CGRO/OSSE
Pipeline Documentation

W.T. Bridgman
Compton Observatory Science Support Center
Goddard Space Flight Center
Code 660.2
Greenbelt, MD 20771
April 23, 1999

Progress report and documentation for designing, building, and maintaining the CGRO/OSSE data pipelines.

Contents

Chapter 1

Running the OSSE Low-Level Pipeline

1.1 Intent

This document describes procedures that the Archive Technician shall follow to process low-level OSSE data (spectral and pulsar) at the COSSC. These procedures will manipulate data from native format on the delivered tapes to compressed FITS format in the COSSC archive.

1.2 Outline

Low-level OSSE data are generally delivered one viewing period at a time on three separate 4mm DATs (see Section 1.3).

The procedure for processing low-level OSSE data is composed of four main segments which run sequentially. All work is done in the OSSEPIPE account on VMS and the ARCHIVER account on Unix.

1. Restores data files in native VAX/VMS format from tape to disk.

2. submits a batch job that converts the native format files to FITS files in a staging area. It also uses VMS TAR and VMS BACKUP to package up text and other files into a single file. The files are also compressed with VAX GZIP in this staging area. The staging area is NFS mounted between the VAX and Unix clusters – an

aspect important for the operation of the third segment of this process. These first two segments run exclusively on VAX/VMS. Attempts to process them on the Alpha/VMS platforms ran into problems with the difference in numeric formats between the VAX and Alphas (see Section A.1).

3. The next segment extracts the FITS keywords necessary for the W3Browse database and then transfers the data files to the archive. The IDL routine MRDFITS is used for the keyword extraction since this function can very quickly read the FITS keywords from the primary header of compressed files. This final step is run on the Unix systems and requires the input FITSified data be available on the Unix side (hence the NFS mounting in the previous step).

4. The final segment generates the TDAT file used by the HEASARC database engine for accessing the archive.

It may be worthwhile in the future to incorporate the PDF file processing into the SDB pipeline.

For control purposes, a processing form is maintained for each data delivery and installation into the archive. For this portion of the pipeline, this form is called OSSEPIPE-01. You should utilize the most recent release of this form.

1.3 Data Tapes

Data from the OSSE instrument are delivered by the Naval Research Laboratory (NRL) after the proprietary term expires. Typically, a delivery consists of three tapes containing OSSE data for a single viewing period. Each tape is labelled with a code composed of six characters. The first two characters can be:

OA	SDB (spectral) data
OB	PDF (timing & pulsar) data
OD	TELEMETRY data (which is not archived)

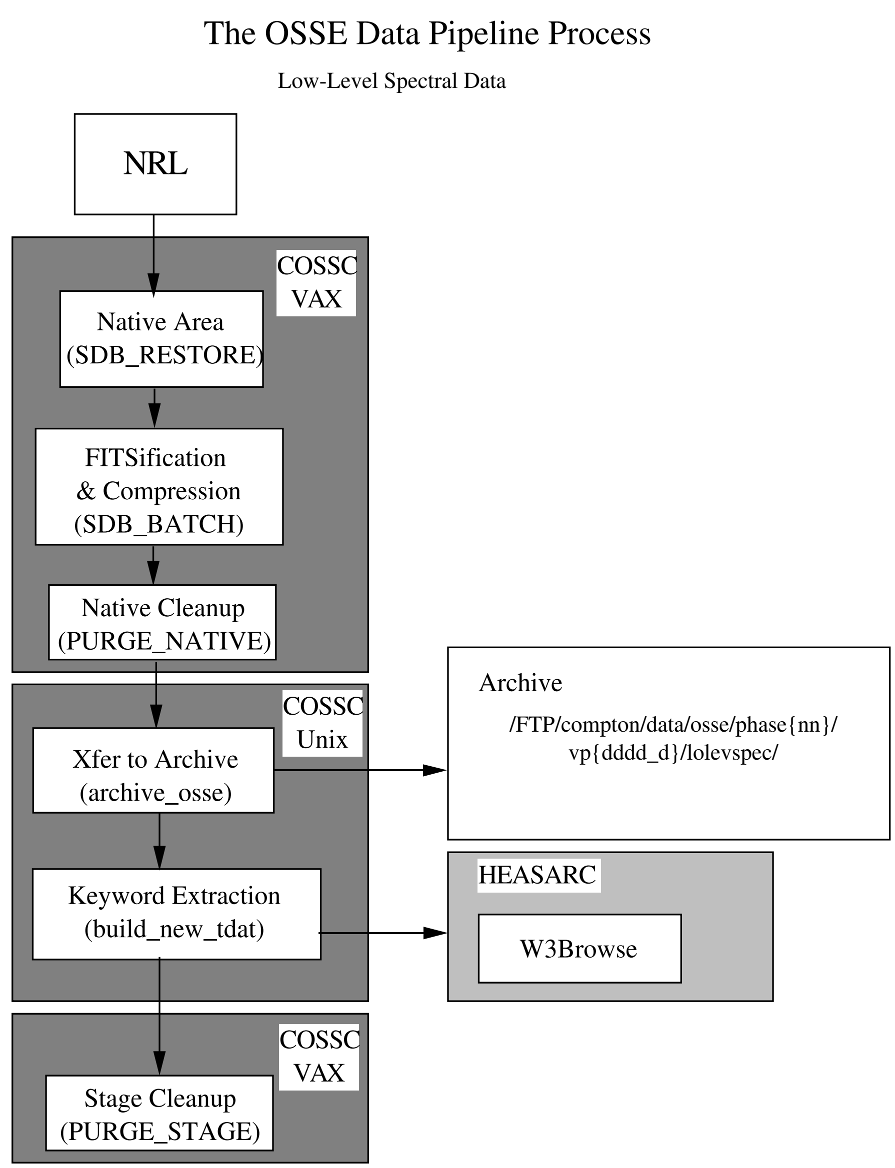

Figure 1.1: Diagram of the pipeline for OSSE low-level spectral data. In the archive path specification, {nn} is the phase number and {dddd_d} is the 6 character viewing period with the underscore replacing the decimal point.

The last four characters of the label are digits denoting the first day of the viewing period. Specifically, the first digit in the code is the last digit of the year number, and the last three digits of the code are the day number within that year. For instance, a tape labeled "OB2136" contains PDF data whose viewing period began on the 136th day of 1992 (year number ends in 2).

Log listings of the contents of the delivered data tapes are stored in $DISK10:[INVENTORY.OSSE].

1.4 Preparing the Pipeline Run

For ISO-9000 and control purposes, each viewing period processed will be recorded on a pipeline control form. For the low-level spectral pipeline, the form is designated OSSEFORM-01.

When the data tapes arrive and are recorded in inventory, the header of the form should be filled out. The header records the basic inventory information - CGRO viewing period number, the data delivery number, tape identifier label, and the tape delivery date.

Get the viewing period date and length from the CGRO viewing plan.

The length of the proprietary period is determined by the CGRO phase number per the table below:

phase	length
≤ 6	12 months
7	6 months
8	3 months
≥ 9	none

From this, determine the date of the end of the proprietary period and record it on the form. If this date has passed, you may continue with the steps to load the data into the archive.

1.4.1 Notes

The total disk space required to process an entire delivery is substantial. In addition to the space required to restore the files from tape to disk in native format, the FITS conversion process will generate output files with an even larger space requirement. For low-level SDB files, for example, the output FITS files in the staging area will require approximately 1.6 times the space occupied by the input files. It is the responsibility of the Archive Technician to check that sufficient disk space exists before performing the processing, and to verify that the entire procedure has completed successfully afterwards.

1.5 Running the Data File Restoration Program

The program to restore data files from tape to disk assumes that the desired tape is loaded in tape drive connected to GROVX0 and referenced by the symbol OSSE_TAPE, and must be run from the OSSEPIPE account on GROVX0. The restoration software includes performing the allocation and mounting of the tape drive.

To execute, type:

```
@SDB_RESTORE  [deliv_num]  [tape_label]
```

or

```
@PDF_RESTORE  [deliv_num]  [tape_label]
```

where [deliv_num] is the tape delivery number and [tape_label] is the identifier marked on the tape.

For example, the command

```
@SDB_RESTORE 111 OA3361
```

will restore all files from tape OA3361 in delivery 111.

1.6 Running the FITS Conversion Program

The next step is to convert the data files from native format to FITS format. This is accomplished by submitting the format conversion program to a batch queue from the OSSEPIPE account.

To execute, type:

```
@SDB_BATCH  [deliv_num]  [view_period]  [time]
```

or

```
@PDF_BATCH  [deliv_num]  [view_period]  [time]
```

where [view_period] is the corresponding viewing period number (note there should always be one digit to the **right** of the decimal point) and [time] is an (optional) start time for the batch job [hh:mm]

For example, the command

```
@SDB_BATCH 111 313.0 20:00
```

will submit the FITS conversion software as a batch job that will begin at 20:00 to process the SDB files from delivery 111, containing data obtained during viewing period 313.0.

Special Case of Two Viewing Periods in One Delivery

As noted in Section 1.3 above, a data delivery from NRL typically covers one viewing period. On occasion, however, a delivery may contain data from two viewing periods. No indication exists on the tapes themselves, however, as to the demarcation date between the two viewing periods. For such cases, the Archive Technician must refer to the CGRO observation timelines to determine this date. Then, a subdirectory appropriately numbered for the second viewing period should be manually created in the staging area, and the corresponding FITS files moved into that directory. A one-day overlap of data files should be maintained between the two viewing periods.

1.7 Final Installation into the Archive

On completion of FITSification, the pipeline activity moves over to the Unix side of the system in the ARCHIVER account. Due to the file naming convention for the resulting TDAT files, this step can be run only once per day. Also note that case is important when typing in these commands.

Once in Unix

```
>cd osse
>idl
```

whichs sets up the account to run IDL v5.0 and starts IDL. Once in IDL, we can transfer the files to the archive:

```
.run tdat_library
.run archive_osse
```

The archive_osse process builds the output directory structure and transfers the data files to the archive. It will generally run for a while (an hour or so). The final step is

```
.run build_new_tdat
```

which examines the FITS headers and extracts the information to generate the data tags for the *entire* OSSE archive. This data is tabulated in a pipe-delimited file: osse_nnnnnnn_build.tdat which appears in the osse_tdat directory on completion of this program (nnnnnnn is the Julian Day number). A copy of this file is also sent (via FTP) to hrcdba@dbsrv:/dba_dbase/work/osse/hdb/. On completion, an e-mail notification is sent to dba_browse@lheamail.gsfc.nasa.gov. When this process completes, type

```
exit
```

to leave IDL.

1.8 Clean-up Operations

As a precaution, the pipelines perform only a limited amount of 'clean-up' behind them to facilitate restarting in the event a run fails. The actual clean-up is best performed manually after there has been plenty of time to ensure the previous runs did complete successfully.

There are two subdirectory areas that should be empty before initiating a new pipeline run. WARNING - MAKE SURE THAT ALL OSSE PIPELINE PROCESSES ARE COMPLETE BEFORE RUNNING THESE PROGRAMS.

- The directories *below* $DISK20:[OSSEPIPE.NATIVE] should be gone. This is the location were data is placed when restored from the delivered data tapes. This can be done with the command:

  ```
  @PURGE_NATIVE [dn]
  ```

  where [dn] is either the data delivery number, to delete a specific delivery, or '*' to delete all low-level data deliveries from the native area. The program prompts you with a warning of what it is about to do. Nonetheless, use it with caution.

- The directories $DISK20:[OSSEPIPE.STAGE.PHASE*...] should all be emptied and deleted. This is the staging area where the FITSified data files are placed after creation and from where they are retrieved for compression and copying into the actual archive. This can be performed with a single command:

  ```
  @PURGE_STAGE [type] [vp]
  ```

  where [type] is the data types: lolevspec, hilevspec, pulsar, or '*' to purge all types; and [vp] is the viewing period you desire to delete, or '*' to delete all.

Chapter 2

Running the OSSE High-Level Pipeline

2.1 Intent

This document describes procedures that the Archive Technician shall follow to process high-level OSSE spectral data at the COSSC.

2.2 Outline

The procedure for processing OSSE data is composed of three main segments that run sequentially. All work is done in the OSSEPIPE accounts on the VAX and Unix platforms.

1. Restores data files in native format from tape to a VAX-mounted disk.

2. Submits a batch job which FITS-ifies the SDB files and uses VMS TAR to package other support files. All files are then compressed using GZIP.

3. The third segment runs under Unix and copies the FITSified data products from the staging area into the actual archive.

The complete pipeline is illustrated in Figure 2.1).

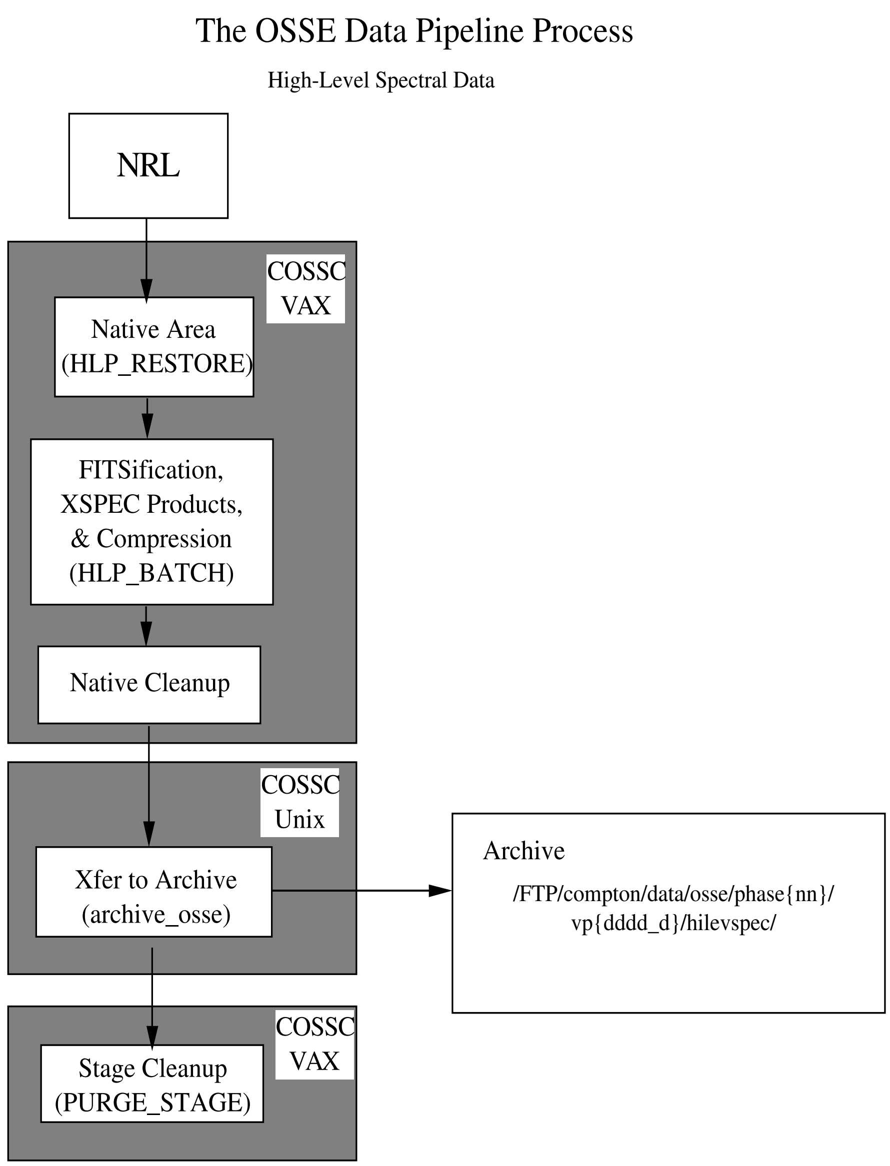

Figure 2.1: Diagram of the pipeline for OSSE high-level spectral data. In the archive path specification, {nn} is the phase number and {dddd_d} is the 6 character viewing period with the underscore replacing the decimal point. {target_name} is a version of the target name modified for compatibility with the file name syntax.

2.3 Data Tapes

Tape labels for high-level OSSE data products have "OE" in their label prefix.

2.4 Running the Data File Restoration Program

The high-level pipeline does not generate TDAT products for installation in the HEASARC database. Therefore, any high-level data products processed **before** the low-level data products will not be visible in W3Browse. In general, the low-level data products should be placed online **before** their high-level counterparts.

The high-level data products are placed online through a DCL script. The data tape is written in VMS BACKUP format.

1. You must be logged in on the OSSEPIPE account on the node GROVX0 (GROVX0 is the only node which has access to the 4mm DAT).

2. Mount the tape in the appropriate tape drive. From the keyboard, type the command

   ```
   $ @HLP_RESTORE
   ```

   This process will restore the backup set to disk in the area indicated in the DCL symbol OSSE_HLPNATIVE. When complete, the procedure will dismount the tape. Remove the tape from the drive and file it appropriately.

This process also performs the step of fixing a file name problem with the *BLN.SDB files. This is a bug which should only affect older deliveries. Basically, the portion of the BLN file prior to .SDB may be truncated due to an old 20-character filename limit in an earlier release of the IGORE package. This process actually occurred at NRL.

2.5 FITSification and Generating Other Data Products

This process requires IGORE (and IDL) so it must be run on a VAX. This is also run from the OSSEPIPE account but on the GROSSC machine.

Run the program from your root directory on VAX with

```
$ @HLP_BATCH [deliv_num]
```

where [deliv_num] corresponds to the high-level delivery number (without the letter). This procedure submits the job to the GROSSC_ONEJOB$BATCH queue. This queue has no time limit since this process may take a long time to run.

This procedure assumes the High Level OSSE products are available in OSSE_HLPNATIVE. It consists of three major steps:

1. FITSify all SDB files. Generates the XSPEC products if TOT files are in the data set. Wrap other supporting files and compress the entire set.
2. Scan the High-Level directory and build/update the status file.
3. Generate XSPEC files for the appropriate pointings. These are placed in the High-Level directories in OSSE_STAGE.

On completion of the FITSification, you should clean out the native high-level directory *before* performing another FITSification run.

The directories *below* $DISK20:[OSSEPIPE.NATIVE.HILEV] should be gone. This is the location were data is placed when restored from the delivered data tapes. This can be done with the by first enabling file and directory deletion with the command:

```
SET PROTECTION=OWNER:RWED \$DISK20:[OSSEPIPE.NATIVE.HILEV\ldots]*.*
```

then delete them with with several applications of the command:

```
DELETE \$DISK20:[OSSEPIPE.NATIVE.HILEV\ldots]*.*;*
```

Exercise extreme caution when performing this step. Currently, PURGE_NATIVE does not support deletion of the high-level data products.

2.6 Final Installation into the Archive

On completion of FITSification, the pipeline activity moves over to the Unix side of the system in the ARCHIVER account. Due to the file naming convention for the resulting TDAT files, this step can be run only once per day. Also note that case is important when typing in these commands.

Once in Unix

```
>cd osse
>idl
```

whichs sets up the account to run IDL v5.0 and starts IDL. Now we can transfer the files to the archive:

```
.run tdat_library
.run archive_osse
```

and when it completes, type

```
exit
```

Note that (unlike the low-level pipeline, you do *not* need to run build_new_tdat. This program will build the output directory structure in the archive and copy the files from the staging area to the archive directory.

2.7 Clean-up Operations

As a precaution, the pipelines perform only a limited amount of 'clean-up' behind them to facilitate restarting in the event a run fails. The actual clean-up is best performed manually after there has been plenty of time to ensure the previous runs did complete successfully.

There are two subdirectory areas that should be empty before initiating a new pipeline run. WARNING - MAKE SURE THAT ALL OSSE PIPELINE PROCESSES ARE COMPLETE BEFORE RUNNING THESE PROGRAMS.

The directories $DISK20:[OSSEPIPE.STAGE.PHASE*...] should all be emptied and deleted. This is the staging area where the FITSified data files are placed after creation and from where they are retrieved for compression and copying into the actual archive. This can be performed with a single command:

```
@PURGE_STAGE [type] [vp]
```

where [type] is the data types: lolevspec, hilevspec, pulsar, or '*' to purge all types; and [vp] is the viewing period you desire to delete, or '*' to delete all.

Chapter 3

OSSE Pipeline Troubleshooting

3.1 Screwball Stuff

Items listed here are some general 'oddities' in the way the pipelines operate. They should be noted and kept in mind when running the pipelines.

- If you process a given viewing period of high-level data products *before* the corresponding low-level data products, the high-level products will *not* be accessible via W3Browse until the low-level products are placed on-line. This is because the TDAT file which is ingested into the W3Browse database usually generated after the low-level pipeline is run.

- The GZIP (used in all runs for compressing the data files) program *will not* overwrite an existing file. This can be a problem if you try to overwrite existing data in the staging area. You should always run SDB_BATCH and HLP_BATCH with the appropriate staging area clean of a previous data run.

- In general, if you experience some type of run failure, the safest course of action is to delete the products from the NATIVE and STAGE areas, reload the data from tape, and rerun the process.

- In principle, build_new_tdat can be run at any time. (Make sure you run tdat_library beforehand.) This procedure generates a TDAT file for the *entire* OSSE archive and replaces any previous TDAT files.

3.2 Process Logging Files

3.2.1 Master Logging File

Stored in the [.logs] directory. Contains the log file of the overall process. This includes messages from FITSification as well as other processes in the pipeline.

3.2.2 Success & Failure Logs

Also stored in the [.logs] directory. The failure log is mailed to the process owner.

3.2.3 Single File FITSification Log

These are stored in the [.native] subdirectory tree. Empty ones are deleted at the close of the run.

Error Messages in Single File FITSification Logs:

targname: used default target name for target ID 0 *BURST.SDB and *SLOT.SDB have no pointed targets. For them, this message is okay.

targname: used default target name for target ID 0 If the error is from an *SPM.SDB file and the target ID corresponds to an actual target, it may be because the conversion program cannot find the PLANNING_CATALOG.CAT file. The pipeline points to the primary catalog in the local IGORE installation. If IGORE has been changed, this may be the problem.

PREP2FITS: TTYPE is missing from primary binary table. onversion process died. If there are no spectral records in the file, you will get this message from FITSification. This can happen to *BURST.SDB, *C60.SDB, and *SLOT.SDB files. This should have a corresponding entry in the failure log.

The failure log is mailed to the OSSEPIPE account on completion of FITSification. It and the Success Log are stored at the symbol OSSE_LOGDISK.

OSSE_FAILURE_DNdddd.LOG

3.2.4 Failure Logs

field name	value	interpretation
filename	-	Name of input SDB or PDF file
type	SDB,PDF	File type
error type	convert reconvert compare	Generated by SDB2FITS Generated by FITS2SDB Generated by SDBCOMPARE
filesize		File size in blocks
delivery number		
viewing period		
date		Date processed
message	'Unknown at this time'	
	y	??
	n	??
SW version		software version

3.2.5 Success Logs

OSSE_SUCCESS_DNdddd.LOG

field name	value	interpretation
infilename	-	Name of input SDB or PDF file
outfilename	-	Name of output FITS file
type	SDB,PDF	File type
filesize		
delivery number		
viewing period		
date		
SW version		software version

3.3 Error Conditions

Error in BURST files Upon completion of the FITS conversion process, the failure log for this portion of the processing is sent to user OSSEPIPE as an e-mail message. Always check the OSSEPIPE e-mail after a run completion. Entries in the failure log which have fields 2, 3, and 4 filled with the string "SDB,convert," may be safely ignored. For example, the following log entry is an acceptable processing event:

```
$disk9:[osse.native.deliv_100]P933010000_BURST.sdb,SDB,convert,
1,26,305.0,7-OCT-1995 09:48:24,Unknown at this time,y,n,6.0
```

This error is probably generated by an empty OSSE BURST file. Before ignoring it, check the file size of the BURST SDB file in the native directory. If it is only a few blocks in size (these files are usually several hundred to several thousand blocks) then this is the most likely source of the error and this message can be safely ignored.

If FITS file shows target not found The planning_catalog.cat file may have been changed. Pipeline points to the primary catalog in the local IGORE installation. If IGORE has been changed, this may be a problem.

Failures of SLOT.SDB files. Due to a processing error at NRL. Contact the OSSE team (what's left of it) to have the files reprocessed and re-delivered.

3.4 Processing Log Files

Appendix A

Things Your Mother Never Told You...

A.1 Alpha/VAX Double Precision Numeric Formats

They're different! Don't try to read a double precision floating point number written by a VAX on an Alpha and vice versa. I'm told there's a compiler flag to get around this problem on the Alphas but have never used it. Most OSSE pipeline stuff is written to be run on the VAX and should not be ported to the Alpha unless you know what you're doing!

Appendix B

OSSE Data Files

B.1 Low-Level Spectral Database Files

B.1.1 Introduction

The OSSE Spectral Database Files (SDB) are VAX/VMS indexed files with variable length records. Each record can be one of a multiple of 'types', called AUX-types. Each AUX-type contains a specific type of information and individual SDB files can contain more than one AUX-type.

B.1.2 Spectral Memory File

File Name Format: Pyyddd0000_SPM.SDB

Product Type: Low-level

File Format: VAX/VMS, indexed, variable length records

Source:

Contents: Raw primary detector count spectra, generally two minute sums. Source and background pointing for an entire day of observations. Spectra are stored as I*4 arrays.

Archived?: Yes. pyyddd000_spm.fits

AUX Record Types:

DAT	0	spectral counts
CAL_LINE	25	Calibration line information
ORBHIST	29	Orbit history information

B.1.3 Shield Data

File Name Format: Pyyddd0000_BURST.SDB

Product Type: Low-level

File Format: VAX/VMS, indexed, variable length records

Source:

Contents: 4096 time bins measured from the shields. The time resolution is adjustable down to 4 milliseconds. These are stored as I*4.

Archived?: Yes. pyyddd000_burst.fits

AUX Record Types:

DAT	0	spectral counts (long integer & Reals)
MULTI_SPEC_DSF	19	burst spectrum?
SDB_DIR	42	SDB directory

B.1.4 Cobalt 60 Spectra

File Name Format: Pyyddd0000_C60.SDB

Product Type: Low-level

File Format: VAX/VMS, indexed, variable length records

Source:

Contents: Spectra of the cobalt 60 calibration source. Stored as I*4.

Archived?: Yes. pyyddd000_c60.fits

AUX Record Types:

DAT	0	spectral counts

B.1.5 Summary of Cobalt 60 Spectra

File Name Format: Pyyddd0000_C60SM.SDB

Product Type: Low-level

File Format: VAX/VMS, indexed, variable length records

Source:

Contents: Sums of the cobalt 60 calibration spectra. Stored as R*4.

Archived?: Yes. pyyddd000_c60sm.fits

AUX Record Types:

DAT	0	spectral counts
CAL_LINE	25	Calibration line information
REL_SPEC	28	Related spectra

B.1.6 Slew Spectra Data

File Name Format: Pyyddd0000_SEC.SDB

Product Type: Low-level

File Format: VAX/VMS, indexed, variable length records

Source:

Contents: Spectral records accumulated during detector slews between sources. Spectra are stored as I*4.

Archived?: Yes. pyyddd000_sec.fits

AUX Record Types:

DAT	0	spectral counts

B.1.7 SLOT?

File Name Format: Pyyddd0000_SLOT.SDB

Product Type: Low-level

File Format: VAX/VMS, indexed, variable length records

Source:

Contents:

Archived?: Yes. pyyddd000_slot.fits

AUX Record Types:

DAT	0	spectral counts

B.2 High-Level Spectral Database Files

B.2.1 Daily Difference Spectra

File Name Format: Pyydddhhmm_xxx[A—B]#.SDB where xxx specifies the table (PRI/SEC/SOL/TRN) and # is a number used to prevent file name collisions.

Product Type: High-level

File Format: VAX/VMS, indexed, variable length records

Source: Each DDS file generated by BLD_QUAD_DIF contains the background subtracted Daily Difference Spectra for one OSSE target for a single day of observation are stored as R*4.

Contents: Daily difference - One SDR exists for each 2-minute analyzable unit.

Archived?: Yes. pyydddhhmm_xxx[a—b]#.fits

AUX Record Types:

AUDEF	34	Analyzable Unit Definition (1st SDR only)
DAT	0	background-subtracted spectral counts
ERR	1	errors for background-subtracted spectral counts
AUID	35	Analyzable Unit IDentifier record

B.2.2 Bottom-Line Spectral File

File Name Format: *_BLN.SDB

Product Type: High-level

File Format: VAX/VMS, indexed, variable length records

Source: Generated by BOTTOMLINE.

Contents: A single SDR exists for each view period. Basically a sum of all Spectral Data Records(SDR), for all days and all detectors. Rebinned and includes a detector response matrix.

Archived?: Yes. *_bln.fits

AUX Record Types:

DAT	0	background-subtracted spectral counts
ERR	1	errors for background-subtracted spectral counts
EDG	5	low edges of energy channels (MeV)
WID	9	widths of energy channels (MeV)
NOTE	30	Information from SDR_SUM and SDR_PUT_MTRX.
CRC	37	Compound Response Coefficients, created by DAY_SUMMER.
COV	45	
RMTRX	38	NCHAN x NPHOT (The order of these dimensions should be capatible with XSPEC?)
RMDEF	43	
RMVEC	44	contains 4 vectors of size = Max(NCHAN,NPHOT)

B.2.3 Good-Summed Count Spectra

File Name Format: *_GSM.SDB

Product Type: High-level

File Format: VAX/VMS, indexed, variable length records

Source: Generated by DAY_SUMMMER.

Contents: Statistical correlations caused by shared backgrounds in pointing strategy have been removed, making it a better result than the regular Day-Summed DSM file.

Archived?: Yes. *_gsm.fits

AUX Record Types:

DAT	0	background-subtracted spectral counts
ERR	1	errors for background-subtracted spectral counts
EDG	5	low edges of energy channels (MeV)
WID	9	widths of energy channels (MeV)
NOTE	30	Information from SDR_SUM and SDR_PUT_MTRX.
CRC	37	Compound Response Coefficients, created by DAY_SUMMER.
COV	45	

B.2.4 Totalled Spectral File (*_TOT.SDB)

File Name Format: *_TOT.SDB

Product Type: High-level

File Format: VAX/VMS, indexed, variable length records

Source: Total count spectra, with response matrix, generated by FIT_PREP.

Contents: An SDR exists for each detector, totalled for the viewing period. A full spectral resolution response matrix is generated for each detector.

Archived?: yes. *_tot.fits.

AUX Record Types:

DAT	0	background-subtracted spectral counts(448 channels)
ERR	1	errors for background-subtracted spectral counts
EDG	5	low edges of energy channels (MeV)
WID	9	widths of energy channels (MeV)
RELSPEC	28	
NOTE	30	Information from SDR_SUM and SDR_PUT_MTRX.
CRC	37	Compound Response Coefficients, created by DAY_SUMMER.
RMTRX	38	Response matrix, created by FIT_PREP (NCHAN x NPHOT) (Nominally 448 x 470)
RMDEF	43	
RMVEC	44	Contains 4 vectors of the size Max(NCHAN,NPHOT)
COV	45	Nominally contains 1 SDB per detector per pointing offset(DSA)

B.2.5 AUX-type Summary

This table maps AUX Record Types for the SDB files delivered to the COSSC.

AUX TYPES BY SDB FILE

SPM	BURST	SLOT	SEC	C60	C60SM	DDS	TOT	BLN	GSM
0	0	0	0	0	0	0	0	0	0
-	-	-	-	-	-	1	1	1	1
-	-	-	-	-	-	-	5	5	5
-	-	-	-	-	-	-	9	9	9
-	19	-	-	-	-	-	-	-	-
25	-	-	-	-	25	-	-	-	-
-	-	-	-	-	28	-	-	-	-
29	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	28	-	-
-	-	-	-	-	-	-	30	30	30
-	-	-	-	-	-	34	-	-	-
-	-	-	-	-	-	35	-	-	-
-	-	-	-	-	-	-	37	37	37
-	-	-	-	-	-	-	38	38	-
-	42	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	43	43	-
-	-	-	-	-	-	-	44	44	-
-	-	-	-	-	-	-	45	-	45

B.3 Pulsar Data Files (PDF)

File Name Format: Pyyddd0000.PDF

Product Type: Low-level

File Format:

Source:

Contents:

Archived?: No.

B.4 Other File Types (not being archived)

B.4.1 Telemetry Scalar File

File Name Format: Pyyddd0000.TSF

Product Type: Low-level

File Format:

Source:

Contents: Count spectra at maximum time resolution with some summing of energy channels. Not stored in SDB format. Includes the shield rates at high time resolution.

Archived?: No.

Even more files not yet documented:

Pyyddd0000.TLM: Raw telemetry data

Pyyddd0000.DQ: Data quality flags. Not stored in a human-readable form.

Pyyddd0000_DAILY.CPE: Charged particle event rate integrated over an orbit. Not stored in a human-readable form.

Pyyddd0000.EVNT: Event lists

Pyyddd0000.HVGL: High-voltage glitches.

Pyyddd0000.EVNTC6: Cobalt 60 event lists

Pyyddd0000_MSCRN.DAT: Generated by the manual screening operations.

Pyyddd0000_COS.DAT:

PLANNING_CATALOG.DAT: Target list

PERMANENT.DQ Listing of maneuver start & stop times.

CMDSTATE_TABLE.DAT

Instrument Command Files

These files (OMP*.CMD) contain scripts that call the instrument command files (*.IDL) files listed below.

- OMP95350A.CMD
- OMP95354A.CMD
- OMP96002A.CMD
- AGCNOM.IDL
- DTW9115101.IDL
- CAL9136101.IDL0
- CO609227502.IDL
- CO609227501.IDL
- CHNSM_TBL_X2.NML: TSF channel definitions (x2 gain).
- CHNSM_TBL_NOM.NML: TSF channel definitions (nominal gain).
- CMT9519101.IDL

- MDC9500301.IDL
- BURST9503001.IDL
- MSR9506101.IDL
- CMT9527001.IDL
- SLW9534201.IDL
- RP19322901.IDL, RP19320801.IDL, RP19321501.IDL, RP29332301.IDL
- FLTPATCHODS.IDL
- FLTPATCHMDC.IDL
- FLTPATCHNOMDC.IDL
- D1CFG9430701.IDL, D2CFG9430701.IDL, D3CFG9430701.IDL, D4CFG9430701.IDL
- PSR9534601.IDL, PSR9533202.IDL, PSR9535401.IDL, PSR9600203.IDL
- PST9534601.IDL, PST9201001.IDL, PST9600201.IDL, PST9224801.IDL
- PRI9534801.IDL, PRI9535401.IDL, PRI9600202.IDL
- CODEPSTPSR9534801.IDL
- CODEPSTPSR9535401.IDL
- TRAN9534802.IDL
- SEC9535001.IDL
- SEC9600201.IDL
- SEC9535401.IDL

Appendix C

High-Level OSSE Spectral Data Products

This appendix describes the new products being delivered.

C.1 Delivered Products Listing

This section lists the products that are being delivered.

The tapes delivered by NRL conataining the new High-Level data products are created by VMS BACKUP command. Each backup directory contains files for one viewing period target and are named in this format:

[HILEV.VP.TARGNAME]

where VP = viewing period number TARGNAME = OSSE target name

All filenames (except the Daily Difference SDB files) will begin with:

<TARGNAME>_<VP>

The filenames for the Daily Difference SDB files are named in this format:

PYYDDDHHMM_PRI[$A|B$]#.SDB for primary target #
PYYDDDHHMM_SEC[$A|B$]#.SDB for secondary target #,etc.

where

YY	=	observation year
DDD	=	day-of-year of observation
HHMM	=	time of start of observation
$[A\|B]\#$	=	a filename modifier used to prevent the possibility of similarly named files on days when the VP changes (2 primary sources for the same day)

* See the Appendix for example contents from the first delivery.

Each Directory will contain the files:

Product Type	Filename Root	Number of files per target directory
Summed DDS Data	_BLN.SDB	1
	_GSM.SDB	1
	_TOT.SDB	1
Graphics Output	_BLN.PS	1
	_COUNTS.PS	1
	_DAILY.PS	1
	_EBAND.PS	1
	_PREP.PS	1
	_STATE.PS	1
Input Parameters	.CFG	1
Flux Table	_PHOT.DAT	1
Log files	.LOG	1
	_BLN.LOG	1
	_PLOTS.LOG	1
	_PREP.LOG	1
Daily Difference SDB	.SDB	28
	TOTAL	53 files per target

Normal Viewing Periods may contain up to 6 targets X 53 = 318 files

C.1.1 Delivered Products Description

This section describes the new HIGH-LEVEL OSSE Products being delivered. The SPECANAL(IGORE v7.4) routines used to generate each product are noted. AUX Record Types for the SDB files are described.

Corresponding AUX Type Numbers are given in parentheses.

Daily Difference Spectra(DDS)

Each DDS file generated by BLD_QUAD_DIF contains the background subtracted Daily Difference Spectra for one OSSE target for a single day of observation.

The new AUX types exist for DDS-SDB files and are briefly described here.

PYYDDDHHMM_xxx[A—B #.SDB]Daily difference - One SDR exists for each 2-minute analyzable unit.

_BLN.SDB 'Bottom-line' count spectra. Basically a sum of all Spectral Data Records(SDR), for all days and all detectors. Generated by BOTTOMLINE. An SDR exists for each detector totalled by day.

_GSM.SDB Good-summed count spectra. Statistical correlations caused by shared backgrounds in pointing strategy have been removed, making it a better result than the regular Day-Summed DSM file. Generated by DAY_SUMMMER

_TOT.SDB Total count spectra, with response matrix, generated by FIT_PREP. An SDR exists for each detector, totalled for the viewing period.

Graphical products

Graphical products consist of 6 SPECANAL(IGORE) postscript outputs:

_STATE.PS Generated by PLOTANAL SHOW_WHEN plot. Shows the pointing strategy used by each detector for each state of the instrument. Good for recognizing possible confusing sources in both source & background pointings.

_EBAND.PS Generated by PLOTANAL TEMPORAL plot (i.e. a light curve). Plots are generated for 4 energy bands: 0.05-0.15

MeV, 0.15-0.30 MeV, 0.30-0.80 MeV, 0.80-1.50 MeV, showing the daily-summed source intensity for each day of the observation. Curves are plotted for each detector and their sum. For the summed plots, the mean is computed for the observation and used as the model for the fit. P is the probability that the (constant) intensity model is consistent with the data points. The final table compares the data from one detector with the model produced by the average of the other detectors.

_BLN.PS Generated by BOTTOMLINE plot. 'Bottom line' Count rate and flux. Generates resampled plots in both count and photon space.

_PREP.PS Generated by FIT_PREP. These are difference count spectra plotted for each detector on the source. The are plots linear in energy from zero to 4 MeV and logarithmically for 0.01 to 10 MeV. These plots are also generated with rebinning.

_COUNTS.PS Generated by PLOTANAL. PLOTANAL count rate plot. The count plots are generated by individual detectors and summed detectors. Each of these plots is run over two energy ranges (linear & logarithmic) and two different energy-binnings. The DSA in the graph title represents the Detector Step Angle corresponding to an on-source pointing.

_DAILY.PS Generated by PLOTANAL. Daily count rate plots consisting of two major components: daily plots showing each detector (and two energy ranges for each detector), and each detector summed for the duration of the observation. Each data set is fit to a constant model of zero, the chi-square representing the probability that the plot is consistent with the zero model (probability of a non-detection).

SPECANAL Logs

.LOG	from Entire batch job
_BLN.LOG	from BOTTOMLINE
_PLOTS.LOG	from PLOTANAL
_PREP.LOG	from FIT_PREP

Miscellaneous products

_PHOT.DAT - SPECANAL Flat file listing 'approximate' photon flux points. Produced using simple powerlaw model $(1 \times 10^{-3}(E/0.1MeV)^{-2.0})$. Useful for plotting the output using your own software. Generated from call to BOTTOMLINE.

_.CFG - Complete configuration of the run including ANAL, BQD, DSM, and FITPREP namelists.

Appendix D

OSSE Pipeline Configuration & Software

D.1 OSSEPIPE/ARCHIVER vs OSSETEST/OSSE

The OSSETEST (VAX) and osse (Unix) accounts are the testing areas for the OSSEPIPE (VAX) and archiver (unix) accounts. The test area should contain all source code, test binary images, and support files as contained in the corresponding live account. There may be additional files present in the test accounts.

D.2 Logicals & Symbols (VMS)

Of primary concern are the ones defined in the VMS LOGIN.COM file. These are set when you log into the corresponding account. The program is set-up to establish the appropriate symbols depending on the account name.

osse_destroot location for FITSified output products (this should be NFS mounted with Unix side)

osse_stage_www location for HTML output products (this should be NFS mounted with Unix side)

osse_archroot Unix root path for archive (WARNING! If this value is changed, be sure to change in in archive_osse.pro on the Unix side of this pipeline!)

osse_native VAX staging area for low-level deliveries

osse_hlpnative VAX staging area for high-level deliveries

osse_batchname Name of VAX batch que used for processing

osse_homedisk location of executables & .com files

osse_logdisk Location of log files

osse_tape Tape drive for loading archive

osse_logdisk location of processing log files

osse_highlevel location of high-level processing software

osse_lowlevel location of low-level processing software

osse_catsave location to save delivery planning catalogs (careful not to confuse with osse_cat)

D.3 Unix Paths, etc.

Here are the control variables defined for the archiver account and defined in archive_osse.pro and build_new_tdat.pro. The paths below are for the live archiver account. A parallel set exists for the osse account.

stageroot = /h/grossc/disk20/ossepipe/stage/ data staging area (NFS-mounted with VMS)

archiveroot = /FTP/compton/data/osse/ root level of archive

tdatpath = /home/cosmic/archiver/osse/osse_tdat/ output location for delimited tdat file

logpath = /home/cosmic/archiver/osse/osse_logs/ location for log files

w3b_stage = hrcdba@dbsrv:/dba_dbase/work/osse/hdb/ W3Browse staging area

w3b_email = browse_dba@lheamail.gsfc.nasa.gov E-mail notification

Appendix E

osse*.tdat: The Pipeline Interface to W3Browse

E.1 Intent

The TDAT file creates an observation identifier (Obs ID) for each OSSE observation of a given target, along with associated information which a researcher might use as a basis for a search. The observation ID provides an additional link to the configuration database which is described in the next section.

The data lines have been wrapped to fit on the page

```
<HEADER>
#       TABLE: heasarc_osse
#    LOCATION:
#  TOTAL ROWS:         1580
#
# CREATION DATE: Tue Sep  8 11:14:38 1998
# PROGRAM:       archive_osse.pro
# VERSION:       v1.1 - August 1998
#
table_name = heasarc_osse
table_description = "CGRO/OSSE Observations"
table_security = "public"
table_document_url = http://heasarc.gsfc.nasa.gov/W3Browse/all/osse.html
```

```
#
# Table Parameters
#
field[target]      = char30         (index) // Target name
field[ra]          = float8_degree  (index) // Right Ascension
field[dec]         = float8_degree  (index) // Declination
field[lii]         = float8_degree  (index) // Galactic longitude
field[bii]         = float8_degree  (index) // Galactic Latitude
field[time]        = float8_mjd     (index) // Start time of observation
field[endtime]     = float8_mjd     (index) // End time of observation
field[exposure]    = float8         (index) // Exposure time
field[instrument]  = char1          (index) // Initial of instrument used
field[obs_id]      = char25         (index) // Observation ID
field[radius]      = float8         (index) // Radius Size or error
field[view_period] = float8         (index) // CGRO viewing period
field[altvp]       = char6          (index) // Alternate view period format
#
parameter_defaults = obs_id ra dec time endtime
#
# Virtual Parameters
#
declination           = @dec
default_search_radius = 480
end_time              = @endtime
equinox               = 2000
frequency_regime      = Gamma ray
observatory_name      = CGRO
right_ascension       = @ra
sequence_id           = @obs_id
start_time            = @time
target_name           = @target
unique_key            = obs_id
#
# Data Format Specification
#
line[1] = view_period altvp instrument target obs_id time endtime exposure ra dec lii bii radius
<DATA>
```

```
1.00000|0001_0|0|1A 0535+262|0.VP0001_0.1A0535+262|48392.504|48407.504|-1.0|84.7300|26.3200|
181.440|-2.64000|240.000
1.00000|0001_0|0|VELA PULSAR|0.VP0001_0.VELAPULSAR|48392.504|48393.504|-1.0|128.920|-45.1800|
263.590|-2.74000|240.000
1.00000|0001_0|0|CRAB PULSAR|0.VP0001_0.CRABPULSAR|48392.504|48407.504|-1.0|83.5200|22.0200|
184.500|-5.87000|240.000
1.00000|0001_0|0|PSR 1957+20|0.VP0001_0.PSR1957+20|48392.504|48407.504|-1.0|299.850|20.8000|
59.1700|-4.66000|240.000
1.00000|0001_0|0|BLANK PER 1|0.VP0001_0.BLANKPER1|48399.504|48407.504|-1.0|345.240|48.2600|
104.630|-10.6500|240.000

<END>
```

Appendix F

OSSE-zzdp.config: The Archive Link to W3Browse

F.0.1 Intent

This configuration file performs two functions.

- The TAG entries generate labels for each file in the archive which matches the template defined.

- The SET entries define logical groupings for the tags. Each SET defines a data group in W3Browse and *MUST* contain a link to the observation ID.

This file is generated periodically by systems in the HEASARC. At the time of this writing (October 1998), it is usually generated over the weekend and installed online sometime Monday. Generally, by Tuesday of a given week

F.0.2 Contents

The working copy of this file is located in dbsrv:/bridgman/cgro-osse.config. The online copy is located at dbsrv:/heasarc/hrcdba/ingest/zzdp/missions/cgro-osse.config

Formatting has been modified to fit on these pages.

```
################################################################################
# Modifications:
#    July 8, 1998, WTB: added targspec, ntargspec, & support data set defs
#    September 25, 1998, WTB: Revised high-level data product organization
#
#
# Directory prefixes for URLs and local file system access:
#
url_prefix = ftp://legacy.gsfc.nasa.gov
dir_prefix = /FTP
#
################################################################################
#
# Directory shortcuts:
#
${cgro} = ${prefix}/compton/data
${osse} = ${cgro}//osse
#
################################################################################
#
# CGRO OSSE data products sets:
#
set[heasarc_osse(support)] = cgro.o.vp@{altvp}.lo.log,
cgro.o.vp@{altvp}.lo.screen,
cgro.o.vp@{altvp}.lo.cat,
cgro.o.vp@{altvp}.lo.cmd
// OSSE Low-Level Support Files
set[heasarc_osse(targspec)] = cgro.o.vp@{altvp}.lo.spm.*
// OSSE Low-Level Targeted Spectra
set[heasarc_osse(calib)] = cgro.o.vp@{altvp}.lo.c60*
// OSSE Calibration Data
set[heasarc_osse(telem)] = cgro.o.vp@{altvp}.lo.tsf.*
// OSSE Telemetry Scalar Data
set[heasarc_osse(ntargspec)]= cgro.o.vp@{altvp}.lo.burst.*,
cgro.o.vp@{altvp}.lo.slot.*,
cgro.o.vp@{altvp}.lo.sec.*
// OSSE Low-Level Non-targeted Spectra
```

```
#
set[heasarc_osse(lotemp)] = cgro.o.vp@{altvp}.lo.pdf.*
// OSSE Low-Level Temporal Data
#
set[heasarc_osse(hitotal)] = cgro.@{obs_id}.hi.tot
// OSSE High-Level Totaled Spectrum
set[heasarc_osse(hisummary)]= cgro.@{obs_id}.hi.bln
// OSSE High-Level Summary Spectrum
set[heasarc_osse(hidaily)] = cgro.@{obs_id}.hi.gsm
// OSSE High-Level Daily Spectra
set[heasarc_osse(hi2minute)] = cgro.@{obs_id}.d.*
// OSSE 2-Minute Difference Spectra
set[heasarc_osse(hixspec)] = cgro.@{obs_id}.hi.xsp
// OSSE XSpec Files
set[heasarc_osse(hiquicklook)] = cgro.@{obs_id}.hi.qkl
// OSSE High-Level Quicklook Products
set[heasarc_osse(hiprocess)] =  cgro.@{obs_id}.hi.pro
// OSSE High-Level Processing Products
#
# CGRO OSSE data products tags:
#
#  Low-level spectral data:
#
tag[cgro.o.vp{%2}.lo.spm.{%3}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/p(.+)0000_spm\.fits\.gz
// OSSE Low-Level Spectra (FITS)
tag[cgro.o.vp{%2}.lo.burst.{%3}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/p(.+)0000_burst\.fits\.gz
// OSSE Low-Level Shield File (FITS)
tag[cgro.o.vp{%2}.lo.log] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/processinglogs(.+)\.tar\.gz
// OSSE Low-Level Processing Log (TAR)
tag[cgro.o.vp{%2}.lo.screen] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/screen_(.+)\.bck\.gz
// OSSE Screening File (BACKUP)
tag[cgro.o.vp{%2}.lo.cat] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/plancat_(.+)\.dat\.gz
```

```
// OSSE Catalog (ASCII)
tag[cgro.o.vp{%2}.lo.cmd] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/cmdstate_(.+)\.tar\.gz
// OSSE Command State (TAR)
tag[cgro.o.vp{%2}.lo.slot.{%3}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/p(.+)0000_slot\.fits\.gz
// OSSE Low-Level Spectra (FITS)
tag[cgro.o.vp{%2}.lo.sec.{%3}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/p(.+)0000_sec\.fits\.gz
// OSSE Low-Level Slew Spectra (FITS)
#
#  Calibration file sets:
#
tag[cgro.o.vp{%2}.lo.c60.{%3}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/p(.+)0000_c60\.fits\.gz
// OSSE Low-Level Calibration File (FITS)
tag[cgro.o.vp{%2}.lo.c60sm.{%3}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/p(.+)0000_c60sm\.fits\.gz
// OSSE Low-Level Calibration Summary (FITS)
#
#  Telemetry:
#
tag[cgro.o.vp{%2}.lo.tsf.{%3}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/p(.+)0000\.tsf\.gz
// OSSE Telemetry Scalar File (BINARY)
#
#  Low-level timing data:
#
tag[cgro.o.vp{%2}.lo.pdf.{%3}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/lolevspec/p(.+)0000_pdf\.fits\.gz
// OSSE Low-Level Pulsar File (FITS)
#
#  High-level spectral data:
#
tag[cgro.o.vp{%2}.{%3}.hi.tot] =
${osse}/phase([0-9]+)/vp([0-9_]+)/hilevspec/(.+)/(.+)_tot\.fits\.gz
// OSSE High-Level Spectra (FITS)
```

```
tag[cgro.o.vp{%2}.{%3}.hi.gsm] =
${osse}/phase([0-9]+)/vp([0-9_]+)/hilevspec/(.+)/(.+)_gsm\.fits\.gz
// OSSE High-Level Spectra (FITS)
tag[cgro.o.vp{%2}.{%3}.hi.bln] =
${osse}/phase([0-9]+)/vp([0-9_]+)/hilevspec/(.+)/(.+)_bln\.fits\.gz
// OSSE High-Level Spectra (FITS)
tag[cgro.o.vp{%2}.{%3}.hi.xsp] =
${osse}/phase([0-9]+)/vp([0-9_]+)/hilevspec/(.+)/(.+)_xspec\.tar\.gz
// OSSE High-Level XSpec Files (TAR)
tag[cgro.o.vp{%2}.{%3}.d.{%4}] =
${osse}/phase([0-9]+)/vp([0-9_]+)/hilevspec/(.+)/p([0-9A-F]+)0000_(.+)\.fits\.gz
// OSSE Daily-Difference Spectral File (FITS)
tag[cgro.o.vp{%2}.{%3}.hi.qkl] =
${osse}/phase([0-9]+)/vp([0-9_]+)/hilevspec/(.+)/(.+)_(.+)_quicklook\.tar\.gz
// OSSE High-Level Quicklook Products (TAR)
tag[cgro.o.vp{%2}.{%3}.hi.pro] =
${osse}/phase([0-9]+)/vp([0-9_]+)/hilevspec/(.+)/(.+)_(.+)_process\.tar\.gz
// OSSE High-Level Processing Products (TAR)
```

Appendix G

FITSification of OSSE Data Files

SDB2FITS & FITS2SDB converters: Need to be operational on low and high- level SDB files.

PDF2FITS & FITS2PDF are having problems since INGRES was removed from COSSC. Apparently the program TARGS.C collects some data from the archive through SQL calls and uses it for adding some keywords to the FITSified PDF files.

The basic converters utilize the ToFU and FITSIO libraries. The ToFU templates for the FITSification can be found in Section **??**.

G.1 SDB2FITS

The SDB2FITS program converts a single OSSE SDB file into a single FITS file. Each AUX type in the SDB file is converted into an individual binary table extension, i.e. all the individual AUX 00 spectral records will be converted into a single binary table.

Calling Sequence:
SDB2FITS [infile] [outfile] [view_period]

infile: Input SDB file name

outfile: Output FITS file name

view_period: CGRO viewing period number, with decimal place

Platform: VAX/VMS

Logicals Used: OSSE_CAT, OSSE_SDB_TEMPLATE

Data Files Used:

Subprograms/Libraries Called: ToFU, FITSIO

G.1.1 Program Outline

This converter acutally works in two parts. The first part is SDB2PREP which separates the individual AUX types into separate files. The PREP2FITS portion of the code looks for each of these temporary files and constructs a binary table extension for each of them.

1.

G.2 FITS2SDB

Calling Sequence:
FITS2SDB [**infile**] [**outfile**]

infile: Input FITS file

outfile: Output SDB file

Platform: VAX/VMS

Logicals Used:

Data Files Used:

Subprograms/Libraries Called: ToFU, FITSIO

G.2.1 Program Outline

1.

G.3 SDB_COMPARE

SDB_COMPARE compares an original OSSE SDB data file and a reconverted data file (original –¿ FITS –¿ reconverted) record by record and byte by byte to determine if the FITS conversion process was correctly performed.

Calling Sequence:

SDB_COMPARE [file1] [file2] [verbose]

file1: compare this SDB file

file2: to this SDB file

verbose:

Platform: VAX/VMS

Logicals Used: SDB_CONVERTERS

Data Files Used: SDB_CONVERTERS:SDB_BYTE_EXCEPTIONS.DAT

Subprograms/Libraries Called: ToFU

G.3.1 Program Outline

SDB_COMPARE compares the first record of each file and exits if not identical. It then grabs the next record from both the original and reconverted files and verfies that both have the same index record key (first 8 bytes of each record). If the index keys are not the same, then it's possible that the original file had a dummy header record inserted which the FITS and reconverted files do not contain; if this is the case then the dummy header record is ignored and the next record from the original file is read; if the original file record is not a dummy header record then SDB_COMPARE exits with an error message. If the index keys match then the record lengths are checked to make sure they are the same. If the record lengths match then the record contents are comparied byte by byte.

The only instance where the condition

```
byte(original\_data\_file) != byte(reconverted\_data\_file)
```

is allowed is when the bytes belong to a real*8 field. Due to differences in VAX and IEEE R*8 representations, the reconverted R*8 bytes can differ from the original R*8 bytes. When a byte mismatch is found in a R*8 field, SDB_COMPARE reconstructs the R*8 numbers and compares the relative error between original and reconverted values; the relative error must be within a specified tolerance for the mismatch to be acceptable. the file

```
SDB\_CONVERTERS:SDB\_BYTE\_EXCEPTIONS.DAT
```

contains a list of byte start positions for the R*8 values.

SDB_COMPARE writes error messages to a log file. It looks for a log file of the form ORIGINAL_DATA_FILE_NAME.log to append its output too, and creates such a file if it does not already exist.

If the VERBOSE flag is set to .TRUE. then SDB_COMPARE provides the user with extra status information, both to the screen and to the log file. This is useful for debugging.

1.

G.4 SDB_PROCESS

The FITS converters described in the previous sections are called in a batch process, SDB_PROCESS, which takes a delivery of raw SDB data files and converts them to FITS.

Platform: VAX/VMS

Logicals Used:

Data Files Used:

Subprograms/Libraries Called: SDB2FITS, FITS2SDB, SDB_COMPARE

G.4.1 Program Outline

1. Setup DCL logicals.
2. Build destination path.
3. Now loop over each SDB file.
   (a) Create the destination directory, if necessary.
   (b) Run SDB2FITS on the native file, creating the FITS file. Record an entry in the log file if it fails.
   (c) Run FITS2SDB on the FITS file creating RECONV.TMP. Record an entry in the log file if it fails.
   (d) Run SDB_COMPARE on the native file and RECONV.TMP. Record an entry in the log file if it fails.
   (e) If the above steps ran without errors, write an entry in the success log and delete the SDB file in the native directory.
4. Erase the zero-length log files in the input and processing area.
5. Mail the log files to the host account.

G.5 PDF2FITS

Platform: VAX/VMS

Logicals Used:

Data Files Used:

Subprograms/Libraries Called:

G.5.1 Program Outline

1.

G.6 TARGS

Platform: Ultrix

Logicals Used:

Data Files Used:

Subprograms/Libraries Called:

G.6.1 Program Outline

1.

G.7 FITS2PDF

Platform: VAX/VMS

Logicals Used:

Data Files Used:

Subprograms/Libraries Called:

G.7.1 Program Outline

1.

G.8 PDF_COMPARE

Platform: VAX/VMS

Logicals Used:

Data Files Used:

Subprograms/Libraries Called:

G.8.1 Program Outline

1.

G.9 PDF_PROCESS

Platform: VAX/VMS

Logicals Used:

Data Files Used:

Subprograms/Libraries Called: PDF2FITS, FITS2PDF, PDF_COMPARE, TARGS

G.9.1 Program Outline

1. Setup DCL logicals.
2. Build destination path.
3. Clean out work files from temporary area.
4. Now loop over each PDF file.
   (a) Create the destination directory, if necessary.
   (b) Run PDF2FITS on the native file, creating the FITS file. Record an entry in the log file if it fails.
   (c) Clean out temporary area.
   (d) Submit targs.c to Ultrix system to build additional FITS primary header info from the INGRES database. (INGRES is no longer supported so this call is a source of much grief.)
   (e) Run FITS2PDF on the FITS file creating DECONV.TMP. Record an entry in the log file if it fails.
   (f) Run PDF_COMPARE on the native file and DECONV.TMP. Record an entry in the log file if it fails.

(g) If the above steps ran without errors, write an entry in the success log and delete the SDB file in the native directory.

5. Clean out the temporary area.

6. Erase the zero-length log files in the input and processing area.

7. Mail the log files to the host account.

G.10 Basic Operations of the FITS Converters

Here are some basic notes on how the FITS converters operate which may be important if modifications are needed in the future. There are 7 basic classes of aux**_extract routines:

Class 0: datatype entities are array elements

Class 1: datatype entities are structures, number of data records is not necessially equal to the number of structures

2. datatype entities are structures, number of data records must be equal to the number of structures (obsolete)

4. datatype is undefined

5. datatype entities are array elments, elements may require I*4 to I*2 conversin (AUX 64 only)

6. datatype entities are structures, number of data records is not necessially equal to the number of structures, elements within the structures may require I*4 to I*2 conversion (AUX 65 only).

7. datatypes are classified as "special" and require non-standard processing (AUX 38 and 39)

Table G.1: Subroutine summary for OSSE FITS converters (Part 1)

AUX Type	Extension name	Entity type	SDB2FITS convert	FITS2SDB extract	Comments
00	SINGLE_SPECT_R	element(0)	custom	custom	(see note 1)
01	SGL_SPEC_STAT_+UNC	element(0)		auxclass0	
02	SGL_SPEC_STAT_-UNC	element(0)		auxclass0	
03	SGL_SPEC_SYS_+UNC	element(0)		auxclass0	
04	SGL_SPEC_SYS_-UNC	element(0)		auxclass0	
05	CHNL_LOW_EDGE_ENG	element(0)		auxclass0	
06	CHNL_LOW_EDGE_UNC	element(0)		auxclass0	
07	CHNL_ENERGIES	element(0)		auxclass0	
08	CHNL_ENERGIES_UNC	element(0)		auxclass0	
09	CHNL_WIDTHS	element(0)		auxclass0	
10	CHNL_WIDTHS_UNC	element(0)		auxclass0	
11	PSD_EFFCIENCY	element(0)		auxclass0	
12	PSD_EFFCIENCY_UNC	element(0)		auxclass0	
13	DET_RESOLUTION	element(0)		auxclass0	
14	DET_RESOLUTION_UNC	element(0)		auxclass0	
15		undefined(2)			
16		undefined(2)			
17		undefined(2)			
18		undefined(2)			
19	MULTI_SPECT_DSF	structure(1)		custom	
20	MULTI_SPECT_R	structure(1)	custom	custom	
21	M_SPEC_STAT_+UNC	structure(1)		custom	
22	M_SPEC_STAT_-UNC	structure(1)		custom	
23	M_SPEC_SYS_+UNC	structure(1)		custom	
24	M_SPEC_SYS_-UNC	structure(1)		custom	
25	CAL_LINE_INFO	structure(1)		auxclass1	

Note 1: Used in conjunction with AUX types 64 & 66(formerly) to handle R*4 and I*4 data types in spectra

Table G.2: Subroutine summary for OSSE FITS converters (Part 2)

AUX Type	Extension name	Entity type	SDB2FITS convert	FITS2SDB extract	Comments
26		undefined(2)			
27	PSD_MODEL_INFO	structure(1)		custom	
28	RELATED_SPECTRA	structure(1)		auxclass1	
29	ORBIT_HISTORY	structure(1)		auxclass1	
30	FOOTNOTE	structure(1)		auxclass1	
31		undefined(2)			
32	EST_BACKGROUND	element(0)		auxclass0	
33	EST_BACKGROUND_UNC	element(0)		auxclass0	
34	AU_DEFINITION	structure(1)		auxclass1	
35	AU_IDENT	structure(1)		custom	
36	SELECT_SPECT_LIST	structure(1)		custom	
37	COMPOUND_RESP_COEF	structure(1)		auxclass1	
38	RESP_MATRIX	special(3)	custom	custom	
39	INST_PP_RESP_VECT	special(3)	custom	custom	
40		undefined(2)			
41		undefined(2)			
42	SDB_DIRECTORY	structure(1)	custom	custom	
43	RESP_MATRIX_DEF	structure(1)		auxclass1	
44	RESP_MATRIX_VECTORS	special(3)	custom	custom	
45	COVAR	structure(1)		auxclass1	
46		undefined(2)			
47		undefined(2)			
48		undefined(2)			
49		undefined(2)			
50		undefined(2)			

Note 1: Used in conjunction with AUX types 64 & 66(formerly) to handle R*4 and I*4 data types in spectra

Table G.3: Subroutine summary for OSSE FITS converters (Part 3)

AUX Type	Extension name	Entity type	SDB2FITS convert	FITS2SDB extract	Comments
51		undefined(2)			
52		undefined(2)			
53		undefined(2)			
54		undefined(2)			
55		undefined(2)			
56		undefined(2)			
57		undefined(2)			
58		undefined(2)			
59		undefined(2)			
60		undefined(2)			
61		undefined(2)			
62		undefined(2)			
63		undefined(2)			
64	SINGLE_SPECT_J		custom		
65	MULTI_SPECT_J		custom		
66	SINGLE_SPECT_I		custom		

Note 1: Used in conjunction with AUX types 64 & 66(formerly) to handle R*4 and I*4 data types in spectra