FOXSI Science User Manual

February 28, 2020

Abstract

This user manual describes the FOXSI rocket data analysis within the FOXSI Science analysis software available on GitHub `https://github.com/foxsi/foxsi-science`. This software works in the framework of the Solarsoft, developed in IDL. This will work with IDL version...

To install the Science software, see chapter 1. To download FOXSI data and for a description of the data files, see chapter 2. To use the software, see explanations and examples in chapters 3 to 6. For instruction to create the different levels of data from the raw data file (only for advanced users), see chapter 7.

Contents

1 Setting the Solarsoft environment, installing the FOXSI software

1.1 Prerequisite

FOXSI software is written in IDL and requires SSWIDL (SolarSoft) to be installed. Installation of SSWIDL is described here: `http://www.mssl.ucl.ac.uk/surf/sswdoc/solarsoft/ssw_install_howto.html`. The FOXSI software can be use to analyze the FOXSI hard X-ray data. For more information on the analysis of the soft X-ray PhoEnIX data, see section 2.1.2.

1.2 Getting the FOXSI Science software ready in Solarsoft

FOXSI analysis software is kept in a GitHub repository at: `https://github.com/foxsi/foxsi-science`. A snapshot of the software can be downloaded as a ZIP file from the GitHub. Alternatively, one can sign up for a GitHub account and use git to keep the software up to date. This second choice takes some knowledge or studying about **git** but is useful if you want to contribute your additions and changes.

Git expects the main directory of the software tree to be foxsi-science. In addition to the directories in the software tree, you need to add three additional directories, called `data_2012`, `data_2014`, and `data_2018`. These directories need to be included in a directory called `foxsidb`. These directories are **not** included in the GitHub repository because they contain large data files.

1.2.1 SSWIDL setup

Include this five lines in your personal `IDL_STARTUP` (Mac) or `IDL_STARTUP_WINDOWS` (Windows) file, located in `ssw/gen/idl/ssw_system/`:

```
setenv,'FOXSIPKG=My/personal/path/to/foxsi-science'
setenv,'FOXSIDB=My/other/personal/path/to/foxsidb'
add_path,'$FOXSIPKG',/expand
add_path,'$FOXSIDB'
@init_param
```

1.2.2 Database

Create a new folder called `foxsidb` in a convenient location for you. This folder should contain three subfolders called `data_2012`, `data_2014` and `data_2018`. Please refer to section 2.1 to download the relevant data files.

2 FOXSI rocket flight data

2.1 Getting FOXSI rocket data

The FOXSI data is provided either in the form of IDL .sav files or in standard FITS files. Only level 2 data is available in FITS format.

The FOXSI level 2 data FITS files can be downloaded from the Virtual Solar Observatory (VSO): https://sdac.virtualsolar.org/cgi/search?time=1&instrument=1&inst_src=1&version=current&build=1.

All levels of FOXSI data (levels 0, 1, 2), in the form of IDL .sav files, can be downloaded from ftp://apollo.ssl.berkeley.edu/pub/foxsi.

Note regarding the PhoEnIX data: the PhoEnIX data is also available on the ftp server mentioned above. However, in the VSO, currently only quicklook data are provided. Please refer to section 2.1.2 for a description of the data.

2.1.1 Hard X-ray level 2 data

FOXSI data analysis can be carried out with the level 2 data. Level 2 data are provided either in IDL .sav files or in standard FITS files. Please refer to one of the sections below depending on your preference regarding the file format.

FOXSI level 2 .sav files

FOXSI data can be downloaded from ftp://apollo.ssl.berkeley.edu/pub/foxsi. Go to the folder associated with the flight of interest (2012, 2014, 2018) and download the following files: `foxsi_level2_data.sav`

Keep these files in the associated directories (`data_2012`, `data_2014`, `data_2018`) in the FOXSI database on your computer (folder `foxsidb`, instructions about the creation of these directories are in section 1.2).

There is no need to download the content of `calibration_data` folder on the FTP server. The most up-to-date calibration files are provided directly in the GitHub repository (see section 1.2): the necessary components to build the instrument response using whichever pieces (optics, detector, blanketing) are desired.

FOXSI level 2 FITS files

FOXSI level 2 FITS files can be downloaded from the VSO: https://sdac.virtualsolar.org/cgi/search?time=1&instrument=1&inst_src=1&version=current&build=1.

It is preferable to keep these files in the associated directories (`data_2012`, `data_2014`, `data_2018`) in the FOXSI database on your computer (folder `foxsidb`, instructions about the creation of these directories are in section 1.2).

Once the files are downloaded....

You are in principle ready to go! To setup the IDL environmennt, see section 1.2.

To start analyzing the data, please refer to section 3.

2.1.2 PhoEnIX (soft X-ray) data

The full, uncalibrated PhoEnIX dataset is available at ftp://apollo.ssl.berkeley.edu/pub/foxsi. This dataset is a list of counts as registered in the PhoEnIX detector. The calibrated list of events will be provided soon. In the meantime, quicklook PhoEnIX data are provided on the VSO: https://sdac.virtualsolar.org/cgi/search?time=1&instrument=1&inst_src=1&version=current&build=1. This data consist of nine full-Sun images in different energy bands. Since the field of view of PhoEnIX does not cover the entire solar disk, these images are mosaic images constructed from the observations of the different targets during the flight.

2.1.3 Detector and optic module pairing for each rocket flight

The FOXSI sounding rocket is composed of seven telescopes (for descriptions of the experiments, see e.g. [Krucker et al., 2013, Glesener et al., 2016]). During the different flights of the FOXSI sounding rockets, possible detectors include Si and CdTe hard X-ray detectors, and a SXR detector (PhoEnIX). Optics modules have seven or ten shells, and can sometimes be paired with a collimator. The summary of the different pairing is given for the three flights of the sounding rocket in table 1.

FOXSI-1 Detector - Optic pairs			
Position number	Detector type	Number of optic shells	Collimator
0	Si	7	
1	Si	7	
2	Si	7	
3	Si	7	
4	Si	7	
5	Si	7	
6	Si	7	

FOXSI-2 Detector - Optic pairs			
Position number	Detector type	Number of optic shells	Collimator
0	Si	7	
1	Si	7	
2	CdTe	10	
3	CdTe	7	
4	Si	7	
5	Si	7	
6	Si	10	

FOXSI-3 Detector - Optic pairs			
Position number	Detector type	Number of optic shells	Collimator
0	Si	10	
1	PhoEnIX	7	Yes
2	Si	7	Yes
3	CdTe	7	
4	Si	10	
5	CdTe	10	
6	Si	10	

Table 1: Detector - Optic pairs for the different FOXSI flights

2.2 Hard X-ray data description

The flight data is the recorded data files at White Sands Missile Range. Levels 0, 1, 2 are IDL data structures (the data volume is sufficiently small that data can be stored as IDL structures in IDL save files, as opposed to fits files). These data format are discussed in this section. Housekeeping data are discussed in appendix A.

There is an important difference between the data structures of FOXSI-1 and the following flights. For FOXSI-1, we kept all triggered events in the higher-level data, even those with the HV off. This was so that preflight data could be used for debugging and troubleshooting as the analysis code was written and the flight data was understood. These data prior to the bias voltage reaching 200V are not useful for analysis, so they are not included in the FOXSI-2 or FOXSI-3 Level 1 and Level 2 data. As a result, the IDL save

files are much smaller for the last two flights. These extra events are still included in the FOXSI-1 higher level data for backward compatibility.

The FOXSI observations covers different targets, e.g. different pointing positions, during each flight. They are labelled as target 1 to 5. Within targets, some pointing adjustments were sometimes performed, and therefore, the different pointings are refered as position 0, position 1, and so on.

Timing and position information for the different targets in a given flight are derived from the SPARCS report and are available in the parameter files in the `parameters` folder of the FOXSI-Science distribution:

- FOXSI-1 parameters: `param2012_20160620.pro`
- FOXSI-2 parameters: `param2014_20160620.pro`
- FOXSI-3 parameters: `param2018_20181003.pro`

2.2.1 Flight data

Recorded flight data is in two formats:

1. File recorded on FOXSI GSE computer: a binary file identical in format to the calibration files with the formatter interface. Each data frame is 256 words starting with the second sync word 0xF628.
   - FOXSI-1 Filename: `data_121102_114631.dat`
   - FOXSI-2 Filename: `data_141211_115911.dat`
   - FOXSI-3 Filename: `data_180907_111001.dat`
2. File recorded by WSMR ground station: the official data record. This binary file contains our data frames with 3 additional words inserted between the second sync word (F628) and the start of our housekeeping data. The three extra words contain time and sync lock info from the ground station. This file was recorded upstream of our GSE computer and thus has fewer opportunities for sync problems (though note there seems to be a 36 time lag for FOXSI-2, cause unknown).
   - FOXSI-1 Filename: `36.255_TM2_Flight_2012-11-02.log`
   - FOXSI-2 Filename: `36_295_Krucker_FLIGHT_HOT_TM2.log`
   - FOXSI-3 Filename: `36_325_Glesener_FLIGHT_HOT_TM2.log`

Note that some dropouts and sync problems are present in both files, on the order of 0.1% of the frames. However, most of these dropouts are preflight or during burnouts.

2.2.2 Level 0 data

The FOXSI level 0 data structure captures all the unprocessed, raw data, organized with each element representing one **triggered event** in one detector. Thus not all data frames have events, and there can be more than one event (in multiple detectors) per frame. The trigger can be caused by a photon interaction or by noise; in this documentation well call both of these **hits**. For convenience, the hit ASIC, strip, and ADC values are specifically called out for the two sides these are determined by identifying which ASIC or strip had the maximum value on each side (at this stage, the calculation is done without subtracting common mode values). However all the recorded detector data, including all transmitted strip data and channel masks, are included in the structure. Together with the housekeeping structure, the level 0 data carries all of the information from the recorded data file, and the raw data file should no longer be needed. Some data frames are flagged as bad packets based on the common mode values, but no data is thrown away at this point, unless all words for that detector are zero in the frame (i.e. no trigger). Events that occurred after the rocket launched and HV ramp neared 200V have an "inflight" flag set.

Note that the time recorded by WSMR is in UT and has microsecond precision BUT time delays of transmission from 100-300km limit this to ∼ms precision unless we do additional refinement. Also, this time is the frame time recorded at the ground station, **not** the detector trigger time onboard the rocket payload, adding additional uncertainty of 1-2 ms. More precise timing could be calculated for higher level data if necessary, but precision of a few ms is expected to be sufficient for FOXSI data analysis. WSMR times recorded in the data structure will only have significant digits to the millisecond place.

For FOXSI-2 data, in the first round of processed data, there was a 36-second time offset in the `wsmr_time` tag. On investigation, it was found that a misinterpretation of "the milliseconds" value from the WSMR ground station recorded time was adding on 36 seconds. (This misinterpretation was in the FOXSI data processing routine; the value recorded by WSMR was correct.) After discovery of the problem, the bug was fixed, the data were reprocessed with the correct times, and all reference to the 36-second time offset were removed from the software.

Level 0 data file: `foxsi_level0_data.sav`

Structures in .sav file: `data_lvl0_D0`, `data_lvl0_D1`, `data_lvl0_D2`, `data_lvl0_D3`, `data_lvl0_D4`, `data_lvl0_D5`, `data_lvl0_D6`

Tags:

- Frame number
- Time (in seconds-of-day, from WSMR. See earlier notes on timing.)
- Frame time
- Detector number
- Trigger time
- Hit ASIC # [n-side, p-side]
- Hit strip # [n-side, p-side]
- Hit ADC value [n-side, p-side]
- Strip numbers with recorded data (all ASICs), 4x3 array
- 3-strip value (all ASICs), 4x3 array
- Common mode values (all ASICs) 4x1 array
- Channel masks (all ASICs) 4x1 array with each mask encoded in a 64-bit long
- Detector bias voltage (HV) in raw form (including status bit)
- Detector temperature, if we have it, from nearest applicable frame
- In flight: this flag is set if the event occurred while HV>190V (FOXSI-1).
- Altitude, in meters. Altitude values are repeated between 0.5 sec cadence, not interpolated.
- Error flag: set if any common mode value is out of range. However, no data is thrown away at this point!

2.2.3 Level 1 data

The intention of level 1 data is to include higher-level information but to exclude any intricate processing steps that could be expected to change later as we make refinements (i.e. detector response, payload pointing, etc). ASIC and strip numbers are replaced with 2D position information, given in both detector coordinates (pixels) and in payload coordinates (arcseconds), using a coordinate system similar to that of the GSE, but with the vertical reflection fixed. The detector coordinate system is arbitrary and wont necessarily match the zoom window on the GSE. For this coordinate system the p- coordinate is given first.

The "detector" coordinate system for CdTe is chosen to be the system which need only the "standard" rotation to get the payload coordinates. The relation between the ASICs and the detector coordinate system is display for each detector in the figure 1.

Coarse payload coordinates means that the design-specified geometric rotation of each detector is taken into account. A position in **fine** payload coordinates is subject to coregistration and so will be part of the next level.

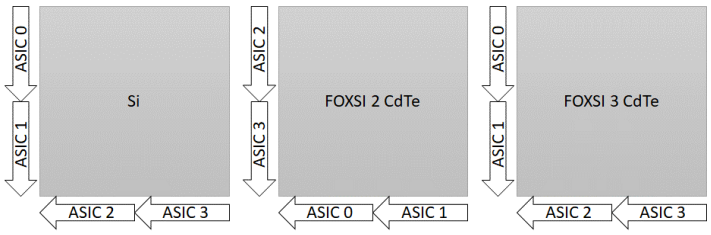

Figure 1: Orientation of the detector in coordinates in level 1 data. Note that in this context, for Si detector, we need to do a reflection of the Y axis and a rotation to get the payload coordinates.

Frame and trigger times are used to calculate a livetime value (for this frame only; doesnt include multiple frames). Within each event, strip values are classified as (a) hit (highest value for that side of the detector), (b) associated (values came from the hit ASIC), and (c) unrelated (values from the other ASIC).

IMPORTANT: As of FOXSI-2, Level 1 data is being truncated to not include events before the HV reaches 200V. For FOXSI-1, all events were carried through all levels of processing, so that preflight data could be used for troubleshooting and diagnostics. This is no longer necessary, so FOXSI-2 and FOXSI-3 data only includes HV=200V data for Level 1 and Level 2. FOXSI-1 data is being retained as before for backwards compatibility.

Level 1 data file: `foxsi_level1_data.sav`

Structures in .sav file: `data_lvl1_D0`, `data_lvl1_D1`, `data_lvl1_D2`, `data_lvl1_D3`, `data_lvl1_D4`, `data_lvl1_D5`, `data_lvl1_D6`

Tags:

- Frame number
- Time (in seconds-of-day, from WSMR. See earlier notes on timing.)
- Frame time
- Detector number
- Trigger time
- Livetime (for this frame only!! Doesnt include extra frames since last event yet.)
- Hit ASIC # [n-side, p-side] (only for files created after Dec 2018)
- Hit strip # [n-side, p-side] (only for files created after Dec 2018)
- Hit ADC values [n-side, p-side], no common-mode subtraction
- Common mode value for the hit ASIC (also use this CM for the associated strips)
- Hit ADC values [n-side, p-side], common-mode subtracted
- Hit 2D position in detector coordinates [p-side, n-side] [pixels]
- Hit 2D position in coarse payload coordinates [x, y] [arcsec]
- Associated ADC values, 3x3x2 array (includes hit!) (last dim is n- or p-side)
- Associated positions, in detector coordinates, 3x3x2 (includes hit!) (last dim is n- or p-side)
- Unrelated ADC values, 3x3x2 array (last dim is n- or p-side)
- Unrelated positions, in detector coordinates, 3x3x2 (last dim is n- or p-side)
- Unrelated common mode values [n-side, p-side]
- Channel masks (all ASICs) 4x1 array with each mask encoded in a 64-bit long.
  - A pixel map is not possible because of the large memory needed to store a 128x128 array for a structure with 3^5 elements.
- Detector bias voltage (HV) in volts
- Detector temperature, if we have it, from nearest applicable frame
- In flight: this flag is set if the event occurred when HV>190V.
- Altitude, in meters. Altitude values are interpolated between 0.5 sec cadence.

- Error flag, stored bitwise. So far 7 suspicious red flags have been identified; each of these gets a bit. By storing the values bitwise, more than one error can be flagged. The bits as of 2/19/2013 are:
  - Bit 0: Any CM value > 1023
  - Bit 1: All zero ADC data from one side (either n or p)
  - Bit 2: p-side CM is 0 (can't use value for spectroscopy)
  - Bit 3: Detector voltage possibly not settled (applies for ∼40 sec after HV ramp finishes and at end, when HV ramp down starts)
  - Bit 4: CM > highest ADC for that ASIC. (This mostly duplicates bit 0 error.)
  - Bit 5: Signal location is within 3 strips from detector edge.
  - Bit 6: Livetime value out of range ([1,2000] us)

  It is emphasized that an error flag doesnt necessarily mean the event should be thrown out (e.g. we may get good values before the detector current settles, and incomplete readouts may still have useful data in them) but these values should be used with caution. For the most conservative analysis, include only data with no flagged errors.

2.2.4 Level 2 data

Level 2 conversion includes most calibration steps. ADC values are replaced with energies (diagonal matrix elements only). SPARCS pointing information is included in order to translate coordinates from detector/payload into heliocentric.

Level 2 data file: `foxsi_level2_data.sav`

Structures in .sav file: `data_lvl2_D0`, `data_lvl2_D1`, `data_lvl2_D2`, `data_lvl2_D3`, `data_lvl2_D4`, `data_lvl2_D5`, `data_lvl2_D6`

Tags:
- Frame number
- Time (in seconds-of-day, from WSMR. See earlier note on timing.)
- Frame time
- Detector number
- Trigger time
- Livetime (for this frame only!! Doesnt include extra frames since last event yet.)
- Hit ASIC # [n-side, p-side] (only for files created after Dec 2018)
- Hit strip # [n-side, p-side] (only for files created after Dec 2018)
- Hit keV values [n-side, p-side], common-mode subtracted
- Hit 2D position in detector coordinates [p-side, n-side] [pixels]
- Hit 2D position in payload coordinates [x, y] [arcsec]
- Hit 2D position in heliocentric coordinates [x, y] [arcsec]
- Associated keV values, 3x3x2 array (includes hit!) (last dim is n- or p-side)
- Associated positions in payload coordinates, 3x3x2 (includes hit!) (last dim is n- or p-side)
- Associated positions in solar coordinates, 3x3x2 (includes hit!) (last dim is n- or p-side)
- Detector bias voltage (HV) in volts
- Detector temperature, if we have it, from nearest applicable frame
- In flight: this flag is set if the event occurred when HV>190V
- Altitude, in meters. Altitude values are interpolated between 0.5 sec cadence.
- SPARCS pitch: gives the payload coord origin on the Sun, horizontal component
- SPARCS yaw: gives the reverse of the payload coord origin on the Sun, vertical component
- Error flag, stored bitwise. So far 7 suspicious red flags have been identified; each of these gets a bit. By storing the values bitwise, more than one error can be flagged. The bits as of 2/19/2013 are:
  - Bit 0: Any CM value > 1023
  - Bit 1: All zero ADC data from one side (either n or p)
  - Bit 2: p-side CM is 0 (can't use value for spectroscopy)

  - Bit 3: Detector voltage possibly not settled (applies for ∼40 sec after HV ramp finishes and at end, when HV ramp down starts)
  - Bit 4: CM > highest ADC for that ASIC. (This mostly duplicates bit 0 error.)
  - Bit 5: Signal location is within 3 strips from detector edge.
  - Bit 6: Livetime value out of range ([1,2000] us)

  It is emphasized that an error flag doesnt necessarily mean the event should be thrown out (e.g. we may get good values before the detector current settles, and incomplete readouts may still have useful data in them) but these values should be used with caution. For the most conservative analysis, include only data with no flagged errors.
- Date of creation: date and time provided by OS when the structure is created (only for files created after Dec 2018)
- Calibration file name: name of the calibration file used when creating the structure (only for files created after Dec 2018)

2.2.5 Calibration

Calibration data include:

- Measurements of the optic modules effective areas
- Measurements of the detectors efficiency curves
- Measurements of the the positions of different peaks in the detector spectra, used for gain calibration in the software
- Theoretical attenuation lengths of different materials in the optic path
- Estimation of the effect of the collimators on the effective area

The current calibration is not the definitive FOXSI calibration, and the user must be aware of the following:

- The spectral resolution of the hard X-ray detectors (both Si and CdTe) is worse on the anode side (ASICs 0 and 1), and therefore the uncertainty on the measured energies is larger for this side.
- The gain calibration of the anode side (ASICs 0 and 1) for the FOXSI-3 CdTe detectors is preliminary and currently has larger uncertainties than the calibration of the cathode side.
- The effective area reduction due to the collimator is approximated in the current calibration: the effective area is multiplied by a factor 0.673.

In general, it is recommended to contact the FOXSI team before reaching scientific conclusions using the FOXSI data, to check the status of the calibration of the data and how the existing uncertainties affects the data analysis and interpretation.

2.3 PhoEnIX data description

The PhoEnIX detector is a photon-counting soft X-ray detector, which provides energy information for every photon collected. These data can be combined in different ways to produce lightcurves, images and spectra at different locations, as illustrated in figure 2. More details about the PhoEnIX observations during the FOXSI-3 flights can be found in [Musset et al., 2019].

2.3.1 Quicklook images

The data currently publicly available are full-Sun images in different energy ranges, with coarse position information and preliminary detector calibration: these data have to be treated as quicklook data. These images were created by integrating the photon flux during the whole FOXSI-3 flight, during the different targets. Because of the PhoEnIX large field of view, integrating over the full flight leads to a a full-Sun image.

The PhoEnIX quicklook images data is the number of photons detected in each detector pixel during the flight. Images were produced for the following energy bands: 0-0.25, 0.25-0.50, 0.50-0.75, 0.75-1.00, 1.00-1.25, 1.25-1.50, 1.50-1.75, 1.75-2.00, and 2.00-2.25 keV.

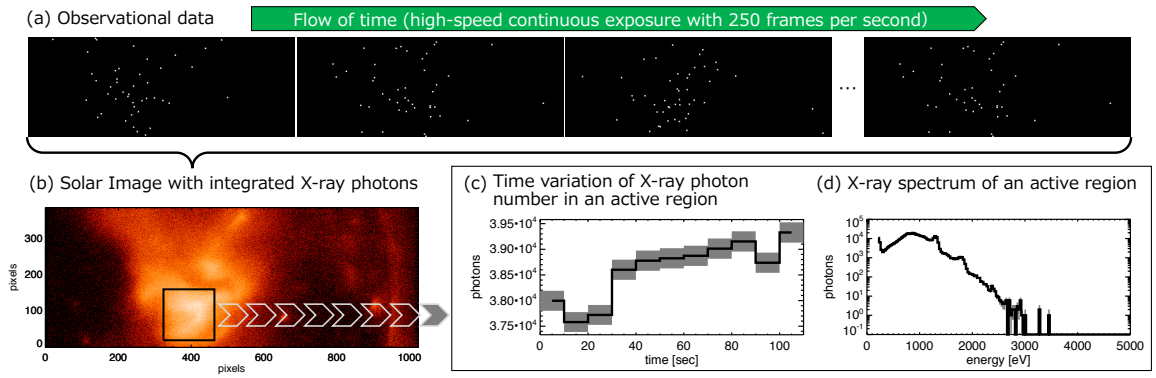

Figure 2: Data reduction for the PhoEnIX observations (FOXSI-3). Observations include position, timing and energy information for every single photon reaching the detector. These informations can be combined to create lightcurves and spectra of different pixels or group of pixels (here, they are shown for the core region of an observed active region), and images at different times and energy bands.

3 First example

By now, you should have a `foxsi-science` directory with several sub-directories obtained from GitHub, as well as three data sub-directories that you populated yourself with either .sav or FITS files.

To start using the FOXSI software, start `SSWIDL`. You have to load the parameters for a given flight, for instance, FOXSI-2. This step will be different if you are using .sav files or FITS files.

If you are using .sav files, type:

```
IDL> foxsi, 2014
```

If you are using FITS files, type:

```
IDL> foxsi, 2014, /read_fits
```

If the FITS file is not in the expected directory (`foxsidb/data_2014` for instance for FOXSI-2 data), you can provide the correct path to the data file: If you are using FITS files, type:

```
IDL> foxsi, 2014, /read_fits, fits_dir='your/path/to/fits/file'
```

This will work with 2012 for FOXSI-1, 2014 for FOXSI-2 and 2018 for FOXSI-3.

Now you are ready to go! The parameters and the hard X-ray data corresponding to the chosen FOXSI flight are loaded. You can continue your analysis with images, spectra, and lightcurves.

4 General useful routines

4.1 Get target data

The routine `get_target_data.pro` will return data for a given target for all detectors. The times used for each target are the times after any pointing adjustments are made. In the future, this can (and should) be upgraded to return data for pre-adjustments too. Heres the relevant info on using this handy routine:

```
; This procedure grabs Level 2 data structures for a given target during a FOXSI flight.
; For FOXSI-1 there are 6 targets to choose from, and for FOXSI-2 there are 5.  For
; FOXSI-2, this includes the data for each target after any pointing adjustments were
; done.  The "shutter in" time is not included in Target 4 (quiet Sun target).
;
; FOXSI-1 Target 1: AR1
```

```
; Target 2: AR2
; Target 3: Quiet Sun
; Target 4: Flare
; Target 5: Correction attempt (~10 sec)
; Target 6: Back to the flare
;
; FOXSI-2 Target 1: AR1, after all repointings
; Target 2: AR2, after all repointings
; Target 3: AR3, after all repointings
; Target 4: Quiet Sun, prior to shutter
; Target 5: AR1, with shutter in
;
; Inputs:
;
; TARGET Options are 1-5 or 1-6; see index above.
;
; Outputs:
;
; D0, D1, etc... Data structures for the given target.
```

Many keywords are available (see the source code), but here are the ones youre likely to want:

```
; Keywords:
;
; YEAR 2012, 2014, or 2018, default 2014
; EBAND Restrict to energy range (2-element array).
; GOOD Only return events with error_flag eq 0
; D0_IN, etc Input data structures.  If not input, these will be
;             restored from the flight data file.  (Inputting them
;      saves time for FOXSI-1 data.)
; DELTA_T Output variable giving the time on that target.

; Example: To select data from the last pointing for FOXSI-2, do:
;
; get_target_data, 6, d0, d1, d2, d3, d4, d5, d6
;
; or, for a selected energy range, and only good events:
;
; get_target_data, 6, d0, d1, d2, d3, d4, d5, d6 /good, eband=[5.,10.]
```

4.2 Calculate the count rate

The script below has been used to measure the FOXSI count rates in any target, time or energy.

```
;
; Make a map to get an overall sense of what we're looking at.
; Example is 1st target after 2 pointing adjustments (target 1, position 2)
;

loadct, 7
reverse_ct
trange=[t1_pos2_start,t1_pos2_end]
m6 = foxsi_image_map( data_lvl2_d6, cen1_pos2, erange=[5.,10.], $
  trange=trange, thr_n=4., /xycorr )
plot_map, m6, /limb, lcol=255, col=255, charsi=1.2, title=m6.id
```

```
;
; Select data for one detector in the desired energy band,
; time range, and location.
;

times = [t1_pos2_start,t1_pos2_end]
area_center = [-200.,-200.]
;area_center = [50.,-200.] ; use this one if you want a flare example for debugging.
area_radius = [100.]
energy_band = [5.,10.]

data = data_lvl2_d6

; area selection, including application of the XY offset (/xycorr)
data = area_cut( data, area_center, area_radius, /xycorr )

; time and energy selection
data = time_cut( data, times[0], times[1], energy=energy_band )

;
; Select only the cleanest events, and keep track of how many we're throwing away.
; The thrown-out events will go into an additional deadtime.
;

n_all = n_elements( data )
data  = data[ where( data.error_flag eq 0 ) ]
n_good = n_elements( data )
live = float(n_good) / n_all
; Later, we should calculate a a more accurate livetime
; accounting for readout time.

;
; Some work to fill in:
; To do: Calculate # count per solar area per second per keV.
; To do: Calculate a statistical error on that value.
; To do: Divide by the instrument effective area to get the value in
;                   photons / sec / keV / cm2 / cm2
; To do: Repeat for several energies and locations.
;        Goal is to plot the rate vs radius
;        for a few energy bands.
; Later, we'll look at setting limits based on these numbers
;        using small-number statistics.

; Some code to help with the flux calculation:

area6 = get_foxsi_effarea( module_num=6, energy=energy_band )
area6_live = area6.eff_area_cm2*live ; correct for livetime
```

5 Lightcurves

The basic routine for generating lightcurves is `foxsi_lc.pro`. Pass to this routine a Level 2 data structure and tell it a time interval to integrate over (`DT`, default 10 seconds). The return variable is a structure containing arrays of times and count rates (in counts per second). The default energy range is `[4,15]` keV.

Examples for a FOXSI-2 lightcurve:

```
; Sample script to generate a FOXSI lightcurve

; Set parameters for 2014 flight (from foxsi-science directory)
foxsi,2014

dt = 5. ; time interval over which to integrate
lc0 = foxsi_lc( data_lvl2_d0, year=2014, dt=dt)
lc1 = foxsi_lc( data_lvl2_d1, year=2014, dt=dt)
lc2 = foxsi_lc( data_lvl2_d2, year=2014, dt=dt)
lc3 = foxsi_lc( data_lvl2_d3, year=2014, dt=dt)
lc4 = foxsi_lc( data_lvl2_d4, year=2014, dt=dt)
lc5 = foxsi_lc( data_lvl2_d5, year=2014, dt=dt)
lc6 = foxsi_lc( data_lvl2_d6, year=2014, dt=dt)

lc0.time -= 36 ; this corrects for the 36-sec offset in WSMR data.
lc1.time -= 36 ; source of this offset is still being investigated.
lc2.time -= 36
lc3.time -= 36
lc4.time -= 36
lc5.time -= 36
lc6.time -= 36

; Plot the lightcurves.
loadct,5
hsi_linecolors
utplot,  lc6.time, lc6.persec, /nodata, yr=[0,100], $
charsi=1.2, charth=2, xth=5, yth=5, ytit='Counts s!U-1!N', title='FOXSI 2014'
outplot, lc0.time, lc0.persec, psym=10, col=6, th=4
outplot, lc1.time, lc1.persec, psym=10, col=7, th=4
;outplot, lc2.time, lc2.persec, psym=10, col=8, th=4
outplot, lc3.time, lc3.persec, psym=10, col=9, th=4
outplot, lc4.time, lc4.persec, psym=10, col=10, th=4
outplot, lc5.time, lc5.persec, psym=10, col=12, th=4
outplot, lc6.time, lc6.persec, psym=10, col=2, th=4
al_legend, ['D0','D1','D3','D4','D5','D6'], /right, /top, box=0, $
textcol=[6,7,9,10,12,2]
```

6 Imaging

The basic routine for generating FOXSI image arrays is `foxsi_det_image.pro`. If a `plot_map` is preferred, the routine `foxsi_image_map.pro` can be used. The user needs to input the desired times and the target center, but these are easily set using the parameters in `foxsi-setup-script-201x`. The energy range n-side threshold can be chosen if desired. A diagram showing how the procedure `foxsi_image_map.pro` operates is available in figure 3 in appendix B.

6.1 Basic imaging: creating a FOXSI map

The following short example produces a FOXSI-2 image of the first target (after 2 pointing adjustments) from detector 6. Note that this example applies a rough pointing offset that was obtained by comparing detector images with AIA 94A. (This offset should probably be improved.) That offset is stored in the setup script.

```
; Set auto parameters for 2014 flight.
foxsi,2014

; Choose the time range and location.
trange = [t1_pos2_start,t1_pos2_end]  ; values defined in setup script
center = cen1_pos2 ; values defined in setup script

; Make the image.
map6 = foxsi_image_map( data_lvl2_d6, center, $
trange=[t1_pos2_start, t1_pos2_end], /xycorr )

plot_map, map6, /log, /cbar
```

6.2 Basic imaging: step-by-step

The following example uses `foxsi_image_det.pro` directly, without the `foxsi_image_map.pro` wrapper, to demonstrate intermediate steps.

```
; Set auto parameters for 2014 flight.
foxsi,2014

; Choose the time range and location.
trange = [t1_pos2_start, t1_pos2_end] ; time range
xc = cen1_pos2[0] ; coords for Target 1
yc = cen1_pos2[1]

time=anytim('2014-12-11') + average(trange)+tlaunch
time = anytim( time, /yo )

; Basic image production
image6 = foxsi_image_det( data_lvl2_d6, year=2014, trange=trange, $
erange=[4.,15.], thr_n=4. )
map6 = make_map( image6, dx=7.78, dy=7.78, xcen=xc, ycen=yc, $
time=time, id='D6' )

; Apply a coarse offset gleaned from comparing images with AIA.
map6 = shift_map( map6, shift6[0], shift6[1] )

; Rotate the image based on the rotation angle for that specific detector.
map6 = rot_map( map6, rot6 )
map6.roll_angle = 0
map6.roll_center = 0

loadct, 5
plot_map, map6
plot_map, map6, /log
```

6.3 Deconvolved images

TBD

7 Spectroscopy

The procedure `foxsi_ospex.pro` can be used to perform spectral fitting of FOXSI data using an optically thin thermal plasma model in OSPEX. The procedure creates two files: a FITS file with the OSPEX fit results and an IDL .sav file with the livetime and good event ratio written. The code is best used for FOXSI-2 data and needs adjustments before using for analysis of FOXSI-1 and FOXSI-3 data. An example of the use of `foxsi_ospex.pro` is provided below.

```
; Set parameters for 2014 flight
foxsi, 2014

; Choose time range and location
trange = [t1_pos2_start, t1_pos2_end]
center = flare1         ;center of source
offaxis = 0.4           ;off-axis source position
radius = 100.           ;radius of source region
cen_map = cen1_pos2     ;target center

; Choose energy range and bin size for spectral analysis
erange = [5.,8.]
bin = 0.5

; Choose module for spectral analysis
det = 5
type = 'si'     ;type of detector
fwhm = 0.5      ;energy resolution
let_file = 'efficiency_averaged.sav'    ;LET efficiency file

; run spectral fitting procedure
foxsi_ospex, 'example_spectral_analysis', det=det, bin=bin, erange = erange, $
    trange = trange, center = center, radius = radius, offaxis = offaxis, $
    cen_map = cen_map, fwhm = fwhm, let_file = let_file, type = type

; read fit results
fit_results = spex_read_fit_results('example_spectral_analysis.fits')
```

8 Creating different levels of data

While users are encouraged to use the level 2 data files that are available at ....., experimented users can also use the lower level data and process them to get to a new set of level 2 data.

8.1 Obtaining level 0 data

Level 0 data can be obtained from 1/ FOXSI GSE files (data files taken with the FOXSI GSE) or 2/ Files from WSMR. The procedure to obtain level 0 data from those files are different and described here.

8.1.1 Level 0 from FOXSI GSE files

Starting with the data file recorded by the WSMR ground station, use the procedure `wsmr_data_to_level0`. An example is shown below:

```
; Create Level 0 data
filename = 'data_2012/36.255_TM1_Flight_2012-11-02.log'
data_lvl0_D0 = wsmr_data_to_level0( filename, det=0, year=2012 )
data_lvl0_D1 = wsmr_data_to_level0( filename, det=1, year=2012 )
data_lvl0_D2 = wsmr_data_to_level0( filename, det=2, year=2012 )
data_lvl0_D3 = wsmr_data_to_level0( filename, det=3, year=2012 )
data_lvl0_D4 = wsmr_data_to_level0( filename, det=4, year=2012 )
data_lvl0_D5 = wsmr_data_to_level0( filename, det=5, year=2012 )
data_lvl0_D6 = wsmr_data_to_level0( filename, det=6, year=2012 )
save, data_lvl0_D0, data_lvl0_D1, data_lvl0_D2, data_lvl0_D3, $
data_lvl0_D4, data_lvl0_D5, data_lvl0_d6, $
file = 'data_2012/foxsi_level0_data.sav'
```

8.1.2 Level 0 from WSMR files

A FOXSI data file recorded by the GSE (i.e. not by WSMR) can be processed in the same pipeline by substituting a different routine for the first one. Use the appropriate routine for either FORMATTER (500 frames/sec from formatter in a binary file) or USB (text file containing info from all strips) style data: `formatter_data_to_level0` OR `usb_data_to_level0`

8.2 Obtaining level 1 data

From a level 0 file, a level 1 file can be obtained using the procedure `foxsi_level0_to_level1`. An example is shown below:

```
; Create Level 1 data
filename = 'data_2012/foxsi_level0_data.sav'
data_lvl1_D0 = foxsi_level0_to_level1( filename, det=0, ground=0 )
data_lvl1_D1 = foxsi_level0_to_level1( filename, det=1, ground=0 )
data_lvl1_D2 = foxsi_level0_to_level1( filename, det=2, ground=0 )
data_lvl1_D3 = foxsi_level0_to_level1( filename, det=3, ground=0 )
data_lvl1_D4 = foxsi_level0_to_level1( filename, det=4, ground=0 )
data_lvl1_D5 = foxsi_level0_to_level1( filename, det=5, ground=0 )
data_lvl1_D6 = foxsi_level0_to_level1( filename, det=6, ground=0 )
save, data_lvl1_D0, data_lvl1_D1, data_lvl1_D2, data_lvl1_D3, data_lvl1_D4, $
data_lvl1_D5, data_lvl1_d6, $
file = 'data_2012/foxsi_level1_data.sav'
```

8.3 Obtaining level 2 data

To create the level 2 data from the level 1 data, it is necessary to specify the calibration files to be used for the detector gain calibration, when calling the procedure `foxsi_level1_to_level2`. An example is shown below:

```
; Create Level 2 data
; Here, you need to know the specific detectors flown.
file0 = 'data_2012/foxsi_level0_data.sav'
file1 = 'data_2012/foxsi_level1_data.sav'
cal0 = 'calibration_data/peaks_det108.sav'
cal1 = 'calibration_data/peaks_det109.sav'
cal2 = 'calibration_data/peaks_det102.sav'
cal3 = 'calibration_data/peaks_det103.sav'
cal4 = 'calibration_data/peaks_det104.sav'
cal5 = 'calibration_data/peaks_det105.sav'
cal6 = 'calibration_data/peaks_det106.sav'
```

```
data_lvl2_D0 = foxsi_level1_to_level2(file0, file1, det=0, calib=cal0 )
data_lvl2_D1 = foxsi_level1_to_level2(file0, file1, det=1, calib=cal1 )
data_lvl2_D2 = foxsi_level1_to_level2(file0, file1, det=2, calib=cal2 )
data_lvl2_D3 = foxsi_level1_to_level2(file0, file1, det=3, calib=cal3 )
data_lvl2_D4 = foxsi_level1_to_level2(file0, file1, det=4, calib=cal4 )
data_lvl2_D5 = foxsi_level1_to_level2(file0, file1, det=5, calib=cal5 )
data_lvl2_D6 = foxsi_level1_to_level2(file0, file1, det=6, calib=cal6 )
save, data_lvl2_D0, data_lvl2_D1, data_lvl2_D2, data_lvl2_D3, $
data_lvl2_D4, data_lvl2_D5, data_lvl2_d6, $
file = 'data_2012/foxsi_level2_data.sav'
```

A Housekeeping data

Housekeeping data is stored separately from the event data because it arrives regularly either every frame (voltages) or every fourth frame (temperatures), instead of sporadically like the detector events do. Each entry in the housekeeping data structure corresponds to one data frame. Housekeeping values that show up every fourth frame are allowed to "leak" into neighboring frames to provide a value for every frame. This is also done for the altitude data (cadence 0.5 sec) to match the formatter frame cadence (0.002 sec).

A.1 Level 0

All data are raw values, meaning that thermistor A/D values have not been decoded into real temperatures, voltages, etc.

Level 0 data file: `foxsi_level0_hskp_data.sav`

Structures in .sav file: `hskp_data`

Tags:
- 1: Frame counter
- 2: Time (in seconds-of-day, from WSMR. See earlier note on timing.)
- 3: Formatter frame time
- 4: High voltage value (verbatim from the frame including status bits)
- 5-8: All voltages (raw values) - four values
- 9-20: All thermistors on electronics package (raw values)
- 21-24: Additional formatter housekeeping words (command count, command1, command2, formatter status)
- 25-31: Status word for each detector (Note: probably no meaningful data here.)
- 32: In-flight flag: '1' for frames after launch
- 33: Error flag: '1' if obvious error is detected (in this case if frame counter does not increment by 1)
- 34: Altitude, in meters. Altitude values are repeated between 0.5 sec cadence, not interpolated.

A.2 Level 1

The level 1 housekeeping data structure is identical to the level 0 structure except that all values are converted into volts and degrees.

B Imaging process: differences in the code between Si and CdTe detectors

As mentioned in section ..., different detectors have different geometries. As illustrated in figure 3, the imaging routine creating a map object in the software (`foxsi_image_map`) is based on the detector coordinates of a

FOXSI structure (it could be either a level2 or a level1 data file). Therefore, differences in geometry between detectors have to be taken care of at different points in the software.

The detector coordinates are calculated in different ways for the Si and CdTe detectors in the procedure `foxsi_level0_to_level1`, so that the following transformations to payload and solar coordinates will be detector independent in the code. The transformation into payload coordinates requires to call the function `get_payload_coordinates`, which essentially apply the correct pixel size to the coordinates, which is detector dependent.

When producing a FOXSI map using `foxsi_image_map`, the detector coordinates are considered to create an image via the function `foxsi_image_det`. Then a map is created using the pixel size appropriate to the detector considered. The remaining step, which consist in feeding the pointing information and the detector orientation, is independent of the type of detector.

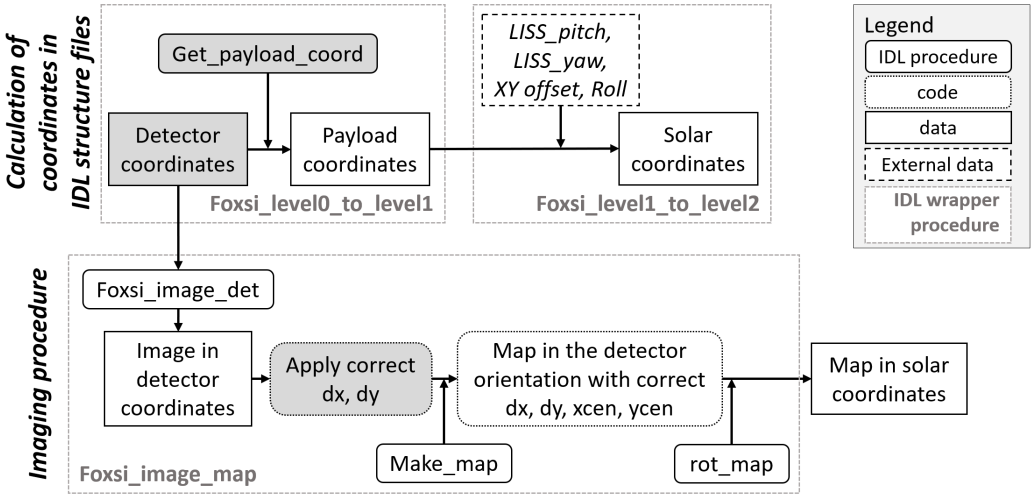

Figure 3: Diagram representing different operations on data and image coordinates in the FOXSI Science software. Grey boxes indicate actions or procedures were there will be a distinction between Si, FOXSI-2 CdTe and FOXSI-3 CdTe detectors.

References

[Glesener et al., 2016] Glesener, L., Krucker, S., Christe, S., Ishikawa, S.-n., Buitrago-Casas, J. C., Ramsey, B., Gubarev, M., Takahashi, T., Watanabe, S., Takeda, S., Courtade, S., Turin, P., McBride, S., Shourt, V., Hoberman, J., Foster, N., and Vievering, J. (2016). The FOXSI solar sounding rocket campaigns. In *Proceedings of the SPIE*, volume 9905 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 99050E.

[Krucker et al., 2013] Krucker, S., Christe, S., Glesener, L., Ishikawa, S., Ramsey, B., Gubarev, M., Saito, S., Takahashi, T., Watanabe, S., Tajima, H., Tanaka, T., Turin, P., Glaser, D., Fermin, J., and Lin, R. P. (2013). The focusing optics x-ray solar imager (FOXSI): instrument and first flight. In *Proceedings of the SPIE*, volume 8862 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 88620R.

[Musset et al., 2019] Musset, S., Buitrago-Casas, J. C., Glesener, L., Bongiorno, S., Courtade, S., Athiray, P. S., Vievering, J., nosuke Ishikawa, S., Narukage, N., Furukawa, K., Ryan, D., Dalton, G., Turin, Z., Davis, L., Takahashi, T., Watanabe, S., Mitsuishi, I., Hagino, K., Kawate, T., Turin, P., Christe, S., Ramsey, B., and Krucker, S. (2019). Ghost-ray reduction and early results from the third FOXSI

sounding rocket flight. In Siegmund, O. H., editor, *UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XXI*, volume 11118, pages 321 – 336. International Society for Optics and Photonics, SPIE.