**FOXSI Analysis Guide**

**2014 Jan 19**

Last updates:

2015 Feb 2: Fixed/removed 36-second time offset in WSMR time.

\*\*Reprocessed data; please re-download.\*\*

2015 Jan 19: Added new routine foxsi\_image\_map.pro LG

2015 Jan 8: Updated pointing info and added doc for get\_target\_data LG

2015 Jan 6: Created document LG

Please send questions and bug reports to glesener@ssl.berkeley.edu

1. **Getting FOXSI software**
   1. FOXSI software is written in IDL and requires SSWIDL to be installed.
   2. FOXSI analysis software is kept in a GitHub repository at <https://github.com/foxsi/foxsi-science>. You can download a snapshot of the software as a ZIP file from there. Alternatively, you can sign up for a GitHub account and use **git** to keep your software tree up to date. This second choice takes some studying but is useful if you want to contribute your additions and changes.
   3. 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 calibration\_data. These directories are **not** included in the GitHub repository because they contain large data files. Their contents are ignored by GitHub via text in the .gitignore file. You’ll need to populate these directories with data; see the next section.
2. **Getting FOXSI data**
   1. Download the following files from <ftp://apollo.ssl.berkeley.edu/pub/foxsi>. Choose either the data\_2012 or data\_2014 directory (for the first or second flight, respectively). Put these files in a data\_2012 or data\_2014 folder in your foxsi-science directory.
      1. foxsi\_level1\_data.sav
      2. foxsi\_level2\_data.sav
   2. On the same FTP site, download everything in the calibration\_data folder. These contain the necessary components to build the instrument response using whichever pieces (optics, detector, blanketing) are desired. Put these files in a calibration\_data folder in your foxsi-science directory.
3. **Setting up**

By now, you should have a foxsi-science directory with several subdirectories obtained from GitHub, as well as three data subdirectories that you’ve populated yourself.

To start using the FOXSI software, start SSWIDL. Run one of the following setup scripts at the prompt. Run only one script, not both, since the script sets flight-specific parameters like dates, times, and pointing positions.

IDL> @foxsi-setup-script-2012

or

IDL> @foxsi-setup-script-2014

Now you’re ready to go! You can skip the next step (“Processing FOXSI data into higher levels”) and go right to the fun parts, like images, spectra, and lightcurves.

1. **Processing FOXSI data into higher levels**

***For most purposes, you don’t need to do this. Just download the FOXSI Level 1 and Level 2 pre-processed files.***

Should you need to reprocess the FOXSI data for some reason, use the routines in the foxsi-science/proc/ directory. Run these in order, starting with the data file recorded by the WSMR ground station.

Procedure order:

* wsmr\_data\_to\_level0
* foxsi\_level0\_to\_level1
* foxsi\_level1\_to\_level2

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
* foxsi\_level0\_to\_level1
* foxsi\_level1\_to\_level2

The above choices allow you to process any FOXSI data file into Level 2 data, whether from flight or calibration, no matter whether it was recorded through the formatter interface, USB interface, or ground station. However, note that there is a different set of software that is used for the FOXSI detector calibration, as the flight analysis software is not optimized to work with a large number of events (and was also written much later).

Example of normal processing starting from WSMR Ground Station file:

; 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'

; 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'

; 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'

1. **Data structures**

Full documentation for the FOXSI data structure is given in a separate document (FOXSI Data Description).

**Info for FOXSI-2 data:**

There is an important difference between the FOXSI-1 and FOXSI-2 data structures. 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** Level 1 and Level 2 data. As a result, the IDL save files are much smaller. These extra events are still included in the FOXSI-1 higher level data in order that Lindsay’s older codes will still work for that data.

For FOXSI-2, the setup file includes information from the 36.295 SPARCS report. As of this writing (Jan 8 2015), this report is preliminary and is not yet officially approved. Some info from this report is included in the Appendix.

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.

The position and timing information are as follows. Targets are labeled 1-5, with 5 a repeat of Target 1, corrected. Pointing adjustments within targets are labeled Position 0, 1, or 2. (pos0 would be before any corrections; pos1 and pos2 are after 1 or 2 corrections, respectively.)

; Timing info from Jesus's preliminary report.

tlaunch = 69060

t1\_pos0\_start = 102.0

t1\_pos0\_end = 134.3

t1\_pos1\_start = 138.0

t1\_pos1\_end = 162.6

t1\_pos2\_start = 166.5

t1\_pos2\_end = 205.0

t2\_pos0\_start = 209.0

t2\_pos0\_end = 219.6

t2\_pos1\_start = 224.0

t2\_pos1\_end = 276.7

t3\_pos0\_start = 280.6

t3\_pos0\_end = 307.2

t3\_pos1\_start = 311.0

t3\_pos1\_end = 330.1

t3\_pos2\_start = 334.0

t3\_pos2\_end = 369.2

t4\_start = 373.5

t\_shtr\_start = 438. ; Conservative time of attenuator insertion

t\_shtr\_end = 442. ; Conservative time for microphonics to die down

t4\_end = 466.2

t5\_start = 470.5

t5\_end = 503.2

date=anytim('2014-dec-11')

t0 = '11-Dec-2014 19:11:00.000'

;

; Positional information

;

; Our crude alignment offset determined in-flight

offset\_xy = [360.,-180.]

; SPARCS pointing positions, from Jesus

cen1\_pos0 = [ -1. ,-251. ] + offset\_xy ; 32 sec

cen1\_pos1 = [-361. ,-251. ] + offset\_xy ; 25 sec

cen1\_pos2 = [-361. , -71. ] + offset\_xy ; 38 sec

cen2\_pos0 = [-361. ,-101. ] + offset\_xy ; 11 sec

cen2\_pos1 = [-750. ,-101. ] + offset\_xy ; 53 sec

cen3\_pos0 = [ 850.5,-251.5] + offset\_xy ; 26 sec

cen3\_pos1 = [ 490. ,-251.5] + offset\_xy ; 19 sec

cen3\_pos2 = [ 490. , -71. ] + offset\_xy ; 35 sec

cen4 = [-160. , 930. ] + offset\_xy ; 92 sec

cen5 = [-360. , -71. ] + offset\_xy ; 36 sec

; Additional shifts for each detector, from comparison with AIA

; THIS SHOULD BE REDONE USING RHESSI!

; This is an example; values in the setup file may be different.

shift6 = [20.,40.] ; offset for D6, eyeballed.

shift0 = [ 34.9, 57.1 ] ; other shifts from comparing centroids

shift1 = [ 9.3, 59.5 ] ; with that of D6.

shift4 = [ 13.3, 18.0 ]

shift5 = [ 42.6, 44.0 ]

As an example, to isolate events from the first target after all pointing adjustments, you would take all events between t1\_pos2\_start and t1\_pos2\_end and use the cen1\_pos2 as the center of your map. Several of the routines are designed to automatically give you data for certain targets, so you usually won’t have to do a lot of minute work with times.

1. **Producing data products**

This is the fun part! Just remember to run @foxsi-setup-script-2014 (or ditto for 2012) before trying any of these.

* 1. **General useful routines**

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. Here’s the relevant info on using this handy routine:

; For each flight, there are 6 targets to choose from:

;

; 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: Quiet Sun with shutter in

; Target 6: AR1, with shutter in

;

; Inputs:

;

; TARGET Options are 1-6 for either flight; 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 you’re likely to want:

; Keywords:

;

; YEAR 2012 or 2014, 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.]

* 1. **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.

Example for a FOXSI-2 lightcurve:

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

lc2.time -= 36

lc3.time -= 36

lc4.time -= 36

lc5.time -= 36

lc6.time -= 36

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]

* 1. **Spectra**

Coming soon…

* 1. **Images**
     1. **Basic images (UPDATED 1/19/2015)**

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.

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.

; Example:

@foxsi-setup-script-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

map6 = foxsi\_image\_map( data\_lvl2\_d6, center, $

trange=[t1\_pos2\_start, t1\_pos2\_end], /xycorr )

plot\_map, map6, /log, /cbar

The following example uses foxsi\_image\_det.pro directly, without the foxsi\_image\_map.pro wrapper, to demonstrate intermediate steps.

@foxsi-setup-script-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

* + 1. Deconvolved images

Coming soon…

Appendix 1: FOXSI-2 SPARCS pointing information from Jesus Martínez

SPARCS Timeline/Tank Pressure

## Event Time (Sec) Tank Press (PSI) Comments

**SPARCS enable 66.0 4939 Enable SPARCS control jets**

**MASS sensor 76.0 4094 Pointing @ ~35º from Sun center**

**Mode sensor 79.5 3765 Pointing @ ~10º from Sun center**

**Intermediate mode 85.3 3234 SPARCS to LISS sensor**

**Fine mode 93.0 2763 High gain & low pressure**

**RLG on / enable 98.6 2793 Turn on / enable RLG via CUS**

**Fine mode stable 102.0 2793 Stable @ Target 1, Roll angle ~266.8°**

**Target #2 received 134.3 2711 Rel Pitch -360” cmd**

**Stable on Target #2 138.0 2696 Stable after relative command**

**Target #3 received 162.6 2584 Rel Yaw -180” cmd**

**Stable on Target #3 166.5 2561 Stable after relative command**

**Target #4 received 205.0 2382 Abs Yaw +100” cmd**

**Stable on Target #4 209.0 2359 Stable after absolute command**

**Target #5 received 219.6 2307 Abs Pitch -750” cmd**

**Stable on Target #5 224.0 2285 Stable after absolute command**

**Target #6 received 276.7 2038 Abs Pitch +850”, Abs Yaw +250”**

**Stable on Target #6 280.6 2015 Stable after absolute commands**

**Target #7 received 307.2 1903 Rel Pitch -360” cmd**

**Stable on Target #7 311.0 1881 Stable after relative command**

**Target #8 received 330.1 1799 Rel Yaw -180” cmd**

**Stable on Target #8 334.0 1784 Stable after relative command**

**Target #9 received 369.2 1642 Rel Pitch -650” cmd**

**Target #10 received 369.5 1642 Rel Yaw -1000” cmd**

**Stable on Target #9 373.2 1612 Stable after relative Pitch command**

**Stable on Target #10 373.5 1612 Stable after relative Yaw command**

**Target #11received 466.2 1260 Rel Pitch -200” cmd**

**Target #12 received 466.5 1253 Rel Yaw 1000” cmd**

**Stable on Target #11 470.0 1238 Stable after relative Pitch command**

**Stable on Target #12 470.5 1238 Stable after relative Yaw command**

**RLG off/disable 503.2 1118 Turn off / disable RLG via CUS**

**Shutter Door Closing 505.6 1111 Shutter door obscuring LISS**

**SPARCS to Coarse mode 506.8 1066 Mode sensor obscured by door**

**Spin up starts 510.0 1044 Start spinning the payload**

**Spin up stops 525.0 625 Payload spun to ~1.0 RPS**

**Vent starts 526.0 588 Residual gas venting begins**

**Loss of magnetometer 560.8 139 Top hat separation on reentry**

**Vent Stops 583.7 42 Venting stops, pressure threshold**

**SPARCS Disable 626.1 57 SPARCS pneumatics is disabled**

##### **Targets**

#### Target #1 (Absolute P-Y-R) Acquired Target #1

#### Pitch = 0.0 Arc-Sec Pitch = - 1.0 Arc-Sec

#### Yaw = 250.0 Arc-Sec Yaw = +251.0 Arc-Sec

#### Roll = + 76.0 º Roll = + 76.4 º

#### Target #2 (Relative Pitch) Acquired Target #2

#### Pitch = - 360 – 1 = -361 Arc-Sec Pitch = -361.0 Arc-Sec

#### Target #3 (Relative Yaw) Acquired Target #3

#### Yaw = -180 + 251 = +71 Arc-Sec Yaw = +71 Arc-Sec

#### Target #4 (Absolute Yaw) Acquired Target #4

#### Yaw = 100.0 Arc-Sec Yaw = +101.0 Arc-Sec

#### Target #5 (Absolute Pitch) Acquired Target #5

#### Pitch = -750 Arc-Sec Pitch = -750.0 Arc-Sec

#### Target #6 (Absolute P-Y) Acquired Target #6

#### Pitch = +850.0 Arc-Sec Pitch = +850.5 Arc-Sec

#### Yaw = +250.0 Arc-Sec Yaw = +251.5 Arc-Sec

#### Target #7 (Relative Pitch) Acquired Target #7

#### Pitch = - 360 + 850 = +490 Arc-Sec Pitch = +490 Arc-Sec

#### Target #8 (Relative Yaw) Acquired Target #8

#### Yaw = -180 + 251.5 = 71.5 Arc-Sec Yaw = +71 Arc-Sec

#### Target #9 (Relative Pitch) Acquired Target #9

#### Pitch = - 650 + 490 = -160 Arc-Sec Pitch = -160 Arc-Sec

#### Target #10 (Relative Yaw) Acquired Target #10

#### Yaw = -1000 + 71 = -929 Arc-Sec Yaw = -930 Arc-Sec

#### Target #11 (Relative Pitch) Acquired Target #11

#### Pitch = - 200 - 160 = -360 Arc-Sec Pitch = -360 Arc-Sec

#### Target #12 (Relative Yaw) Acquired Target #12

#### Yaw = 1000 - 929 = +71Arc-Sec Yaw = +71 Arc-Sec

