in level-1 such as the aux files and MKF file. It may also be noted that intermediate files produced by the level-2

pipeline modules are not included as part of the level-2 archive. Details of the level-2 files in the archive are given below.

1. **Header file**(.hdr): File with instrument parameters extracted from frame headers
2. **Bunch clean event file**(bc.evt): Event file at the end of stage-1 with pulse invariant channels added, after running cztscience2event, cztbunchclean, and cztpha2energy.
3. **Bunch clean livetime file**(bc_livetime.fits): Livetime file at the end of stage-1 of pipeline.
4. **GTI file**(.gti): Standard Good Time Interval (GTI) file generated by default inputs to cztgtigen
5. **Aspect file**(.aspect): Includes time dependant aspect information
6. **Clean event file**(clean.evt): Final clean event file at end of stage-2 of the pipeline, selected with the standard GTI.
7. **Livetime file**(livetime.fits): Final livetime file at end of stage-2.
8. **Badpix file**(badpix.fits): Badpixel file with information of disables, noisy pixels etc.
9. **DPI**(.dpi): Default Detector plane image
10. **Image**(.img): Default Sky image generated from the DPI
11. **Spectrum**(.pha): Default background subtracted source spectrum
12. **Response**(.rsp): Response matrix associated with the spectrum
13. **Light curve**(.lc): Default background subtracted source light curves
14. **Clean event file with maskweights**(clean_weight.evt): Clean event files with mask weight information added

4 Download and Installation

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

http://astrosat-ssc.iucaa.in/cztiData

Please follow the instructions below to install the software. If you have a previous version of CZTI pipeline installed, be sure to remove that and the install this version.

Pre-Requisites

OS: Linux/Unix (CentOS-5.8+, Ubuntu 14.04+, OS X 10.10+, Scientific Linux 6.5+)
Compiler: gcc (GCC) 4.4+, gfortran, Python 3.x

OS X users: Do not mix apple compilers with those installed from other sources. We have tested the pipeline with the macports versions of python and gcc

Installation Instructions

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

2. Untar the package.

```
tar -xvf czti_pipeline_yyyymmdd_Vm.n.tar
```

This will create a directory named "czti_pipeline"

3. Set following environment variables in ~/.bashrc file similar to the example given

```
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
GLOG_log_dir: Directory where log files will be saved
CZTI_templates: Directory where file templates are stored
```

by adding following lines to ~/.bashrc file

```
export as1czt=<path to czti_pipeline directory>/czti_pipeline/czti
export PFILES="$PFILES:$as1czt/paramfiles"
export PATH=$as1czt/bin:$as1czt/scripts:$PATH
export GLOG_log_dir=$as1czt/log
export CZTI_templates=$as1czt/templates
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$as1czt/lib
export PYTHONPATH="$PYTHONPATH:$as1czt/scripts"
export CXXFLAGS="-fpermissive"
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
```

If the user has installed any other package that uses PIL with PFILES environment defined in /.bashrc, make sure that the other declaration are appending the pfile paths to the environment variable.

If the user has HEASOFT installation, it is recommended that the command to source the initialization script of HEASOFT, commonly aliased as heainit, if included in /.bashrc shall be after the above commands. This will create a local copy of Parameter files under $HOME/pfiles.

4. CALDB installation:

Untar czti_caldb_yyyymmdd.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>
```

such that under $CALDB directory the directory named data resides.

5. Execute the following commands in bash

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

$as1czt folder would contain include/, lib/ and bin/ directories and executables would be generated under $as1czt/bin, which completes the installation. One can use cztpipeline to excute multiple stages of pipeline modules together.

User manual

- To view complete user manual : **czthelp**
- To view help for a task : **czthelp moduleName**

5 Quick Start Guide with cztpipeline

In this section, some illustrative examples of usage of cztpipeline for few specific cases are shown.

To view the help of cztpipeline, use `cztpipeline -h` or `czthelp cztpipeline` To get information on a specific input keyword, use `cztpipeline -h keyword`.

5.1 Generating spectrum for entire observation

In this example, spectrum of a source for the period of entire observation will be generated starting from level-2 data archive. For this, we consider an observation of crab with observation id 9000002432. As the first step, download the level-2 data zip file from Astrobrowse for this observation and then unzip the same as follows at desired location

```
mithun@Thinkpad:~/Desktop/data_czti$ unzip LEVEL2AS1CZT20181014C04_007T02_9000002432.zip
```

This would generate a directory named `20181014_C04_007T02_9000002432_level2` which contains the standard level-2 products including spectra. However, it is likely that the products are generated with previous versions of software and CALDB. Still, there would not be any changes in the products until end of stage-1 of the data analysis pipeline. Hence, we will execute the cztpipeline stages 2 and 3 using the level-2 products files from stage 1 that was available in the downloaded data.

As our objective is to generate spectrum for entire observation, the inputs to cztpipeline are straightforward as shown in the example below. Note that the level-2 directory given as the input is where the output files will also be generated and will replace the original files in the example given below.

```
mithun@Thinkpad:~$ cztpipeline
------------------------------------------------------------------------------------------------------

                                        ASTROSAT CZTI PIPELINE
                                      Version: V3.0  December 2022

------------------------------------------------------------------------------------------------------



Enter Start Step [2]: 2
Enter End Step [3]: 3
Enter path to create L2 pipeline products []:
/home/mithun/Desktop/data_czti/20181014_C04_007T02_9000002432_level2
Enter full path of user-gtifile [-]:
Enter output type <lc/spec/both>(for cztbindata) [both]: spec
Apply maskweights (YES/NO)(for cztbindata) [Yes]: Yes
Enter RA of source in decimal degrees [0]: 83.63308
Enter DEC of source in decimal degrees [0]: 22.01446
Overwrite existing file?(YES/NO): [YES]: Yes
------------------------------------------------------------------------------------------------------
                                      CZTPIPELINE PARAMETERS
------------------------------------------------------------------------------------------------------
Task Name                           : cztpipeline.py
Level2 Directory                    : /home/mithun/Desktop/data_czti/20181014_C04_007T02_9000002432_level2
Entry Stage                         : 2
Exit Stage                          : 3
Right Ascension of Source           : 83.63308
Declination of Source               : 22.01446
User GTI File                       : -
Event Selection Algorithm           : default
Clobber                             : YES

------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------


Do you want to continue? [Y/n]: Y
```

After this, cztpipeline would execute the modules of the pipeline and print the details on screen finally showing the message that the excution completed successfully. Spectral files and respective response files for each quadrant are generated within the level-2 directory under modeM0. Figure 5.1 shows the spectra of each quadrant for this observation loaded in XSPEC with below commands.

```
XSPEC12>data 1:1 AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q0.pha
XSPEC12>data 2:2 AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q1.pha
XSPEC12>data 3:3 AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q2.pha
XSPEC12>data 4:4 AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q3.pha
XSPEC12>ignore bad
```

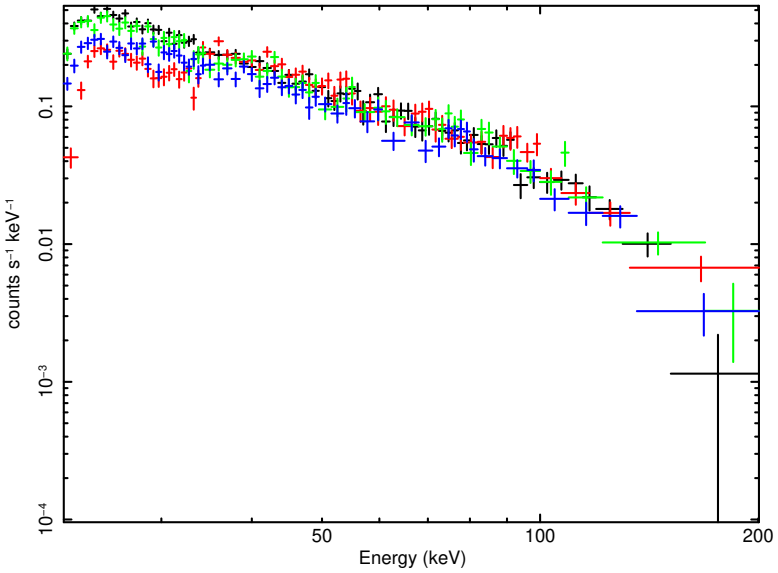


Figure 5.1: CZTI crab spectra for all quadrants for obsid 2432 generated with cztpipeline and loaded in XSEPC.

It is also possible to have optional suffix added to the output files generated so that they have different names than that available in the level-2 data archive. For example, running `cztpipeline` with options as below would produce output files from stage 2 and 3 with a suffix 'fullobs' added to the file names.

```
mithun@Thinkpad:~$ cztpipeline suffix='fullobs'
-------------------------------------------------------------------------------------------------

                                    ASTROSAT CZTI PIPELINE
                                  Version: V3.0  December 2022

-------------------------------------------------------------------------------------------------



Enter Start Step [2]: 2
Enter End Step [3]: 3
Enter path to create L2 pipeline products []:
/home/mithun/Desktop/data_czti/20181014_C04_007T02_9000002432_level2
Enter full path of user-gtifile [-]:
Enter output type <lc/spec/both>(for cztbindata) [spec]:
```

```
Apply maskweights (YES/NO)(for cztbindata) [Yes]:
Enter RA of source in decimal degrees [83.63308]:
Enter DEC of source in decimal degrees [22.01446]:
Overwrite existing file?(YES/NO): [YES]:
```

This option of suffix to the file names will be useful when a user need to generate spectra for multiple intervals of an observation as discussed in Section 5.3.

5.2 Adding quadrant wise spectra to get total spectrum

For bright sources such as crab in the example above, it is always advisable to load quadrant wise spectra and fit them simultaneously allowing for cross-normlization constants. However, for faint sources (typically less than 100 mCrab), it may be useful to co-add spectra from all quadrants to get the total spectrum. CZTI data analysis pipline includes a task `cztaddspec` for this purpose. As an example, here we add quadrant-wise spectra generated from the previous example to get the total spectrum.

As `cztaddspec` requires a text file with list of spectrum files, first we create such a file named 'speclist.txt' with content as below:

```
AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q0.pha
AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q1.pha
AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q2.pha
AS1C04_007T02_9000002432cztM0_level2_quad_clean_Q3.pha
```

Now execute `cztaddspec` from the directory containing these spectra as follows:

```
mithun@Thinkpad:~/Desktop/data_czti/20181014_C04_007T02_9000002432_level2/czti/modeM0$ cztaddspec
Enter Input List of spectrum files [] : speclist.txt
Enter Name of output spectrum file [] : AS1C04_007T02_9000002432cztM0_level2_quad_clean_quadadded.pha
Whether to add response files? [y] : y
Whether to add exposure times (same detector multiple obs)? [y] : n
[ cztaddspec.cpp:96] ------------------------------------------------------------------------
[ cztaddspec.cpp:97]                            CZTADDSPEC PARAMETERS
[ cztaddspec.cpp:98] ------------------------------------------------------------------------
[ cztaddspec.cpp:99] Modulename                    : cztaddspec_v3.0
[ cztaddspec.cpp:100] Input List file               : speclist.txt
[ cztaddspec.cpp:101] Output spectrum file          : AS1C04_007T02_9000002432cztM0_level2_quad_clean_quadadded.pha
[ cztaddspec.cpp:102] Output response file          : AS1C04_007T02_9000002432cztM0_level2_quad_clean_quadadded.rsp
[ cztaddspec.cpp:104] Addresp                       : YES
[ cztaddspec.cpp:111] Addexposure                   : NO
[ cztaddspec.cpp:114] Clobber                       : YES
[ cztaddspec.cpp:118] History                       : YES
[ cztaddspec.cpp:121] ------------------------------------------------------------------------
[ main.cpp:38] CZTADDSPEC STARTED.....................
[ cztaddspec.cpp:146] Adding 4 spectra together
[ utils.cpp:1605] Template file used : /home/mithun/work/czti/gitczti/czti/czti_pipeline/czti/templates/spectrumTemplate
[ cztHeaderParam.cpp:314] Read all the header keywords
[ cztHeaderParam.cpp:531] Copied all the header keywords sucessfully
[ level2validation.cpp:4925] Total signal to noise ratio is 115.548
[ level2validation.cpp:4926] Selected good channels are 34 - 390
[ level2validation.cpp:4937] Grouping spectrum into 100 logarithmic bins
[ utils.h:732] Updating keys in file: AS1C04_007T02_9000002432cztM0_level2_quad_clean_quadadded.pha
[ utils.cpp:1605] Template file used : /home/mithun/work/czti/gitczti/czti/czti_pipeline/czti/templates/rspTemplate
[ main.cpp:49] CZTADDSPEC COMPLETED SUCCESSFULLY
[ main.cpp:51] Time Elapsed:0 seconds
```

This produces total spectrum file and corresponding response file. It may be noted that the same tool maybe used to add spectra of one quadrant from multiple observations. In that case the input parameter addexposure need to be set to yes.

5.3 Generating spectra for specific time intervals

To generate spectra for user specified time intervals, there is provision to provide user GTI while generating final GTI for analysis. The user gti file is a text file with two columns in each row corresponding to start and end times of a good time interval. The times have to be in AstroSat time, which is number of UTC seconds from 2010-Jan-1 0 UTC. A tool to convert UTC to AstroSat time and vice-versa is available at http://astrosat-ssc.iucaa.in:8080/astrosattime/

As an example we will generate spectra for two time intervals of the observation ID 2432 in the example given in Section 5.1.

We will generate spectra for two time intervals as listed below as an example.

1. 277202800 – 277208000
2. 277209000 – 277213800

Two text files with these user GTI are to be created first, named as `usergti_trange1.txt` and `usergti_trange2.txt` in this example. Then, we execute `cztpipeline` twice with these user GTI inputs as given below. Also note that we use suffix option to have different names for the output files for each time range.

```
mithun@Thinkpad:~$ cztpipeline suffix='trange1'
-------------------------------------------------------------------------------------------------------

                                        ASTROSAT CZTI PIPELINE
                                      Version: V3.0  December 2022

-------------------------------------------------------------------------------------------------------



Enter Start Step [2]:
Enter End Step [3]:
Enter path to create L2 pipeline products []:
/home/mithun/Desktop/data_czti/20181014_C04_007T02_9000002432_level2/
Enter full path of user-gtifile [-]: usergti_trange1.txt
Enter output type <lc/spec/both>(for cztbindata) [lc]: spec
Apply maskweights (YES/NO)(for cztbindata) [Yes]: Yes
Enter RA of source in decimal degrees [0]: 83.63308
Enter DEC of source in decimal degrees [0]: 22.01446
Overwrite existing file?(YES/NO): [YES]:

mithun@Thinkpad:~$ cztpipeline suffix='trange2'
-------------------------------------------------------------------------------------------------------
```

```
                    ASTROSAT CZTI PIPELINE
                    Version: V3.0  December 2022

-------------------------------------------------------------------------------------------------

Enter Start Step [2]:
Enter End Step [3]:
Enter path to create L2 pipeline products []:
/home/mithun/Desktop/data_czti/20181014_C04_007T02_9000002432_level2/
Enter full path of user-gtifile [-]: usergti_trange2.txt
Enter output type <lc/spec/both>(for cztbindata) [lc]: spec
Apply maskweights (YES/NO)(for cztbindata) [Yes]: Yes
Enter RA of source in decimal degrees [0]: 83.63308
Enter DEC of source in decimal degrees [0]: 22.01446
Overwrite existing file?(YES/NO): [YES]:
```

This would produce spectra and responses for the two time ranges within the level-2 directory having names with suffixes trange1 and trange2. Figure 5.2 shows quadrant 0 spectra for the two time ranges.

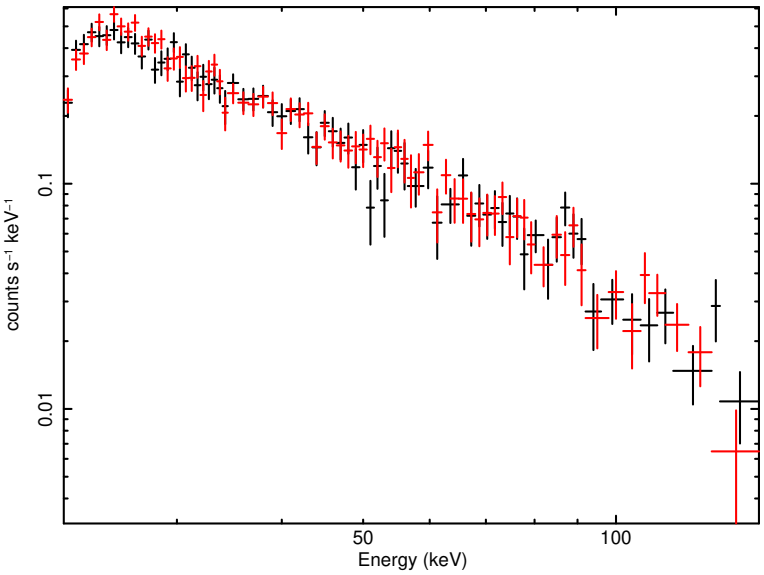


Figure 5.2: CZTI crab spectra with quadrant 0 of CZTI for two intervals.

5.4 Generating light curve of a GRB

In addition to the primary sources observed within FOV, CZTI also detects off-axis transients such as Gamma Ray Bursts (GRB) (see for example Rao et al, 2016). In this example, we gen-

erate the CZTI light curve of a GRB, GRB160821A. This GRB occurred during the observation ID 9000000618, so first download the level-1 zip file for this observation ID from AstroBrowse.

We start from level-1 data as we would use the generalized event selection algorithm for the analysis instead of the default one. The default event selection algorithm is optimized for on-axis sources and is not best suited for GRBs, instead a generalized event selection algorithm is employed. Please see Ratheesh et al (2021) for more details.

Unzip the level-1 zip file `LEVEL1AS1CZT20160821G05_009T02_9000000618.zip` to desired location to get level-1 directory `20160821_G05_009T02_9000000618_level1`. Also create a level-2 directory `20160821_G05_009T02_9000000618_level2` where the output files will be generated.

Now execute `cztpipline` with inputs as below. Note that `evtselalgo='generalized'` is given as a command line option to `cztpipeline` and the mkfthresholdfile is given to be the threshold file specific to transients which excludes the conditions of earth occultation of the primary source etc. Also, note that we use mask-weight = No option as the light curve for GRB, which is an off-axis source, cannot be generated by mask-weighting.

```
mithun@Thinkpad:~$ cztpipeline evtselalgo='generalized' mkfthresholdfile=
'${as1czt}/config/mkfThresholds_transients.txt'
------------------------------------------------------------------------------------------------------

                                    ASTROSAT CZTI PIPELINE
                                  Version: V3.0  December 2022

------------------------------------------------------------------------------------------------------



Enter Start Step [2]: 1
Enter End Step [3]: 3
Enter L1 Directory Path []:
/home/mithun/Desktop/data_czti/20160821_G05_009T02_9000000618_level1
Enter path to create L2 pipeline products []:
/home/mithun/Desktop/data_czti/20160821_G05_009T02_9000000618_level2
Enter full path of user-gtifile [-]:
Enter output type <lc/spec/both>(for cztbindata) [spec]: lc
Apply maskweights (YES/NO)(for cztbindata) [Yes]: No
Enter time bin size for lc (for cztbindata) [100]: 1
Enter minimum energy for lc (keV) (for cztbindata) [20]: 20
Enter maximum energy for lc (keV) (for cztbindata) [200]: 200
Overwrite existing file?(YES/NO): [YES]:
------------------------------------------------------------------------------------------------------
                                    CZTPIPELINE PARAMETERS
------------------------------------------------------------------------------------------------------
Task Name                          : cztpipeline.py
Level1 Directory                   : /home/mithun/Desktop/data_czti/20160821_G05_009T02_9000000618_level1
Level2 Directory                   : /home/mithun/Desktop/data_czti/20160821_G05_009T02_9000000618_level2
Entry Stage                        : 1
Exit Stage                         : 3
Right Ascension of Source          : 0.0
Declination of Source              : 0.0
User GTI File                      : -
```

```
Event Selection Algorithm            : generalized
Clobber                              : YES

-------------------------------------------------------------------------------------------------------

Generalized mode selected.
CZTBUNCHCLEAN will be executed with skipT1, skipT2, and skipT3 set to zero.
CZTDATASEL will be executed with gtitype = QUAD

-------------------------------------------------------------------------------------------------------

Do you want to continue? [Y/n]: Y
```

This generates the light curves for each quadrant for the entire observation. It may also be noted that the pipeline also generates images, but has no meaning in this case. Figure 5.3 shows the light curves loaded in `lcurve` showing the time around the GRB.

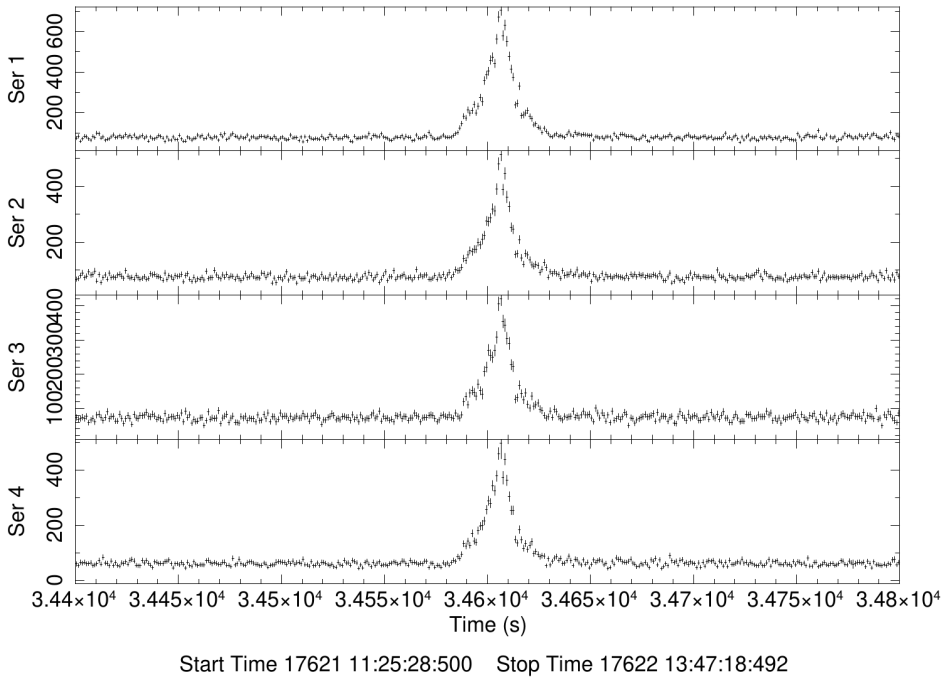


Figure 5.3: CZTI light curve for GRB. Four panels show the light curves of each quadrant.

5.5 Using cztpipeline in python scripts

Users may want to do processing of mutiple data sets or the same data set with different parameters in scripts. This can be done by calling cztpipeline from within python scripts of the users. An example of calling cztpipeline in python script is shown in the code snippet below. For this, import the class `Cztpipeline` from the module `cztpipeline` and create an instance of the same. Then use the method `run` with the parameters of cztpipeline are given as an array

of strings with keyword value pair as shown in the example below. All parameters including the hidden parameters are accessible in this manner.

In the below code snippet, spectra and images of a crab observation are generated by calling cztpipline from within python.

```
from cztpipeline import Cztpipeline

l2dir = '/home/mithun/Desktop/data_czti/20181014_C04_007T02_9000002432_level2'
parameter_list = ['entrystage=2','exitstage=3','l2dir={}'.format(l2dir),\
'maskWeight=yes' ,'usergtifile=-','outputtype=spec','clobber=yes',\
'rasrc=83.63308','decsrc=22.01446']

czt = Cztpipeline()
czt.run(parameter_list)
```

It may be noted that the full CZTI pipeline software needs to be installed in the system for this to work including the the addition of python path in bashrc as mentioned in the installation instructions.

6 Pipeline Modules and Usage

This section provides complete usage of each module of the pipeline listing all the available parameters. Hidden optional parameters are listed in square brackets.

6.1 cztpipeline

Purpose

This script is used to run all modules of CZTI pipeline together for the requires stages of the pipeline.

Parameters

- **entrystage** : Start stage of cztpipeline (1-3)
- **exitstage** : End stage of cztpipeline (1-3)
- **l1dir** : Level-1 directory path
- **l2dir** : Level-2 directory path
- **usergtifile** : User gti file to be used for additional filtering of data. Input to cztgtigen
- **rasrc** : RA of source in decimal degrees. Input to cztbindata
- **decsrc** : DEC of source in decimal degrees. If RA and DEC are zero, the source coordinates from MKF header is used. Input to cztbindata
- **outputtype** : Output to be generated by cztbindata. (spec/lc/both)
- **maskWeight** : Whether to apply mask-weights (Y/N) to generate background subtracted spectrum/light curve in cztbindata

- **timebinsize** : Time bin size in seconds for light curve. Input to cztbindata
- **emin** : Minimum energy for light curve (keV) for cztbindata
- **emax** : Maximum energy for light curve (keV) for cztbindata
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes
- [**debug**] : Whether to generate debug information for modules (Y/N). Default is no
- [**remintermedfile**] : Whether to remove intermediate files generated by different modules(Y/N). Default is yes.
- [**evtselalgo**] : Event selection algorithm to be employed in stage 2 (default/generalized). Default value is 'default'
- [**suffix**] : Optional suffix to be added to all output file names. This can be used when startstage is 2 or 3 and it would expect input files to have standard names.
- [**prefix**] : Optional prefix to be added to all output file names. This will be useful only if startstage is 1 as it expects the same prefix for all level-2 files.
- [**nPackets**] : Number of packets to be read into buffer at once. This parameter needs to be reduced if there is insufficient memory on the system. Default is 100000..
- [**BigEndian**] : Whether the data is bigendian (Y/N). Default is yes
- [**outbunchevtfile**] : Name of optional output event file with bunch events. Default is "-" which does not produce this file
- [**livetime_binsize**] : Time binsize in seconds for which livetimes have to be recorded. Default is 1.
- [**bunchdeftime**] : Bunch definition time in micro seconds. Default is 30.
- [**bunch_length_thresh**]: Bunch length threshold. Default is 20
- [**skipT1**] : Time parameter skipT1 in seconds. Default is 0
- [**skipT2**] : Time parameter skipT1 in seconds. Default is 1e-3
- [**skipT3**] : Time parameter skipT1 in seconds. Default is 1e-3
- [**buffer**] : Number of events to be read into buffer. This needs to be reduced if if there is insufficient memory on the system. Default is 100000000
- [**mkfthresholdfile**] : Parameter threshold file with ranges of various MKF parameters to be used in GTI selection. Recommended threholds are given in "$as1czt/config/mkfThresholds.txt", which is the default. The threshold file has four columns. First column is the name of the MKF parameter, second column is a boolean flag (0/1) to say if any filtering based on this

parameter is required. The third and fourth columns provide minimum and maximum values of the parameter to be included in GTI. "-" in these two columns mean that no lower (or upper) limit will be applied. It is required to have entries for all 67 parameters in the threshold file and the filtering criteria can be modified by changing the values of the last three columns as required.

- [**minexposure**] : Minimum exposure of each GTI interval. GTIs with stop-start less than this value are excluded from the output GTI. This is meant to exclude events from frames where event saturation occured. Default value is 1 second.
- [**gtitype**] : Whether to use individual quadrant GTI (quad) or use the common GTI for all quadrants (common). Default is quad
- [**writedblevt**] : Whether to generate double pixel event list (Y/N) by cztpixclean in case of default evtselalgo and by czteventsep in case of generalized evtselalgo. Default is No
- [**nsigma**] : N-sigma value to be used for identifying gross noisy pixels by iterative N-sigma clipping. Default is 5.
- [**det_tbinsize**] : Time binsize for generating detector light curve for identifying flickering detectors. Default is 1 sec.
- [**pix_tbinsize**] : Time binsize for generating pixel light curve for identifying flickering detectors. Default is 1 sec.
- [**det_count_thresh**] : Count threshold in a detector within det_tbinsize for considering as a flickering detector. Default is 25
- [**pix_count_thresh**] : Count threshold in a pixel within pix_tbinsize for considering as a flickering pixel. Default is 2
- [**applyLLD**] : Whether to apply LLD cuts to events (Y/N). Default is yes.
- [**applyULD**] : Whether to apply ULD cuts to events (Y/N). Default is yes.
- [**timegcor**] : Whether to apply time dependant gain correction factor(Y/N). Default is yes.
- [**alphaval**] : Alpha values to be accepted (1/0). Default is 0 which rejects alpha-tagged events
- [**vetorange**] : Veto Range/Ranges to accept (0-127). Default is 0 which rejects all veto-tagged events.
- [**badpixThreshold**] : Threshold of bad pixel quality flag to be applied for generating DPI, spectrum and light curve. Default is 0 which selects only good pixel events.
- [**quadsToProcess**] : Quadrants to be processed. Default is "-" which uses all four quadrants of CZTI.

- [**timerange**] : Time range to be used to generate DPI. Default is "-" which uses entire observation
- [**ebins**] : Energy range to be used for DPI generation. Default is "-"
- [**algorithm**] : Which imaging algorithm to be used (fft/balcc). Default is balcc
- [**oversamplingfactor**] : Oversampling factor for dpi and mask. Default is 16
- [**generate_eventfile**] : Whether to generate event file with weight column (Y/N). Default is no.
- [**genresp**] : Whether to generate response matrix for spectrum (Y/N). Default is yes.
- [**dogrouping**] : Whether do default grouping for mask-weighted spectrum (Y/N). Default is yes.
- [**grpfile**] : Text file with grouping of spectral channels, same as the input to grppha. Default is "-". If a file is provided and dogrouping==yes, grouping flags are set according to the text file input, otherwise default grouping is done. Not available in present version.
- [**deltatx**] : Additional shift in thetax to be applied in degrees. Default is 0. Only for use by Instrument team.
- [**deltaty**] : Additional shift in thetay to be applied in degrees. Default is 0. Only for use by Instrument team.
- [**gofile**] : CALDB gain file. Default is 'CALDB'
- [**caldbbadpix**] : CALDB bad pixel file name. Default is 'CALDB'
- [**maskfile**] : CALDB maskpattern file. Default is 'CALDB'
- [**shadowlib**] : CALDB shadow library file. Default is 'CALDB'
- [**eboundsfile**] : CALDB ebounds file. Default is 'CALDB'
- [**camgeomfile**] : CALDB camera geometry file. Default is 'CALDB'
- [**effareafile**] : CALDB effective area file. Default is 'CALDB'
- [**bkgscalfile**] : CALDB bkg spectra scale file. Default is 'CALDB'
- [**lldfile**] : CALDB LLD file. Default is 'CALDB'
- [**uldfile**] : CALDB ULD file. Default is 'CALDB'
- [**respparfile**] : CALDB response parameter file. Default is 'CALDB'
- [**pixrespfile**] : CALDB pixel response file. Default is 'CALDB'
- [**gaincorfile**] : CALDB gaincor file. Default is 'CALDB'
- [**effareacorfile**] : CALDB effective area correction file. Default is 'CALDB'
- [**syserrfile**] : CALDB systematic error file. Default is 'CALDB'

6.2 cztscience2event

Purpose

This module reads the level 1 science data file and decodes the data packets into events and generate the raw event file. Timestamps for events are also assigned. Relevant housekeeping information present in headers and detector temperature information present in modeSS data are also decoded and written to output files.

Parameters

- **infile** : Input level-1 science data file. It can be modeM0 or modeSS
- **TCTfile** : Input time calibration table file
- **outfile** : Name of output event file
- **hdrInfoFile** : Name of output event header information file
- **bunchfile** : Name of output bunch file
- [**nPackets**] : Number of packets to be read into buffer at once. This parameter needs to be reduced if there is insufficient memory on the system. Default is 100000..
- [**BigEndian**] : Whether the data is bigendian (Y/N). Default is yes
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.3 cztbunchclean

Purpose

This module identifies multi-hit events bunched in time due to charged particles and removes those events.It also keeps track of the dead time caused by this and records this in an output live time file generated along with the output event file.

Parameters

- **infile** : Input event file generated by cztscience2event
- **bunchfile** : Input bunch file generated by cztscience2event
- **outfile** : Name of output event file
- **livetimefile** : Name of output livetime file
- **outbunchfile** : Name of output bunch file

- [**outbunchevtfile**] : Name of optional output event file with bunch events. Default is "-" which does not produce this file
- [**livetime_binsize**] : Time binsize for which livetimes have to be recorded. Default is 1 second.
- [**bunchdeftime**] : Bunch definition time in micro seconds. Default is 30.
- [**bunch_length_thresh**]: Bunch length threshold. Default is 20
- [**skipT1**] : Time parameter skipT1 in seconds. Default is 0
- [**skipT2**] : Time parameter skipT1 in seconds. Default is 1e-3
- [**skipT3**] : Time parameter skipT1 in seconds. Default is 1e-3
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.4 cztpha2energy

Purpose

This module uses temperature dependent pixel wise gain values from ground calibration 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. It adds two columns to the event file named Energy and PI.

Parameters

- **infile** : Input event file generated by cztpha2energy
- **outfile** : Name of output event file
- **tempextfile** : Input level-2 mode SS file generated by cztscience2event having temperature extension
- [**eboundsfile**] : Name of Ebounds file defining PI channels. Default is 'CALDB'
- [**gofile**] : Name of gain file. Default is 'CALDB'
- [**buffer**] : Number of events to be read into buffer. This needs to be reduced if if there is insufficient memory on the system. Default is 100000000
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.5 cztgtigen

Purpose

This module creates the good time interval (GTI) file for each quadrant of CZTI based on thresholds on various parameters given as input, as well as the default GTI present in the event file that accounts for data gaps etc. It also has provision to accept user defined GTIs for specific analysis cases.

Parameters

- **eventfile** : Input event file generated by cztpha2energy (or by cztnoisypixclean with generalized event selection algorithm)
- **mkffile** : Input MKF (makefilter) file
- **thresholdfile** : Parameter threshold file with ranges of various MKF parameters to be used in GTI selection. Recommended threholds are given in "$as1czt/config/mkfThresholds.txt", which is the default. The threshold file has four columns. First column is the name of the MKF parameter, second column is a boolean flag (0/1) to say if any filtering based on this parameter is required. The third and fourth columns provide minimum and maximum values of the parameter to be included in GTI. "-" in these two columns mean that no lower (or upper) limit will be applied. It is required to have entries for all 67 parameters in the threshold file and the filtering criteria can be modified by changing the values of the last three columns as required.
- **outfile** : Name of the output GTI file
- **usergtifile** : Option user GTI file in text format. Default is "-" which means no user GTI is considered. The GTI text file should have two columns with start and stop times in each row. Times should be in AstroSat UTC seconds (seconds since 2010-01-01 0 UTC).
- [**minexposure**] : Minimum exposure of each GTI interval. GTIs with stop-start less than this value are excluded from the output GTI. This is meant to exclude events from frames where event saturation occured. Default value is 1 second.
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.6 cztgaas

Purpose

This module computes the time dependent pointing direction of CZTI axes as well as the average value required for assigning coordinates for images.

Parameters

- **infile** : Input event file generated by cztpha2energy
- **mkffile** : Input MKF (makefilter) file
- **outAspectfile** : Name of output aspect file
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.7 cztdatasel

Purpose

This pipeline module filters the events in the input event file based on the GTI file provived and generates an output event file with the selected events.

Parameters

- **infile** : Input event file generated by cztpha2energy (or by cztnoisypixclean with generalized event selection algorithm)
- **gtifile** : GTI file generated by cztgtigen with any optional user GTI
- **outfile** : Name of output event file
- [**gtitype**] : Whether to use individual quadrant GTI (quad) or use the common GTI for all quadrants (common). Default is quad
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.8 cztpixclean

Purpose

This pipeline module filters the event file for noisy pixels and detectors. It identified gross noisy pixels and removes events from those pixels and also removes events from flickering pixels and detectors for those specific periods. A bad pixel file with the combined list of bad pixels identified by this module and that from CALDB is also written out along with the cleaned event file.

Parameters

- **infile** : Input event file generated by cztdatasel
- **inlivetimefile** : Input livetime file generated by cztbunchclean
- **outfile1** : Name of output single event file

- **outlivetimefile** : Name of output livetime file
- **badpixfile** : Name of output badpix file
- [**writedblevt**] : Whether to generate double pixel event list (Y/N). Default is No
- [**outfile2**] : Name of optional double event file. Default is default.dblevt
- [**nsigma**] : N-sigma value to be used for identifying gross noisy pixels by iterative N-sigma clipping. Default is 5.
- [**det_tbinsize**] : Time binsize for generating detector light curve for identifying flickering detectors. Default is 1 sec.
- [**pix_tbinsize**] : Time binsize for generating pixel light curve for identifying flickering detectors. Default is 1 sec.
- [**det_count_thresh**] : Count threshold in a detector within det_tbinsize for considering as a flickering detector. Default is 25
- [**pix_count_thresh**] : Count threshold in a pixel within pix_tbinsize for considering as a flickering pixel. Default is 2
- [**caldbbadpix**] : CALDB bad pixel file name. Default is 'CALDB'

6.9 cztflagbadpix

Purpose

This module combines bad pixel information from multiple sources, if required. It reads a list of bad pixel files and writes out a single bad pixel file combining the information from all of them. A list of all bad pixels mentioned in the input files is made, and the unique elements in the list are written out.

Parameters

- **nbadpixFiles** : Number of input badpixel files to be read
- **badpixfile** : Input bad pixel file name
- **outfile** : Name of output bad pixel file
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes
- [**debug**] : Whether to generate debug information (Y/N). Default is no.

6.10 cztevtencor

Purpose

This module applies additional gain correction on the event PI values based on the time independant and time-dependant correction factors obtained from analysis of in-flight data. It also optinally removes events below LLD and above ULD.

Parameters

- **infile** : Input event file with energy and PI columns.
- **outfile** : Name of output event file
- [**gaincorfile**] : CALDB gaincor file with additional gain correction factors. Default is 'CALDB'
- [**lldfile**] : CALDB LLD file. Default is 'CALDB'
- [**uldfile**] : CALDB ULD file. Default is 'CALDB'
- [**applylld**] : Whether to apply LLD cuts to events (Y/N). Default is yes.
- [**applyuld**] : Whether to apply ULD cuts to events (Y/N). Default is yes.
- [**timegcor**] : Whether to apply time dependant gain correction factor(Y/N). Default is yes.
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.11 cztevtclean

Purpose

This module provides the functionality to select/reject alpha and veto-tagged events in various combinations according to user-supplied choices.

Parameters

- **infile** : Input event file generated by cztpixclean (or by czteventsep with generalized event selection algorithm)
- **outfile** : Name of output event file
- [**alphaval**] : Alpha values to be accepted (1/0). Default is 0 which rejects alpha-tagged events
- [**vetorange**] : Veto Range/Ranges to accept (0-127). Default is 0 which rejects all veto-tagged events.

- [**isdoubleEvent**] : Whether the input file is a double event (Y/N). Default is no. If the input file is a double event, this parameter should be set to yes to get correct results.
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.12 cztdpigen

Purpose

This module generates detector plane histogram (DPH) and detector plane image (DPI) from clean event file. DPI is obtained by applying effective area correction on the DPH.

Parameters

- **infile** : Input clean event file generated by cztevtclean
- **badpixFile** : Input bad pixel file generated by cztpixclean
- **outDPHfile** : Name of output DPH file
- **outDPIfile** : Name of output DPI file
- [**badpixThreshold**] : Threshold of bad pixel quality flag to be applied for generationg of DPI. Default is 0 which selects only good pixel events.
- [**effAreafile**] : CALDB effective area file. Default is 'CALDB'
- [**quadsToProcess**] : Quadrants to be processed. Default is "-" which uses all four quadrants of CZTI.
- [**timerange**] : Time range to be used to generate DPI. Default is "-" which uses entire observation
- [**ebins**] : Energy range to be used. Default is "-"
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.13 cztimage

Purpose

This module generates sky images from the DPI using Balanced cross-correlation or FFT techniques.

Parameters

- **infile** : Input DPI file generated by cztdpigen
- **aspectfileQ0** : Average aspect file for Q0 generated by cztgaas
- **aspectfileQ1** : Average aspect file for Q1 generated by cztgaas
- **aspectfileQ2** : Average aspect file for Q2 generated by cztgaas
- **aspectfileQ3** : Average aspect file for Q3 generated by cztgaas
- **outImgFile** : Name of output image file
- [**algorithm**] : Which imaging algorithm to be used (fft/balcc). Default is balcc
- [**intype**] : Input file type (DPI/DPH). Default is DPI.
- [**oversamplingfactor**] : Oversampling factor for dpi and mask. Default is 16
- [**quadsToProcess**] : Qudrant IDs to be processed. Default is "-" which processes all quadrants.
- [**maskfile**] : CALDB mask pattern file. Default is 'CALDB'
- [**shadowlib**] : CALDB shadow library file. Default is 'CALDB'
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes
- [**debug**] : Whether to generate debug information (Y/N). Default is no. If set to yes when imaging algorithm is balcc, images for each detector module are generated.

6.14 cztbindata

Purpose

This module generates background subtracted sources spectrum (with response matrix) and light curves for a given source coordinates within the FOV by using mask-weighting tecnique. There is also option to generate total spectrum or light curve including background.

Parameters

- **inevtfile** : Input clean event file generated by cztevtclean
- **mkffile** : Input MKF file
- **badpixfile** : Input badpix file generated by cztpixclean
- **livetimefile** : Input livetime file generated by cztpixclean
- **outfile** : Base name of the output files

- **outputtype** : Output type (lc, spec, both)
- **maskWeight** : Whether to apply mask-weights (Y/N) to generate background subtracted spectrum/light curve
- **rasrc** : RA of source in decimal degrees.
- **decsrc** : DEC of source in decimal degrees. If RA and DEC are zero, the source coordinates from MKF header is used.
- **timebinsize** : Time bin size for light curve in seconds
- **emin** : Minimum energy for light curve (keV)
- **emax** : Maximum energy for light curve (keV)
- **outevtfile** : Name of optional output event file with mask-weights. Generated only if generate_eventfile is set to yes.
- [**badpixThreshold**] : Bad pixel quality flag threshold to be applied (0-4). Default is 0.
- [**quadsToProcess**] : Quadrants to be processed. Default is "-"
- [**generate_eventfile**] : Whether to generate event file with weight column (Y/N). Default is no.
- [**genresp**] : Whether to generate response matrix for spectrum (Y/N). Default is yes.
- [**dogrouping**] : Whether do default grouping for mask-weighted spectrum (Y/N). Default is yes.
- [**grpfile**] : Text file with grouping of spectral channels, same as the input to grppha. Default is "-". If a file is provided and dogrouping==yes, grouping flags are set according to the text file input, otherwise default grouping is done. Not available in present version.
- [**applyLLD**] : Whether to apply LLD cut for non-maskwt spec(Y/N). Default is Yes
- [**applyULD**] : Whether to apply ULD cut for non-maskwt spec(Y/N). Default is Yes
- [**deltatx**] : Additional shift in thetax to be applied in degrees. Default is 0.
- [**deltaty**] : Additional shift in thetay to be applied in degrees. Default is 0.
- [**maskfile**] : CALDB maskpattern file. Default is 'CALDB'
- [**eboundsfile**] : CALDB ebounds file. Default is 'CALDB'
- [**camgeomfile**] : CALDB camera geometry file. Default is 'CALDB'
- [**effareafile**] : CALDB effective area file. Default is 'CALDB'
- [**bkgscalfile**] : CALDB bkg spectra scale file. Default is 'CALDB'
- [**lldfile**] : CALDB LLD file. Default is 'CALDB'

- [**uldfile**] : CALDB ULD file. Default is 'CALDB'
- [**respparfile**] : CALDB response parameter file. Default is 'CALDB'
- [**pixrespfile**] : CALDB pixel response file. Default is 'CALDB'
- [**gaincorfile**] : CALDB gaincor file. Default is 'CALDB'
- [**effareacorfile**] : CALDB effective area correction file. Default is 'CALDB'
- [**syserrfile**] : CALDB systematic error file. Default is 'CALDB'
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes
- [**debug**] : Whether to generate debug information (Y/N). Default is no. If set to yes spectra for each detector module is generated.

6.15 cztaddspec

Purpose

This module provides functionality to add multiple CZTI spectra together. It is possible to add different quadrant spectra of one observation or add each quadrant spectra of multiple observations.

Parameters

- **listfile** : Text file with list of input spectrum files
- **specfile** : Name of output spectrum file
- **addresp** : Whether to create response file along with added spectrum (Y/N).
- **addexposure** : Whether to add exposure times(Y/N), which is applicable when multiple observations of same detector is added. For adding spectra of four quadrants of same observation, set this to No
- [**respfile**] : Name of output response file. Default is "default" which geneartes response having same basename as the output spectrum file
- [**dogrouping**] : Whether to apply default grouping to output spectrum file (Y/N). Default is yes.
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.16 cztnoisypixclean

Purpose

This module is part of generalized event selection algorithm that may be used instead of cztpixclean module in default pipeline. It identifies the gross noisy pixels and removes all the events from such pixels. It carried out iterative 5-sigma clipping of noisy pixels, seperately for nominal gain pixels and low gain pixels.

Parameters

- **infile** : Input event file generated by cztpha2energy
- **outfile** : Name of output event file
- **outbadpixfile** : Name of output badpix file
- [**caldb_badpix_file**] : CALDB badpixel file. Default is 'CALDB'.
- [**thresholdfile**] : Text file with threshold parameters. Default file with recommended values is "$as1czt/config/noiseReductionThreshold.txt".
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.17 cztsuperbunchclean

Purpose

This module is part of generalized event selection algorithm that may be used instead of cztpixclean module in default pipeline.It is used to identify post-bunch noises that span different modules. It identifies the bunches with large number of events which causes post bunch noises along the path of the interacted particle. These events are identified by an algorithm (outlined in Paul et al 2021), and those GTIs are removed from further analysis. The livetime file is updated with the newly updated GTI. The threshold file contains the recommended threshold values given in Ratheesh et al 2021b.

Parameters

- **infile** : Input event file generated by cztdatasel
- **inbunchfile** : Input Bunch file generated by cztbunchclean
- **inlivetimefile** : Input livetime file generated by cztbunchclean
- **outfile** : Name of output event file
- **outlivetimefile** : Name of output livetime file

- [**thresholdfile**] : Text file with threshold parameters. Default file with recommended values is "$as1czt/config/superBunchCleanThreshold.txt".
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.18 cztheavybunchclean

Purpose

This module is part of generalized event selection algorithm that may be used instead of cztpixclean module in default pipeline. It removes the post bunch noises due to cosmic particles from the genuine X-ray events, based on their properties which differ from the latter. The threshold file contains recommended threshold values given in Ratheesh et al 2021b.

Parameters

- **infile** : Input event file generated by cztsuperbunchclean
- **inbunchfile** : Input bunch file generated by cztbunchclean
- **outfile** : Name of output event file
- [**thresholdfile**] : Text file with threshold parameters. Default file with recommended values is "$as1czt/config/heavyBunchThreshold.txt".
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.19 cztflickpixclean

Purpose

This module is part of generalized event selection algorithm that may be used instead of cztpixclean module in default pipeline. It identifies and flags all the flickering pixels (or noisy pixels at shorter time scales). The identified flickering pixels are updated in the badpix file. It further identfies and removes events from those pixels having counts beyond the Poissonian limit using the DPH at 1 s and 10 s time scales.

Parameters

- **infile** : Input event file generated by cztheavybunchclean
- **inbadpixfile** : Input badpix file gnerated by cztnoisypixclean
- **outfile** : Name of output event file
- **outbadpixfile** : Name of output badpixel file

- [**thresholdfile**] : Text file with threshold parameters. Default file with recommended values is "$as1czt/config/flickPixThreshold.txt".
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

6.20 czteventsep

Purpose

This module is part of generalized event selection algorithm that may be used instead of cztpixclean module in default pipeline. It seperates single and double events in an event file.

Parameters

- **infile** : Input event file generated by cztflickpixclean
- **outfile_single** : Name of output single event file
- [**writedblevt**] : Whether to generate double pixel event list (Y/N). Default is No
- [**outfile_double**] : Name of optional output double event file. Default is default.dblevt
- [**clobber**] : Whether to overwrite existing output files (Y/N). Default is yes
- [**history**] : Whether to write history keywords in fits file (Y/N). Default is yes

7 Description of Algorithms in the Software Modules

Brief descriptions of the algorithms employed in each of the modules of CZTI data analysis pipeline are given here.

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 GTIs and common GTI (GTI common to all quadrants) 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 cztevtencor

In-flight calibration of CZTI showed that some additional time-independant and time-depandant corrections are required in the gain parameters from ground calibration (Mithun et al, 2022, in prep). This module incorporates this additional gain correction. If the event files are not passed through this module, the gain correction is applied on the fly in cztbindata while generating spectra and light curves.

Input

- Event file
- Gain correction parameter file from CALDB

Method

- Read the time-independant gain correction parameters from CALDB file
- Compute the time-dependant gain correction parameter based on the time keywords in the header of the input event file
- For each event, compute the corrected PI values and energy values with the additional gain correction
- Update the PI and energy columns in the output event file with the corrected values

- Also remove the events below LLD and above ULD based on the user input
- Updated/add header keyword GAINCOR in the output event file to YES

Output

- Event file with corrected PI and energy columns

7.10 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

7.11 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)

7.12 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 as well as by direct balance cross-correlation with shadow library. Details of the imaging algorithm is presented in Vibhute et al (2021).

Input

- Detector plane image
- Mask pattern file from CALDB and shadow 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.
- If the algorithm chosen is balanced cross correlation, then the image is generated with that method.
- Read the aspect file to get the average 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

7.13 cztbindata

Spectra 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. Response matrix associated with the spectra are also generated. Details of spectra and light curve extraction by mask-weighting is discussed in Mithun et al (2022, in prep).

Input

- Clean event file
- MKF 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 information in the MKF 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

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

  where B_{ij} is the relative background counts for pixel i at energy channel j which is read from the CALDB and the summation is over the active pixels alone. This factor is estimated for each module separately.

- 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

- Response matrix (.rsp file) corresponding to the spectrum is also generated taking into account the mask-weights. It may be noted that the response matrix includes both redistribution matrix and effective area.

Output

- Spectrum

- Light curve

- Response matrix

7.14 cztaddspec

Often, users may want to add multiple spectra together. It could either be the spectra of four quadrants of CZTI for a sngle observation or spectra of the same source from multiple observations. This modules provides this functionality to add the spectra and create added spectra and associated response.

Input

- List of spectrum files (spectra and associated response files as listed should also be available)

Method

- Read individual spectra and respective response files if the responses are to be added

- If the spectra are from multiple exposures of the same detector, counts are simply added together and total exposure is computed as sum of individual exposures. Response matrix is obtained as exposure weighted mean of individual response matrices
- If the spectra are of different quadrants for same observation, counts are added together while the exposure is taken to average of the exposures of input spectra. Individual response matrices are added together weighting by the respective exposure times
- Added spectra and response are written in standard format

Output

- Added spectrum
- Corresponding respose file

7.15 cztnoisypixclean

Noisy pixels are identified in each of the quadrant by flagging pixels with counts larger than 5 sigma in an iterative way untill no further pixels are identified noisy. This is done seperately for regular pixels and low gain pixels. A bad pixel file is generated by updating the newly found noisy pixels in the CALDB badpix file.

Input

- Event file after bunch clean
- Badpix file from CALDB

Method

- Detector Plane Histogram is generated and noisy pixels are identified for each quadrant by iterative *nsigma* (5) clipping, seperately for normal and low gain pixels in that particular quadrant.
- The badpix list is updated and stored in a seperate badpixel file.
- The events from noisy pixels are removed from further analysis.
- Write the events other than noise to the output event file.

Output

- Event file after removing events from noisy pixels.
- Badpix file after updating the newly found noisy pixels in the badpix list from CALDB.

7.16 cztsuperbunchclean

Noise events are seen after some bunches for timescales until around 200 ms. Such events can be found by looking for clustering in the detector plane histogram after every bunch with events more than a certain threshold. If clustering is seen, such time intervals are removed from further analysis. These are mainly the noise events that span more than 1 detector in every quadrant.

Input

- Event file after cleaning for noisy pixels
- Bunch file
- Livetime file

Method

- Identify all the bunches with events more than a threshold.
- Make the DPH of a quadrant for a duration of 100 ms after all such bunches identified in previous step.
- Check for clustering in the DPH using the algorithm described in Paul et al 2021. If clustering is found then remove the time interval from analysis.
- Updated pixel exposures.
- Update GTI for each quadrant.
- Write the events other than noise to the output event file.
- Update the livetime file with the new GTI.

Output

- Event file after removing noise events identified in this module.
- Livetime file after updating the fractional exposure in every bin.

7.17 cztheavybunchclean

The noise remaining after cztsuperbunchclean, caused due to local bunches in a module can be removed by their peculiar properties in time, DPH, and energy. These events are found until 200 ms, near the LLD of the module, and in the module where the cosmic ray has interacted. Further they are found to be clustered with another similar event. Using these properties we can distinguish such events from genuine X-rays. This module does this task.

Input

- Event file (output of cztsuperbunchclean)
- Bunch file

Method

- Identify all the bunches with events more than a threshold.
- Identify the module of the bunch, by taking the maximum occurence among all the available module IDs in that bunch report.
- For the next 200 ms (after the bunches identified in the previous step), check for events in the same module within 10 keV from the LLD of that module. If any such events are found then check if there is any other such event within 10 ms to it, and if found, then flag both the events as noise and remove them.
- Write the events other than noise to the output event file.

Output

- Event file after removing noise events identified in this module.

7.18 cztflickpixclean

The remaining pixel noises are identified in this module. Flickering pixels or pixels with noise at shorter time scales can be identified after cleaning for post bunch and gross noisy pixels. Each pixel is checked for its non-Poissonian behavior with respect to its own lightcurves normalized by the module count rate. Further, at 1 s and 10 s timescales, the DPH too is checked for non-poissonian behavior.

Input

- Event file (output of cztheavybunchclean)
- Badpix file (output of cztsuperbunchclean)

Method

- Generate lightcurves for each of the pixels in a quadrant. Normalize those lightcurves with the count rate of the respective module.
- If a pixel is found to go beyond a n sigma for m number of times, then flag the pixel as flickering and remove all events from it. n and m are decided by the exposure time of the observation.
- Generate DPHs at 1s and 10s intervals from the start to the end of the observation, and exclude any pixel going beyond the expected Poissonian limit for that time interval.

- Update pixel exposures.
- Write the events other than noise to the output event file.

Output

- Event file after removing noise events identified in this module.
- Updated badpix file

7.19 czteventsep

This module seperates the single and double events and write them into seperate event files.

Input

- Event file (output of cztflickpixclean)

Method

- Identify single events or events which do not have a simultaneous event in the same quadrant with 0.040 ms, and write them into an event file.
- Write events which have other simulataneous events into a seperate event file if prompted by the user.

Output

- Single event file
- Double event file (optional)

8 References

1. The Cadmium Zinc Telluride Imager on AstroSat, Bhalerao, V. et al., 2017, Journal of Astrophysics and Astronomy, DOI: 10.1007/s12036-017-9447-8.

2. AstroSat CZT Imager Observations of GRB 151006A: Timing, Spectroscopy, and Polarization Study, Rao, A. R. et al, 2016, ApJ, DOI:10.3847/1538-4357/833/1/86

3. Imaging calibration of AstroSat Cadmium Zinc Telluride Imager (CZTI), Vibhute, A. et al., 2021, Journal of Astrophysics and Astronomy,DOI:10.1007/s12036-021-09762-y

4. A generalized event selection algorithm for AstroSat CZT imager data, Ratheesh, A. et al., 2021, Journal of Astrophysics and Astronomy, DOI:10.1007/s12036-021-09716-4.

5. Characterisation of cosmic ray induced noise events in AstroSat-CZT imager, Paul, D. et al, 2021, Journal of Astrophysics and Astronomy, DOI:10.1007/s12036-021-09750-2.