URL: http://cxc.harvard.edu/ciao3.4/mtl.html
Last modified: December 2006

AHELP for CIAO 3.4

mtl

Context: chandra

Jump to: Description See Also

Synopsis

A description of the Mission Time Line (MTL)

Description

What is the mission time line?

The quick answer is that is it a set of observational parameters that have been rebinned to a common time line. So aspect data (at 32 second intervals) and ephemeris data (at 300 second intervals) and other engineering (HRMA temps), etc. Telescope pointing, as well as Solar Particle count rates and detector voltages, are all examples of "observational parameters". All these values are rebinned and are available as the *_mtl1.fits file.

What are the limits on the MTL?

There are actually two sets of limits that get applied to data. One is a static set of rules, the second is derived for each observation. The static data lives in the calibration data base ($ASCDS_INSTALL/CALDB/data/chandra/[ah]*/bcf/gtilim/*) and there are different ones for HRC−I, HRC−S and for ACIS. HRC−I also has a time variability factor.

The second set of limits applied to the data is derived by the Observation Determination pipelines (obidet). An observation can have multiple Observation Intervals (obis) if, for example, the telescope shuts down due to dangerous solar particle counts. The obidet limits are stored in the *_olims0a.fits files. Since they are pipeline products, they are not part of the standard data distribution, but are available using ChaSeR, which is available from the Chandra Data Archive pages.

How can I see which limits were violated and when?

The *_mtl1.fits file, for pipeline processing from DS6.1.0, has a third extension (visible via "dmlist infile=*_mtl1.fits opt=blocks") which contains a "smoothed" (historical name) MTL and the "limit_status" column.

The smoothed MTL extension will contain the data used to determine the GTI limits in your GTI, with two additional pieces of information. First, there will be a "limit_status" column. This is a bit column where each bit represents each limit. So, if you dmlist the file and look at the limit_status column you can tell by counting bits (from left to right) which limit was violated. For example, a value of 00000000000000000000000100000 would indicate that the 24th limit was violated. To find out what this limit actually means, we look at the "limits" extension of the file.

For this example (an HRC data set), we find that the violation occurred because:

```
dmlist "*_mtl1.fits[limits]" data
  ...
24 ((fabs(IMHVLV−79)<=1)&&(fabs(IMHBLV−86)<=1))
```

An easy way to see when a limit was violated

One way to see when any limit was violated is to create a new column which is 1 for all times that "good", 0 when a limit is violated, and then plot this column against time to see when the violations occur. The tool dmtcalc can be used to create the new column, as shown below:

```
dmtcalc smoothed_mtl.fits flagged_mtl.fits \
  exp="flag=(int)(limit_status==0)"
```

See Also

tools

dmgti, mtl_build_gti

URL: http://cxc.harvard.edu/ciao3.4/mtl.html
Last modified: December 2006