5.6.2.1.5 **grat_mon** **mta**

Grat_mon

Description: This tool performs the basic monitoring functions on gratings data (L1.5). The goal of the tool is to cover HRMA monitoring tasks and SI source monitor laid out in SE03: 5.6.2.1.5.2, 5.6.2.1.5.4, 5.6.2.1.5.5 and 5.6.2.1.5.7

Open Issues: We can chose to run this procedure more or less as is, or we can choose to apply peicemeal to upper level data products. The OBS_ANAL procedure is designed to handle L1.5 files, includeing multiple sources and Masks. This code is something we nned now. So we will use it as is, in future we should run more off L1.5 and L2 products. The data source and more notes have been written by Dan Dewey, they were last updated April 1, 1999 and can be found at: http://space.mit.edu/HETG/flight/obs.

Inputs: The inputs section summarizes how obs_anal was called, what the input file is, which keywords were set, etc. Note that no values are read from the FITS header(s). For example, GRATING and DETECTOR (if different from the defaults HETG, ACIS-S) must be supplied by the user. The goal is to minimize dependancies on parameters which may not exist or be incorrect. (To-do: The code has not been made robust to GRATING='NONE' or to no source in the field.)

- Level 1.5 grating files (L1 files work too).
- Level 1 event files

Outputs:

- Fits file containing selected data.
- Html file with graphics. (see algorithm)
- Selected data for the database:

Control Parameters:

- obsid, tstart, tstop, grating.
- filtering on all HEADER keywords.

Intent: It is the intent that this code use for its bulk the source provided as obs_anal.pro (thanks dd and crew). Therefore, a lot of algorithm discussion has been excluded unless there is a need.
Algorithm:

1. Extract the following from header: DETNAM, GRATING, OBJECT, OBS_ID, SIM_X, SIM_Y,SIM_Z, DATAMODE, OBS_MODE, RA_NOM, DEC_NOM, ROLL_NOM ,Date_obs Date_end,naxis2
2. Extact from each row: TIME EXPNO GDPX GDPY TG_R TG_D CHIPX CHIPY TDETX TDETY DETX DETY X Y CCD_ID PHA PI GRADE FLTGRADE STATUS CCDNODE TG_M TG_LAM TG_MLAM TG_SRCID TG_PART TG_SMAP
3. Showing and Selecting Events:
   Plots are made to give an idea of what is going on in the data: X-Y plot, X-Energy plot, and an Energy histogram. If ENERGY is not available, PHA is used in its place for these plots, if available. Note for real acis data cosmic rays, etc. are very visible until grade selection takes place.

   The events are then selected based on a variety of criteria with the percentage of events kept by each selection reported. Most of the ACIS cosmic ray events are removed in this selected acis data The uniformly blurred Y axis values are seen in CC mode data. The follows the obs_anal.pro code lines 544 to 614 All these figures should be on the same html page.

   - Create a plot in X,Y (Dety, DetX if X,Y not available)
   - Create a plot of Energy vs Det X
   - Create a historgram of counts/bin vs. PHA
4. Filter events:
   This follows lines 880-1342 of the obs_anal.pro code This includes recreating the same plots as in the previous item, for only the selected events.

- If ASCA grades are available only accept 0,2,3,4 and 6.
- ELSE accept all events

5. Timing Analysis:
   Plots are made of X, Y, and Energy vs time to show any temporal behavior. The time-between-events is calculated and a histogram of this arrival time (since last event) distribution is a useful indicator of the ACIS readout mode. In TE 3.3 second mode events either have the same time or differ by 3.3 s. This is to be compared with an alternating frame mode where there are several distinct values of event-to-event time that occur because of the readout mode implementation. Subarray modes also have a different arrival time histogram. Note that a MARX 2.20 simulation does not simulate these timing properties.

   The follows the obs_anal.pro code lines 1371 to 1495 All these figures should be on the same html page.

   - Create a plot of Detx vs. time
   - Create a plot of Dety vs. time
   - Create a plot of x vs. time (if available)
   - Create a plot of y vs. time (if available)
   - Create a plot of Pha vs. time
   - Create the Arrival time distribution following obs_anal
   - Store AVE_EVENT_RATE - Details at end

6. Determine Zero-order:
   Zero-order is found by successively averaging the event locations and zooming in on the average. Using the /MOUSE keyword, the user can "guide" the s/w. (For example, see the XRCF offaxis example where the s/w's guesses are shown by square symbols and the operator's inputs are the asterisks.) The final zero-order center is determined by an average of the events in the zero-order-region. Using the /OVERRIDE_ZERO keyword, the user can have the final word in zero-order location if this averaging does not work (e.g., not zero-order present!). A final plot set is made of the zero-order region.

The follows the obs_anal.pro code lines 1500 to 1789. This code segment includes a zero order finding routine which should be sufficient. If there are multiple sources, multiple pages/ gif should be generated, one per source. The current IDL Procedure can't do this, but these following tasks could be run with a filter on TG_SRCID. I.E.,
For each TG_SRCID do the following:

- Create a plot of x vs. y (use detx, dety if no aspect)
- Create a plot of Zero order region (2x) y vs x
- Create a plot of Zero order region (6x) y vs x
- Store Zero_order_Location_DETX, Zero_order_Location_DETY, Events_in_Zero_order_Region, and Event_Rate_in_Zero_order_Region.

7. Examine Zero-order Spatial:
   Whatever the spatial coordinates which have been selected for the processing method, they are placed in the arrays AX, AY (whether aspect correction was applied or not) and they are examined in a little more detail here. A crude FWHM of the projected zero-order is made as well as plots vs time for the X and Y event values.

   If the TDET or DET coordinates are the ones being used and if Sky X, Y coordinates are available, then a corresponding plot of the Sky X, Y values is made. This comparison can be used to assess the internal aspect solution quality, etc.

   A rough measure of the zero-order FWHM is generated here and passed on and will later effect the size of the grating-dispersed bins, making them larger if appropriate. The "measure" is by simple determination of bin whose counts are above 1/2 peak height – this is susceptable to noise. In addition, the LRF is fit by a Gaussian and the effective FWHM (2.35*sigma) is reported. As an example of a large zero-order, see the broad zero-order of NGC1068; in this case the bin size for grating spectra in later processing was reduced from an equivalent of E/dE=2000 to E/dE= 300.

   These follows the obs_anal.pro code lines 2252 to 2530 All these figures should be on the same html page. Do the following for each zero order region determined above.

- Create a plot of detx vs dety
- Plot histogram of number of photons in each detx ordinate
- Plot detx vs time.
- Plot dety vs time.
- Plot (sky)y vs Sky(x).
- Plot x vs time.
- Plot y vs time.
- Store DET_RMS_Radius, No_DETX_LRF_pixels_GTp1_peak, XY_RMS_Radius No_X_LRF_pixels_GTp1_peak

8. Measure Grating Angle:
   With zero-order established, events are assigned membership in a spectral "part", this is labeled by HEG/MEG/LEG and plus/minus order. This assignment is based on which spatial region (quadrant) of the image they are in. Additionally, two input KEYWORDS can be used to apply tighter spatial selection (CD_WIDTH) and to offset the extraction region in the cross-dispersion direction (CD_OFFSET).

   These dispersed events, in the TDET system, can be fit to a straight line to determine the grating dispersion axis in the detector system. (To-do: There are better ways to do this determintation including using the known angles and selecting tighter regions, or making event-angle histograms as suggested by Pat Broos.)

   Note that this cannot be carried out for CC mode ACIS-S data. In this case the angle determination is skipped but the events are still put into selections by plus and minus orders. Note that without the extra spatial information the HEG and MEG events can only (and not completely) be separated by the ACIS energy - so all plus order events are assigned to HEG and to MEG and order selection determine which belong or not.

   The follows the obs_anal.pro code lines 2543 to 2715. This segment includes code for determining the grating angles. Once these angles are determined store them (LEG-minus LEG-plus slope, LEG-all, MEG-minus MEG-plus slope, MEG-all, HEG-minus HEG-plus slope, HEG-all (in degrees)) All the listed figures should be on the same html page.

Plot Dety vs Detx for each arm (HEG or MEG or LEG, (plus, minus and all for each)) The arm is indicated by the TGREGION column,

9. Crude Dispersed Spectra:
   The dispersion distances are turned into energies/wavelengths for each of the spectrum "parts" (plus,minus sides, HEG/MEG is present.) For the CC mode events the distances are first corrected by a factor of 1/COS(angle). If ACIS energy information is available order selection can take place – retaining only those events which have ACIS-measured energies within some range of the assumed-first-order diffracted energy. Note that these order selection plots look different for MARX simulated data , and real (CC mode) data (MEG was shuttered so all the MEG events are HEG PHA low-energy tail events!) !

   Histograms are then created from the selected events, e.g. for the MARX simulated data and the real CC mode data (again, the MEG spectra here are not meaningful, the HEG spectra do represent the source, because of frame dropouts (see top plot) there is an exposure map correction that is not accounted for here.)

   This follows the obs_anal.pro code lines 2727 to 3102. Note this section relies heavily on data determined in previous sections. All these figures should be on the same html page.

   - if ACIS - Plot ACIS energy vs grating energy (on a per arm basis) with order boundries overplotted.
   - Plot Counts/bin vs energy (log-log) on a per arm basis.
   - Store E/dEAT1keV

10. Compare Zero-order and Dispersed Spectral:
    The ACIS (if available) and grating spectra are then shown together, corrected for their effective area. This allows a check between the zero-order spectra from ACIS and the grating-dispersed spectra from HEG, MEG, LEG. Note that because the resolution is so different, all the spectra are smoothed by a common E/dE blur of 40 - these smoothed versions are then superposed on the bottom panel. Of course spectra from MARX simulations agree very well at low energies and at high energies ; but even XRCF data is not too bad if allowance is made for pileup and the large MEG image size...

This follows the obs_anal.pro code lines 3102 to 3333 All these figures should be created per arm.

11. Characterize line widths, transform to grating resolution:
    This follows the obs_anal.pro code lines 3333 to 3543 All these figures should be created per arm.

    - Plot of cnts/bin vs energy (broken into 4 segments)
    - Plot fit of each line with peak of over 10 counts. These fits calculate Energy, Wavelength, peak counts, meas_Energy and meas_E/dE (store these)
    - Plot E/dE vs energy
    - Plot dety vs time.

    This item should be created one per L1.5 file. One per grating

    - Plot TG_D vs TG_R.
    - Plot TG_D vs hc/TG_LAM.

To the Database:

- General grating data: OBI, AVE_EVENT_RATE Zero_order_Location_DETX, Zero_order_Location_DETY, Events_in_Zero_order_Region, Event_Rate_in_Zero_order_Region, DET_RMS_Radius, No_DETX_LRF_pixels_GTp1_peak, XY_RMS_Radius, No_X_LRF_pixels_GTp1_peak LEG-minus slope, LEG-plus slope, LEG-all slope, MEG-minus slope MEG-plus slope, MEG-all slope, HEG-minus slope, HEG-plus slope, HEG-all slope (all slopes in degrees)),tstart
- Line data: index, Energy, Wavelength, Peak_counts, meas_Energy, meas_E/dE, meas_FWHM, error (eV), ACF, ACF_error, ASCDS Level Grating,arm,OBI,tstart

Release: 4

Group: DSOPS

Analysis Domain: Spatial

DS Tool Class: 2

DS Tool Category: Monitoring

Spec Name: grat_mon.tex

Spec Category: New

Code Type: ASCDS:w

Code Source: New

This Revison: 0.1 - by SJW

Last Edit: April 6, 1999

```
PRO obs_anal, filename, $
OUTPUT_PREFIX=output_prefix, OUT_DIR=out_dir, $
        COORDS_NAME = coords_name, TDET_SELECT = tdet_select, $
MOUSE=mouse, OVERRIDE_ZERO=override_zero, $
ASPECT=aspect, LINE_ANALYSIS=line_analysis, $
GRATING=grating, LETG=letg, $
DETECTOR=detector, FLIPY=flipy, $
        SCREEN24 = screen24, ZBUFFER=zbuffer, $
TEST_CASE = test_case, XRCF_ROWLAND=xrcf_rowland
;
; - - - - - - - - - - Documentation - - - - - - - - - - - - - - - -
;
; PURPOSE - "Like, what's it all for, dude?"
;  Procedure to analyze the L1 or L1.5 FITS file from an HETG observation
;  Prototype MTA, etc. algorithms..., create informative plots, etc.
;
; ARGUMENTS:
;
;  filename - full file specification to a FITS event file
;
; KEYWORDS:
;
;  OUTPUT_PREFIX - if present, this character string is used as
;                  the prefix name for all output products; if not
;                  specified it is set to the last part of the filename
;                  with .fit or .FITS removed (if present.)
;  OUT_DIR     - specifies directory where products are placed, default is
;                 present directory ( = '.'), trailing / not needed.
;  COORDS_NAME - takes on the values: 'TDET', 'DET', or 'Sky'.
```

```
;                If provided on input it selects which X,Y coordinates to
;                use from the file.
;                If not supplied, it is set to the most advanced coords available
;                in the file.
;  /TDET_SELECT - if set then COORDS_NAME is set to 'TDET'
;  /MOUSE - stops after each plot and waits for a mouse click...  Uses user
;                guidance for zero-order location, but s/w has the final word.
;  /OVERRIDE_ZERO - if set, the s/w stops for mouse input of the zero-order
;                location and the final location used is from the mouse input.
;                (Normally a final average is performed in the zero-order
;                 region to set the location.)
;  /ASPECT - try to create an aspect solution from TDETX, TDETY vs time.
;  /LINE_ANALYSIS - grating energy histograms will be analysed for
;                  lines and E/dE values if this keyword is set.
;  GRATING  - set to 'HETG', 'LETG' or 'NONE', default is 'HETG'
;  /LETG    - short way to set GRATING='LETG' is with /LETG
;  DETECTOR - set to 'ACIS-S','ACIS-I','HRC-I', or 'HRC-S', default is ACIS-S
;  /FLIPY - changes the sign of the y-axis, e.g., HRC DETY to get HEG/MEG in correct
;              quadrants...
;  /SCREEN24 - causes the displayed colors to look OK on a 24bit monitor
;             (and/but messes up the 8 bit gif output!)
;  /ZBUFFER   - use set_plot, 'Z', so physical output device is not needed,
;               e.g., for batch or non-xterm (telnet) operation.
;  TEST_CASE - set to an integer to select from an internal set of
;              test input files and keyword values.
;  /XRCF_ROWLAND - use the XRCF value of the Rowland Diameter
;----------------------------------------------
; 1999/01/22 - 02/05 dd Initial hacking to prototype the HETG Analysis Kit
;                       (HAK) code algorithms and plots...
; 1999/02/06 - 02/12 dd Create HTML output, use .gif plots and color...
;                       Try to clean up the humongeous and kludgy (HAK) code.
;                       Nope, it's still a mess!
; 1999/02/13 -       dd Add ISIS output format for histograms
; 1999/02/16 -       dd tweaking, experimenting w/TDETX for HRC-S, etc.  Add fcp/Tbl_AO1.
; 1999/02/18 -       dd Change HRC DETX,DETY pixel size to agree with ICD
; 1999/02/20 -       dd Add keywords for Z buffer and output directory
; 1999/02/22 -       dd Add KEYWORD COORDS_NAME, and associated logic...
```

```
; 1999/02/25 -       dd Write HTML output to a temp file and later copy to HTML out file;
;                       this allows the "proc_method" value to be known and the
;                       html output file to have proc_method as part of the filename...
; 1999/02/25 -       dd Look into the aspect solution a bit... add apodization
; 1999/02/26 -       dd add Etouse array...
; 1999/03/05 - 03/08 dd clean things up..., add TDET_SELECT, adapt for CC mode,
;                       add XRCF_ROWLAND, keep "low" frequ.s in aspect sol'n.
;----------------------------------------------


; - - - - - Test Cases - - - - - - - - - - - - - -
; The KEYWORD TEST_CASE is used to hard code test cases for obs_anal -
; they can then be simply run with the command:
;        IDL> obs_anal, TEST=3
; The cases dd includes here cover a range of real (XRCF) and
; simulated data sets...

; Check for TEST_CASE first...
if KEYWORD_SET(TEST_CASE) then begin
  ; Some files dd uses... along with any parameters/options
  dd_testfile_dir = '/nfs/spectra/d0/mta'
  CASE test_case OF

  ; ---- XRCF ----
    ; These first two use /MOUSE to let user guide s/w
    ; XRCF: HETG-ACIS
    ; from /nfs/atum/d3/dd/dd_d6/ACIS_anal/H-HAS-PI-1.011new_dd_evt.fits
    1: begin
         filename= dd_testfile_dir+'/H-HAS-PI-1.011ddL1'
         MOUSE = 1  ; need mouse to get zero-order
         XRCF_ROWLAND = 1
         LINE_ANALYSIS = 1
       end
    ; XRCF: LETG-HRC-S
    2: begin
         filename = dd_testfile_dir+'/hrc_glhsdf3006.fits'
         LETG=1
```

```
      DETECTOR='HRC-S'
      COORDS_NAME = 'DET'
      OVERRIDE_ZERO = 1  ; need to force location and FWHM
      LINE_ANALYSIS = 1
      XRCF_ROWLAND = 1
    end

  ; XRCF: HETG-ACIS
  ; from /nfs/atum/d3/dd/dd_d6/ACIS_anal/H-HAS-EA-6.003new_dd_evt.fits
  3: begin
      filename = dd_testfile_dir+'/H-HAS-EA-6.003ddL1'
      XRCF_ROWLAND = 1
    end

  ; XRCF: LETG-ACIS
  ; from /nfs/atum/d3/dd/dd_d6/ACIS_anal/H-LAS-PI_SF-1.006new_dd_evt.fits
  4: begin
      filename= dd_testfile_dir+'/H-LAS-PI_SF-1.006ddL1'
      LETG=1
      LINE_ANALYSIS = 1
      XRCF_ROWLAND = 1
    end

  ; XRCF: HETG-HRC-I
  5: begin
      filename = dd_testfile_dir+'/hrc_ghhiea10003.fits'
      DETECTOR='HRC-I'
      COORDS_NAME = 'DET'
      FLIPY = 1
      XRCF_ROWLAND = 1
    end

  ; XRCF CC example :
  ; obs_anal, '/spectra/d0/mta/H-HAS-MC-3.001new_dd_evt.fits', $
  ; OUT_DIR='/spectra/d0/Obs_out' , /TDET, /OVER, /Line, /XRCF

; ---- marx2fits ----
```

```
; MARX with dither Capella simulation (from marx2fits):
; HETG-ACIS-S Capella
6: begin
     filename = dd_testfile_dir+'/hetg_acis_capella.p3.fits'
     ; Optionally use TDET with aspect to compare with MARX X,Y
     COORDS_NAME='TDET' & ASPECT=1  ; need to do aspect solution
     LINE_ANALYSIS = 1
   end

; LETG-ACIS-S Capella
7: begin
     filename = dd_testfile_dir+'/letg_acis_capella.p3.fits'
     LETG=1
     ; Optionally use TDET with aspect to compare with MARX X,Y
     COORDS_NAME='TDET' & ASPECT=1  ; need to do aspect solution
     LINE_ANALYSIS = 1
   end

; LETG-HRC-S Capella
8: begin
     filename = dd_testfile_dir+'/letg_hrc_capella.p4a.fits'
     LETG=1
     DETECTOR='HRC-S'
     ; Optionally use DET with aspect to compare with MARX X,Y
     COORDS_NAME='DET' & ASPECT=1  ; need to do aspect solution
     LINE_ANALYSIS = 1
   end

;  w/HRC:
; HETG-HRC-I Capella
9: begin
     filename = dd_testfile_dir+'/hetg_hrc_capella.p4a.fits'
     DETECTOR='HRC-I'
     ; Optionally use DET with aspect to compare with MARX X,Y
     COORDS_NAME='DET' & ASPECT=1  ; need to do aspect solution
     LINE_ANALYSIS = 1
   end
```

```
  ; Patrick's NGC4151 MARX Level 1 file
  10: begin
       filename= dd_testfile_dir+'/ngc4151_marxL1.fits'
       ; Use TDETX, TDETY
       TDET_SELECT = 1
       LINE_ANALYSIS = 1 ; any lines?
     end

  ; Patrick's NGC1068 MARX Level 1 file
  11: begin
       filename= dd_testfile_dir+'/ngc1068_d.fits'
       ; Use TDETX, TDETY
       TDET_SELECT = 1
       LINE_ANALYSIS = 1
     end

; ---- Level 1.5 example ----
  ; DaveH's example Level 1.5 file
  12: begin
       filename= dd_testfile_dir+'/AS_004_L1a.fits'
       output_prefix = 'hetgs_L1.5'
       ; Compare Level 1.5 output to TDET analysis:
       COORDS_NAME='TDET'
       LINE_ANALYSIS = 1
     end

  ; Patrick's NGC4151 Level 1.5 file
  13: begin
       filename= dd_testfile_dir+'/ngc4151_L1a.fits'
       ; Compare Level 1.5 output to TDET analysis:
       COORDS_NAME='TDET'
     end

  ELSE: begin
    print, ''
    print, ' TEST_CASE value ('+STRCOMPRESS(TEST_CASE)+') not valid'
```

```
      print, ''
      RETURN
    end
  ENDCASE
end


; - - - - OK, check out filename... - - - - - - - - -
; It's the only required argument, if not present help 'em out...
if n_elements(filename) EQ 0 then begin
  print,''
  print, " Usage: obs_anal, 'fits_file_name' [,KEYWORDs]
  print, '   '
  print, '  or  : obs_anal, TEST_CASE=1 [,KEYWORDs (unless specified in code)]'
  print, ''
  ; Take documentation above and print it:
  print, "   ; ARGUMENTS:"
  print, "   ;"
  print, "   ;  filename - full file specification to a FITS event file"
  print, "   ;"
  print, "   ; KEYWORDS:"
  print, "   ;"
  print, "   ;  OUTPUT_PREFIX - if present, this character string is used as"
  print, "   ;                  the prefix name for all output products; if not"
  print, "   ;                  specified it is set to the last part of the filename"
  print, "   ;                  with .fit or .FITS removed (if present.) "
  print, "   ;  OUT_DIR     - specifies directory where products are placed, default is"
  print, "   ;                 present directory ( = '.')."
  print, "   ;  COORDS_NAME - takes on the values: 'TDET', 'DET', or 'Sky'."
  print, "   ;                If provided on input it selects which X,Y coordinates to"
  print, "   ;                use from the file."
  print, "   ;                If not supplied, it is set to the most advanced coords available"
  print, "   ;                in the file."
  print, "   ;  /TDET_SELECT - if set then COORDS_NAME is set to 'TDET'"
  print, "   ;  /MOUSE - stops after each plot and waits for a mouse click...  Uses user"
  print, "   ;                guidance for zero-order location, but s/w has the final word."
  print, "   ;  /OVERRIDE_ZERO - if set, the s/w stops for mouse input of the zero-order"
```

```
  print, "   ;                location and the final location used is from the mouse input."
  print, "   ;                (Normally a final average is performed in the zero-order"
  print, "   ;                 region to set the location.)"
  print, "   ;  /ASPECT - try to create an aspect solution from TDETX, TDETY vs time."
  print, "   ;  /LINE_ANALYSIS - grating energy histograms will be analysed for"
  print, "   ;                  lines and E/dE values if this keyword is set."
  print, "   ;  GRATING  - set to 'HETG', 'LETG' or 'NONE', default is 'HETG'"
  print, "   ;  /LETG    - short way to set GRATING='LETG' is with /LETG"
  print, "   ;  DETECTOR - set to 'ACIS-S','ACIS-I','HRC-I', or 'HRC-S', default is ACIS-S"
  print, "   ;  /FLIPY - changes the sign of the y-axis, e.g., HRC DETY to get HEG/MEG in correct"
  print, "   ;              quadrants..."
  print, "   ;  /SCREEN24 - causes the displayed colors to look OK on a 24bit monitor"
  print, "   ;             (and/but messes up the 8 bit gif output!)"
  print, "   ;  /ZBUFFER   - use set_plot, 'Z', so physical output device is not needed,"
  print, "   ;               e.g., for batch or non-xterm operation."
  print, "   ;  TEST_CASE - set to an integer to select from an internal set of"
  print, "   ;              test input files..."
  print, "   ;  /XRCF_ROWLAND - use the XRCF value of the Rowland Diameter"
  print, ''
  RETURN
end


; - - - - Default values for OUTPUT_PREFIX and OUT_DIR - - - - - - -
if NOT(KEYWORD_SET(OUTPUT_PREFIX)) then begin
  pieces = STR_SEP(filename,'/')
  ; get the last piece...
  last_str = pieces(n_elements(pieces)-1)
  ; remove .fits or .FITS or even .FitsIsSoHardToRead !
  where_fits = STRPOS(STRUPCASE(last_str),'.FITS')
  if where_fits GT 0 then begin
    ; Use everything upto the .FITS:
    output_prefix = STRMID(last_str, 0, where_fits)
  end else begin
    ; OK, no FITS, use it as is...
    output_prefix = last_str
  end
```

```
end
if NOT(KEYWORD_SET(OUT_DIR)) then begin
  out_dir = '.'
end


; - - - - - - - - KEYWORD defaults, etc. - - - - - - - -

; COORDS_NAME and TDET logic...
;
if KEYWORD_SET(TDET_SELECT) then COORDS_NAME = 'TDET'

if n_elements(coords_name) EQ 0 then begin
  coords_name = 'Not supplied on input'
end else begin
  ; user supplied a valid value?
  allowed_coords = ['DET','TDET','Sky']
  if TOTAL(coords_name EQ allowed_coords) EQ 0 then begin
    print, ''
    print, ' *** COORDS_NAME not valid (DET, TDET, Sky, or Not supplied) ***'
    print, '     COORDS_NAME set to: Not supplied on input'
    print, ''
  end
end

; Default grating logic...
if KEYWORD_SET(LETG) then begin
  GRATING='LETG'
end
if NOT(KEYWORD_SET(GRATING)) then GRATING='HETG'
; Default DETECTOR logic
if NOT(KEYWORD_SET(DETECTOR)) then DETECTOR='ACIS-S'
;
; Check consistancy of ZBUFFER and MOUSE, SCREEN24
if KEYWORD_SET(ZBUFFER) then begin
  if KEYWORD_SET(MOUSE) OR KEYWORD_SET(OVERRIDE_ZERO) then begin
    print, ''
```

```
    print, ' *** MOUSE and OVERRIDE_ZERO not available with ZBUFFER! ***'
    print, '     Setting them to 0 '
    print, ''
    MOUSE=0
    OVERRIDE_ZERO=0
  end
  if KEYWORD_SET(SCREEN24) then begin
    print, ''
    print, ' *** SCREEN24 not useful with ZBUFFER! ***'
    print, '     SCREEN24 set to 0 '
    print, ''
    SCREEN24=0
  end
end


; - - - - - - - create a temporary output file for now - - - - - - -
OPENW, out_unit, out_dir+'/temp_summary.html', /GET


; - - - - - - - - Start output... - - - - - - - - - - -
; Summarize what the inputs are... (except for action-creating
; keywords ...)
print, ''
print, ' - - - obs_anal.pro - - - - - - - - - - '+SYSTIME()+' - - - '
print, ''
printf, out_unit, SYSTIME()+' : obs_anal.pro started'
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Inputs</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

if KEYWORD_SET(TEST_CASE) then begin
  print, '   Test case '+STRCOMPRESS(test_case,/REMOVE_ALL)+':'
  print, ""
  printf, out_unit, '   Test case '+STRCOMPRESS(test_case,/REMOVE_ALL)+':'
```

```
  printf, out_unit, ""
end

print, ' Filename        = '+filename
print, ' Output_prefix   = '+output_prefix
print, ' Output dir.     = '+out_dir
print, ' Grating         = '+grating
print, ' Detector        = '+detector
print, '   Spatial coord = '+coords_name

printf, out_unit, ' Filename        = '+filename
printf, out_unit, ' Output_prefix   = '+output_prefix
printf, out_unit, ' Output dir.     = '+out_dir
printf, out_unit, ' Grating         = '+grating
printf, out_unit, ' Detector        = '+detector
printf, out_unit, '   Spatial coord = '+coords_name


; Indicate if the FLIPY is set
if KEYWORD_SET(FLIPY) then begin
  print, '   * FLIPY IS set!'
  printf, out_unit, '   * FLIPY IS set!'
end else begin
  print, '   (FLIPY is not set)'
  printf, out_unit, '   (FLIPY is not set)'
end

if KEYWORD_SET(MOUSE) then begin
  print, ' MOUSE keyword set'
  printf, out_unit, ' MOUSE keyword set'
end
if KEYWORD_SET(OVERRIDE_ZERO) then begin
  print, ' OVERRIDE_ZERO keyword set'
  printf, out_unit, ' OVERRIDE_ZERO keyword set'
end
if KEYWORD_SET(ASPECT) then begin
  print, ' ASPECT keyword set'
```

```
  printf, out_unit, ' ASPECT keyword set'
end
if KEYWORD_SET(XRCF_ROWLAND) then begin
  print, ' XRCF_ROWLAND keyword set'
  printf, out_unit, ' XRCF_ROWLAND keyword set'
end
if KEYWORD_SET(LINE_ANALYSIS) then begin
  print, ' LINE_ANALYSIS keyword set'
  printf, out_unit, ' LINE_ANALYSIS keyword set'
end
print, ''
printf, out_unit, ''

; NOTE: since this is one big huge hunk of code, there are a ton
; of parameters, arrays, etc that are created during execution.
; Sprinkled though the code are these reminders/summaries of
; what (useful) parameters are available at this stage of
; processing.
; .......................................
;    - Data and Parameters available at this stage:
; filename, output_prefix, theKEYWORDS, out_unit, iwind, clr_*, color24_values
; .......................................

print, ' - - - - - - - - - - - - - Getting Data ... - - - - - - - - - - '
print, ''
print, ' Source of data: '+filename
print, ''

printf, out_unit, '</PRE>'
printf, out_unit, '<HR>'
printf,out_unit, '<H3>Getting Data ...</H3>'
printf,out_unit, '<PRE>'
printf,out_unit, ''
printf,out_unit, ' Source of data: '+filename
printf,out_unit, ''
; Read in the file...
print, ' Reading FITS file: '+filename
```

```
print, ''
printf,out_unit, ' Reading FITS file: '+filename
printf,out_unit, ''
; Always an Empty primary
iext=0
dummy = mrdfits(filename, iext)
; Look for the first extension with TDETX in it...
; (needed because other extensions are possible too, e.g.,
;  the GTI(?) extension with START and STOP values)
found_it = 0 GT 0
while NOT(found_it) do begin
  iext = iext+1
  evts = mrdfits(filename, iext)
  evts_tags = tag_names(evts)
  found_it = TOTAL(evts_tags EQ 'TDETX') GT 0
end

;
; What could/do we have:
;
;     L1 Columns
;
; Timing related:
;  EXPNO TIME
;
; Event Location:
;   (aspect correction is first included in Y X)
;  Y X DETY DETX TDETY TDETX CHIPY CHIPX CCDNODE CCD_ID
;
; Event PHA, Energy, etc. properties
;  STATUS FLTGRADE GRADE ENERGY PI PHA
;
;     L1.5 Columns
;
;  TG_SMAP TG_PART TG_SRCID TG_MLAM TG_LAM TG_M TG_D TG_R GDPY GDPX
;
```

```
print, ' Tag names are:'
print, evts_tags
print, ''
printf,out_unit, ' Tag names are:'
printf,out_unit, evts_tags
printf,out_unit, ''

; Check on ENERGY and TIME here...
; Setup arrays for each of these with "converted" values for
; uniformity, e.g., energy in keV.
Etouse = FLTARR(n_elements(evts))

; Put ENERGY into keV
energy_unit = 'n/a'
if TOTAL(evts_tags EQ 'ENERGY') GE 1 then begin
  ; If the energy is present, convert it to keV
  Etouse = evts.ENERGY
  if MAX(Etouse) GT 1000.0 then begin
    print, '* Looks like ENERGY is in eV, converting to keV'
    printf, out_unit, '* Looks like ENERGY is in eV, converting to keV'
    Etouse = Etouse/1000.0
  end else begin
    print, ' Looks like ENERGY is in keV'
    printf, out_unit, ' Looks like ENERGY is in keV'
  end
  energy_unit = 'keV'
end else begin
  print, '* No ENERGY available.'
  printf, out_unit, '* No ENERGY available.'
end

if TOTAL(evts_tags EQ 'TIME') GE 1 then begin
  ; Create the time since first event... ("Floating Time")
  print, ' Creating ftime from TIME'
  printf, out_unit, ' Creating ftime from TIME'
  abs_start_time = MIN(evts.TIME)
  ftime = DOUBLE(evts.TIME - abs_start_time)
```

```
  interval_duration = MAX(ftime)
  time_unit = 'second'
end else begin
  print, ' No TIME information, use event number as TIME'
  printf, out_unit, ' No TIME information, use event number as TIME'
  ftime = DOUBLE(LINDGEN(n_elements(evts)))
  abs_start_time = 0.0
  interval_duration = MAX(ftime)
  time_unit = 'event'
end
live_interval = interval_duration
print, ''
printf, out_unit, ''

; And report the most basic info...
print, ' Total Events = '+STRCOMPRESS(n_elements(evts))
print, ' Start time = '+STRCOMPRESS(abs_start_time)+' '+time_unit+'s'
print, ' Interval duration = '+STRCOMPRESS(interval_duration)+' '+time_unit+'s'
print, ' Ave Event Rate = '+STRCOMPRESS(FLOAT(n_elements(evts))/ $
interval_duration)+' events/'+time_unit
print, ''
printf, out_unit, ' Total Events = '+STRCOMPRESS(n_elements(evts))
printf, out_unit, ' Start time = '+ $
STRCOMPRESS(abs_start_time)+' '+time_unit+'s'
printf, out_unit, ' Interval duration = '+ $
STRCOMPRESS(interval_duration)+' '+time_unit+'s'
printf, out_unit, ' Ave Event Rate = '+ $
STRCOMPRESS(FLOAT(n_elements(evts))/ $
interval_duration)+' events/'+time_unit
printf, out_unit, ''
; - - - - - end of FITS input - - - - -


; ....................................
;    - Data and Parameters available at this stage:
; filename, output_prefix, theKEYWORDS, out_unit, iwind, clr_*, color24_values
; evts, evts_tags, energy_unit,
```

```
; abs_start_time, interval_duration, ftime, time_unit
; ......................................


; - - - - - - - - Coordinates and Pixel size - - - - - - - -
print, ' - - - - - - - - - - - - Coordinates and Pixel Size... - - - - - - - - '
print, ''
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Coordinates and Pixel Size...</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

; Check for ACIS-S CC mode where TDETY is a constant value
; This will not work for ACIS-I ?

if (DETECTOR EQ 'ACIS-S') then begin
  if  (MIN(evts.TDETY) EQ MAX(evts.TDETY)) then begin
    cc_mode = (1 EQ 1)
    print, '* Looks like ACIS-S CC mode data'
    printf, out_unit, '* Looks like ACIS-S CC mode data'
  end else begin
    print, ' (Not CC mode data)'
    printf, out_unit, ' (Not CC mode data)'
    cc_mode = (1 EQ 0)
  end
end else cc_mode = (1 EQ 0)

; Set the value of COORDS_NAME if it was not supplied
if STRPOS(coords_name, 'Not') GE 0 then begin
  ; Check for the Sky coord.s, X,Y :
  if (TOTAL(evts_tags EQ 'X') GE 1) then begin
    coords_name = 'Sky'
  end else begin
    if (TOTAL(evts_tags EQ 'DETX') GE 1) then begin
      coords_name = 'DET'
    end else begin
```

```
      coords_name = 'TDET'  ; assuming these are always available...
    end
  end
end else begin
  ; OK, the user specified a desired coordinates to use
  ; Verify that the desired coordinates are in the tags - if not demote them...
  ; Asked for Sky but not there
  if ( (coords_name EQ 'Sky') AND NOT(TOTAL(evts_tags EQ 'X') GE 1) )then begin
    coords_name = 'DET'
    print, '* Sky coordinates (X,Y) not available - try DET'
    printf, out_unit, '* Sky coordinates (X,Y) not available - try DET'
  end
  ; Asked for DET but not there
  if ( (coords_name EQ 'DET') AND NOT(TOTAL(evts_tags EQ 'DETX') GE 1) )then begin
    print, '* DET coordinates not available - using TDET'
    printf, out_unit, '* DET coordinates not available - using TDET'
    coords_name = 'TDET'
  end
end

; Set the "processing method" to indicate how the spatial coordinates
; for grating dispersion are arrived at... (for L1.5 will use 'L1.5').
; Use this in the .gif output filenames.
if KEYWORD_SET(ASPECT) then begin
  proc_method = 'a'+coords_name
end else begin
  proc_method = coords_name
end


; For plot labeling define the basic axis names... to be modified
; in the following to indicate sign flipping, bluring, etc.
x_name = coords_name+'X'
y_name = coords_name+'Y'

; Based on the COORDS selection, create and fill the arrays Xtouse Ytouse and
; set the pixel size and axis names.
```

```
; The DETX, DETY and SkyX, SkyY pixels are defined in angular units, so the
; conversion to mm requires the angular size of a pixel and
; a focal length:
;    pix_ang_size = 0.132   ; arc sec
;    pix_foc_leng = 10065.5 ; mm
;    pixel_size = pix_foc_leng * !DTOR*pix_ang_size/3600.
;
; NOTE that the DetY nominally gets a -1 to keep same orientation
; as TDET and Sky systems...
CASE coords_name OF
  'Sky': begin
    ; No bluring applied here: sky coords should be aspect corrected, etc.
    ; and in real, non-quantized, values
    Xtouse = FLOAT(evts.X)
    Ytouse = FLOAT(evts.Y)
    if STRPOS(DETECTOR,'HRC') GE 0 then begin
      pix_ang_size = 0.132   ; arc sec
      pix_foc_leng = 10065.5 ; mm
      pixel_size = pix_foc_leng * !DTOR*pix_ang_size/3600.
    end else begin
      pix_ang_size = 0.492   ; arc sec
      pix_foc_leng = 10065.5 ; mm
      pixel_size = pix_foc_leng * !DTOR*pix_ang_size/3600.
    end
  end
  'DET': begin
    Xtouse = FLOAT(evts.DETX)
    Ytouse = -1.0*FLOAT(evts.DETY)
    ; These coordinates are still quantized to a pixel, so
    ; blur by +/- 0.5 pixel to avoid aliassing with binning...
    Xtouse = FLOAT(Xtouse) + (randomu(SEED,n_elements(evts))-0.5)
    Ytouse = FLOAT(Ytouse) + (randomu(SEED,n_elements(evts))-0.5)
    x_name = x_name + 'b'
    y_name = y_name + 'b'
    if STRPOS(DETECTOR,'HRC') GE 0 then begin
      pix_ang_size = 0.132   ; arc sec
      pix_foc_leng = 10065.5 ; mm
```

```
      pixel_size = pix_foc_leng * !DTOR*pix_ang_size/3600.
    end else begin
      pix_ang_size = 0.492   ; arc sec
      pix_foc_leng = 10065.5 ; mm
      pixel_size = pix_foc_leng * !DTOR*pix_ang_size/3600.
    end
  end
  'TDET': begin
    Xtouse = FLOAT(evts.TDETX)
    Ytouse = FLOAT(evts.TDETY)
    ; These coordinates are still quantized to a pixel, so
    ; blur by +/- 0.5 pixel to avoid aliassing with binning...
    Xtouse = FLOAT(Xtouse) + (randomu(SEED,n_elements(evts))-0.5)
    Ytouse = FLOAT(Ytouse) + (randomu(SEED,n_elements(evts))-0.5)
    x_name = x_name + 'b'
    y_name = y_name + 'b'
    if STRPOS(DETECTOR,'HRC') GE 0 then begin
      pixel_size = 0.0064294 ; mm
    end else begin
      pixel_size = 0.0240000 ; mm
    end
  end
  ELSE: begin
    ; Can't arrive here...
  end
ENDCASE

; Now some Y-madness: CC and FLIPY
; add prefixes to y_name
if cc_mode then begin
  ; The Y value is meaningless so
  ; assign the yaxis a +/-4 pixel blur
  Ytouse = Ytouse + $
4.0 * 2.*(randomu(SEED,n_elements(Ytouse))-0.5)
  print, '* CC mode: blur Y by +/- 4 pixels'
  printf, out_unit, '* CC mode: blur '+y_name+' by +/- 4 pixels'
  y_name = 'cc'+y_name
```

```
end else begin
  ; The FLIPY is here for HRC but it will work for other detectors too...
  ; Add a sign to the Y name:
  if KEYWORD_SET(FLIPY) then begin
    print, '* Changing the sign of '+y_name
    printf, out_unit, '* Changing the sign of '+y_name
    Ytouse = -1.0 * Ytouse
    ; YFLIP: TDET and Sky get -, DET gets +
    if coords_name EQ 'DET' then begin
      y_name = '+'+y_name
    end else begin
      y_name = '-'+y_name
    end
  end else begin
    ; No flip: TDET and Sky get no prefix, DET gets -
    if coords_name EQ 'DET' then begin
      y_name = '-'+y_name
    end else begin
      ; y_name = y_name
    end
  end
end
print, ''
printf, out_unit, ''

; Report the Coords results...
print, ' Spatial coordinates are : '+ x_name +', '+ y_name
printf, out_unit, ' Spatial coordinates are : '+ x_name +', '+ y_name
print, ' pixel size  = '+STRING(1.E3*pixel_size,FORMAT='(F7.4)')+' microns'
printf, out_unit, ' pixel size  = '+STRING(1.E3*pixel_size,FORMAT='(F7.4)')+' microns'
print, ' Processing method = '+proc_method
printf, out_unit, ' Processing method = '+proc_method
print, ''
printf, out_unit, ''
```

```
; - - - - - - - - - - - - Setup HTML output - - - - - - - -
; OK that we have proc_method we can open the output HTML file
; and write out the lines that have been stored up...
; - - - - - - - - - HTML output file - - - - - - - - -
;
; Close the temporary file
CLOSE, out_unit
; and open the real file...
OPENW, out_unit, out_dir+'/'+output_prefix+'_'+proc_method+'_summary.html', /GET
; Setup HTML header etc.
printf, out_unit, '<HTML>'
printf, out_unit, '<HEAD>'
printf, out_unit, '  <TITLE>'+output_prefix+'</TITLE>'
printf, out_unit, '</HEAD>
printf, out_unit, '
printf, out_unit, '<BODY TEXT="#DDDDDD"  BGCOLOR="#000000" '
printf, out_unit, 'LINK="#FF0000" VLINK="#FFFF22" ALINK="#7500FF">'
printf, out_unit, ''
printf, out_unit, '  <H2 align="center">'+output_prefix+'_'+proc_method+'</H2>'
printf, out_unit, ''
; Keep it simple, default to <PRE> format...
printf, out_unit, '<PRE>'
;
; Now add the temporary info to the output file
OPENR, temp_unit, out_dir+'/temp_summary.html', /GET
while NOT(EOF(temp_unit)) do begin
  line_in = ''
  readf, temp_unit, line_in
  printf, out_unit, line_in
end
CLOSE, temp_unit
FREE_LUN, temp_unit


; - - - - - - - - - - - - Setup plot output - - - - - - - -
; Plot to .gif
; Setup a window...
```

```
;
; Save the original device and plot values (thanks Jim!)
Orig_device = !d.name
Orig_plot = !p      ; save it all?
;
if KEYWORD_SET(ZBUFFER) then begin
  set_plot, 'Z'
end else begin
  set_plot, 'X'
  zbuffer = 0  ; can pass it to other routines...
end

; Output window
iwind = 0
if KEYWORD_SET(ZBUFFER) then begin
  device, set_resolution=[700,700]
end else begin
  window, iwind, xsi=700, ysi=700
end

; Hang in there with me on this - I'm new to 24bit color!
; See if it is 24 bit color...
color24bit = !d.n_colors EQ 16777216

; Color table for PHA-->color coding
; 0=black, 1=white, 2=darkbackground, 3-255 = red to blue
  dd_load_ct, RED=red_ct, GREEN=green_ct, BLUE=blue_ct, /OTHER

; Set colors: if it is 8 bit then the colors are
; 0 to 255 according to the table above; if it is
; 24 bit then the "color" is the 24bit RGB combination
; in that case the 0-255 color table value is converted
; to 24 bit color:
color24_values = red_ct + 256L * (green_ct + 256L * blue_ct)

;
; Color is used here for two purposes:
```

```
;  1) delineate different curves on plots
;  2) color-code photons by their energy
;
; Set some curve colors based on 8 bit, then convert
; to 24 bit values if needed.
clr_back = 0  ; black
clr_wht = 1  ; white
clr_red = 3
clr_org = 40
clr_yel = 85
clr_grn = 125
clr_blu = 250
;
if color24bit AND KEYWORD_SET(SCREEN24) then begin
  print, ' Using 24 bit colors for display...'
  print, ''
  ic = clr_back
  clr_back = color24_values(ic)
  ic = clr_wht
  clr_wht = color24_values(ic)
  ic = clr_red
  clr_red = color24_values(ic)
  ic = clr_org
  clr_org = color24_values(ic)
  ic = clr_yel
  clr_yel = color24_values(ic)
  ic = clr_grn
  clr_grn = color24_values(ic)
  ic = clr_blu
  clr_blu = color24_values(ic)
end
;
; NOTE: The SCREEN24 KEYWORD will caluse 24 bit colors to show on the screen as
; correct colors BUT the write_gif will NOT be correct -
; So, use /SCREEN24 for viewing at terminal, rerun without /SCREEN24 to
; get valid .gif files...
;
```

```
; ......................................
;    - Data and Parameters available at this stage:
; filename, output_prefix, theKEYWORDS, out_unit, iwind, clr_*, color24_values
; evts, evts_tags, energy_unit, interval_duration, ftime, time_unit
; cc_mode, pixel_size, Xtouse, x_name, Ytouse, y_name, proc_method
; ......................................

; - - - - - - - Selection Criteria for Nominally Valid Events - - - -
; Report the effects of each selection...
; Parameters:
; Fractional time range to use:
ft_frac_min = 0.0  ; 0.0 default, for MC-3.001: 0.460 (4ks)
ft_frac_max = 1.0  ; 1.0 default,               0.578 (5ks)
; Energies to use:
max_sel_energy = 12.0 ; keV
min_sel_energy = 0.2  ; keV
; Energies for color coding (red to blue)
max_clr_energy = 8.0 ; keV
min_clr_energy = 0.2  ; keV
; Bin size for ENERGY histograms
e_bin_ratio = 1.02  ; En+1/En
; PHA region for selection
min_sel_pha = 40.0  ; for HRC only
max_sel_pha = 250.0  ; for HRC only
; PHA range for plotting and color coding (not for selection)
min_clr_pha = 0
if STRPOS(DETECTOR,'ACIS') GE 0 then begin
  max_clr_pha = 2500
end else begin
  max_clr_pha = 256
end
; For HRC-I: Select ACIS-like cross-dispersion width for
; consideration...
hrci_yreg = 4000.0  ; hrc pixels ~ 1024 ACIS pixels
;
print, ' - - - - - - - - - - - - Showing and Selecting Events... - - - - - - - '
```

```
print, ''
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Showing and Selecting Events...</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

; First...
; Show all events in Xtouse, Ytouse
; and a histogram of ENERGY (or PHA) if available

!p.multi = [0,1,3]
; ENERGY or PHA available?
if (TOTAL(evts_tags EQ 'ENERGY') GE 1) OR $
(TOTAL(evts_tags EQ 'PHA') GE 1) then begin
  ; OK, we have something - make more plots
  ; Prefer to use energy if it is present
  if TOTAL(evts_tags EQ 'ENERGY') GE 1 then begin
    ; Log energy color coding...
    ; energy_colors = 3+FIX(252.* $
    ; (ALOG(Etouse) - ALOG(min_clr_energy))/ $
    ; (ALOG(max_clr_energy) - ALOG((min_clr_energy > 0.06))) )
    ; Prefer..
    ; Linear coding...
    energy_colors = 3+FIX(252.* $
(Etouse - min_clr_energy)/ $
(max_clr_energy - min_clr_energy) )
  end else begin
    ; PHA available...
    energy_colors = 3+FIX(252.* $
(evts.PHA - min_clr_pha)/ $
(max_clr_pha - min_clr_pha) )
  end
  if color24bit AND KEYWORD_SET(SCREEN24) then begin
    ; convert the colors to 24 bit...
    energy_colors = color24_values(energy_colors)
  end
```

```
end else begin
  ; No ENERGY or PHA to provide color coding...
  energy_colors = clr_grn
end

plot, Xtouse, Ytouse, PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK = clr_back, COLOR = clr_wht, /NODATA, $
        TITLE='X-Y Plot of ALL Events', $
        XTITLE=x_name+' (pixels)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.81
plots, Xtouse, Ytouse, PSYM=3, COLOR=energy_colors
; ENERGY plots if available
if (TOTAL(evts_tags EQ 'ENERGY')) GE 1 then begin
  ; /CLIP doesn't seem to work here??
  ; brute force clip the values at 2x max_sel_energy
  use_em = where(Etouse LT 2.*max_sel_energy)
  plot_io, Xtouse, Etouse, PSYM=3, $
XSTYLE=1, YSTYLE=1, YRANGE=[min_sel_energy,max_sel_energy], $
BACK=clr_back, COLOR=clr_wht, /NODATA, /CLIP, $
        TITLE='X-E Plot of ALL Events', $
        XTITLE=x_name+' (pixels)', $
        YTITLE='Detector Energy ('+energy_unit+')', $
CHARSIZE=1.81
  plots, Xtouse(use_em), Etouse(use_em), $
PSYM=3, COLOR=energy_colors(use_em), /CLIP
  log_hist, Etouse, e_bin_ratio, bines, bincounts
  plot_oo, bines, bincounts, PSYM=10, $
XRANGE=[min_sel_energy,max_sel_energy], XSTYLE=1, $
YRANGE=[0.5,2.0*MAX(bincounts)], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, $
        TITLE='Detector Pulse Height Distribution of ALL Events', $
        XTITLE='Detector Energy ('+energy_unit+')', $
        YTITLE='Counts/bin', $
CHARSIZE=1.81
end else begin
```

```
  ; PHA plots if available and ENERGY isn't
  if (TOTAL(evts_tags EQ 'PHA')) GE 1 then begin
    plot, Xtouse, evts.PHA, PSYM=3, $
XSTYLE=1, YRANGE=[0.,MAX(evts.PHA)], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, /CLIP, $
        TITLE='X-PHA Plot of ALL Events', $
        XTITLE=x_name+' (pixels)', $
        YTITLE='Detector PHA', $
CHARSIZE=1.81
    plots, Xtouse, evts.PHA, $
PSYM=3, COLOR=energy_colors, /CLIP
    lin_hist, evts.PHA, 1.0, bines, bincounts
    plot, bines, bincounts, PSYM=10, $
XSTYLE=1, XRANGE=[0.,MAX(evts.PHA)], $
YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, $
        TITLE='Detector Pulse Height Distribution of ALL Events', $
        XTITLE='Detector PHA', $
        YTITLE='Counts/bin', $
CHARSIZE=1.81
  end
end
nevts = n_elements(evts)

; Finish the .gif output
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_showALL.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+'Plots of ALL Events'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"
```

```
if KEYWORD_SET(MOUSE) then begin
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end

; Now perform selections in sequence...

; TIME
keep = where(ftime GE ft_frac_min*interval_duration AND $
ftime LE ft_frac_max*interval_duration, nfound)
if nfound GT 0 then begin
  print, ' Keeping TIME-in-range events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
  print, '  (from '+STRCOMPRESS(ft_frac_min*interval_duration)+$
' to '+STRCOMPRESS(ft_frac_max*interval_duration)+')'
  printf, out_unit, ' Keeping TIME-in-range events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
  printf, out_unit, '  (from '+STRCOMPRESS(ft_frac_min*interval_duration)+$
' to '+STRCOMPRESS(ft_frac_max*interval_duration)+')'
  evts = evts(keep)
  fstart_time = MIN(ftime(keep))
  ftime = ftime(keep) - fstart_time
  interval_duration = MAX(ftime)
  live_interval = interval_duration
  abs_start_time = abs_start_time + fstart_time
  Etouse = Etouse(keep)
  Xtouse = Xtouse(keep)
  Ytouse = Ytouse(keep)
  nevts = n_elements(evts)
  print, ' Start time = '+STRCOMPRESS(abs_start_time)+ $
' '+time_unit+'s'
  print, ' Interval duration = '+STRCOMPRESS(interval_duration)+ $
' '+time_unit+'s'
  printf, out_unit, ' Start time = '+STRCOMPRESS(abs_start_time)+ $
' '+time_unit+'s'
```

```
  printf, out_unit, ' Interval duration = '+STRCOMPRESS(interval_duration)+ $
' '+time_unit+'s'
end else begin
  print, ' There were no TIME-in-range events, keep them all.'
  printf, out_unit, ' There were no TIME-in-range events, keep them all.'
end

; HRC Spatial
; Limit HRC to ~ACIS width region in Y for grating observations...
if (STRPOS(DETECTOR,'HRC') GE 0) AND (GRATING NE 'NONE') then begin
  aveY = TOTAL(FLOAT(Ytouse))/n_elements(evts)
  yoffset = Ytouse - aveY
  keep = where( ABS(yoffset) LT hrci_yreg/2.0, nfound)
  if nfound GT 0 then begin
    print, ' HRC: Keeping central stripe Y events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    printf, out_unit, ' HRC: Keeping central stripe Y events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    evts = evts(keep)
    ftime = ftime(keep)
    Etouse = Etouse(keep)
    Xtouse = Xtouse(keep)
    Ytouse = Ytouse(keep)
    nevts = n_elements(evts)
  end else begin
    print, ' There were no HRC events out of the central stripe region,' + $
' keep them all.'
    printf, out_unit, ' There were no HRC-I events out of the central stripe region,' + $
' keep them all.'
  end
end


; STATUS/QUALCODE
if STRPOS(DETECTOR,'HRC') GE 0 then begin
  ; HRC STATUS...
  if TOTAL(evts_tags EQ 'STATUS') GE 1 then begin
```

```
    keep = where( (evts.STATUS AND '3FFF00'xL) EQ 0, nfound)
    if nfound GT 0 then begin
      print, ' Keeping "(STATUS AND 3FFF00xL) EQ 0" events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
      printf, out_unit, ' Keeping "(STATUS AND 3FFF00xL) EQ 0" events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
      evts = evts(keep)
      ftime = ftime(keep)
      Etouse = Etouse(keep)
      Xtouse = Xtouse(keep)
      Ytouse = Ytouse(keep)
      nevts = n_elements(evts)
    end else begin
      print, ' There were no "(STATUS AND 3FFF00xL) EQ 0" events,' + $
' keep them all.'
      printf, out_unit, ' There were no "(STATUS AND 3FFF00xL) EQ 0" events,' + $
' keep them all.'
    end
  end else begin
    print, ' No STATUS selection'
    printf, out_unit, ' No STATUS selection'
  end
end else begin
  ; ACIS STATUS...
  if TOTAL(evts_tags EQ 'STATUS') GE 1 then begin
    keep = where(evts.STATUS EQ 0, nfound)
    if nfound GT 0 then begin
      print, ' Keeping STATUS=0 events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
      printf, out_unit, ' Keeping STATUS=0 events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
      evts = evts(keep)
      ftime = ftime(keep)
      Etouse = Etouse(keep)
      Xtouse = Xtouse(keep)
      Ytouse = Ytouse(keep)
      nevts = n_elements(evts)
```

```
    end else begin
      print, ' There were no STATUS=0 events, keep them all.'
      printf, out_unit, ' There were no STATUS=0 events, keep them all.'
    end
  end else begin
    print, ' No STATUS selection'
    printf, out_unit, ' No STATUS selection'
  end
end

if TOTAL(evts_tags EQ 'QUALCODE') GE 1 then begin
  keep = where(evts.QUALCODE EQ 0, nfound)
  if nfound GT 0 then begin
    print, ' Keeping QUALCODE=0 events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    printf, out_unit, ' Keeping QUALCODE=0 events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    evts = evts(keep)
    ftime = ftime(keep)
    Etouse = Etouse(keep)
    Xtouse = Xtouse(keep)
    Ytouse = Ytouse(keep)
    nevts = n_elements(evts)
  end else begin
    print, ' There were no QUALCODE=0 events, keep them all.'
    printf, out_unit, ' There were no QUALCODE=0 events, keep them all.'
  end
end else begin
  print, ' No QUALCODE selection'
  printf, out_unit, ' No QUALCODE selection'
end

; Make a nominal selection on GRADE (= 0,2,3,4,6)
if TOTAL(evts_tags EQ 'GRADE') GE 1 then begin
  ; print, the grade distribution
  print, ' GRADE distribution:'
  printf, out_unit, ' GRADE distribution:'
```

```
  grade_hist = histogram(evts.grade)
  for ig=0,n_elements(grade_hist)-1 do begin
    print, '  '+STRING(ig,FORMAT='(I3)')+' : ' + $
STRING(grade_hist(ig),FORMAT='(I8)')
    printf, out_unit, '  '+STRING(ig,FORMAT='(I3)')+' : ' + $
STRING(grade_hist(ig),FORMAT='(I8)')
  end
  keep = where( (evts.GRADE EQ 0 OR evts.GRADE EQ 2 OR $
 evts.GRADE EQ 3 OR evts.GRADE EQ 4 OR $
evts.GRADE EQ 6), nfound)
  if nfound GT 0 then begin
    print, ' Keeping GRADE = 0,2,3,4,6 events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    printf, out_unit, ' Keeping GRADE = 0,2,3,4,6 events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    evts = evts(keep)
    ftime = ftime(keep)
    Etouse = Etouse(keep)
    Xtouse = Xtouse(keep)
    Ytouse = Ytouse(keep)
    nevts = n_elements(evts)
  end else begin
    print, ' There were no GRADE = 0,2,3,4,6 events, keep them all.'
    printf, out_unit, ' There were no GRADE = 0,2,3,4,6 events, keep them all.'
  end
end else begin
  print, ' No GRADE selection'
  printf, out_unit, ' No GRADE selection'
end

; Make a nominal selection on ENERGY
if TOTAL(evts_tags EQ 'ENERGY') GE 1 then begin
  keep = where( Etouse GT min_sel_energy AND $
Etouse LT max_sel_energy, nfound)
  if nfound GT 0 then begin
    print, ' Keeping ENERGY selected events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
```

```
    print, '  (from '+STRCOMPRESS(min_sel_energy)+$
' to '+STRCOMPRESS(max_sel_energy)+' keV)'
    printf, out_unit, ' Keeping ENERGY selected events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    printf, out_unit, '  (from '+STRCOMPRESS(min_sel_energy)+$
' to '+STRCOMPRESS(max_sel_energy)+' keV)'
    evts = evts(keep)
    ftime = ftime(keep)
    Etouse = Etouse(keep)
    Xtouse = Xtouse(keep)
    Ytouse = Ytouse(keep)
    nevts = n_elements(evts)
  end else begin
    print, ' There were no acceptible ENERGY events, keep them all.'
    printf, out_unit, ' There were no acceptible ENERGY events, keep them all.'
  end
end else begin
  print, ' No ENERGY selection'
  printf, out_unit, ' No ENERGY selection'
end

; For HRC, make a nominal selection on PHA
if (STRPOS(DETECTOR,'HRC') GE 0) AND (TOTAL(evts_tags EQ 'PHA') GE 1) then begin
  keep = where( evts.PHA GT min_sel_pha AND $
evts.PHA LT max_sel_pha, nfound)
  if nfound GT 0 then begin
    print, ' Keeping HRC PHA selected events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    print, '  (from '+STRCOMPRESS(min_sel_pha)+$
' to '+STRCOMPRESS(max_sel_pha)+' )'
    printf, out_unit, ' Keeping HRC PHA selected events, '+ $
STRCOMPRESS((100.0*nfound)/nevts)+' %'
    printf, out_unit, '  (from '+STRCOMPRESS(min_sel_pha)+$
' to '+STRCOMPRESS(max_sel_pha)+' )'
    evts = evts(keep)
    ftime = ftime(keep)
    Etouse = Etouse(keep)
```

```
    Xtouse = Xtouse(keep)
    Ytouse = Ytouse(keep)
    nevts = n_elements(evts)
  end else begin
    print, ' No HRC PHAs in range(?), keep them all.'
    printf, out_unit, ' No HRC PHAs in range(?), keep them all.'
  end
end else begin
  if (STRPOS(DETECTOR,'HRC') GE 0) then begin
    print, ' HRC PHA selection not performed'
    printf, out_unit, ' HRC PHA selection not performed'
  end
end

print, ''
printf, out_unit, ''

; All done selecting, make new energy_color array:
; ENERGY or PHA available?
if (TOTAL(evts_tags EQ 'ENERGY') GE 1) OR $
(TOTAL(evts_tags EQ 'PHA') GE 1) then begin
  ; Prefer to use energy if it is present
  if TOTAL(evts_tags EQ 'ENERGY') GE 1 then begin
    ; Log energy color coding...
    ; energy_colors = 3+FIX(252.* $
    ; (ALOG(Etouse) - ALOG(min_clr_energy))/ $
    ; (ALOG(max_clr_energy) - ALOG((min_clr_energy > 0.06))) )
    ; Prefer..
    ; Linear coding...
    energy_colors = 3+FIX(252.* $
(Etouse - min_clr_energy)/ $
(max_clr_energy - min_clr_energy) )
  end else begin
    ; PHA available...
    energy_colors = 3+FIX(252.* $
(evts.PHA - min_clr_pha)/ $
(max_clr_pha - min_clr_pha) )
```

```
  end
  if color24bit AND KEYWORD_SET(SCREEN24) then begin
    ; convert the colors to 24 bit...
    energy_colors = color24_values(energy_colors)
  end
end else begin
  ; No ENERGY or PHA to provide color coding...
  energy_colors = clr_grn
end

!p.multi = [0,1,3]

plot, Xtouse, Ytouse, PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='X-Y Plot of Selected Events', $
        XTITLE=x_name+' (pixels)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.81
plots, Xtouse, Ytouse, PSYM=3, COLOR=energy_colors, /CLIP
if TOTAL(evts_tags EQ 'ENERGY') GE 1 then begin
  plot_io, Xtouse, Etouse, PSYM=3, $
XSTYLE=1, YSTYLE=1, YRANGE=[min_sel_energy,max_sel_energy], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='X-E Plot of Selected Events', $
        XTITLE=x_name+' (pixels)', $
        YTITLE='Detector Energy ('+energy_unit+')', $
CHARSIZE=1.81
  plots, Xtouse, Etouse, PSYM=3, COLOR=energy_colors, /CLIP
  log_hist, Etouse, e_bin_ratio, bines, bincounts
  plot_oo, bines, bincounts, PSYM=10, $
XRANGE=[min_sel_energy,max_sel_energy], XSTYLE=1, $
YRANGE=[0.5,2.0*MAX(bincounts)], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, $
        TITLE='Detector Pulse Height Distribution of Selected Events', $
        XTITLE='Detector Energy ('+energy_unit+')', $
        YTITLE='Counts/bin', $
```

```
CHARSIZE=1.81
  print, ' Log binning from '+STRCOMPRESS(min_sel_energy)+ $
' to '+STRCOMPRESS(max_sel_energy)+' keV'
  print, ' E/dE of one bin width = '+STRCOMPRESS(1./(e_bin_ratio-1.0))
  print, ''
  printf, out_unit, ' Log binning from '+STRCOMPRESS(min_sel_energy)+ $
' to '+STRCOMPRESS(max_sel_energy)+' keV'
  printf, out_unit, ' E/dE of one bin width = '+STRCOMPRESS(1./(e_bin_ratio-1.0))
  printf, out_unit, ''
end else begin
  ; PHA plots if available and ENERGY isn't
  if (TOTAL(evts_tags EQ 'PHA')) GE 1 then begin
    plot, Xtouse, evts.PHA, PSYM=3, $
XSTYLE=1, YRANGE=[0.,MAX(evts.PHA)], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, /CLIP, $
        TITLE='X-PHA Plot of Selected Events', $
        XTITLE=x_name+' (pixels)', $
        YTITLE='Detector PHA', $
CHARSIZE=1.81
    plots, Xtouse, evts.PHA, $
PSYM=3, COLOR=energy_colors, /CLIP
    lin_hist, evts.PHA, 1.0, bines, bincounts
    plot, bines, bincounts, PSYM=10, $
XSTYLE=1, XRANGE=[0.,MAX(evts.PHA)], $
YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, $
        TITLE='Detector Pulse Height Distribution of Selected Events', $
        XTITLE='Detector PHA', $
        YTITLE='Counts/bin', $
CHARSIZE=1.81
  end
end


; Finish the .gif output
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
```

```
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_showselect.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+'Plots of Selected Events'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"

if KEYWORD_SET(MOUSE) then begin
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end

; - - - - - end of event selection - - - -

; ......................................
;    - Data and Parameters available at this stage:
; filename, output_prefix, theKEYWORDS, out_unit, iwind, clr_*, color24_values
; evts, evts_tags, energy_unit, interval_duration, ftime, time_unit
; cc_mode, pixel_size, Xtouse, x_name, Ytouse, y_name, proc_method
; ft_frac_min, ft_frac_max, max_sel_energy, min_sel_energy, e_bin_ratio,
;   min_clr_energy, max_clr_energy, min_clr_pha, max_clr_pah, energy_colors
; ......................................


; - - - - - - - - - - Timing Analysis - - - - - - - - - - -
; Parameters:
tbinsize = 0.1  ; seconds
tgapsize = 15.0 ; seconds
print, ' - - - - - - - - - - - - Timing Analysis... - - - - - - - - - '
print, ''
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
```

```
printf, out_unit, '<H3>Timing Analysis...</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

print, ' Total Events = '+STRCOMPRESS(n_elements(evts))
print, ' Start time = '+STRCOMPRESS(abs_start_time)+' '+time_unit+'s'
print, ' Interval duration = '+STRCOMPRESS(interval_duration)+' '+time_unit+'s'
print, ' Ave Event Rate = '+STRCOMPRESS(FLOAT(n_elements(evts))/ $
interval_duration)+' events/'+time_unit
print, ''

printf, out_unit, ' Selected Events = '+STRCOMPRESS(n_elements(evts))
printf, out_unit, ' Start time = '+STRCOMPRESS(abs_start_time)+' '+time_unit+'s'
printf, out_unit, ' Interval duration = '+STRCOMPRESS(interval_duration)+' '+time_unit+'s'
printf, out_unit, ' Ave Event Rate = '+STRCOMPRESS(FLOAT(n_elements(evts))/ $
interval_duration)+' events/'+time_unit
printf, out_unit, ''

; Create the time-between-events...
; The events do not need to be in time-sorted order
; so sort them for this
tsort = SORT(ftime)
sftime = ftime(tsort)
delta_ts = sftime(1:*)-sftime(0:n_elements(ftime)-2)

; Find, report and "clip" any large gaps...
large_gaps = where(delta_ts GT tgapsize, nfound)
if nfound GT 0 then begin
  tot_gap_time = TOTAL(delta_ts(large_gaps))
  live_interval = interval_duration-tot_gap_time

  print, '* Found '+STRCOMPRESS(nfound)+' big ( > '+STRCOMPRESS(tgapsize)+ $
' s) gaps, totalling '+ $
STRCOMPRESS(tot_gap_time)+' '+time_unit+'s'
  print, ''
  print, ' Live Interval = '+STRCOMPRESS(live_interval)+ $
' '+time_unit+'s'
```

```
  printf, out_unit, '* Found '+STRCOMPRESS(nfound)+' big ( > '+STRCOMPRESS(tgapsize)+ $
' s) gaps, totalling '+ $
STRCOMPRESS(tot_gap_time)+' '+time_unit+'s'
  printf, out_unit, ''
  printf, out_unit, ' Live Interval = '+STRCOMPRESS(live_interval)+ $
' '+time_unit+'s'
  print, ' Revised Event Rate = '+STRCOMPRESS(FLOAT(n_elements(evts))/ $
(live_interval) )+' events/'+time_unit
  printf, out_unit, ' Revised Event Rate = '+STRCOMPRESS(FLOAT(n_elements(evts))/ $
(live_interval) )+' events/'+time_unit

  ; "Clip" the large gaps for this plotting purpose
  delta_ts(large_gaps) = tgapsize
end else begin
  print, ' Found no big ( > '+STRCOMPRESS(tgapsize)+ $
' s) gaps in event data.'
  printf, out_unit, ' Found no big ( > '+STRCOMPRESS(tgapsize)+ $
' s) gaps in event data.'
end

print, ''
printf, out_unit, ''

; Make some timing plots
!p.multi=[0,1,4]

; PLOT Xtouse vs ftime
plot, ftime, Xtouse, PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t-X Plot of Selected Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=x_name+' (pixels)', $
CHARSIZE=1.81
plots, ftime, Xtouse, PSYM=3, COLOR=energy_colors

; PLOT Ytouse vs ftime (for ACIS-S can see dropped chips...)
```

```
plot, ftime, Ytouse, PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t-Y Plot of Selected Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.81
plots, ftime, Ytouse, PSYM=3, COLOR=energy_colors

if TOTAL(evts_tags EQ 'ENERGY') GE 1 then begin
  ; PLOT ENERGY vs ftime
  plot_io, ftime, Etouse, PSYM=3, $
XSTYLE=1, YSTYLE=1, YRANGE=[min_sel_energy,max_sel_energy], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t-E Plot of Selected Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE='Detector Energy ('+energy_unit+')', $
CHARSIZE=1.81
  plots, ftime, Etouse, PSYM=3, COLOR=energy_colors
end else begin
  if TOTAL(evts_tags EQ 'PHA') GE 1 then begin
    plot, ftime, evts.PHA, PSYM=3, $
XSTYLE=1, YRANGE=[0.,MAX(evts.PHA)], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, /CLIP, $
        TITLE='t-PHA Plot of Selected Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE='Detector PHA ', $
CHARSIZE=1.81
    plots, ftime, evts.PHA, PSYM=3, COLOR=energy_colors
  end
end

; PLOT the event-to-event time distribution
lin_hist, delta_ts, tbinsize, tbins, tcounts
plot_io, tbins, tcounts, PSYM=10, YRANGE=[0.5,2.0*MAX(tcounts)],YSTYLE=1, $
TITLE='Arrival Time Distribution of Selected Events', $
XTITLE= 't_N+1 - t_N ('+time_unit+'s)', $
```

```
YTITLE= 'Number of occurances', CHARSIZE=1.81
oplot, tbins, tcounts, PSYM=4

; Finish the .gif output
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_timing.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+'Timing Plots'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"

if KEYWORD_SET(MOUSE) then begin
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

; ......................................
; Data and Parameters available at this stage:
;  filename, grating, evts, evts_tags,
;  max_sel_energy, min_sel_energy, e_bin_ratio,
;  ftime, tsort, delta_ts, tbinsize, tgapsize, interval_duration
; ......................................

; - - - - - - - - Zero-order Determination - - - - - - - - - - -
; Find time-average location of zero-order in the selected coords
; Use all events...
; Parameters:
; Zero-order region of 250 pixels
;    = +/- 3 mm, i.e. from S2-S3 gap to nominal aim point
```

```
;    = +/- 1 arc min, less than dither pattern...
zoregsize = 6.0 / pixel_size

print, ' - - - - - - - - - - - - Determine Zero-order... - - - - - - -'
print, ''
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Determine Zero-order...</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""


!p.multi = [0,1,2]
; Plot the events ...
plot, Xtouse, Ytouse, PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='X-Y Plot of Selected Events', $
        XTITLE=x_name+' (pixels)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.21
plots, Xtouse, Ytouse, PSYM=3, COLOR=energy_colors
; Compute the Average Xtouse and Ytouse
nevts = n_elements(evts)
aveXtouse = TOTAL(Xtouse)/FLOAT(nevts)
aveYtouse = TOTAL(Ytouse)/FLOAT(nevts)
print, ' Average '+x_name+','+y_name+' gives x,y = ' + $
STRCOMPRESS(aveXtouse)+', '+STRCOMPRESS(aveYtouse)
printf, out_unit, ' Average '+x_name+','+y_name+' gives x,y = ' + $
STRCOMPRESS(aveXtouse)+', '+STRCOMPRESS(aveYtouse)

; Show the current Zero-order location
; and have operator improve the guess...
oplot, [aveXtouse], [aveYtouse], PSYM=6
if KEYWORD_SET(MOUSE) OR KEYWORD_SET(OVERRIDE_ZERO) then begin
  ; this is the first time the operator is needed to think about
  ; the click, so give 'em a beep !
```

```
  xgef_beep
  xyouts, aveXtouse, aveYtouse, '  Click Near Zero-order'
  print, ''
  print, '    !!! Click Mouse ON ZERO-ORDER to Continue !!!'
  print, ''
  cursor, x, y, 3
  print, ' Mouse clicked at x,y = '+STRCOMPRESS(x)+', '+STRCOMPRESS(y)
  printf, out_unit, ' Mouse clicked at x,y = '+STRCOMPRESS(x)+', '+STRCOMPRESS(y)
  oplot, [x],[y],PSYM=2
  aveXtouse = x
  aveYtouse = y
end

; Zoom-in on a square region around this location
; at six-times the desired zero-order region size
!p.multi=[2,2,2]
plot, Xtouse, Ytouse, PSYM=3, $
XRANGE=aveXtouse+ 6.*zoregsize*[-0.5,0.5], XSTYLE=1, $
YRANGE=aveYtouse+ 6.*zoregsize*[-0.5,0.5], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='Zero-order Region (x6)', $
        XTITLE=x_name+' (pixels)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.21
plots, Xtouse, Ytouse, PSYM=3, COLOR=energy_colors

; Improve the average by using only these events and iterating...
sel = where( ABS(Xtouse-aveXtouse) LT 0.5*6.0*zoregsize AND $
ABS(Ytouse-aveYtouse) LT 0.5*6.0*zoregsize  )
aveXtouse = TOTAL(Xtouse(sel))/FLOAT(n_elements(sel))
aveYtouse = TOTAL(Ytouse(sel))/FLOAT(n_elements(sel))
print, ' Average '+x_name+','+y_name+' gives x,y = ' + $
STRCOMPRESS(aveXtouse)+', '+STRCOMPRESS(aveYtouse)
printf, out_unit, ' Average '+x_name+','+y_name+' gives x,y = ' + $
STRCOMPRESS(aveXtouse)+', '+STRCOMPRESS(aveYtouse)

; Have operator improve the guess...
```

```
oplot, [aveXtouse], [aveYtouse], PSYM=6
if KEYWORD_SET(MOUSE) OR KEYWORD_SET(OVERRIDE_ZERO) then begin
  xyouts, aveXtouse, aveYtouse, '  Click Near Zero-order'
  print, ''
  print, '    !!! Click Mouse ON ZERO-ORDER to Continue !!!'
  print, ''
  cursor, x, y, 3
  print, ' Mouse clicked at x,y = '+STRCOMPRESS(x)+', '+STRCOMPRESS(y)
  printf, out_unit, ' Mouse clicked at x,y = '+STRCOMPRESS(x)+', '+STRCOMPRESS(y)
  oplot, [x],[y],PSYM=2
  aveXtouse = x
  aveYtouse = y
end

; Next-to-final zero-order computation at 2X region size
sel = where( ABS(Xtouse-aveXtouse) LT zoregsize AND $
ABS(Ytouse-aveYtouse) LT zoregsize  )
aveXtouse = TOTAL(Xtouse(sel))/FLOAT(n_elements(sel))
aveYtouse = TOTAL(Ytouse(sel))/FLOAT(n_elements(sel))

; and show the events and zero-order approx location
plot, Xtouse(sel), Ytouse(sel), PSYM=3, $
XRANGE=aveXtouse+ 2.0*zoregsize*[-0.5,0.5], XSTYLE=1, $
YRANGE=aveYtouse+ 2.0*zoregsize*[-0.5,0.5], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='Zero-order Region (x2)', $
        XTITLE=x_name+' (pixels)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.21
plots, Xtouse(sel), Ytouse(sel), PSYM=3, COLOR=energy_colors(sel)

; Final selection and Calculation of the Zero-order region
; (well, almost: aspect solution will change things possibly...)
zosel = where( ABS(Xtouse-aveXtouse) LT 0.5*zoregsize AND $
ABS(Ytouse-aveYtouse) LT 0.5*zoregsize  )
aveXtouse = TOTAL(Xtouse(zosel))/FLOAT(n_elements(zosel))
aveYtouse = TOTAL(Ytouse(zosel))/FLOAT(n_elements(zosel))
```

```
print, ' Average '+x_name+','+y_name+' gives x,y = ' + $
STRCOMPRESS(aveXtouse)+', '+STRCOMPRESS(aveYtouse)
print, ''
printf, out_unit, ' Average '+x_name+','+y_name+' gives x,y = ' + $
STRCOMPRESS(aveXtouse)+', '+STRCOMPRESS(aveYtouse)
printf, out_unit, ''

; plot cross-hairs at this location
oplot, zoregsize*[-0.5,0.5]+aveXtouse, [1.,1.]+aveYtouse
oplot, [1.,1.]+aveXtouse, zoregsize*[-0.5,0.5]+aveYtouse

; Not so fast!  User wants the last word...
if KEYWORD_SET(OVERRIDE_ZERO) then begin
  xyouts, aveXtouse, aveYtouse, '  *** Click on THE Zero-order ***'
  print, ''
  print, '    !!! *** Click Mouse on THE ZERO-ORDER to Continue !!!'
  print, ''
  cursor, x, y, 3
  print, ' Mouse clicked at x,y = '+STRCOMPRESS(x)+', '+STRCOMPRESS(y)
  printf, out_unit, ' Mouse clicked at x,y = '+STRCOMPRESS(x)+', '+STRCOMPRESS(y)
  oplot, [x],[y],PSYM=2
  aveXtouse = x
  aveYtouse = y
  ; and define the zero-order region w.r.t. these
  zosel = where( ABS(Xtouse-aveXtouse) LT 0.5*zoregsize AND $
ABS(Ytouse-aveYtouse) LT 0.5*zoregsize  )
end


print, '                    ==> MTA Monitor Task 5.2 <=='
print, ''
print, ' Zero-order Location '+x_name+','+y_name+' approx = '+ $
STRCOMPRESS(aveXtouse)+', '+STRCOMPRESS(aveYtouse)
print, ''

printf, out_unit, '</PRE>'
```

```
printf, out_unit, '<P align="center"><FONT COLOR="#00D000" SIZE=+1>'
printf, out_unit, ' ==> MTA Monitor Task 5.2 <== '
printf, out_unit, '</FONT></P>'
printf, out_unit, '<PRE>'

printf, out_unit, ' Zero-order Location '+x_name+','+y_name+' approx = '+ $
STRCOMPRESS(aveXtouse)+', '+STRCOMPRESS(aveYtouse)
printf, out_unit, ''

; Finish the .gif output
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_zerofind.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+ $
'Finding Zero-order Plots'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"

if KEYWORD_SET(MOUSE) then begin
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end

; Show close-up of the region...
!p.multi=[0,2,2]

printf, out_unit, ' Events in Zero-order Region = '+ $
STRCOMPRESS(n_elements(zosel))
printf, out_unit, ' Ave. Event Rate in Zero-order Region = '+ $
STRCOMPRESS(n_elements(zosel)/interval_duration)+ $
```

```
' events/'+time_unit+'s'
printf, out_unit, ''

print, ' Events in Zero-order Region = '+ $
STRCOMPRESS(n_elements(zosel))
print, ' Event Rate in Zero-order Region = '+ $
STRCOMPRESS(n_elements(zosel)/interval_duration)+ $
' events/'+time_unit+'s'
print, ''

; Plot the zero-order region
plot, Xtouse(zosel), Ytouse(zosel), PSYM=3, $
XRANGE=aveXtouse+ zoregsize*[-0.5,0.5], XSTYLE=1, $
YRANGE=aveYtouse+ zoregsize*[-0.5,0.5], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='Zero-order Region', $
        XTITLE=x_name+' (pixels)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.51
plots, Xtouse(zosel), Ytouse(zosel), PSYM=3, COLOR=energy_colors(zosel)

; plot cross-hairs at the z-o location
oplot, zoregsize*[-0.5,0.5]+aveXtouse, [1.,1.]+aveYtouse
oplot, [1.,1.]+aveXtouse, zoregsize*[-0.5,0.5]+aveYtouse
; Calculate the r-rms of these events
rs = SQRT((Xtouse(zosel)-aveXtouse)^2 + (Ytouse(zosel)-aveYtouse)^2)
rrms = SQRT(TOTAL(rs*rs)/FLOAT(n_elements(zosel)))

print, ' '+x_name+','+y_name+' RMS Radius = '+STRCOMPRESS(rrms)+' pixels'
printf, out_unit, ' '+x_name+','+y_name+' RMS Radius = '+ $
STRCOMPRESS(rrms)+' pixels'

; Plot the Dispersion axis profile ~ LRF
lin_hist, Xtouse(zosel), 1.0, xbins, xcounts
plot, xbins, xcounts, PSYM=10, $
        TITLE='Zero-order '+x_name+' Profile (~ LRF)', $
        XTITLE=x_name+' (pixels)', $
```

```
        YTITLE='Counts/bin', $
CHARSIZE=1.51
above10 = where(xcounts GE 0.10*MAX(xcounts), nabove10)
above50 = where(xcounts GE 0.50*MAX(xcounts), nabove50)

print, ' Number of '+x_name+' LRF pixels GT 10% of peak = '+ $
STRCOMPRESS(nabove10)
printf, out_unit, ' Number of '+x_name+' LRF pixels GT 10% of peak = '+ $
STRCOMPRESS(nabove10)
print, ' Number of '+x_name+' LRF pixels GT 50% of peak = '+ $
STRCOMPRESS(nabove50)
printf, out_unit, ' Number of '+x_name+' LRF pixels GT 50% of peak = '+ $
STRCOMPRESS(nabove50)

; Plot the zero-order events vs time
plot, ftime(zosel), Xtouse(zosel), PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t-X Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=x_name+' (pixels)', $
CHARSIZE=1.51
plots, ftime(zosel), Xtouse(zosel), PSYM=3, COLOR=energy_colors(zosel)

plot, ftime(zosel), Ytouse(zosel), PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t-Y Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.51
plots, ftime(zosel), Ytouse(zosel), PSYM=3, COLOR=energy_colors(zosel)

print, ''
printf, out_unit, ''

; Finish the .gif output
```

```
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_zerospatial.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+'Zero-order Region Plots'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"

if KEYWORD_SET(MOUSE) then begin
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end

; end of zero-order determine/examine
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

; .......................................
; Data and Parameters available at this stage:
;  filename, grating, evts, evts_tags,
;  max_sel_energy, min_sel_energy, e_bin_ratio,
;  ftime, tsort, delta_ts, tbinsize, tgapsize, interval_duration
;  zoregsize, zosel, aveXtouse, aveYtouse, xyshowtoo
; .......................................


; Aspect and Zero-order Examination Parameters:
; size of region to zoom in on for plots...
zo_exam_size = (3.0*0.050)/pixel_size  ; +/- 3 arc seconds
; number of sub-pixel bins to use
zo_exam_subbins = 10.0
```

```
; - - - - - - - - - Aspect determination - - - - - -
if KEYWORD_SET(ASPECT) then begin
  ; Parameters
  time_bin_size = 30.0  ; seconds

  ; Have a 1/6th total time apodization region at beginning and end
  ; of the time series - thus 2/3 of the points will have valid solution
  apod_region_time = interval_duration/6.0  ; set to 0 for no apodization

  fft_thresh = 0.01 ; fraction of peak to include in smooth output

  low_f_defn = 0.33 ; define "low" frequencies as some fraction of
                    ; the Nyquist frequency

  print, ' - - - - - - - - - - - - - Creating Aspect Solution... - - - - - - - '
  print, ''
  printf, out_unit, "</PRE>"
  printf, out_unit, '<HR>'
  printf, out_unit, '<H3>Creating Aspect Solution...</H3>'
  printf, out_unit, "<PRE>"
  printf, out_unit, ""

  ; Remove the average values from the Xtouse and Ytouse
  Xtouse = Xtouse - aveXtouse
  Ytouse = Ytouse - aveYtouse

  ; Create the average Xtouse and Ytouse values of events
  ; in each time bin...
  ; Could use Dave's reverse indices but do it brute force...
  ; Number of whole bins
  nbins = LONG( interval_duration/time_bin_size )
  binaveX = FLTARR(nbins)
  binaveY = binaveX
  binT = binaveX
  binApod = binaveX  ; array for apodization function
  binnave = INTARR(nbins)
  print, ' Aspect solution from '+x_name+','+y_name+' data...'
```

```
  print, ' Calculating '+x_name+','+y_name+' averages every '+ $
STRCOMPRESS(time_bin_size)+' '+time_unit+'s.'
  print, ' Apodizing during the first and last '+ $
STRCOMPRESS(apod_region_time)+' '+time_unit+'s of data.'
  print, ' Requires '+STRCOMPRESS(nbins)+' bins ...'

  printf, out_unit, ' Aspect solution from '+x_name+','+y_name+' data...'
  printf, out_unit, ' Calculating '+x_name+','+y_name+' averages every '+ $
STRCOMPRESS(time_bin_size)+' '+time_unit+'s.'
  printf, out_unit, ' Apodizing during the first and last '+ $
STRCOMPRESS(apod_region_time)+' '+time_unit+'s of data.'
  printf, out_unit, ' Requires '+STRCOMPRESS(nbins)+' bins ...'

  ; Go through the time bins and fill things in
  ; Use a plot to show how it's going!
  !p.multi = [0,1,5]
  plot, [0.,0.],[0., 100.0], PSYM=-3, $
XRANGE=[0., 100.0], XSTYLE=1, $
YRANGE=[-1.,1.], YSTYLE=1, $
        XTITLE='% Complete', $
        TITLE='Aspect Solution Progress Monitor', $
CHARSIZE = 2.0
  for ib=0,nbins-1 do begin
    oplot, [100.0*FLOAT(ib)/nbins],[0.0], PSYM=4
    ; this time bin...
    tb = FLOAT(ib)*time_bin_size
    te = FLOAT(ib+1)*time_bin_size
    binT(ib) = (tb+te)/2.0
    binT(ib) = (tb+te)/2.0
    ; find the zero-order events in this time region
    sel = where( ABS(Xtouse) LT 0.5*zoregsize AND $
ABS(Ytouse) LT 0.5*zoregsize AND $
ftime GT tb AND ftime LE te, nfound )
    ; found any?
    if nfound GE 1 then begin
      binaveX(ib) = TOTAL(Xtouse(sel))/FLOAT(nfound)
      binaveY(ib) = TOTAL(Ytouse(sel))/FLOAT(nfound)
```

```
      binnave(ib) = nfound
    end else begin
      ; if no events were found in the interval
      ; use 0 (relative to average) - this could be made
      ; smarter (use previous value, extrapolate, etc.)
      ; but hopefully this case will be infrequent!
      binaveX(ib) = 0.0
      binaveY(ib) = 0.0
      binnave(ib) = nfound
    end
    ; Evaluate the apodization function
    binApod(ib) = 1.0 ; default
    if binT(ib) LT apod_region_time then begin
      binApod(ib) = 0.5*(1.0 - COS(!PI*binT(ib)/apod_region_time))
    end
    if binT(ib) GT (interval_duration - apod_region_time) then begin
      binApod(ib) = 0.5*(1.0 - COS(!PI*(interval_duration-binT(ib))/apod_region_time))
    end
  end

  print, ' Min / Max number of events per time bin = '+ $
STRCOMPRESS(MIN(binnave)) + ' / '+STRCOMPRESS(MAX(binnave))
  printf, out_unit, ' Min / Max number of events per time bin = '+ $
STRCOMPRESS(MIN(binnave)) + ' / '+STRCOMPRESS(MAX(binnave))

  ; Apodize the time series and
  ; Take the transform of the average locations...
  fftx = FFT(binaveX*binApod)
  ffty = FFT(binaveY*binApod)

  ; Now just keep the "significant" FFT components
  ; to reconstruct the smoothed aspect for each axis...
  ; To reduce noise due to non-concentrated events
  ; (e.g. background, LETG cross-dispersion events, etc.)
  ; keep only the "low" frequencies
  frequs = INDGEN(nbins)
  low_frequ = ( (frequs LT low_f_defn*0.5*n_elements(fftx)) OR $
```

```
 (frequs GT (1.0-low_f_defn*0.5)*n_elements(fftx)) )
  sigx = where( (ABS(fftx) GT fft_thresh*MAX(ABS(fftx(1:*))) ) AND $
low_frequ)
  sigy = where( (ABS(ffty) GT fft_thresh*MAX(ABS(ffty(1:*))) ) AND $
low_frequ)

  fftxsm = 0.0*fftx
  fftxsm(sigx) = fftx(sigx)
  fftysm = 0.0*ffty
  fftysm(sigy) = ffty(sigy)

  ; Do the reverse transformations... and keep the real part...
  smaveX = FLOAT(FFT(fftxsm, /INVERSE))
  smaveY = FLOAT(FFT(fftysm, /INVERSE))

  print, ''
  printf, out_unit, ''

  !p.multi = [0,1,5]

  ; PLOT "Light curve", events per time bin
  plot, binT, binnave, PSYM=10, $
XSTYLE=1, YRANGE=[0.,1.1*MAX(binnave)],YSTYLE=1, $
        TITLE='Light Curve: Events per Time Interval', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE='Counts/bin', $
CHARSIZE=1.51

  ; Plot the "measured" aspect motion and the smooth fit
  plot, binT, binaveX*binApod, PSYM=-4, $
XSTYLE=1, YSTYLE=1, $
        TITLE='Average Zero-order '+x_name+' vs t; aspect solution overplotted', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE='Ave '+x_name+' (pixels)', $
CHARSIZE=1.51
  oplot, binT, smaveX, THICK=2, COLOR=clr_grn
  ; Indicate the apodization regions
```

```
  if apod_region_time GT 0.0 then begin
    oplot, apod_region_time*[1.,1.], [MIN(binaveX), MAX(binaveX)], $
LINESTYLE=2, COLOR=clr_grn
    oplot, interval_duration-apod_region_time*[1.,1.], $
[MIN(binaveX), MAX(binaveX)], $
LINESTYLE=2, COLOR=clr_grn
  end
  plot_io, ABS(fftx(0:nbins/2)), PSYM=-1, $
XSTYLE=1, YSTYLE=1, $
        TITLE='FFT of Ave '+x_name+'; selected frequ.s indicated', $
        XTITLE='Frequency Bin (0 to f_Nyquist)', $
        YTITLE='FFT Amplitude', $
CHARSIZE=1.51
  oplot, ABS(fftxsm(0:nbins/2)), PSYM=6, COLOR=clr_grn

  plot, binT, binaveY*binApod, PSYM=-4, $
XSTYLE=1, YSTYLE=1, $
        TITLE='Average Zero-order '+y_name+' vs t; aspect solution overplotted', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE='Ave '+y_name+' (pixels)', $
CHARSIZE=1.51
  oplot, binT, smaveY, THICK=2, COLOR=clr_grn
  ; Indicate the apodization regions
  if apod_region_time GT 0.0 then begin
    oplot, apod_region_time*[1.,1.], [MIN(binaveY), MAX(binaveY)], $
LINESTYLE=2, COLOR=clr_grn
    oplot, interval_duration-apod_region_time*[1.,1.], $
[MIN(binaveY), MAX(binaveY)], $
LINESTYLE=2, COLOR=clr_grn
  end
  plot_io, ABS(ffty(0:nbins/2)), PSYM=-1, $
XSTYLE=1, YSTYLE=1, $
        TITLE='FFT of Ave '+y_name+'; selected frequ.s indicated', $
        XTITLE='Frequency Bin (0 to f_Nyquist)', $
        YTITLE='FFT Amplitude', $
CHARSIZE=1.51
```

```
oplot, ABS(fftysm(0:nbins/2)), PSYM=6, COLOR=clr_grn

; Finish the .gif output
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_aspectsolve.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+'Aspect Solution'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"

; Make the aspect corrected values:
; binT is the aspect time and smaveX and smaveY are
; the locations of zero-order at that time... so
; use ftime for each event to interpolate zero-order
; for that event at that time...
print, ''
print, ' Making aspect corrected coordinates, AX,AY ...'
printf, out_unit, ''
printf, out_unit, ' Making aspect corrected coordinates, AX,AY ...'

; Subtract the aspect solution
AX = Xtouse - INTERPOL(smaveX, binT, ftime)
AY = Ytouse - INTERPOL(smaveY, binT, ftime)
ax_name = 'a'+x_name
ay_name = 'a'+y_name
; and add back the average values to AX, AY and Xtouse, Ytouse
Xtouse = Xtouse + aveXtouse
Ytouse = Ytouse + aveYtouse
AX = AX + aveXtouse
AY = AY + aveYtouse
aveAX = TOTAL(AX(zosel))/FLOAT(n_elements(zosel))
```

```
  aveAY = TOTAL(AY(zosel))/FLOAT(n_elements(zosel))

  print, ''
  printf, out_unit, ''

  if KEYWORD_SET(MOUSE) then begin
    print, '    !!! Click Mouse anywhere to Continue !!!'
    print, ''
    cursor, x, y, 3
  end

  ; - - - - - - - - - ASPECT Spatial Examination - - - - - - - - - -
  ; Now we compare Xtouse,Ytouse to AX, AY (similar to the comparison
  ; of Xtouse,Ytouse and SkyX,SkyY above.)

  !p.multi = [0,2,4]

  print, ' Before aspect correction:'
  print, ''
  printf, out_unit, ' Before aspect correction:'
  printf, out_unit, ''

  ; Zero-order Region and uncorrected corrds
  plot, Xtouse(zosel), Ytouse(zosel), PSYM=3, $
XRANGE=aveXtouse+ zoregsize*[-0.5,0.5], XSTYLE=1, $
YRANGE=aveYtouse+ zoregsize*[-0.5,0.5], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='Zero-order Region', $
        XTITLE=x_name+' (pixels)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.51
  plots, Xtouse(zosel), Ytouse(zosel), $
PSYM=3, COLOR=energy_colors(zosel)

  ; plot cross-hairs at the z-o location
  oplot, zoregsize*[-0.5,0.5]+aveXtouse, [1.,1.]+aveYtouse
  oplot, [1.,1.]+aveXtouse, zoregsize*[-0.5,0.5]+aveYtouse
```

```
  ; Calculate the r-rms of these events
  rs = SQRT((Xtouse(zosel)-aveXtouse)^2 + (Ytouse(zosel)-aveYtouse)^2)
  rrms = SQRT(TOTAL(rs*rs)/FLOAT(n_elements(zosel)))
  print, ' '+x_name+','+y_name+' RMS Radius = '+STRCOMPRESS(rrms)+' pixels'
  printf, out_unit, ' '+x_name+','+y_name+' RMS Radius = '+ $
STRCOMPRESS(rrms)+' pixels'

  ; Plot the Dispersion (Xtouse) LRF
  lin_hist, Xtouse(zosel), 1.0, xbins, xcounts
  plot, xbins, xcounts, PSYM=10, $
        TITLE='Zero-order '+x_name+' Profile (~ LRF)', $
        XTITLE=x_name+' (pixels)', $
        YTITLE='Counts/bin', $
CHARSIZE=1.51

  above10 = where(xcounts GE 0.1*MAX(xcounts), nabove10)

  print, ' Number of '+x_name+' LRF width at 10% of peak = '+ $
STRCOMPRESS(nabove10)
  printf, out_unit, ' Number of '+x_name+' LRF width at 10% of peak = '+ $
STRCOMPRESS(nabove10)

  ; Plot the events vs time
  plot, ftime(zosel), Xtouse(zosel), PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t-X Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=x_name+' (pixels)', $
CHARSIZE=1.51
  plots, ftime(zosel), Xtouse(zosel), $
PSYM=3, COLOR=energy_colors(zosel)

  plot, ftime(zosel), Ytouse(zosel), PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t-Y Plot for Zero-order Events', $
```

```
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=y_name+' (pixels)', $
CHARSIZE=1.51
  plots, ftime(zosel), Ytouse(zosel), $
PSYM=3, COLOR=energy_colors(zosel)

  ; Show the AX,AY value for these events
  plot, AX(zosel), AY(zosel), PSYM=3, $
XSTYLE=1, XRANGE = aveAX + zo_exam_size*[-1.,1.], $
YSTYLE=1, YRANGE = aveAY + zo_exam_size*[-1.,1.], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='Aspect X, Aspect Y: Zero-order Region', $
        XTITLE=ax_name+' (pixels)', $
        YTITLE=ay_name+' (pixels)', $
CHARSIZE=1.51
  plots, AX(zosel), AY(zosel), $
PSYM=3, COLOR=energy_colors(zosel)

  print, ''
  printf, out_unit, ''
  print, ' Aspect correction yields (including apodized events):'
  printf, out_unit, ' Aspect correction results in:'
  print, ''
  printf, out_unit, ''

  ; Calculate the r-rms of these events
  axyrs = SQRT((AX(zosel)-aveAX)^2 + (AY(zosel)-aveAY)^2)
  raxyrms = SQRT(TOTAL(axyrs*axyrs)/FLOAT(n_elements(zosel)))
  print, ' AX,AY RMS Radius = '+STRCOMPRESS(raxyrms)+' pixels'
  printf, out_unit, ' AX,AY RMS Radius = '+STRCOMPRESS(raxyrms)+' pixels'

  ; Plot the Dispersion axis (AX) LRF
  ; bin size is 1./zo_exam_subbins
  lin_hist, AX(zosel) - aveAX, 1.0/zo_exam_subbins, axxbins, axxcounts
  plot, axxbins, axxcounts, PSYM=10, $
        TITLE='Zero-order Aspect X Profile (~ LRF)', $
        XTITLE=ax_name+' (pixels, bin = '+ $
```

```
STRING(1.0/zo_exam_subbins, FORMAT='(F5.3)')+' pixel)', $
XRANGE=zo_exam_size*[-1.,1.], XSTYLE=1, $
        YTITLE='Counts/bin', $
CHARSIZE=1.51
  axabove10 = where(axxcounts GE 0.10*MAX(axxcounts), naxabove10)

  print, ' Width of AX LRF at 10% of peak = '+ $
STRING(naxabove10/zo_exam_subbins,FORMAT='(F6.1)')+ ' pixels'
  printf, out_unit, ' Width of AX LRF at 10% of peak = '+ $
STRING(naxabove10/zo_exam_subbins,FORMAT='(F6.1)')+ ' pixels'


  ; Plot the events vs time
  plot, ftime(zosel), AX(zosel) - aveAX, PSYM=3, $
XSTYLE=1, YSTYLE=1, YRANGE=zo_exam_size*[-1.,1.], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t - Aspect X Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=ax_name+' - '+STRING(aveAX,FORMAT='(F9.1)'), $
CHARSIZE=1.51
  plots, ftime(zosel), AX(zosel) - aveAX, $
PSYM=3, COLOR=energy_colors(zosel)
  ; Indicate the apodization regions
  if apod_region_time GT 0.0 then begin
    oplot, apod_region_time*[1.,1.], zo_exam_size*[-1.,1.], $
LINESTYLE=2, COLOR=clr_grn
    oplot, interval_duration-apod_region_time*[1.,1.], $
zo_exam_size*[-1.,1.], LINESTYLE=2, COLOR=clr_grn
  end

  plot, ftime(zosel), AY(zosel) - aveAY, PSYM=3, $
XSTYLE=1, YSTYLE=1, YRANGE=zo_exam_size*[-1.,1.], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t - Aspect Y Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=ay_name+' - '+STRING(aveAY,FORMAT='(F9.1)'), $
CHARSIZE=1.51
```

```
  plots, ftime(zosel), AY(zosel) - aveAY, $
PSYM=3, COLOR=energy_colors(zosel)
  ; Indicate the apodization regions
  if apod_region_time GT 0.0 then begin
    oplot, apod_region_time*[1.,1.], zo_exam_size*[-1.,1.], $
LINESTYLE=2, COLOR=clr_grn
    oplot, interval_duration-apod_region_time*[1.,1.], $
zo_exam_size*[-1.,1.], LINESTYLE=2, COLOR=clr_grn
  end

  print, ''
  printf, out_unit, ''

  ; Finish the .gif output
  if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
  image = tvrd()
  tvlct, red, green, blue, /GET
  gif_out = output_prefix+'_'+proc_method+'_aspectspatial.gif'
  write_gif, out_dir+'/'+gif_out, image, red, green, blue

  ; Add .gif link
  printf, out_unit, "</PRE>"
  printf, out_unit, "<UL>"
  printf, out_unit, '<LI><B><A href="'+gif_out+'">'+ $
'Zero-order Aspect Corrected'+'</A></B>'
  printf, out_unit, "</UL>"
  printf, out_unit, "<PRE>"

  if KEYWORD_SET(MOUSE) then begin
    print, '    !!! Click Mouse anywhere to Continue !!!'
    print, ''
    cursor, x, y, 3
  end
  ; end of ASPECT zero-order examination
  ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

  ; Select only the events in the un-apodized time range to pass on
```

```
  ; for futher analysis
  ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  keep = where(ftime GT apod_region_time AND $
ftime LT (interval_duration - apod_region_time) )
  evts = evts(keep)
  nevts = n_elements(evts)
  ftime = ftime(keep) - MIN(ftime(keep))
  interval_duration = MAX(ftime)
  ; assume that aspect solution is done in a time region lacking
  ; large gaps (solution would have problems I imagine!) so that:
  live_interval = interval_duration  ; is a good estimate
  ftime_start = MIN(ftime(keep))
  abs_start_time = abs_start_time + ftime_start
  Etouse = Etouse(keep)
  Xtouse = Xtouse(keep)
  Ytouse = Ytouse(keep)
  AX = AX(keep)
  AY = AY(keep)
  ; and redefine zosel using the previous average X,Y values
  ; applied to AX, AY
  zosel = where( ABS(AX-aveXtouse) LT 0.5*zoregsize AND $
ABS(AY-aveYtouse) LT 0.5*zoregsize  )
  aveAX = TOTAL(AX(zosel))/FLOAT(n_elements(zosel))
  aveAY = TOTAL(AY(zosel))/FLOAT(n_elements(zosel))

end else begin
  ; Define AX and AY in anycase, to be used in further analysis...
  AX = Xtouse
  AY = Ytouse
  ax_name = x_name
  ay_name = y_name
  aveAX = aveXtouse
  aveAY = aveYtouse
end
; end of ASPECT
; - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
print, ' - - - - - - - - Detailed Examine Zero-order Spatial... - - - - - - -'
print, ''
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Detailed Examine Zero-order Spatial...</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

; Summarize the event status here because Aspect correction can remove
; a bunch (33%) of events...

print, ' Total Events = '+STRCOMPRESS(n_elements(evts))
print, ' Start time = '+STRCOMPRESS(abs_start_time)+' '+time_unit+'s'
print, ' Interval duration = '+STRCOMPRESS(interval_duration)+' '+time_unit+'s'
print, ' Ave Event Rate = '+STRCOMPRESS(FLOAT(n_elements(evts))/ $
interval_duration)+' events/'+time_unit
print, ''
; Also, the "AX,AY" coords are going to be used for grating spectra
print, ' The spatial coordinates to be used for grating spectra are:'
print, '   AX, AY  =  '+ax_name+', '+ay_name
print, ''

printf, out_unit, ' Total Events = '+STRCOMPRESS(n_elements(evts))
printf, out_unit, ' Start time = '+STRCOMPRESS(abs_start_time)+ $
' '+time_unit+'s'
printf, out_unit, ' Interval duration = '+STRCOMPRESS(interval_duration)+ $
' '+time_unit+'s'
printf, out_unit, ' Ave Event Rate = '+STRCOMPRESS(FLOAT(n_elements(evts))/ $
interval_duration)+' events/'+time_unit
printf, out_unit, ''
printf, out_unit, ' The spatial coordinates to be used for grating spectra are:'
printf, out_unit, '   AX, AY  =  '+ax_name+', '+ay_name
printf, out_unit, ''


; Calculate the r-rms of these events
```

```
axyrs = SQRT((AX(zosel)-aveAX)^2 + (AY(zosel)-aveAY)^2)
raxyrms = SQRT(TOTAL(axyrs*axyrs)/FLOAT(n_elements(zosel)))
print, ' AX,AY RMS Radius = '+STRCOMPRESS(raxyrms)+' pixels'
printf, out_unit, ' AX,AY RMS Radius = '+STRCOMPRESS(raxyrms)+' pixels'

; Create the 1-D projection along nominal dispersion axis
lin_hist, AX(zosel) - aveAX, 1.0/zo_exam_subbins, axxbins, axxcounts

; Measure and report LRF widths
axabove10 = where(axxcounts GE 0.10*MAX(axxcounts), naxabove10)
axabove50 = where(axxcounts GE 0.50*MAX(axxcounts), naxabove50)

; Save the zero-order FWHM in pixels:
zo_fwhm_pixels = FLOAT(naxabove50)/FLOAT(zo_exam_subbins)

print, ' Width of AX LRF at 10% of peak = '+ $
STRING((1000.0*pixel_size*naxabove10)/zo_exam_subbins, $
FORMAT='(F7.1)')+ ' microns'
print, ' Width of AX LRF at 50% of peak = '+ $
STRING((1000.0*pixel_size*naxabove50)/zo_exam_subbins, $
FORMAT='(F7.1)')+ ' microns'
printf, out_unit, ' Width of AX LRF at 10% of peak = '+ $
STRING((1000.0*pixel_size*naxabove10)/zo_exam_subbins, $
FORMAT='(F7.1)')+ ' microns'
printf, out_unit, ' Width of AX LRF at 50% of peak = '+ $
STRING((1000.0*pixel_size*naxabove50)/zo_exam_subbins, $
FORMAT='(F7.1)')+ ' microns'

; If OVERRIDE_ZERO was requested then perhaps the zero-order size
; is also not accurate - ask the user to confirm or replace the
; measured FWHM above
if KEYWORD_SET(OVERRIDE_ZERO) then begin
  print, ''
  print, '     !!!  Zero-order measured FWHM is '+ $
STRCOMPRESS(1000.0*pixel_size*zo_fwhm_pixels)+' microns.'
  print, '          This value will set the grating spectra bin size.'
  print, ''
```

```
  quest_out = '          Do you want to enter a manual value (y/n): '
  change_fwhm=''
  READ, change_fwhm, PROMPT = quest_out
  if STRPOS(STRUPCASE(change_fwhm), 'Y') GE 0 then begin
    quest_out = '          Enter zero-order FWHM in microns: '
    new_fwhm=50.0 ; microns
    READ, new_fwhm, PROMPT = quest_out
    zo_fwhm_pixels = new_fwhm/(1000.0*pixel_size)
  end
  print, ''
  print, '* Zero-order FWHM manually set to: '+ $
STRCOMPRESS(1000.0*pixel_size*zo_fwhm_pixels)+' microns.'
  print, ''
  printf, out_unit, ''
  printf, out_unit, '* Zero-order FWHM manually set to: '+ $
STRCOMPRESS(1000.0*pixel_size*zo_fwhm_pixels)+' microns.'
  printf, out_unit, ''
end

; Ideally we have a point source with FWHM under 50 microns (2 ACIS
; pixels...) but incase it is extended or aspect correction is not
; good, increase the examination size to make a useful plot...
; Keep doubling the zero-order examination size in order to get
; the "axabove50" to be less than 1/3 of 2times exam_size:
; (zo_exam_size is in pixels)
while (2./3.)*zo_exam_size LT zo_fwhm_pixels do begin
  zo_exam_size = 2.*zo_exam_size
end

!p.multi=[0,2,2]

; Show the AX,AY value for these events
plot, AX(zosel), AY(zosel), PSYM=3, $
XSTYLE=1, XRANGE = aveAX + zo_exam_size*[-1.,1.], $
YSTYLE=1, YRANGE = aveAY + zo_exam_size*[-1.,1.], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='Aspect X, Aspect Y: Zero-order Region', $
```

```
        XTITLE=ax_name+' (pixels)', $
        YTITLE=ay_name+' (pixels)', $
CHARSIZE=1.51
plots, AX(zosel), AY(zosel), $
PSYM=3, COLOR=energy_colors(zosel)

; Plot the Dispersion axis (AX) LRF
plot, axxbins, axxcounts, PSYM=10, $
        TITLE='Zero-order Aspect X Profile (~ LRF)', $
        XTITLE=ax_name+' (pixels, bin = '+ $
STRING(1.0/zo_exam_subbins, FORMAT='(F5.3)')+' pixel)', $
XRANGE=zo_exam_size*[-1.,1.], XSTYLE=1, $
        YTITLE='Counts/bin', $
CHARSIZE=1.51

; Plot the events vs time
plot, ftime(zosel), AX(zosel) - aveAX, PSYM=3, $
XSTYLE=1, YSTYLE=1, YRANGE=zo_exam_size*[-1.,1.], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t - Aspect X Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=ax_name+' - '+STRING(aveAX,FORMAT='(F9.1)'), $
CHARSIZE=1.51
plots, ftime(zosel), AX(zosel) - aveAX, $
PSYM=3, COLOR=energy_colors(zosel)

plot, ftime(zosel), AY(zosel) - aveAY, PSYM=3, $
XSTYLE=1, YSTYLE=1, YRANGE=zo_exam_size*[-1.,1.], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t - Aspect Y Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE=ay_name+' - '+STRING(aveAY,FORMAT='(F9.1)'), $
CHARSIZE=1.51
plots, ftime(zosel), AY(zosel) - aveAY, $
PSYM=3, COLOR=energy_colors(zosel)

; Finish the .gif output
```

```
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_detailedzero.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+ $
'Zero-order Examination Plots'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"

if KEYWORD_SET(MOUSE) then begin
  print, ''
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end

; Compare AX, AY  to X,Y if available...
; and if COORDS_NAME is not Sky !
xyshowtoo = (TOTAL(evts_tags EQ 'X') GT 0) AND NOT(coords_name EQ 'Sky')

if xyshowtoo then begin
  !p.multi = [0,2,2]
  print, ''
  print, ' Sky X,Y (Level 1) Coordinates are present... showing them too.'
  print, ''
  printf, out_unit, ''
  printf, out_unit, ' Sky X,Y (Level 1) Coordinates are present... showing them too.'
  printf, out_unit, ''
  ; Show the X,Y value for these events
  aveX = TOTAL(evts(zosel).X)/FLOAT(n_elements(zosel))
  aveY = TOTAL(evts(zosel).Y)/FLOAT(n_elements(zosel))
  plot, evts(zosel).X, evts(zosel).Y, PSYM=3, $
```

```
XSTYLE=1, XRANGE = aveX + zo_exam_size*[-1.,1.], $
YSTYLE=1, YRANGE = aveY + zo_exam_size*[-1.,1.], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='Sky X, Sky Y (Level 1) : Zero-order Region', $
        XTITLE='Sky X (Level 1)', $
        YTITLE='Sky Y (Level 1)', $
CHARSIZE=1.51
  plots, evts(zosel).X, evts(zosel).Y, PSYM=3, COLOR=energy_colors(zosel)

  ; Calculate the r-rms of these events
  xyrs = SQRT((evts(zosel).X-aveX)^2 + (evts(zosel).Y-aveY)^2)
  rxyrms = SQRT(TOTAL(xyrs*xyrs)/FLOAT(n_elements(zosel)))
  print, ' X,Y RMS Radius = '+STRCOMPRESS(rxyrms)+' pixels'
  printf, out_unit, ' X,Y RMS Radius = '+STRCOMPRESS(rxyrms)+' pixels'

  ; Plot the Dispersion axis (X) LRF
  lin_hist, evts(zosel).X - aveX, 1.0/zo_exam_subbins, xxbins, xxcounts
  plot, xxbins, xxcounts, PSYM=10, $
        TITLE='Zero-order Sky X Profile (~ LRF)', $
        XTITLE='Sky X (pixels, bin = '+ $
STRING(1.0/zo_exam_subbins, FORMAT='(F5.3)')+' pixels)', $
XRANGE=zo_exam_size*[-1.,1.], XSTYLE=1, $
        YTITLE='Counts/bin', $
CHARSIZE=1.51
  xabove10 = where(xxcounts GE 0.10*MAX(xxcounts), nxabove10)
  xabove50 = where(xxcounts GE 0.50*MAX(xxcounts), nxabove50)
  print, ' Width of LRF at 10% of peak = '+ $
STRING((1000.0*pixel_size*nxabove10)/zo_exam_subbins, $
FORMAT='(F7.1)')+ ' microns'
  print, ' Width of LRF at 50% of peak = '+ $
STRING((1000.0*pixel_size*nxabove50)/zo_exam_subbins, $
FORMAT='(F7.1)')+ ' microns'
  printf, out_unit, ' Width of LRF at 10% of peak = '+ $
STRING((1000.0*pixel_size*nxabove10)/zo_exam_subbins, $
FORMAT='(F7.1)')+ ' microns'
  printf, out_unit, ' Width of LRF at 50% of peak = '+ $
STRING((1000.0*pixel_size*nxabove50)/zo_exam_subbins, $
```

```
FORMAT='(F7.1)')+ ' microns'

  ; Plot the events vs time
  plot, ftime(zosel), evts(zosel).X - aveX, PSYM=3, $
YRANGE = zo_exam_size*[-1.,1.], XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t - Sky X Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE='Sky X - '+STRING(aveX,FORMAT='(F9.1)'), $
CHARSIZE=1.51
  plots, ftime(zosel), evts(zosel).X - aveX, PSYM=3, COLOR=energy_colors(zosel)

  plot, ftime(zosel), evts(zosel).Y - aveY, PSYM=3, $
XSTYLE=1, YSTYLE=1, YRANGE=zo_exam_size*[-1.,1.0], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='t - Sky Y Plot for Zero-order Events', $
        XTITLE='Time ('+time_unit+'s)', $
        YTITLE='Sky Y - '+STRING(aveY,FORMAT='(F9.1)'), $
CHARSIZE=1.51
  plots, ftime(zosel), evts(zosel).Y - aveY, PSYM=3, COLOR=energy_colors(zosel)

  ; Finish the .gif output
  if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
  image = tvrd()
  tvlct, red, green, blue, /GET
  gif_out = output_prefix+'_'+proc_method+'_skyxyzero.gif'
  write_gif, out_dir+'/'+gif_out, image, red, green, blue

  ; Add .gif link
  printf, out_unit, "</PRE>"
  printf, out_unit, "<UL>"
  printf, out_unit, '<LI><B><A href="'+gif_out+'">'+ $
'Sky X,Y Plots for Comparison'+'</A></B>'
  printf, out_unit, "</UL>"
  printf, out_unit, "<PRE>"

  if KEYWORD_SET(MOUSE) then begin
```

```
    print, ''
    print, '    !!! Click Mouse anywhere to Continue !!!'
    print, ''
    cursor, x, y, 3
  end
  print, ''
  printf, out_unit, ''
end else begin
  ; any message
  if (TOTAL(evts_tags EQ 'X') EQ 0) then begin
    print, ''
    print, ' ( No X,Y coord.s to compare this to. )'
    print, ''
    printf, out_unit, ''
    printf, out_unit, ' ( No X,Y coord.s to compare this to. )'
    printf, out_unit, ''
  end
end


; end of detailed zero-order determine/examine
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


; .......................................
; Data and Parameters available at this stage:
;  filename, grating, evts, evts_tags,
;  max_sel_energy, min_sel_energy, e_bin_ratio,
;  ftime, tsort, delta_ts, tbinsize, tgapsize, interval_duration
;  zoregsize, zosel, aveXtouse, aveYtouse, xyshowtoo,
;  ASPECT: time_bin_size, fft_thresh, binT, binaveX, binaveY, binnave,
;         smaveX, smaveY, AX, AY, aveAX, aveAY
; .......................................

; - - - - - - - - - - - ANGLES - - - - - - -
; This section forms selections of the events by sign and grating.
; Angles in detector space can then be measured for the selected
```

```
; events.  Note that for CC mode events are assigned to HEG and
; MEG - hopefully ENERGY will be present to separate them.
;
; Parameters:
; expected angles:
ang_meg = 4.75  ; degrees
ang_heg = -5.18 ; degrees

print, ' - - - - - - - - - - - - Measure Grating Angle(s)... - - - - - - - '
print, ''
print, '                    ==> MTA Monitor Task 5.5 <=='
print, ''

printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Measure Grating Angle(s)...</H3>'
printf, out_unit, '<P align="center"><FONT COLOR="#00D000" SIZE=+1>'
printf, out_unit, ' ==> MTA Monitor Task 5.5 <== '
printf, out_unit, '</FONT></P>'
printf, out_unit, '<PRE>'

; Go through the Quadrants...
if GRATING EQ 'LETG' then begin
  ; setup two plots
  !p.multi=[0,1,3]
  ; set quadrants
  iqstart = 6
  iqstop = 8
end else begin
  ; set quadrants
  iqstart = 0
  iqstop = 5
  ; setup four plots
  !p.multi=[0,2,3]
end

qnames = ['HEG-minus', 'MEG-plus', 'MEG-minus', 'HEG-plus', $
```

```
'HEG-all', 'MEG-all', $
'LEG-minus', 'LEG-plus', 'LEG-all']
qangles = FLTARR(9)

for iq=iqstart,iqstop do begin
  CASE iq OF
    0: begin
         if cc_mode then begin
           sel = where(Xtouse LT aveXtouse - zoregsize/2.0)
         end else begin
           sel = where(Xtouse LT aveXtouse - zoregsize/2.0 AND $
Ytouse GT aveYtouse)
         end
         hegmsel = sel
       end
    1: begin
         if cc_mode then begin
           sel = where(Xtouse GT aveXtouse + zoregsize/2.0)
         end else begin
           sel = where(Xtouse GT aveXtouse + zoregsize/2.0 AND $
Ytouse GT aveYtouse)
         end
         megpsel = sel
       end
    2: begin
         if cc_mode then begin
           sel = where(Xtouse LT aveXtouse - zoregsize/2.0)
         end else begin
           sel = where(Xtouse LT aveXtouse - zoregsize/2.0 AND $
Ytouse LT aveYtouse)
         end
         megmsel = sel
       end
    3: begin
         if cc_mode then begin
           sel = where(Xtouse GT aveXtouse + zoregsize/2.0)
         end else begin
```

```
          sel = where(Xtouse GT aveXtouse + zoregsize/2.0 AND $
Ytouse LT aveYtouse)
         end
         hegpsel = sel
       end
    4: begin
         sel = [hegmsel,hegpsel]
         hegallsel = sel
       end
    5: begin
         sel = [megmsel,megpsel]
         megallsel = sel
       end
    6: begin
         sel = where(Xtouse LT aveXtouse - zoregsize/2.0)
         legmsel = sel
       end
    7: begin
         sel = where(Xtouse GT aveXtouse + zoregsize/2.0)
         legpsel = sel
       end
    8: begin
         sel = [legmsel,legpsel]
         legallsel = sel
       end
  ELSE: begin
        end
  ENDCASE

  if NOT(cc_mode) then begin
    ; Select the Quadrant Events
    ; Use AX, and AY
    qxs = AX(sel)
    qys = AY(sel)

    plot, qxs, qys, PSYM=3, $
XRANGE=[MIN(qxs), MAX(qxs)], XSTYLE=1, $
```

```
YRANGE=[MIN(qys), MAX(qys)], YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE = 'Measuring Angle for '+qnames(iq)+' Events', $
        XTITLE=ax_name+' (pixels)', $
        YTITLE=ay_name+' (pixels)', $
CHARSIZE=1.51
    plots, qxs, qys, PSYM=3, COLOR=energy_colors

    ; Try to fit these...!?!
    fit_result = linfit(qxs, qys)
    oplot, [min(qxs), max(qxs)], $
fit_result(0)+fit_result(1)*[min(qxs), max(qxs)]

    ; Report the slope
    slope = fit_result(1)
    angle = ATAN(slope)/!DTOR
    qangles(iq) = angle
    print, ' '+qnames(iq)+' slope is '+STRCOMPRESS(slope)+', --> '+ $
STRCOMPRESS(angle)+' degrees on detector'
    printf, out_unit, ' '+qnames(iq)+' slope is '+STRCOMPRESS(slope)+', --> '+ $
STRCOMPRESS(angle)+' degrees on detector'
  end
end

if cc_mode then begin
  print, ''
  printf, out_unit, ''
  print, " CC mode data: can't measure the angles."
  printf, out_unit, " CC mode data: can't measure the angles."
  print, ''
  printf, out_unit, ''
end else begin
  print, ''
  printf, out_unit, ''

  ; Finish the .gif output
  if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
```

```
  image = tvrd()
  tvlct, red, green, blue, /GET
  gif_out = output_prefix+'_'+proc_method+'_angles.gif'
  write_gif, out_dir+'/'+gif_out, image, red, green, blue

  ; Add .gif link
  printf, out_unit, "</PRE>"
  printf, out_unit, "<UL>"
  printf, out_unit, '<LI><B><A href="'+gif_out+'">'+ $
'Grating Angle Analysis Plots'+'</A></B>'
  printf, out_unit, "</UL>"
  printf, out_unit, "<PRE>"

  if KEYWORD_SET(MOUSE) then begin
    print, '    !!! Click Mouse anywhere to Continue !!!'
    print, ''
    cursor, x, y, 3
  end
end

; end of selection and angles
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

; .......................................
; Data and Parameters available at this stage:
;  filename, grating, evts, evts_tags, max_sel_energy, min_sel_energy, e_bin_ratio,
;  ftime, tsort, delta_ts, tbinsize, tgapsize, interval_duration
;  zoregsize, zosel, aveXtouse, aveYtouse,
;  ASPECT: time_bin_size, fft_thresh, binT, binaveX, binaveY, binnave,
;          smaveX, smaveY, AX, AY, aveAX, aveAY
;  qnames, qangles, [heg|meg][m|p|all]sel
; .......................................

; - - - - - - - - - - Crude Dispersed Spectra - - - - - - -
; Parameters:
; Rowland spacings
if KEYWORD_SET(XRCF_ROWLAND) then begin
```

```
  if GRATING EQ 'HETG' then begin
    ; HETG XRCF value:
    hetg_rs = 8782.80 ;mm
  end else begin
    hetg_rs = 8788.04  ;mm, LETG spacing at XRCF
  end
end else begin
  ; FLIGHT value:
  hetg_rs = 8634.0 ;mm, for both gratings
end
p_meg = 4001.41
p_heg = 2000.81
p_leg = 9912.16
hc = 12.3985

ge_min = 0.4 ; keV
; Go a bit lower if LETG and ACIS:
if GRATING EQ 'LETG' AND (STRPOS(DETECTOR,'ACIS') GE 0) then ge_min=0.20
; Go even lower if LETG and HRC:
if GRATING EQ 'LETG' AND (STRPOS(DETECTOR,'HRC') GE 0) then ge_min=0.05
ge_max = 10.0 ; keV

; Define order-selection bands in PHA-grating_energy space
order_sel_accuracy = 0.25  ; PHA can be from 0.75 to 1.25 of grating E

; Bin size: E_n+1/E_n at 1 keV:
; This has E/dE = 2000. at 1 keV, reasonable for HEG and MEG point
; sources...
ge_bin_ratio = 1.0005
; If the zero-order FWHM is larger than 50 microns FWHM, then
; increase this bin size:
ge_bin_ratio = ge_bin_ratio ^ ( FIX(zo_fwhm_pixels*pixel_size/0.05) > 1)
if GRATING EQ 'LETG' then ge_bin_ratio = 1.0015  ; 3x coarser

print, ' - - - - - - - - - - - - - Crude Dispersed Spectra... - - - - - - - '
print, ''
printf, out_unit, "</PRE>"
```

```
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Crude Dispersed Spectra...</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

print, ''
print, ' Rowland spacing used is '+ $
STRING(hetg_rs,FORMAT='(F7.1)')+' mm'
print, ''

printf, out_unit, ''
printf, out_unit, ' Rowland spacing used is '+ $
STRING(hetg_rs,FORMAT='(F7.1)')+' mm'
printf, out_unit, ''

; For LETG use the meg arrays and calculations by setting
;  megm=legm, and p_meg = p_leg ...
; Provide a string to flag this:
megleg = 'meg'
if GRATING EQ 'LETG' then begin
  p_meg = p_leg
  megpsel = legpsel
  megmsel = legmsel
  megleg = 'leg'
end

; Dispersion distance using AX, AY
d_megp = pixel_size*SQRT( (AX(megpsel) - aveAX)^2 + (AY(megpsel) - aveAY)^2)
d_megm = pixel_size*SQRT( (AX(megmsel) - aveAX)^2 + (AY(megmsel) - aveAY)^2)
if GRATING EQ 'HETG' then begin
  d_hegp = pixel_size*SQRT( (AX(hegpsel) - aveAX)^2 + (AY(hegpsel) - aveAY)^2)
  d_hegm = pixel_size*SQRT( (AX(hegmsel) - aveAX)^2 + (AY(hegmsel) - aveAY)^2)
end

; Modify these distances if it is CC mode data with the HETG
if cc_mode and (GRATING EQ 'HETG') then begin
  d_megp = d_megp / COS(ang_meg*!DTOR)
```

```
  d_megm = d_megm / COS(ang_meg*!DTOR)
  d_hegp = d_hegp / COS(ang_heg*!DTOR)
  d_hegm = d_hegm / COS(ang_heg*!DTOR)
end

; Simple diffraction equation for 1st order
e_megp = hc/(p_meg*d_megp/hetg_rs)
e_megm = hc/(p_meg*d_megm/hetg_rs)
if GRATING EQ 'HETG' then begin
  e_hegp = hc/(p_heg*d_hegp/hetg_rs)
  e_hegm = hc/(p_heg*d_hegm/hetg_rs)
end

; Select events whose diffracted energy matches PHA energy
; if ENERGY is available AND it is not HRC!
if (TOTAL(evts_tags EQ 'ENERGY') GE 1) AND $
NOT(STRPOS(DETECTOR,'HRC') GE 0) then begin
  print, ' Use ACIS ENERGY for Order selection...'
  printf, out_unit, ' Use ACIS ENERGY for Order selection...'
  ; Create plots here to show order selection...
  !p.multi = [0,1,2]  ; leg/meg +/-

  ; Will overplot the selection criteria lines - make up the data points
  order_plt_es = 12.0*(indgen(101)/100.0) ; 0-12 keV

  if GRATING EQ 'HETG' then !p.multi = [0,1,4]

  m1_megm = where(approx_equal(Etouse(megmsel)/e_megm,1.0, $
order_sel_accuracy))
  plot_oi, e_megm(m1_megm), Etouse(megmsel(m1_megm)), PSYM=3, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.0, MAX(Etouse(megmsel(m1_megm)))], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
XTITLE=STRUPCASE(megleg)+'-minus Energy (keV)', $
YTITLE='ACIS Energy (keV)', $
CHARSIZE=1.51
  plots, e_megm(m1_megm), Etouse(megmsel(m1_megm)), PSYM=3, $
```

```
COLOR=energy_colors
  oplot, order_plt_es, COLOR=clr_blu, $
(1.0+order_sel_accuracy)*order_plt_es, LINESTYLE=2
  oplot, order_plt_es, COLOR=clr_blu, $
(1.0-order_sel_accuracy)*order_plt_es, LINESTYLE=2

  m1_megp = where(approx_equal(Etouse(megpsel)/e_megp,1.0, $
order_sel_accuracy))
  plot_oi, e_megp(m1_megp), Etouse(megpsel(m1_megp)), PSYM=3, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.0, MAX(Etouse(megpsel(m1_megp)))], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE = 'Grating-E - ACIS-E : Order Selection Plots', $
XTITLE=STRUPCASE(megleg)+'-plus Energy (keV)', $
YTITLE='ACIS Energy (keV)', $
CHARSIZE=1.51
  plots, e_megp(m1_megp), Etouse(megpsel(m1_megp)), PSYM=3, $
COLOR=energy_colors
  oplot, order_plt_es, COLOR=clr_blu, $
(1.0+order_sel_accuracy)*order_plt_es, LINESTYLE=2
  oplot, order_plt_es, COLOR=clr_blu, $
(1.0-order_sel_accuracy)*order_plt_es, LINESTYLE=2


  print, ' '+STRUPCASE(megleg)+' minus order selection keeps '+ $
STRCOMPRESS((100.0*n_elements(m1_megm))/n_elements(megmsel))+' %'+$
',  '+STRCOMPRESS(n_elements(m1_megm))+' events'

  print, ' '+STRUPCASE(megleg)+' plus  order selection keeps '+ $
STRCOMPRESS((100.0*n_elements(m1_megp))/n_elements(megpsel))+' %'+$
',  '+STRCOMPRESS(n_elements(m1_megp))+' events'

  printf, out_unit, ' '+STRUPCASE(megleg)+' minus order selection keeps '+ $
STRCOMPRESS((100.0*n_elements(m1_megm))/n_elements(megmsel))+' %'+$
',  '+STRCOMPRESS(n_elements(m1_megm))+' events'

  printf, out_unit, ' '+STRUPCASE(megleg)+' plus  order selection keeps '+ $
```

```
STRCOMPRESS((100.0*n_elements(m1_megp))/n_elements(megpsel))+' %'+$
',  '+STRCOMPRESS(n_elements(m1_megp))+' events'

  if GRATING EQ 'HETG' then begin

    m1_hegm = where(approx_equal(Etouse(hegmsel)/e_hegm,1.0, $
order_sel_accuracy))
    plot_oi, e_hegm(m1_hegm), Etouse(hegmsel(m1_hegm)), PSYM=3, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.0, MAX(Etouse(hegmsel(m1_hegm)))], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
XTITLE='HEG-minus Energy (keV)', $
YTITLE='ACIS Energy (keV)', $
CHARSIZE=1.51
    plots, e_hegm(m1_hegm), Etouse(hegmsel(m1_hegm)), PSYM=3, $
COLOR=energy_colors
    oplot, order_plt_es, COLOR=clr_blu, $
(1.0+order_sel_accuracy)*order_plt_es, LINESTYLE=2
    oplot, order_plt_es, COLOR=clr_blu, $
(1.0-order_sel_accuracy)*order_plt_es, LINESTYLE=2

    m1_hegp = where(approx_equal(Etouse(hegpsel)/e_hegp,1.0, $
order_sel_accuracy))
    plot_oi, e_hegp(m1_hegp), Etouse(hegpsel(m1_hegp)), PSYM=3, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.0, MAX(Etouse(hegpsel(m1_hegp)))], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
XTITLE='HEG-plus Energy (keV)', $
YTITLE='ACIS Energy (keV)', $
CHARSIZE=1.51
    plots, e_hegp(m1_hegp), Etouse(hegpsel(m1_hegp)), PSYM=3, $
COLOR=energy_colors
    oplot, order_plt_es, COLOR=clr_blu, $
(1.0+order_sel_accuracy)*order_plt_es, LINESTYLE=2
    oplot, order_plt_es, COLOR=clr_blu, $
(1.0-order_sel_accuracy)*order_plt_es, LINESTYLE=2
```

```
    print, ' HEG minus order selection keeps '+ $
STRCOMPRESS((100.0*n_elements(m1_hegm))/n_elements(hegmsel))+' %'+$
',  '+STRCOMPRESS(n_elements(m1_hegm))+' events'

    print, ' HEG plus  order selection keeps '+ $
STRCOMPRESS((100.0*n_elements(m1_hegp))/n_elements(hegpsel))+' %'+$
',  '+STRCOMPRESS(n_elements(m1_hegp))+' events'

    printf, out_unit, ' HEG minus order selection keeps '+ $
STRCOMPRESS((100.0*n_elements(m1_hegm))/n_elements(hegmsel))+' %'+$
',  '+STRCOMPRESS(n_elements(m1_hegm))+' events'

    printf, out_unit, ' HEG plus  order selection keeps '+ $
STRCOMPRESS((100.0*n_elements(m1_hegp))/n_elements(hegpsel))+' %'+$
',  '+STRCOMPRESS(n_elements(m1_hegp))+' events'

  end
  print, ''
  printf, out_unit, ''

  ; Finish the .gif output
  if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
  image = tvrd()
  tvlct, red, green, blue, /GET
  gif_out = output_prefix+'_'+proc_method+'_orderselect.gif'
  write_gif, out_dir+'/'+gif_out, image, red, green, blue

  ; Add .gif link
  printf, out_unit, "</PRE>"
  printf, out_unit, "<UL>"
  printf, out_unit, '<LI><B><A href="'+gif_out+'">'+ $
'Order Selection Plots'+'</A></B>'
  printf, out_unit, "</UL>"
  printf, out_unit, "<PRE>"

  if KEYWORD_SET(MOUSE) then begin
```

```
    print, '    !!! Click Mouse anywhere to Continue !!!'
    print, ''
    cursor, x, y, 3
  end

  em1_megp = e_megp(m1_megp)
  em1_megm = e_megm(m1_megm)
  if GRATING EQ 'HETG' then begin
    em1_hegp = e_hegp(m1_hegp)
    em1_hegm = e_hegm(m1_hegm)
  end
end else begin
  ; No ENERGY, just keep em all...
  em1_megp = e_megp
  em1_megm = e_megm
  if GRATING EQ 'HETG' then begin
    em1_hegp = e_hegp
    em1_hegm = e_hegm
  end
  print, ' ACIS ENERGY not available: keeping all orders'
  print, ''
  printf, out_unit, ' ACIS ENERGY not available: keeping all orders'
  printf, out_unit, ''
end

print, ' Constant d-Lambda binning from '+STRCOMPRESS(ge_min)+' to '+STRCOMPRESS(ge_max)+$
' keV'
print, ''
print, ' E/dE of one bin width at 1 keV = '+STRCOMPRESS(1./(ge_bin_ratio-1.0))
print, ''

printf, out_unit, ' Constant d-Lambda binning from '+STRCOMPRESS(ge_min)+' to '+STRCOMPRESS(ge_max)+$
' keV'
printf, out_unit, ''
printf, out_unit, ' E/dE of one bin width at 1 keV = '+STRCOMPRESS(1./(ge_bin_ratio-1.0))
printf, out_unit, ''
```

```
; Here is the grating binning...
; Going to be dragged kicking and screaming into wavelength land
; by instrument (blur ~ constant in lambda) and ISIS very soon...
; These histograms are created here for plotting purposes, they are
; written out to rdb and ISIS data files later in code.

; Do a binning in Lambda with E/dE at 1 keV (= hc A) of :
;    1.0/(ge_bin_ratio -1.0) :
lin_hist, hc/em1_megp, (hc)*(ge_bin_ratio-1.0), mpbins, mpcounts
mpbins = hc/mpbins
binsort = SORT(mpbins)
mpbins = mpbins(binsort)
mpcounts = mpcounts(binsort)

lin_hist, hc/em1_megm, (hc)*(ge_bin_ratio-1.0), mmbins, mmcounts
mmbins = hc/mmbins
binsort = SORT(mmbins)
mmbins = mmbins(binsort)
mmcounts = mmcounts(binsort)


if GRATING EQ 'HETG' then begin
  lin_hist, hc/em1_hegp, (hc)*(ge_bin_ratio-1.0), hpbins, hpcounts
  hpbins = hc/hpbins
  binsort = SORT(hpbins)
  hpbins = hpbins(binsort)
  hpcounts = hpcounts(binsort)

  lin_hist, hc/em1_hegm, (hc)*(ge_bin_ratio-1.0), hmbins, hmcounts
  hmbins = hc/hmbins
  binsort = SORT(hmbins)
  hmbins = hmbins(binsort)
  hmcounts = hmcounts(binsort)
end

if GRATING EQ 'LETG' then begin
  !p.multi=[0,1,2]
```

```
  plot_oo, mmbins, mmcounts, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.5, 2.0*MAX(mmcounts)], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='LETG Spectra', $
XTITLE='LEG-minus Energy (keV)', $
YTITLE='Counts/bin', $
CHARSIZE=1.51
  oplot, mmbins, mmcounts, PSYM=10, COLOR=clr_yel
  plot_oo, mpbins, mpcounts, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.5, 2.0*MAX(mmcounts)], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
XTITLE='LEG-plus Energy (keV)', $
YTITLE='Counts/bin', $
CHARSIZE=1.51
  oplot, mpbins, mpcounts, PSYM=10, COLOR=clr_yel
end else begin
  !p.multi=[0,1,4]
  plot_oo, mmbins, mmcounts, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.5, 2.0*MAX(mmcounts)], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
        TITLE='HETG Spectra', $
XTITLE='MEG-minus Energy (keV)', $
YTITLE='Counts/bin', $
CHARSIZE=1.51
  oplot, mmbins, mmcounts, PSYM=10, COLOR=clr_yel
  plot_oo, mpbins, mpcounts, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.5, 2.0*MAX(mmcounts)], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
XTITLE='MEG-plus Energy (keV)', $
YTITLE='Counts/bin', $
CHARSIZE=1.51
  oplot, mpbins, mpcounts, PSYM=10, COLOR=clr_yel
  plot_oo, hpbins, hpcounts, PSYM=10, $
```

```
   XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.5, 2.0*MAX(mmcounts)], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
XTITLE='HEG-minus Energy (keV)', $
YTITLE='Counts/bin', $
CHARSIZE=1.51
  oplot, hpbins, hpcounts, PSYM=10, COLOR=clr_blu
  plot_oo, hmbins, hmcounts, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE=[0.5, 2.0*MAX(mmcounts)], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
XTITLE='HEG-plus Energy (keV)', $
YTITLE='Counts/bin', $
CHARSIZE=1.51
  oplot, hmbins, hmcounts, PSYM=10, COLOR=clr_blu
end

; Finish the .gif output
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_gspectra.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+$
'Grating Spectra Plots'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"

if KEYWORD_SET(MOUSE) then begin
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end
```

```
; end of crude disp spectra...
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

; .......................................
; Data and Parameters available at this stage:
;  filename, grating, evts, evts_tags,
;  max_sel_energy, min_sel_energy, e_bin_ratio,
;  ftime, tsort, delta_ts, tbinsize, tgapsize, interval_duration
;  zoregsize, zosel, aveXtouse, aveYtouse,
;  ASPECT: time_bin_size, fft_thresh, binT, binaveX, binaveY, binnave,
;          smaveX, smaveY, AX, AY, aveAX, aveAY
;  qnames, qangles, [heg|meg][m|p|all]sel
;  pixel_size, hetg_rs, p_meg, p_heg, hc, ge_min, ge_max, ge_bin_ratio
;  megleg, em1_[heg|meg][p|m], [h|m][m|p]bins, [h|m][m|p]counts
; .......................................


; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; Check Zero-order spectra (if available) with dispersed spectra
;   Parameters:
; smoothing kernel to apply to ACIS and Grating spectra...
; A rectangular kernel is applied three times to produce
; a parabolic line response, FWHM of this is ~1.23 times the
; kernel length.

compare_EdE = 40.0  ; want to smooth the spectra to this level
                    ; to blur grating and ACIS spectra to similar level.
; The grating spectra are binned to ge_bin_ratio - how many of these
; bins is compare_EdE? :
compare_bins = ALOG(1. + 1./compare_EdE)/ALOG(ge_bin_ratio)
; empirical factor to convert kernel length to effective FWHM:
len2fwhm = 1.23
; needed k_len is then
k_len = FIX(compare_bins/len2fwhm > 3) ; keep 3 bins at least

print, ' - - - - - - - - - - - - Compare Zero-order and Dispersed spectra - - - - - - '
print, ''
```

```
print, '                   ==> MTA Monitor Task 5.7 <=='
print, ''

printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Compare Zero-order and Dispersed Spectra...</H3>'
printf, out_unit, '<P align="center"><FONT COLOR="#00D000" SIZE=+1>'
printf, out_unit, ' ==> MTA Monitor Task 5.7 <== '
printf, out_unit, '</FONT></P>'
printf, out_unit, '<PRE>'

; Get the predicted effective areas for:
;  HETG/LETG zero-order
;  MEG/LEG combined 1st orders
;  HEG combined 1st orders
; "AO-1" versions of these are in:
; tbl_dir = '/nfs/atum/d3/dd/dd_d6/CIP/Tbl'
; Put the tables in fcp area:
tbl_dir = !DDHETGCAL+'/fcp/Tbl_AO1'
; and file names are:
if STRPOS(DETECTOR,'HRC') GE 0 then begin
  if STRPOS(DETECTOR,'-S') GE 0 then begin
    hetg0file = 'EA_'+GRATING+'0-HS.rdb'
    meg1file = 'EA_MEG1-HS.rdb'
    heg1file = 'EA_HEG1-HS.rdb'
    leg1file = 'EA_LETG1-HS.rdb'
  end else begin
    hetg0file = 'EA_'+GRATING+'0-HI.rdb'
    meg1file = 'EA_MEG1-HI.rdb'
    heg1file = 'EA_HEG1-HI.rdb'
    leg1file = 'EA_LETG1-HI.rdb'
  end
end else begin
  hetg0file = 'EA_'+GRATING+'0-AS.rdb'
  meg1file = 'EA_MEG1-AS.rdb'
  heg1file = 'EA_HEG1-AS.rdb'
  leg1file = 'EA_LETG1-AS.rdb'
```

```
end

if GRATING EQ 'LETG' then meg1file = leg1file
; TAGs are energy and aeff
;
eahetg0 = rdb_read(tbl_dir+'/'+hetg0file)
eameg1 = rdb_read(tbl_dir+'/'+meg1file)
if GRATING EQ 'HETG' then begin
  eaheg1 = rdb_read(tbl_dir+'/'+heg1file)
end

; Create the spectra
; The counts/(s bin) is converted to photons/cm^2 s bin
zoeavail = (TOTAL(evts_tags EQ 'ENERGY') GE 1) AND $
NOT(STRPOS(DETECTOR,'HRC') GE 0)

if zoeavail then begin
  log_hist, Etouse(zosel), ge_bin_ratio, zobins, zocounts
end
log_hist, [em1_megp,em1_megm], ge_bin_ratio, megbins, megcounts
if GRATING EQ 'HETG' then begin
  log_hist, [em1_hegp,em1_hegm], ge_bin_ratio, hegbins, hegcounts
end
if zoeavail then begin
  zoarea = INTERPOL(eahetg0.aeff, eahetg0.energy, zobins)
  zospect = (zocounts/interval_duration)/(zoarea > 0.01)
end
megarea = INTERPOL(eameg1.aeff, eameg1.energy, megbins)
megspect = (megcounts/interval_duration)/(megarea > 0.01)
if GRATING EQ 'HETG' then begin
  hegarea = INTERPOL(eaheg1.aeff, eaheg1.energy, hegbins)
  hegspect = (hegcounts/interval_duration)/(hegarea > 0.01)
end

k_fwhm = k_len*len2fwhm
k_ede = 1.0/((ge_bin_ratio^k_fwhm)-1.0)
print, ' Smoothing the spectra to E/dE ~ '+STRCOMPRESS(k_ede)
```

```
print, ''
printf, out_unit, ' Smoothing the spectra by E/dE ~ '+STRCOMPRESS(k_ede)
printf, out_unit, ''

; Convolve with rectangle function three times (make parabola LRF)
kernel = (1.0+FLTARR(k_len))/FLOAT(k_len)
if zoeavail then begin
  czospect = CONVOL(zospect, kernel)
  czospect = CONVOL(czospect, kernel)
  czospect = CONVOL(czospect, kernel)
end
cmegspect = CONVOL(megspect, kernel)
cmegspect = CONVOL(cmegspect, kernel)
cmegspect = CONVOL(cmegspect, kernel)
if GRATING EQ 'HETG' then begin
  chegspect = CONVOL(hegspect, kernel)
  chegspect = CONVOL(chegspect, kernel)
  chegspect = CONVOL(chegspect, kernel)
end

nplots = 4
if GRATING EQ 'LETG' then nplots = nplots-1
if NOT(zoeavail) then nplots = nplots-1
!p.multi=[0,1,nplots]

if zoeavail then begin
  zovis = where(zobins GT ge_min AND zobins LT ge_max)
end
megvis = where(megbins GT ge_min AND megbins LT ge_max)

; Use the same Y range for the spectra
if zoeavail then begin
  max_spec_range = 4.0*MAX(czospect(zovis))
end else begin
  max_spec_range = 4.0*MAX(cmegspect(megvis))
end
spec_range = max_spec_range*[1.E-3,1.0]
```

```
if zoeavail then begin
  plot_oo, zobins, czospect, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE= spec_range, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE='ACIS Zero-order Spectrum: Raw and Smoothed', $
XTITLE='Energy (keV)', $
YTITLE='Photons / cm^2 s bin', $
CHARSIZE = 1.51
; The counts/(s bin) is converted to photons/cm^2 s bin
  oplot, zobins, czospect, PSYM=10, COLOR=clr_org
  oplot, zobins, zospect, LINESTYLE=1, COLOR=clr_org
end
plot_oo, megbins, cmegspect, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE= spec_range, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE=STRUPCASE(megleg)+' Spectrum: Raw and Smoothed', $
XTITLE='Energy (keV)', $
YTITLE='Photons / cm^2 s bin', $
CHARSIZE = 1.51
oplot, megbins, cmegspect, PSYM=10, COLOR=clr_yel
oplot, megbins, megspect, LINESTYLE=1, COLOR=clr_yel
if GRATING EQ 'HETG' then begin
  plot_oo, hegbins, chegspect, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE= spec_range, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE='HEG Spectrum: Raw and Smoothed', $
XTITLE='Energy (keV)', $
YTITLE='Photons / cm^2 s bin', $
CHARSIZE = 1.51
  oplot, hegbins, chegspect, PSYM=10, COLOR=clr_blu
  oplot, hegbins, hegspect, LINESTYLE=1, COLOR=clr_blu
end

; Put all of them together
```

```
if zoeavail then begin
  plot_oo, zobins, czospect, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE= spec_range, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE='All Smoothed Spectra for Comparison', $
XTITLE='Energy (keV)', $
YTITLE='Photons / cm^2 s bin', $
CHARSIZE = 1.51
  oplot, zobins, czospect, PSYM=10, COLOR=clr_org
  oplot, megbins, cmegspect, LINESTYLE=2, COLOR=clr_yel
end else begin
  plot_oo, megbins, cmegspect, PSYM=10, $
XSTYLE=1, XRANGE=[ge_min, ge_max], $
YSTYLE=1, YRANGE= spec_range, LINESTYLE=2, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE='All Smoothed Spectra for Comparison', $
XTITLE='Energy (keV)', $
YTITLE='Photons / cm^2 s bin', $
CHARSIZE = 1.51
  oplot, megbins, cmegspect, PSYM=10, LINESTYLE=2, COLOR=clr_yel
end
if GRATING EQ 'HETG' then begin
  oplot, hegbins, chegspect, LINESTYLE=1, COLOR=clr_blu
end

; Finish the .gif output
if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
image = tvrd()
tvlct, red, green, blue, /GET
gif_out = output_prefix+'_'+proc_method+'_gcompare.gif'
write_gif, out_dir+'/'+gif_out, image, red, green, blue

; Add .gif link
printf, out_unit, "</PRE>"
printf, out_unit, "<UL>"
printf, out_unit, '<LI><B><A href="'+gif_out+'">'+ $
```

```
'Plots to Compare the Spectra'+'</A></B>'
printf, out_unit, "</UL>"
printf, out_unit, "<PRE>"

if KEYWORD_SET(MOUSE) then begin
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end
; end of zero-order spectra comparison
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; Check BI - FI ratio etc. (S1 vs S4,S5)


; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; Check as-measured E/dE vs E for bright lines
; Do this by writing out histograms from this analysis
; and from L1.5 analysis if available.  Then call
; procedure to do the analysis...
print, ' - - - - - - - - - Outputing Histograms of Spectra - - - - - -'
print, ''
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Outputing Histograms of Spectra...</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

if KEYWORD_SET(LINE_ANALYSIS) then begin
  print, ''
  print, '                  ==> MTA Monitor Task 5.4 <=='
  print, ''

  printf, out_unit, '</PRE>'
  printf, out_unit, '<P align="center"><FONT COLOR="#00D000" SIZE=+1>'
```

```
  printf, out_unit, ' ==> MTA Monitor Task 5.4 <== '
  printf, out_unit, '</FONT></P>'
  printf, out_unit, '<PRE>'
end

; Define available spectra:
if GRATING EQ 'HETG' then begin
  if zoeavail then begin
    tg_part_strs = ['MEGm1','MEGp1','HEGm1','HEGp1']
  end else begin
    tg_part_strs = ['MEGmAll','MEGpAll','HEGmAll','HEGpAll']
  end
end
if GRATING EQ 'LETG' then begin
  if zoeavail then begin
    tg_part_strs = ['LEGm1','LEGp1']
  end else begin
    tg_part_strs = ['LEGmAll','LEGpAll']
  end
end

; Loop through the available histograms (spectra):
for is=0,n_elements(tg_part_strs)-1 do begin
  tg_part_str = tg_part_strs(is)

  ; set bins and counts for the histogram to write out
  CASE tg_part_str OF
    'MEGm1': begin
      bins = mmbins
      counts = mmcounts
    end
    'MEGp1': begin
      bins = mpbins
      counts = mpcounts
    end
    'HEGm1': begin
      bins = hmbins
```

```
  counts = hmcounts
end
'HEGp1': begin
  bins = hpbins
  counts = hpcounts
end
'MEGmAll': begin
  bins = mmbins
  counts = mmcounts
end
'MEGpAll': begin
  bins = mpbins
  counts = mpcounts
end
'HEGmAll': begin
  bins = hmbins
  counts = hmcounts
end
'HEGpAll': begin
  bins = hpbins
  counts = hpcounts
end
'LEGm1': begin
  bins = mmbins
  counts = mmcounts
end
'LEGp1': begin
  bins = mpbins
  counts = mpcounts
end
'LEGmAll': begin
  bins = mmbins
  counts = mmcounts
end
'LEGpAll': begin
  bins = mpbins
  counts = mpcounts
```

```
    end
  ELSE:    ; skip it
  ENDCASE


  ; Write the histogram file

  ; START - - - - - - hist_write_template.pro - - - - -
  ; Include and modify the following code to output data
  ; to an rdb histogram file
  ;
  ; column names
  hist_struct = {bin: 0.0, count:0.0, err:0.0}
  hist = REPLICATE(hist_struct, n_elements(bins))
  ;
  ; fill the strucuture from desired arrays...
  hist.bin = bins
  hist.count = counts
  hist.err = SQRT(FLOAT(counts))
  ;
  ; create header information
  ; each line is an element in the string array
  ; each line should start with a "#"
  ; optional parameters have the format:
  ; "# param_name[:][TAB]value_string"
  ;
  ; customize for application...
  your_header=['# created by obs_anal.pro, '+SYSTIME(),$
'# event_filename: '+filename, $
'# grating: '+grating, $
'# detector: '+detector, $
'# tg_part_str: '+tg_part_str, $
'# processing: '+proc_method ]
  ;
  ; parameters that hist routines might use:
  hist_header=['# for histogram routines use:', $
'# title: '+filename+': '+tg_part_str+' ('+ $
```

```
proc_method+')', $
'# bin_axis: Energy', $
'# bin_unit: keV' ]
  header = [your_header, hist_header]
  ;
  ; and write it out to desired file
  hist_file = output_prefix+'_'+ $
proc_method+'_'+tg_part_str+'_hist.rdb'
  print, ' Creating '+hist_file+' ...'
  printf, out_unit, ' Creating <B><A href="'+hist_file+'">'+hist_file+'</A></B> ...'
  ; prepend the output directory...
  hist_file = out_dir+'/'+hist_file
  rdb_write, hist_file, hist, HEAD = header, /SILENT
  ; END - - - - - - hist_write_template.pro - - - - -

  ; Write out the ISIS format file
  isis_file = output_prefix+'_'+proc_method+'_'+tg_part_str+'_isis.dat'
  ; Following could be turned into procedure: isis_write_data, isis_file, ...,
  ; Convert back to wavelength histogram (arg!  "Energy -- going once, going twice, ...")
  lambins = hc/bins
  sortlam = SORT(lambins)
  lambins = lambins(sortlam)
  lamcounts = counts(sortlam)
  dlam2 = (lambins(1) - lambins(0))/2.0  ; these histograms were linear in lambda
  print, ' Creating ISIS format: '+isis_file+' ...'
  printf, out_unit, ' Creating ISIS format: <B><A href="'+isis_file+'">'+isis_file+'</A></B> ...'
  OPENW, isis_unit, out_dir+'/'+isis_file, /GET
  printf, isis_unit, n_elements(lambins)
  for ib=0,n_elements(lambins)-1 do begin
    printf, isis_unit, lambins(ib)-dlam2, lambins(ib)+dlam2, $
lamcounts(ib), SQRT(lamcounts(ib))
  end
  close, isis_unit
  free_lun, isis_unit

  if KEYWORD_SET(LINE_ANALYSIS) then begin
```

```
    if KEYWORD_SET(MOUSE) then begin
      hist_lines, hist_file, /MOUSE, $
OUTPUT_PREFIX=hist_prefix, OUT_DIR = out_dir, ZBUFFER=zbuffer
    end else begin
      hist_lines, hist_file, $
OUTPUT_PREFIX=hist_prefix, OUT_DIR = out_dir, ZBUFFER=zbuffer
    end

    ; Make links to the output files
    ; hist_lines produces files named hist_prefix + :
    ; _specpanes.gif
    ; _linefits.gif
    ; _eoverde.gif
    ; _linelist.txt
    ; Build links for these...

    printf, out_unit, '</PRE>'
    printf, out_unit, '<TABLE BORDER=1>'
    printf, out_unit, '<TR>'
    this_file = hist_prefix + '_specpanes.gif'
    printf, out_unit, '<TD><B><A href="'+this_file+'"> Spectrum </A></B>'
    this_file = hist_prefix + '_linefits.gif'
    printf, out_unit, '<TD><B><A href="'+this_file+'"> Line Fits </A></B>'
    this_file = hist_prefix + '_eoverde.gif'
    printf, out_unit, '<TD><B><A href="'+this_file+'"> E/dE vs E </A></B>'
    this_file = hist_prefix + '_linelist.txt'
    printf, out_unit, '<TD><B><A href="'+this_file+'"> Line List </A></B>'
    printf, out_unit, '</TABLE>'
    printf, out_unit, '<PRE>'
  end

print,''
printf, out_unit,''
end ; of loop over spectra to write out...
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

if KEYWORD_SET(MOUSE) then begin
```

```
  print, '    !!! Click Mouse anywhere to Continue !!!'
  print, ''
  cursor, x, y, 3
end

; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; Level 1.5 summary (numbers of events in sources, parts, )
; TG_SRCID, TG_PART
print, ' - - - - - - - - - - Check Level 1.5 Columns... - - - - - - -'
print, ''
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Check Level 1.5 Columns...</H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

; List all evts tags that have tg_ in them:
l1p5tags = where(STRPOS(evts_tags,'TG_') GE 0, nfound)

if nfound EQ 0 then begin
  print, ' No Level 1.5 columns found in the file.'
  print, ''
  printf, out_unit, ' No Level 1.5 columns found in the file.'
  printf, out_unit, ''
end else begin

  print, ' Found Level 1.5 columns: '
  print, '     '+evts_tags(l1p5tags)
  print, ''

  printf, out_unit, ' Found Level 1.5 columns: '
  printf, out_unit, '     '+evts_tags(l1p5tags)
  printf, out_unit, ''

  if TOTAL(evts_tags EQ 'TG_SRCID') GT 0 then begin
    maxsrcid = MAX(evts(where(evts.tg_srcid LT 99)).tg_srcid)
    print, ' TG_SRCID: MAX value (<99) is: '+STRCOMPRESS(maxsrcid)
```

```
  print, ''
  printf, out_unit, ' TG_SRCID: MAX value (<99) is: '+STRCOMPRESS(maxsrcid)
  printf, out_unit, ''
end
if TOTAL(evts_tags EQ 'TG_M') GT 0 then begin
  maxtgord = MAX(evts(where(evts.tg_m LT 99)).tg_m)
  print, ' TG_M: MAX value (<99) is: '+STRCOMPRESS(maxtgord)
  print, ' TG_M: MIN value is: '+STRCOMPRESS(MIN(evts.TG_M))
  print, ''
  printf, out_unit, ' TG_M: MAX value (<99) is: '+STRCOMPRESS(maxtgord)
  printf, out_unit, ' TG_M: MIN value is: '+STRCOMPRESS(MIN(evts.TG_M))
  printf, out_unit, ''
end
if TOTAL(evts_tags EQ 'TG_PART') GT 0 then begin
  ; Make a table of the number of events in each Source-Part
  ; Plot the events in TG_R, TG_D while we're here...
  if GRATING EQ 'LETG' then begin
    nspectra = 1 ; one spectra for each source
  end else begin
    nspectra = 2 ; heg, meg spectra for each source
  end
  print, ' Events in different Source-Parts:'
  print, ''
  print, '   Source      Zero-order     HEG     MEG     LEG'
  print, '  - - - - - - - - - - - - - - - - - - - - - - - - -'
  printf, out_unit, ' Events in different Source-Parts:'
  printf, out_unit, ''
  printf, out_unit, '   Source      Zero-order     HEG     MEG     LEG'
  printf, out_unit, '  - - - - - - - - - - - - - - - - - - - - - - - - -'
  bkgnd = where(evts.TG_PART EQ 99, nbkgnd)
  print, '   Backgnd    '+STRING(nbkgnd, FORMAT='(I8)')
  printf, out_unit, '   Backgnd    '+STRING(nbkgnd, FORMAT='(I8)')
  print, '  . . . . . . . . . . . . . . . . . . . . . . . . .'
  printf, out_unit, '  . . . . . . . . . . . . . . . . . . . . . . . . .'
  for is=1,maxsrcid do begin
    if is GT 1 then begin
      print, '  . . . . . . . . . . . . . . . . . . . . . . . . .'
```

```
        printf, out_unit, '  . . . . . . . . . . . . . . . . . . . . . . . . .'
      end
      ; and plot TG_D vs hc/TG_LAM in ge_min to ge_max range also!
      !p.multi = [0,1,2*nspectra]
      found = where(evts.tg_srcid EQ is AND evts.tg_part EQ 0, nzopart)

      found = where(evts.tg_srcid EQ is AND evts.tg_part EQ 2, nmegpart)
        if nmegpart GT 0 then begin
          plot, evts(found).TG_R, evts(found).TG_D, PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE = 'TG_R - TG_D Plot for MEG Part of Source '+$
STRCOMPRESS(is,/REMOVE_ALL), $
XTITLE = 'TG_R (degrees)', YTITLE='TG_D (degrees)', $
CHARSIZE=1.71
          plots, evts(found).TG_R, evts(found).TG_D, PSYM=3, $
COLOR=energy_colors(found)
          plot_oi, hc/evts(found).TG_LAM, evts(found).TG_D, PSYM=3, $
XSTYLE=1, YSTYLE=1, XRANGE=[ge_min,ge_max], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE = 'TG_Energy - TG_D Plot for MEG Part of Source '+$
STRCOMPRESS(is,/REMOVE_ALL), $
XTITLE = 'hc/TG_LAM (keV)', YTITLE='TG_D (degrees)', $
CHARSIZE=1.71
          plots, hc/evts(found).TG_LAM, evts(found).TG_D, PSYM=3, $
COLOR=energy_colors(found)
        end

      found = where(evts.tg_srcid EQ is AND evts.tg_part EQ 1, nhegpart)
        if nhegpart GT 0 then begin
          plot, evts(found).TG_R, evts(found).TG_D, PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE = 'TG_R - TG_D Plot for HEG Part of Source '+$
STRCOMPRESS(is,/REMOVE_ALL), $
XTITLE = 'TG_R (degrees)', YTITLE='TG_D (degrees)', $
CHARSIZE=1.71
```

```
          plots, evts(found).TG_R, evts(found).TG_D, PSYM=3, $
COLOR=energy_colors(found)
          plot_oi, hc/evts(found).TG_LAM, evts(found).TG_D, PSYM=3, $
XSTYLE=1, YSTYLE=1, XRANGE=[ge_min,ge_max], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE = 'TG_Energy - TG_D Plot for MEG Part of Source '+$
STRCOMPRESS(is,/REMOVE_ALL), $
XTITLE = 'hc/TG_LAM (keV)', YTITLE='TG_D (degrees)', $
CHARSIZE=1.71
          plots, hc/evts(found).TG_LAM, evts(found).TG_D, PSYM=3, $
COLOR=energy_colors(found)
        end

      found = where(evts.tg_srcid EQ is AND evts.tg_part EQ 3, nlegpart)
        if nlegpart GT 0 then begin
          plot, evts(found).TG_R, evts(found).TG_D, PSYM=3, $
XSTYLE=1, YSTYLE=1, $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE = 'TG_R - TG_D Plot for LEG Part of Source '+$
STRCOMPRESS(is,/REMOVE_ALL), $
XTITLE = 'TG_R (degrees)', YTITLE='TG_D (degrees)', $
CHARSIZE=1.71
          plots, evts(found).TG_R, evts(found).TG_D, PSYM=3, $
COLOR=energy_colors(found)
          plot_oi, hc/evts(found).TG_LAM, evts(found).TG_D, PSYM=3, $
XSTYLE=1, YSTYLE=1, XRANGE=[ge_min,ge_max], $
BACK=clr_back, COLOR=clr_wht, /NODATA, $
TITLE = 'TG_Energy - TG_D Plot for MEG Part of Source '+$
STRCOMPRESS(is,/REMOVE_ALL), $
XTITLE = 'hc/TG_LAM (keV)', YTITLE='TG_D (degrees)', $
CHARSIZE=1.71
          plots, hc/evts(found).TG_LAM, evts(found).TG_D, PSYM=3, $
COLOR=energy_colors(found)
        end
      print, STRING(is,FORMAT='(I7)')+ '       '+ $
STRING(nzopart, FORMAT='(I8)') + '    ' + $
STRING(nhegpart, FORMAT='(I8)') + $
```

```
STRING(nmegpart, FORMAT='(I8)') + $
STRING(nlegpart, FORMAT='(I8)')

      ; Finish the .gif output
      if NOT(KEYWORD_SET(ZBUFFER)) then wshow,iwind,iconic=0
      image = tvrd()
      tvlct, red, green, blue, /GET
      gif_out = output_prefix+'_'+proc_method+'_S'+STRCOMPRESS(is,/REMOVE_ALL)+'_tgrdlam.gif'
      write_gif, out_dir+'/'+gif_out, image, red, green, blue

      printf, out_unit, STRING(is,FORMAT='(I7)')+ '       '+ $
STRING(nzopart, FORMAT='(I8)') + '    ' + $
STRING(nhegpart, FORMAT='(I8)') + $
STRING(nmegpart, FORMAT='(I8)') + $
STRING(nlegpart, FORMAT='(I8)')

      ; add gif link (indent a bunch with DL)
      printf, out_unit, "</PRE>"
      printf, out_unit, "<DL><DT><DD><UL>"
      printf, out_unit, '<LI><B><A href="'+gif_out+'">'+'TG_R, TG_D, TG_LAM plots'+'</A></B>'
      printf, out_unit, "</UL></DL>"
      printf, out_unit, "<PRE>"


      ; Wait after each source spectra are displayed...
      if KEYWORD_SET(MOUSE) then begin
        print, '    !!! Click Mouse anywhere to Continue !!!'
        print, ''
        cursor, x, y, 3
      end

    end
    print, '  - - - - - - - - - - - - - - - - - - - - - - - - - -'
    print, ''
    printf, out_unit, '  - - - - - - - - - - - - - - - - - - - - - - - - - -'
    printf, out_unit, ''
  end
```

```
  print, ''
  printf, out_unit, ''

end ; of checking for L1.5
;  end of Simple Level 1.5 stuff
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


; - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; Output Level 1.5 histograms
print, ' - - - - - - - - - Output Histogram files from L1.5 - - - - - -'
print, ''
printf, out_unit, "</PRE>"
printf, out_unit, '<HR>'
printf, out_unit, '<H3>Output Histogram files from L1.5... </H3>'
printf, out_unit, "<PRE>"
printf, out_unit, ""

; L1.5 tags available?
l1p5tags = where(STRPOS(evts_tags,'TG_') GE 0, nfound)
if nfound EQ 0 then begin
  print, ' No Level 1.5 columns - cannot create L1.5 histograms.'
  print, ''
  printf, out_unit, ' No Level 1.5 columns - cannot create L1.5 histograms.'
  printf, out_unit, ''
end else begin
  if KEYWORD_SET(LINE_ANALYSIS) then begin
    print, ''
    print, '                    ==> MTA Monitor Task 5.4 <=='
    print, ''

    printf, out_unit, '</PRE>'
    printf, out_unit, '<P align="center"><FONT COLOR="#00D000" SIZE=+1>'
    printf, out_unit, ' ==> MTA Monitor Task 5.4 <== '
    printf, out_unit, '</FONT></P>'
    printf, out_unit, '<PRE>'
  end
```

```
print, ' Level 1.5 columns - creating L1.5 histograms...'
print, ''
printf, out_unit, ' Level 1.5 columns - creating L1.5 histograms...'
printf, out_unit, ''
; Flag the processing...
proc_method = 'L1.5'
; All orders are in TG_LAM...
orders_used_str = 'All'
; unless ACIS is available for sorting
if STRPOS(DETECTOR,'ACIS') GE 0 then orders_used_str = '1'

; Go through the sources
maxsrcid = MAX(evts(where(evts.tg_srcid LT 99)).tg_srcid)
for is=1,maxsrcid do begin
  if GRATING EQ 'HETG' then begin
    parts = [2,1]
    tg_part_strs = ['MEG','HEG']
    orders = [-1,1]
  end else begin
    parts = [3]
    tg_part_strs = ['LEG']
    orders = [-1,1]
  end
  ; Loop over the parts and orders
  for ip=0,n_elements(parts)-1 do begin
    this_part = parts(ip)
    for io=0,n_elements(orders)-1 do begin
      this_order = orders(io)
      ; make the tg_part_str for hist filename output...
      ord_str = 'p'
      if this_order EQ 0 then ord_str = 'z'
      if this_order LT 0 then ord_str = 'm'
      if this_order NE 0 then begin
        ord_str = ord_str + STRCOMPRESS(ABS(this_order),/REMOVE_ALL)
      end
      tg_part_str = tg_part_strs(ip)+ord_str
      ; Get the events for this source, part, order
```

```
        this_spo = where( (evts.tg_srcid EQ is) AND $
(evts.tg_m EQ this_order) AND $
(evts.tg_part EQ this_part), nfound)
        print, ' Source '+ STRCOMPRESS(is) + '  '+tg_part_str + $
' : Number of events = '+ STRCOMPRESS(nfound)
        printf, out_unit, ' Source '+ STRCOMPRESS(is) + '  '+tg_part_str + $
' : Number of events = '+ STRCOMPRESS(nfound)

        ; Make a histogram of the energies
        log_hist, hc/evts(this_spo).tg_lam, ge_bin_ratio, bins, counts
        ; Write out the histogram

  ; START - - - - - - hist_write_template.pro - - - - -
  ; Include and modify the following code to output data
  ; to an rdb histogram file
  ;
  ; column names
  hist_struct = {bin: 0.0, count:0.0, err:0.0}
  hist = REPLICATE(hist_struct, n_elements(bins))
  ;
  ; fill the strucuture from desired arrays...
  hist.bin = bins
  hist.count = counts
  hist.err = SQRT(FLOAT(counts))
  ;
  ; create header information
  ; each line is an element in the string array
  ; each line should start with a "#"
  ; optional parameters have the format:
  ; "# param_name[:][TAB]value_string"
  ;
  ; customize for application...
  your_header=['# created by obs_anal.pro, '+SYSTIME(),$
'# event_filename: '+filename, $
'# grating: '+grating, $
'# detector: '+detector, $
'# source: '+STRCOMPRESS(is,/REMOVE_ALL), $
```

```
'# tg_part_str: '+tg_part_str, $
'# processing: '+proc_method ]
  ;
  ; parameters that hist routines might use:
  hist_header=['# for histogram routines use:', $
'# title: '+filename+': '+tg_part_str+' ('+ $
proc_method+')', $
'# bin_axis: Energy', $
'# bin_unit: keV' ]
  header = [your_header, hist_header]
  ;
  ; and write it out to desired file
  hist_file = output_prefix+'_'+ $
proc_method+'_'+'S'+STRCOMPRESS(is,/REMOVE_ALL)+ $
tg_part_str+'_hist.rdb'
  print, ' Creating '+hist_file+' ...'
  printf, out_unit, ' Creating <B><A href="'+hist_file+'">'+hist_file+'</A></B> ...'
  ; prepend the output directory...
  hist_file = out_dir + '/' + hist_file
  rdb_write, hist_file, hist, HEAD = header, /SILENT
  ; END - - - - - - hist_write_template.pro - - - - -

  if KEYWORD_SET(LINE_ANALYSIS) then begin

    if KEYWORD_SET(MOUSE) then begin
      hist_lines, hist_file, /MOUSE, $
OUTPUT_PREFIX=hist_prefix, OUT_DIR = out_dir, ZBUFFER=zbuffer
    end else begin
      hist_lines, hist_file, $
OUTPUT_PREFIX=hist_prefix, OUT_DIR = out_dir, ZBUFFER=zbuffer
    end

    ; Make links to the output files
    ; hist_lines produces files named hist_prefix + :
    ; _specpanes.gif
    ; _linefits.gif
    ; _eoverde.gif
```

```
    ; _linelist.txt
    ; Build links for these...

    printf, out_unit, '</PRE>'
    printf, out_unit, '<TABLE BORDER=1>'
    printf, out_unit, '<TR>'
    this_file = hist_prefix + '_specpanes.gif'
    printf, out_unit, '<TD><B><A href="'+this_file+'"> Spectrum </A></B>'
    this_file = hist_prefix + '_linefits.gif'
    printf, out_unit, '<TD><B><A href="'+this_file+'"> Line Fits </A></B>'
    this_file = hist_prefix + '_eoverde.gif'
    printf, out_unit, '<TD><B><A href="'+this_file+'"> E/dE vs E </A></B>'
    this_file = hist_prefix + '_linelist.txt'
    printf, out_unit, '<TD><B><A href="'+this_file+'"> Line List </A></B>'
    printf, out_unit, '</TABLE>'
    printf, out_unit, '<PRE>'

  end else begin
    print, ''
    printf, out_unit, ''
  end
  ; finished writing histogram

      end ; loop over orders

    end ; loop over parts

  end ; loop over sources


end ; of Level 1.5 histogram output
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


print, ' - - - - - - - - - obs_anal: Analyses Complete - - - - - -'
print, ''
printf, out_unit, "</PRE>"
```

```
printf, out_unit, '<HR>'
printf, out_unit, "<PRE>"
printf, out_unit, SYSTIME()+ ' : obs_anal.pro finished'
printf, out_unit, "</PRE>"

; Close the html file
printf, out_unit, "</HTML>"
CLOSE, out_unit
FREE_LUN, out_unit

; Reset the device and plot parameters...
set_plot, Orig_device
!p = Orig_plot      ; restore it all?

RETURN
END
```