r e p o r t

MIT Kavli Institute
1 Kendall Sq., Building 300
Cambridge MA 02139–4307
Tel: 617-253-7277
Fax: 617-253-8084

To: ACIS Science Operations Team

From: Peter Ford, NE83-545 <pgf@space.mit.edu>

Date: September 12th 2022

Subject: Monitoring ACIS Background Rates for the TXings Patch (v 1.0)

1. Introduction

To provide ACIS with autonomous protection against high background radiation, the TXings flight software patch has been developed (see §9a,b). It monitors the rate at which the CCD detectors report "threshold crossings", *i.e.*, excess electric charge in their pixels, either from the observed X-ray sources or from background radiation. TXings averages the rates, normalizing them to units of threshold crossings per 100 CCD rows per second. The rates are separated according to whether they come from front-illuminated (FI) or back-illuminated (BI) CCDs, since the FI rates are an order of magnitude higher than the BI (see Fig.1). Consecutive rates are examined for patterns that best match background events rather than X-ray sources or instrumental artifacts. When TXings determines that a match is likely, it sets the ACIS bi-level channels to a pattern that is recognized by Chandra's on-board computer (OBC), which initiates a payload safing action. Since the average threshold crossing rate changes, often unpredictably, over the 11-year solar cycle, TXings parameters must be updated from time to time to account for this. This report describes how this is accomplished.

2. Daily Monitoring

ACIS telemetry is extracted on a daily basis from a combination of realtime data and data recorded onboard (SSR) and downlinked at a high rate. The latter contains fewer outages and bit errors than the former, but both are examined and compared to better identify data corruption. The telemetry is archived in phases, each containing about 60 days of data in 150-450 science runs, and each phase is further divided into 5 segments labelled a through e. Each segment begins and ends during a perigee passage when ACIS generates no data. Typically, each phase is archived into 10 Gb of SSR telemetry, 1 Gb of realtime telemetry, and 5 segments of decommutated X-ray events and ancillary data, each of 2 Gb.

Whenever new telemetry data is added to the archive, the entire segment is reprocessed. The threshold crossing rates are extracted from ACIS exposure packets by running the following command in the "`/nfs/maaxnew/r5/txings/`" directory:

```
cd /nfs/maaxnew/r5/pgf/txings
make rates N=nc
```

where integer *n* is the phase and lowercase letter *c* is the segment. This runs two scripts, *make-rates.sh* to extract ACIS crossing rates and *make-antico.sh* to do the same for the HRC anti-coincidence shield rates. These scripts write output files "`acis`*nc*`.txt`" in subdirectories "`DATA1`" and "`HRC1`", respectively.

The *make-rates.sh* script processes ACIS telemetry packets from the ACIS archive through the *txings_test* program which is compiled with the TXings patch routines, ensuring that the threshold crossing information is treated in the same manner as in the flight unit. Each time that TXings running onboard would compute new average rates, *txings_test* will output an ASCII record containing the phase and science run number within the phase, its OBSID index, the starting time of the integration, the number of seconds since start-of-run, a "1" denoting that TXings has triggered or "0" otherwise, the current FI and BI ascending rate limits, and the average FI and BI rates over that integration interval. The *make-antico.sh* script is simpler: it runs "*ehs2mnf …* | *find-antico.pl*" to extract the HRC channels directly from SSR files in the ACIS archive.

The final step in the "`make rates`" procedure is to execute *gnuplot* to re-read the "`acis`*nc*`.txt`" and create plots in PostScript and GIF format displaying the ACIS and HRC rates (if available) for each day in the phase segment. The plots are written to the *acisweb.mit.edu* web server. They are examined to select "Runs of Interest" (see §9.i) to include in the set of observations used to derive optimal TXings parameter values as described in §9.c.

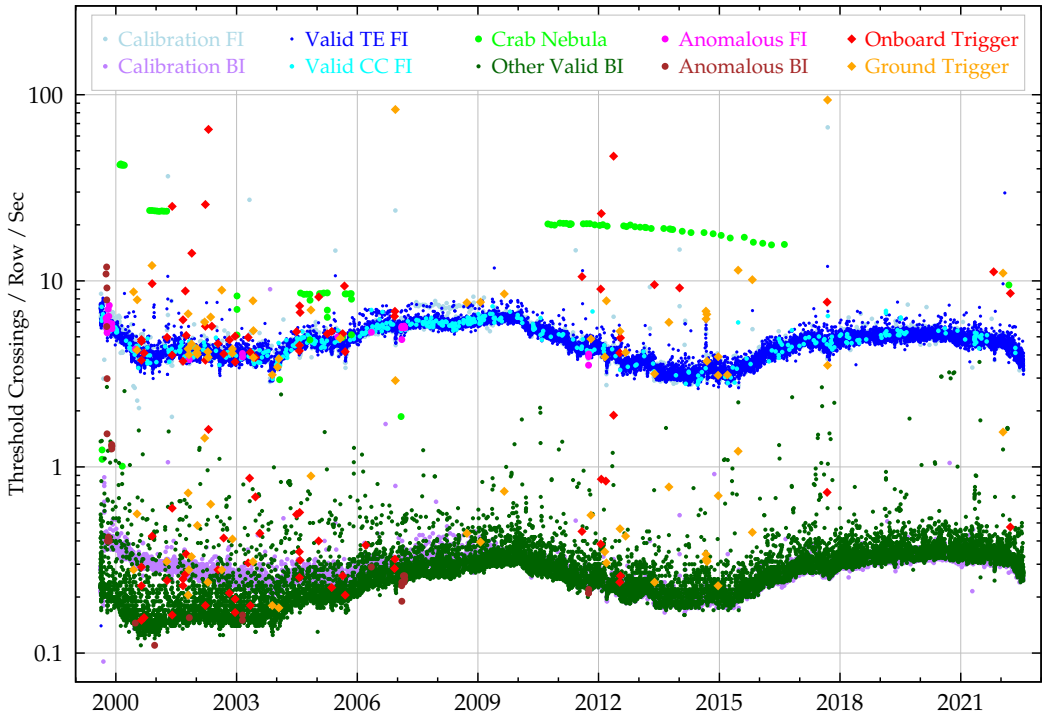
3. Long-Term Monitoring

At the completion of each processing phase, several ACIS data sets are examined for long-term trending. In the case of threshold crossings, this consists of averaging the FI and BI rates over the mission lifetime and plotting the result in various formats. The procedure is similar to daily monitoring:

```
cd /nfs/maaxnew/r5/pgf/txings
make N=n
```

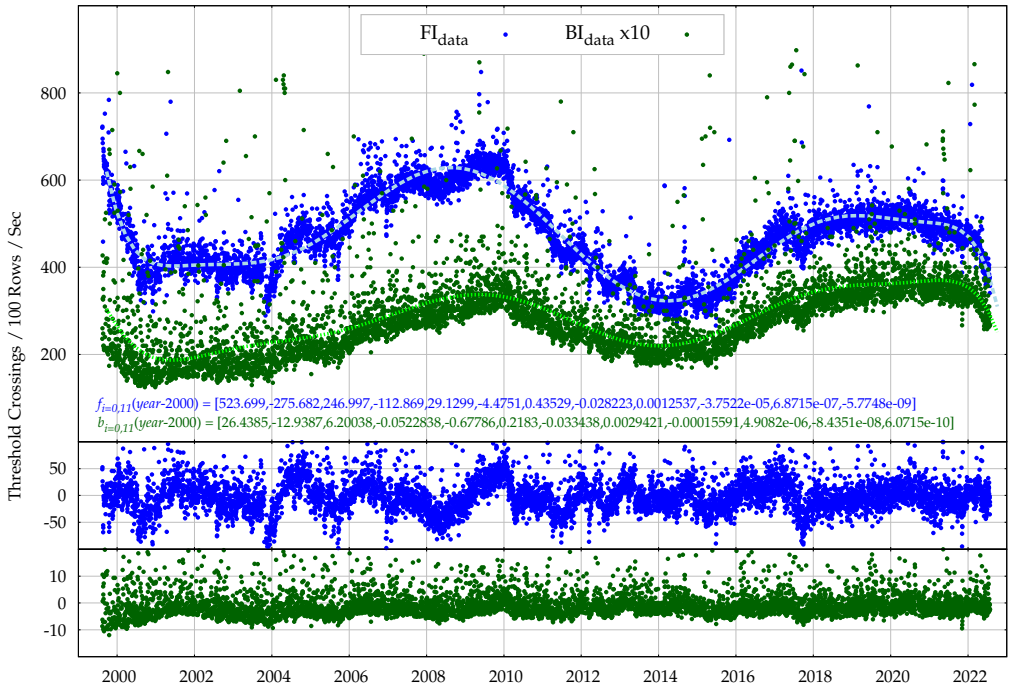
where *n* is the integer phase number. The threshold crossings are extracted from the "`science.log`" files in the ACIS archive by the *xings.pl* script which writes 3-minute average crossing rates of each CCD into "`DATA/txings-`*n*`.txt`". The output from all phases is concatenated into "`xings.txt`" and processed into data products. The first product, "`gplot.pdf`", is shown in Fig.1 where the rates are further averaged over each run and plotted in colors that indicate their assigned categories (defined in "`txings_parms.h`"). High radiation autonomous shut-downs (from EPHIN, HRC, or ACIS) are indicated in red and shut-downs commanded from the ground are in orange. Calibration runs are mostly hidden behind science runs except for BI CTI runs prior to 2009, when crossings from the X-ray calibration source dominated over charged particle background.

Fig.1: Threshold crossing rates for various categories of observing run



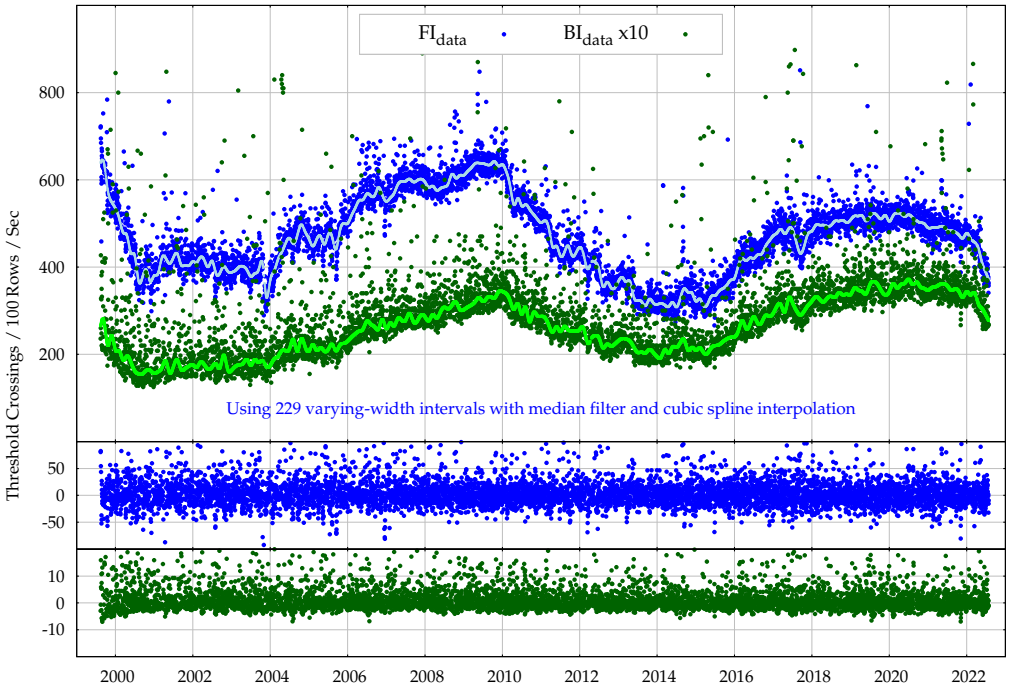
The second product shown in Fig.2 (and online in §9.e) fits the rates of "valid" observing runs (*i.e.*, those not marked as special in "`txings_parms.h`") to 11th order polynomials in years since 2000.0.

Fig.2: Polynomial fits to the threshold crossing rates of valid observing runs



While polynomial fits are a convenient way to parametrize the long-term variation of the crossing rates, they are subject to ±10% variations that could lead TXings to trigger on false positives or not on false negatives. A better approximation is to use splines, shown in Fig.3 (and online in §9.f), in which the median rates were calculated at 36-day intervals and fitted with cubic splines. Comparing the residuals between Fig.2 and Fig.3 shows that the spline interpolation is superior, but as reported in §9.c, the choice of parametrization has little effect on the choice of optimal TXings parameters.

Fig.3: Cubic spline fits to the threshold crossing rates of valid observing runs



4. TXings Parameters

The patch is controlled by the values of 22 32-bit integers. The 6 independent tests performed by TXings — on data from FI and BI chips, looking for ascending, descending, and very large crossing rates — are determined by the three 6-element arrays and `MINUTES`, the common integration interval.

```
struct _TX {
    unsigned MINUTES;              // integration interval x 64 seconds
    unsigned TX_MODE;              // txings mode (flags to ignore CCDs, etc)
    unsigned MAX_TX_PER_ROW;       // max crossings per row
    unsigned CC_TICKS;             // ticks per frame in CC mode
    unsigned TRIGGER_COUNT[6];     // number of integration steps before trigger
    unsigned RATE_LIMIT[6];        // trigger thresholds per 100 rows per sec
    unsigned TX_INCR[6];           // trigger threshold increments/decrements
} TX;
```

When first loaded via a warm boot, the default values are as follows:

```
MINUTES        = 3                       // integration interval 192 seconds
TX_MODE        = 0                       // txings mode (accept all CCDs)
MAX_TX_PER_ROW = 512                     // no more than 512 crossings per row
CC_TICKS       = 291840                  // exposure time 2.9184 sec in CC mode
TRIGGER_COUNT  = 4 6 5 6 4 6             // integration steps before trigger
RATE_LIMIT     = 458 23 498 48 1298 4493 // rate thresholds per 100 rows per sec
TX_INCR        = 5 2 4 6 0 0             // rate threshold increments/decrements
```

These values were determined by running the *txings_test* program on the 22 years of observations in the ACIS archive, concentrating on the runs of interest as described in §9.c. When used to compute crossing rates for daily monitoring, as described in §2, *txings_test* uses the actual on-board parameters at the time of each run, but when used to determine optimal parameters, the `RATE_LIMIT` values are modified by the average rates at the time of each run, as determined by the polynomial or spline fits in Figs. 2 and 3, and *txings_test* reports only whether the run would have caused TXings to trigger.

The default rates listed are the optimal values with the `RATE_LIMIT`s set for the epoch at which the average FI and BI rates were minimal. While the polynomial- and spline-adjusted limits each found 41 triggers from the 25,000+ runs in the ACIS archive (see §9.h), *txings_test* with the default `RATE_LIMIT` values found 1382. The limits must be adjusted to account for the varying averages.

Table 1: Number of RoIs causing True and False Triggers as rate Rate Limits are varied away from optimal

Front Illuminated			Back Illuminated		
δ limit	True	False	δ limit	True	False
−50	41	5	−25	13	5
−40	40	5	−20	20	5
−30	39	2	−15	15	5
−20	39	1	−10	15	4
−10	39	0	−5	12	1
0	39	0	0	11	0
10	34	0	5	11	0
20	33	0	10	9	0
30	32	0	15	9	0
40	32	0	20	8	0
50	32	0	25	8	0

Table 1 shows how the number of true and false triggers varies as the `RATE_LIMIT`s are moved away from their optimal values. The sudden drop in FI true triggers as the FI limits are increased from optimal to optimal+10 is partly biased by the incomplete data archive: EPHIN and HRC required less time to trigger than ACIS, after which no crossing rates were telemetered. Nevertheless, 2 of the 5 runs that didn't trigger at 10 above optimal FI were not associated with autonomous shutdowns: one was an ascending rate, the other descending, showing the sensitivity of the FI algorithm to realistic `RATE_LIMIT` values.

5. Extrapolation

The *cplot4* script fits a selected period of crossing rate data to a linear or quadratic function and computes the optimal TXings `RATE_LIMIT` value at some future date. The rate data are read from a file created by running the "`DATA1/acis`*nc*`.txt`" files in "`/nfs/maaxnew/r5/pgf/txings`" through the *cplot.pl* script. *cplot4* may be controlled by command-line options or by *gnuplot* commands of the form "`variable = value`" in a separate file. Dates are entered as "*year–month–day*", *e.g.*, "`2022-9-15`". Type "`man cplot4`" for details.

Fig.4: cplot4 output showing a linear rate extrapolation to September 15th 2022

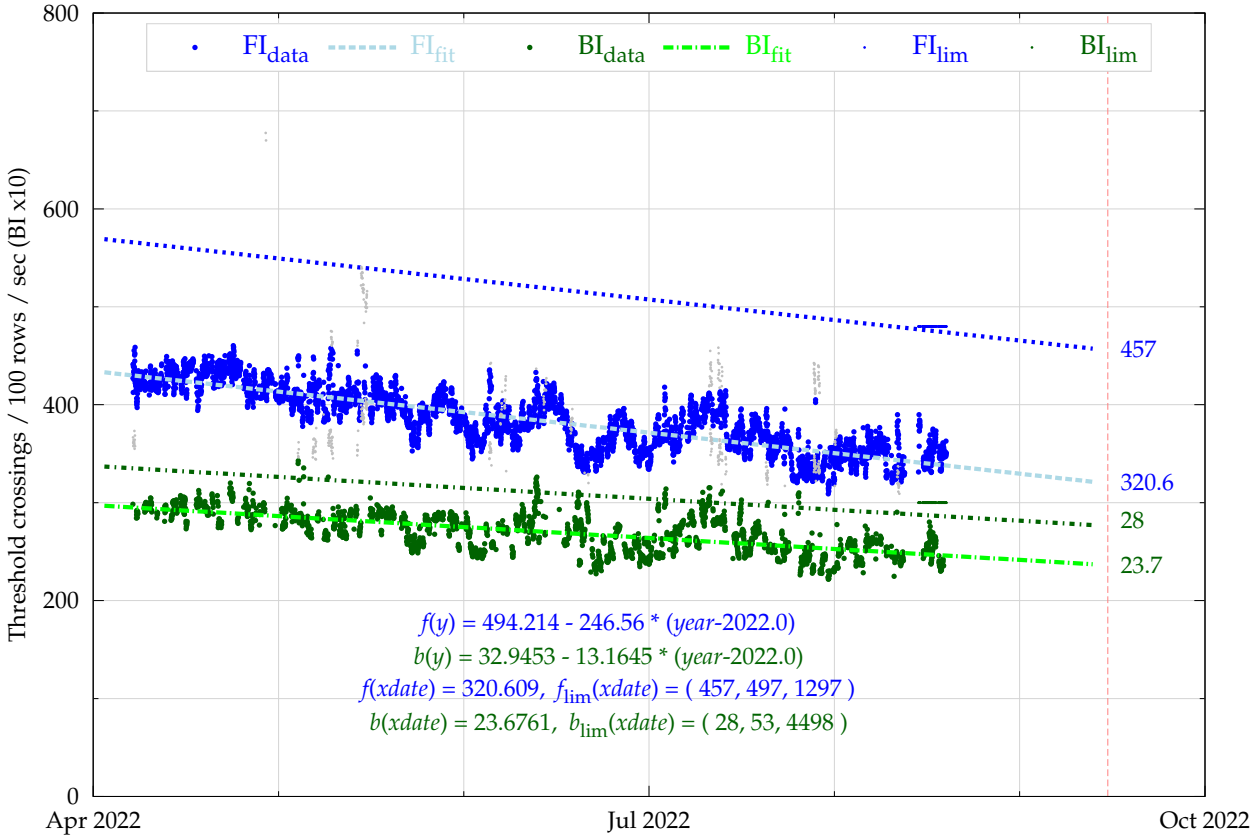


Fig.4 shows a linear *cplot4* extrapolation of rates from April 1st through August 18th 2022. The average FI (ascending) and BI (ascending) rates are estimated to be 320.6 and 23.07, respectively, on September 15th, and the optimal `RATE_LIMIT`s, rounded to the nearest integer, are 457 and 28. Since TXings parameter blocks are created with FI rates in multiples of 10 and BI rates in multiples of 5, the optimal TXings parameter load for September 15th would be 460 and 30. The plot was created by the following commands:

```
cd /nfs/maaxnew/r5/pgf/txings/QTLY
../cplot.pl -s 10 ../txings_parms.h ../DATA1/acis1???.txt > cplot.txt
cplot4 -1 -F 2022-4-1 -f 2022-4-1 -r B -T 2022-10-1 -d cplot.txt 2022-9-15
```

The *Makefile* in that directory is kept up to date so typing "`make`" followed by "`make update`" will refresh the current extrapolation, display it using *xpdf*, and push it to the *acisweb* server (see §9.j).

6. ACIS Command Tables

Once the optimal extrapolated rates are known, and are not already present in the ACIS offline command table, a new `WBTX` command must be created. The SACGS maintainer (see §9.ℓ) first assigns the new command a `commandIdentifier`, an integer `0–65535` that uniquely refers to this ACIS command. Once the identifiers are known, the *make-txcmd.pl* script can create a binary ACIS command table with one or more `WBTX` commands, *e.g.*,

```
make-txcmd.pl WBTXB48030,18726 WBTXB47030,18723 > txings.dat
```

where "`WB`" indicates an ACIS w*riteBep* command, "`TX`" refers to TXings, "`B`" is a character unique to this version of the TXings patch (which is also currently the version of the patch itself, but may change in future), "`480`" and "`470`" are the FI ascending rates and "`30`" is the common BI ascending rate of the two commands. The optional "`,`*n*" suffixes assign the unique `commandIdentifier` fields to the *writeBep* commands (see §9.d). "*make-txcmd.pl*" uses the 5th character of the packet name to determine the address of the BEP's "`TXinit`" block that contains the TXings parameters, and the values of the other parameters.

For convenience, executing *make-txcmd.pl* with the **–S** flag will write the command(s) in the format used by the SACGS as input to an ACIS command table:

```
make-txcmd.pl -S WBTXB48030,18726
# WBTXB48030
write 18726 0x8003dc30 { ## WBTXB48030
  3 0 512 291840
  4 6 5 6 4 6
  480 30 520 55 1320 4500
  5 2 4 6 0 0
  3 0 512 291840
  4 6 5 6 4 6
  480 30 520 55 1320 4500
  5 2 4 6 0 0
}
```

The individual command entries in an ACIS command table can be listed in several formats using the *lpkt* command, as in the following example. The `-B` flag causes the parameter values to be written in decimal. By default, *lpkt* will look for commands in "`/nfs/acis/h1/www/bin/current.dat`". Type "`man lpkt`" for details.

```
# lpkt -B WBTXB47030
write 18723 0x8003dc30 {
            3           0         512      291840
            4           6           5           6
            4           6         470          30
          510          55        1310        4500
            5           2           4           6
            0           0           3           0
          512      291840           4           6
            5           6           4           6
          470          30         510          55
         1310        4500           5           2
            4           6           0           0
}
```

7. Testing

Before an ACIS command table containing new **WBTX** command packets is submitted to the Chandra Offline System, it must first be tested on the ACIS software and/or hardware simulators: the QEMU simulator and the Engineering Unit (EU). For this, there are two test scripts, *txparam* and *txpchtest*. The first loads software patches before testing WBTX commands; the second assumes that the patches are already loaded. Here is an example of *txparam* using the EU:

```
acisShim -h cypress eu >& eulog.txt &
sleep 200
makecfg.pl GIJ > GIJ.cfg
ln -s `find $ACISFS -name options-release-G-opt-I`/opt_txings.map GIJ.map
txparam -l ./GIJ.cfg -p GIJ -- WBTXB48030
```

The output is as follows:

```
info: dryrun:vanilla; patches(GIJ); txings updates
info: selecting BEP a; coldboot; load patches; warmboot; dump icache (ref);
      dump dcache (ref); dump txings (ref)
info: WBTXB48030: warmboot; sending; dumping icache; dumping dcache;
      dumping txings
info: WBTXB48030: i-cache okay
info: WBTXB48030: d-cache okay
info: WBTXB48030: txings okay
```

When the EU interface is no longer needed, the user should bring the *acisShim* task into the foreground and kill it by typing CTRL-C. To run the same test on the QEMU emulator, the *acisShim* and *sleep* commands should be replaced with:

```
acisQEMUrun –s -n >& qemulog.txt &
```

where the **–s** flag starts the *acisShim* interface and **–n** closes the emulator's standard input. The *txpchtest* command is quick and easy on the QEMU simulator since the latter can be told to pre-load its patches (the **–p** option) and write its telemetry at high speed (**–f** flag):

```
acisQEMUrun -f -n -s -p acis-GIJ.bin >& qemulog.txt &
txpchtest -p GIJ WBTXB48030
```

The output is as follows:

```
Starting txpchtest at ...

Patch:   GIJ
Map:     /nfs/acis/h3/acisfs/patchbld/release-G-opt-I/release/dist/\
         options-release-G-opt-I/opt_ccignore.map
TabDir:  /nfs/acis/h1/www/bin
Table:   current.dat
Test:    current.dat
Time:    10
DCache:  0x80000000 0x80040000
ICache:  0x80080000 0x80100000
Scrat:   /tmp/txpch179232
Pkts:    WBTXB48030
```

```
Turning off all FEPs and video boards
Dumping unmodified Icache and Dcache
Executing WBTXB48030
Testing WBTXB48030
Comparing Icache 1-4: matches
Comparing Dcache 2-5: matches
WBTXB48030: RATE_LIMIT      = { 480, 30, 520, 55, 1320, 4500 }

===  Ending txpchtest at ...  ===
*** PASS ***
```

Finally, running *txpchtest* on the EU uses *makecfg.pl* to generate the commands necessary to cold boot the BEP, upload the necessary patches, and warm boot to install them.

```
acisShim -h cypress eu >& eulog.txt &
sleep 200
makecfg.pl -r GIJ | cmdClient -c -d
txpchtest -p GIJ WBTXB48030
```

8. Timeline

MKI will run *cplot4* once a day to extrapolate the crossing rates to the middle of the following month, with results available on the Web (§9.j).

a. The *cplot4* extrapolated limits will be discussed at each monthly meeting of the ACIS operations team. If no action is required, repeat the discussion at the next meeting.

b. If the closest `WBTX` command is already included in the current Offline System table, skip to step g.

c. Obtain a `commandIdentifier` for the new command from the SACGS operator and use an editor or run "`make-txcmd.pl -S`" to create a new `write` command in *bcmd* format, as described in §6, above.

d. Remake the ACIS offline table, including the new `WBTX` command, by running the SACGS scripts *cmd2dat* and *log2cfg*. For details, see Steps 3.1 and 4.1 of the "Short Form" operating scenarios in §9.ℓ.

e. Test the `WBTX` command in the new ACIS offline table with *txparam* or *txpchtest* as described in §7, above.

f. Transfer the new ACIS table to the Offline System.

g. Request that the FOT build a realtime CLD file to execute the `WBTX` command.

h. Write a CAP based on §9.k to uplink the new TXings parameters as close to the extrapolated date as possible. The CAP should first execute the `WBTX` command and then `RBTXINGBLL` to dump the newly installed parameters.

i. Execute the CAP and verify that the fields of the *bepReadReply* packet coming from `RBTXINGBLL` match those in the `WBTX` command.

9. References

a. "Using ACIS on the Chandra X-ray Observatory as a particle radiation monitor," C. E. Grant, B. LaMarr, M. W. Bautz and S. L. O'Dell, SPIE, June 2010.

b. "Flight S/W patch to report high background radiation levels, Rev B," MIT ECO-1058, April 20, 2022.

c. "Optimal Parameters for Release B of the TXings Patch," MIT Software Report, Rev. 1.5, June 6, 2022.

d. "ACIS Software IP&CL Structure Definition Notes," MIT 36-53204.0204, Revision N, March 15, 2001.

e. "Polynomial Fits to ACIS Crossing Rates," ACIS web page, June 1, 2022.

f. "Cubic Spline Fits to ACIS Crossing Rates," ACIS web page, June 1, 2022.

g. "ACIS Threshold Crossing Rates," ACIS web page, June 4, 2022.
h. "Optimum ACIS TXings Triggers ," ACIS web page, June 4, 2022.
i. "ACIS Runs of Interest (RoIs)," MIT web page, June 4, 2022.
j. "Extrapolated ACIS Threshold Crossing Rates," MIT web page, updated frequently.
k. "Update TXINGS Parameter Values," Chandra CAP 1622, July 28, 2022.
l. "SACGS User's Guide," SAO online manual.

10. Glossary

BEP ..... ACIS Back End Processor—the unit that interfaces between FEPs and RCTU.
BI ..... A CCD that detects x-rays incident on the opposite face to its junctions.
Bi-Level ..... A one-bit data channel from BEP to RCTU.
CAP ..... Command Action Procedure
D-cache ..... The radiation-hard data cache memory used in BEPs and FEPs.
EPHIN ..... Electron, Proton and Helium Instrument—flown on Chandra and SOHO.
EU ..... Engineering Unit—the ACIS hardware simulator at MIT.
FEP ..... ACIS Front End Processor—extracts event candidates from a pixel stream.
FI ..... A CCD that detects x-rays incident on the same face as its junctions.
Gb ..... Gigabytes
HRC ..... High Resolution Camera—sharing the Chandra payload with ACIS.
IP&CL ..... Instrumentation Program and Command List – Chandra command and telemetry formats.
OBC ..... Chandra's On-Board Computer.
OBSID ..... Observation ID—a unique integer associated with an observing run.
QEMU ..... A generic open-source machine emulator and virtualizer.
RCTU ..... Remote Command and Telemetry Unit.
RoI ..... Run of Interest—an ACIS observation that may contain high background.
SACGS ..... SOT ACIS Command Generation System
SOT ..... Science Operations Team
spline ..... A means of fitting a function by a series of smooth polynomials.
SSR ..... Solid State Recorder.
TX ..... An array in BEP memory that contains TXings parameters.
tx ..... An array in BEP memory that contains TXings accumulators, etc.