Yohkoh Analysis Guide
Reference Guide

Ver 1.06 Ed: M.D. Morrison 6-Nov-92
Ver 2.00 Ed: M.D. Morrison 25-Jan-94
Ver 2.91 Ed: R.D. Bentley 1-Mar-98

This Copy Produced: March 6, 1998

Prepared at
Mullard Space Science Laboratory
University College London

Contents

List of Figures

1 Introduction to the Yohkoh Data and Software

1.1 YS, YDB, and YDn Directories

The Yohkoh data are organized into three general parts.

YS The software, documentation, and some calibration data files are located in the Yohkoh Software (YS) tree. There is a separate branch for each instrument, and one for general software which holds software and data that can be used by all instruments.

YDB The secondary database files are contained in the Yohkoh Database (YDB) tree. The secondary database files include S/C attitude information, observing logs, event logs, detailed calibration files, summary data files, and several ground-based database files.

YD The raw reformatted Yohkoh data (YD) are placed in the YD directories. At ISAS, there are twenty one Yohkoh data directories (/yd0, /yd1, /yd2, ... /yd20). The standard method for putting data on-line is to create a directory with the week number, for example /yd10/93_41a and put all the data for week 93_41 into that directory (except for the weekly files).

1.2 Yohkoh Software Organization

It is assumed that you have successfully installed the Yohkoh software on your system or you are using a machine that has the Yohkoh software. In addition to having made the software installation, it is necessary to have executed the Yohkoh initialization routine ('/ys/gen/script/idl_setup' or '.yslogin' for the Unix machines). If you have not done this yet, see the Appendix of the Yohkoh Reference Guide.

The Yohkoh software is organized under one tree. The top directory is $ys for Unix systems, and is the logical YS: for the VMS systems. Under that tree are the following branches:

Branch	*Description*
SITE	Site specific software
GEN	General software and documents that all instruments can use
BCS_F77	BCS FORTRAN specific software and documentation
BCS	BCS specific software and documentation
HXT	HXT specific software and documentation
HXT_F77	HXT FORTRAN specific software and documentation
SXT	SXT specific software and documentation
WBS	WBS specific software and documentation
ATEST	Newly created or modified software and documentation
UCON	User Contributed software and documentation
IDLFIX	Work-arounds for systems using older IDL versions

Under each of the above instrument directories, there are the following directories (for example, '/ys/sxt/doc')

Branch	*Description*
DOC	Documentation
RESPONSE	Instrument calibration and response data files
SOFT	Software
STATUS	Instrument status information

Software that has been thoroughly tested is put into the instrument release directory 'soft'.[1] Software which has just been written, or software that is modified is put into the 'atest' area for a period of a few weeks. Certain privileged users have write privilege to those directories. If problems develop with modified software, it is possible to recover the old version by copying it from the 'soft' directory. The software developed by general users is placed online under the 'ucon' (user contributed) branch. Generally each user who is contributing has a directory of their own. The directories under the 'soft' branch are broken up by function for the 'gen' and instrument branchs, but by person under the 'ucon' branch. If a user cannot remember the name of a function, he/she can do a listing on the different directories and will probably recognize it. A list of the directories that currently exist for the instrument teams is:

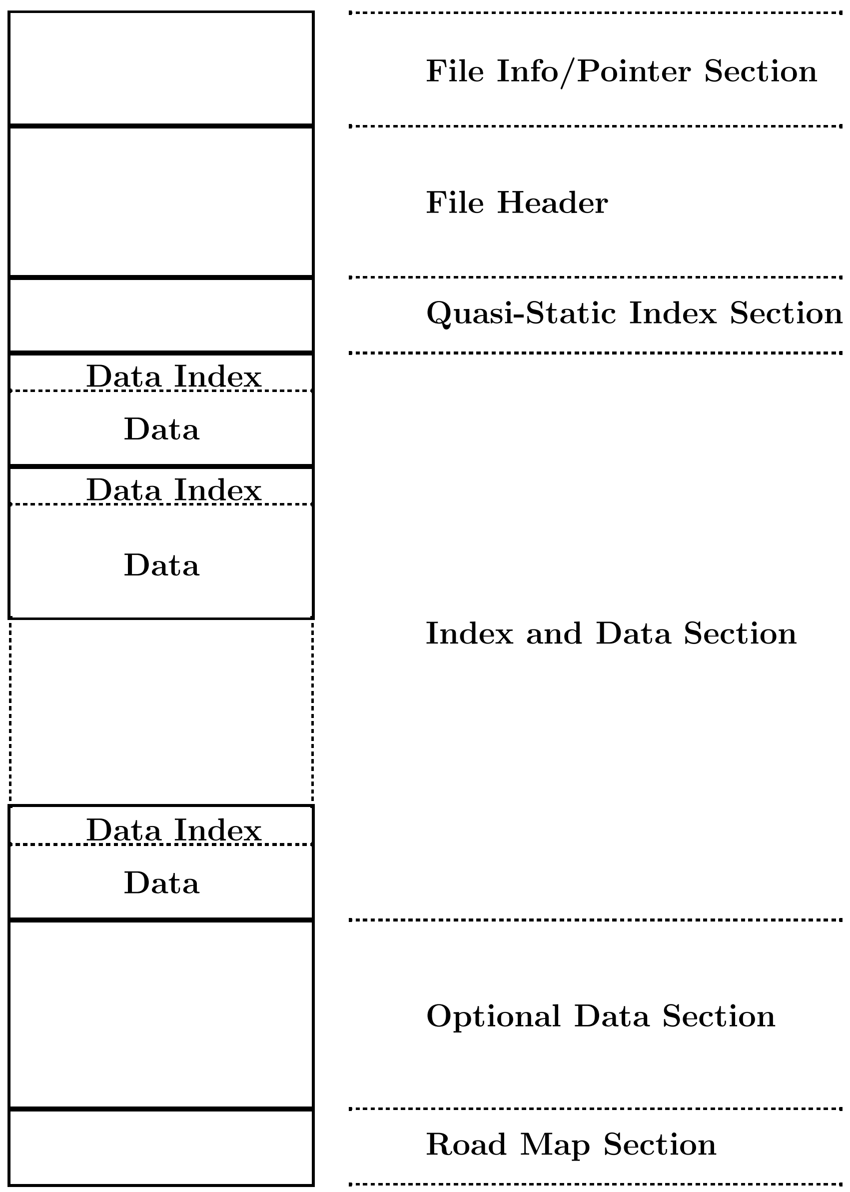
GEN	*BCS*	*HXT*	*SXT*	*WBS*
dbase	bda	util	register	util
gbo	bsc		sensitivity	
jhuapl	bsd		util	
mo_disk	to_be_deleted		widgets	
movie	util			
orbit				
pointing				
ref_access				
reformat				
spectra				
tape				
util				
utplot				

1.3 How to Contribute Software to the Yohkoh Tree

See the appendix of the Reference Guide.

[1] The previous organization had an 'atest', 'rel', and 'usercontrib' branch under each 'soft' directory.

SOLAR-A reformatted files



1.4 Yohkoh Data File Format

For a full description of the Yohkoh database and all the different data structures, see the File Control Document.

Each file is logically divided into the following six sections. Some files will not use all the sections described below, but all will have a Pointer and File Header Section.

1. File Information and Pointer Section
2. File Header Section
3. Quasi-Static Index Section
4. Index and Data Section
5. Optional Data Section
6. Road Map Section

The program which reads the file learns from the Pointer Section how to read the rest of the file and where to go to get certain data.

The File Header Section provides information on what data are contained in the file, generally the extent of the time covered by the contents.

The Quasi-Static Section of the file contains index information that does not vary during the course of an orbit, or varies slowly.

The Index and Data section contains 'data sets'. A data set is a single image for SXT, single spectrum for BCS, a single major frame of data for HXT, and two major frames of data for WBS (it takes two major frames for a complete set of WBS data). For each data set there is an index which describes the date and time that the data were taken, the mode and position of the instrument's peripherals (e.g., filters, HV), and information on temperature and gain information.

The Optional Data Section is only used by the BCS data (BDA) files and the spacecraft attitude (ADA) files. The BDA files hold the 'DP_SYNC' information, which is information that is coming down every major frame. Since the BCS spectra are asynchronous to the major frames, it is stored separately. The ADA file holds the full 2048 point HXA scans.

The Roadmap Section allows a user to access a brief summary of the contents of the file and to perform searches on that summary to select what data should be extracted.

1.5 Yohkoh Data File Names

1.5.1 Yohkoh Directory Logicals/Environment Variables

There is a set of VMS logicals and Unix environment variables which point to the database directories. By using these logicals in all of the access routines (along with the routine CONCAT_DIR) it is possible for software to be directly portable between VMS and Unix machines. The following is a list of the logicals that exist for the Yohkoh database.

DIR_GEN_ADS	DIR_SXT_ATABLES	DIR_BCS_ATODAT
DIR_GEN_ATR	DIR_SXT_CAL	DIR_BCS_BALDAT
DIR_GEN_ATT	DIR_SXT_CALIBRATE	DIR_BCS_CAL
DIR_GEN_DATA	DIR_SXT_DOC	DIR_BCS_CALDAT
DIR_GEN_DOC	DIR_SXT_ENGIN	DIR_BCS_CAT
DIR_GEN_EVN	DIR_SXT_SDC	DIR_BCS_DOC
DIR_GEN_FEM	DIR_SXT_SDL	DIR_BCS_EXE
DIR_GEN_GBE	DIR_SXT_SDW	DIR_BCS_LOGS
DIR_GEN_GBL	DIR_SXT_SENSITIVE	DIR_BCS_MICRO
DIR_GEN_GEV	DIR_SXT_SFC	DIR_BCS_SYNSPEC
DIR_GEN_GOL	DIR_SXT_SFM	
DIR_GEN_GXT	DIR_SXT_SFS	DIR_GBO_TEMP
DIR_GEN_MO	DIR_SXT_SLD	
DIR_GEN_MOVIE	DIR_SXT_SLS	
DIR_GEN_NAR	DIR_SXT_SOT	
DIR_GEN_NEL	DIR_SXT_SSL	
DIR_GEN_NTS	DIR_SXT_SSX	
DIR_GEN_OBD	DIR_SXT_SXA	
DIR_GEN_OBS	DIR_SXT_SXC	
DIR_GEN_ORBIT	DIR_SXT_SXL	
DIR_GEN_ORBIT_RAW	DIR_SXT_SXL2	
DIR_GEN_ORBIT_SOL	DIR_SXT_TABLES	
DIR_GEN_ORBIT_SW		
DIR_GEN_OSF	DIR_WBS_CAL	
DIR_GEN_OSP		
DIR_GEN_OWH	DIR_HXT_CAL	
DIR_GEN_PAN_LASER	DIR_HXT_EXE	
DIR_GEN_PNT		
DIR_GEN_SCRIPT		
DIR_GEN_SETUP		
DIR_GEN_SETUPD		
DIR_GEN_SETUPDBACK		
DIR_GEN_SPECTRA		
DIR_GEN_STATUS		
DIR_GEN_SYNOPTIC		
DIR_GEN_TAPECOPY		
DIR_GEN_TBB		
DIR_GEN_XAD		
DIR_GEN_XBD		

1.5.2 Yohkoh Prefixes

Prefix	*Description*	*File Type*
ADA	S/C Attitude Raw Reformatted Data	per Orbit
ATR	Raw S/C Attitude Database	Weekly
ATT	Processed S/C Attitude Database	Weekly
BDA	BCS Raw Reformatted Data	per Orbit
BSD	*OLD* BCS Instrument Calibrated Spectra	User Specified
BPC	*OLD* BSDCAL Output (parameters)	User Specified
BTH	*OLD* BSDFIT Output (Fitted Spectra)	User Specified
BFT	*OLD* BSDFIT Output (parameters)	User Specified
BSC	*NEW* BCS Instrument Calibrated Spectra	User Specified
BSA	*NEW* Answer File for BSC Generation	User Specified
BSF	*NEW* Output from Spectral Fitting	User Specified
CBA	S/C Common Basic Raw Reformatted Data	per Orbit
EVN	Yohkoh Event Log	Weekly
FEM	Yohkoh Orbital Ephemeris	Weekly
GEV	GOES Event Log	Weekly
GXT	GOES One Minute Light Curve Data	Weekly
G6D	GOES 3 Sec Light Curve for S/C 6	Weekly
G7D	GOES 3 Sec Light Curve for S/C 7	Weekly
GXD	GOES Derived One Minute Light Curve Data	Weekly
GBE	GRO BATSE Event Log	Mission
GBL	GRO BATSE Light Curves	Weekly
G_x	Ground-Based Observation (GBO) Images (Sites)	per Image (FITS)
Gx_	Ground-Based Observation (GBO) Images (Data Type)	per Image (FITS)
GBC	GBO Big Bear Large Scale H-alpha PFI FITS Image	per Image (FITS)
GBH	GBO Big Bear H-alpha FITS Image	per Image (FITS)
GBK	GBO Big Bear Calcium FITS Image	per Image (FITS)
GBW	GBO Big Bear White Light FITS Image	per Image (FITS)
GCH	GBO Boulder Colorado H-alpha FITS Image	per Image (FITS)
GGH	GBO GSFC H-alpha FITS Image	per Image (FITS)
GHH	GBO Holloman H-alpha FITS image	per Image (FITS)
GLH	GBO Leamonth Australia H-Alpha FITS Image	per Image (FITS)
GNH	GBO NAOJ Japan H-alpha FITS Image	per Image (FITS)
GJH	GBO CRL Hiraiso H-alpha FITS Image	per Image (FITS)
GKI	GBO Kitt Peak He 10830 FITS Image	per Image (FITS)
GKM	GBO Kitt Peak Magnetogram FITS Image	per Image (FITS)
GNP	GBO Nobeyama Polarized Radio Image (17 GHz)	per Image (FITS)
GNT	GBO Nobeyama Total Intensity Radio Image (17 GHz)	per Image (FITS)
GOL	GBO Observing Log	Weekly

Prefix	*Description*	*File Type*
HDA	HXT Raw Reformatted Data	per Orbit
HXI	HXT Synthesized Image	User Specified
MDM	Magneto-Optical (MO) Map	Mission
NAR	NOAA Active Region	Weekly
NEL	Nobeyama Radio Observatory Event Log	Mission
NTS	Nobeyama Radio Observatory Time Series	Weekly
OBS	*OLD* Yohkoh Observing Log	Weekly
OBD	BCS Observing Log	Weekly
OSF	SXT Full Frame Observing Log	Weekly
OSP	SXT Partial Frame Observing Log	Weekly
OWH	WBS/HXT Observing Log	Weekly
PNT	*OLD* S/C Pointing File	Weekly
SDL	SXT Dark Current Log	Weekly
SDC	SXT Dark Current Images	Weekly (SDA)
SDW	SXT Dark Current Images for Warm CCD	Weekly (SDA)
SDP	SXT Dark Current PFI Images	Weekly (SDA)
SFD	SXT FFI Desaturated Composite Images	Weekly (SDA)
SFR	SXT FFI Raw Reformatted Data	per Orbit (SDA)
SFS	SXT FFI Special Images (diffuser, flood...)	Weekly (SDA)
SFT	SXT FFI Terminator Images	Weekly (SDA)
SFW	SXT FFI White Light Images	Weekly (SDA)
SFC	SXT FFI Calibration (Leak) Image	per Image (SDA)
SFM	SXT FFI Monthly File (daily SFD image)	Monthly (SDA)
SLS	SXT Leak Scatter Log	Weekly
SLD	SXT Leak Dark Current Log	Weekly
SPR	SXT PFI Raw Reformatted Data	per Orbit (SDA)
SOT	SXT Optical Telescope Log	Weekly
SSC	SXT Synoptic Images (centered)	Weekly (SDA)
SSE	SXT Synoptic Images (east of center)	Weekly (SDA)
SSW	SXT Synoptic Images (west of center)	Weekly (SDA)
SSL	SXT Summary Log	Weekly
SXC	SXT X-Ray Sun Center Log	Weekly
SXL	SXT X-Ray Histogram Log	Weekly
SXA	SXT X-Ray Average	Weekly
WDA	WBS Raw Reformatted Data	per Orbit
XAD	Exabyte ASCII Directory for Archive Tape	Weekly per Tape
XBD	Exabyte Binary Directory for Archive Tape	Weekly per Tape

1.5.3 Yohkoh File IDs

The orbit file ID's are 11 characters long in the format shown below:

```
YYMMDD.HHMM

where YY - Year of data
      MM - Month of data
      DD - Day of data
      HH - Hours
      MM - Minutes
```

1.5.4 Yohkoh Week IDs

The Yohkoh week starts on Sunday and ends on Saturday. CAUTION: Yohkoh data taken at 00:01 (one minute after midnight) on Sunday morning is most likely in a file and tape for the previous week since the orbit started on Saturday and the first full orbit for Sunday has not started. Up to the first 90 minutes of data taken on Sunday could be stored in the previous week, depending on when the spacecraft day begins.

The weekly IDs are of the form:

```
YY_WWa.NN

where YY - Year of data
      WW - Week number of the data (1 to 53)
       a - is fixed (reserved for future use)
      NN - is the program version number which created the file
```

A list of the dates covered by each of the weeks is given in Appendix xxx.

1.5.5 Yohkoh Carrington IDs

The Carrington IDs are of the form:

```
  _crRRRRa.NN

where cr   - signifies a Carrington Rotation ID
      RRRR - is the rotation number
       a   - is fixed (reserved for future use)
      NN   - is the program version number which created the file
```

The dates for the Carrington Rotations are defined in the table.

1.6 Interactive Data Language (IDL)

Most of the Yohkoh software is written in the programming language IDL of Research Systems, Inc (RSI). If certain rules are followed, then the same IDL software can be run on a wide variety of computers.

Carrington Rotation	*Starting Date*	*Carrington Rotation*	*Starting Date*	*Carrington Rotation*	*Starting Date*
1846	21-Aug-91	1888	10-Oct-94	1930	29-Nov-97
1847	18-Sep-91	1889	6-Nov-94	1931	26-Dec-97
1848	15-Oct-91	1890	3-Dec-94	1932	22-Jan-98
1849	11-Nov-91	1891	31-Dec-94	1933	18-Feb-98
1850	8-Dec-91	1892	27-Jan-95	1934	18-Mar-98
1851	5-Jan-92	1893	23-Feb-95	1935	14-Apr-98
1852	1-Feb-92	1894	23-Mar-95	1936	11-May-98
1853	28-Feb-92	1895	19-Apr-95	1937	7-Jun-98
1854	26-Mar-92	1896	16-May-95	1938	5-Jul-98
1855	23-Apr-92	1897	12-Jun-95	1939	1-Aug-98
1856	20-May-92	1898	10-Jul-95	1940	28-Aug-98
1857	16-Jun-92	1899	6-Aug-95	1941	25-Sep-98
1858	14-Jul-92	1900	2-Sep-95	1942	22-Oct-98
1859	10-Aug-92	1901	29-Sep-95	1943	18-Nov-98
1860	6-Sep-92	1902	27-Oct-95	1944	15-Dec-98
1861	3-Oct-92	1903	23-Nov-95	1945	12-Jan-99
1862	31-Oct-92	1904	20-Dec-95	1946	8-Feb-99
1863	27-Nov-92	1905	17-Jan-96	1947	7-Mar-99
1864	24-Dec-92	1906	13-Feb-96	1948	3-Apr-99
1865	20-Jan-93	1907	11-Mar-96	1949	1-May-99
1866	17-Feb-93	1908	7-Apr-96	1950	28-May-99
1867	16-Mar-93	1909	5-May-96	1951	24-Jun-99
1868	12-Apr-93	1910	1-Jun-96	1952	22-Jul-99
1869	10-May-93	1911	28-Jun-96	1953	18-Aug-99
1870	6-Jun-93	1912	26-Jul-96	1954	14-Sep-99
1871	3-Jul-93	1913	22-Aug-96	1955	11-Oct-99
1872	30-Jul-93	1914	18-Sep-96	1956	8-Nov-99
1873	27-Aug-93	1915	15-Oct-96	1957	5-Dec-99
1874	23-Sep-93	1916	12-Nov-96	1958	1-Jan-00
1875	20-Oct-93	1917	9-Dec-96	1959	29-Jan-00
1876	17-Nov-93	1918	5-Jan-97	1960	25-Feb-00
1877	14-Dec-93	1919	1-Feb-97	1961	23-Mar-00
1878	10-Jan-94	1920	1-Mar-97	1962	19-Apr-00
1879	6-Feb-94	1921	28-Mar-97	1963	17-May-00
1880	6-Mar-94	1922	24-Apr-97	1964	13-Jun-00
1881	2-Apr-94	1923	22-May-97	1965	10-Jul-00
1882	29-Apr-94	1924	18-Jun-97	1966	6-Aug-00
1883	26-May-94	1925	15-Jul-97	1967	3-Sep-00
1884	23-Jun-94	1926	11-Aug-97	1968	30-Sep-00
1885	20-Jul-94	1927	8-Sep-97	1969	27-Oct-00
1886	16-Aug-94	1928	5-Oct-97	1970	24-Nov-00
1887	13-Sep-94	1929	1-Nov-97	1971	21-Dec-00

1.6.1 Some Comments about IDL

In using IDL it is very easy to create too many variables to the point where the computer memory capacity is exceeded. It is possible to delete old variables by using the DELVAR command. For example, if you wanted to delete the variable DATA, you would type:

IDL> **delvar, data**

There is a peculiarity about using the IDL routine FINDFILE and using the Unix symbol ∼ for the user's home directory. FINDFILE does not return any files when using the command:

IDL> **ff = findfile('∼/*')**

It is recommended to use the full path (for example '/2p/morrison/*') instead of ∼.

It is possible to change the default directory from within IDL by using the IDL command "cd". Some sample commands are:

IDL> **cd, '∼'**
IDL> **cd, '/yd3'**

The following will NOT work since it sets the default for the sub-process and then exits that process.

IDL> **$ cd /yd3** ;WILL NOT WORK

For Unix machines, it is possible to press *control-z* to place an IDL session in the background so that you can do some work from the Unix system. When you are ready to go back to the IDL session, type:

% **fg**

Be careful to remember that you have an IDL session hidden in the background since you can waste memory and corrupt the color table settings if you have multiple IDL sessions going.

1.6.2 Programs, Procedures, and Functions

An IDL main program requires a .RUN command to run, and the code within that file starts executing after successful compilation.

An IDL procedure is a kind of subroutine and has something like

```
PRO PROCEDURE_NAME, PARAM1, PARAM2
```

at the top of the file. The variables in the procedure definition can be input or output. It is possible to have keywords with a command like

```
PRO PROCEDURE_NAME, KEY1=KEY1, KEY2=KEY2
```

It can be executed with a command like:

IDL> **procedure_name, a, b**
IDL> **procedure_name, a, b, key1=c**

An IDL function is another kind of subroutine and has something like

```
FUNCTION FUNCTION_NAME, PARAM1, PARAM2
```

at the top of the file. The primary output is passed to *result* but it is possible to have parameters in the call which can receive output. It is executed with a command like:

IDL> **result = function_name(a,b)**

1.6.3 Running IDL programs

When running on a Unix system, it is important to use lower case when using the .RUN command because Unix is case sensitive. This is not necessary when doing an 'implied' compilation when accessing procedures or functions since IDL will convert to lower case for you. You cannot use upper case for a procedure to do an explicit compilation (.RUN PROCEDURE_NAME) to recompile a procedure.

2 General Yohkoh Software

2.1 Routines for Reading and Saving Data

2.1.1 RD_XDA, RD_xxx

The RD_xxx routines (RD_BDA, RD_HDA, RD_SDA, RD_WDA, RD_XDA) allow you to read the reformatted data files. Each routine has the same calling sequence, and there is a generic reading routine called RD_XDA which will read data for any of the instruments. After you have established the input file name(s) (*infil*) and which data sets to extract (*dset_arr*), you can read the data by a command:

IDL> **rd_sda, infil, dset_arr, index, data**
IDL> **rd_sda, infil, dset_arr, index, data, roadmap**
IDL> **rd_xda, infil, -1, index, /nodata**

index is a structure which describes the data. There is one index structure for each dataset. The roadmap is for all datasets in the files (not just the selected datasets). It is possible to read all datasets in the file by specifying -1 for *dset_arr*. The */nodata* switch tells the routine to only read the index and roadmap, not the data.

2.1.2 RD_ROADMAP

RD_ROADMAP will read the roadmap from a file (or list of files) by typing:

IDL> **rd_roadmap,infil,roadmap,ndset**

where *infil* is a string array of file names. The roadmap variable is the same as you get from YODAT

2.1.3 RD_QS

RD_QS will read the quasi-static portion of a single reformatted file. A sample calling sequence is:

IDL> **rd_qs, infil, qs**

2.1.4 SAVEGEN and RESTGEN

Details on SAVEGEN and RESTGEN are given in the SolarSoft Reference Manual. NOTE: It is recommended to use SAV_SDA, SAV_BDA, SAV_BSC, and SAV_HXI whenever possible for standard SXT, BCS, and HXI data sets.

2.1.5 SAV_SDA

SDA_SDA will store an index and data array in an SDA format file. The data array can be byte, integer*2, integer*4 or real*4. The dimensions of the image do not have to be the original dimensions. A sample is:

IDL> **sav_sda, outfil, index, data**
IDL> **sav_sda, outfil, index, data, qs**

where *outfil* is the name of the file you wish to store the data in. If you wish to append to an existing file, then you can use the /APPEND switch. For example:

IDL> **sav_sda, outfil, index, data, /append**

Use RD_SDA to restore the data.

2.1.6 SAV_BDA and SAV_BSC

SAV_BDA and SAV_BSC work in the same manner as SAV_SDA but for raw BDA data or BSC processed data. Sample calls would be:

IDL> **sav_bda, outfil, index, data, qs, dp_sync**
IDL> **sav_bsc, outfil, index, data**

2.1.7 SAV_HXI

SAV_HXI allows you to save HXT synthesized images. A sample call would be:

IDL> **sav_hxi, outfil, index, data**

2.1.8 SXT2FITS

It is possible to take an SXT data file or an *index* and *data* that has already been read in, and write a single FITS file for each image. The FITS header has all of the information on the date and time, the filters used, the exposure duration, the resolution, and the DP mode and rate. The following command will create a file for each image in the *data* array:

IDL> **sxt2fits, index, data**

The default file names are SF_FITSyymmdd.hhmmss and SP_FITSyymmdd.hhmmss for FFI and PFI images, respectively. If a single output file name is passed to the SXT2FITS routine, but there are several images to save, then it will append an image number to the end of the file name. In the following example, the input array is 512×512×3, so it will create the files 'flare.0001', 'flare.0002' and 'flare.0003'.

IDL> **sxt2fits, index, data, outfil='flare'**

It is possible to specify an input file name in which case all images in that file will have FITS files created. It is also possible to specify a list of the images to be saved. Some examples of these calls are:

IDL> **sxt2fits, index, data, ss=ss**
IDL> **sxt2fits, infil=infil**
IDL> **sxt2fits, infil=infil, ss=ss**

2.2 Routines for Data Selection

2.2.1 TIM2DSET

TIM2DSET will take the roadmap and return the dataset number which is closest to that time. Examples,

IDL> **dset = tim2dset(roadmap, input_times)**
IDL> **dset = tim2dset(roadmap, '23-jun-92 6:00')**

2.3 Routines for Making Time Plots

2.3.1 PLOTY

PLOTY plots light curves from all the Yohkoh instruments on the same plot with the same time axis. If you have the observing log on-line and wish to use it to make the plots, you can specify the plot times. It is recommended to plot no more than approximately 24 hours of data at a time. An example is:

IDL> **ploty, '8-may-92 15:00', '8-may-92 18:40'**

If you have identified an instrument file for which you want to see the light curve for all the other instruments, you can use the following command (it assumes that the other instrument data files exist on the same directory as the input file):

IDL> **ploty, infil='/yd5/flares/spr911115.2141'**

You can read the observing log data, and the use PLOTY with a command like:

IDL> **rd_obs, '8-may-92', '8-may-92 12:00', bcs, sxtf, sxtp, w_h**

IDL> **ploty, bcs, w_h, sxtp, w_h**

2.3.2 OBS_PlOT

TODO - needs text

2.4 Routines for Data and Information Extraction

2.4.1 The Concept Behind the GT Routines

A series of GT routines was created so that a single piece of information can be extracted from a structure, whether the structure was a 'roadmap' or an 'index' (the data are saved in a different location). The routines also allow for conversions to a string mnemonic or to physical units. It is possible to get a list of what the different values are by typing a command like:

IDL> **print, gt_filta()**

It is also possible to convert the output value to the string mnemonic by using the */STRING* switch. For example:

IDL> **print, gt_filta(roadmap, /string)**

This command only works for the routines which return a coded value. A list of all of the GT routines currently available are:

GEN	*BCS*	*HXT*	*SXT*	*WBS*
gt_conv2str	gt_blockid	gt_sum_h	gt_adoor	gt_grs1
gt_day	gt_bsc_bincal	gt_sum_l	gt_center	gt_grs2
gt_dp_mode	gt_bsc_chan	gt_sum_m1	gt_comp	gt_hxs
gt_dp_rate	gt_bsc_crate	gt_sum_m2	gt_corner	gt_rbmsc
gt_hxa	gt_bsc_flux		gt_corner_cmd	gt_rbmsd
gt_iru	gt_bsc_time		gt_dpe	gt_sxs1
gt_tfss	gt_bsc_wave		gt_entry	gt_sxs2
gt_time	gt_total_cnts		gt_expdur	
			gt_explat	
			gt_expmode	
			gt_filta	
			gt_filtb	
			gt_mbe	
			gt_obsregion	
			gt_or_expnum	
			gt_percentd	
			gt_percentover	
			gt_pfi_ffi	
			gt_pix_size	
			gt_res	
			gt_seq_num	
			gt_seq_tab	
			gt_shape	
			gt_shape_cmd	
			gt_ssl_explab	
			gt_sxt_axis	
			gt_sxt_cen	
			gt_sxt_roll	
			gt_temp_ccd	
			gt_temp_hk	

2.4.2 GT_DP_RATE

GT_DP_RATE will extract the information on the DP telemetry rate. The values are shown below.

1	Low (1 Kbits/sec)
2	Med (4 Kbits/sec)
4	High (32 Kbits/sec)

IDL> **dprate = gt_dp_rate(index)**
IDL> **dprate = gt_dp_rate(roadmap, /string)**

2.4.3 DPRATE2SEC

DPRATE2SEC will take a structure (or an integer value 1, 2 or 4 for low, medium, or high) and return the number of seconds that passes for a major frame at that telemetry rate.

1	64 sec (Low)
2	16 sec (Med)
4	2 sec (High)

IDL> **nsec = dprate2sec(index)**

2.4.4 GT_DP_MODE

GT_DP_MODE will extract the information on the DP mode. There are several modes, but the most common are:

9	Flare
11	BCS-OUT
12	Night
13	Quiet

IDL> **dpmode = gt_dp_mode(index)**
IDL> **dpmode = gt_dp_mode(roadmap, /string)**

2.4.5 GET_INFO

GET_INFO will take the roadmap or index structure and return a string describing the main observing mode parameters for each dataset.

IDL> **info_array = get_info(roadmap)**

An example for a dataset for SXT data is:

```
0 15-NOV-91  22:42:24  FL/H  NaBan/Open  Full Norm C  16   238.0   64x 64
```

The “0” is the image number in the index or roadmap which was passed to GET_INFO, “15-NOV-91 22:42:24” is the start date and time of the exposure, “FL/H” says the DP was in flare mode in high telemetry rate, “NaBan/Open” are the filters, “Full” is the pixel resolution, “Norm” says it was a normal exposure (not a dark frame), “C” is compressed data, “16 238.0” is the DP exposure level (DPE) and the expected exposure duration in milliseconds, and “64x 64” is the image shape.

An example for a dataset for SXT data when using the /LONG option is:

```
0 15-NOV-91  22:42:24  FL/H  NaBan/Open  Full Norm C  16   238.0   64x 64  100%  S13W18   0% SAA  2.6
```

The first 80 columns are the same as above. The “100%” says that all of the image was received from the S/C, “S13W18” are the heliocentric coordinates of the PFI, “0%” of the image was saturated, “SAA” signifies that it was during an South Atlantic Anomaly (SAA) passage, and “2.6” is the number of minutes before spacecraft night.

For HXT,WBS and BCS data, the routine only returns the date, time, DP mode and DP rate.

2.5 Routines for Calibration and Analysis

2.5.1 FSP_PROC [*]

FSP_PROC is a general routine for doing spectral fits. *detector* is a string variable containing the detector name, *cfile* is the input file name, the format for an input file is given in /ys/ucon/soft/mctiernan/spectral_data_format, *ofile* is an output file name. *tyspec* denotes the type of spectrum you want to fit (try doc_library, 'fsp_proc' for a complete list). *fit_pars* is the structure containing the results. Optional parameters: *sc_par* is a structure containing spectrometer channel info, *ch_dta* is a structure containing the data for each channel, *pfile* is the name for an output plot file, *sdel* is an array of channels you'd like to delete, *flux_corr* is any correction to the overall flux you might want to make, *nsigmas* tells you how many sigmas above background a “good” channel will be (default is 3.0), *noplot* suppresses screen plots.

IDL> **FSP_PROC, detector, cfile, ofile, tyspec, fit_pars**

2.5.2 FSP_11 [*]

FSP_11 is a general routine for doing spectral fits. *inpf* is a strarr containing the detector response filenames, *tyspec* denotes the type of spectrum you want to fit (try doc_library, 'fsp_11' for a complete list). Data must be put in the proper fields of the structures *fit_pars, sc_par* and *ch_dta*. *fit_pars* is the structure containing the results, *sc_par* is a structure containing spectrometer channel info, *ch_dta* is a structure containing the data for each channel. Optional parameters: *ofile* is the name for an output file, *pfile* is the name for an output plot file, *sdel* is an array of channels you'd like to delete, *flux_corr* is any correction to the overall flux you might want to make, *nsigmas* tells you how many sigmas above background a “good” channel will be (default is 3.0), *noplot* suppresses screen plots.

IDL> **FSP_11, detector, inpf, tyspec, fit_pars, sc_par, ch_dta**

2.6 Routines for Image Alignment

The routines described here can generally be used on SXT or ground-based images.

2.6.1 SXT_PREP

See the description of SXT_PREP on page 51 for details on aligning SXT images.

2.6.2 Other Routines

See the SolarSoft Reference Manual.

2.7 Accessing Secondary Databases

2.7.1 CONTACTS

CONTACTS will give information on when the ground station contacts are for a given day. The output from the command below is shown.

IDL> **contacts,'2-jun-92'**

```
             Kagoshima Space Center Contacts           Minutes of
                  Starts                      Ends       Day  Ngt  Tot
         JST               (UT)               JST
 2-JUN-92 02:56:14 ( 1-JUN-92 17:56:14)   03:07:14    3.5  7.5 11.0
 2-JUN-92 04:39:59 ( 1-JUN-92 19:39:59)   04:48:59    7.8  1.2  9.0

 2-JUN-92 20:27:14 ( 2-JUN-92 11:27:14)   20:37:14    4.0  6.0 10.0
 2-JUN-92 22:09:14 ( 2-JUN-92 13:09:14)   22:21:14    0.0 12.0 12.0
 2-JUN-92 23:52:29 ( 2-JUN-92 14:52:29)   00:04:14    0.0 11.8 11.8
```

A blank line is inserted to show that the second set of contacts is for a different series of contacts (they come in clusters of five or six per day). It is possible to get the station contacts times for DSN stations by using the */CANBERRA*, */GOLDSTONE* or */MADRID* switches. It is possible to specify an end time and output files in the following example:

IDL> **contacts, '1-jun-92', '10-jun-92', outfil='contacts.txt'**

2.7.2 PR_FEM

PR_FEM will print out Yohkoh's day and night events by typing:

IDL> **pr_fem, '1-jan'**

2.7.3 PR_EVN

PR_EVN helps you find when there is Yohkoh data available. An event is driven by a change between QUIET and FLARE mode, or when there is a data gap of more than 60 seconds. By typing:

IDL> **pr_evn, '23-jun-92'**

a list of the times that Yohkoh data are available and the number of datasets available for each instrument are listed for 24 hours starting at 23-jun-92 00:00. By typing:

IDL> **pr_evn, '15-nov-91 20:00', '17-nov-91 15:00', /flare**

all times that Yohkoh was in FLARE mode between those times is listed. By typing:

IDL> **pr_evn, '1-jan-92', '1-jan-93', /flare, /counts, mindur=5, outfil='pr_evn.results'**

the FLARE modes for 1992 are listed, and since the /COUNTS option was used, the maximum counting rate for certain WBS, HXT, and BCS channels is printed instead of the number of datasets available. It also prints the GOES classification when it is available. See the User's Guide for a sample output listing.

2.7.4 PLOT_EVN

PLOT_EVN will plot a very basic time line showing when the station contacts are available, when the SAA passages are, when the S/C day and nights are, and the periods when there is Yohkoh data available. It shows when there is FLARE and QUIET data.

2.7.5 RD_FEM

To read the FEM, EVN ,GEV or NAR structures, use a command similar to:

IDL> **rd_fem, sttim, entim, fem**
IDL> **rd_evn, index(0), index(n-1), evn**
IDL> **rd_nar, '1-nov-91', '2-nov-91', evn**

Look at the File Control Document for a description of the structures which are returned.

2.7.6 RD_OBS

A sample standard calling sequence for reading the observing log data files for all instruments is:

IDL> **rd_obs, '8-may-92', '8-may-92 12:00', bcs, sxtf, sxtp, w_h, fid**

It is possible to only read SXT full-frame data by using the one of the following command:

IDL> **rd_obs, '8-may-92', '8-may-92 12:00', bcs, sxtf, sxtp, w_h, fid, /sxtf**
IDL> **rd_obs, '8-may-92', '8-may-92 12:00', bcs, sxtf, /sxtf**
IDL> **rd_obs, '8-may-92', '8-may-92 12:00', bcs, sxtf, /nobcs**

To read the WBS data, you can use the command:

IDL> **rd_obs, '8-may-92', '8-may-92 12:00', bcs, sxtf, sxtp, w_h, /w_h**

The observing log output structures are positional and are sorted alphabetically. The observing log files must be in the $DIR_GEN_OBD, $DIR_GEN_OSF, $DIR_GEN_OSP, and $DIR_GEN_OWH directories appropriately.

2.8 Reading Exabyte Archive Tapes

See the description of GO_RDTAP in the User's Guide and the description of RDTAP on page 73 in the appendix of the Reference Guide.

3 Bragg Crystal Spectrometer (BCS)

3.1 Routines for Data Selection

3.1.1 LIST_BDA

A sample call would be:

IDL> **list_bda, roadmap, start, nrec, ss=ss**

where *roadmap* is the input.

3.1.2 SELECT_BDA

Use SELECT_BDA to plot the roadmap light curve of a selected channel and select the time period of interest using the cursors. (Note: there is an option to read the data in with SELECT_BDA if required). A sample call would be:

IDL> **.run select_bda**

where *roadmap* is the input.

3.2 Routines for Making Time Plots

3.2.1 BCS_24HR_PLOT

If you interested only in whether the BCS has seen anything, use BCS_24HR_PLOT. This works using only roadmap data and allows intervals to de selected by date (all files covering this data are then used). By default, channel 3 (Ca XIX) is plotted. Note: YODAT does NOT need to have been run before BCS_24HR_PLOT. In its default mode, BCS_24HR_PLOT uses the observing log files. Samples are:

IDL> **bcs_24hr_plot, '15-nov-91'**
IDL> **bcs_24hr_plot, '15-nov-91', chan=4**

3.2.2 PLOTT_BDA

PLOTT_BDA will make a light curve plot using either the index or roadmap. The second 'T' signifies that it is a time plotting routine. Four plots are made on the page, one for each channel. Some sample calls are:

IDL> **plott_bda, index**

IDL> **plott_bda, roadmap**
IDL> **plott_bda, roadmap(100:200), psym=10**

3.2.3 LCBDA [*]

LCBDA will make a light curve plot using either the index or roadmap. It is almost identical to PLOTT_BDA except that you can specify a channel and only get that channel plotted. Some sample calls are:

IDL> **lcbda, index**
IDL> **lcbda, roadmap, chan=1**

3.3 Routines for Making Spectral Plots

3.3.1 PLOTS_BDA

PLOTS_BDA allows for the spectra for all four channels to be plotted to one page. A sample calling sequence is:

IDL> **plots_bda, index, data**

3.3.2 PLOTBDA

The light curve and spectra from the BDA file may be plotted using PLOTBDA. This is an interactive program and it will read its own data.

IDL> **.run plotbda**

3.3.3 BCS_MULTI

Many spectra may be plotted on a page with BCS_MULTI.

IDL> **bcs_multi, index, data**
IDL> **bcs_multi, index, data, chan=1**

3.3.4 BCS_CONT

The evolution of spectra against time can be displayed as a contour map using the routine BCS_CONT. Note: The time axis of BCS_CONT is uniform, but those of DISP_BDA and GS are not.

IDL> **bcs_cont, index, data**

IDL> **bcs_cont, index, data, chan=1**

3.4 Routines for 2-D Spectral Display

3.4.1 DISP_BDA

The evolution of spectra against time can be displayed as a pseudo-image using the routine DISP_BDA. Note: The time axis of DISP_BDA is not uniform.

IDL> **disp_bda, index, data**

3.4.2 BCS_SPMOVIE

After selecting a time interval, a movie of the changing spectra for a given channel may be displayed by BCS_SPMOVIE. Note: This program will only run on an X-windows workstation.

IDL> **bcs_spmovie, index, data**
IDL> **bcs_spmovie, index, data, chan=1**

3.4.3 GS [*]

If you are interested in more detail of what spectra the BCS has observed, use GS — this routine must be run after the data has been read in with YODAT. There is an upper limit on how much data can be handled at a time (about 900 spectra), and some selection of the required time interval may be needed using the methods described above, but for the per-orbit files, this limit may not be a problem. The advantage of GS is that you can get a good idea of what has been seen in the spectra — a particularly useful tool if you are trawling for data. Warning: The time axis on the greyscale plot is not uniform!

IDL> **.run gs**

All the routines that run from GS require that spectra be selected first using the cursor routine.

IDL> **.run gs_cur**

3.5 Routines for Data and Information Extraction

3.5.1 EXT_BCSCHAN

The spectra for a single channel may be extracted with EXT_BCSCHAN

IDL> **data_out = ext_bcschan(index, data, chan)**

3.5.2 LIST_BDA

If you are interested in what modes the BCS was executing, or what the count rate in a particular channel has observed at a particular time, this can be determined by using LIST_BDA. By default the count rate for channel 3 (Ca XIX) is given in the listing. Sample calls are:

IDL> **list_bda, roadmap, start, nrec**
IDL> **list_bda, roadmap, start, nrec, chan=4**

where *roadmap* is the input.

3.5.3 GT_TOTAL_CNTS

GT_TOTAL_CNTS extracts the information from the TOTAL_CNTS field which used the actual spectra to calculate the total number of counts. The routine also normalizes so that it returns counts/sec. In the first example below, all four channels are returned, so the output is 4xN. In the second example, only channel 1 is extracted. It is possible to get a string defining the channel selected by using the *title* keyword option.

IDL> **x = gt_total_cnts(roadmap)**
IDL> **x = gt_total_cnts(roadmap,1)**
IDL> **x = gt_total_cnts(index,2,title=title)**

3.5.4 GT_BLOCKID

GT_BLOCKID returns information on how the BCS data were blocked.

IDL> **x = gt_blockid(roadmap)**

0	Normal Queue Data Block
1	Fast Queue Data Block
2	Micro Dump Block (fixed extraction)
3	Cal Data Block (fixed extraction)
4	Queue data where the modeID in the header is not recognized
5	Normal or fast queue data which have fill data (garbage)

3.6 Routines for Calibration and Analysis

3.6.1 BCS_DECOMP

The BCS data are normally compressed from a 12-bit word to an 8-bit word. BCS_DECOMP takes the compressed data array (*data*) and creates a floating point array (*ndat*) of the decompressed numbers by typing:

IDL> **ndat = bcs_decomp(data)**

3.6.2 BCS_NORM

BCS_NORM normalizes data recorded by the BCS for integration time, and extracts the Fast Queue data if present. A sample call is:

IDL> **dspec = bcs_norm(index, data)**

3.6.3 SUM_BDA

All spectra with mode ID *modeid* will be summed by SUM_BDA with the following command:

IDL> **data_out = sum_bda(index, data, modeid, nsum)**

3.6.4 PLOT_REF [*]

Overplot several spectra to show evolution of blue wing and line width.

IDL> **plot_ref, index, data, channel, dset_arr**

3.6.5 BCS_DTFAC

The dead-time correction factors can be calculated with BCS_DTFAC:

IDL> **dd = bcs_dtfac(index,dp_sync,type,stime,etime)**

3.6.6 BCS_POINT

The position of a source on the solar disk may be determined from the bin number of the resonance line with BCS_POINT. If the epoch is provided, this routine will also return the heliocentric latitude:

IDL> **dd = bcs_point(wbin [,chan=n [,date=date, etc.]])**

3.6.7 BCS_SYNTHETIC

BCS_SYNTHETIC is a simple main program to compute BCS synthetic spectra in flux units. The program will prompt the user for the BCS channel number (1,2,3,4 = Fe XXVI, Fe XXV, Ca XIX, S XV), the electron temperature (MK) and the Doppler temperature (MK). To run the program type:

IDL> **.run bcs_synthetic**

The program calls the procedure BCS_SPEC to perform the spectral calculations. It defaults to a wavelength coverage which is appropriate for mid-latitude observations. The BCS_SPEC defaults for atomic code, ionization balance and abundance will be used. The computation is made for an emission measure of EM = 10^{50} cm^{-3}.

3.6.8 BCS_SPEC

BCS_SPEC is a procedure to compute a BCS synthetic spectrum in flux units. Sample calling sequences are:

IDL> **flux = bcs_spec(Chan,Te6) ; Use Td6=Te6 and default wavelengths**
IDL> **flux = bcs_spec(Chan,Te6,Td6,Wave,Waveo=Waveo,Ainfo=Ainfo,Vel=Vel)**

where the inputs are *Chan* (the BCS channel number of a BSC structure), *Te6* and *Td6* (electron and "Doppler" temperatures in MK), and *Wave* is a vector of the edges of the BCS wavelength bins. Returned will be *flux* (assuming EM = 10^{50} cm^{-3}), *Waveo* (wavelengths at the centers of the bins), and *Ainfo* (a structure variable containing information about the calculated spectrum). *Vel* is an optional input keyword to specify the bulk velocity in km/s. More information is available in the documentation header. BCS_SPEC is called by BCS_SYNTHETIC and FIT_BSC.

Three routines control the defaults for the atomic data used in the calculation. These are GET_ELEMABUN (abundances), GET_IONBAL (ionization balance) and GET_ATOMIC (the atomic data for the line calculation).

3.7 BSC Routines

3.7.1 HELP_BSC

To get quick on-line help for various BSC procedures and functions, type:

IDL> **help_bsc**

3.7.2 MK_BSC

MK_BSC makes BSC_INDEX and BSC_DATA structures from the BDA INDEX and DATA. These structures contain instrumentally corrected BCS data for selected channels and accumulation times. A typical run would look like:

IDL> **mk_bsc,index,data,bsc_index,bsc_data,dp_sync=dp_sync**

To analyze BCS data, you would begin by using YODAT to read the BDA data file for the dates and times you are interested in. Yodat will return a BDA index structure INDEX, a BDA data array DATA, and a structure DP_SYNC which contains the DP synchronous data needed for deadtime corrections.

MK_BSC will prompt for various inputs such as start and stop times, channels to process, integration times, and which corrections (deadtime, crystal curvature, flux calibration) to apply. The DP_SYNC keyword is optional, but must be entered the first time if deadtime corrections are required. Subsequent calls to MK_BSC will use the last values of DP_SYNC data passed via this keyword.

The following keywords are also accepted:

`/LAST`	– run using the information stored in the latest BSC_INDEX.
`/DEF`	– run using the default settings.
`/WID`	– run via a widget interface.
`/BSA`	– run to make BSA index only (do not make BSC_DATA).
`/SAVE`	– save BSC (or BSA) index (and data) in a file. (The file will be named BSCyymmdd.hhmm, based on the time of the first data interval).
`/VERBOSE`	– echo extra information about SAA/NIGHT or MODEID/BLOCKID
`BDAFILE`	– input BDAFILE to read.

The instrumentally corrected data will be output to a new set of structures BSC_INDEX and BSC_DATA. The data in BSC_DATA are stored as 256 element floating point arrays in fields with the following definitions:

`.COUNTS`	– accumulated counts (per detector bin)
`.BIN`	– physical detector position
`.ERROR`	– spectral uncertainties (in counts)
`.WAVE`	– nominal wavelength array (Å)
`.FLUX`	– flux-calibrated data (ph cm^{-2} s^{-1} Å$^{-1)}$
`.WAVE_FIT`	– wavelength array for the fitted spectra (Å)
`.FLUX_FIT`	– fitted spectra

Not all the above fields will be present. The fields that are present will depend on the corrections requested in MK_BSC. The actual data in each field array will occupy the first 0:NBINS-1 locations, where NBINS is the number of actual detector bins containing valid data.

The value of NBINS is stored in BSC_INDEX.BSC.NBINS.

3.7.3 SEL_BSC

SEL_BSC extracts wavelength and flux fields from BSC_DATA.

IDL> **sel_bsc,bsc_index,bsc_data,new_index,new_data,wave,flux,eflux**

The selected index and data structures are returned in NEW_INDEX and NEW_DATA. The wavelength, flux, and uncertainties in the flux are returned in the arrays WAVE, FLUX, and EFLUX.

The following keywords are also accepted:

`CHAN (in)`	- BCS channel (def is all)
`TSTART (in)`	- start time to extract (UTPLOT format)
`TEND (in)`	- end time to extract
`BIN (out)`	- Physical detector positions
`COUNTS (out)`	- observed counts (per bin) in the accumulation interval
`SS (out)`	- indices matching requested channel
`FWAVE (out)`	- fitted wavelength array
`FIT (out)`	- fitted spectrum array
`SFIT (out)`	- fitted second component array

3.7.4 LIST_BSC

LIST_BSC lists times, total counts, and accumulation times of BSC spectra.

IDL> **list_bsc,bsc_index,chan**

where CHAN is the desired channel to list.

3.7.5 SAV_BSC

To save BSC index and data to a BSC file, use:

IDL> **sav_bsc,'bsc_file_name',bsc_index,bsc_data**

Note that MK_BSC will also save the data if the */FILE* switch is set.

3.7.6 RD_BSC

To read a BSC file, use:

IDL> **rd_bsc,'bsc_file_name',ss,bsc_index,bsc_data**

This procedure will return the BSC index and data structures. SS is a vector with indices of a particular subset of data within the BSC file. For example, using SS=INDGEN(10) will extract the first 10 data sets from the BSC file. To extract ALL the data sets, use $SS = -1$.

3.7.7 BSC2BSD

To convert a BSC file to a BSD file, use:

IDL> **bsc2bsd,'bsc_file_name','bsd_file_name'**

The BSD file name is optional, and will be created from the BSC file name if not supplied.

3.7.8 MAP_BSC

MAP_BSC plots a pseudo-map of BSC spectra by stacking successive spectra.

IDL> **map_bsc,bsc_index,bsc_data**

The following keywords are accepted:

`CHAN`	BCS channel to plot.
`XSPACE`	the value by which to shift successive spectra in the x-direction. [def=0]
`YSPACE`	the value by which to shift successive spectra in the y-direction. [def=0]
`TSTART`	the time of the first spectrum to plot (with date optional). [sample format,TSTART='9:32', def = first spectrum in BSC_DATA]
`TEND`	the time of the last spectrum to plot. [def = last spectrum in BSC_DATA]

3.7.9 LC_BSC

LC_BSC plots light curves of selected spectral regions.

IDL> **lc_bsc, bsc_index, bsc_data, t, y, chan=chan,/flux**

This program first plots a spectrum at the peak of the flare. The user can select a spectral region with the cursor. The light curve derived from the sum of wavelength points within this region is returned in the array Y. The time (UT) is returned in T. If the BSC data are flux calibrated, then using the $FLUX$ keyword will return the light curve in units of photons cm^{-2} s^{-1}. The optional keyword *over* will overplot successive light curves.

3.7.10 WBDA

WBDA provides a simple widget interface that allows browsing of BDA files. It is invoked by:

IDL> **wbda**

The program will provide a set of menus from which BDA files can be selected either by date, filename, or directory location. WBDA will plot light curves and spectra for a specific channel, accumulate spectra for selected integration times, and apply dead-time, curvature, and flux corrections. The BDA INDEX and DATA variables produced by YODAT can be also input directly to WBDA:

IDL> **wbda,index,data**

3.7.11 GT_BSC_BINCAL

This function returns a structure containing miscellaneous calibration information for specified BCS channels: For example, for channel 1,

IDL> **bincal=gt_bsc_bincal(1)**

will return the following structure:

.CHAN	- channel number
.NBINS	- number of bins per channel
.MODEID	- mode id
.WAVE	- nominal wavelength array (Å)
.PHYSPOS	- physical detector position array
.FLUXFAC	- conversion factor array for counts/bin to photons/cm/Å
.SENSIT	- mean sensitivity array for grouped bins
.DBIN	- detector bin width
.W0	- start wavelength of spectrum
.DW	- dispersion (angstrom/bin)
.BINARC	- conversion factor for arcmin to bin (bins/arcmin)
.GAUSSW	- FWHM Gaussian detector width (Å)
.RWID	- FWHM rocking width (Å)
.EFAREA	- effective area (cm^2)
.NSTART	- first valid bin
.NEND	- last valid bin
.VERSION	- version number of BCS calibration file used

3.7.12 FIT_BSC

FIT_BSC fits synthetic spectra to observed spectra in BSC_DATA:

IDL> **fit_bsc,bsc_index,bsc_data,fit_index,fit_data,chan=chan**

In the above call, the fitted results are returned in the new structures FIT_INDEX and FIT_DATA for the channel specified by the keyword CHAN. If CHAN is not specified, then all channels will be fit (not recommended).

The FIT_DATA structure contains two additional fields:

.FLUX_FIT	a 256-element floating point array containing the fitted model spectrum matching the observed spectrum in the .FLUX field
.WAVE_FIT	a 256-element floating point array containing computed wavelengths of the fitted model spectrum in .FLUX_FIT.
.FLUX_FIT2	a 256-element floating point array containing the fitted model spectrum corresponding to a secondary velocity shifted spectrum in the .FLUX field

The FIT_INDEX structure contains an additional .FIT field that contains important fit information (parameters fit, number of iterations, chi squared, etc).

FIT_BSC accepts the following optional keywords:

`SS`	index selection vector or scalar (e.g. SS=[45,46] to fit only spectra 45 and 46)
`TSTART`	UT string (e.g. '9:45'). Only spectra at or after TSTART will be fit.
`TEND`	Only spectra at or before TEND will be fit.
`LRANGE`	2-element wavelength vector range (Å) within which to limit line fitting [Default is entire range]
`CRANGE`	2-element wavelength vector range (Å) within which to limit continuum fitting [Default is entire range]
`TE6`	input starting guess for electron temperature (MK) [Default is to use internal values]
`TD6`	input starting guess for resonance line Doppler broadening (MK) [Default is to infer from observed width]

`ATTEMPT` level of fit attempt

1. solve only for primary emission measure (EM50) and line to continuum normalization factor (CNORM). Electron temperature and (TE6) and total Doppler broadening (TD6) fixed.
2. solve for TE6, TD6, EM50, CNORM.
3. solve for primary+secondary shifted component.

`/BLUE`	equivalent to attempt=3
`/NEWGUES`	set to NOT use results of previous fit as starting input for fit to next spectrum
`/PLOT`	set to plot progress of fits
`/VERB`	set for more information about fitting progress
`/LAST`	set to use last converged parameters as starting guess for fitting next spectrum.

For example, to fit a Ca XIX spectrum at UT='09:04' use:

IDL> **fit_bsc,index,data,findex,fdata,chan=3,tstart='09:04'**

To fit a blueshifted secondary component:

IDL> **fit_bsc,index,data,findex,fdata,chan=3,tstart='09:04',/blue**

3.7.13 BCS_BROAD

The relation between BCS line broadening and velocity can be computed with BCS_BROAD. Some example calling sequences are:

IDL> **Td6 = bcs_broad(Chan,TDoppw=TDoppw)** ; Return temperature in (MK)
IDL> **Vel = bcs_broad(Chan,Td6=Td6,/vel)** ; Return velocity (km/s)
IDL> **TDoppw = bcs_broad(Chan,Td6=Td6,/Gauss)** ; Return Doppler FWHM (Ang)

where $TD6$ is the equivalent Doppler Temperature (MK) and $CHAN$ is the BCS channel number. The parameter $TDOPPW$ does not include the contribution due to the crystal (Lorentz) broadening, but is included by BCS_BROAD when it makes its calculations, based on the value of $CHAN$. There are several additional options available and there is a keyword to return the crystal rocking curve width. Most conversions between BCS line width and velocity or between line width and temperature can probably be accomplished using this routine.

3.7.14 PLOT_BSC

The results of MK_BSC and FIT_BSC can be examined with PLOT_BSC.

IDL> **plot_bsc,bsc_index,bsc_data**

PLOT_BSC will plot light curves and spectra for user selected channels and times, as well as fitted spectra from FIT_BSC.

Optional keywords:

`CHAN`	- channel to plot
`SS`	- selected indicies of spectra to plot (default is all)
`/NOWID`	- to not use widget mode (def is widgets, if more than 1 spectrum)

The following keywords apply only when /NOWID is set:

`/EBAR`	- oplot spectral uncertainties
`/HC`	- make a hard copy (single spectrum only)
`/PS`	- make a PostScript file (single spectrum)

The following keywords apply only when /NOWID is set, and BSC_INDEX and DATA contain fitted spectra:

`/BLUE`	- to oplot second blueshift component (if fitted by FIT_BSC)
`/VUNIT`	- to print fitted Doppler widths in km/s units (default is MK)
`/NOFIT`	- inhibit oplot of fitted spectra (default is oplot fit)
`/NOOBS`	- inhibit plot of observed spectra (default is plot obs)

3.7.15 FIT_BSC_PLOT

FIT_BSC_PLOT produces time series plot results from FIT_BSC. The routine will plot total counts, electron temperature, emission measure, and turbulent velocity as a function of time. Example calling statements are:

IDL> **fit_bsc_plot,bsc_index**
IDL> **fit_bsc_plot,bsc_index,tstart=tstart**
IDL> **fit_bsc_plot,bsc_index,/hc** ; Produce hardcopy output

Consult the document header for additional options.

3.7.16 MK_FIT_BSC_EPS

MK_FIT_BSC_EPS will produce a BCS spectrum in portrait mode. It optionally will specify the output file to be an EPS (encapsulated PostScript file). Some sample calls are given below:

IDL> **mk_fit_bsc_eps, index, data, /hc** ; Send plot to the printer

IDL> **mk_fit_bsc_eps, index, data, /ps** ; Make file but don't print
IDL> **mk_fit_bsc_eps, index, data, /eps** ; Create an EPS file
IDL> **mk_fit_bsc_eps, index, data, /noatomic** ; Suppress atomic code label

MK_FIT_BSC_EPS sets up the parameters to call BCS_SPEC_PLOT correctly.

The default X and Y size is about right for an A4 page. The optional *xoffs=xoffs* and *yoffs=yoffs* enable the user to position the plot at the desired location on the page.

4 Hard X-Ray Telescope (HXT)

4.1 Routines for Making Time Plots

4.1.1 PLOTT_HDA

PLOTT_HDA will make a light curve plot of all four channels using either the index or roadmap. The second 'T' signifies that it is a time plotting routine. Four plots are made on the page, one for each channel. Some sample calls are:

IDL> **plott_hda, index**
IDL> **plott_hda, roadmap**
IDL> **plott_hda, roadmap(100:200), psym=10**

4.2 Routines for 2-D Spectral Display

4.2.1 DISP_HDA

The evolution of sensor intensity against time can be displayed as a pseudo-image using the routine DISP_HDA. Note: The time axis of DISP_HDA is not uniform.

IDL> **disp_hda, index, data**

4.3 Routines for Data and Information Extraction

4.3.1 GT_SUM_L

These routines will extract the average counts/sec of all 64 sensors. The input can be roadmap, index, or observing log and it will get the proper structure tag and decompress it properly to return counts/sec/sensor. It is possible to get a string defining the channel selected by using the *title* keyword option.

IDL> **y = gt_sum_l(roadmap)**
IDL> **y = gt_sum_l(w_h, title=title)**

4.3.2 GET_INFO

See the description in the "General Yohkoh Software" chapter on page 17.

4.4 Routines for Calibration and Analysis

4.4.1 HXT_QLOOK

HXT_QLOOK makes a series of HXT quick-look images (i.e. a quick-look movie). This program is much faster than the MEM code, but should be used only for quick-look movies. It is very useful for verifying positions etc. before running a long MEM job.

IDL> **hxt_qlook, index, data, movie, hxi_index**
IDL> **hxt_qlook, index, data, movie, hxi_index, patterns**

index	HXT index from an HDA file
data	HXT data from an HDA file
patterns	the sub-collimator modulation patterns. Computed if not supplied. The computation is slow, so if you plan to make a series of movies, use patterns to receive the computed patterns the first time you call HXT_QLOOK and then use patterns to pass the modulation patterns back to the routine on subsequent calls to HXT_QLOOK. You must recompute the modulation patterns whenever you use a different pointing.
movie	a data cube of the HXT images (output)
hxi_index	an index describing various parameters of the images. The time in the index is the start of the accumulation period (output).
/new_mod_patterns	Force a new computation of the modulation patterns, even if the patterns parameter is set. Normally, if patterns is set, the modulation patterns are not computed. You must recompute the modulation patterns whenever you use a different pointing.
/show_images	show each image as it is computed
/quiet	don't print any diagnostics
/verbose	print lots of diagnostics
smooth	smoothing parameter which determines how much the inverse computation of the image is smoothed. Default = 0.1.

4.4.2 HXTPRO

See the User's Guide for a description of this program.

4.4.3 HXT_IMG

See the User's Guide for a description of this program.

4.4.4 GET_HXT_POS

The routine GET_HXT_POS will return the HXT address of the flare location. It will perform the following checks in this order until it finds a match.

1. Read the $DIR_HXT_CAL/hxt_flare_pos.txt file to see if the time is in that list
2. Use the GOES heliocentric location to determine the address
3. Use an SXT image index to define the address
4. Pass in heliocentric coordinates and convert them
5. Ask for user to specify HXT address directly

Sample calls are:

IDL> **xy0 = get_hxt_pos('15-Nov-91 22:37')**
IDL> **xy0 = get_hxt_pos(index(20), helio='N32W23')**
IDL> **xy0 = get_hxt_pos(index(20), status=status)**

The optional keywords */get_sxt, /get_goes* and *helio* will force the calculation of the function to use that technique. The keyword option */nouser* should be used when running in batch mode as it will inhibit asking the user for input. The *status* output is a −1 if no match has been found, 1 for HXT flare list, 2 for SXT image, 3 for GOES event list, 4 for heliocentric coordinate passed in, and 5 for user input.

4.4.5 HXT_FSP and HXTBOX_FSP [*]

HXT_FSP is the same as HXS_FSP, but for HXT spectra. See HXS_FSP for a better description. Index and data come from an hda file.

IDL> **HXT_FSP, index, data, fit_pars**

HXTBOX_FSP fits spectra to a given box, chosen by LCUR_IMAGE, for a set of HXT images in the HXI structure; index and data come from and hxi file of deconvolved images.

IDL> **HXTBOX_FSP, index, data, fit_pars**

4.4.6 SXTHXT_FSP and SXTHXTBOX_FSP[*]

SXTHXT_FSP fits combination SXT and HXT spectra, no spatial resolution! *index* and *data* are uncompressed SXT images, *hindex* and *hdata* are uncompressed HXT data, from an HDA file.

IDL> **SXTHXT_FSP, index, data, hindex, hdata, fit_pars**

SXTHXTBOX_FSP fits SXT-HXT spectra for a box chosen using LCUR_IMAGE. *hindex* must be the HXI structure, and *hdata* are HXT images. You should be sure of alignment beforehand.

IDL> **SXTHXTBOX_FSP, index, data, hindex, hdata, fit_pars**

4.4.7 HXT_DECOMP

HXT_DECOMP will decompress the HXT data. A sample call is

IDL> **odata = hxt_decomp(data)**

5 Soft X-Ray Telescope (SXT)

5.1 Routines for Data Selection

5.1.1 SSWHERE

SSWHERE is a widget-driven program that creates an array of subscripts (*ss*) that fulfill a set of criteria that you define to select SXT images. For example to make an *ss* array for YODAT where you want to select images from a data set represented by roadmap, you merely type:

IDL> **ss=sswhere(roadmap)**

SSWHERE works on both the roadmap and index structures. Available selection criteria are:

- completeness of the image
- SXT mode (normal/dark image/calibration)
- pixel resolution
- compression mode
- filters
- exposure DPE
- DP mode
- field of view location (see lower buttons)
- time period (see lower buttons)

If you get a "No matching Data" response, check that you have selected all possible options.

5.1.2 SHOW_OBS3

SHOW_OBS3 plots a time line summary of the SXT images that are in the index or roadmap. In addition to plotting a tick for each image available, it shows the DP mode, DP rate, where the S/C days are, and where SAA passages occur. Some sample calls to SHOW_OBS3 are:

IDL> **show_obs3, roadmap**
IDL> **show_obs3, index**

It is possible to select images using SHOW_OBS3 by clicking on diagonally opposite corners of a box surrounding the images you wish to select. The calling sequence for that is:

IDL> **show_obs3, roadmap, sel, ss**

where *sel* is a returned logical array the same length as roadmap, and is set true for all dsets selected, and *ss* is the returned subscripts of the selected datasets. A few examples on how to use the *sel* variable are:

IDL> **ss = where(sel)**
IDL> **ss = where(sel and (gt_dp_mode(roadmap) eq 9))**

5.1.3 SHOW_OBS4

The calling sequence of SHOW_OBS4 is the same as SHOW_OBS3, except there are a few additional options in SHOW_OBS4. Run SHOW_OBS4 with one of the following commands:

IDL> **show_obs4, roadmap**
IDL> **show_obs4, index**

It is possible to click on a time range for a closer look by:

1. Clicking with the left button at the start time
2. Clicking with the right button at the end time
3. Clicking with the left button on the box in the lower right
4. If you wish to exit, click with the middle button somewhere on the plot

It is possible to select several regions using the following steps.

1. Click with the right button on the box in the lower left (Select Option). A square box should appear on the screen.
2. Click and hold the left button on the lower left corner of the box and drag the box to the lower left corner of the region you want to select, then release.
3. Click and hold the middle button on the upper right corner of the box and stretch the box to encompass the images you want to select.
4. Repeat steps 2 and 3 until you have positioned the box where you want it.
5. Click on the right button to exit the selection option when you have successfully positioned the box.
6. Now you can exit SHOW_OBS4 by clicking on the middle button, or you can select another set of images by running steps 1 through 5 again.

5.1.4 PLOT_FOV

It is possible to make a simple display of the sun and to overlay the box showing the field of view size and location by using PLOT_FOV. The index, roadmap or observing log entry can be used with this routine. A sample call is:

IDL> **plot_fov, index, /box**

5.2 Routines for Making Time Plots

5.2.1 LCUR_IMAGE

See the description in the "General Yohkoh Software" chapter on page **??** or the description in the User's Guide.

5.3 Routines for Image Display and Enhancement

5.3.1 STEPPER

See the description in the "General Yohkoh Software" chapter on page **??** or the description in the User's Guide.

5.3.2 XY_RASTER

XY_RASTER makes a mosaic of several images and prints the times in the corner of each image. Sample calls are:

IDL> **xy_raster, index, data**
IDL> **xy_raster, index, data, factor**

where *factor* is the rebin factor (a value of 1 will perform no rebinning). If there are too many images, the program will display the number of images which will fit on a page, and will prompt the user for when to move to the next page. The keyword */notimes* will inhibit the time labels, */noscale* will inhibit image scaling (it will use TV instead of TV_SCL), */individual_scale* will scale each image individually.

5.3.3 DISP_MONTH

The routine DISP_MONTH will display a month of SXT images in a calendar form (assuming you have the SFM database on-line). A sample call is:

IDL> **disp_month, 'oct-93'**

5.3.4 UNSHARP_MASK

UNSHARP_MASK is a basic image enhancement routine which uses the unsharp mask method. The calling sequence is:

IDL> **dataout=unsharp_mask(index, data)**

where *data* and *dataout* are 2D arrays, and *index* is the index entry that corresponds to *data*. If no keywords are given the defaults are used.

Optional Keyword Parameters:

`smooth`	Size of smoothing box, integer.
`lowdelt`	Lower δ cutoff, a percentage of minimum, decimal 0.xx.
`updelt`	Upper δ cutoff, a percentage of maximum, decimal 0.xx.
`lowint`	Saturation cutoff, don't add back when signal is BELOW a given percentage of the saturation value, decimal 0.xx.
`upint`	Saturation cutoff, don't add back when signal is ABOVE a given percentage of the saturation value, decimal 0.xx.
`deltcoef`	Multiplier for δ field, float x.x.

The keyword parameters allow you to customize the enhancer, though they are all assigned default values if omitted in the call. These defaults depend on the images resolution, and may be changed as we gain experience.

5.3.5 DE_SPIKER

The DE_SPIKER function returns a cleaned image by removing spikes at a specified input level. The technique used is to difference a smoothed image and the input image and find all pixels that deviate by more than a given threshold.

IDL> **clean_img = de_spiker(image, thresh)**
IDL> **clean_img = de_spiker(image, thresh, /neg)**
IDL> **clean_img = de_spiker(image, thresh, indices=indices)**

where, *image* is the input data array, *thresh* is the specified spike amplitude to remove. Optional input/output parameter *indices* contains the indices of spikes at the level specified, and optional input switch */neg* requests that "dark holes" be removed.

5.3.6 SXT_GRID

Draws heliocentric grid over an image. It reads the ATT file to get information on the spacecraft pointing, and also calls GET_RB0P to get the solar radius. A sample call could be:

IDL> **tvscl, data(*,*,0)**
IDL> **sxt_grid, index(0)**

The keyword option */read* will allow the user to move the cursor around the image and to display the heliocentric coordinates. If */angle* is used with */read*, then the value displayed is the angle relative to the sun center, which is useful to be used with SXT_PREP.

5.4 Routines for Data and Information Extraction

5.4.1 GET_INFO

See the description in the "General Yohkoh Software" chapter on page 17

5.4.2 EXT_SUBSET [*]

EXT_SUBSET extracts a subset (in x and y) of a data cube and creates a new data and index array (*data_out* and *index_out*) by typing:

IDL> **ext_subset,index_in,data_in,index_out,data_out**

It is a mouse driven program.

5.4.3 GT_FILTA

GT_FILTA uses an index or roadmap structure to determine the setting of the A filter:

IDL> **a = gt_filta(index)**

The filter setting will be returned as an integer (1-6) where

1	Open	Op	Op
2	Optical - narrow band	NaBan	NB
3	Quartz defocusing lens	Quart	Qz
4	Diffuser	Diffu	Df
5	Optical - Wide band	WdBan	WB
6	8% neutral density X-ray filter	NuDen	ND

The last two columns show what GT_FILTA returns when using the */string* and */short* options. The routine GET_INFO uses the */string* option. It is possible to get a string type description of the filters used by typing:

IDL> **b = gt_filta(roadmap, /string)**

You can get a listing of these at any time by typing the following:

IDL> **print,gt_filta()**

5.4.4 GT_FILTB

GT_FILTB works exactly like GT_FILTA but works for the X-ray filters

1	Open	Open	Op
2	Thin Aluminum (1265 Å)	Al.1	A1
3	Dagwood Sandwich	AlMg	AM
4	Beryllium (119 μm)	Be119	Be
5	Thick Aluminum (11.5 μm)	Al12	A2
6	Magnesium (2.5 μm)	Mg3	Mg

The last two columns show what GT_FILTB returns when using the */string* and */short* options. The routine GET_INFO uses the */string* option.

5.4.5 GT_PFI_FFI

GT_PFR_FFI extracts the information on whether the image is a partial frame image (PFI) or a full frame image (FFI). Even values are PFI, odd values are FFI in general.

0	PFI
1	FFI
2	PFI Observing Region
3	FFI Patrol Image

The */ffi* keyword option can be used, and it will return a 1 if the image is an FFI and return a 0 if it is a PFI (PFI strip or observing region). The */true_ffi* keyword option will return a 1 if the image is an FFI and return a 0 if it is a PFI (PFI strip or observing region). The */true_ffi* option will also check if it is an FFI strip or if it is a PFI which was extracted from an FFI.

5.4.6 GT_RES

GT_RES extracts what resolution each SXT image was taken at by typing:

IDL> **res = gt_res(roadmap)**

It outputs the resolution as a integer where:

0	full resolution
1	half resolution
2	quarter resolution

It is possible to get a string type description of the resolution by typing:

IDL> **res = gt_res(roadmap, /string)**

5.4.7 GT_COMP

GT_COMP extracts what compression was used for each SXT image was taken at by typing:

IDL> **comp = gt_comp(roadmap)**

It outputs the compression as a integer where:

0	Compressed (see routine SXT_DECOMP)
1	Low 8 bits (of 12 bits)
2	High 8 bits (of 12 bits)

5.4.8 GT_DPE

GT_DPE extracts what DP exposure (DPE) level was used. The DPE is a function of the shutter exposure level (MBE) and the neutral density filter use. It is possible to return the effective exposure duration in milliseconds by using the */conv* switch).

IDL> **dpe = gt_dpe(index)**
IDL> **msec = gt_dpe(roadmap, /conv)**

The effective exposure for each of the DPE values are shown in the following table. For DPE values above 22, the effective exposure is

```
exposure = (DPE-21) * 0.25 seconds
```

DPE	Effective Exposure	MBE	Mask	DPE	Effective Exposure	MBE	Mask
0	0.08 msec	0	with ND	19	668.00 msec	12	
1	0.23	1	with ND	20	948.00	13	
2	0.96	0		21	1.338 sec	14	
3	1.37	2	with ND	22	1.888	15	
4	2.88	1		23	2.668	16	
5	3.06	4	with ND	24	3.778	17	
6	4.67	5	with ND	25	5.338	18	
7	6.28	6	with ND	26	7.548	19	
8	9.50	7	with ND	27	10.678	20	
9	17.00	2		28	15.108	21	
10	28.00	3		29	21.358	22	
11	38.00	4		30	30.208	23	
12	58.00	5		31	42.718	24	
13	78.00	6		32	60.418	25	
14	118.00	7		33	85.438	26	
15	168.00	8		34	120.828	27	
16	238.00	9		35	170.878	28	
17	338.00	10		36	241.668	29	
18	468.00	11					

5.4.9 GT_MBE

GT_MBE extracts what mail box exposure (MBE) level was used. This is the actual shutter exposure level commanded. It is possible to convert to commanded shutter duration in milliseconds by using the */conv* switch).

IDL> **mbe = gt_mbe(index)**
IDL> **shutdur = gt_mbe(roadmap, /conv)**

5.4.10 GT_EXPDUR

GT_EXPDUR will calculate the effective exposure duration (not the commanded DPE) of a series of images given an index structure to work with. You can get an array of exposures (exps) in msec by typing one of the following. In the second example, it returns the shutter duration (which is also the CCD integration period). For dark images the value returned is the number of milliseconds the CCD integrated for which is in multiples of 8 msec (8 msec is the minimum for dark current images).

IDL> **expos = gt_expdur(index)**
IDL> **expos = gt_expdur(index, /shut_dur)**

5.4.11 GT_EXPLAT

GT_EXPLAT will calculate the latency time between the time which is stored in the index and the actual time the integration started. The value returned is in milliseconds and is typically between 50 and 100.

IDL> **explat = gt_explat(index)**

5.4.12 GT_EXPMODE

GT_EXPMODE routine returns the exposure mode. The values it can return are:

0	Normal shuttered exposure
1	Dark image (shutter stays closed)
2	Calibration image (shutter open during readout)

IDL> **expmode = gt_expmode(index)**

5.4.13 GT_CORNER

GT_CORNER will return the full resolution IDL (FRE) coordinates of the center of the pixel in the lower left hand corner.

IDL> **corners = gt_corner(roadmap)**
IDL> **x0 = gt_corner(index, /x)**

5.4.14 GT_CENTER

The output from GT_CENTER can be the location of the center of the field of view in (a) full resolution IDL (FRE) pixels, (b) heliocentric coordinates, (c) the angle E/W and N/S from the center of the sun in arcseconds, or (d) the HXT equivalent pixel coordinates. The default is to read the ATT S/C pointing database for the true de-jittered S/C pointing. It is possible to use the commanded S/C pointing information by using the /CMD option, which is much faster and only of slightly reduced accuracy.

IDL> **pix = gt_center(index)**
IDL> **helio = gt_center(index, /helio)**
IDL> **helio = gt_center(index, /helio, /cmd)**
IDL> **ang = gt_center(roadmap)**

where *ang(0,*)* is arcseconds east/west with east being negative and *ang(1,*)* is arcseconds north/south with north being positive. *helio* is in degrees, and *pix* is in full resolution pixels. It is possible to get the heliocentric coordinates in a string of the form "N12W23" by using the /STRING option.

5.4.15 GT_SXT_AXIS

The optical axis of the X-ray telescope has been determined from pre-launch measurements of the effective area as a function of off-axis angle. The analysis is described more fully in SXT Calibration Note 36 (R. Fuller, Jan-94). The relative offset between the X-ray optical axis and the optical axis in the Wide band or Narrow band filters was determined by Tom Metcalf (22-Oct-92). The combined results are summarized below:

```
                    Optical Axis (CCD Full-Res Pixels)
                          x            y

     X-ray                515          633
     Narrow Band (NaBan)  514.64       634.47
     Wide Band   (WdBan)  515.24       634.26
```

The values can be quickly acquired on line using GT_SXT_AXIS. The next three lines will reproduce the values in the table above. Note that the routine also accepts an index or roadmap data structure and that the filter will be determined from the SXT structure.

IDL> **print,gt_sxt_axis(0)** ; 0=X-ray
IDL> **print,gt_sxt_axis(1)** ; 1=NaBan
IDL> **print,gt_sxt_axis(2)** ; 2=WdBan
IDL> **print,gt_sxt_axis(1,/rel)** ; NaBan relative to X-ray

The sign convention can be understood if you remember that the NaBan image is Northeast of the X-ray image.

The X-ray optical axis has a somewhat large uncertainty, since the effective area function is broad near the center of the field of view of the telescope. The relative precision between the optical and X-ray axes has been determined to better than 0.4 Full-Res pixels (see the Instrument Guide).

The */rel_xray* keyword option can be used to return the relative offset to the X-ray axis.

5.4.16 GT_PIX_SIZE

The angular size of the SXT full-resolution pixel was determined from pre-launch focal length measurements to be 2.45280 arc sec for the X-ray telescope. Half- and quarter-resolution images are twice and four times this value, respectively. A routine to provide this number is GT_PIX_SIZE which is called as:

IDL> **print,gt_pix_size()**
IDL> **print,gt_pix_size(index)**

If the SXT index or roadmap is passed in, the routine will return the appropriate pixel size in arc sec for the resolution of the data. The pixel size of the optical images is thought to be within 1% of the X-ray pixel size.

5.4.17 GT_TEMP_CCD

GT_TEMP_CCD returns the temperature of the CCD in degrees Celsius.

IDL> **temp = gt_temp_ccd(index)**

5.4.18 GT_TEMP_HK

GT_TEMP_HK returns the temperature measured by the platinum resistance thermometer (PRT) on the spacecraft and SXT instrument. There are 16 temperatures saved in the SXT index.

IDL> **temp = gt_temp_hk(index, 0, title=title)**

5.4.19 SXTPNT_SUM

The routine SXTPNT_SUM will read the observing log and give a summary of the unique PFI pointing for a given time period. It groups the summary listing based on each unique pointing location over time. See the User's Guide for a sample listing. For the command:

IDL> **sxtpnt_sum, '5-oct-93 13:00', '6-oct-93 1:00'**

it will produce a file in your home directory with the name "sxtpnt_sum.txt" (unless you specify the output file name with the *outfil* keyword option).

5.4.20 PR_SXTOBS

The routine PR_SXTOBS will read the observing log and list each and every FFI and PFI image taken for the times requested. The format of the output is simply the output of GET_INFO which is described in detail in the Reference Guide. See the User's Guide for a sample listing. The keyword */long* changes the formatting and information displayed.

IDL> **pr_sxtobs,'15-nov-91 22:00','15-nov-91 22:05'**
IDL> **pr_sxtobs,'15-nov-91 22:00','15-nov-91 22:05',/long**

5.4.21 PFI_LOC

PFI_LOC will display the SXT PFI location. There are two modes which can be used when calling PFI_LOC. If the first parameter is a date and time string, it will read the observing log and make a simple plot showing the location relative to the sun's limbs. If the first parameter is an SXT index, then it assumes that you have displayed this full frame image, and it reads the observing log and puts a box on the image where the PFI is located. Some sample calls are:

IDL> **pfi_loc, '23-jun-93 1:00'**
IDL> **pfi_loc, timstr, hours=2**
IDL> **pfi_loc, index**
IDL> **pfi_loc, index, pfi_struct=sxtp**
IDL> **pfi_loc, index, '6-nov-93 3:30'**
IDL> **pfi_loc, index, '6-nov-93 3:30', /hour**

It is possible to use the *hours* keyword option to specify the number of hours plus and minus the full frame time for which you wish to have PFI locations displayed. The *pfi_struct* option exists so that you can pass a PFI roadmap or index structure directly to the PFI_LOC to be displayed over the image. If a second positional parameter is passed, the routine will get the PFI locations for that time and overlay them on the image.

When using STEPPER, the "o" option will call PFI_LOC automatically for you, taking into account any rebinning which has been performed on the image.

5.5 Routines for Calibration and Analysis

5.5.1 SXT_PREP

SXT_PREP can perform a wide variety of the standard processing required to use SXT data. The User's Guide has a description of the capabilities and some of the fundamentals, and it should be read before reading this section.

The input specifying which images to process can be either an *index* and *data*, or it can be the filenames *infil* and dataset numbers *dset_arr*. Some sample calling sequences are:

IDL> **sxt_prep, input1, input2, index_out, data_out**
IDL> **sxt_prep, index, data, index_out, data_out**
IDL> **sxt_prep, infil, dset_arr, index_out, data_out, $**
IDL> **helio=[-9.8,-20.3], date_helio='14-JUN-92 02:37:41'**
IDL> **sxt_prep, index, data, index_out, data_out, /helio, ref_image=10**
IDL> **sxt_prep, index, data, index_out, data_out, ref_image=spr_index(0)**
IDL> **sxt_prep, index, data, index_out, data_out, /roll**

A description of some of the keywords follows:

—————— Calibration Parameters ——————

nocorrections	If set, then LEAK_SUB, and therefore DARK_SUB, are not called.
dc_scalar	If set, then a simple scalar value is used for dark current subtraction. Default is to take the image closest in exposure.
dc_interpolate	If set, then perform interpolation on the dark frame images to get a better background subtraction. Default is to take the image closest in exposure.
upper_only	If set, only flag +1 pixels above (in a column) the saturated area (not the −1 pixels) [see call to SXT_SATPIX].
sfd_corr	If set, then the input image is an SFD image. Use the SFD decompression, perform the registration, and recompress using the SFD compression.

—————— Alignment Parameters ——————

register	If set, then perform the registration and correct only for the S/C jitter. Using the HELIO or SUNCOORD options selects this option also.
ref_image	A single roadmap or index structure which will be used to define the sun or heliocentric coordinates for alignment. It can be the image number (subscript) of the images being extracted which should be used (not the absolute dataset number within the images listed in "infil")
helio	the reference latitude and longitude for the alignment. This should be the position of the active region at some time, preferably near the time it crossed closest to the center of the disk. helio(0) = longitude E/W (E is negative) helio(1) = latitude N/S (S is negative) It can be a non-zero scalar value if REF_IMAGE is passed in. In this case, the heliocentric coordinates are derived from REF_IMAGE.

date_helio	the reference date/time for the heliocentric input.
suncoord	The coordinates of the center of the portion to extract in angular arcseconds relative to sun center. suncoord(0) = E–W coordinate (E is negative) suncoord(1) = N–S coordinate (S is negative)
ccdcoord	The FRE (full resolution CCD pixel equivalent) of the center of the portion to extract.
outres	The resolution of the output image (0=FR, 1=HR, 2=QR). If not set, then convert all to full res. Only relevant if registering the images.
outsize	The dimension of the output image [#col, #lin]. If it is a scalar value, it will make the #lin = #col. If not set, it will make it the largest size of the input field of views. If PFI and FFI are mixed, it will take the largest PFI.

fill_gap	A gap can exist between two PFI strips for a single observing region because of a change in S/C pointing. If this variable is set, then the gap between the PFI strips is filled by interpolating the lines above and below.
roll_do	If set, then perform the corrections for the roll offset.
normalize	If set, perform exposure normalization.
	——————— Miscellaneous Parameters ———-
force_read	If set, read all datasets at once even if input was specified with INFIL/DSET. This is available because doing LEAK_SUB on all images at once if the input is PFI is more efficient.
outfil	If specified, the data will not be passed back in INDEX_OUT and DATA_OUT. Instead, it will be saved in an SDA file with the name specified by this parameter. If "uncert" and "satpix" are included in the procedure call, they will be saved with the same file name with a "_unc" and a "_sat" appended to the file name.

SXT_PREP always stores the actual pixel address of the portion extracted and used. The .HIS.SUN_CENTER value stored in the index is always the actual pixel address where the sun center fell on the CCD for that time. When a registration is performed, the CORNER_SAV value will not be set to the same value, even though it is a registered cube. This is because the registration is RELATIVE TO SUN CENTER, and that is changing. Plotting the corner minus the sun center WILL result in a flat plot with all values identical.

5.5.2 XSXT_PREP

XSXT_PREP was under development as this volume was being written. See the User's Guide for a overview of the capabilities of XSXT_PREP.

5.5.3 SXT_DECOMP

The SXT data are normally compressed from a 12-bit word to an 8-bit word via a pseudo-square-root compression algorithm. SXT_DECOMP takes the compressed data array (*data*) and creates an integer array (*ndat*) of the decompressed numbers by typing:

IDL> **ndat = sxt_decomp(data)**

5.5.4 SXT_COMP

It is possible to re-compress the data. The routine SXT_COMP will compress integer data which is in the range between 0 and 4095 into byte data using the SXT compression algorithm.

IDL> **bdat = sxt_comp(data)**

5.5.5 RESTORE_LOW8

RESTORE_LOW8 will combine a compressed image with a low 8-bit image to restore the full 12 bits of resolution. The images need to be taken relatively close in time (less than hours apart), and it must be for the same PFI location. A sample call is

IDL> **out = restore_low8(low8, comp)**

where *low8* is the SXT image taken with the low 8 bits, and *comp* is the comparable compressed image (ideally, acquired at nearly the same time).

5.5.6 SFD_COMP

The SFD (SXT Full-frame Desaturated) images use a logarithmic compression in order to save the data as byte type. SXT_COMP compresses the data into byte type. The compression algorithm assumes that the data input is in DN/sec/HR pixel and the values range between 0 and 10^6.

IDL> **data_out = sfd_comp(data_in)**

5.5.7 SFD_DECOMP

The SFD (SXT Full-frame Desaturated) images use a logarithmic compression in order to save the data as byte type. The decompression algorithm is data_out = $10.^{(data\_in/255.*6)}$. The routine SFD_DECOMP will do this decompression for you. Beware: the output are floating type numbers and therefore, require four times as much memory.

IDL> **data_out = sfd_decomp(data_in)**

5.5.8 EXP_NORM

EXP_NORM [2] takes a data array and produces an exposure and summation mode normalized floating point array. Note the floating array takes four times more room so don't use too big an array to start with. You need to give it an index array – make sure that the two correspond exactly or else the results will be meaningless. To run it, type:

IDL> **ndat = exp_norm(data,index)**
IDL> **ndat = exp_norm(data,index,0)**

[2] If this code is used as a template, we recommend that the calling sequence be *index,data* for all routines

To do this it follows the following steps:

- decompresses the data array
- subtracts background
- divides by the exposure time
- divides by the number of pixels binned

5.5.9 SXT_COMPOSITE

SXT_COMPOSITE combines two or three SXT images with different exposures into a "composite" image. The longest exposure is used as the basis for the final exposure. SXT_COMPOSITE uses the results of SXT_SATPIX to identify the locations of saturated pixels and replaces them with equivalent pixels from the shorter exposure images, scaled for the long exposure image. This routine uses the algorithm that is used to produce the "sfd" images in the standard SXT movie. Example calls are:

IDL> **cimg = sxt_composite(index,data)**
IDL> **cimg = sxt_composite(index,data,satpix=satpix)**
IDL> **cimg = sxt_composite(index,data,dc_data,sx=[1,2,3],sd=[0,2,1])**

By default, SXT_COMPOSITE will do dark and leak frame subtraction if *data* is byte-type. The user may optionally specify the dark-frame images in *dc_data* and will default to $data - dc\_data$ unless the explicit X-ray and Dark-frame indices are given in *sx* and *sd*, respectively.

Note that the replaced pixels are box-car smoothed with a smoothing window of 5 pixels.

5.5.10 MK_SFD

MK_SFD creates a desaturated set of images from a series of long and short exposures by typing:

IDL> **mk_sfd,infil,outfil,filpref**

where $infil$ is a string array of one or more SFR filenames (and locations), $outfil$ is the string array containing the name of directory where you want the file to be created, and $filpref$ is a string array which contains the three letter file prefix name (usually SFD). The desaturated image is compressed such that:

```
  0 =         1 dn/sec/5 arcsec pixel
255 = 1,000,000 dn/sec/5 arcsec pixel
```

Use the routine SFD_DECOMP to restore the data from compressed bytes to real numbers.

5.5.11 DARK_SUB

It is possible to have the dark current removed from PFI and FFI images by using the following command:

IDL> **data_out = dark_sub(index,data)**

The routine can interpolate between two reference images to simulate the input exposure duration by using the */interpolate* option. It will call the routine GET_DC_IMAGE to get the proper image and take the subsection for the PFI cases. The SDC data files for the time period being calibrated must be in the directory $DIR_SXT_SDC.

The */interpolate* option can be turned on and run automatically every time DARK_SUB is run by setting the environment variable `ys_dark_interp`. An example of how to turn this on and off is:

% **setenv ys_dark_interp 1** # will enable automatic interplation
% **unsetenv ys_dark_interp** # will disable automatic interplation

LEAK_SUB calls DARK_SUB if the input data is BYTE type, so normally a user does not need to call DARK_SUB directly.

5.5.12 GET_DC_IMAGE

These routines get an appropriate dark frame (*dcdata*) and its index (*dcindex*) to match the images you are working with. The routines find the full-frame dark image that is closest in time and exposure level. The routine can interpolate between two reference images to simulate the input exposure duration by using the /INTERPOLATE option. It can be called with the index for a single image by typing:

IDL> **get_dc_image, index(3), dcindex, dcdata**

where *index(3)* is the index of the image you are working with, or you can pass a list array of indicies by typing:

IDL> **get_dc_image, index, dcindex, dcdata, imap**

For this case, *imap* is the same length as *index* and tells you which image to use in *dcdata* for the corresponding index. For example, if *index* is 10 elements long, and *imap(8)=3*, then you should use *dcdata(*,*,3)* for the image that goes with *index(8)*.

It is possible to extract the dark frames manually using a command similar to the following example:

IDL> **get_dc_image, xxx,dc_index, dc_data, times='1-sep-92 1:00', res=0, dpe=15**

The SDC data files for the time period being calibrated must be in the directory $DIR_SXT_SDC (and the SDW data files must be in $DIR_SXT_SDW when calibrating images which were taken when the TEC is off).

5.5.13 LEAK_SUB

On 13-Nov-92, there was an entrance filter failure which caused visible light to contaminate the X-ray images. It is possible to have the leak image removed from PFI and FFI images by using the following command:

IDL> **data_out = leak_sub(index,data)**

The routine will call DARK_SUB if it comes in as byte type. The technique at the time of writing this guide was to find the sunset leak image which matched the following criteria

- The same X-ray analysis filter.
- Using the ND filter if the input X-ray image used the ND filter.
- The nominal S/C pointing must be the same (there is one set of images between 13-Nov-92 and 19-Aug-93, and another set for images taken after 19-Aug-93)

The small changes in S/C pointing were not being accounted for as of Jan-94, even though there is a relatively strong dependence on pointing drift. Future modifications to LEAK_SUB might take this factor into account.

The leak image files must be in the directory $DIR_SXT_SFC.

5.5.14 INTERP_IMG

To linearly interpolate two images to a specified time type:

IDL> **interp_img, index1, img1, index2, data2, time, index_out, img_out**
IDL> **interp_img, idx(10),img(*,*,10),idx(12),img(*,*,12), time, nidx, nimg**

where the two input images to be interpolated are specified by the paired variables *index* and *img*. The input interpolation variable *time* can be in any Yohkoh standard time format. The returned parameters *index_out* and *img_out* refer to the interpolated *index* information and the interpolated image *data* respectively.

5.5.15 SXT_FSP and SXTBOX_FSP [*]

SXT_FSP is the same as HXS_FSP, but for SXT spectra, you now have the option to fit temperatures and emission measures using more than two filters.

IDL> **SXT_FSP, index, data, fit_pars**

SXTBOX_FSP fits spectra to a given box, chosen by LCUR_IMAGE, for a set of SXT images.

```
IDL> SXTBOX_FSP, index, data, fit_pars [, sc_par, ch_dta, sdel=sdel, $
IDL>             outfile=outfile, pfile=pfile, same_bx=same_bx, boxq=boxq]
```

5.5.16 SXTHXT_FSP and SXTHXTBOX_FSP [*]

See the description under in the HXT chapter.

5.5.17 SXT_TEEM

SXT_TEEM enables you to derive a temperature map (Te) from two filters taken of the same region. Some sample calls are:

```
IDL> sxt_teem,index1,data1,index2,data2,Te
IDL> sxt_teem,index1,data1,index2,data2,Te,EM,dTe,dEM,subs=subs
IDL> sxt_teem,index1,data1,index2,data2,Te,EM,dTe,dEM,$
IDL>             sat1=sat1,sat2=sat2,unc1=unc1,unc2=unc2
IDL> sxt_teem,index1,data1,index2,data2,Te,EM,dTe,dEM,$
IDL>             sat1=sat1,sat2=sat2,unc1=unc1,unc2=unc2,/average
```

index1,data1 should contain data for a single SXT filter. *index2,data2* should contain data for a single SXT filter which is different from that contained in *index1* and *data1*.

The input data should be decompressed and background subtracted. If index is a vector, then the data arrays should be 3-d arrays (cubes).

Normally, SXT_TEEM will get the proper exposure times from the index. However, if you have renormalized the data in a non-standard manner, then use the *t1=t1* and *t2=t2* keyword to over-ride the value in index. t1 = Integration time in msec. If Image1 is a cube, then t1 must be a vector and whose length = n_elements(Image1(0,0,*))

The keyword option *sum = sum* calls SXT_SUMXY to rebin the data. The input image must be a 2-d or 3-d array *sum* must be an integral fraction of the number of X and Y elements. If either of the two preceding conditions is not met, summing is not done.

If *data1* is a cube, *Te* will be computed for each image by default. If *subs=subs* is present, *subs* should contain the indices of image pairs to analyze.

The keyword option *thresh1=thresh1, thresh2=thresh2* sets the level above which data are considered for processing. The level is set on the background subtracted, rebinned data values. If *sum=2*, then you must increase the value of *thresh1* by a factor of 4 to have the same effect.

5.5.18 SXT_ERG_PER_DN

SXT_ERG_PER_DN returns the log of the radiative energy per DN assuming a coronal spectral model with temperature, T_e. The units of the result are $\log_{10}$(erg/DN), where DN is assumed to be the decompressed and background subtracted value that is read out of the CCD camera. Example calling statements are:

IDL> **log10_erg_dn = sxt_erg_per_dn(log10_Te,filter)**
IDL> **log10_erg_dn = sxt_erg_per_dn(log10_Te,filter,date='1-dec-92')**

and $log10\_Te$ and $filter$ are input variables which may be vectors. $log10\_Te$ is the log of the electron temperature (K), of the solar plasma and $filter$ is the SXT filter number. For example, to compute $\log 10$(erg/DN) for all filters over the range of temperatures between $\log 10(T_e)$ between 5.5 and 8.0 use the following calling sequence:

IDL> **Te = 5.5 + findgen(26)*.1** ; Generate a temperature vector
IDL> **log10_erg_dn = sxt_erg_per_dn(Te,indgen(6)+1)**

The purpose of the *date* keyword option is to allow the post-November 1992 SXT response functions to be used. This can be invoked by specifying any date later than "14-Nov-92", the day after a hole in the entrance filter appeared, and caused the SXT response function to change.

5.5.19 SXT_VIGNETTE

The SXT vignette function was determined from pre-launch calibration data obtained at White Sands Missile Range (WSMR) and was further refined using flight data. The vignette function is approximated by two non-concentric cones. The routine SXT_VIGNETTE computes the SXT vignette function for a given SXT image as described in the scalar SXT *index* as follows:

IDL> **vignette = sxt_vignette(index)** ; SXT Index should be a scalar
IDL> **vignette = sxt_vignette(array)**
IDL> **vignette = sxt_vignette(index,energy)**

If *array* is given then $array = [x0, y0, Nx, Ny, Mode]$ and $Mode = 0$, 1, or 2, for full, half and quarter resolution. The second argument, *energy* will specify whether to use the low-energy results (derived from C-K and Al-K test data) or the high-energy results (derived from Ag-L test data). SXT_VIGNETTE is called by SXT_OFF_AXIS to correct SXT images. A quick summary is available by typing:

IDL> **help_vignette** ; Display SXT vignette function

5.5.20 SXT_OFF_AXIS

The X-ray vignetting function can be approximately removed with the routine SXT_OFF_AXIS. Example calling statements are:

IDL> **data_out = sxt_off_axis(index, data, index_out)**
IDL> **data_out = sxt_off_axis(index, data, /update_index)**
IDL> **vignette = sxt_off_axis(index, /vignette_only)**

SXT_OFF_AXIS calls SXT_VIGNETTE to calculate the SXT X-ray vignette function. See SXT Calibration Note 37 and the Instrument Guide of this manual for more information. Two Vignette functions have been determined: the high-energy function is applied to the Al12 and Be119 filters and the low-energy function is applied to all other filters. Note that this approach is somewhat approximate and care should be used when applying this correction to SXT data.

5.5.21 WL_PROCESS [*]

Alan McAllister wrote the routine WL_PROCESS to process the SXT aspect sensor images and generate limb darkening profiles, flat field images, white light images minus the dark current, flat field, and limb darkening signals. The routine is still under development and you should look at the documentation header of that routine for more details.

5.6 Routines for Image Alignment

The routines described here are specific to SXT. There are also some routines listed in the General Yohkoh Software chapter which can be used for either SXT or other ground-based images.

5.6.1 SXT_PREP

See the earlier section describing the details of SXT_PREP.

5.6.2 SXT_CENTER [*]

SXT_CENTER calculates the row (x) and column (y) of the center of the solar disk and the radius (r) of the Sun given a data cube (*data*) and the index structure (*index*). The routine performs a fit to the limbs of the image. The fit can be off by several pixels in the E/W direction for X-ray images. See the SXC database if you are interested in the X-ray center for all SFD images as calculated by SXT_CENTER.

IDL> **sxt_center,data,index,x,y,r**

There are many options that are possible to use, please check them with DOC_LIBRARY.

5.6.3 GET_GBO_PFI [*]

Given a SXT PFI image and either the file name of a ground-based image, or the ground-based image itself, GET_GBO_PFI will extract the portion of the GBO image corresponding to the SXT PFI. The routine (a modification of the routine GBO_PFI written by Alan McAllister) determines the PFI corners from *index*, rotates these to the time of the GBO image, and extracts the section of the GBO image. You can either pass in the GBO image itself, in which case you MUST also supply the time for it, or a string with the path/filename using the Yohkoh file name convention. Some sample calls are:

IDL> **gbo_img = get_gbo_pfi(index, gbo_fullimg, gbo_time)**
IDL> **gbo_img = get_gbo_pfi(index, '/yd5/gbo91_46a/gkm911116.1958')**

5.6.4 OCENTER

OCENTER will obtain the center of the Sun from a optical full frame image. It makes a "silhouette image" consisting of 1's and 0's, then calculates the centroid to determine x and y. The area is used to determine the image radius. The input image can be compressed or decompressed, and in any summation mode. The output is in SXT full resolution pixels unless */noscale* is set.

IDL> **ocenter, image, x, y, radius**
IDL> **ocenter, image, x, y, radius, /noscale, cut=0.3**

where *image* is the 2-D input image, and x and y are the E/W and N/S position of the sun. *radius* is the solar radius in pixels and *cut* is the fraction of image maximum to define the limb (default is 0.2)

5.7 Routines for Accessing Secondary Database Files

5.7.1 GTAB_COMM, GTAB_ENTRY, GTAB_ROI

It is possible to get information on what was used in an SXT table by using one of these routines. GTAB_COMM returns information on the common table, GTAB_ENTRY returns information on the entry tables, and GTAB_ROI returns information on the ROI location and image shape tables. Some sample calls:

IDL> **print, gtab_comm(index)**

IDL> **print, gtab_comm('15-nov-91')**

5.7.2 GTAB_PFI and GTAB_FFI

These routines allow a user to print information on the partial-frame and full-frame observing sequence being used. There are four possible sequences for PFI and four more for FFI. Which sequence is used is determined by the DP mode and telemetry rate. If the SXT index for an image is passed, then it will figure out which of the four sequences was running at that time.

IDL> **print, gtab_ffi(index)**
IDL> **print, gtab_pfi('15-nov-91', 2)**

If a time is passed, it will default to show sequence number 0. It is possible to specify which sequence table by passing it as a second parameter.

5.8 Routines for Engineering Plots

5.8.1 PLOT_SOT

PLOT_SOT will plot out the aspect sensor degradation as a function of time by typing:

IDL> **.run plot_sot**

It will produce plots for both the narrow-band and wide-band filters.

5.8.2 PLOT_TEMPS2 [*]

PLOT_TEMPS2 plots the SXT instrument temperatures. This program can only be run after the variable *index* exists which came from an SXT file. A common practice is to run PLOT_SOT which gets images over the whole mission.

IDL> **.run plot_temps2**

A mosaic of time histories of the various instrument temperatures will be plotted.

5.8.3 PLOT_SSL

PLOT_SSL will plot the SXT summary log (SSL) information. A series of plots are made which include:

- Plots of measured exposure levels
- Plots of SXT temperatures
- Plots of number and duration of exposures through different filters
- Plots of percentage of data received

- Plots of average/max SXS1,SXS2,HXT-Low,RBM counting rates

Sample calls are:

IDL> **plot_ssl**
IDL> **plot_ssl, [0,1]**
IDL> **plot_ssl, 3**

where the second example only plots the first two series of plots (exposure levels and SXT temperatures), and the last example only plots percentage of data received.

6 Wide Band Spectrometer (WBS)

6.1 Routines for Making Time Plots

6.1.1 PLOTT_WDA

PLOTT_WDA will make a light curve plot for all channels using either the index or roadmap. The second 'T' signifies that it is a time plotting routine. Seven plots are made, one for each channel. Some sample calls are:

IDL> **plott_wda, index**
IDL> **plott_wda, roadmap**
IDL> **plott_wda, roadmap(100:200), psym=10**

6.2 Routines for Making Spectral Plots

6.2.1 PLOTS_WDA

PLOTS_WDA is a very rudimentary plotting routine that will display the spectra for any of the WBS sub-instruments. It is interactive and will prompt the user for which sub-instrument to display, and which datasets. A sample call is:

IDL> **plots_wda, index, data**

6.3 Routines for 2-D Spectral Display

6.3.1 DISP_WDA

The evolution of all of the WBS spectra against time can be displayed as a pseudo-image using the routine DISP_WDA. Note: The time axis of DISP_WDA is not uniform.

IDL> **disp_wda, index, data**

6.3.2 SXSPC

SXSPC makes the time profiles of SXS-PC11, PC12, PC-21, and PC22 with various options[3].

IDL> **pc_data= sxscv(data, index)**
IDL> **sxspc,pc_data,index**

[3] If this code is used as a template, we recommend that the calling sequence be *index,data* for all routines

6.3.3 WBSPC

WBSPC makes four counting rate time profiles of SXS-PC11(21), SXS-PC12(22), HXS-PC1 and HXS-PC2.

IDL> **wbspc,data,index**

6.3.4 PLOT_HXSPC

PLOT_HXSPC makes a counting rate time profile of HXS-PC1, HXS-PC2 or HXS-PC1 + PC2 with UTPLOT.

IDL> **plot_hxspc,index,data,pc**

6.3.5 PLOT_HXSPH

PLOT_HXSPH makes a counting rate time profile of HXS-PH with UTPLOT. You choose the energy or channel range.

IDL> **plot_hxsph,index,data,range=range**
IDL> **plot_hxsph,index,data,channel=channel**

6.3.6 PLOT_GRSPCL

PLOT_GRSPCL makes a counting rate time profile of GRS-PC11, PC12, PC21 or PC22 with UTPLOT.

IDL> **plot_grspcl,index,data,pc**

6.3.7 PLOT_GRSPCH

PLOT_GRSPCH makes a counting rate time profile of GRS-PC13, PC14, PC15, PC16, PC23, PC24, PC25 or PC26 with UTPLOT.

IDL> **plot_grspch,index,data,pc**

6.3.8 PLOT_GRSPHL

PLOT_GRSPHL makes a counting rate time profile of GRS-PHL1 or GRS-PHL2 with UTPLOT. You choose the energy or channel range.

IDL> **plot_grsphl,index,data,ph=1,range=range,channel=channel**

6.3.9 PLOT_GRSPHH

PLOT_GRSPHH makes a counting rate time profile of GRS-PHH1 or GRS-PHH2 with UTPLOT. You choose the energy or channel range.

IDL> **plot_grsphh,index,data,ph=1,range=range,channel=channel**

6.3.10 Light Curve of RBM-PC

See the description in the User's Guide.

6.4 Routines for Data and Information Extraction

6.4.1 GT_ Routines

The routines GT_SXS1, GT_SXS2, GT_HXS, GT_GRS1, GT_GRS2, GT_RBMSD, and GT_RBMSC routines will extract the average counts/sec for the selected channel.

SXS1	Only SXS_PC21	3-15 keV
SXS2	Only SXS_PC22	15-40 keV
HXS	HXS_PC1 plus HXS_PC2	20-657 keV
GRS1	GRS_PC11 plus GRS_PC21	0.3-1.2 MeV
GRS2	GRS_PC12 plus GRS_PC22	1.2-5.6 MeV
RBMSD	PC1 plus PC2	5-300 keV
RBMSC	Only RMSSC	>20 keV

The input can be roadmap, index, or observing log and it will get the proper structure tag and decompress it properly to return counts/sec/sensor. It is possible to get a string defining the channel selected by using the *title* keyword option.

IDL> **y = gt_sxs1(roadmap)**
IDL> **y = gt_hxs(w_h, title=title)**

6.4.2 GET_INFO

See the description in the "General Yohkoh Software" chapter on page 17.

6.5 Routines for Calibration and Analysis

6.5.1 HXS_FSP [*]

With HXS_FSP you interactively choose data intervals, and fit the HXS spectrum, using the routine fsp_11.pro. *index* and *data* are the WBS index and data structures. *fit_pars* is the structure containing the results. Optional parameters are – *sc_par* is a structure containing spectrometer channel information, *ch_dta* is a structure containing the data for each channel, *outfile* and *pfile* are filenames for printouts and plots, respectively; *countfile* is a filename, if this option is set, the count rates are printed into that file which may be fit by the routine FSP_PROC described above.

IDL> **HXS_FSP, index, data, fit_pars**
IDL> **HXS_FSP, index, data, fit_pars, sc_par, ch_dta**
IDL> **HXS_FSP, index, data, outfile=outfile**
IDL> **HXS_FSP, index, data, pfile=pfile**
IDL> **HXS_FSP, index, data, countfile=countfile**
IDL> **HXS_FSP, index, data, fit_pars, sc_par, ch_dta, sdel=sdel**

6.5.2 HXS_SPL

HXS_SPL makes a HXS spectrum fitted by single power law. You interactively choose the background and flare time intervals for the spectral analysis.

IDL> **hxs_spl,data,index**

6.5.3 HXS_SP2

HXS_SP2 makes HXS spectrum fitted by double power law in a similar way to HXS_SP1.

IDL> **hxs_sp2,data,index**

6.5.4 GRS_SPEFF

GRS_SPEFF makes preliminary GRS-1 and 2 spectra (0.3 – 100 MeV).You interactively choose the flare and background time intervals for the spectral analysis. The flux of each channel is calculated by dividing the counts by full energy absorption efficiency. The first three channels data of GRS-1 and the four channels of data of GRS-2 are not plotted because these channel widths are not exactly determined.

IDL> **grs_speff,data,index,ph=1,/disp**
IDL> **grs_speff,data,index,ph=2,/disp**

IDL> **grs_speff,data,index,ph=1,fltime=fltime,bgtime=bgtime**

6.5.5 Further Analysis of HXS and GRS Data

There are several routines for further HXS and GRS data analysis. With these routines you can get the following:

- Time profile of ratio of HXS-PC2 to HXS-PC1 counting rates
- Time profile of ratio of HXS-PHi-j to HXS-PHm-n counting rates
- Time profile of ratio of GRS-PCi to GRS-PCj counting rates
- Time profile of ratio of GRS-PHi-j to GRS-PHm-n counting rates
- HXS spectral fitting by single power law, double power law, thermal and power law plus Gaussian etc.
- Residuals and chi-square map for HXS spectral analysis
- Gamma-ray spectral analysis (continuum plus lines) with the maximum number of fitting parameters is 62
- Residuals and chi-square map for GRS spectral analysis

See the detailed description in the User's Guide.

6.6 Instrument Response Files

The instrument response files for SXS, HXS and GRS are listed here. These files are ASCII files and are located in $DIR_WBS_CAL.

6.6.1 SXS

dtcorrf.dat	Table of the deadtime correction factor vs. counting rates of SXS-PC11, PC12, PC21 and PC22

6.6.2 HXS

hxs_01.rel	Table of channel number (0 – 31) vs. energy (keV) relation for HXS- PH for HXS-OS of "01 (hex)". (1992 June 9 – present).
hxs_21.rel	Table of channel number vs. energy relation for HXS-PH for HXS-OS of "21 (hex)". (1991 Oct. 1 – 1992 June 9).
hxs_01_conv260.rel	Table of incident photon energies (keV) corresponding to the 260 rows of the 260 x 32 response matrix for HXS-PH spectral analysis for HXS-OS of "01 (hex)". (1992 June 9 – present).
hxs_21_conv260.rel	Table of incident photon energies (keV) corresponding to the 260 rows of the 260 x 32 response matrix for HXS-PH spectral analysis for HXS-OS of "21 (hex)". (1991 Oct. 1 – 1992 June 9).
hxs_01_conv260.resp	260 (row) x 32 (column) response matrix for HXS-PH spectral analysis for HXS-OS of "01 (hex)". The row and column represent the incident photon energy and observed channel number, respectively. The matrix elements of i-th row represent a response function for incident photons of i-th energy in hxs_conv260.rel.
hxs_21_conv260.resp	260 (row) x 32 (column) response matrix for HXS-PH spectral analysis for HXS-OS of "21 (hex)". This is similar to hxs_01_conv260.resp.
dtcf_pc_hxs.pro	Deadtime correction factor for HXS-PC1 and PC2. The correction is not exact when the deadtime-corrected count of total PC exceeds about 100,000 counts/sec. It is due to instrument limitation.
dtcf_ph_hxs.pro	Deadtime correction factor for HXS-PH. The correction is not exact when the deadtime-corrected count of total PH exceeds about 35,000 counts/sec. In addition, spectral distorsion occurs when the deadtime-corrected count of total PH exceeds about 21,000 counts/sec. It is due to instrument limitation.

6.6.3 GRS

grs1_40.rel	Table of channel number (0 – 127) vs. energy (keV) relation for GRS- 1 for GRS1-OS of “40 (hex)”. (1991 Oct.1-present). Energy widths of the first 3 channels are not exact.
grs2_40.rel	Table of channel number (0 – 127) vs. energy (keV) relation for GRS-2 for GRS2-OS of “40 (hex)”. (1991 Oct.1-present). Energy widths of first 4 channels are not exact.
grs1_40_conv260.rel	Table of incident photon energies (keV) corresponding to the 260 rows of the 260 (row) x 128 (column) response matrix for GRS- PHL1 spectral analysis.
grs1_40_conv260.resp	260 (row) x 128 (column) response matrix for GRS-PHL1 spectral analysis. The row and column represent the incident photon energies and observed channel number, respectively. The matrix elements of i-th row represent a response function for incident photon of i-th energy in grs1_conv260.rel.
grs2_40_conv260.rel	Table of incident photon energies (keV) corresponding to the 260 rows of the 260 (row) x 128 (column) response matrix for GRS- PHL2 spectral analysis.
grs2_40_conv260.resp	260 (row) x 128 (column) response matrix for GRS-PHL2 spectral analysis. This is similar to grs1_conv260.resp.
dtcf_ph_grs11.pro	Deadtime correction factor for GRS-PHL1.
dtcf_ph_grs21.pro	Deadtime correction factor for GRS-PHL2.

A Yohkoh Distribution and Archive Tapes

A.1 The Organization of the Tapes

A.2 Overview of Yohkoh Data Archive

The YOHKOH data archive is created on Unix computers at ISAS, Japan, using standard archiving tools from the Unix operating system. All archiving is done on 8mm Exabyte tapes. Nominally, each archive tape will contain one calendar week (always beginning with Sunday) of reformatted data and all the software used to access the data. Tapes are created as soon as a complete week of the data are assembled; but, because data are collected from two sources (KSC in Japan and NASA's DSN) some delay is expected. The tapes are then sent to California for duplication and distribution to the U.S. Co-I's. At ISAS the archive tapes are copied to MO disk.

Archive tapes can be read on Unix and VMS machines such as SUN, SGI, and Digital using software package written in Interactive Data Language (IDL).

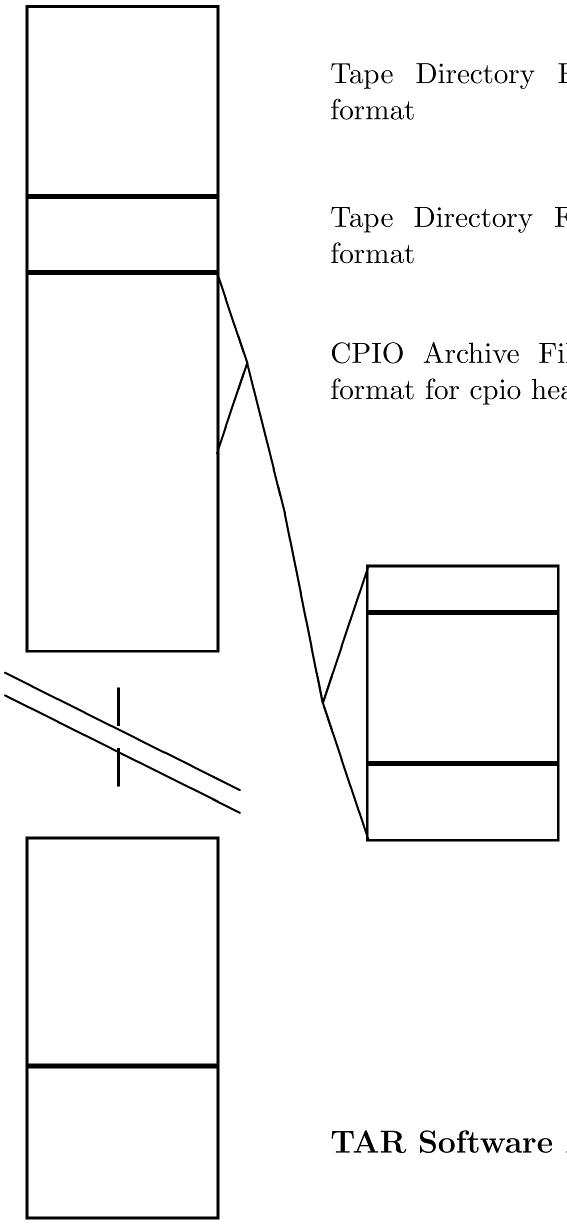
For more information on reading archive tapes see the GO_RDTAP description in the User's Guide.

A.3 Structure of Archive Tape

YOHKOH archive tapes contain a tape directory, reformatted data and software. Tapes are created on DIGITAL's Unix (Ultrix) system using 'cp', 'cpio' and 'tar' archiving utilities. Reformatted data files and the weekly files are all copied to tape using the 'cpio' utility. The software archive is copied to tape with the 'tar' utility. Since, the software archive comprises several different directory trees the 'tar' utility was the logical choice because it allows one preserve these directory structures during restore operations. Archive tapes will nominally contain a single calendar week (always beginning with Sunday) of reformatted data. (Where each reformatted file contains data from a single orbit which spans approximately 90 minutes.) This calendar week is given a number (week number: 1–53) which is then encoded into the name of the tape in the format 'yy_wka.xx', where 'yy' is the year, 'wk' is the week number, 'a' is the media number (always "a" for tapes, but can be "b" for other media, such as MO disks), 'xx' is the revision number or level of the archive tape.

The detailed contents of a YOHKOH data archive tape begins with two directory files in different formats (ASCII and binary), followed by a series of 'cpio' data archive files, and lastly a 'tar' software archive file.

YOHKOH ARCHIVE TAPE



Tape Directory File ASCII format

Tape Directory File Binary format

CPIO Archive Files (ASCII format for cpio headers)

Archive file 1 with weekly files

Archive file 2 with reformatted files: ada, cba, bda, hda, sfr, spr, and wda.

Archive file 3 with reformatted files: ada, cba, bda, hda, and wda.

TAR Software Archive Files

Structure of the data and software archive tape. Each cpio archive file contains only those reformatted data that share the same file ID (e.g. archive file 2 contains files from the file ID:

911201.1545, file 3 would contain files with file ID 911201.1715).

The tape directory file lists the contents of each tape. The file contains general information such as when the tape was made, the name of the tape, the number of 'cpio' archive files, the first and last data file identification (fileID) contained within the 'cpio' archive, and a breakdown on the contents of the software 'tar' archive file giving names and sizes of embedded 'tar' files. The tape directory also has two detailed tables. The first of which tabulates the weekly files present within the first 'cpio' archive and the second tabulates the 'orbit' fileID and the sizes of each reformatted file (e.g. ada, bda, cba, hda, sfr, spr, wda) for each 'cpio' archive file. This last table is very useful for identifying whether a given file is on a tape.

The reformatted data are placed on tape with the help of the Unix archiving utility 'cpio' which allows a specified list of files to inserted into a single archive file. Using this utility we have bundled the data by placing files with the same data file identification or fileID (such as 911201.1654, which corresponds to the date 1-Dec-91 at 16 hours and 54 minutes) together into a single archive file and copied to tape. Each fileID is unique and represents one orbit of data. Since there are about 16 orbits per day, each tape could have 112 + 1 different 'cpio' archive files. The additional archive file is for the first 'cpio' file which only contains the weekly files such as the observing log file (obs*) and the pointing file (pnt*).

Even though the software is placed on the tape, it is best to obtain the software across the network. See Appendix **??** for more details.

A.4 Access Software

There are three IDL utility programs which access the archive tapes. An IDL procedure called MKTAP to create archive tapes, another procedure called RDTAP to read any of the various sections or parts of sections of the tape (see Figure), and finally a robust procedure called YOHKOH_SW to read and restore just the software archive files. The YOHKOH_SW procedure is very useful and is recommended for creating a software environment which would be quite similar to the ISAS computing environment. Documentation for RDTAP and MKTAP are given below and the 'yohkoh_sw' documentation can be found in separate guide which is attached to this users guide.

A.4.1 Reading Tapes (RDTAP)

When using RDTAP to read and extract data from the archive tapes be aware that the newly created disk files are always placed in the users default directory. RDTAP has an on-line brief help information listing to aid the unfamiliar user. To activate this feature type RDTAP without any parameters at the IDL prompt. To get a listing of the contents of the tape one can either use the RDTAP utility with the option to extract the tape directory files

IDL> **rdtap, device, /dir**

which creates the two files called 'xadyy_wkd.xx' (ASCII) and 'xbdyy_wkd.xx' (binary). An alternative method is to simply copy the first file (the ASCII file) from the tape to disk via

DCL or Unix commands (e.g. for VMS mount/for mt: and copy mt: tapedir.lis or for Unix cp dev tapedir.lis). When extracting the binary data files from tape to disk the files are restored without any byte swapping. This keeps the files in their original format so that any byte swapping, if necessary, is handled automatically and transparently to the user at the time the disk files are read by the analysis software. Note that the tape is rewound at the beginning of each RDTAP request, so it is more efficient to specify all requests within a single RDTAP command than it is to make several smaller requests. Also the requested list of the files need not be in order of appearance on the tape as the routine will sort all input requests for the most efficient extraction.

The general calling sequence for RDTAP is:

IDL> **rdtap, dev** [**, file, prefix= prefix, fileID= fileID, week=week, $**
IDL> **dir=dir, range= range, software= software**]

Where the optional parameters are enclosed within brackets (note that when calling RDTAP with the *dev* parameter specified at least one other optional parameter must also be included. In general, all non-switch parameter inputs have a string data-type (e.g. dev = '/dev/nrmt0h'). The definitions of the parameters is as follows:

```
dev     is the device name (Ultrix example: '/dev/nrmt0h').
file    is a list of requested file names (e.g. spr910925.1020).
prefix  is a list of requested file prefixes.
fileID  is a list of requested fileIDs (e.g. 910925.1020).
week    is a list of weekly files to restore.
dir     is a switch to restore the tape directory file.
range   is a switch to create a list of files from input.
soft*   is a switch to restore the software tar file to disk.
```

Some examples are:

IDL> **rdtap, dev, ['spr911029.0300','bda911030.0440']**
IDL> **rdtap, dev, prefix=['spr','sfr'] or rdtap, dev, prefix=['s']**
IDL> **rdtap, dev, fileID=['911029.0900']**
IDL> **rdtap, dev, prefix='spr',fileID='910923.1234', week='all'**
IDL> **rdtap, dev, prefix='s', fileid=['911101.0200','9111015.0900'],/range**
IDL> **rdtap, def_tapd(1), /software, /dir**

The first example restores just the requested spr and bda files. The second example restores ALL SXT files from the tape (could take about two hours). The third example restores all archived files of this fileID. The fourth example restores just the spr910923.1234 file and all weekly files. Note, that week=['obs', 'pnt', 'fem', 'evn'] will restore any of these files found on the tape (i.e. if FEM and EVN files were missing, only OBS and PNT files would be restored. The fifth example restores sfr and spr files between orbits dated 1-Nov at 02:00 UT and 15-Nov at 09:00 UT. The last example decompresses the software archive file and restores the tape directory files to the default directory. Using the DEF_TAPD function to specify the device name by selecting the device number is a simple way to get the correct device name.

In using the tape directory file to review tape contents and select data files to be extracted from the tape, find the final table within the file which tabulates the fileID and file prefix. The heading of table reads FileID and cpio, followed by all of the valid file prefixes. To find a given file with the table, match the fileID with the corresponding row within the FileID column and then match the file prefix with the corresponding prefix column. A non-zero entry for the size of the file (bytes) indicates that the file is on the tape.

A.4.2 Tape Reading on VMS

For VMS systems, there are two DCL commands which must be done prior to entering IDL in order to access the tape drive. Issue the following two commands before starting IDL:

```
$ Allocate tapeDrive mt*        where tapeDrive refers to the 'mu'
                                 device (example: 'mub0') and mt*
                                 refers to mt0 thru mt9, pick one.

$ Mount/FOREIGN mt*     this mounts the tape to the 'logical device'
                        name mt* (again mt0 thru mt9).
```

Then enter IDL. Assuming one selected 'mt0' as the logical device name for the tape drive in the allocation command, the valid rdtap command restoring the tape directory files would look like:

IDL> **rdtap, def_tapd(0), /dir**

A.4.3 Tape Read Errors

Any tape read errors encountered while reading the tape are flagged to the current session and an attempt is made to continue reading the tape. All file(s) which were not restored as a result of the tape read errors are logged to a tape error log file called 'rdtape.log'.

In general, recovering unread file(s) may be difficult. If an error resulted in the loss of two or more consecutive files one can first attempt to restart the file extraction beginning with the second file in the 'rdtape.log'. If this procedure is successful, it is most likely that the first file will be unreadable, however one should still attempt to re-read the file as the error may have been due to dust or other contaminates on the tape. If the error log contains several of these situations of two or more consecutive file losses repeat the above procedure for each case.

A.4.4 Writing Tapes

Tapes are created with the IDL procedure MKTAP. A blank 8mm tape should be in the desired tape drive before one starts the procedure. The nominal mode starting this procedure is to specify the device as a parameter:

IDL> **mktap, dev**

where, dev can be either a string-type variable such as '/dev/nrmt0h' or an integer 0, 1, 2, etc., which specifies the device number. The procedure can also be started with out a device specified. However, in this case the program assume the tape is in one of the follow defaults devices (system dependent): '/dev/nrmt0h' for Ultrix systems, '/dev/rmt/tps0d6nrnsv' for IRIX systems, and '/dev/nrst0' for SUN systems. The first question one must answer is the name of the data directory (note this is NOT the path). For example, if the path to the archive data is '/yd4/92_12a' then the data directory is '92_12a'. The second and typically last question asks whether to archive all files or part of the files within the data directory. After answering this final question a script file called 'mktape.com' and two tape directory files xbdyy_wka.xx (binary version) and xadyy_wka.xx (ascii version) are created in the data directory. It is important to note that the file ID of the tape directory files (xad and xbd) is the same as the name of the tape which will hold these files. This completes the IDL procedure MKTAP. To make the tape one types the following command:

% **source mktap.com**

MKTAP also has a number of switches which can be specified with the MKTAP command to allow the program to run in different modes. Listed below are a few examples, see the program header to MKTAP for a complete list:

IDL> **mktap, 1, datadir='91_50a,91_51a'**

will use tape drive '/dev/nrmt1h' on Ultrix systems, and archive weekly files from sub-directory '91_50a' and all requested reformatted data from both sub-directories.

IDL> **mktap, 1, datadir='91_50a,91_51a', /alldata**

same as above command, except that there will be no prompt for which reformatted files to place on the tape as all reformatted data from both directories will be archived.

As final comment there are a number of constraints one should be aware of when creating archive tapes. First, one must have WRITE privilege to the data directory. This is necessary since MKTAP creates a script file and two versions of the tape directory files in the data directory. Secondly, an observing log with the name which corresponds to the dates of the data archived must be on the directory or the program will prompt the user for its name.

B Institutes and Nodes with the Yohkoh Package

B.1 List of Institutes

The following are a list of the Yohkoh principle investigator and co-investigator sites which have the Yohkoh software and database package. The node names and a contact person are listed.

Institute	Computer Nodes	Contact Person
Institute of Space and Astronautical Sciences (ISAS)	isass0 thru isass4 flare1 thru flare12 isasxa thru isasxc isass6	Freeland Shimizu Bentley Hudson
Lockheed Palo Alto Research Laboratory (LPARL)	sxt0 thru sxt6, sag	Freeland
Montana State University	sxt4, msusxt6	Acton
University of Hawaii	akala, akoko, maile hau, iliahi, wiliwili	Canfield
University of California, Berkeley	sunspot	McTiernan
Stanford University	flare, panache	Klimchuk
Solar Physics Research Corporation (SPRC)	noao	Harvey
National Astronautical Observatory of Japan (NAOJ)	spot	Tsuneta
University of Tokyo	neptune	Khan
Mullard Space Science Laborartory (MSSL)	MSSL Cluster	Bentley
Rutherford Appleton Laboratory (RAL)	solg1,RLSAC?	Phillips
Naval Research Laboratory (NRL)	aspen,dogwood,sola solb	Mariska

This next table shows the various other institutions which are not primary or co-investigator institutions and which have the Yohkoh package.

University of Alabama at Huntsville	nessie	Mariska (NRL)
University of Birmingham	??	Simnett
Solar Data Analysis Center (GSFC)	umbra,sdac	Gurman,Zarro
California Institute of Technology	sundog	Zirin,Lim
NSSDC (GSFC)	??	Batchelor
Marshall Space Flight Center	golem, golem2	Porter
University of Glasgow	astro?	Norman?
Harvard-Smithsonian Center for Astrophysics	darmok?	Kline?
National Center for Atmospheric Research (NCAR/HAO)	riker	Stanger
Harvard University	daystrom	Bookbinder
Stanford University	quake, kilohoku	Milford,Gethyn
Space Environment Laboratory, NOAA	seldmc,sunspot	McAllister,Bornmann
Department of Astronomy, Kyoto University	??	Kawai
Geophysical Institute, Kyoto Unversity	step4ku	Yatagai
CRL - Akioka	planet	Akioka
CRL - Marubashi?	??	??
Shibaura? Institute of Technology	ogwl	Sabah
Nagoya ?	??	Kato
Nobeyama ?	??	??
Astronomical Institute, Czech Republic	147.231.24.1	Farnik

B.2 List of Node Names and Numbers

A list of the most of the computer nodes which have the Yohkoh software is listed here. To verify that the lists are accurate, we recommend logging into sxt, isass0 or flare2 and looking at the file `/etc/hosts`.

```
          DECNET            Internet                           Machine

SAG      24.131   192.68.162.134 (sag.space.lockheed.com)   DEC-VMS
SXT      24.134   192.68.162.107 (sxt.space.lockheed.com)   SGI-Unix
SXT1     24.362   192.68.162.100 (sxt1.space.lockheed.com)  DEC-Ultrix
SXT2     24.365   192.68.162.109 (sxt2.space.lockheed.com)  DEC-Ultrix
SXT3     24.366   192.68.162.110 (sxt3.space.lockheed.com)  DEC-Ultrix
SXT4              192.31.215.58  (sxt4.oscs.montana.edu)    DEC-Ultrix
SXT6              192.68.162.158 (sxt6.space.lockheed.com)  DEC-Ultrix

spot              133.40.2.120 (spot.mtk.nao.ac.jp)         SUN
neptune           133.11.3.23  (neptune.astron.s.u-tokyo.ac.jp) SUN
```

ISASS0	40.932	133.74.8.100	(isass0.solar.isas.ac.jp)	DEC-Ultrix
ISASS1	40.933	133.74.8.101	(isass1.solar.isas.ac.jp)	DEC-Ultrix
ISASS2	40.934	133.74.8.102	(isass2.solar.isas.ac.jp)	DEC-Ultrix
ISASS3	40.935	133.74.8.103	(isass3.solar.isas.ac.jp)	DEC-Ultrix
ISASS4	40.936	133.74.8.104	(isass4.solar.isas.ac.jp)	DEC-Ultrix
ISASS6		133.74.8.106	(isass6.solar.isas.ac.jp)	SUN
ISASXA	40.927	133.74.8.110	(isasxa.solar.isas.ac.jp)	DEC-VMS
ISASXB	40.928			DEC-VMS
ISASXC	40.929			DEC-VMS
FLARE1		133.74.8.1	(flare1.solar.isas.ac.jp)	SUN
FLARE2		133.74.8.7	(flare2.solar.isas.ac.jp)	SUN
FLARE3		133.74.8.3	(flare3.solar.isas.ac.jp)	SUN
FLARE4		133.74.8.4	(flare4.solar.isas.ac.jp)	SUN
FLARE5		133.74.12.94		SUN
FLARE6		133.74.8.6	(flare6.solar.isas.ac.jp)	SUN
FLARE7		133.74.8.87	(flare7.solar.isas.ac.jp)	MIPS
FLARE8		133.74.8.88	(flare8.solar.isas.ac.jp)	MIPS
FLARE9		133.74.8.89	(flare9.solar.isas.ac.jp)	MIPS
FLARE10		133.74.8.90	(flare10.solar.isas.ac.jp)	MIPS
FLARE11		133.74.8.91	(flare11.solar.isas.ac.jp)	MIPS
FLARE12		133.74.8.92	(flare12.solar.isas.ac.jp)	MIPS
FLARE13		133.74.8.93	(flare13.solar.isas.ac.jp)	MIPS
FLARE14		133.74.8.94	(flare14.solar.isas.ac.jp)	MIPS
FLARE15		133.74.8.95	(flare15.solar.isas.ac.jp)	MIPS
MSSL	19.253			DEC-VMS
MSSLV1		128.40.70.62	(msslv1.mssl.ucl.ac.uk)	VMS-4000
MSSLV2		128.40.70.63	(msslv2.mssl.ucl.ac.uk)	VMS-4000
MSSLVB		128.40.70.70	(msslvb.mssl.ucl.ac.uk)	VMS-3100
MSSLA1		128.40.70.58	(mssla1.mssl.ucl.ac.uk)	VMS-3000
ssd0		128.60.8.1	(ssd0.nrl.navy.mil)	VAX
aspen		128.60.8.72	(aspen.nrl.navy.mil)	SUN
dogwood		128.60.108.102	(dogwood.nrl.navy.mil)	SUN
sola		128.60.8.41	(sola.nrl.navy.mil)	SUN
solb		128.60.8.42	(solb.nrl.navy.mil)	SUN
nessie		146.229.9.11	(nessie.optics.uah.edu)	SUN
koa		128.171.167.1	(koa.ifa.hawaii.edu)	SUN?
mamane		128.171.2.2	(mamane.ifa.hawaii.edu)	SUN

```
sunspot                      (sunspot.ssl.berkeley.edu)  SUN?
star              36.10.0.5      (star.stanford.edu)         ???
flare             36.14.0.166    (flare.stanford.edu)        DEC-Ultrix
panache           36.14.0.51     (panache.stanford.edu)      DEC-Ultrix
ogwl                             (ogwl.se.shibaura-it.ac.jp) ???
setp4ku                          (step4ku.kugi.kyoto-u.ac.jp) SUN

riker                            (riker.hao.ucar.edu)        SGI
darmok                           (darmok.harvard.edu)        ???
astro                            (astro.glasgow.ac.uk)       ???
sundog   14.905   131.215.139.154 (sundog.caltech.edu)       VMS
umbra             128.183.57.24 (UMBRA.gsfc.nasa.gov)        DEC-Ultrix

NOTE: To go from DECNET xx.yyy convention to the single node number
notation, take the xx portion, multiply by 1024, and add ‘yyy’.
So SAG becomes 24*1024 + 131 = 24707::
```

C Accessing the University of Hawaii SPAM Database

Mees Solar Observatory (MSO) is located on Haleakala, Maui, at 165E longitude and 20N latitude. MSO instrument operation hours vary from 1800-0300 UT in December to 1630-0400 UT in June. A searchable catalog called SPAM (Spectroscopy and Polarimetry at Mees) contains MSO data logs (not data). Some MSO data are available on line. Other MSO data can be obtained through the Data Use Coordinator, canfield@mamane.ifa.hawaii.edu.

SPAM logs (and searchable attributes, in addition to date, day of year, and time) include: MSO Log (instrument, NOAA AR, data type, observing setup), SEL Event List (NOAA AR, X-ray Class), SEL Region Report (NOAA AR), Yohkoh Orbit Summary, Yohkoh SXT Quiet Mode PFI Observations (latitude, longitude, X-ray and optical image size), Yohkoh Flare Observations (latitude, longitude, specific channel counts or ratios). SPAM runs on a Sun Unix workstation at MSO, and is available over Internet. Simply access (e.g., telnet) koa.ifa.hawaii.edu (128.171.167.1) from any vt100, Sun, or xterm emulator. Log on as spam (lower case); there is no password. New users are asked to read release notes and hints.

Vector magnetograms from the MSO Stokes Polarimeter are available on line for about a month after acquisition. These data are available by anonymous ftp over Internet. Use ftp koa.ifa.hawaii.edu, log in as anonymous, provide your name as the password, and then respond to the ftp prompt with cd pub/stokes. Use get to obtain a copy of the README file for further information and conditions for data use.

D Accessing the Optical Solar Data of NAOJ

1. Introduction

This document describes how to make access to various optical data of the sun obtained at the National Astronomical Observatory of Japan (NAOJ). >From workstations which support the X-window and are connected to NAOJ via network, you may log-in to our workstation

'spot' (spot.mtk.nao.ac.jp, IP-address=133.40.2.120),

under the username 'guest'. The password can be obtained by sending an e-mail request to either one of the following.

sakurai@spot.mtk.nao.ac.jp (T.Sakurai)
ichimoto@spot.mtk.nao.ac.jp (K.Ichimoto)

For remote users it would be more convenient to transfer the data and the software to your own workstation and run the software locally. The IDL package should be installed on your workstation. Files can be transferred by anonymous ftp on 'spot'. After you are connected to 'spot' via ftp, enter 'anonymous' as the username. For password prompt, you are requested to enter your name, which will be recorded.

These data can be utilized freely in your data analysis. If you write a paper or give a talk by using these data, you are requested to consult with us in terms of co-authorship.

For technical details, please contact:

Norikura heliograms/filtergrams	: K.Ichimoto, K.Kumagai
Flare Telescope magnetograms/Dopplergrams	: K.Ichimoto, T.Sakurai
Okayama magnetograms	: T.Sakurai
H-alpha full-disk images	: Y.Suematsu, N.Tanaka

These people can be accessed by e-mail at <name>@spot.mtk.nao.ac.jp.

In any case, only representative data (a few images per day) are provided on-line. Once you figure out that the data you are interested in are in archive tapes, you are encouraged to visit NAOJ, recover the data from tapes, and make use of them.

2. Observing Instruments

This section briefly describes the sources of data, namely the optical observing facilities currently being operated at NAOJ.

2.1 Mitaka Campus

(a) H-alpha flare patrol
    optics: 4cm refractor + Halle H-alpha filter
    field of view: full disk
    pixel size: 4"
    form of data: digital images of H-alpha line center
    image cadence: one minute
    recording media: digital audio tape

(b) The Solar Flare Telescope
    This is made of four telescopes T1-T4.
    field of view: 340" x 320"
    pixel size: 0.66"
(T1) vector magnetograph
    optics: 20cm refractor + birefringent filter (band width=1/8A)
    form of data: digital I,Q,U,V intensities of Fe I 6303A line
    image cadence: 3 minutes
    recording media: digital audio tape
(T2) continuum telescope
    optics: 15cm refractor + broad-band filters
    form of data: analog images taken mostly at the wavelength of g-band
    image cadence: video rate
    recording media: SVHS video tape
    (tapes are recycled unless no notable events are found)
(T3) H-alpha telescope
    optics: 15cm refractor + Zeiss H-alpha filter
    form of data: analog images of H-alpha line center
    image cadence: 10 seconds
    recording media: laser video disk
(T4) Dopplergraph
    optics: 20cm refractor + birefringent filter (band width=1/5A)
    form of data: digital I,V,Doppler shifts of Fe I 6337A line
    image cadence: 3 minutes
    recording media: digital audio tape

(c) STEP Full-Disk Magnetograph (STEP=Solar Terrestrial Energy Program)

20cm heliostat + 6cm refractor + birefringent filter
field of view: full disk
pixel size: 5"
form of data: digital I,V,Doppler shifts of Fe I 5324A line
image cadence: (the instrument is now under testing)
recording media: digital audio tape

2.2 Norikura Coronagraph Station

(a) Automated Digital Coronagraph
optics: 10cm coronagraph + interference filters
field of view: 2600" x 2400"
pixel size: 5"
form of data: digital images of the corona in the following wave bands

wavelength	band width
5303A	3A
He D3	4A
H-alpha	3A
continuum at 6630A	21A

image cadence: 3 minutes
recording media: digital audio tape

(b) Spectroheliographic Observations
optics: 25cm coronagraph + Littrow spectrograph
raster area: four swaths of 500" wide (2.5" step)
synthesis of full disk maps is made after the observation
spectral coverage: 2A (0.2A x 10 wavelength points)
form of data: digital spectroheliogram of He I 10830A
image cadence: one map per day (one raster scan takes 30 minutes)
recording media: optical disk

2.3 Okayama Observatory
optics: 65cm reflector + Littrow spectrograph + photoelectric magnetograph
raster area: 500" x 450" (10" step) or 420" x 390" (6" step)
spectral coverage: 27-80mA from the center of 5250A line
(detectors are photomultipliers; no wavelength resolution)
form of data: digital I,Q,U,V,Doppler shifts of Fe I 5250A
image cadence: a few maps per day (one raster scan takes one hour)
recording media: magnetic tape and magneto-optical disk

The data available on-line are:
part of Mitaka H-alpha full-disk images (2.1a),

part of Flare Telescope vector magnetograms (2.1b-T1),
all of Norikura He 10830 fill-disk heliograms (2.2b), and
all of Okayama vector magnetograms (2.3).
Other data are archived in tapes in digital form, or recorded on analog media.

3. Browsing the Data by 'show'

After you log-in at 'spot' under the username 'guest', follow the instruction shown on the banner message. You simply type

```
spot% show
```

and then enter your machine's IP-address after the prompt. An interactive, menu-driven session on the X-window will start.
By using the mouse you first select from

- magnetic data of flare telescope
- Ha full disk image of Mitaka
- Norikura data (He10830, corona)
- Okayama magnetic data
- exit

On all the menus, <return> means that you go back to the main menu, and the 'show' program stops by selecting <exit>.

3.1 Flare Telescope Vector Magnetograms

The data have names 'fbyymmdd.hhmm', where yymmdd and hhmm designate, respectively, the date and time in UT. By using the mouse, select the file you want to see. When the menu is too big and is split into several pages, you may select 'next page' and the next page of the file list will be shown. The other options control plotting styles and printer output.
<contour> : magnetogram in contour
<spot+contour> : magnetogram in contour, sunspots in gray scale
<red-blue> : magnetogram in red-blue, sunspots in contour
<PostScript> : make postscript output file
<printout> : print postscript file at a local printer at Mitaka
<PostScript> is only selectable after you have selected data to be displayed.
In order to transfer the postscipt file, use anonymous ftp. The contour plot is saved in 'tmp/magcont.ps' under the anonymous ftp directory.

After you exit, the data are stored in arrays: intensity map in 'img', longitudinal magnetid field in 'Bl', and transverse magnetic fields in

'Bx,By'. The header h contains observing parameters. The magnetic data are 2 byte integer in unit of 0.1 Gauss and each image has 256*240 pixels, where a pixel corresponds to about 1.35 arcseconds. The instrument tends to saturate for magnetic fields stronger than about 1700 Gauss. The field strength in large sunspots should be used with some caution.

The direction of the transverse magnetic field (i.e. the resolution of the 180 degree ambiguity) is determined by referring to the potential field calculated from the longitudinal magnetic field distribution. The direction may not be correct in regions with high magnetic shear.

In order to obtain the observation status, type
IDL> fstplot
and then choose yymmdd.

3.2 Mitaka Full-Disk H-alpha Images

The data have names 'hayymmdd.hhmm', where yymmdd and hhmm designate, respectively, the date and time in UT. By using the mouse, select the file you want to see. After you exit, the image data are stored in an array 'img', and the header h includes the observing parameters.

3.3 Norikura Spectroheliograms

In the 'Norikura' menu (He10830, corona) you should select He10830. The other entry is not yet implemented. The data have names 'heyymmdd.hhmm', where yymmdd and hhmm designate, respectively, the date and time in UT. The data names followed by '.v' are Doppler images. By using the mouse, select the file you want to see. After you exit, the image data are stored in an array 'img', and the header h includes the observing parameters.

3.4 Okayama Vector Magnetograms

First you select the year (1982 -), and the file list appears. The data have names 'ymddn.D', where y is a letter indicating the year (B=1982, C=1983, ,,, M=1993), m indicates the month (1=Jan, ,, 9=Sept, A=Oct, B=Nov, C=Dec), dd is the date, and n=A,B,C,,, are sequence numbers of observation on the day. Then you are asked to enter several numbers from the keyboard as follows.

```
obs.area    : 0=skip, 1=show: _
     (location of the observed area on the sun)
```

```
halftone map: 0=skip, 1=    I, 2=   BL, 3=   VL: _
contour map : 0=skip, 1=    I, 2=   BL, 3=   VL: _
vector map  : 0=skip, 1=(   BX,   BY): _
```

Here I is the intensity map, BL is the longitudinal magnetogram, VL is the Doppler map, and (BX,BY) are the transverse field vectors. If 0 is entered, the corresponding map is not shown. If the potential field lines had been computed and stored in a file 'ymddn.L', you will be asked:

```
field lines : 0=skip, 1=show : _
```

At the next prompt

```
1=plot again, 2=hardcopy, 9=exit > : _
```

you can plot the map in a different style by selecting 1. The postscript output file will be created if you select 2. The postscript file can be later transferred by anonymous ftp.

After you exit, the data are stored in a floating array 'rdata', and the structure 'mparam' contains observing parameters. The 'mparam.status' includes fields such as i, q,u,v, bx,by,bl,,,,. If for example the data By are stored in rdata(*,*,2), mparam.status.by takes the value 2. In order to know which part of rdata contain valid data, type

```
IDL> help,/st,mparam.status
```

The data whose status value is -1 are not stored in rdata array.

The electric current distribution can be seen by typing

```
IDL> .run jn
IDL> magplot, mparam, rdata, /nomenu
```

Then you will see the following prompt.

```
obs.area    : 0=skip, 1=show: _
halftone map: 0=skip, 1=    I, 2=  BLP, 3=   BL, 4=   VL, 5=   JN: _
contour map : 0=skip, 1=    I, 2=  BLP, 3=   BL, 4=   VL, 5=   JN: _
vector map  : 0=skip, 1=(  BXP,  BYP), 2=(   BX,   BY): _
1=plot again, 2=hardcopy, 9=exit > : _
```

Here JN is the normal component of the electric currents, and the field components followed by P are those of the potential magnetic field.

4. Anonymous ftp Access to spot

When you are connected to 'spot' via anonymous ftp, under the home directory you have directories

```
flare                         : Flare Telescope magnetograms
         - log                : Flare Telescope magnetograms
monochro                      : H-alpha full-disk images
norikura - he10830            : He10830 spectroheliograms
               - corona
okayama  - 1982               : Okayama magnetograms
               - ...
doc                           : documents
idlsoft                       : IDL routines
```

All the IDL routines used here are stored under 'idlsoft'.
Compressed tar files of IDL software is provided as naojidl.tar.Z in the home directory. The 'doc' directory contains text files explaining the software and the data analysis.
The observation log files of the Flare Telescope are in /solar/flare/log.
The files 'yymmdd.log' give you the time and the observing region for all data on the day. The files 'lv##.log' contain list of H-alpha video disk recordings of the Flare Telescope T3.

5. Description of the Software

The 'show' program assumes the directory structure of NAOJ workstations. Therefore, when you want to display and analyse the data obtained via ftp, you have to invoke lower-level IDL routines.

5.1 Flare Telescope Vector Magnetograms

To read a file named 'fbyymmdd.hhmm' and store the data into img,Bx,By,Bl:

```
IDL> f1data, 'fbyymmdd.hhmm', h, img, Bx, By, Bl
```

Or an interactive program can be invoked by

```
IDL> .run fdisp
```

The map can be displayed by

```
IDL> magtv1, h=h, img, Bx, By, Bl
or
IDL> magtv2, h=h, img, Bx, By, Bl
```

or

```
IDL> magcont, h=h, img, Bx, By, Bl
```

To calculate and display electric current distribution:

```
IDL> .run jnft1
```

(The data should exist in h,img,Bx,By,Bl.)

5.2 Mitaka H-alpha Images and Norikura He10830 Heliograms

To read a file and store the image into 'img':

```
IDL> nkrget, <filename>, h, img
```

To display:

```
IDL> tvscl, img
```

5.3 Okayama Vector Magnetograms

To read a magnetogram file:

```
IDL> readm, mparam, rdata, filename=<filename>
```

To display magnetograms and potential field lines:

```
IDL> magplot, mparam, rdata, /nomenu
```

To calculate and display electric current distribution:
(The data should exist in mparam and rdata.)

```
IDL> .run jn
```

E Using the Laser Disk System at ISAS

E.1 Viewing Existing Movies (VIDEO_MENU)

This program provides interactive computer controlled viewing of the library of movie sequences recorded on the ISAS SONY laser disks. The user must be logged on to the isass3 computer (remote logon from another ISAS workstation is fine) in order to use the SONY laser disk player.

To run the software the user must log on to isass3. The calling sequence is:

IDL> **video_menu**

The user is prompted to check for the proper settings on the SONY disk drive, and then a listing is displayed from which the user selects a movie sequence for display. Upon selection the sequence is played repeatedly. While the sequence is playing the user may adjust the speed by keyboard input. The user may interrupt the movie at any time and return to the menu. The library of available movie sequences is updated manually as movies are recorded on the SONY, so the listing could be out of date.

In addition to this basic mode, VIDEO_MENU may be called with a keyword *vcr* set which allows non-interactive playing of frame sequences specified in an input file. This mode of operation is convenient for demonstrations and for making video tape copies of selected sequences of movies.

The centerpiece of the laser disk movie catalogue is the full disk movie which covers the duration of the mission. However, there are over 100 entries in the menu, corresponding to a variety of interesting sequences showing flares, large scale coronal transients, and theoretical modeling.

E.2 Making Title Pages

The video title pages may be prepared by using an IDL program called VIDEO_TITLE. First, you must prepare an ASCII control file that specifies what the title pages should contain. The IDL routine reads this file and writes the text to a Z-buffer window (using ordinary XYOUTS commands) and then reads it back into an array that can be stored in a `*.genx` file. The font type and size and placement can be easily adjusted. Some example calls are:

IDL> **array = video_title(filename)**
IDL> **array = video_title(filename,/plot)** ; Display the results
IDL> **array = video_title(filename,/plot,outfile='video1',/high)**
IDL> **array = video_title(filename,/plot,outfile='video1')**

The variable *filename* specifies the name of an ASCII control file that contains the text of the page. For example try the following:

IDL> **array = video_title('/ys/gen/doc/howto_video_title.doc',/plot)**

where `/ys/gen/doc/howto_video_title.doc` is the name of a "how to" file which is also an example control file. This file explains the format of the control file. For the first time users, it is recommended to copy that file over and make changes to it to accommodate your needs.

The control file specifies the position on the screen, the text, the size and thickness of the characters. Different fonts can be used also. A portion of that file is shown here:

```
;  howto_video_title.doc                J. R. Lemen  Rev A,  4-Jan-94
;
;  The control file contains 5 items per line separated by a double
;  slash or "//".   The first and second fields are the positions in
;  the 620 by 480 sized video window (0,0 is the lower left corner).
;  It is advisable not to write too close to any edge, since the
;  text may be lost, depending on the display monitor.  Setting xpos
;  equal to -1 will cause the text to be automatically centered (and
;  leading or trailing blanks will be discarded).  The third field
;  is the text and the fourth and fifth fields are the character
;  size and thickness, respectively.
;
;  The Fonts can be changed using the standard !n IDL vector commands
;  (see the example below).
;
;  This entire document is an example control file (since every line
;  except the important ones below begin with an ";").  Below the
;  dashed line is a specific example.  This file will cause two
;  pages to be generated (because of the NEWPAGE// line).  It is not
;  necessary to have more than one title page per file.
;
; -----------------------------------------------------------------
;  Semicolons are treated as comments
;  Separate fields with //
;  xpos // ypos // Text // size // thickness
;  (size and thickness will default to 1.0 if omitted)
;  (setting xpos to -1 will cause x position to auto-center)
-1//425//X-RAY CORONA OF THE SUN                            //2.8//2.8
-1//375//Images from the Soft X-ray Telescope               //2.1//1.4
-1//345//on the YOHKOH mission                              //2.1//1.4
-1//275//Data from:  November 11 to 27, 1991                //2.1//1.4
-1//220//!8Prepared by:                                     //2.1//1.4
-1//190//J. Lemen, G. Slater, K. Strong, and S. Tsuneta    //1.7//1.7
-1//100//Institute for Astronautical Research, Japan       //1.4/1.4
-1// 75//National Aeronautics and Space Administration, USA//1.4//1.4
```

```
-1// 30//(V1.0 13 April 1992)//1.4
NEWPAGE//
-1//300//Data from:  November 11 to 27, 1991              //2.1//1.4
-1//200//!18Displayed 100,000 times real time             //2.1//1.4
```

The */high* switch makes the routine run slow, but gives a higher quality result. It is recommended to debug the output without the */high* switch and only use it for the final result.

The $outfile =$ switch will cause the title pages to be written to a `*.genx` file that can be written to the ISAS Sony Laser disk using GO_NVS5. If you want the background color to be blue, make sure that you specify color table 1 in GO_NVS5.

E.3 Recording a New Movie to Laser Disk

E.3.1 Hardware: Positioning the Sony Laser Disk

1. Check that the green light on the NVS is on. Power cycle if necessary.

2. Check that the RS-232 cable is connected from the NVS to the Sony LVR-5000.

3. Make sure the correct optical disk is inserted (check side A or B).

4. Put the 'LOCAL/REMOTE' switch in the local position.

5. Depress the 'REC STANDBY' button to initiate blank frame check. This will find positions of all blank areas on the disk. When completed, the disk will be positioned to the last recorded frame before the nearest (not necessary the first) blank frame.

6. Use the record 'SKIP' buttons to position the laser disk to the last recorded frame before the desired starting frame number (consult the Sony Laser disk log book). On the display, Cont. Rec = is the last recorded frame before the blank frame; DUR = number of blank frames.

7. Toggle the format of the numbers displayed to show the time-format. Record numbers in the log book. To start over, press the 'CLEAR' button.

8. Set the left console switch to 'remote'.

E.3.2 Software

1. Login onto a non-isass0 machine (isass1-4).
2. Enter IDL. Make sure no other IDL window jobs are running on the same workstation. If there are two jobs, the color table could be corrupted.
3. Then enter one of the following:

IDL> **go_nvs5,/sfd**	; Standard weekly movie making
IDL> **go_nvs5,/sfd,/nosfw**	; Standard weekly movie making
IDL> **go_nvs5,/help**	; For on-line documentation
IDL> **go_nvs5,/genx**	; to use a GENX type file as the input

The temporary NVS files will be written to your home directory. Use the nvsdirfn='/1p/user/nvs0' switch to redirect to another location.

For the SFD option, the default increment is 10 (process 10 images at a time). This is good if no one else is using the workstation at the same time. If the /sfd switch is given, go_nvs5 will prompt the user with the list of files which are in the /yd1/sfd directory.

For the GENX option, the array will not be autoscaled (it should be a byte type). If the number of images in the genx file is large, IDL may run out of core. In that case, try increment=1. The program will prompt the user for the GENX filename.

E.3.3 Adding Another Movie

The above procedure is for starting from scratch. If you have just completed a run and the Sony frame number has not been changed, then just type:

IDL> **go_nvs5** [**,/sfd**]

to make the next movie.

E.3.4 NVS/Sony Problems

- Invalid Num – If the message 'Invalid Num' appears on the display monitor, the nvs box is not in sync with the Sony laser disk. To correct this situation begin with a blank frame check (see above hardware procedure) and follow that with software procedure shown above.
- One common problem is to forget to attach the RS-232 cable. This is sometimes switched to ISASS3 to control the recorder during playback to record with the VCR).
- How to toggle the format frame/time format: depress the frame/time button located on the lower-left side of the console number keypad–where frame # is a simple number and time has the format of mm:ss:ff.

E.4 Transferring a Movie to VCR Video

MK_VCR is an IDL procedure which can be used to control the Sony Laserdisk recorder. In this way sequences can be displayed easily in order to make VCR video recordings.

The MK_VCR procedure reads an ASCII file (described) below and then calls sonyloop to actually control the laser disk player.

E.4.1 Initial setup

1. First, make sure the RS-232 cable from isass3 is connected to the back of the Sony recorder (it requires an adapter connector).
2. Switch the local/remote switch to remote.
3. Log on to isass3 and enter IDL.

E.4.2 Short test of hardware communications.

You can preform a simple test to see if things are working correctly by typing:

IDL> **sonyloop,6000**	; Position laser disk to frame number 6000
IDL> **sonyloop,6000,6200**	; Play from 6000 to 6200 at normal speed
IDL> **sonyloop,6000,6200,sec=2**	; Play a factor of 2 slower

E.4.3 Running MK_VCR

If things look like they are working correctly, you can then try mk_vcr. To get a template of the file the mk_vcr will read, execute the following Unix command to copy the example file to your own directory:

% **cp /ys/gen/data/mk_vcr_example.txt .**

After you copy the example file, do the following. Insert Side B of the laser optical disk. Then,

IDL> **mk_vcr,'mk_vcr_example.txt',1,3**

which execute lines 1 to 3 of the file. (Line 0 puts up a blank frame). If you type

IDL> **mk_vcr,'mk_vcr_example.txt**

the routine will inform you of the number of lines in the file and ask you to enter the start and stop line numbers.

If you want to make a movie, you could try the following IDL program:

IDL> **sonyloop,8000**	;Get it to a blank frame
IDL> **for i=0,2 do mk_vcr,'mk_vcr_example.txt',0,23**	;There are 24 lines in the file

IDL> **sonyloop,8000** ;End on a blank frame
IDL> **end**

As you can see, the sequence in the 'for loop' will execute 3 times. Each time through takes about 7 minutes, so the whole movie will be just over 20 minutes in length.

E.4.4 How to create/modify the file that MK_VCR uses

The file has some descriptions about how to change it. Basically, the lines in the file look like the following:

```
 8000 ; Position to a blank frame (provide leader)
 6077, 6077,    4,    0, [ -1  0  0  0] "Movie Title Page"
 6078, 6078,    2,    1, [ -1  0  0  0] "Nov 1991 Movie - Title"
 5048, 5225,    2,    1, [  1  2  1 -1] "Nov 1991 Movie"
; c0    c1      c2    c3       [c4]          "  c5   "
```

The above sequence will display the Nov 91 press release sequence if Side B is inserted. Some notes about the format:

- Text following a ':' is considered to be a comment.
- Only the first column is required. All subsequent columns are optional.
- c0 is starting frame number.
- c1 is the ending frame number.
- c2 is the wait (in sec) after displaying the first frame
- c3 is the wait (in sec) after displaying the last frame
- c4: The number in the boxes are the speeds that are used by sonyloop. Up to four different speeds can be specified.

```
0 is no operation
1 is normal speed (30 frames per second)
2 is twice slower (15 frames per second)
3 is 3x slower, etc (10 frames per second)
-2 is 3x faster.
```

If a fifth parameter is specified in the brackets, it is a repeat number. If this number is negative, sonyloop will play the sequence forward and backward. For example,

```
 5048, 5225, 2,  1, [  1  0  0  0  1] "Nov 1991 Movie" ;1 time forward
 5048, 5225, 2,  1, [  1  0  0  0 -1] "Nov 1991 Movie" ;1 time forward/backward
 5048, 5225, 2,  1, [  1  0  0  0 -3] "Nov 1991 Movie" ;3 times forward/backward
```

SPECIAL NOTES:

- Sonyloop doesn't send the correct commands to the Sony Laser Recorder to go backwards. Thus, no matter what speed you specify, it always goes at normal speed in reverse. (Maybe a Japanese reader can study the manual and fix this problem).

- Sonyloop tries to compute the time it will take the laser disk recorder to play the specified sequence. Because the timings in IDL can be off by a certain amount (less than or equal to about a second), it is wise to make one of the wait parameters (c2 or c3) equal to at least 1. If they are set to 0, IDL will sometimes send the next sequence before the previous sequence has had a chance to complete.

F Using the Laser Disk System at LPARL

The following is a guide to record and play laser disk movies at LPARL.

F.1 Viewing Existing Movies

F.1.1 Using IDL and Menu Driven Software

The program VIDEO_MENU described in section E.1 is not available at LPARL during the writing of this guide, however it may exist now. See one of the software engineers to see if it is available.

F.1.2 Displaying a Single Recorded Image

A. Manual Operation

The Panasonic optical disk recorder/player and the Panasonic optical disk player may be "manually" controlled by using either the buttons on the front of each machine or the remote controllers. These buttons and the remote controller are labeled in English and are easily understood.

B. IDL Operation

1. Log onto sxt0 on any terminal
2. Enter IDL
3. Suppose you want to display frame 1000. Here's the command:

   IDL> **pana,1000**

F.1.3 Playing the Panasonic Laser Disks

A. Manual Operation

The Panasonic optical disk recorder/player and the Panasonic optical disk player may be played "manually" by using either the buttons on the front of each machine or the remote controllers. These buttons and the remote controller are labeled in English and are easily understood.

B. IDL Operation

1. Log onto sxt0 on any terminal
2. Enter IDL

3. Suppose you want to play frames 100 to 200 on the recorder for 10 repetitions at a rate of 90 frames per second. Here's the command:

   IDL> **movie_pana,100,200,rate=10,num_repeat=3**

4. The general command is

   IDL> **movie_pana,start_frame,stop_frame,device,device=device1,rate=rate, $**
   IDL> **num_repeat=num_repeat,backforth=backforth**

   The device parameters should be ignored because at the present time there is only one available device, which is the default. More information on this routine can be obtained with the command:

   IDL> **doc_library,'movie_pana'**

F.2 Making Title Pages

See the description in the previous chapter on page 90 describing the laser disk system at ISAS.

F.3 Recording a New Movie to Laser Disk

1. Turn on power to Panasonic recorder and Sony monitor labeled 'Laser Disk Recorder'.

2. Find the disk you want to record on (it may be in the recorder). Note that each disk has two sides. The 'up' side is the active side. Note also that there is blue 'write protect' sliding switch on the right side of each side of the disk. Move this switch to its right hand position ('read/write') using a Bic pen or equivalent tool. Now insert the laser disk you want to record on into the recorder after first ejecting the disk already in there if there is one.

3. Log onto sxt0 on an X terminal

4. Enter IDL

5. At the prompt do this:

   IDL> **ptest**

   Look at the monitor. If after a few seconds a test pattern appears the equipment is properly set up and you may proceed. If no test pattern appears or if it looks unusual then something is wrong. Refer to section IV on problems below.

6. Images may be recorded either from an SDA file or from a data cube resident in memory. The images will be re-sized to fit within the 630 x 486 size of the Peritek video image. The re-sizing will use linear interpolation unless the /NOINTERP option is used, By default they will be centered. The images may be of any size. If they are in a data cube then the entire cube will be normalized to its maximum value and byte scaled, unless the /NOSCALE keyword is used in the procedure call. If the images are in an SDA file the images must be pre-scaled by the user. Decide on a color table. A gray scale (IDL standard color table 1) will be used by default. If the color table desired is not one of the first 15 IDL standard color tables then you must supply r, g, and b color vectors.

7. Decide what frame on the laser disk you wish to start recording at. The IDL recording program will automatically find the first sufficiently large unrecorded area if you skip this step. To do this manually, type:

   IDL> **command_pana,'OF'**

   Type the 'display' button on the remote controller to display the current frame number on the monitor. Then use the remote controller to play or scan through the disk until you find the frame number at which you want to start recording. Key the number of frames you wish to record using the keypad. Then hit the red 'rec mode' key. This will verify that there are enough contiguous blank frames available following the selected frame.

8. Suppose you have the cube in a 3-D IDL array called DATA and the color vectors in the variables R,G, and B. Here's how to record it:

   IDL> **pana_record,data,r=r,g=g,b=b,/record**

9. For recording images from an SDA file the call would be

   IDL> **pana_record,file_name,r=r,g=g,b=b,/record**

   where file_name is a string naming the SDA file, including path. More information on this routine can be obtained like this:

   IDL> **doc_library,'pana_record'**

F.3.1 Panasonic Problems

If no test image appears after issuing the PTEST command in the instructions described in the previous section, then do these things:

- Make sure that the power switch on the back of the Peritek 'clock box' is in the on position. The clock box is located in the middle shelf of the SXT0 computer cabinet.
- Check that the four gold colored video cables from the Peritek are connected to the RGB and sync terminals on the back of the recorder.

F.4 Transferring a Movie to VCR Video

As of the writing of this document, the routine MK_VCR (which is described on page 93) was not available for the LPARL video system. The routine may be expanded in the future. The best method for transferring a laser disk movie to VCR video is to talk to one of the LPARL software engineers.

G Using Magneto-Optical (MO) Disks at ISAS

A complete archive of the Yohkoh data for the whole mission is available on Magneto-Optical (MO) disks at ISAS.

A Unix script and associated alias is available to mount and dismount MO (magneto optical) disks at ISAS. An IDL interface routine is also provided to allow MO mounts and dismounts from within an IDL session or callable from other IDL routines. Once mounted, access to data on these drives is identical to other magnetic disks — standard Unix operations (ex: ls) are permitted. For MO disks containing Yohkoh data, the standard IDL functions (ex: yodat access) are available.

MO disks are available on several machines of the SUN machines at ISAS. including flare3, flare4, flare6, flare13, flare14, flare15. The first (and in most case, only) MO drive on any machine is referred to as drive 0. Some machines may have a second drive (drive 1) — all drives are clearly marked with a drive number. Find an available (empty) drive and insert your disk — if you intend to write, verify that the write protect switch allows writing. By default, the mount is executed on the machine which contains the MO driver. If that machine is different than the machine which executes the command, then the MO disk is automatically 'exported' and mounted on the local machine. Use the /all switch to force mounting on all ISAS machines.

G.1 Mounting MO Disks From Unix

The alias *momount* executes the script which performs all MO operations. Parameters and switches allow specification of machine name, drive number, access mode, dismounts, etc. Options include */write /umount /eject /1* and */all*. Some sample calling sequences are:

% **momount flare13**	# mount flare13 , drive 0, readonly
% **momount flare13 /umount**	# dismount same
% **momount flare4 /1 /write**	# mount flare4, drive 1, read/write
% **momount flare4 /1 /eject**	# dismount and eject same
% **momount flare15 /all**	# mount flare15, drive 0, on all nodes

G.2 Mounting MO Disks From IDL

An IDL driver sets up the parameters and spawns a call to the same script file described above to allow IDL MO control (interactive or via routine). Keyword options similar to the Unix script are available with the IDL routine. Some examples are:

IDL> **momount,'flare13'**	; mount flare13 , drive 0, readonly
IDL> **momount,'flare13',/umount**	; dismount same
IDL> **momount,'flare4', /one, /write**	; mount flare4, drive 1, read/write
IDL> **momount,'flare4', /one, /eject**	; dismount and eject same

H Guides to Yohkoh Operations at ISAS

The following documents are useful for understanding different aspects of the operations of Yohkoh.

/ys/gen/doc/ssoc_tohban_manual_*	A detailed description of how to perform the SSOC tohban duties
auto_toban2.pro	The header of this program describes in detail the different steps on how the daily KSC reformatting is performed
/ys/gen/doc/ref_super.doc	This document describes the different steps which are performed during the standard reformatting
/ys/gen/doc/cron.doc	Describes the different cron jobs that are run automatically at ISAS and LPARL to process data, distribute software and data, and to copy data from GBO sites
/ys/gen/doc/howto_term_images.doc	Explains the procedure SSOC Tohbans should follow when planning SXT terminator images
/ys/gen/doc/krasm.doc	Explains how the SSOC Tohbans may access the RASM scheduling computer by modem