1 Overview of XRT Housekeeping Data Processing

Processing of the XRT Housekeeping data is done in two parts:

On the J-side, data are extracted from the Data Distributor and stored in .DAT format. Data are rsync'ed to SAO and the .DAT files are converted to ASCII .LOG format which are subsequently parsed and plotted via perl & IDL scripts. Additionally, the data are compared against Health & Safety (H&S) yellow/red limits and emails are sent if the limits are violated.

1.1 J-Side

On the J-side, the 'merge_bands.sh' script is used to run xskim hourly via a cron-job. It requests the data for all band and antenna combinations that it knows about for the date the script is run. Since we do not know what data are ingested; it's even possible/likely that data acquired on the previous day is ingested on the current day.

The directory structure encodes the year, month, and date; this is used to seed the time conversion code with the 'rough' time it needs.

The executables are in ~sbusxrt/install on norway. The source code in in ~sbusxrt/staff/kjg/

1.1.1 Modification to .DAT to include s/c time-tag into proto-packet.

The spacecraft time recorded in the PDU needs to be available to the code that does the time conversion in xdump, xmerge, and dat2fits. A modification was made to xskim to extract this 4-byte time from the PDU and store it in the time-stamp record that begins at byte 8 in the .DAT format file. This change is recognized by kProtocolVersion = 102.

in xskim.cc:

```
<    packet.timestamp = RULong(ms);
---
>    packet.timestamp = rec->spacecraft_time;
```

References to 'timestamp recording when command was sent to XRT, in milliseconds since log file started' should be updated with something like 'timestamp extracted from the PDU packet enclosing the XRT data'

1.1.2 Sirius db vs. Data Distributor

	Sirius	Data Distributor
Request data for which antenna	All at once	Individually, one by one
Request data for which band	All at once	Individually, one by one
How retrieved data are sorted	Time sorted by data acquisition	Time sorted by data ingest, often out of acquisition time order

Times used to requested data	Time data were acquired	Time data were ingested
When are data available	Substantial latency between data acquisition and data ingest.	Data are available shortly after acquisition
Are duplicate real time and dump data packets included	Yes	Yes
Potential for gaps in data	No	Yes

So, there is a significant trade-off: The Sirius database (Sdb) provides data in a logical, time ordered fashion but with large latency. The Data distributor (DD) provides data more quickly, but one doesn't know what data are available when. There was at one point a file that was being maintained that gave the Antenna/Band with dates/time of ingest that could in principal be parsed to remove the need to loop through the band/antenna combinations.

1.1.3 Time conversion, needs clock files

The J-side provided the 'sbtime' library to convert space craft times to Universal time (TBD). This library needs 3 inputs: current clock-time, rough estimate of the time, and an external file that correlates the two. This time file named, CNV_FILE_yyyy (where yyyy is the 4-digit year) is updated by (some manual) process several times each day and the copy is continually overwritten. The sbtime library does do some forward extrapolation if the latest time entry is not available*; however, we have seen instances where when the file isn't updated we can get times that deviate significantly from reality (months off)*. The sbtime library also uses the 'Lepasec.dat' file (yes, spelled incorrectly) to get the needed information about leap-seconds.

Since the times are computed by both xdump (SAO) and xmerge (J-side) these files get included in the directories that are rsync'ed over.

1.1.4 Accessing Sirius database

Note: Originally, separate application were required to access the Sdb and the DD due to the J-side providing separate libraries with different APIs. *I believe that a new version of the SDTP library was released that allowed access to both using a unique SOCKFILE. However, the limitations with the 'All' antenna and difficulties with time based queries remain.*

To access the Sdb, one uses the xskim_sirius application. The flags are identical to xskim with the exception that it allows -a all and -b all to access all bands and all antenna.

1.2 Processing @ SAO

Data are rsync'ed hourly from the directory on the J-side to SAO via cron job (machine: kurasuta, user: xrt_data) script and stored in a mirror directory structure at SAO.

The processing that is performed at SAO includes: running xmerge to combined the .DAT files from multiple antenna (and possibly multiple bands; now irrelevant). Running xdump on the merged .DAT file to generate the ASCII .LOG file format which is subsequently parsed by the /data/solarb/XRT/hk_pro/make_hk_plots_merge.cron script to generate engineering plots, and to run xdump a 2nd time in Health & Safety (H&S) mode to evaluate the values against Yellow and Red limits.

Early in the mission; all data were merged and dumped on the J-side; recently those operations have been moved to SAO; they are run in the machine 'machida' (user: xrt_data); the software is installed in /data/solarb/XRT/hk_pro/xdump_install

1.2.1 H&S Alerts

The H&S alerts are encoded in a self-describing ASCII file

/data/solarb/XRT/hk_pro/xdump_install/xrt_limits

The real-valued temperatures, voltages, and currents are adjustable via this file; however, bit-encoded flags checks are hard-coded in the software.

Since the data for the same .DAT are continually updated and processed over and over again throughout the day, it is necessary to create a bookmark to let the processing know which H&S violations have already been reported. This bookmark is the xrt_limits.state file in that same directory.

1.2.2 Dat2fits: FITS HK files

The utility 'dat2fits' was written to take the .DAT format files and convert them into a series of FITS files : a set of files representing each packet type (Normal, Extended, etc) that would be written on hourly (configurable) boundaries.

The expectation was that these FITS files would be generated from the complete set of data extracted from the Sirius Database (Sdb). As the .DAT format files are not extracted from the Sdb, the .FITS file creation was never added to the processing thread.

2 Adding additional bands

The code modifications needed to add additional antenna are outlined below. Both must happen on the J-side.

2.1 Xskim

xskim.h : add new #define ANT_FOO <value>

xskim.cc: in main() routine, add ANT_FOO into Antenna parsing list
in PrintOpenArguments() routine, add ANT_FOO into switch() statement.

2.2 merge_bands.sh

merge_bands.sh: add the new antenna into the list of antenna being checked

as long as the new antenna starts with a lower-case file name; the rest of the processing should continue without modifications.