MDI Guide
MDI/SOI Data Flow

Ver 0.54
14-Feb-95

M. Morrison
J. Aloise

This Copy Printed: February 14, 1995

MDI-????
SOI Tech Note#: 121

Some suggested reference books, material, and methods of finding more information:

- "MDI Guide: An Overview of the /mdisw System"
- TBD - what is useful to list here????
- The official MDI IDL software should have a detailed description of the routine in the header of the source file. It is possible to access this header from within IDL by using DOC_LIBRARY. A sample call is:

  IDL> **doc_library,'mdipict'**

If you have general comments, suggestions, or questions regarding this document or software, please send them to:

`morrison@quake.stanford.edu`

Contents

List of Figures

Preface

The conventions used in this manual are:

- Bold face for examples of what the user should type
- Program names are capitalized when mentioned in the text, but are not capitalized in the user input examples
- Variable names are italicized when mentioned in the text
- Dates and times are in Universal Time (UT) and have the convention "dd-mmm-yy"
- There are several different types of prompts that appear in the examples. The prompt IDL> is used for when the user has already entered and is running IDL. The prompt % is used to signify Unix commands, and the prompt $ signifies VMS commands. There are also FTP> prompts for 'File Transfer Protocol.'

For example, you would only type **.run show_data** for the following description. You would not type IDL>.

IDL> **.run show_data**

1 Introduction

This document describes the Stanford SOI-MDI Science Support Center (SSSC) for the MDI and spacecraft data coming from the SOHO spacecraft. There are different levels of processing which will be discussed. The definition of data sets and data series will be described, and parts of the pipeline execution (PE) and the Data Storage and Distribution System (DSDS) will be discussed.

1.1 Overview of SSSC

1.1.1 Production System

TBD

1.1.2 Analysis System

TBD

1.1.3 Processing Environment

The processing environment consists of four major component:

- Pipeline Scheduler (PS): Determines the data processing requirements and configures and executes Pipeline Execution tasks to perform them.
- Pipeline Execution (PE): Performs the data processing in cooperation with DSDS and Analysis Modules, as directed by PS.
- Data Storage and Distribution System (DSDS): Retrieves, stores, catalogs, and archives data products as requested by PE.
- Analysis Modules: The actual processing modules run by PE that transform input data to output data to create various data products.

Pipeline Scheduler (PS): The PS is the normal top level operator interface for data processing. As various tapes and CD-roms of MDI telemetry data become available, the PS is run to process them. The PS allocates and reserves the resources needed to process the input data. Processing occurs in various stages from the initial ingestion of the raw data into the DSDS through to the creation of advanced data products. This is accomplished by PS launching Pipeline Execution with various specification files that define the processing.

The PS has a checkpoint after each major stage of the processing and can be restarted at the last checkpoint in the event of any failure.

Pipeline Execution (PE): This is the process that manages a group of analysis modules that take a set of input data and transforms it to various output products. The analysis modules from a pipeline in that the output of one module is available as input to the next module on the pipeline. Pe sequences the data through the modules.

PE provides input data location, output data storage and output data cataloging and archiving on behalf of the analysis modules via PE's interface with DSDS.

The processing that pe is to do is determined by a plain text specification file that defines the analysis modules to run, the sequence to run them and the input and output data products required, as well as any storage and archiving requirements.

Data Storage and Distribution Subsystem (DSDS): The DSDS is a central Oracle database and related servers. It's primary functions are data storage, data retrieval and data export.

All data that are produced by analysis modules are input and output in DSDS managed storage. The managed storage consists of a disk farm and DD2 tape storage. Data migrates to and from disk and tape according to demand and age of data.

The analysis modules request data from the database via PE via abstract dataset names. The data will be moved from tape to disk if necessary, and the module will be returned the location of the data on the DSDS managed disks.

Tools are provided to query the database, select data of interest and export this data to local storage or make it available on 8mm tape.

Analysis Modules: These contain the algorithms that trasform the input to output data products. An analysis module has a definite structure that encapsulates and stadardizes system-dependent features such as I/O and communications between programs.

1.1.4 Configuration Management (CM)

The CM is based on BCS (A Baseline Configuration System) and RCS (Revision Control System). BCS is a set of utilities for maintaining a single baseline and multiple staging areas for a software development effort. BCS provides configuration management functionality as well as the means for multiple users to work concurrently on a common source tree with mininal conflict. RCS is a widely used version control system.

1.2 Estimates of the Amount of Data

The following table shows estimates of the amount of data. The increase from raw to level-0 is a factor of 2-3 because the telemetry compression is removed, and the data is often embedded back into a rectangular array. The data is still INTEGER*2 at level-0. There is potentially a factor of two increase at level-1 and above if the data is converted to REAL*4.

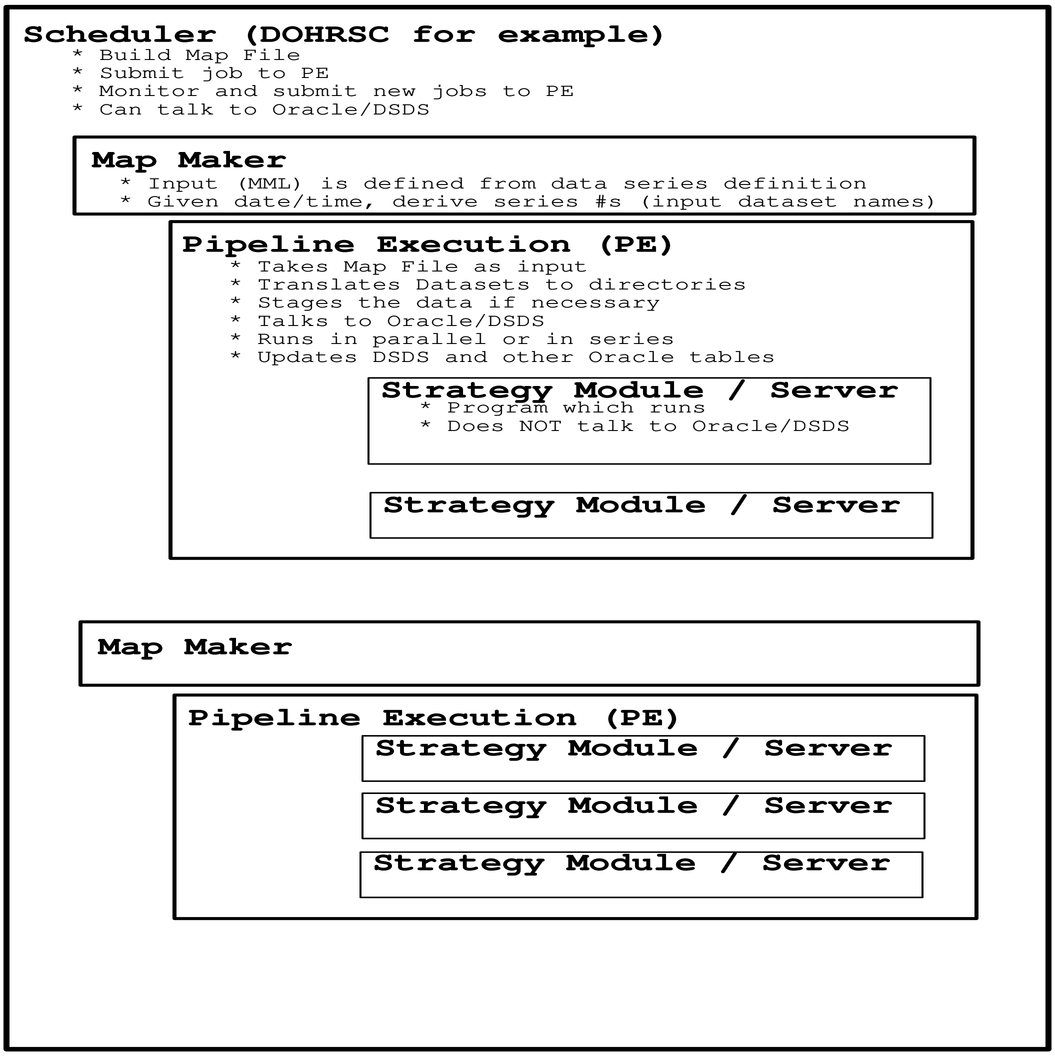

Figure 1: Overview of scheduler, map maker, pipeline execution (PE) and strategy modules

Amount of Data per Day (in Megabytes)						
Type		*Raw*	*Merged*	*Level-0*	*Level-1*	*Level-1.5*
Housekeeping	MDI	4	-	-	-	-
	SC1,2,4	3	-	-	-	-
SCI5K		55	-	150	300	-
SCI160K	Campaign	1728	1728	5200	10200	10200
	Dynamics	576	576	1800	3600	3600
Ancillary	S/C Pos & Motion	??	-	-	-	-
	??	??	-	-	-	-

The high rate data is 160 kbits/sec == 72 MBytes/hour. MDI is in Campaign mode 10 months a year, and in Dynamics mode the other 2 months a year.

1.3 Overview of Data Flow

Figure 3 shows an overview of the SCI160K data processing, which is very similar to the data flow for all of the different types of data. An overview of the levels of processing:

- **Raw** is the raw telemetry stream as received on the ground.
- **Merged** is merging and sorting through the raw telemetry files and creating files which are blocked (typically per hour).
- **Level-0** consists of:
  - Decompress the telemetry stream where necessary
  - Assemble a data product
  - Embed cropped images into a rectangular data array where necessary
- **Level-1** consists of:
  - Remove the instrument effects (flat fielding, ...)
  - Apply lookup table corrections
  - Convert the data numbers (DN) send down in the telemetry into calibrated engineering units.
- **Level-1.5** consists of:
  - Optionally make the data evenly spaced
  - Gather a group of input datasets into a single dataset
  - Gather information from ancillary databases
  - Allow the input to level 2 modules to be simple and well understood
- **Level-2** consists of analysis programs and modules.

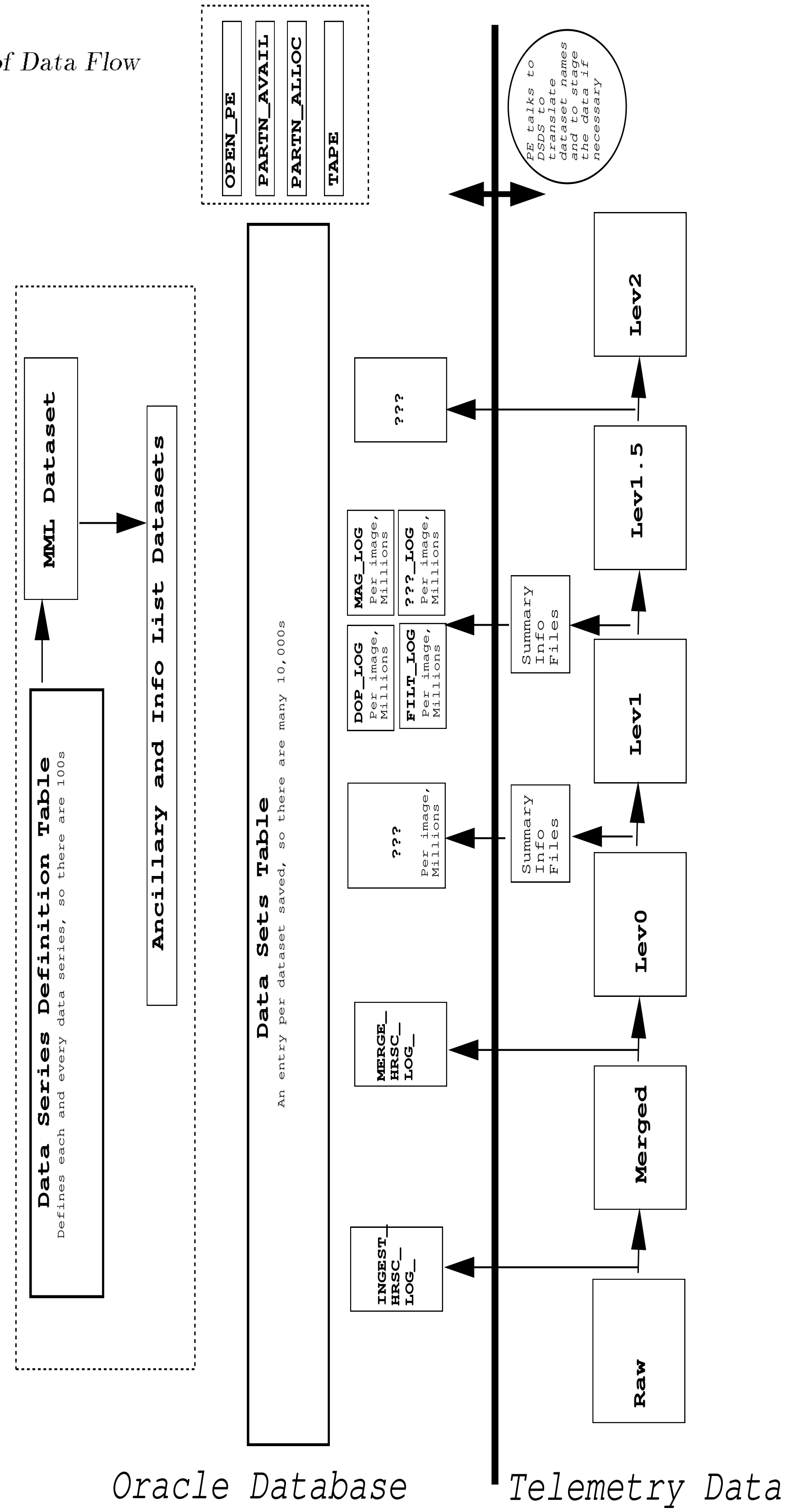

Figure 2: Overview of relation between Oracle and datasets

The processing will be *data driven (pushed)* up through the creation of level 1 data, and probably to level 1.5 for certain data products. From level 2 and onward, the data is *processing driven (pulled)*

1.4 Overview of the Dataset Naming Convention

A single dataset can contain many files and the files can be of different formats. The general rule is that all files for a given dataset are contained in a single directory.

The SSSC dataset naming consists of six parts. Only three parts are required to specify a data series: *prog* specifies the project, *level* generally specifies the amount of processing that has been performed, and *series* specifies which data product the data series refers to. Figure 4 show how the datasets are grouped according to prog, level and series names. Figure 5 shows how a dataset is just a grouping of images for a given block in the continuous series of images available for the mission.

Six Parts of a Data Set Name	
Label	*Description*
prog	Project name (highest level of grouping)
level	Level of processing
series	The data product
series #	Specifies which dataset in the data series
level #	The version of the dataset
prog #	(reserved)

It is important to understand the difference of a *data series* and a *dataset*. In general, the data series refers to the whole mission of data, and the dataset specifies a given block within that series. A dataset are all of the images (of a certain type) for a given block (usually the block is a block of time). The data series refers to all of the datasets for the whole mission.

A data series requires *prog*, *level* and *series* to be specified. A dataset requires those three parts plus the *series number*. To specify the dataset will complete accuracy requires six parts, the four just described plus the *prog number* and the *level number*.

1.4.1 *prog* Portion of a Data Series Name

The *prog* portion of the data series name allow for the grouping of data series into projects. Some of the different *prog* values which are available are shown in the following table.

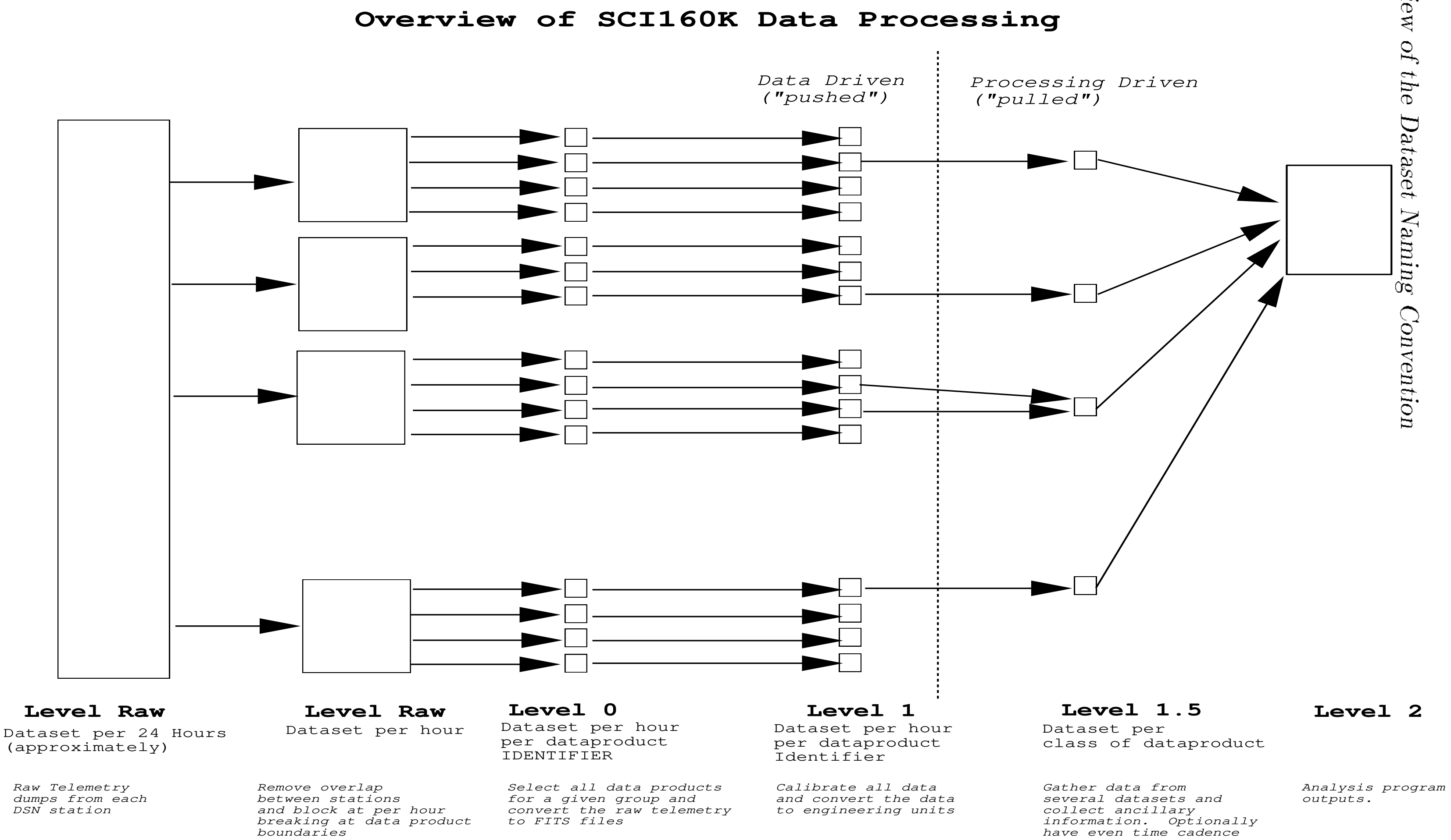

Figure 3: Overview of SCI160K Data Processing

Possible Values for Prog	
Prog Name	*Description*
mdi	MDI Flight data post-launch
mdi_eof	MDI Flight data post-launch, quick look recieved at EOF
soho	SOHO spacecraft data
gong	GONG data
south_pole	South Pole data
mdi_mml	MDI Map Maker Language files (DSDS files)
mdi_ds_in	Datasets specifying list of acceptable input datasets (DSDS files)
mdi_klst	Datasets specifying list of keywords (DSDS files)
maps	Map files prior to execution
mdi_ground	Calibration data, pre-launch received by EGSE
mdi_gsfc	Test data routed through PACOR
mdi_gsfc_egse	Test data, received by EGSE at GSFC/EOF
mdi_sim	MDI simulator data

Details on the Possible Sources and Routing of MDI Data				
Proj Name	*Instrument*	*Source*	*Path*	*Phase*
mdi_ground	flight	EGSE	Instim/CCS	pre-launch
mdi_gsfc	flight	PACOR	CCS	pre-launch
mdi_gsfc_egse	flight	EGSE	CCS	pre-launch
mdi	flight	PACOR	DSN/Tapes	post-launch
mdi_eof	flight	EGSE	PACOR/ECS	post-launch
mdi_sim	simulator	EGSE	Instim	pre and post launch

1.4.2 *level* Portion of a Data Series Name

The *level* portion of the data series names indicates the amount of processing that has been performed on a given dataset.

Possible Values for Level	
Value	*Description*
raw	Raw telemetry
merge	Raw telemetry merged and sorted
lev0	Level-0 processing (FITS or CDF images/tables)
lev1	Level-1 processing (Calibrated data)
lev1.5	"Gathered" and sorted calibrated data
lev2	Analysis results data
calib	Calibration data
ancil	Ancillary data
archive	Pre-launch archive data

1.4.3 *series* Portion of a Data Series Name

The series name is the final piece to specify a data series. In order to specify a **dataset**, the series number of the data series must be specified.

1.4.4 Sample Data Series/Dataset Naming

For example, the SCI160K data will be grouped by data product code (DPC) at level 0, and the blocking will normally be 24 hours. So the data series would be: `prog:mdi, level:lev0, series:ffffff03_01h`. Since the datasets are blocked per hour, the series number is the hour of the mission. Assuming the launch is 1-Nov-95, then the dataset for 25-Dec-95 04:00 to 05:00 would be `prog:mdi, level:lev0, series:ffffff03_01h[1300]`.

It is possible to specify a range of dataset and to also provide a list of datasets within the data series. Some examples are: `prog:mdi, level:lev0, series:ffffff03_01h[1300-1305]` to specify 6 hours of data, or `prog:mdi, level:lev0, series:ffffff03_01h[1300,1324,1348]` to specify 3 one hour datasets separated by 24 hours each.

A list of some series names is shown in section 2.4 on page 19

1.5 File Types, Formats, Organization and Names

There are several different types of files which will be present in a given dataset. To be a **conforming** dataset, the dataset must have an .overview file and a .record file, along with the data file. The data can be saved as individual FITS files, a single FITS file with all images in a 3-D cube, or in a CDF file.

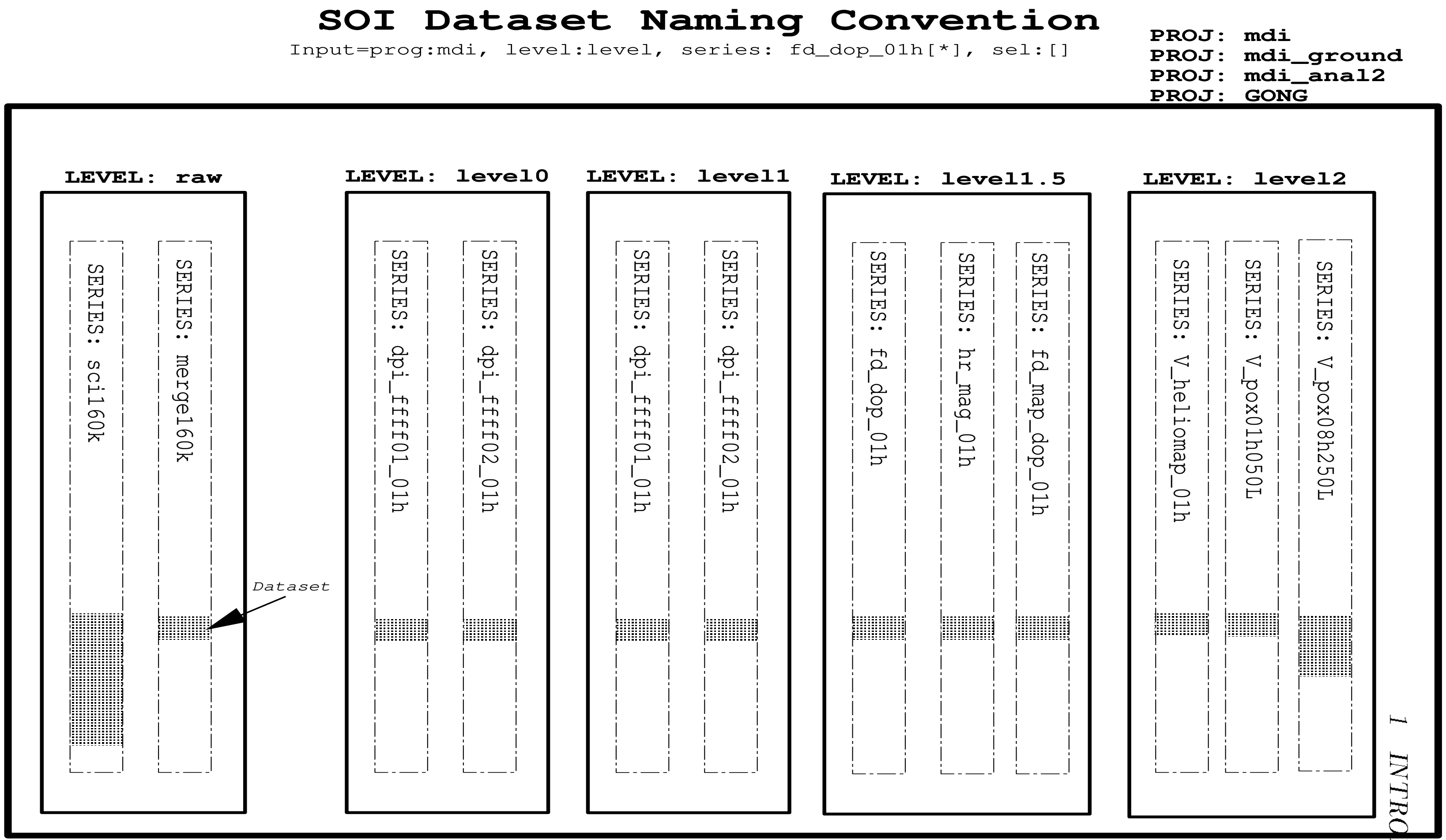

Figure 4: Dataset Naming Convention (PROG and LEVEL)

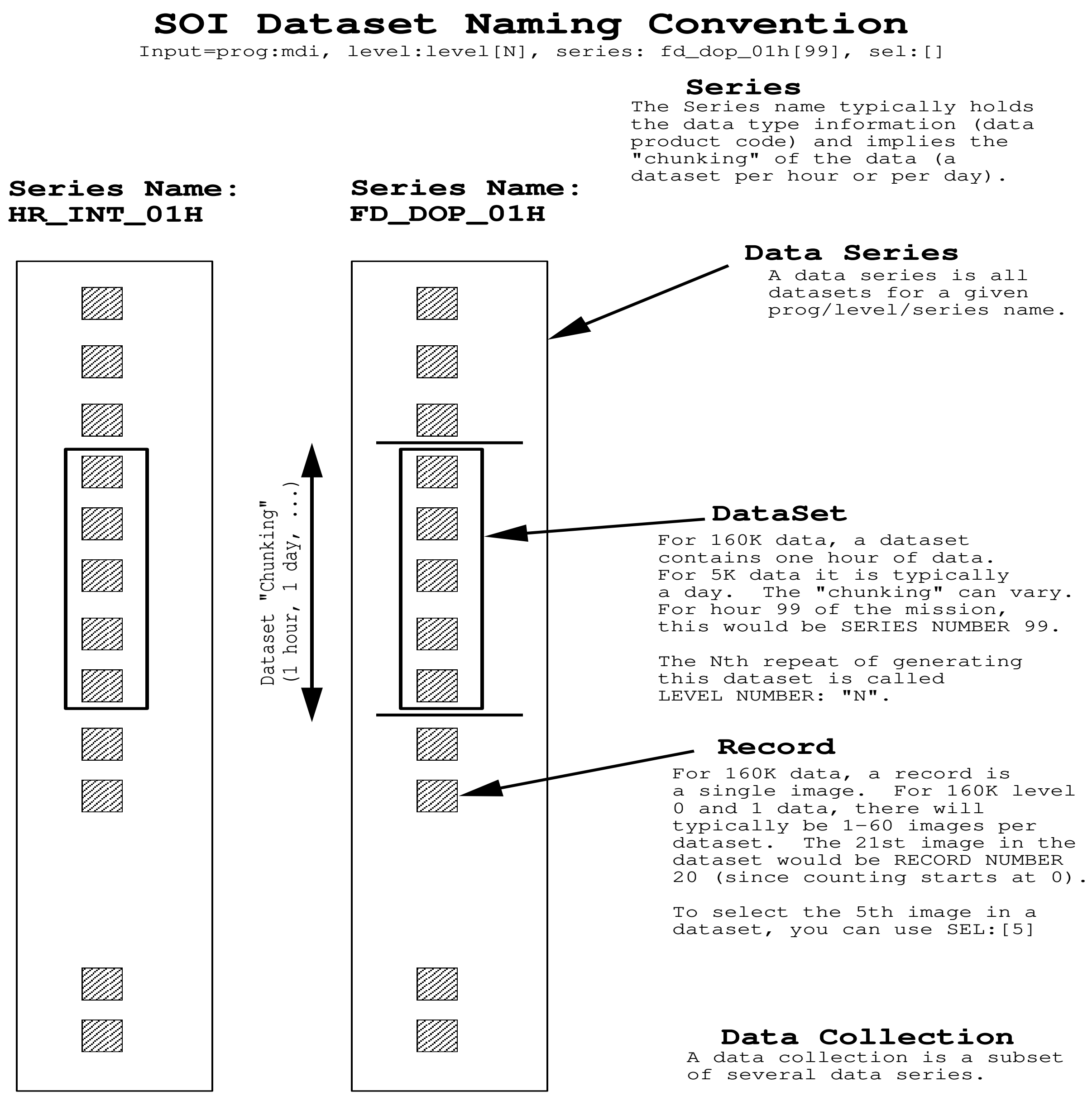

Figure 5: Dataset Naming Convention (LEVEL and SERIES)

Different Files in a Dataset	
File	*Description*
job.mmli	Copy of the MMLI map file
job.map	The strategy map file
job.param	The PARAM file created when the job.map file was created
job.<module>.<pid>.log	The history of processing – "ships log" (events as they happen)
job.ds_inputs	The names of the datasets used as input to create the dataset
*.index	details on the contents of a file and pointers into the file
*.ccdh	A binary copy of the IP and CCD header (level-0 only)
*.overview.fits	Summary information on the whole dataset
*.record.rdb	Information on each record (a line per image)
*.####.fits	The data (per image) – Several FITS files per dataset
*.info.rdb	Subset of the .record file to be placed in mission long table (obsolete?)

1.5.1 Sample Files for a Dataset

```
{dbase}/job.mmli
{dbase}/job.map
{dbase}/job.param
{dbase}/job.log
{dbase}/job.ds_inputs
{dbase}/ffffff01_01h.000099.overview.fits
{dbase}/ffffff01_01h.000099.record.rdb
{dbase}/ffffff01_01h.000099.00000.fits
{dbase}/ffffff01_01h.000099.00001.fits
{dbase}/ffffff01_01h.000099.00002.fits
{dbase}/ffffff01_01h.000099.00003.fits
.
.
.
{dbase}/ffffff01_01h.000099.00059.fits
```

1.5.2 Multi-Variable Datasets

A dataset can have several variables which are stored to separate files. For a dataset `prog: mdi, level: calib, series: dop_calib`, you can specify the variable name portion by using the "sel" portion, so it would be `sel: mult` to access the variable "mult". For the dataset: `prog:mdi, level:calib, series:dop_calib[4]`, there could be the following files:

```
{dbase}/dop_calib.mult.00001.fits
{dbase}/dop_calib.mult.00002.fits
{dbase}/dop_calib.mult.00003.fits
{dbase}/dop_calib.add.00001.fits
{dbase}/dop_calib.add.00002.fits
{dbase}/dop_calib.add.00003.fits
```

So there are three records for that dataset, and two variables. To specify the exact variable and record, and example would be: `prog:mdi, level:calib, series:dop_calib[4], sel:mult[2]` to get the second "mult" image in the dataset.

1.5.3 File Formats

TODO - CDF/ASCII/FITS/RDB... TODO - how to recognize if there is a file per record or all images in a 3-D cube in a single file. TODO - subset of keywords copied into FITS header from the .record file.

1.5.4 Ancillary and Calibration Datasets

TODO - It varies from set to set. Some sets are a single dataset for the whole mission, some have a dataset per hour, and others per day. The format ... TODO - calibration datasets

2 Data Series Definitions

2.1 The Elements Needed to Define a Data Series

There are several pieces of information that are needed to define a data series. The following list defines some of the different parts and where they can be found in the SCCS data system.

- The data series name. It is defined in the Oracle **Data Series Definition/Naming Table**.
- The type of data series is defined in the *CONFORM* field of the **Naming Table**. The CONFORM type implies the *IDREC* definition.
- How the data should be blocked into datasets is defined in the *BLOCK* field of the **Naming Table**
- Information on how to calculate the series number is defined in the *SER_TYP* field of the **Naming Table**. The SER_TYP value implies the epoch and the T_UNITS (??).
- The modules/programs required to create the data series are contained in the **MML File**. The name of which MML file to use is specified in the **Naming Table**.
- Optional additional information needed to fully specify a data series is contained in the *EXT_DEF* field of the **Naming Table**. For "Equally Spaced Time Series (TS_EQ)" data series, the time between records (images) is specified in this field. A sample value for this field is "T_STEP=60sec".
- Which data series can be used as input to create the data series. The **MML File** has the name of a dataset which has the list of acceptable input data series.
- The number of variables, the variable name(s), units, and notes are specified in the **MML File**.
- The keywords (individual pieces of "header" information) required for each record (image). There are several potential sources for keyword values.
  - Internally from the module creating the output
  - Keywords which should be copied from the input dataset keyword values. The list of keyword names to be copied from the input dataset is contained in a **separate dataset specified inside the MML file**
  - Keywords which have to be extracted from an ancillary database. The list of keyword names to be extracted is contained in a **separate dataset specified inside the MML file**. The ancillary dataseries name is also specified in the **MML File**
- The name of the Oracle mission long information table is specified in the **Naming Table**. The list of keywords which go into that table is contained in a **separate dataset which is specified inside the MML file**. This organization allows for a generic Oracle updating module which takes a subset of the information from the ".record" file and puts it into an Oracle table.

2.2 Conforming Datasets

TODO - define what is required to be a conforming dataset - .overview files, .record file, ...

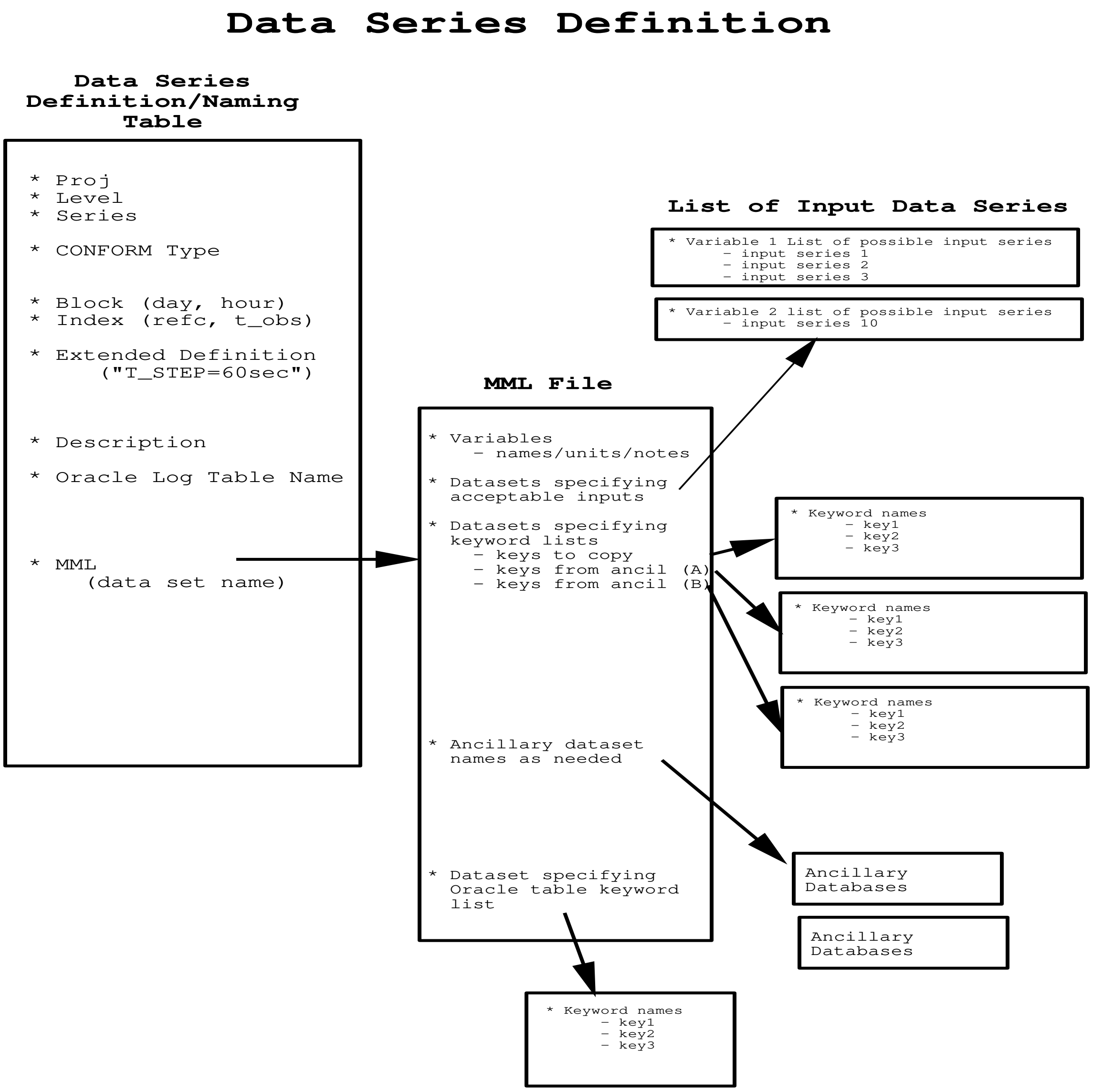

Figure 6: Overview of Defining a Data Series

Table of CONFORM Types			
Type	*Description*	*EXT_DESC Field Required*	*IDREC Implied*
SIMPLE	No dependence on time		none
AR_DPS	As Received, data product series		Integer second of mission from t_refc
TS_ES	Time series, equally spaced	T_STEP	Record # in mission long data series
FTS_EQ	Tile data series	T_STEP,F_STEP	Record # in mission long data series

2.2.1 Files Present in a Conforming Dataset

- **.overview** File: This overview file contains the information that a program can then use to determine what is in the dataset and how to access it. It tells how to interpret the series number and how to interpret the record number. Part of the file is a copy of the data series definition and part of the file is specific to the particular dataset. The variable information (min,max,avg,rms) is a summary of all records in the dataset. The variable statistics would be gathered from the individual image record statistics stored in the D_... series of names in the .record.rdb file.
- **.record** File: This file is a copy of the level 1 .record RDB file and contains the information that accompanies each image. There is a separate record file for each variable. If the image that should exist for record "n" does not exist, then there is a line in the RDB file, but with mostly blanks. This file includes any ancillary information which was gathered at the time the dataset was created.
- Individual data files: The individual data records are stored one time step and one variable per FITS file. The FITS headers contain at least a minimum number of keywords to describe the physical storage and to identify the image. The SOHO/SOI keyword information available in the data FITS header is a copy of the per record information, and the per record information file should be used as the master. If the image that should exist for record "n" does not exist, then there is no FITS data file.

2.2.2 Sample .overview File

```
# ------ Information from the Data Series Definition --------

SIMPLE  =                    T / file conforms with FITS standard; SOI v 0.7
NAXIS   =                    0

# The following keywords present for conforming SOI datasets
CONFORMS        'TS_ES'                 # Conformance level
```

```
TS_TYPE         'UT'                   # Record axis is UT
T_EPOCH         '1995.09.11_00:00:00'  # Day of launch, Base for nums
T_BLOCK         3660                   # number of seconds per dataset
T_STEP          60                     # seconds per record

V_I_DREC        'I_DREC'               # keyword to cross reference entries

# The following keywords describe the set of variables (selects) present
D_VARS          1                      # Numnber of variables
DNAME1          'FD_DOP'               # Variable name
DFNAME1         '.dop'                 # Data record filename prefix
DUNIT1          'm/s'                  # line-of-sight velocity in m/s
DNOTE1          'MDI observed velocity' # comment for variable no. 1.

# ------ Information about this particular Dataset --------

PROTOCOL        'SOI_FITS'             # Data is one record, one var per file
DS_PRG          'mdi'                  # Project name   (portion of dataset name)
DS_PRGN         1                      # Project number (portion of dataset name)
DS_LEV          'lev1.5'               # Level name     (portion of dataset name)
DS_LEVN         1                      # Level number   (portion of dataset name)
DS_SER          'fd_dop_01h'           # Series name    (portion of dataset name)
DS_SERN         003600                 # Series number  (portion of dataset name)

# The following keywords present for all types, found in DSDS main table
T_FIRST         '1996.02.08_00:00:25'  # Start of this dataset
T_LAST          '1996.02.08_07:59:25'  # Time of last data in this ds

D_MIN1          -2020.00               # Min velocity in entire dataset
D_MAX1          2035.23                # Max velocity in entire dataset
D_RMS1          1234.56                # RMS velocity in entire dataset
D_NUM1          48000000               # number of non-missing pixels

# The following keywords may be present
T_EPOCH         '1970.01.01_00:00:00'  # Epoch for any binary times

END                                    # Mark the end of the file
```

2.3 Typical Blocking for Standard Processing Datasets

Typical Blocking for Data Series			
Type	*Level*	*Block*	
Housekeeping	Raw	24 hour	
	Level-0	24 hour	
SCI5K	Raw	24 hour	
	Level-0	24 hour	per DPI
SCI160K	Raw	∼8 hour	
	Merged	1 hour	
	Level-0	1 hour	per DPC
	Level-1	1 hour	per DPC
	Level-1.5	??	per DP Class
	Level-2	??	??
Ancillary	S/C Pos & Motion	8 hours?	
	MDI Derived Sun Center	1 hour?	

2.4 Data Series Definition/Naming Table

Data Series Definition/Naming Table (Part I)

PROG	LEVL	SERIES	CONFORM	EXT_DESC	BLOCK	SER_TYP	LOG_TBL
mdi	ancil	sc_time_cor	SIMPLE		none	none	
mdi	lev0	ffffff01_01h	DPS_AR		1hr	t_refc	
mdi	lev0	ffffff02_01h	DPS_AR		1hr	t_refc	
mdi	lev0	ffffff03_01h	DPS_AR		1hr	t_refc	
mdi	lev0	ffffff04_01h	DPS_AR		1hr	t_refc	
mdi	lev1.5	fd_dop_01h	TS_EQ	T_STEP=60sec	1hr	t_obs	
mdi	lev2	v_heliomap_01h	TS_EQ	T_STEP=60sec	1hr	t_obs	
mdi	lev2	v_mlat_01h	TS_EQ		1hr	t_obs	
mdi	lev2	v_pox_0-49_01h	TS_EQ		1hr	t_obs	
mdi	lev2	v_pox_0-49_08h	TS_EQ		8hr	t_obs	
mdi	merge	merge160k	DPS_AR		1hr	t_ref	merge_hrsc_log
mdi	raw	sci160k	DPS_AR		1tape	tape	ingest_hrsc_log
mdi_ds_in	lev1	fd_dop	SIMPLE		none	none	
mdi_klist	copy	fd_dop	SIMPLE		none	none	
mdi_klist	extract	ephem1	SIMPLE		none	none	
mdi_klist	extract	sc_pos1	SIMPLE		none	none	
mdi_klist	oracle	fd_dop	SIMPLE		none	none	
mdi_mml	lev1.5	fd_dop	SIMPLE		none	none	
soho	ancil	sc_pos_mot	SIMPLE		none	none	

Data Series Definition/Naming Table (Part II)

PROG	LEVL	SERIES	MML_DS	DESCR
mdi	ancil	sc_time_cor		s/c time correction
mdi	lev0	ffffff01_01h		One hour of DPC ffffff01
mdi	lev0	ffffff02_01h		One hour of DPC ffffff02
mdi	lev0	ffffff03_01h		One hour of DPC ffffff03
mdi	lev0	ffffff04_01h		One hour of DPC ffffff04
mdi	lev1.5	fd_dop_01h	prog:mdi_mml,level:lev1.5,series:fd_dop	Full disk dopplergram evenly spaced
mdi	lev2	v_heliomap_01h		Remaped evenly spaced
mdi	lev2	v_mlat_01h		One hour mlat from v_heliomap
mdi	lev2	v_pox_0-49_01h		Tile one hr by 50 L evenly spaced
mdi	lev2	v_pox_0-49_08h		Tile 8 hr by 250 L evenly spaced
mdi	merge	merge160k		Raw hrtlm data w/station merge
mdi	raw	sci160k		Raw hrtlm data from Goddard tape
mdi_ds_in	lev1	fd_dop		Ds w/"series".IN file to define the input dataseries
mdi_klist	copy	fd_dop		Ds w/list of keywords to be copied from in to output
mdi_klist	extract	ephem1		Ds w/list of keywords to extract from ephem program
mdi_klist	extract	sc_pos1		Ds w/list of keywords to extract from sc pos db
mdi_klist	oracle	fd_dop		Ds w/list of keywords to cp from .rec to Oracle tbl
mdi_mml	lev1.5	fd_dop		Contains fd_dop series mml
soho	ancil	sc_pos_mot		The s/c position and motion db

Possible Values of Series Index			
SER_TYP	*Description*	*Epoch*	*T_Unit*
None	Series number is not used	-	-
t_ref	RefTime	TBD	TAI
t_refc	Corrected RefTime	TBD	TAI
t_obs	Observation Time	TBD	UT
ft_obs	Frequence and Obs Time	TBD	UT/Mode
t_ref0	Pre-launch RefTime	1993.01.01_00:00:00_UT	TAI
t_refc0	Pre-launch Corrected RefTime	1993.01.01_00:00:00_UT	TAI
t_obs0	Pre-launch Observation Time	1993.01.01_00:00:00_UT	UT
ft_obs0	Pre-launch Frequence and Obs Time	1993.01.01_00:00:00_UT	UT/Mode

3 Map Maker Language (MML)

An MML file will be created for most data series which will contain part of the information which defines that data series (the parts which are not already specified in the Oracle naming/definition table). The MML file is converted into a map file which consists of the exact modules (and the parameters) which are called to make the output dataset. The MML file looks very similar to a map file except that it makes references to data series instead of specific datasets.

3.1 MML Interpreter (MMLI)

There is a strategy level module (MMLI) that will take an MML text file and make a PE executable map file. The interpreter (MMLI) must have DSDS access, and should probably have the flexibility of any of our other modules.

Normally MMLI will be run by the level 1.5 grouping program which will get the MML text file from a database lookup for the target output dataseries. The grouping program would execute MMLI to build the map file and then call PE with this map file. A sample MMLI file is shown in section 3.4.1

PE executes and converts the MMLI map file into a specific map file. The resulting map file and param file go into an output dataset. The data series name specifications in the MML and MMLI are converted to **datasets** based on the times requested. There are three general sources of inputs to create the map file. They are:

1. The Information in the MMLI is passed in
   - The name of the output data series (*ds_out*)
   - The start and end times desired (*t_first* and *t_last*)
   - The output directory where the MMLI map file should be written (*outdir*)
   - The MML file name (*mml*)
2. The Oracle Data Series Definition/Naming Table is consulted and the following pieces are extracted from the table
   - The dataset blocking information (*block*)
   - The conforming type (*conforms*)
   - The epoch (*t_epoch*)
   - OPTIONALLY (for certain data series)
     - The separation between records (images) (*t_step* and *f_step*)
     - The mission long Oracle info table (*orc_log_table*)
3. The MML file has information on:
   - The number of variables (*d_vars*)
   - The variable names, units and information (*d_name1, d_unit1, dnote1*)
   - The name of a dataset which holds a list of the acceptable input data series to create this output dataset

- The name(s) of datasets which hold list of keywords to assemble
- The name(s) of ancillary datasets needed to assemble keyword values

3.2 The MML File

The MML file will normally be pointed to by a dataset name entry in the **dataseries_naming table** for the target output dataseries. A sample MML file is shown in section 3.4.2

Some rules for the MML file contents are:

- The MML has two sections. The definition section and the map building section. The map building section must come last and begins with the definition START_MAP.
- Everything is keyword=value pairs except for any shell escape commands starting with "!" which are passed on directly to the map file. The pairs are handled with CollectParams(), so no spaces are allowed in the value.
- All keyword=value pairs in the definition section are saved as environmental variables.
- Any keyword starting with DS_ is assumed to be a dataseries spec and will be expanded to its corresponding datasets given the time range and its blocking and so forth, as determined by an Oracle dataseries_naming table query.
- Any keyword starting with LK_ indicates that a "lookup" is required from the Oracle dataseries_naming table or another table. It is normally necessary to specify which data series is being looked up. For example, LK_DS_OUT_BLOCK says to get the "BLOCK" value for the $DS_OUT data series.
- Wherever $keyword is used in the map building section, its expansion will be substituted by getting its value from the environment.
- The variable $DS_OUT_NUM is derived internally by the MMLI program based on the input args of ds_out, t_first, etc.. For example, if the time span requested is 5 hours, and T_BLOCK is 1 hour, then $DS_OUT_NUM will be set to 5.
- The input arguments are available as keywords. For example, since "ds_map" is an argument, the keyword $DS_MAP is defined and available.
- The expession <$DS_x> is used only to specify input datasets for a module. <$DS_x> means the contents of the dataset given by $DS_x. This means that a file exists in the dataset with the name "series".IN file. For example, if $DS_x was `proj:mdi_ds_in, level:lev1, series:fd_dop` then the file `fd_dop.IN` exists in that dataset directory.

With our new thinking about how we build a conforming dataset, two changes are required of pe, that are both handled by the new map directives: START_ARCHIVE=1 and END_ARCHIVE=1. Currently, pe will determine to archive or mark for deletion the output wds after each module. Also, pe will not allow two different modules to output to the same wd. This is a sanity check that you knew what you were doing. However, now with a START_ARCHIVE/END_ARCHIVE block, pe will be changed to determine archive mode on the last module of the block, and will allow same wd output within the block.

TODO-REMOVE? - Notice these limitations: The series numbers currently come from the unix epoch and t_epoch is not used. This will change when our new standard mdi time functions are available. Also the 3 output series will really only be 2 with the new routines that know to get series numbers for series >= t_first and < t_last.

3.3 Use of Keyword Lists

It is possible to write generic modules if certain information is passed by a file containing the list of keywords to be extracted.

Some examples of how a keyword list could be used are:

- Specifying the keywords which should be copied from the input dataset into the output dataset
- Specify a set of keywords to be extracted from an ancillary dataset.
  - specify one list for values to be extracted from the S/C motion and position database
  - specify another list for values to be extracted from the fine S/C pointing and roll database
- Specify a list of keywords to pass out of a routine (solar ephemeris information for example)
- Specify a list of mnemonics to be extracted from the IP or camera header. This list will tell the location within the camera header for extraction.

3.4 Sample Pipeline Processing Steps

3.4.1 STEP 1: Interactive Creation of MMLI Map File

The module *mkmmli* prompts the user to specify what he/she wants created:

- The name of the output data series (*ds_out*)
- The start and end times desired (*t_first* and *t_last*)
- The MML file name (not the dataset name) (*mml*). Either the user specifies the MML file name, or the program does an Oracle lookup based on the output data series and converts the dataset name to a full-path file name.
- The dataset name for the DS_OUT map file (*ds_map*). The default is to derive a dataset name based on the time that the MKMMLI program is run (ie: `proj:maps,level:sched,series:950213_152`
- The MMLI map file name.

A sample MMLI file is shown below:

```
#map2.mmli
p=mmli d=1 a=t \
        ds_out=prog:mdi,level:lev1.5,series:fd_dop_01h \
        t_first=1994.02.01_00:00:00_UT \
        t_last=1994.02.01_02:00:00_UT \
        mml=/home/jim/STAGING/src/pipe/mml.txt \
        ds_map=prog:maps,level:sched,series:950210_164617
```

3.4.2 STEP 2: The DS_OUT Map file is Created

PE executes and converts the MMLI inputs into the map file to create the output dataset(s). The data series name specifications in the MML and MMLI are converted to datasets based on the times requested.

The MMLI map file is copied into the DS_OUT map dataset when the MMLI is executed, and the MMLI module creates the map file and param file.

Output: prog:maps, level:sched, series:YYMMDD.HHMMSS

```
{dbase}/maps/sched/YYMMDD.HHMMSS/job.mmli
{dbase}/maps/sched/YYMMDD.HHMMSS/job.map
{dbase}/maps/sched/YYMMDD.HHMMSS/job.params
```

A sample MML file is shown below:

```
#/home/jim/STAGING/src/pipe/mml2.txt

D_VARS=1                #Number of variables
D_NAME1=dop             #First variable name
D_UNIT1=m/s
DNOTE1=MDI_observed_velocity

CONFORM=$LK_DS_OUT_CONFORM
SER_TYP=$LK_DS_OUT_SER_TYP
ORA_LOG_TBL=$LK_DS_OUT_ORA_LOG_TBL
BLOCK=$LK_DS_OUT_BLOCK
T_EPOCH=$LK_DS_OUT_T_EPOCH
T_STEP=$LK_DS_OUT_T_STEP
#F_STEP=$LK_DS_OUT_F_STEP

# --- dataset with the list of acceptable input series
DS_IN1=prog:mdi_ds_in,level:lev1,series:fd_dop

# --- dataset with the list of keywords to be copied from input to output
DS_COPY_KLST=prog:mdi_klist,level:copy,series:fd_dop

# --- dataset with the list of keywords to be extracted from the s/c
#     position and motion ancillary database
DS_SC_POS_KLST=prog:mdi_klist,level:extract,series:sc_pos1
# --- the s/c position and motion database
DS_SC_POS_IN=prog:soho,level:ancil,series:sc_pos_mot

# --- dataset with the list of keywords to be extracted from the ephemeris program
DS_EPHEM_KLST=prog:mdi_klist,level:extract,series:ephem1
```

```
# --- dataset with the subset of keywords from the map params file to be copied to the overview
DS_OVERV_KLST=prog:mdi_klist,level:overview,series:general1

# --- dataset with the list of keywords to be copied from the .record file
#     into the Oracle db table
DS_ORA_LIST=prog:mdi_klist,level:oracle,series:fd_dop

#Now start the map definition section which must come last
START_MAP=1

!setenv mdi {dbase}/{level}/{series}/{#%06d#series}

HOST=tarax.Stanford.EDU
DSDSOUT=$DS_OUT_NUM*50

START_ARCHIVE=1

#copy selected keywords from the input datasets to the output
p=rec_copy_keys d=1 \
        in=<$DS_IN1> out=$DS_OUT keylist=$DS_COPY_KLST  \
        t_epoch=$T_EPOCH block=$BLOCK t_step=$T_STEP ser_typ=$SER_TYP

#extract certain keywords from the s/c pos/motion database
p=rec_get_sc_pos d=1 \
        in=$DS_SC_POS_IN out=$DS_OUT keylist=$DS_SC_POS_KLST

#extract certain keywords from ephemeris
p=rec_get_ephem d=1 \
        out=$DS_OUT keylist=$DS_EPHEM_KLST

#copy the data portions
p=ds_copy d=1 a=n \
        in=<$DS_IN1> out=$DS_OUT
        #OPTIONALLY: dfname1=$D_NAME1, dfname2=$D_NAME2

#build the overview file
p=bld_overvfile d=1 \
        out=$DS_OUT \
        params=$DS_MAP, keylist=$DS_OVERV_KLST

#update the mission long Oracle table from the output .record files
```

```
p=update_ora_tbl d=1 \
        in=$DS_OUT list=$DS_ORA_LIST out=$ORA_LOG_TBL

END_ARCHIVE=1
```

A sample Map file is shown below:

```
#/home/jim/STAGING/src/pipe/map.auto
#built by mmli from /home/jim/STAGING/src/pipe/mml2.txt
#
!setenv mdi {dbase}/{level}/{series}/{#%06d#series}
HOST=tarax.Stanford.EDU

DSDSOUT=150

START_ARCHIVE=1

p=rec_copy_keys \
        d=1 \
        in=prog:mdi,level:lev0,series:ffffff03_01h[9504-9506];prog:mdi,level:lev0,series:fff
        out=prog:mdi,level:lev1.5,series:fd_dop_01h[9504-9506],sel:[] \
        t_epoch=1990.01.01_00:00:00_UT block=3600 t_step=60 ser_typ=t_obs
        keylist=prog:mdi_klist,level:copy,series:fd_dop

p=rec_get_sc_pos \
        d=1 \
        in=prog:soho,level:ancil,series:sc_pos_mot \
        out=prog:mdi,level:lev1.5,series:fd_dop_01h[9504-9506],sel:[] \
        keylist=prog:mdi_klist,level:extract,series:sc_pos1

p=rec_get_ephem \
        d=1 \
        out=prog:mdi,level:lev1.5,series:fd_dop_01h[9504-9506],sel:[] \
        keylist=prog:mdi_klist,level:extract,series:ephem1

p=ds_copy \
        d=1 \
        a=n \
        in=prog:mdi,level:lev0,series:ffffff03_01h[9504-9506];prog:mdi,level:lev0,series:fff
        out=prog:mdi,level:lev1.5,series:fd_dop_01h[9504-9506],sel:[], \
        dfname1='dop', dfname2='mag'
```

```
p=bld_overvfile \
        d=1 \
        out=prog:mdi,level:lev1.5,series:fd_dop_01h[9504-9506],sel:[]
        params=prog:maps,level:sched,series:950210_164617 \
        keylist=prog:mdi_klist,level:overview,series:general1

p=update_ora_tbl \
        d=1 \
        in=prog:mdi,level:lev1.5,series:fd_dop_01h[9504-9506],sel:[] \
        list=prog:mdi_klist,level:oracle,series:fd_dop \
        out=dummy

END_ARCHIVE=1
```

When the MMLI is executed, it creates the map file, and it also creates a **param** file. This param file is the list of environment variables and their values. A sample Param file is shown below:

```
DS_OUT          prog:mdi,level:lev1.5,series:fd_dop_01h
T_FIRST         1994.02.01_00:00:00_UT
T_LAST          1994.02.01_02:00:00_UT


MML_DS          prog:mdi_mml,level:lev1.5,series:fd_dop
CONFORM         TS_ES
T_STEP          60
BLOCK           3600
SER_TYP         t_obs
ORA_LOG_TBL     ingest_hrsc_log


D_VARS          1
D_NAME1         dop
D_UNIT1         m/s
DNOTE1          MDI_observed_velocity

DS_IN1          prog:mdi_ds_in,level:lev1,series:fd_dop
DS_COPY_KLST    prog:mdi_klist,level:copy,series:fd_dop
DS_SC_POS_KLST  prog:mdi_klist,level:extract,series:sc_pos1
DS_SC_POS_IN    prog:soho,level:ancil,series:sc_pos_mot
```

```
DS_EPHEM_KLST   prog:mdi_klist,level:extract,series:ephem1
DS_ORA_LIST     prog:mdi_klist,level:oracle,series:fd_dop

HOST            tarax.Stanford.EDU
DS_OUT_NUM      3
DSDSOUT         150
```

3.4.3 STEP 3: The Map File is Executed

The map file does not have to be executed immediately after it is built. The processing steps for this sample map file are discussed in the "SCI160K Level 1.5 Gathering" chapter. See page 52

When PE is completed executing the map file, it:

- Copies the map file over to the output dataset(s)
- Copies the param file over to the output dataset(s)
- Creates a file "dsds.ds_inputs" which lists all of the datasets which were referenced during the execution of the map file
- Copies the runtime log file over to the output dataset(s)

3.4.4 Sample .overview File

```
# ------ Information from the Data Series Definition --------

SIMPLE  =                    T / file conforms with FITS standard; SOI v 0.7
NAXIS   =                    0

# The following keywords present for conforming SOI datasets
CONFORMS        'TS_ES'                # Conformance level
TS_TYPE         'UT'                   # Record axis is UT
T_EPOCH         '1995.09.11_00:00:00'  # Day of launch, Base for nums
T_BLOCK         3660                   # number of seconds per dataset
T_STEP          60                     # seconds per record

V_I_DREC        'I_DREC'               # keyword to cross reference entries

# The following keywords describe the set of variables (selects) present
D_VARS          1                      # Numnber of variables
DNAME1          'FD_DOP'               # Variable name
DFNAME1         '.dop'                 # Data record filename prefix
DUNIT1          'm/s'                  # line-of-sight velocity in m/s
```

```
DNOTE1          'MDI observed velocity' # comment for variable no. 1.

# ------ Information about this particular Dataset --------

PROTOCOL        'SOI_FITS'              # Data is one record, one var per file
DS_PRG          'mdi'                   # Project name   (portion of dataset name)
DS_PRGN         1                       # Project number (portion of dataset name)
DS_LEV          'lev1.5'                # Level name     (portion of dataset name)
DS_LEVN         1                       # Level number   (portion of dataset name)
DS_SER          'fd_dop_01h'            # Series name    (portion of dataset name)
DS_SERN         003600                  # Series number  (portion of dataset name)

# The following keywords present for all types, found in DSDS main table
T_FIRST         '1996.02.08_00:00:25'   # Start of this dataset
T_LAST          '1996.02.08_07:59:25'   # Time of last data in this ds

D_MIN1          -2020.00                # Min velocity in entire dataset
D_MAX1          2035.23                 # Max velocity in entire dataset
D_RMS1          1234.56                 # RMS velocity in entire dataset
D_NUM1          48000000                # number of non-missing pixels

# The following keywords may be present
T_EPOCH         '1970.01.01_00:00:00'   # Epoch for any binary times

END                                     # Mark the end of the file
```

4 SCI160K Level 0 Processing

4.1 Purpose

Level 0 processing is used to convert the raw telemetry stream into data products (typically images). The processing will:

- Decompress the telemetry stream where necessary
- Assemble a data product from the raw telemetry stream
- Embed cropped images into a rectangular data array (for cropped images only – where only the pixels on the solar disk are send down). For limb figures the data will be saved as a 1-dimensional stream of words.
- Save the resulting data product in FITS or CDF files.

4.2 Data Header Contents

Level 0 .record file will contain:

- Information from the IP and camera header
  - ref time
  - sequence number
  - michelson motor tuning positions (?)
  - exposure duration (?)
  - ...
- UT Date/Time derived from ref_time (?)
  - Solution number for the conversion (?)
- Information on the data
  - Data completeness (%)
- ??
- ??

4.2.1 Sample FITS Header

```
SIMPLE  =                    T / file conforms with FITS standard; SOI V0R8B3
BITPIX  =                   16
NAXIS   =                    2
NAXIS1  =                  256
NAXIS2  =                  256
PKTSTART=                 9982
DPC     = 'ffffff03'
DATE-OBS= '26/04/94'
TIME-OBS= '04:40:37'
```

```
IMAGE_T = '1994:04:26:04:40:37'
CONTROL = '   0   1   0   0   0'
DEPSCR_0= ' 232   3'
DEPSCR_1= '   0   0'
DEPSCR_2= ' 120   0'
DEPSCR_3= '  80   2'
DEPSCR_4= ' 112   0'
DEPSCR_5= '   0   0'
DEPSCR_6= '   0   0'
DEPSCR_7= '   0   0'
DEPSCR_8= '   1   0'
DEPSCR_9= '   0   0'
DEPSCR_A= '   0   0'
DEPSCR_B= '   1   8'
DEPSCR_C= '   3   0'
DEPSCR_D= '   8   2'
DEPSCR_E= '  69   2'
DEPSCR_F= '  54   2'
SQID    =          -1072546176
SQNUM   =                    5
SQFRM   =                   20
REFTIME =     1.1460264311E+09
RSTRT   =                    0
FRMNUM  =                  229
HKPACK  =                 2749
DEPVR   = '   0  21'
OCTVR   = '   0   0'
SEQVR   = '   0   4'
IPVR    = '   0   0'
QUEUEVR = '   0   0'
ALIGN   =          -1044201028
C1POS   =                   55
C2POS   =                   89
MISCSTAT=                  132
LTERRS  = '   0   0   0   0'
LTSTAT  =                    0
HTRSTAT = '  11   0  18   0'
OVNTEMP = ' 125   0'
OPTEMPS = '   0 194'
CCDTMP_0= '   0   0   0   0   0   0   0   0   0   0'
CCDTMP_1= '   0   0   0   0   0   0   0   0   0   0'
MKM1POS =                    2
```

```
MKM2POS =                    2
MKPAPOS =                    2
END
```

4.3 Processing Steps

4.3.1 STEP 1 Ingest HRSC Data Tape from Goddard

Input: 160K high rate science data 8mm tape from Goddard. 1 tape/day, 1 file per contiguous station contact. Approximately 8 hrs of data per station, with possible data overlap in end and start of stations. Assume the tape is labeled with a 6 digit tape number in this case tape# 001000. (This may change, but we control the tape numbers.) Assume the file names are something like MDI_HRSC.G01_yyyymmdd. Assume the tape is initially tar'd into a scratch staging directory such as /stage/hrsc_tape. Once the ingest_hrsc_svc is run this staging area is free.

```
!setenv mdi wd:{dbase}/{level}/{series}/{#%06d#series}
HOST=fault.Stanford.EDU
p=ingest_hrsc a=n \
    in=/stage/hrsc_tape/ \
    out=prog:mdi,level:raw,series:sci160k[1000]
```

Processing: The ingest_hrsc server reads the input directory and copies each file into the assigned db storage directory given by the mdi env variable. The server looks at each station file and makes an info file containing each station file name with the start and end sfdu times and ref times of the station file as well as the number of packets. The server also makes an index file for each station file that contains per image information. The info file and index files are part of the same output data series as the station files. When the server completes, pe catalogs the dataset name in the database. Formerly, we stated that the server would return the start and end times found for all the station files, however this will now be done by the next step - the ingest_hrsc_log_svc. This eliminates any special handling by pe for different servers.

Output:

```
{dbase}/raw/sci160k/001000/
   MDI_HRSC.G01_yyyymmdd          (etc. for each sta file on the tape)
   MDI_HRSC.G01_yyyymmdd.index    (etc. for each sta file on the tape)
   ingest_hrsc_station.info       (one file w/info on all stations)
```

SAMPLE INGEST_HRSC .INDEX file (STEP #1) [One file per station contact / One entry per image]					
File Packet#	MDI Pkt_Seq#	DPC	#ofPkts	DP#	RefTime
2190	2687	ffffff03	185	0	1146025984.00
2375	2872	ffffff03	247	1	1146026112.00
2622	3119	ffffff03	185	2	1146026240.00

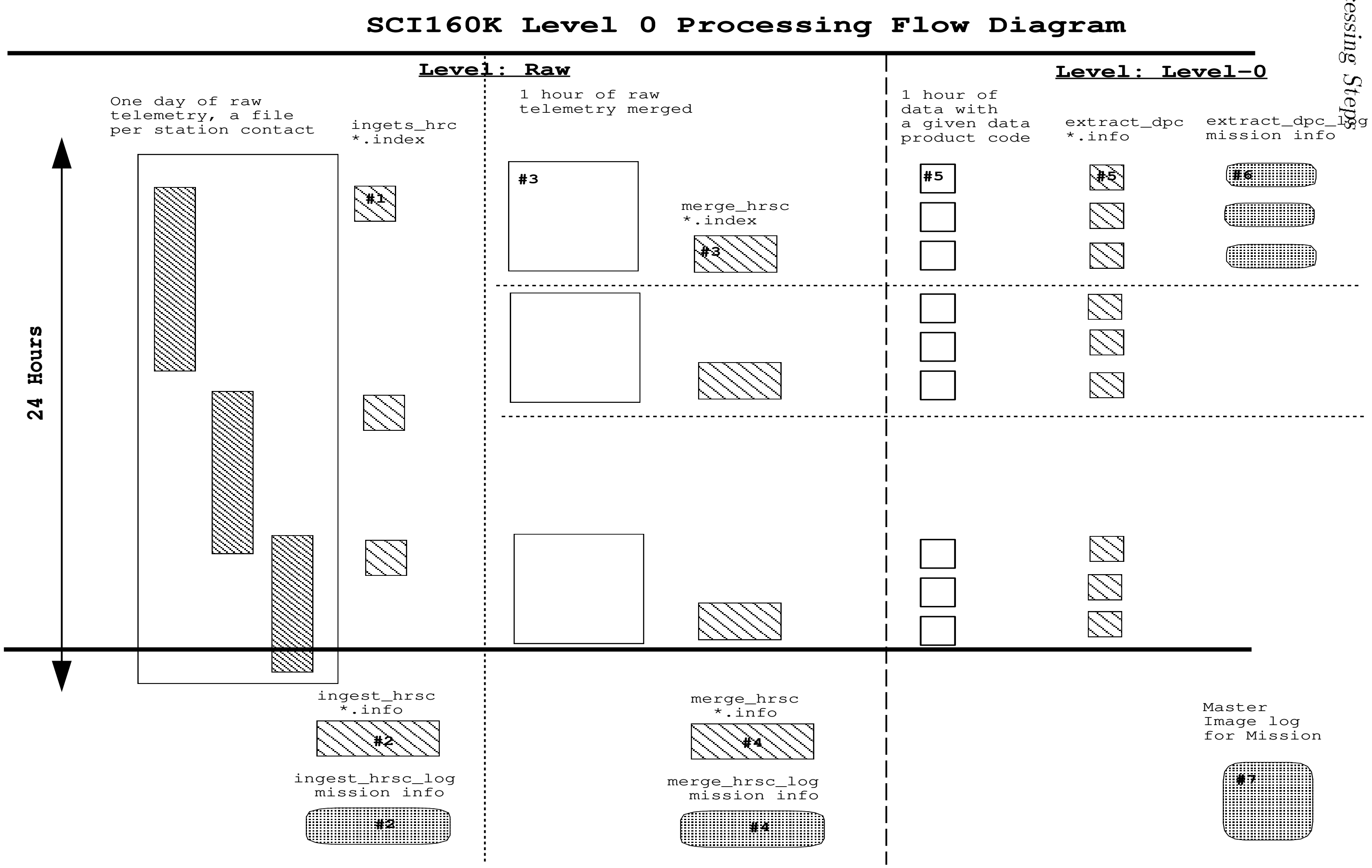

Figure 7: Flow and Steps for Level 0 Processing

The *MDI Pkt_Seq#* value rolls over at a value of 16384 (14 bits) which corresponds to approximately 15 minutes of SCI160K data.

SAMPLE INGEST_HRSC .INFO file (STEP #1) [One file per tape (day)]						
SAMPLE INGEST_HRSC_LOG Database (STEP #2) [One mission long table / One entry per station contact file]						
Tape#	*Station File*	*Start SFDU Time*	*End SFDU Time*	*#ofPkts*	*Start RefTime*	*End RefTime*
1000	MDI_HRSC.GO1_19940426	1146025984.00	1146026752.00	15650	1146025984.00	1146026752.00
1000	MDI_HRSC.GO2_19940426	1146025984.00	1146026752.00	15650	1146025984.00	1146026752.00
1001	MDI_HRSC.GO1_19940426	1146025984.00	1146026752.00	15650	1146025984.00	1146026752.00

Size: For 160Kbits/sec this is 72Mbytes/hour. For 72Mb/hr this is 1.7Gb/day. Say 2Gb/day with station overlap. To archive this at 300Kb/sec tape transfer rate this is $\sim$ 2 hrs.

NOTE: A "sanity" check for the quality of the Ref_Time is performed at this time to make sure that the values in the files are not corrupted. This consists of making sure that the value is contained within the day for the raw telemetry file, and to make sure that it is always increasing.

TODO ?? - New step which is to patch bad ref_time if it found some (??)

4.3.2 STEP 2 Log the Ingested HRSC Data

Input: The ingest highrate science log server is run for each dataset output by the ingest_hrsc server.

```
p=ingest_hrsc_log d=1 \
    in=prog:mdi,level:raw,series:sci160k[1000]
```

Notice that no env variable mapping or out dataset name are required for this server.

Processing: This server reads the info file for the given input dataset, i.e. ingest_hrsc_station.info. For each entry in the info file, a row is added to the ingest_hrsc_log table in the database. This table has the fields shown in table above.

It is assumed that from the tape number the full dataset name is implied, either by fixed convention or from a db table that defines the log tables and their associated dataset types or from some index put in the log table entries. All this is to avoid repeating the full dataset name in each row of the info table. The server also keeps track of the earliest and latest start and stop times found for the entire info file. These times are updated in the main database table for the given dataset. In this way, the time of any data contained in the dataset is bracketed.

Output: No new dataset. A row is added the the ingest_hrsc_log table in the database for each entry in the info file. The main table for the dataset is updated with the earliest and latest data times.

4.3.3 STEP 3 Merge the HRSC Data

Input: The merge server is run for every hour of merged station data that you want output. The hour of interest is found by querying the db for any ingested data that contains that hour. This is a reftime hour (H) that starts at 0 for some baseline and runs up for each hour of the mission. The query will give you all the TTTT numbers that contain that hour. Suppose that the query gives you tapes 1000 and 1001 that contain the hour 99.

```
!setenv mdi wd:{dbase}/{level}/{series}/{#%06d#series}
p=merge_hrsc d=1 a=n \
    in=prog:mdi,level:raw,series:sci160k[1000-1001] \
    out=prog:mdi,level:merge,series:merge160k[99]
```

Notice that the series name must be different because it has a different index, i.e. hour. (The previous alternative of having the same series name but a different prog name was not popular.)

Processing: The merge server looks at all the ingest_hrsc_station.info files of the given "in" datasets. It then knows the station file names that may contain the data. It reads these files and creates a merged file removing any station overlap. The server also outputs an index file containing all the images for all the dpcs. It also outputs a dpc.info file containing a summary of all the dpc found.

Output:

```
{dbase}/merge/merge160k/000099/
merge160k                (one file w/one hour of merged data)
merge_hrsc.index         (one file for all images and dpc)
merge_hrsc_dpc.info      (one file with dpc summary info)
```

The output file is one hour of merged data (72MB). (The file has all the DPCs that were received for that hour in increasing time.) The index file has the location of each image in the merged file for all the dpc.

SAMPLE MERGE_HRSC .INDEX file (STEP #3) [One file per hour / One entry per image		
DPC	Pkt#	RefTime
ffffff03	0	1146026112.00
ffffff03	186	1146026112.00
ffffff03	374	1146025984.00
ffffff03	559	1146025984.00

SAMPLE MERGE_HRSC .INFO file (STEP #3) [One file per hour]		
SAMPLE MERGE_HRSC_LOG database (STEP #4) [One table per mission / One entry per DPC per hour]		
Hour#	DPC	# of Occurences
99	ffffff03	73

Final Step: Build the map file which will drive the extract_dpc server. This build is done now because it is necessary to know which DPCs exist for the given hour in order to call the proper level 0 processing program.

Size: This is 1.7Gb/day. To archive it at 300Kb/sec is 1.6 hrs.

4.3.4 STEP 4 Log the Merged HRSC Data

Input: The merged hrsc data for each hour of the mission as output by merge_hrsc_svc.

```
p=merge_hrsc_log h=fault.Stanford.EDU d=1
    in=prog:mdi,level:merge,series:merge160k[99],sel:[]
```

Notice that no env variable mapping or out dataset name are required for this server.

Processing: This server reads the merge160k.000099.merge_hrsc.info file for the given input dataset. For each entry in the info file, a row is added to the merge_hrsc_log table in the database. See the table in the previous section.

4.3.5 STEP 5 Extract DPC from the Merged Data:

TODO - Jim - is it true that the database is queried (?)

Input: The extract dpc server is run for each dpc in the merged data. The data base is queried for all the dpc that are in the hour of interest. For each dpc, an extract_dpc_svc is run with the particular dpc as a parameter. An addition parameter is the location of the dpc table that has info for all possible data product codes (like size).

```
!setenv mdi wd:{dbase}/{level}/{series}; bn:{series}.{#%06d#series}
p=extract_dpc h=fault.Stanford.EDU d=1 a=1 \
    in=prog:mdi,level:merge,series:merge160k[99],sel:[] \
    out=prog:mdi,level:lev0,series:fulldskV[99],sel:[] \
    dpc=ffffff01 \
    dpc_table=/home/soi/CM/tables/dpc_table
p=extract_dpc h=fault.Stanford.EDU d=1 a=1 \
    in=prog:mdi,level:merge,series:merge160k[99],sel:[] \
    out=prog:mdi,level:lev0,series:fulldskV[99],sel:[] \
    dpc=ffffff03 \
    dpc_table=/home/soi/CM/tables/dpc_table
```

(note: ffffff03 is not necessarily the fulldskV dpc code.)
Output:

```
{dbase}/lev0/ffffff01_01h/000099/nnnn.fits        (or 0.cdf)
{dbase}/lev0/ffffff01_01h/000099/nnnn.ccdh
{dbase}/lev0/ffffff01_01h/000099/extract_dpc.info
```

Where extract_dpc.info has relevant per image information pertinent to that dpc.

SAMPLE EXTRACT_DPC .INFO file (STEP #5) [One file per hour per DPC]								
SAMPLE EXTRACT_DPC_LOG database (STEP #6) [One table per mission per DPC / One entry per image]								
Dataset Name (?)	RefTime	DPC	SeqID	HK Pkt	ImgMin	ImgMax	ImgAvg	ImgDev
???	1146026431	ffffff01	ff778833	234245	100	200	50	20
???	1146026491	ffffff01	ff778833	234249	110	200	50	30
???	1146026551	ffffff01	ff778833	234366	120	200	50	25

TODO - See Katie for definition/contents

TODO - will there be a table per data series, or will we combine data series into one table (think of level 0 type data where all we want to log is min/max/avg/dev. We don't want a table for each DPC.

NOTE: Definition of These Fields Needs Considerable Work

Size: This data is decompressed and represents 1024x1024 images (2 Mb as shorts). 1440 min/day * 2 Image/min * 2 Mb/Image = 5.7 Gb/day. To archive it at 300 Kb/sec = 5.3 hrs.

4.3.6 STEP 6 Log the Extracted DPC Data:

Input:

```
p=extract_fulldskV_log d=1 \
     in=prog:mdi,level:lev0,series:fulldskV[99],sel:[]

p=extract_fulldskB_log d=1 \
in=prog:mdi,level:lev0,series:fulldskB[99],sel:[]
```

Reads the extract_dpc.info in the given input dataset and updates a dpc specific database table with the per image info.

Final Step: Build the map file which will drive the calib_dpc_class server. This build is done now because it is necessary to know which DPCs exist for the given hour in order to call the proper level 1 processing program.

4.3.7 STEP 7 Make a Master Image Log

Takes the extract_dpc.info files for all the dpc for the given hour and makes one info file with all the info summarized. There is no seperate database info generated. This is just a flat file of summary data.

5 SCI5K Level 0 Processing

5.1 Purpose

For the SCI5K data: Level 0 processing takes the raw telemetry and creates images. The telemetry compression and the image cropping (only sending down the pixels on the solar disk) are removed and the output is a rectangular two dimensional image saved in a FITS file, except for the limb figure which is saved as a 1-D stream of data values.

TODO - Expand and clean up

5.2 Data Record File Contents

Level 0 .RECORD file will contain:

- Information from the data product header
  - RefTime
  - Number of pixels in the data product (NpixImage)
  - ... (?)
- Information gathered from the housekeeping based on RefTime
  - ????
  - ????
  - ????
  - ????
- Information from the data
  - Min/Max/Avg/Dev (?)
  - Data completeness (%)
- ??

5.2.1 Data Products within SCI5K Data

There are 24 SCI5K data packets per major frame, which results in approximately 96 packets per minute (14.98 sec per major frame). Each packet has a 14 byte packet header and 384 bytes of data. At the start of a data product, there is a 38 byte header. A single data product can span up to 128 packets, but will most likely span less than 96 packets.

Item	*Description*	*% of 5K Telemetry*
Medium-L	Velocity	80%
Limb Figure	Intensity	5%
Flux Budget	Continuum	5%
Golf Proxy	Line Depth	5%
LOI	Velocity and Intensity	2-5%

5.3 Telemetry Format

For a complete description of the telemetry format, see "Low Rate Science Data Description" MDI-MD340016, written by M. Levay. Below are a few points regarding the telemetry format.

The limb figure, flux budget and golf proxy are averaged over 24 minutes, but comes down once every 12 minutes. 1/12th of the data product is sent down each minute. For the 12 minute average data, the data is broken into 1/12th parts and then compressed into the data segments. The RefTime in the 24 minute average data products is the RefTime for the minute when the segment comes down (ie: it will have 12 different RefTimes for the one data product).

Each data product segment is allocated a fixed number of bytes of telemetry. Since the compression efficiency if not known, the data handler is generally given much more un-compressed data than will fit into the data segment. If the compressed data does not fill up the data segment, then the next data segment will imediately follow (ie: there is no padding/fill). The maximum size of a data segment is 8192 words. The maximum nuber of data segments within a minute (4 major frames) is 31 separate segments.

The start of a data header can be anywhere in the packet and there are no fixed patterns from minute to minute. The data header can be split across packets.

The segment number for a packet where the data header starts at the first word will be 1. If the packet starts in the middle of a packet, then the segment number will be "N", where "N" is the number of packets needed for the previous data segment, and the following packet will be number 2. This applies to a data header split across packets too.

5.4 Processing Steps

Since a data product for a given day may show up in telemetry packets for the following day (at the day boundry), the processing for a given day will not be performed until the next day's worth of data is available.

5.4.1 STEP 1 Ingest Data

Purpose: To determine what data products are in the input data and where the start of each data product occurs.

Input: SCI5K science data is delivered from GSFC on CDROM with all sci5k packets for a given day in one file. The temporary near-real-time data files made at the EOF will also be a single file per day.

Processing: The data is copied from CDROM to a magnetic disk within the DSDS (if necessary). An index file is created describing the contents of the the input file and the locations of the start of each data segment in the file.

Output:

```
{dbase}/raw/sci5k/001000/yyyymmdd.sci5k
{dbase}/raw/sci5k/001000/yyyymmdd.sci5k.index
```

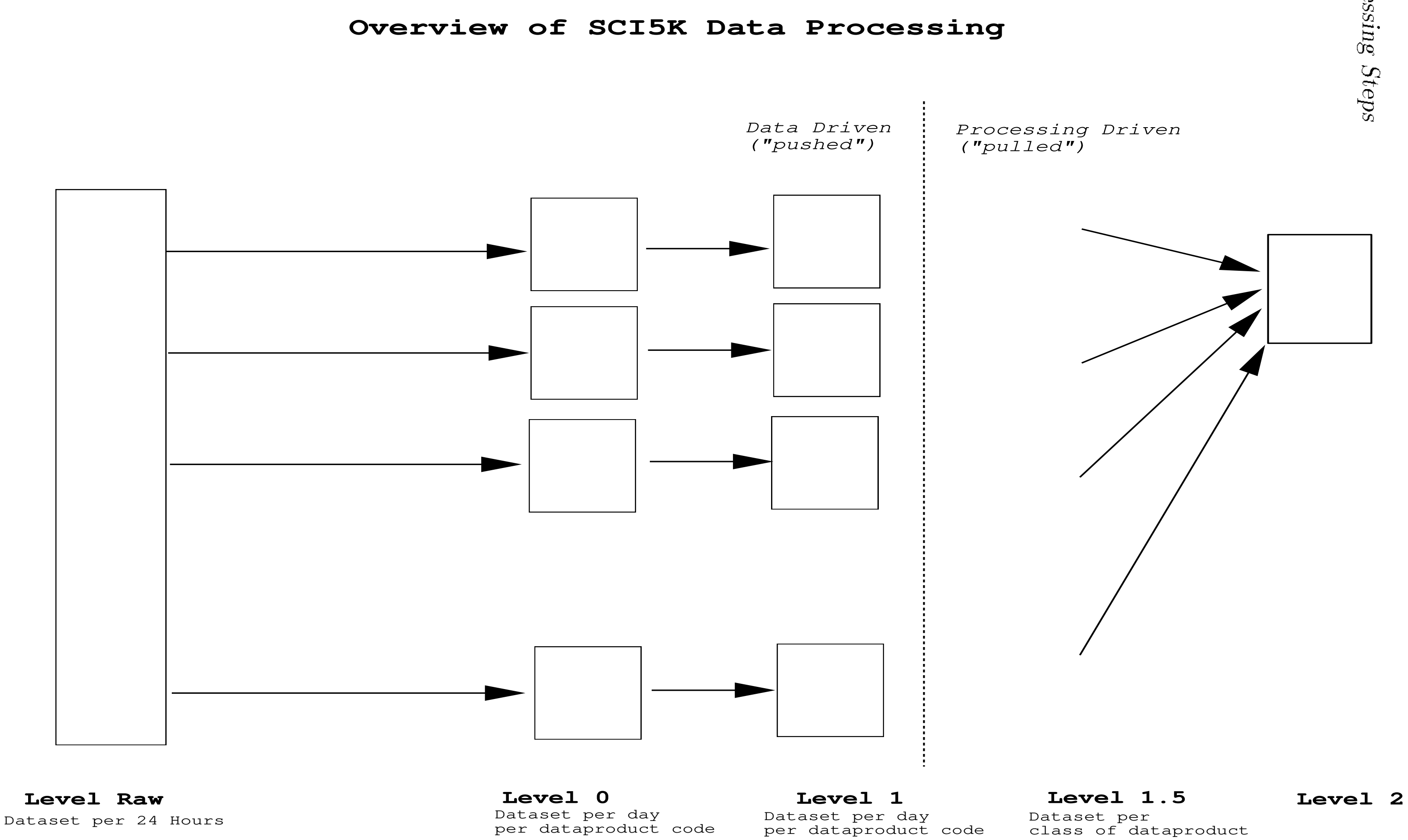

Figure 8: Overview of SCI5K Data Flow

SAMPLE INGEST_MRSC .INDEX file (STEP #1) [One file per 24 hours / One entry per data product]						
File Packet#	Seg# in DP	Byte Offset	DPC	# Words in DP Seg	RefTime	#Miss Packets
2190	1	0	ffffff01	200	1146025924.00	0
2375	1	300	ffffff02	220	1146025924.00	0
2622	1	54	ffffff03	300	1146025924.00	0
3190	1	200	ffffff04	20	1146025924.00	0
3375	5	123	ffffff05	4096	1146025924.00	0
4375	4	23	ffffff05	4096	1146025924.00	1
5175	3	135	ffffff05	4096	1146025924.00	0
6075	2	245	ffffff05	4096	1146025924.00	0
7175	1	44	ffffff05	456	1146025924.00	0
8190	1	10	ffffff01	200	1146025984.00	0
8375	1	340	ffffff02	220	1146025984.00	0
8622	1	5	ffffff03	300	1146025984.00	0

Size: 24 packets per 15 seconds * 398 bytes = 55 megabytes per day

NOTE: A "sanity" check for the quality of the Ref_Time is performed at this time to make sure that the values in the files are not corrupted. This consists of making sure that the value is contained within the day for the raw telemetry file, and to make sure that it is always increasing.

TODO ?? - New step which is to patch bad ref_time if it found some (??)

5.4.2 STEP 2 Extract DPI:

Purpose: To (1) to read each data segment in the data product and to decompress the stream of data words, (2) to gather some status information from the housekeeping database for the minute the data product was taken, (3) to get the bin list information for the data product, and (4) to optionally insert the cropped image into a rectangular array. All limb figure DPCs map to a single DPI, otherwise the DPI equals the DPC.

Input: The extract dpi server is run for each dpi in the sci5k data. The input .index file is inspected and a separate extract_5k_dpi module call is made for each unique DPI, with the DPI as a parameter. An addition parameter is the location of the dpc table that has info for all possible data product codes (like size). Two days of input are required for one day of output because data for a given day may be telemetered down in a packet in the next day.

```
!setenv mdi wd:{dbase}/{level}/{series}; bn:{series}.{#%06d#series}
p=extract_5k_dpi h=fault.Stanford.EDU d=1 a=1 \
    in=prog:mdi,  level:raw,  series:sci5k[99,100],    sel:[] \
    out=prog:mdi, level:lev0, series:ffffff01_24h[99], sel:[] \
    dpc=ffffff01 \
```

```
    dpc_table=/home/soi/CM/tables/dpc_table
p=rec_get_hk h=fault.Stanford.EDU d=1 a=1 \
    in=prog:mdi, level:lev0, series:ffffff01_24h[99], sel:[] \
    hk=prog:mdi, level:raw,  series:mdihk[99], sel:[] \

p=extract_5k_dpi h=fault.Stanford.EDU d=1 a=1 \
    in=prog:mdi,  level:raw,  series:sci5k[99,100],    sel:[] \
    out=prog:mdi, level:lev0, series:ffffff02_24h[99], sel:[] \
    dpc=ffffff02 \
    dpc_table=/home/soi/CM/tables/dpc_table
p=rec_get_hk h=fault.Stanford.EDU d=1 a=1 \
    in=prog:mdi, level:lev0, series:ffffff02_24h[99], sel:[] \
    hk=prog:mdi, level:raw,  series:mdihk[99], sel:[] \

# And so forth for each DPI
```

Output: A single FITS file will contain all of the data for the 24 hours. For non limb figure data products, the data will be a 3-D array where the third dimension is time. For the limb figure data products, the data will be 2-D where the first dimension is the stream of data words and the second dimension is time. The first dimension will have to be set to the largest possible number of data words since the number of words will vary for different bin lists.

```
{dbase}/lev0/ffffff01_24h.overview.fits
{dbase}/lev0/ffffff01_24h.record.rdb
{dbase}/lev0/ffffff01_24h.data.fits                or 0000.fits??
```

For limb figure data, it is also necessary to save the bin list which describes how to recreate the limb figure. The bin list will either be regenerated from a subroutine or read from a database. A single bin list will be written out to the dataset for each DPC.

```
{dbase}/lev0/ffffff01_24h.overview.fits
{dbase}/lev0/ffffff01_24h.record.rdb
{dbase}/lev0/ffffff01_24h.data.fits
{dbase}/lev0/ffffff01_24h.binlist_ffffff01.fits
{dbase}/lev0/ffffff01_24h.binlist_ffffff02.fits
{dbase}/lev0/ffffff01_24h.binlist_ffffff03.fits
```

Size: This data is decompressed and certain datasets are re-embedded into rectangular data arrays. The size is TBD - probably a factor of 2-4 times more than the input? Which means 200 meg per day?

6 Housekeeping Level 0 Processing

TODO - talk about creating an empty shell 24 hour file and having the real-time EGSE writing into the proper slot, and how when the playback data is available from FTP it is copied over into that file...

7 SCI160K Level 1 Processing

7.1 Purpose

Level 1 processing takes the level 0 product and performs calibration corrections. It will:

- Remove the instrument effects
  - Flat fielding
  - ??
- Apply lookup table corrections
- Convert the data numbers (DN) send down in the telemetry into calibrated engineering units.

For every level 0 series name (data product code) there is a corresponding level 1 series name. At level 1.5 the data products are gathered and a series name which is not simply the data product code will be created.

Mode	*Telemetry*	*Frequency*
Campaign	160K (VC3)	8 hours a day, 10 of 12 months
Dynamics	160K (VC3)	24 hours a day, 2 of 12 months
Magnetic	160K (VC2)	5 minutes a day, 1 to 3 times a day, whole mission
Structure	5K	24 hours a day, whole mission

7.2 Data Header Contents

Level 1 .record file will contain:

- Full copy of Level 0 FITS header
- Date/Time/ProgVer/ProgName of level 1 processing
- Full data set name(s) of calibration images used
- The sun center pixel address and roll used for min/max/average/stddev
  - The x,y pixel address
  - The predicted S/C roll (?)
  - The box size or radius used for the sub-image (90
  - The size of the sun (?)
  - The technique and version used
- Indication of the data quality and values
  - The min/max/average/stddev
  - TBD - Statistical information

7.3 Processing Steps

There will be only three separate programs to perform the calibration, one for dopplergram images, one for magnetogram images, and one for intensity images. A set of parameters will tell if it is OBS or CAL mode, full disk pixels or high resolution pixels, IP pixel summed, or sub-portion extracted.

7.3.1 STEP 1 Determine Which Classes of Data Product Exist for an Hour

By looking at the ??? (master image log) determine which DPCs exist for the hour. By using the table which maps DPCs (level 0 series names) to data product "classes" (dop, mag or intensity), determine which calibration programs have to be run, and build the map files.

7.3.2 STEP 2 For Each DPC Class, Peform the Calibration Corrections

Input: The calibration DPC class server is run for each dpc class which exists in the level 0 data.

```
!setenv mdi wd:{dbase}/{level}/{series}; bn:{series}.{#%06d#series}
p=calib_dop h=fault.Stanford.EDU d=1 a=1 \
    in           =prog:mdi, level:lev0,    series:ffffff01_01h[99],      sel:[] \
    out          =prog:mdi, level:lev1,    series:fd_dop_01h[99],        sel:[] \
    dpc          =ffffff01 \
    dpc_table    =prog:mdi, level: ancill, series: dpc_table,            sel:[] \
    time_corr    =prog:mdi, level: ancill, series: time_corr.table,      sel:[] \
    ref2dp_table =prog:mdi, level: ancill, series: ref2dp_time.table,    sel:[] \
    hk_table     =prog:mdi, level: ancill, series: house_keeping.conv,   sel:[] \
    roll_corr    =prog:mdi, level: ancill, series: roll_corr.table,      sel:[] \
    c_sun_cen    =prog:mdi, level: ancill, series: coarse_sun_cen.table, sel:[] \
    calib        =prog:mdi, level: calib,  series: fd_flat_field,        sel:[] \

p=calib_mag h=fault.Stanford.EDU d=1 a=1 \
    in           =prog:mdi, level:lev0,    series:ffffff02_01h[99],      sel:[]; \
                  prog:mdi, level:lev0,    series:8fffff02_01h[99],      sel:[] \
    out          =prog:mdi, level:lev1,    series:fd_mag_01h[99],        sel:[] \
    dpc          =ffffff02 \
    dpc_table    =prog:mdi, level: ancill, series: dpc_table,            sel:[] \
    time_corr    =prog:mdi, level: ancill, series: time_corr.table,      sel:[] \
    ref2dp_table =prog:mdi, level: ancill, series: ref2dp_time.table,    sel:[] \
    hk_table     =prog:mdi, level: ancill, series: house_keeping.conv,   sel:[] \
    roll_corr    =prog:mdi, level: ancill, series: roll_corr.table,      sel:[] \
    c_sun_cen    =prog:mdi, level: ancill, series: coarse_sun_cen.table, sel:[] \
    calib        =prog:mdi, level: calib,  series: fd_flat_field,        sel:[] \
```

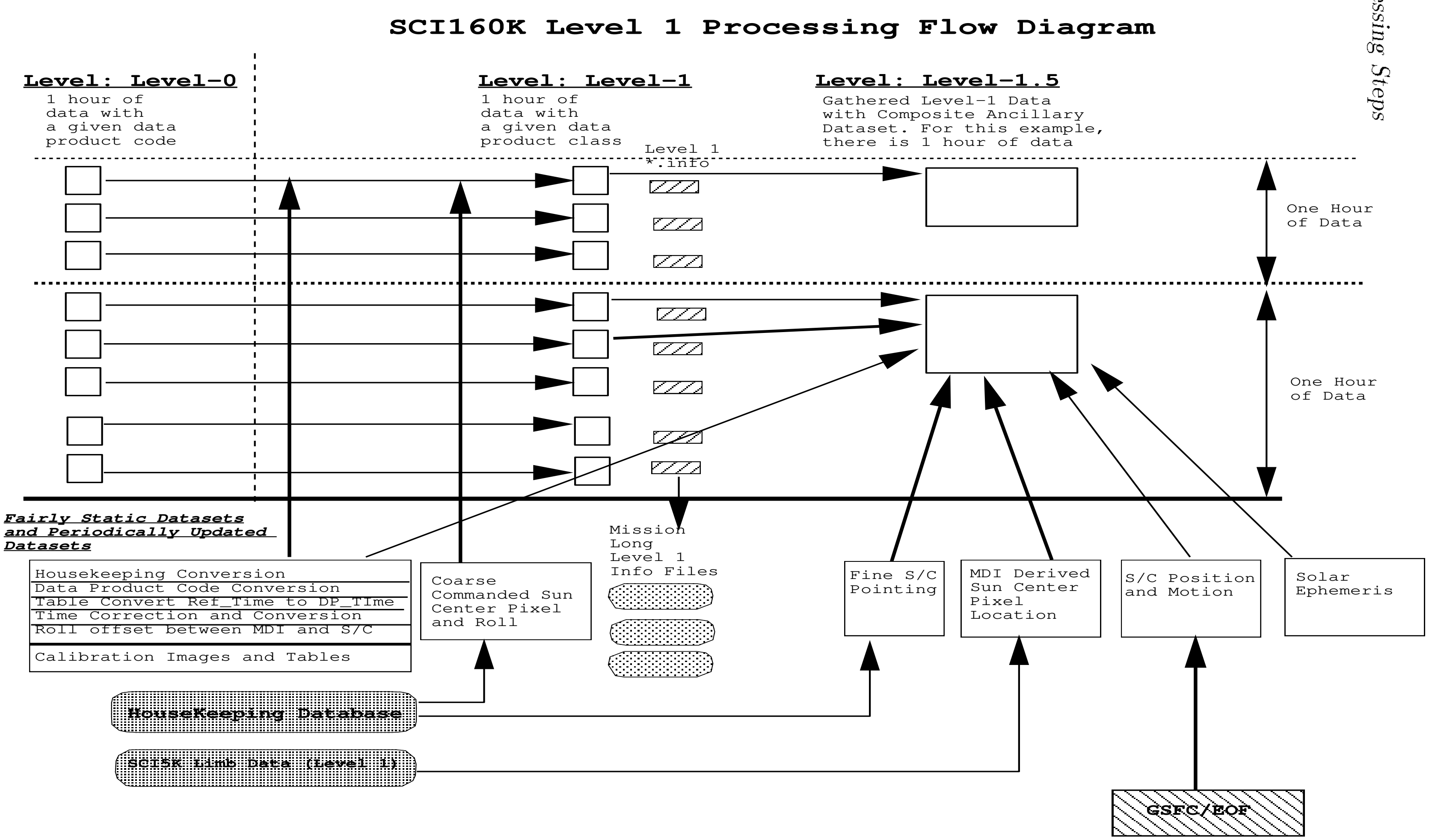

Figure 9: Flow and Steps for Level 1 Processing

```
p=calib_int h=fault.Stanford.EDU d=1 a=1 \
    in          =prog:mdi, level:lev0,    series:ffffff03_01h[99],     sel:[] \
    out         =prog:mdi, level:lev1,    series:fd_int_01h[99],       sel:[] \
    dpc         =ffffff03 \
    dpc_table   =prog:mdi, level: ancill, series: dpc_table,           sel:[] \
    time_corr   =prog:mdi, level: ancill, series: time_corr.table,     sel:[] \
    ref2dp_table =prog:mdi, level: ancill, series: ref2dp_time.table,  sel:[] \
    hk_table    =prog:mdi, level: ancill, series: house_keeping.conv,  sel:[] \
    roll_corr   =prog:mdi, level: ancill, series: roll_corr.table,     sel:[] \
    c_sun_cen   =prog:mdi, level: ancill, series: coarse_sun_cen.table, sel:[] \
    calib       =prog:mdi, level: calib,  series: fd_flat_field,       sel:[] \
```

For a given data product class, for each image do the following:

1. Read a single level 0 image
2. Determine calibration images to use based on the DPC. Select calibration image(s) based on date/time (and IP lookup table used?)
    - additive correction image
    - multiplicative correction image
    - lookup correction table (based on image intensity)
3. Make adjustments/corrections to the calibration images due to:
    - Michelson temperatures
    - Date/time (drift in some correction factor)
    - Pointing (?) - probably not
    - Size of the sun - NO
    - Roll (?)
    - Other parameters?
4. Fit the limb and disk (done in this loop or later ??)
    - Derive the x,y pixel of sun center
5. For Dopplergrams
    - Fit the gross gradient to a plane - gives the rough roll of the image – there should be an east-west gradient due to solar rotation.
    - This value is a data quality item?
6. Quality Flags
    - Avg/min/max/dev of the image
    - Some manually entered quality flag (?)
    - OK-to-use flag (bits signify which programs can use the data) (?)
7. Write the single image out to a FITS file

Output: For every level 0 image, there is a level 1 image output.

```
{dbase}/lev0/fd_dop_01h/000099/nnnn.fits   (or 0.cdf)
{dbase}/lev0/fd_dop_01h/000099/calib_dop.info
```

SAMPLE ????? .INFO file (STEP #2) [One file per hour per DPC]								
SAMPLE ?????_LOG database (STEP #3) [One table per mission per DPC / One entry per image]								
Dataset Name (?)	RefTime	DPC	SeqID	HK Pkt	ImgMin	ImgMax	ImgAvg	ImgDev
???	1146026431	ffffff01	ff778833	234245	100	200	50	20
???	1146026491	ffffff01	ff778833	234249	110	200	50	30
???	1146026551	ffffff01	ff778833	234366	120	200	50	25

7.3.3 STEP 3 Update the Mission Long Level 1 Info Database

For certain data series, an oracle table exists which holds information for the whole mission period. The oracle table contents are a subset of the `.record` file so a generic module can update the oracle table.

7.3.4 STEP 4 Fit the Solar Limb for Certain Data Products

Input: By making a database query (?) determine which data product classes
Using the
Fit the limb and disk

- Derive the x,y pixel of sun center
- Record the RMS error

8 Level 0 and Level 1 Processing at EOF/GSFC

TODO - talk about temporary pipeline execution/databases

9 SCI160K Level 1.5 Gathering

9.1 Purpose

The level "1.5" was created to facilitate the handling of the data for the level 2.0 programs. Almost all level 1.5 data series are conforming datasets. See section **??** for details on the definition of a conforming dataset. Level 1.5 processing will typically:

- Make the data evenly spaced: It takes the input data and creates a data series where the separation in records is known. For level 0 and 1.0 the dataset have a record for each image as it is received. As an example, for level 1.5 the output will have a record for every minute.
- Sub-sample the data: This gather level allows the simple creation of data series which are a reduced sampling of the data. For example, a data series which is one dopplergram per hour can be created.
- Group the input data series: Since the level 1 series names are simply the data product code, the gather level will allow the specification of a list of input data series which can be grouped together to make the output series. For example, there could be several DPCs which are "full disk dopplergrams".
- Gather information from ancillary databases: It will be very common to need to gather information from several ancillary databases, and this step allows all of that information to be assembled and to be available directly into the level 2.0 program as input.
- Allow multiple variables: A level 1.5 series can have more than one variable. For example, it could have a dopplergram variable from one set of input series, and a magnetogram variable from anther set of input series.

9.2 Data Header Contents

Level 1 `.record` file will contain:

- Full copy of Level 1 header
- Date/Time/ProgVer/ProgName of level 1.5 processing
- OPTIONALLY solar ephemeris information (B-angle, P-angle, solar radius)
  - The B-angle
  - The p-angle
- OPTIONALLY Ephemeris derived S/C position and motion
  - x,y,z,x-dot,y-dot,y-dot in heliocentric ecliptic coordinate system
  - x,y,z,x-dot,y-dot,y-dot in geocentric solar coordinate system
  - The apparent angular solar diameter
  - Code value for algorithm version used
- OPTIONALLY MDI Derived Pixel Address of Sun Center
  - x,y pixel address

  - Technique and source of data
  - Quality flag on values
- OPTIONALLY Pointing Stability Information
  - 2 minute drift or STDDEV of S/C fine sun pointing
  - ISS stability information (?)

9.3 Processing Steps

Because of the way that a data series is defined by an MML file, the processing steps and modules run to create a level 1.5 dataset can be almost anything. See the sample MML file in section 3.4.2 on page 24. The steps shown below are for general example.

9.3.1 STEP 1 Initialize the output `.record` file

The `.record` file needs to be initialized to have the list of times desired. For an equally spaced time series, the list of desired dates and times is created based on the T_STEP, BLOCK, T_EPOCH, and SER_NUM values.

9.3.2 STEP 2 Copy a subset of the input `.record` keywords

A subset of the input dataset(s) keywords will be copied into the output `.record` file. This step and the previous step of initializing the output `.record` file can typically be done in a single module call. Convert HK information into engineering units (volts, degrees, ...)

9.3.3 STEP 3 Gather a set of keyword values from various sources

For each different ancillary data series, a list of keywords can be specified which should be extracted from the datasets. These values are appended to the output `.record` file. The solar ephemeris information can be gathered during this stage too. Typically there will be a separate module call for each ancillary data series.

9.3.4 STEP 4 Copy the data to the output dataset

The data records (images) need to be copied to the output dataset. This will typically be done by either spawning a *copy* command or reading the image, and then writing it out.

9.3.5 STEP 5 Create the overview file

The overview file needs to be created. Information from the `.record` file is assembled, plus information from the MML. See the sample overview file in section 3.4.4 on page 28.

9.3.6 STEP 6 Optionally update an oracle table

For certain data series, an oracle table exists which holds information for the whole mission period. The oracle table contents are a subset of the `.record` file so a generic module can update the oracle table.

9.4 Sample files for a Level 1.5 Dataset

Example: Files in a Gather Dataset with Level 1.5 Series Name "fd_dop_01h"

```
fd_dop_01h.003600.overview.fits                 - Overview info
fd_dop_01h.003600.ancil.rdb                     - Ancillary keyword info
fd_dop_01h.003600.record.rdb                    - Copy of the level 1 info
fd_dop_01h.003600.0001.fits
fd_dop_01h.003600.0002.fits
fd_dop_01h.003600.0003.fits
fd_dop_01h.003600.nnnn.fits
fd_dop_01h.003600.info.rdb                      - Info file (subset of .record.rdb)
```

Example: Files in a Gather Dataset with Level 1.5 Series Name "fd_dop_mag_01h" with Two Variables

```
fd_dop_mag_01h.003600.overview.fits             - Overview info

fd_dop_mag_01h.003600.dop.record.fits           - Copy of the level 1 info for DOP vari
fd_dop_mag_01h.003600.dop.0001.fits
fd_dop_mag_01h.003600.dop.0002.fits
fd_dop_mag_01h.003600.dop.0003.fits
fd_dop_mag_01h.003600.dop.nnnn.fits
fd_dop_mag_01h.003600.dop.info.rdb              - Info file (subset of .record.rdb)

fd_dop_mag_01h.003600.mag.record.fits           - Copy of the level 1 info for MAG vari
fd_dop_mag_01h.003600.mag.0001.fits
fd_dop_mag_01h.003600.mag.0002.fits
fd_dop_mag_01h.003600.mag.0003.fits
fd_dop_mag_01h.003600.mag.nnnn.fits
fd_dop_mag_01h.003600.mag.info.rdb              - Info file (subset of .record.rdb)

^^^^^^^^^^^^^^                              Series name
               ^^^^^^                       Series Number
                      ^^^                   Optional variable name
                          ^^^^^^^^^^        Type of data
```

10 Level 2 Processing: l-nu Pipe

10.1 Purpose

TODO - description

10.2 Processing Steps

10.2.1 STEP 1 Remap

Go from calibrated lev1 files to lev2 remapped files. One output image for each input image.

```
!setenv mdi wd:{dbase}/{level}/{series};bn:{series}.{#%06d#series}
p=v2helio h=fault d=1 a=1\
   in= prog:mdi, level:lev1, series:fd_dop_01h[99], sel:[],fmt:%04d
   out=prog:mdi, level:lev2, series:V_heliomap[99], sel:[],fmt:%04d
```

Processing: Images for each hour are remapped from CCD pixel coordinates to heliographic coordinates, i.e. longitude and sin(latitude). This strategy module includes a velocity correction.

Output:

```
{dbase}/lev2/V_heliomap/V_heliomap.000099.nnnn.fits
```

No log file necessary.

Assumptions: Record numbers correspond to minutes within the hour so that a missing record number corresponds to a missing image. The output will be archived at least during initial tests.

10.2.2 STEP 2 FFT in Longitude

Go from remapped images to ”mlat images” One output image for each input image.

```
!setenv mdi {dbase}/{level}/{series};bn:{series}.{#%06d#series}
p=helio2mlat h=fault a=1\
    in= prog:mdi, level:lev2, series:V_heliomap[99], sel:[],fmt:%04d \
    out=prog:mdi, level:lev2, series:V_mlat[99],     sel:[],fmt:%04d \
    LMAX=249
```

Processing: For each image, performs fft on map rows. Performs transpose. Saves complex values for each (sin)latitude up to maximum m of LMAX.

Output:

```
{dbase}/lev2/V_mlat/V_mlat.000099.nnnn.fits
```

No log file necessary.

Assumptions: The default value for LMAX in the production system is 750. The maximum possible value for LMAX to be used in indexing tiles is 1500.

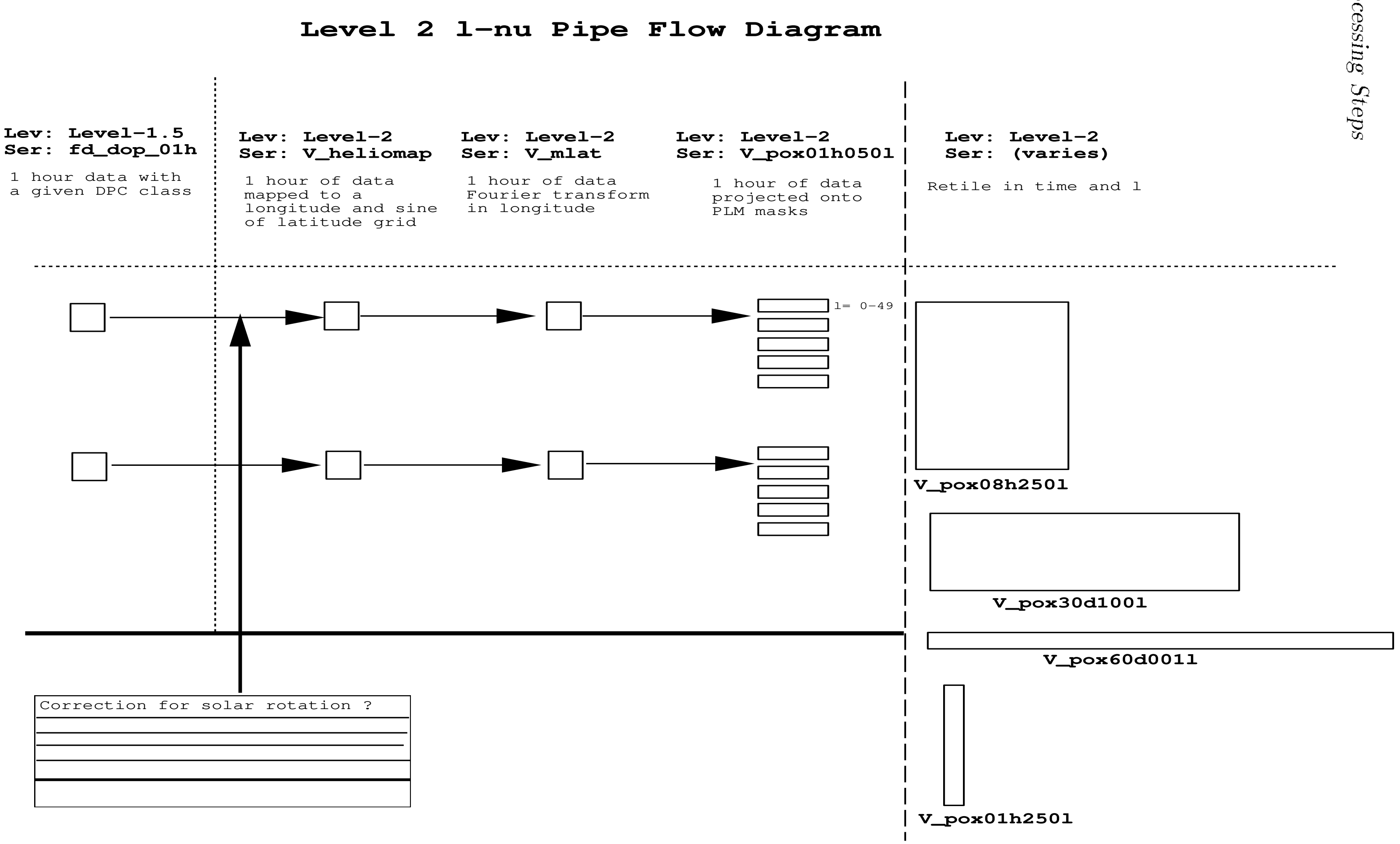

Figure 10: Flow and Steps for l-nu Level 2 Processing

10.2.3 STEP 3 Dotproduct

Convert a series of images for one hour into mode amplitude matrices/tiles. Each output tile is a dataset. This is to be consistent with the general retiling process. There is more description of tile names and indices in the section on general retiling below.

Note that this assumes that pe is capable of expanding output data names and creating multiple output directories. This is symetrical with the handling of multiple input datasets. Just as multiple input datasets result in keylist entries of, for example, in_0_wd, in_1_wd ..., so will multiple output datasets result in keylist entries of out_0_wd, out_1_wd...

```
 !setenv mdi wd:{dbase}/{level}/{series};bn:{series}.{#%06d#series}
 !setenv mdi_tile wd:{dbase}/{level}/{series}/{#06d#series};bn:{series}.{#06d#series}
p=qdotprod h=fault a=1\
    in= prog:mdi, level:lev2, series:V_mlat[99],            sel:[],fmt:%04d
    out=prog:mdi_tile, level:lev2, series:V_pox01h50l[2970-2974], sel:[],fmt:%02d
    LMAX=249
```

Output:

```
{dbase}/lev2/V_pox01h050l/002970/V_pox01h050l.002970.00.fits
{dbase}/lev2/V_pox01h050l/002971/V_pox01h050l.002971.00.fits
{dbase}/lev2/V_pox01h050l/002972/V_pox01h050l.002972.00.fits
{dbase}/lev2/V_pox01h050l/002973/V_pox01h050l.002973.00.fits
{dbase}/lev2/V_pox01h050l/002974/V_pox01h050l.002974.00.fits
{dbase}/lev2/V_pox01h050l/002970/V_pox01h050l.002970.info
```

The .info file containts the tile name, tile index, start time, time width, start l, l width, modes

Questions: Was there an objection to storing output tiles in separate directories? What was it?

10.2.4 STEP 4 Retiling

When enough mode amplitude data has been produced by qdotprod it can be retiled into datasets which contain amplitudes for longer periods of time. General retiling selects data from a set of input tiles and produces one output tile. This could be generalized to produce a set of output tiles but this is simpler and is probably adequate.

It is assumed that tiles will be produced in a small set of fixed sizes. The series name for the tile will reflect its size in time and in l (or mode). The idea is to yield tiles which are no larger (in bytes) than some fixed maximum. Some possible series names (and therefore tile sizes) are:

Series Name	*Description*		
V_pox01h050l	1 hour	by	50 l
V_pox08h250l	8 hours	by	250 l
V_pox01d050l	1 day	by	50 l
V_pox03d015l	3 days	by	15 l
V_pox10d010l	10 days	by	10 l
V_pox60d001l	60 days	by	1 l

where V_pox01h050l is produced by qdotprod. To keep tile sizes in bytes more even there may be series names like V_pox08h1000m which is 8 hours by approximately 1000m. The size is approximate because the data for an l value would not be split across tiles. See Phil's notes on tiling for more detail.

The tiles for each tile size will be indexed so that either the mapmaker/scheduler or the strategy module can, knowing the starting time and l (or mode) for a tile, determine the index. In the case of tile sizes based on l, this could be a simple formula:

$$tile\_index = (1500/l\_width) * (start\_time/time\_width) + start\_l/l\_width$$

where 1500 is a maximum l and time and l indices start at 0. That is, the tiles would be ordered by increasing time and increasing l with l varying most rapidly.

So to produce a tile for the 8 hours beginning at hour 56 with l's from 250 to 499 the map file would be:

```
!setenv mdi_tile wd:{dbase}/{level}/{series}/{#06#series};bn:{series}.{#06d#series}
p=retile h=fault a=1\
    in=prog:mdi_tile,level:lev2, series:V_pox01h050l[1625-1629,1655-1659,\
        1685-1689,1715-1719,1745-1749,1775-1779,1805-1809,1835-1839],
        sel:[],fmt:%02d
    out=prog:mdi_tile,level:lev2,series:V_pox08h250l[43],sel:[],fmt:%02d
```

Output:

```
{dbase}/lev2/V_pox08h250l/000043/V_pox08h250l.000043.00.fits
{dbase}/lev2/V_pox08d250l/000043/V_pox08d250l.000043.info
```

The .info file containts the tile name, tile index, start time, time width, start l, l width, modes

Or to produce a tile for the fifth day (day 4 indexed from 0) with l's from 100 to 149 the map file would be:

```
!setenv mdi_tile wd:{dbase}/{level}/{series}/{series#};bn:{series}.{#06d#series}
p=retile h=fault a=1\
        in=prog:mdi_tile,level:lev2,series:V_pox08h250l[72,78,84],sel:[],fmt:%02d
        out=prog:mdi_tile,level:lev2,series:V_pox01d050l[122],sel:[],fmt:%02d
```

Output:

```
{dbase}/lev2/V_pox01d050l/000122/V_pox01d050l.000122.00.fits
{dbase}/lev2/V_pox01d050l/000122/V_pox01d050l.000122.info
```

Assumptions: Although it would be possible to retile data from any size tiles to any other if the source datasets "covered" the destination datasets, we will make the simplifying assumptions that we will retile only from shorter lengths in time to longer lengths in time and that tiles will be defined with common boundaries so that we will not deal with partial data slices. Tile time boundaries will be multiples of time width and tile l boundaries will be multiples of l width.

Comments: This scheme makes it possible to avoid impossibly large datasets but it is the responsibility of the scheduler/mapmaker to do this.

To do:

- Define a strategy module to transfer tile log file info to a database table.
- Define the database table.
- Make a table of tile sizes (in bytes) corresponding to our fixed set of tile sizes in time and l/m.
- Define what needs to be in a tile log file to make sense even when the tiles are defined in terms of time and m.

11 Importing and Exporting Data from DSDS

TODO - fill in

12 Ancillary Database Processing

List of Ancillary Data		
Prog	*Series*	*Description*
mdi	hk_conv	Housekeeping conversion database
mdi	dpc_table	Data Product Code (DPC) conversion database
mdi	ip_table	Copies of the IP tables
mdi	dser_index	Data series name index table (?)
mdi	reftim2dptim	Table to convert RefTime to DPTime
mdi	mdi_roll_offset	Roll offset between MDI and S/C
soho	sc_time_cor	Table to correct gross RefTime errors
soho	orbit_sol	Orbital solution database
solar	ephem	Solar ephmeris database
soho	sc_pos_mot	S/C position and motion database
soho	sc_fine_pnt_roll	Fine S/C pointing and roll database
soho	sc_cmd_pnt_roll	Commanded S/C pointing and roll database (coarse)
mdi	fine_sun_center	MDI derived fine sun center pixel location database
mdi	cmd_sun_center	MDI commanded sun center pixel location database
mdi_sc	cmd_history	MDI and S/C command history log
mdi	science_ops	Ground based sicence and operations database and log
mdi_sc	hk_summary	MDI and S/C housekeeping summary database

12.1 Housekeeping Conversion Database

- Gives conversion coefficents from digital values to voltages and temperatures
- Conversions from leg offset values to sun center pixel coord

Input Source: Manually, by hand whenever a new set of coefficients are derived.

12.2 Data Product Code (DPC) Conversion Database

- Gives image size (NX and NY)
- Gives the pixel summing factor
- Gives which cropping table to use
- Gives which lookup tables were used

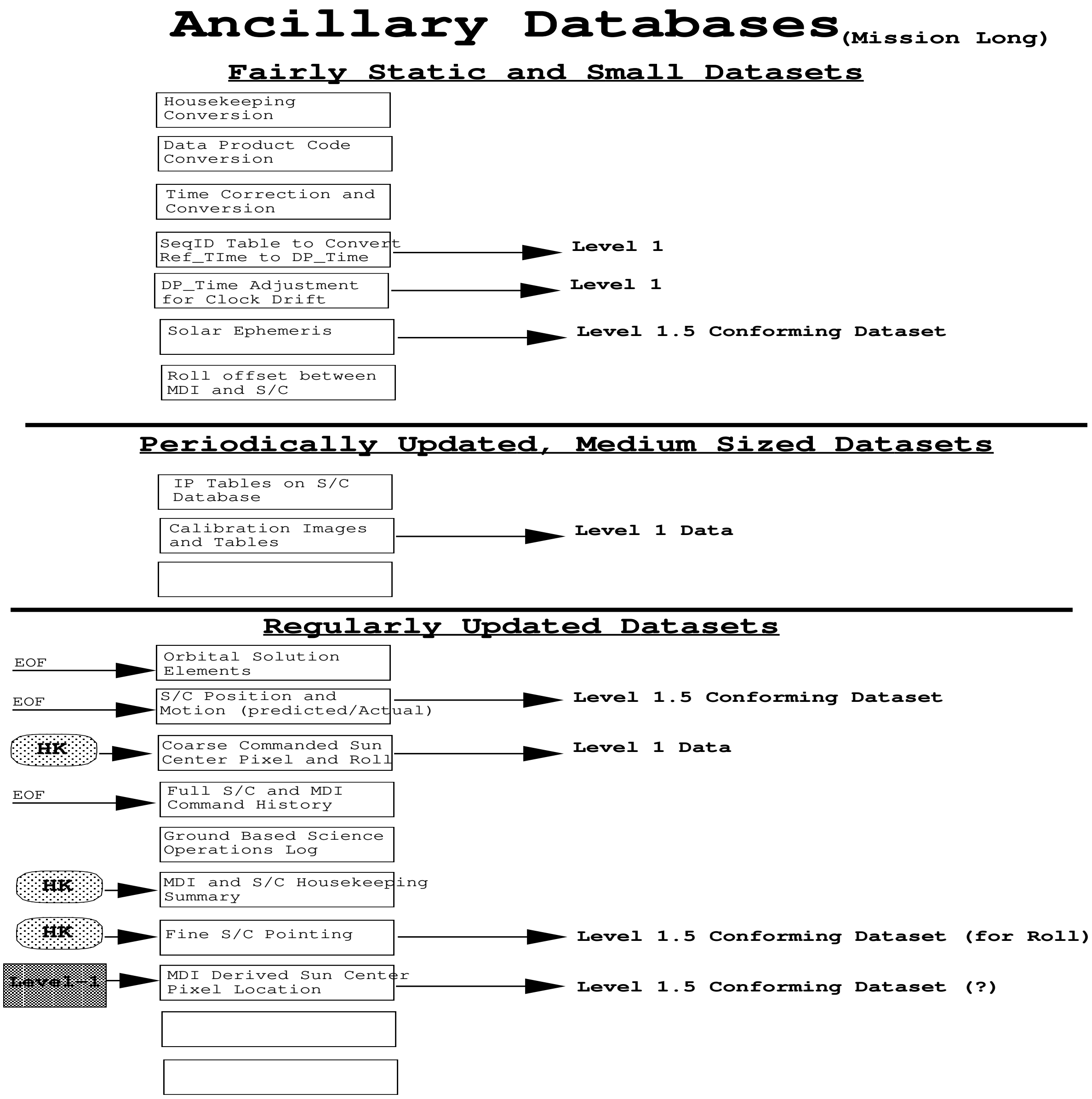

Figure 11: Overview of Ancillary Data

- Gives a mnemonic
- Level 1 series name associated with this DPC

TODO - Add column which has a value, and which bits signify if a given program can be run on that set (ie: solar disk/limb fit)

Input Source: Manually, by hand whenever a new DPC is defined.

12.2.1 Mapping of Level 0 Series Names to Level 1 Series Names

Contained within that table is the information on how to match up level 0 and level 1 series names.

- For a given level 1 series, there can be several different level 0 DPCs that go into that go into that series
- Example:

LEV1 SERIES	*LEV0 SERIES*
HR_DOP_01H	FFFFFF01_01H
FD_DOP_01H	FFFFFF02_01H
FD_DOP_01H	8FFFFF02_01H
HR_CAL_INT_01H	FFFF8801_01H

Input Source: Manually, by hand whenever a new DPC is defined.

12.3 Dataset Name Index Table

A table is used to provide an index so that a single integer value can specify a **prog:**, **level:**, and **series:**. The bits for the dataset name index are used as follows:

```
xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
xxxxxxxx                              = Specifies the PROG
         xxxxxxxx                     = Specifies the LEVEL
                  xxxxxxxx xxxxxxxx   = Specifies the SERIES
```

Index	*Prog*	*Level*	*Series*	*Series Field*	*Description*	*Orc Tbl?*
?	mdi	ancil	hk_conv	none?	Housekeeping Conversion Table	No
?	mdi	ancil	dpc_table	none?	Data Product Code Conversion Table	No
?	mdi	ancil	dset_name	none?	Dataset Name Index Table	No
?	mdi	ancil	corr_time	none?	Table to Correct/Convert Ref Time	No
?	mdi	ancil	ref2dp_time	none?	Table to Convert Ref to DP Time	No
?	mdi	ancil	roll_corr	none?	Roll Offset between MDI and S/C	No
?	mdi	calib	ip_tables	type?	IP Tables Used by MDI	No
?	mdi	calib	dop_cor	none?	Doppler Calibration Images/Tables	
?	mdi	calib	mag_cor	none?	Magnetogram Calibration Images/Tables	
?	mdi	calib	int_cor	none?	Intensity Calibration Images/Tables	
?	mdi	ops	sci_ops_plan	day?	Ground Based Science and Ops Log	No
?	soho_data	ancil	orb_elem	none?	Orbital Element Database	No
?	soho_data	ancil	sc_pos	2wks?	S/C Position and Motion Database	No
?	soho_data	ancil	sc_cmd_hist	day?	Full S/C and MDI Command History	No
?	mdi	raw	sci160k	tape#	Raw highrate telemetry from DSN	Yes
?	mdi	raw	sci5k	day	Raw med rate telemetry from CD-ROM	
?	mdi	raw	mdihk	day	Raw housekeeping from CD-RO	
?	mdi	raw	schk1	day	Raw S/C HK Packet #1 from CD-ROM	
?	mdi	raw	schk2	day	Raw S/C HK Packet #2 from CD-ROM	
?	mdi	raw	schk3	day	Raw S/C HK Packet #3 from CD-ROM	
?	mdi	merge	sci160k	hour	Telemetry merged & broken 1 hr chunks	
?	mdi_info	merge	sci160k	none?	Merged Telemetry Info (mission long)	Yes?
?	mdi	lev0	ffffff01_01h	hour	One hour of Data Product ffffff01	
?	mdi	lev0	ffffff02_01h	hour	One hour of Data Product ffffff02	
?	mdi	lev0	ffffff03_01h	hour	One hour of Data Product ffffff03	
?	mdi_info	lev0	???	none?	Level 0 Info (Mission Long)	Yes?
?	mdi	lev0b	csunc_cmd	none?	Coarse Commanded Sun Center and Roll	
?	mdi	lev0b	mdihk_sum	day?	MDI and S/C housekeeping Summary	
?	mdi	lev0b	sc_fine_pnt	day	Fine S/C Pointing Database	

Index	*Prog*	*Level*	*Series*	*Series Field*	*Description*	*Orc Tbl?*
?	mdi	lev1	fd_dop_01h	hour	Full Disk Dopplergrams	
?	mdi	lev1	fde_dop_01h	hour	Full Disk Dop, a Portion Extracted	
?	mdi	lev1	fdb_dop_01h	hour	Full Disk Dop, IP Pixel Summed	
?	mdi	lev1	fd_cal_dop_01h	hour	Full Disk Dopplergrams in CAL Mode	
?	mdi	lev1	fde_cal_dop_01h	hour	Full Disk Dop, CAL Mode, Extracted	
?	mdi	lev1	fdb_cal_dop_01h	hour	Full Disk Dop, CAL Mode, Summed	
?	mdi	lev1	hr_dop_01h	hour	High Res Dopplergrams	
?	mdi	lev1	hre_dop_01h	hour	High Res Dop, a Portion Extracted	
?	mdi	lev1	hrb_dop_01h	hour	High Res Dop, IP Pixel Summed	
?	mdi	lev1	hr_cal_dop_01h	hour	High Res Dopplergrams in CAL Mode	
?	mdi	lev1	hre_cal_dop_01h	hour	High Res Dop, CAL Mode, Extracted	
?	mdi	lev1	hrb_cal_dop_01h	hour	High Res Dop, CAL Mode, Summed	
?	mdi_info	lev1	???	none?	Level 1 Info (Mission Long)	Yes?
?	mdi	lev1	fd_mag_01h	hour	Full disk magnetograms (all of the same "dop" combinations)	
?	mdi	lev1	fd_int_01h	hour	Full disk magnetograms (all of the same "dop" combinations)	
?	mdi	lev1	limb_fig	day	Structures Program Calibrated Limb Fig	
?	mdi	lev1b	sunc_fd_int	hour	MDI Derived Sun Center from FD Int Images	
?	mdi	lev1b	sunc_5k_limb	hour	MDI Derived Sun Center from Limb Figure	
?	mdi	ancil	composite	hour	Ancillary Info from Several Datasets	No
?	mdi	lev2	V_heliomap_01h	???	???	
?	mdi	lev2	V_mlat_01h	???	???	
?	mdi	lev2	V_pox01h050	???	???	
?	mdi	lev2	V_pox08h250L	???	???	
?	mdi	lev2		???	???	
?	mdi	lev2		???	???	

TODO - If there is a Oracle database table with a summary of the dataset (a .info type file for example), it will have a name that uses the level and series. For example, there is a `merge_sci160k_info` table and there is also a `raw_sci160k_info` table (TODO - JIM OK?).

Input Source: Manually, by hand whenever a new dataset is defined.

12.4 Table to Convert Ref_Time to DP_Time

A correction must be applied to the Ref_Time value to get the data product time (DP_Time) which is the time for the center of the series of images taken which produced the data product. It is a lookup table using the SeqID as the input.

SeqID	*DPC (?)*	*Offset*
?	?	?
?	?	?
?	?	?

Input Source: Manually, by hand whenever a new SeqID or DPC (?) is defined.

12.4.1 Gross Ref_Time Error Corrections (???)

Whenever a large error in the Ref_Time is found in the raw telemetry data file on the DSN tape, an entry is made in this database so that a record is kept. This is done so that if the data is processed again starting from the DSN tape, the Ref_Time patching routine knows what to do.

UT Start Time	*UT End Time*	*Pkt_Seq#*	*Bad Ref_Time*	*Offset to Correct*
?	?	?	?	?
?	?	?	?	?
?	?	?	?	?

12.5 IP Tables Database

- A copy of the flat field image loaded in the IP
- A copy of the lookup tables
- A copy of the cropping table

Input Source: Manually copy from a separate file

12.6 Time Correction and Conversion Database

- Correction factors will be derived and available for time periods in the past
- Save each solution with a serial number. Track the serial number of the conversion technique used.
- Time errors can be:
  - Due to glitch in MDI (cosmic ray) - the value says bad until the DEP is reset.
  - Due to corruption in the MDI telemety (bad in one image)
  - Due to drift or offset in MDI clock relative to S/C clock
  - Due to drift in the S/C clock

- The errors can be gross large errors (fairly easy to spot) or small "fine tuning" adjustments

Input Source: Manually, by hand

12.7 Solar Ephemeris Database

- b-angle
- p-angle
- solar radius
- Save each solution with a serial number. Track the serial number of the calculation/ephemeris used.

Input Source: Existing programs (?)

12.8 Orbital Solution Database

- orbital elements for a given epoch
- ??

Input Source: FTPed automatically from EOF

12.9 S/C Position and Motion Database

TODO - Insert portions of St Cyr memo – table with a value per 10 minutes – 3 coordinate systems - heliocentric ecliptic – file per two weeks – cdf

The spacecraft position the motion relative to the sun derived from orbital elements solutions.

- x,y,z,x-dot,y-dot,y-dot
- Gives S/C motion relative to the sun (velocity vector)
- Gives distance from the sun
- Versions
  1. Predicted solution for the future
  2. Actual solution for the past with updated orbital elements
- Actual x,y,z values are not saved in the database, just the orbital elements. Given a date/time, a routine calculates the value

Input Source: Orbital solution database

12.10 Roll Offset between MDI and S/C Database

Derived by comparing MDI images with ground based images

- Roll offset between MDI and S/C

- Function of temperature (?)
- Function of time (?)
- Only a handful of values

Input Source: Manually, by hand

12.11 Full S/C and MDI Operations Command History

TODO - Is this a straight copy of the CDF datasets provided by EOF ?

12.12 Ground Based Science and Operations Database and Log

- Sequence name/number/serial number ...
- IP lookup table version numbers
- Dates/times of when flat field images are changed in the IP
- Names of sequences and their purposes
- ??

Input Source: Manually, by hand (?)

12.13 Coarse Commanded Sun Center Pixel Location Database

Intended "commanded" approximate sun center pixel location using MDI leg positions, ISS commanded offset, and S/C commanded pointing/roll. Derived from commanded values (read from housekeeping packets)

- rough value good to only to an arcminute (?)
- x,y of S/C pointing relative to sun center
- S/C roll (0 versus 90 degrees)
- MDI leg positions
- MDI ISS offset values commanded
- Code for algorithm used (commanded rough value)
- Algorithm version number

SOURCE: Database derived from S/C and MDI housekeeping

12.14 MDI and S/C Housekeeping Summary Database

- Some temperatures (?)
- Some voltages (?)
- IP and DEP software version numbers (?)
- ISS feedback loop value (MDI pointing stability)
  (STDDEV of values over 2 minutes (?))
- IP lookup table version numbers
- S/C panel temperature

12.15 Fine S/C Pointing Database

This data is not very useful for MDI since it has an image stabilization system (ISS) which keeps the location of the solar image centered at a given pixel location. This database will contain a detailed time history of the S/C jitter and the S/C roll, of which the roll will be useful to analyzing MDI data.

Derived from S/C housekeeping packets

- x,y of S/C pointing relative to sun center (from fine sun sensor)
- S/C roll (from IRU)
- Gives indication of S/C pointing stability
- Save a position per second (?)

Input Source: S/C housekeeping

TODO - Is this a straight copy of the CDF datasets provided by EOF ? Is this the raw sun sensor and IRU values or is it a processed and reduced X,Y,and Z.

12.16 MDI Derived Fine Sun Center Pixel Location Database

The pixel address for the center of the solar image can be derived from the MDI intensity images. These values are derived by fitting the MDI image to a function which takes the limb darkening into account. The SCI5K limb 12 minute average images can also be used to derive the pixel address for the center of the sun.

- The x,y pixel value of the sun center
- Code for algorithm used (commanded rough value)
- Algorithm version number
- Versions:
    1. Fixed location on the CCD (512,512)
    2. Intended "commanded" pointing (from leg position and ISS offset values, and S/C commanded pointing)
    3. MDI derived from full sun filtergrams
    4. MDI derived from limb data (12 minute averages)
    5. MDI derived from full sun dopplergram (?)
    6. MDI derived from full sun magnetogram (?)
- A set of values per image per image type (DPC)

Input Source: From Level 1 Processing of Level 0 data

13 Oracle Tables

13.1 Dataseries Naming/Definition Table

There is an entry per data series to allow the data series to be defined. There are several hundres entries. Table name: DS_NAMING.

Data Series Naming/Definition Table			
Name	*Null?*	*Type*	*Description*
DATA_SERIES_INDEX	NOT NULL	NUMBER(8)	Integer number pointing to this data series name
PROG_NAME	NOT NULL	VARCHAR2(20)	Project name (portion of dataset name)
LEVEL_NAME	NOT NULL	VARCHAR2(20)	Level name (portion of dataset name)
SERIES_NAME	NOT NULL	VARCHAR2(20)	Series name (portion of dataset name)
CONFORM	NOT NULL	VARCHAR2(8)	Conforming type (ES=equally spaced, AR=as received)
SOI_TYPE	NOT NULL	VARCHAR2(8)	SOI Type (DPS = data product series, TS=time series)
BLOCK	NOT NULL	VARCHAR2(8)	Blocking type (none, by hour, by day)
SERIES_NUM_FLD	NOT NULL	VARCHAR2(16)	Series number field (t_obs, t_refc, tim_freq)
DESCRIPTION	NOT NULL	VARCHAR2(80)	Description of the data series
N_VARIABLE	NOT NULL	NUMBER(8)	Number of variables in each data set
DB_LOG_TABLE		VARCHAR2(32)	DB table name associated with this data series
DESCRIPTION		VARCHAR2(80)	Description

13.2 Dataset Table

There is an entry per dataset. There are many 10,000 entries. Table name: DSDS_TEST.

DataSet Table			
Name	*Null?*	*Type*	*Description*
PROG_NAME	NOT NULL	VARCHAR2(20)	Project name (portion of dataset name)
PROG_NUM	NOT NULL	NUMBER(8)	Project number (portion of dataset name)
LEVEL_NAME	NOT NULL	VARCHAR2(20)	Level name (portion of dataset name)
SERIES_NAME	NOT NULL	VARCHAR2(20)	Series name (portion of dataset name)
SERIES_NUM	NOT NULL	NUMBER(8)	Series number (portion of dataset name)
SVC_VERSION	NOT NULL	VARCHAR2(16)	CM version number of the creating server
SVC_NAME	NOT NULL	VARCHAR2(20)	Name of the creating server
LEVEL_NUM		NUMBER(8)	Level number (portion of dataset name)
S_SERIAL_NUM		NUMBER(8)	Starting serial number
E_SERIAL_NUM		NUMBER(8)	Ending serial number
ONLINE_STATUS		VARCHAR2(5)	Flag whether the data is on-line or not
ONLINE_LOC		VARCHAR2(80)	Description of on-line location (disk/directory)
ARCHIVE_STATUS		VARCHAR2(5)	Flag whether the data is archived or not
ARCH_TAPE_ID		NUMBER(5)	Tape number
WORK_TAPE_ID		NUMBER(5)	Tape number
SAFE_TAPE_ID		NUMBER(5)	Tape number
HISTORY_COMMENT		VARCHAR2(60)	Comment field
BYTES		NUMBER(20)	Size of dataset in bytes
CREATE_UID	NOT NULL	NUMBER(38)	DSDS unique ID of the creator of the dataset
CREAT_DATE		DATE	Date created
TAPE_POSITION		VARCHAR2(15)	Position on tape where archived
TAPE_FILE_NUM		NUMBER(5)	File number on tape where archived
INFO_LIST_USED (?)		VARCHAR2(80)	Name of the info keyword list used to create info file
LISTS_USED (?)		VARCHAR2(80)	Name of ancillary list used to create the dataset
START_TIME (?)		NUMBER(8)	First time data is available in this dataset(?)
STOP_TIME (?)		NUMBER(8)	Last time data is available in this dataset(?)
DMIN (?)		NUMBER(8)	Minimum value for all images in dataset (or avg min)
DMAX (?)		NUMBER(8)	Maximum value for all images in dataset (or avg max)
DAVG (?)		NUMBER(8)	Average value for all images in dataset
DRMS (?)		NUMBER(8)	Stddev value for all images in dataset (or avg dev)

13.3 Ingest High Rate Science Log

There is an entry per station contact. There are several 1000 entries. Table name: INGEST_HKSC_LOG.

Ingest High Rate Science Log			
Name	*Null?*	*Type*	*Description*
STATION_FILE		VARCHAR2(255)	Full path name of station file
S_SFDU_TIME		NUMBER	Start SFDU date/time
E_SFDU_TIME		NUMBER	End SFDU date/time
S_REF_TIME		NUMBER	Start REFC date/time
E_REF_TIME		NUMBER	End REFC date/time
TAPE_NUM	NOT NULL	NUMBER(20)	Tape number (assigned when ingested)
NUM_OF_PKT		NUMBER(10)	Number of packets in the station file
STATION_FILE_NAME	NOT NULL	VARCHAR2(60)	Station file name only

13.4 Merge High Rate Science Log

There is an entry per DPC per hour, so it logs what kinds of DPCs exist in the hour. There are many 10,000s. Table name: MERGE_HRSC_LOG.

Merge High Rate Science Log			
Name	*Null?*	*Type*	*Description*
HOUR_NUM	NOT NULL	NUMBER(8)	Hour number of mission (or epoch)
DPC	NOT NULL	NUMBER(38)	Data product code
DPC_HEX		VARCHAR2(8)	Data product code as a hex value
NUM_OF_DPC		NUMBER(8)	Number of occurances of that DPC in the hour

13.5 Level 0 High Rate Science Log

TODO - a single table with min/max/avg/dev?

13.6 Mapping Which DPCs Go Into Which DPIs

TODO ??

A Details About the Dataset Naming Convention

Include Kay notes, phil notes, Tech Note 104 and ???

B Data Product Code (DPC) Encoding

The Data Product Code (DPC) is a 4 byte parameter that defines the type of data produced by the MDI instrument. It is suggested that the DPC be split into the following fields:

```
        bit 31   0 indicates flight data product
                 1 indicates ground data product

        bit 30   0 indicates 5 kbps science telemetry data product
                 1 indicates 160 kbps science telemetry data product

    bits 28-29  00 indicates standard science data products
                01 indicates limb figure science data products
                10 indicates tables and lists
                11 indicates other data products including
                   memory dumps, ISS mode etc.

    bits 16-27     indicate science observable or type of acquisition

    bits 00-15     indicate data format and parameters or identifier
```

The Data Product Code described above seen as bit fields:

```
    byte 1   byte 2   byte 3   byte 4
   XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

   X                                       = Ground or flight
   0                                         *** Flight ***
   1                                         *** Ground ***

    X                                      = 5 kbps or 160 kbps
    0                                        *** 5kbps telemetry ***
    1                                        *** 160kbps telemetry ***

     XX                                    = Type of data product
     00                                       *** Standard Science Products ***
     01                                       *** Limb Figure Products ***
     10                                       *** Lists and Table Products ***
     11                                       *** Other Products ***

       XXXX XXXXXXXX                       = Science observable, table/list type,
```

```
                                  or other identifier

        XXXXXXXX XXXXXXXX  = Data format & parameters
```

For standard science and limb figure data products bits 16-27 indicates science observables. At a minimum, the science observables include full disk and high resolution image and calibration mode for each of the following:

```
        filtergram
        Sa - 4 summed filtergrams
        Sa - 8 summed filtergrams
        Sc - 4 summed filtergrams
        Sc - 8 summed filtergrams
        Sd - 8 summed filtergrams
        Sd - 16 summed filtergrams
        Si - 2 summed filtergrams
        Si - 4 summed filtergrams
        dopplergram - 4 filtergram
        dopplergram - 8 filtergram
        dopplergram - 16 filtergram
        line depth - 8 filtergram
        line depth - 16 filtergram
        continuum intensity - 10 filtergram
        continuum intensity - 20 filtergram
        magnetogram - 8 filtergram
        magnetogram - 16 filtergram
```

For standard science data products, bits 00-15 indicate the data format and any parameters necessary to fully describe the image. Data format descriptions include normal, IP compressed, and time weighted averages for each of the following:

```
        cropped image with a crop radius identifier
        nxm extract with an extract parameter identifier
        nxm binning with a rebin list identifier
        rectangular weighted binning with a rwbin list identifier
        vector weighted binning with a vwbin list identifier
        irregular binning with a irregular list identifier
```

For limb figure data products, bits 00-15 provide the limb figure radius and thickness, with bits 00-11 indicating the inner crop radius in 1/8 pixels and bits 12-15 indicating the delta crop radius. A suggested formula for the outer radius is:

    outer_radius = inner_radius + delta_radius + offset

If the delta_radius is given in 1/8 pixel values, then the offset should be 40 in order to provide a 5 pixel baseline for the limb figure radius. For an inner radius of 480 pixels, bits 00-15 would then be 8F00 (hex) for a 6 pixels delta radius and the outer radius would be 486.

For lists and tables bits, 16-27 indicate the type of list or table and bits 00-15 are an identifying index. Typical list/table types are velocity, square root, and reciprocal look up tables, and binning, crop list and flat field tables. The available index space is sufficient that a unique index identifier could be associated with each list or table and that they could be associated with the corresponding data format.

For other data products, bits 16-27 indicates the type of acquisition and bits 00-15 provide a data identifier. "Other" data products include ground calibration data, IP memory dumps, ISS high rate sampled data, and simulated test data.

The ground calibration data that has been collected over the past two years falls into the "other" data product category with an acquisition type of xFFF with data identifiers from x0000 to xFFFF.

B.1 Table of Existing DPCs

TODO - Merge or replace with Katie's table. New columns (imin, imax, limb_fit ok?)

```
;FILE:
;       $MDI_CAL_INFO/dpc_table.txt
;HISTORY:
;       Created 18-Apr-94 by M.Morrison
;        5-May-94 (MDM) - Adjusted format slightly.  Added Mnemonic column
;
; For Summing:
;       0 means non-image, 1 is full resolution, 4 is 4x4 IP summing
; For NX:
;       -1 means variable length
;
; Data Product    NX      NY    Summing         Mnemonic                          Comments
;   Code

'FFFFFF01'XL     256     256    4       256x256_sum_5fast       # 4 x 4 summed image, 5 images taken fast and stored in IP
'FFFFFF02'XL     256     256    4       256x256_sum             # 4 x 4 summed image
```

```
'FFFFFF03'XL     256      256    1        256x256_ext             # 256 x 256 image extracted from 1024x1024
'FFFFFF04'XL    1024     1024    1        1024x1024               # 1024 x 1024 image

'FFFFFF15'XL     128      128    1        128x128_ext             # 128 x 128 image extracted from 1024x1024

'FFFFFF40'XL    1024     1024    1        1024_velocity           # 1024X1024 Velocity
'FFFFFF41'XL    1024     1024    1        1024_continuum          # 1024X1024 square root compressed continuum
'FFFFFF42'XL    1040     1033    1        1024_raw_i1-i3          # i1-i3 raw
'FFFFFF43'XL    1040     1033    1        1024_raw_i2-i4          # i2-i4 raw
'FFFFFF44'XL    1040     1033    1        1024_raw_continuum      # raw continuum

'FFFFFF82'XL    1024     1024    1        mathur_test             # D.Mathur test IP data product
'FFFFFF84'XL    1024     1024    1        1024x1024_comp          # 1024 x 1024 telemetry compressed image from IP

'FFFFFF50'XL    1040     1033    1        ipdmatst_raw            # ipdmatst - raw image
'FFFFFF51'XL    1040     1033    1        ipdmatst_sum            # ipdmatst - DMA addition of two raw images (1st - 2nd)
'FFFFFF52'XL    1040     1033    1        ipdmatst_diff           # ipdmatst - DMA difference of two raw images

'FFFFFF70'XL    '4000'XL  1      0        iptbltst_sqrt           # iptbltst - SQRT table
'FFFFFF71'XL    '8000'XL  1      0        iptbltst_reciprocal     # iptbltst - reciprocal table
'FFFFFF72'XL    '8000'XL  1      0        iptbltst_velocity       # iptbltst - velocity table
'FFFFFF73'XL    -1       1       0        crop_table              # crop table

'FFFFFF60'XL    1040     1033    0        ipmemtst_raw            # ipmemtst - FILL pattern raw
'FFFFFF61'XL    1040     1033    0        ipmemtst_compressed     # ipmemtst - FILL pattern compressed
'FFFFFF62'XL    1040     1033    0        ipmemtst_packed         # ipmemtst - FILL pattern packed

'10000000'XL    -1       1       0        unknown                 # ??
```

C Sequence ID Encoding

The "Observation ID" is associated with the list of frames that makes up an individual observation, while the "Campaign List ID" is associated with a list of observations and their associated parameters (including repeats).

```
Here's how sequence ID is being implemented:

        xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
        x                                     = Ground or Flight
         x                                    = Telecommand or Sequence
          x                                   = Standard or "other" sequence
           xxx                                = Sequencer version
              xx x                            = Part of sequence executing
                  xxxxxxx xxxx                = Campaign List ID
                              xxxx xxxxxxxx   = Observation ID


Bit 31: 0:        Ground Test
        1:        Flight

Bit 30: 0:        Telecommand
        1:        Sequence

Bit 29: 0:        Standard Sequencer
        1:        Some other sequence

Remaining bits are meaningful only for standard sequencer

Bits 28-26:       Era, i.e. version of the standard sequencer
Bits 25-22:       Where am I, i.e. which part of the sequence is
                  executing:
                0:       prime60
                1:       Head transistion on the left
                2:       Prime 30 on the left
                3:       Alt observation on the left
                4:       Tail transition on the left
                8:       Head transition on the right
                9:       Prime 30 on the right
                A:       Campaign
                B:       Alt observation on the right
                C:       Tail transition on the right
```

```
Bits 21-12:       Campaign List ID
Bits 11-0:        Observation ID
```

D DSDS Data Dictionary

Already exists (?)

E FITS Keys (Field) Name Dictionary

————— SOHO/SOI Keywords Sorted by Category —————						
Keyword	Type	Category	Defined By	Defined 0-1-2-	Present 0.I 1.I 2.I	Description
SFDUADID	string	??	SOHO	- - -	. . .	SFDU ADID for a data product ???? Standard Formatted Data Unit (SFDU) Authority a
THEAP	???	??	FITS	- - -	. . .	
T_DEL##	float	??	SOI	- - -	. . .	???
DFNAME##	string	DSDS	SOI	- - -	. . .	File name prefix for each variable
DNAME##	string?	DSDS	SOI	- - -	. . .	Names of variables
DSDS_LEV	string	DSDS	SOI	Y-Y-Y-	Y. Y. Y.	DSDS level portion of this dataset name SOURCE: DSDS
DSDS_PRG	string	DSDS	SOI	Y-Y-Y-	Y. Y. Y.	DSDS prog portion of this dataset name SOURCE: DSDS
DSDS_REC	string	DSDS	SOI	Y-Y-Y-	Y. Y. Y.	DSDS record (sel) of this dataset SOURCE: DSDS
DSDS_SER	string	DSDS	SOI	Y-Y-Y-	Y. Y. Y.	DSDS series name and # of this dataset SOURCE: DSDS
D_VARS	string?	DSDS	SOI	- - -	. . .	How many variables
CCD_AMP	string	MDI-STAT	SOI	Y- - -	Y. . .	CCD Amplified used (A or B)
EXT_ROWS	integer	MDI-STAT	SOI-TEST	- - -	. . .	Number of extended rows (eler) Number of extended rows (eler)
MKDPFRMN	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Frame Number SOURCE: CCDHD word-054 32-bits
MKDPHKPN	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Housekeeping Packet Number SOURCE: CCDHD word-058 32-bits
MKDPOCVR	integer	MDI-STAT	SOI	- - -	. . .	Optics Config Table Version SOURCE: CCDHD word-064 8-bits
MKDPOLVR	integer	MDI-STAT	SOI	- - -	. . .	OCT Lookup Table Version SOURCE: CCDHD word-065 8-bits
MKDPRSTN	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	DEP Restart Number SOURCE: CCDHD word-052 16-bits
MKDPSQPN	integer	MDI-STAT	SOI	- - -	. . .	DEP Sequence List Patch SOURCE: CCDHD word-067 8-bits
MKDPSQVR	integer	MDI-STAT	SOI	- - -	. . .	DEP Sequence List Version SOURCE: CCDHD word-066 8-bits
MKDPSWPN	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	DEP Software Patch SOURCE: CCDHD word-063 8-bits
MKDPSWVR	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	DEP Software Version SOURCE: CCDHD word-062 8-bits
MKFDMTR1	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Door Motor 1 Status SOURCE: CCDHD word-078 1-bits
MKFDMTR2	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Door Motor 2 Status SOURCE: CCDHD word-078 1-bits
MKFDOPN1	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Door Open Status 1 SOURCE: CCDHD word-078 1-bits
MKFDOPN2	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Door Open Status 2 SOURCE: CCDHD word-078 1-bits
MKIPDMP1	integer	MDI-STAT	SOI	- - -	. . .	IP Dump Word 1 SOURCE: CCDHD word-114 16-bits
MKIPDMP2	integer	MDI-STAT	SOI	- - -	. . .	IP Dump Word 2 SOURCE: CCDHD word-116 16-bits
MKIPDMP3	integer	MDI-STAT	SOI	- - -	. . .	IP Dump Word 3 SOURCE: CCDHD word-118 16-bits
MKIPDMP4	integer	MDI-STAT	SOI	- - -	. . .	IP Dump Word 4 SOURCE: CCDHD word-120 16-bits
MKIPDMP5	integer	MDI-STAT	SOI	- - -	. . .	IP Dump Word 5 SOURCE: CCDHD word-122 16-bits
MKIPDMPS	integer	MDI-STAT	SOI	- - -	. . .	IP Dump Error Status SOURCE: CCDHD word-112 16-bits
MKIPFWPN	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	IP Firmware Patch SOURCE: CCDHD word-069 8-bits
MKIPFWVR	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	IP Firmware Version SOURCE: CCDHD word-068 8-bits
MKIPIQPN	integer	MDI-STAT	SOI	- - -	. . .	IP Instruction Patch SOURCE: CCDHD word-071 8-bits
MKIPIQVR	integer	MDI-STAT	SOI	- - -	. . .	IP Instruction Queue Version SOURCE: CCDHD

————————— SOHO/SOI Keywords Sorted by Category —————————						
MKIPTID2	integer	MDI-STAT	SOI	- - -	. . .	IP Table ID #2 SOURCE: CCDHD word-102 16-bits
MKIPTNE1	integer	MDI-STAT	SOI	- - -	. . .	IP Table Number of Entries #1 SOURCE: CCDHD word-100 16-bits
MKIPTNE2	integer	MDI-STAT	SOI	- - -	. . .	IP Table Number of Entries #2 SOURCE: CCDHD word-110 16-bits
MKIPTVR1	integer	MDI-STAT	SOI	- - -	. . .	IP Table Version #1 SOURCE: CCDHD word-094 16-bits
MKIPTVR2	integer	MDI-STAT	SOI	- - -	. . .	IP Table Version #2 SOURCE: CCDHD word-104 16-bits
MKLTLOOP	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	ISS Loop Open/Closed SOURCE: CCDHD word-078 1-bits
MKLTXOFS	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Limb Tracker X offset SOURCE: CCDHD word-083 8-bits
MKPWHCM	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	CCD Heater Power (K30) SOURCE: CCDHD word-078 1-bits
MKPWHRO	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	B/U Oven Heater Pwr (K29) SOURCE: CCDHD word-078 1-bits
MKSQAM1	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	AM1 Position and Status SOURCE: CCDHD word-072 16-bits
MKSQAM2	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	AM2 Position and Status SOURCE: CCDHD word-074 16-bits
MKSQC1	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	C1 Position SOURCE: CCDHD word-076 8-bits
MKSQC2	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	C2 Position SOURCE: CCDHD word-077 8-bits
MKSQFRM	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Frame Number in Sequence SOURCE: CCDHD word-044 16-bits
MKSQID	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Sequence Identifier SOURCE: CCDHD word-038 32-bits
MKSQLTST	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Limb Tracker Status SOURCE: CCDHD word-083 8-bits
MKSQNUM	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Sequence Number SOURCE: CCDHD word-042 16-bits
MKSQREF0	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	LOBT Reference 0 SOURCE: CCDHD word-046 8-bits
MKSQREF1	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	LOBT Reference 1 SOURCE: CCDHD word-047 8-bits
MKSQREF2	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	LOBT Reference 2 SOURCE: CCDHD word-048 8-bits
MKSQREF3	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	LOBT Reference 3 SOURCE: CCDHD word-049 8-bits
MKSQREF4	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	LOBT Reference 4 SOURCE: CCDHD word-050 8-bits
MKSQREF5	integer	MDI-STAT	SOI	Y- - -	Y. . .	LOBT Reference 5 SOURCE: CCDHD word-051 8-bits
MKSQRG0	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 0 SOURCE: CCDHD word-006 16-bits
MKSQRG1	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 1 SOURCE: CCDHD word-008 16-bits
MKSQRG2	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 2 SOURCE: CCDHD word-010 16-bits
MKSQRG3	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 3 SOURCE: CCDHD word-012 16-bits
MKSQRG4	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 4 SOURCE: CCDHD word-014 16-bits
MKSQRG5	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 5 SOURCE: CCDHD word-016 16-bits
MKSQRG6	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 6 SOURCE: CCDHD word-018 16-bits
MKSQRG7	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 7 SOURCE: CCDHD word-020 16-bits
MKSQRG8	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 8 SOURCE: CCDHD word-022 16-bits
MKSQRG9	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register 9 SOURCE: CCDHD word-024 16-bits
MKSQRGA	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register A SOURCE: CCDHD word-026 16-bits
MKSQRGB	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register B SOURCE: CCDHD word-028 16-bits
MKSQRGC	integer	MDI-STAT	SOI	- - -	. . .	Sequence Register C SOURCE: CCDHD word-030 16-

———————— SOHO/SOI Keywords Sorted by Category ————————						
MTSQOPT1	integer	MDI-STAT	SOI	Y- - -	Y. . .	Front Window Temp SOURCE: CCDHD word-088 8-bits
MTSQOPT4	integer	MDI-STAT	SOI	Y- - -	Y. . .	Oven Temperature SOURCE: CCDHD word-089 8-bits
MTSQOPT5	integer	MDI-STAT	SOI	Y- - -	Y. . .	Oven Temperature SOURCE: CCDHD word-090 8-bits
MTSQOPT6	integer	MDI-STAT	SOI	Y- - -	Y. . .	CCD Temp SOURCE: CCDHD word-091 8-bits
MTSQTMP1	float	MDI-STAT	SOI	-Y- -	. Y. .	Front window temp in degrees UNITS: celcius SOURCE: LEVEL0 MTSQOPT1
MTSQTMP4	float	MDI-STAT	SOI	-Y- -	. Y. .	Oven temperature in degrees UNITS: celcius SOURCE: LEVEL0 MTSQOPT4
MTSQTMP5	float	MDI-STAT	SOI	-Y- -	. Y. .	Oven Temperature in degrees UNITS: celcius SOURCE: LEVEL0 MTSQOPT5
MTSQTMP6	float	MDI-STAT	SOI	-Y- -	. Y. .	CCD Temperature in degrees UNITS: celcius SOURCE: LEVEL0 MTSQOPT6
MUHOPCTL	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Optics package Heater Controller SOURCE: CCDHD word-084 8-bits
MUHPOBRG	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Filter Oven Bridge Out SOURCE: CCDHD word-085 8-bits
MUHPOLOP	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Filter Oven Loop Out SOURCE: CCDHD word-086 8-bits
MUHROCTL	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Filter Oven Redundant Controller SOURCE: CCDHD word-087 8-bits
MULTXAVE	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Limb Tracker X Average Error SOURCE: CCDHD word-081 8-bits
MULTXMAX	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Limb Tracker X Max Error SOURCE: CCDHD word-079 8-bits
MULTYAVE	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Limb Tracker Y AverageError SOURCE: CCDHD word-082 8-bits
MULTYMAX	integer	MDI-STAT	SOI	Y- - -	Y. Y. .	Limb Tracker Y Max Error SOURCE: CCDHD word-080 8-bits
TEMP##	integer	MDI-STAT	SOI-TEST	- - -	. . .	Temperature ## from camera string Temperature n from camera string (temp(n))
EXT_COLS	integer	MDI_STAT	SOI-TEST	- - -	. . .	Number of extended pixels (eper) Number of extended pixels (eper)
DATAMAX	real	data	FITS	- - -	. . .	Data minimum SOURCE: DATA
DATAMIN	real	data	FITS	- - -	. . .	Data maximum SOURCE: DATA
D_AVG##	float	data	SOI	- - -	. . .	Data average value SOURCE: DATA
D_BOXSIZ	float	data	SOI	- - -	. . .	Box size used to calc min/max/avg/rms SOURCE: Program
D_MAX##	float	data	SOI	- - -	. . .	Data maximum value SOURCE: DATA Same as DATAMAX?
D_MIN##	float	data	SOI	- - -	. . .	Data minimum value SOURCE: DATA Same as DATAMIN?
D_NUM##	float	data	SOI	- - -	. . .	???
D_RMS##	float	data	SOI	- - -	. . .	Data standard deviation SOURCE: DATA
OBSRVABL	string	data	SOI	- - -	. . .	String containing observable quantity.
OBSSIGN	string	data	SOI	- - -	. . .	Sign of observable Sign of velocty +1 if velocities are away from the observer, -1 if towards the observer
OBSUNITS	string	data	SOI	- - -	. . .	String containing units of OBSRVABL. SOURCE: Same as BUNIT?
BITPIX	integer	data-info	FITS	Y-Y-Y-	Y. Y. Y.	The data type (byte, integer, float, ... Mandatory in all headers. Must be second keyword in header. SOURCE: DATA
BLANK	integer	data-info	FITS	- - -	. . .	Undefined pixels are set to this value May be used only if BITPIX value is positive (integer data).
BSCALE	real	data-info	FITS	- - -	. . .	Scaling factor to convert data to float true_data_value = (value*BSCALE + BZERO) SOURCE: Program
BUNIT	string	data-info	FITS	- - -	. . .	Units of the data SOURCE: Program?
BZERO	real	data-info	FITS	- - -	. . .	Offset applied to true pixel values true_data_value = (value*BSCALE + BZERO)
CDELT##	real	data-info	FITS	- - -	. . .	Increment in physical coord along axis##
CROTA##	real	data-info	FITS	- - -	. . .	Rotation of axis ## Rotation of axis ## from a standard coordinate system in degrees. The rotation will be counter-clockwise about the center of the reference pixel given by CRPIX##

————— SOHO/SOI Keywords Sorted by Category —————						
NAXIS	integer	data-info	FITS	- - -	. . .	Number of axes Mandatory in all headers. Must be third keyword in header. SOURCE: DATA
NAXIS##	integer	data-info	FITS	Y-Y-Y-	Y. Y. Y.	Size of the dimension ## Mandatory in all headers up to n = NAXIS of the header. Must follow NAXIS keyword in order (NAXIS1, NAXIS2, ...) unless NAXIS = 0. It is not clear whether additional NAXISn keywords may be present, but the implication is that the number of NAXISn keywords should be exactly equal to the value of NAXIS. SOURCE: DATA
AUTHOR	string	file-info	FITS	- - -	. . .	The author of the program SOURCE: Program
BLOCKED	bool	file-info	FITS	- - -	. . .	If present, value must apparently be T. Deprecated.
COMMENT	none	file-info	FITS	- - -	. . .	Comment string No value. Any number may appear. Column 9 may not contain '='. SOURCE: Program
COMPRES	??	file-info	FITS	- - -	. . .	
COMPRES##	??	file-info	FITS	- - -	. . .	
END	none	file-info	FITS	- - -	. . .	Marks the end of the FITS header Mandatory in all headers. Must be last keyword in header. SOURCE: Program
EXTEND	bool	file-info	FITS	- - -	. . .	Mandatory in primary header if data set contains extensions, in which case it must immediately follow the last NAXISn keyword card image (or NAXIS if NAXIS = 0).
EXTLEVEL	integer	file-info	FITS	- - -	. . .	
EXTNAME	string	file-info	FITS	- - -	. . .	
EXTVER	integer	file-info	FITS	- - -	. . .	
FILENAME	string	file-info	SOHO	- - -	. . .	Name of data file Name of data file. The use of this keyword will not be construed to mean that the FITS header will need to be changed if the file is copied from one computer to another, or to another media such as tape. SOURCE: Program
FILEORIG	string	file-info	SOHO	- - -	. . .	Original filename as collected Original filename as collected.
GCOUNT	integer	file-info	FITS	- - -	. . .	Mandatory in extension headers, in which case it must immediately follow the PCOUNT keyword.
GROUPS	bool	file-info	FITS	- - -	. . .	
HISTORY	none	file-info	FITS	- - -	. . .	History comments No value. Any number may appear. Column 9 may not contain '='. SOURCE: Program
LAXIS	??	file-info	FITS	- - -	. . .	
LAXIS##	??	file-info	FITS	- - -	. . .	
LBITPIX	??	file-info	FITS	- - -	. . .	
LEXTEND	??	file-info	FITS	- - -	. . .	
LEXTENSN	??	file-info	FITS	- - -	. . .	
ORIGIN	string	file-info	FITS	- - -	. . .	Organization creating file (SOHO-SOI) Organization creating the FITS file. Possible values could be "SOHO", "SOHO-EOF", SOHO-Univ of Maryland", etc. SOURCE: Program
PCOUNT	integer	file-info	FITS	- - -	. . .	Mandatory in extension headers.
PSCAL##	float	file-info	FITS	- - -	. . .	
PTYPE##	string	file-info	FITS	- - -	. . .	
PZERO##	real	file-info	FITS	- - -	. . .	
REFERENC	string	file-info	FITS	- - -	. . .	
SEQONLIN	string	file-info	SOHO	- - -	. . .	'Y' or 'N' whether data is on-line !?!?! Either 'Y' or 'N', denoting whether or not the data from an observing sequence is online or not.
SIMPLE	bool	file-info	FITS	Y-Y-Y-	Y. Y. Y.	FITS file type Mandatory in primary header; not permitted in extension headers. The value must be false if the file does not conform to the FITS standard. Otherwise it should be true. Must be first keyword in header. SOURCE: Program
XTENSION	??	file-info	FITS	- - -	. . .	
BLS_PIX	integer	instr-info	SOI-TEST	- - -	. . .	Number of BLS pixels (bls) Number of BLS pixels (bls)
CADENCE	float	instr-info	SOI	- - -	. . .	Interval between measurements in a set The interval between measurements of a data product. For example, the limb figure is sent down every 12 minutes, but it is a 24 minute average (==SPANTIME)
EXPTIME	float	instr-info	SOHO	Y- - -	Y. Y. .	Exposure time in seconds Filtergram exposure time in seconds, to millisecond accuracy. UNITS: seconds

—————— SOHO/SOI Keywords Sorted by Category ——————						
WAVEMIN	float	instr-info	SOHO	- - -	. . .	Minimum wavelength of obs, in nm. Minimum wavelength of observation, in nm. Also, TWMIN for binary tables.
CMD_DESC	string	obs-info	SOHO	- - -	. . .	A description of the campaign
CMP_NAME	string	obs-info	SOHO	- - -	. . .	Name of observing campaign Name of observing campaign.
CMP_NO	integer	obs-info	SOHO	- - -	. . .	Unique identifier for observing campaign Unique identifier for an observing campaign.
COMMENTS	string	obs-info	SOHO	- - -	. . .	'Y' or 'N' if database is commented Either 'Y' or 'N', depending on whether or not an entry in a database is commented or not. For use in the online catalogs only.
COMM_NO	integer	obs-info	SOHO	- - -	. . .	Comment number Comment number. For use in the online catalogs only.
COUNT	integer	obs-info	SOHO	- - -	. . .	Number of repeated observing sequences Number of repeated observing sequences within an observing program.
DETECTOR	string	obs-info	SOHO	Y- - -	Y. Y. Y.	Detector used to acquire the data Detector used to acquire the data. SOURCE: Program
EXPCOUNT	integer	obs-info	SOHO	- - -	. . .	Number of exposures within an obs seq Number of exposures within an observing sequence.
EXP_IND	integer	obs-info	SOHO	- - -	. . .	Index of exposure within an obs seq Index of an exposure within an observing sequence, from one to the value of EXPCOUNT.
FRAME	string	obs-info	SOI-TEST	- - -	. . .	Frame position descriptor (frame) Frame position descriptor (frame)
FTYPE	string	obs-info	SOI-TEST	- - -	. . .	Datatype descriptor (ftype) Datatype descriptor (ftype)
IMAGENO	integer	obs-info	SOI-TEST	- - -	. . .	Image serial number (imnum) Image serial number (imnum)
INSTITUT	string	obs-info	SOHO	- - -	. . .	Name of intrument team or institution Name of an intrument team or institution taking part in an observing campaign. SOURCE: Program
INSTRUME	string	obs-info	FITS	- - -	. . .	Instrument (MDI, CDS, SUMER, ...) 'MDI ' and 'Michelson Doppler Imager' are appropriate values. SOURCE: Program
OBJECT	string	obs-info	FITS	- - -	. . .	Name of object observed
OBJ_ID	string	obs-info	SOHO	- - -	. . .	Object identifier, eg active region # Object identifier, e.g. active region number.
OBSERVER	string	obs-info	FITS	- - -	. . .	Observer
OBS_MODE	string	obs-info	SOHO	- - -	. . .	Observing mode Observing mode.
OBS_PROG	string	obs-info	SOHO	- - -	. . .	The observing program The observing program.
OBS_SEQ	string	obs-info	SOHO	- - -	. . .	Name of the observing sequence used The name of the observing sequence used.
PROG_ID	integer	obs-info	SOHO	- - -	. . .	Observing program identification number Observing program identification number. PROG_I
PROG_NUM	integer	obs-info	SOHO	- - -	. . .	Unique identifier for the obs program Unique identifier for the observing program. Each observing program is assigned a unique identifier number each time it's run, even if it's a continuation of an earlier program.
SCI_OBJ	string	obs-info	SOHO	- - -	. . .	The science objective The science objective.
SCI_SPEC	string	obs-info	SOHO	- - -	. . .	Specific scientific objective Modifies or claifying SCI_OBJ. For example, SCI_OBJ may be "corronal hole study", while SCI_SPEC might be "Density diagnostics"
SEQUENCE	string	obs-info	SOI-TEST	- - -	. . .	sequencename, incl. serial number (seq) sequencename, incl. serial number (seq)
SEQVALID	string	obs-info	SOHO	- - -	. . .	'Y'or 'N' whether data is usable Either 'Y'or 'N', denoting whether or not the data from an observing sequence has scientific value.
SEQ_FROM	integer	obs-info	SOHO	- - -	. . .	First value of SEQ_NUM First value of SEQ_NUM in a series of observing sequences.
SEQ_IND	integer	obs-info	SOHO	- - -	. . .	Index of an obs seq within obs prog Index of an observing sequence within an observing program, from one to the value of COUNT.
SEQ_NUM	integer	obs-info	SOHO	- - -	. . .	Unique number of execution of obs seq Unique number assigned to an execution of an observing sequence. Depending on how an instrument team organizes their

————————— SOHO/SOI Keywords Sorted by Category —————————						
OBS_CT	??	observer	SOI	- - -	. . .	carrtime?
OBS_DIST	float	observer	SOI	- - -	. . .	Distance from observer to sun center
OBS_L0	float	observer	SOI	- - -	. . .	Carrington long of observer's disk center
OBS_VN	float	observer	SOI	- - -	. . .	'Northward' velocity of observer in m/s
OBS_VR	float	observer	SOI	- - -	. . .	Radial velocity of observer in m/s
OBS_VW	float	observer	SOI	- - -	. . .	'Westward' velocity of observer in m/s
SOLAR_B0	float	observer	SOHO	- - -	. . .	The solar B0 angle The solar B0 angle. SOURCE: Ephemeris
SOLAR_P0	float	observer	SOHO	- - -	. . .	The solar P0 angle The solar P0 angle. SOURCE: Ephemeris
SUNDIST	float	observer	SOI	- - -	. . .	Distance from Observer to center of Sun Distance from Observer to center of Sun SOURCE: Same as OBS_DIST?
SUNRAD	float	observer	SOI	- - -	. . .	Apparent semidiameter of Solar Disc Apparent semidiameter of Solar Disc
ANGLE	float	pnt/orient	SOHO	- - -	. . .	Angle of rotation of the vertical axis Angle of rotation of the vertical axis of the instrument field-of-view relative to solar north.
AZ	float	pnt/orient	SOHO	- - -	. . .	Aximuth of a data point UNITS: degrees
AZ_END	float	pnt/orient	SOHO	- - -	. . .	Ending azimuth UNITS: degrees
AZ_START	float	pnt/orient	SOHO	- - -	. . .	Staring ecliptic azimuth of observation UNITS: degrees
CENTER_X	integer	pnt/orient	SOI-TEST	- - -	. . .	X-coord of image center in millipixels x-Coordinate of image center in millipixels used in gain calculation
CENTER_Y	integer	pnt/orient	SOI-TEST	- - -	. . .	Y-coord of image center in millipixels y-Coordinate of image center in millipixels used in gain calculation
DET_X	integer	pnt/orient	SOHO	- - -	. . .	X Position of pixel within a detector
DET_Y	integer	pnt/orient	SOHO	- - -	. . .	Y position of pixel within a detector
EL	float	pnt/orient	SOHO	- - -	. . .	Elevation of a data point UNITS: degrees
EL_END	float	pnt/orient	SOHO	- - -	. . .	Ending elevation UNITS: degrees
EL_START	float	pnt/orient	SOHO	- - -	. . .	Starting ecliptic elevation of obs UNITS: degrees
HGDCLAT	float	pnt/orient	SOI	- - -	. . .	Heliographic Latitude of observed center Heliographic Latitude of observed Solar Disc Center
HGDCLON	float	pnt/orient	SOI	- - -	. . .	Heliographic Longitude of center Heliographic Longitude of observed Solar Disc Center
INS_ROLL	float	pnt/orient	SOHO	- - -	. . .	Instrument roll angle relative to S/C Instrument roll angle relative to the spacecraft, i.e. the angle between the instrument Y-axis and the spacecraft Y-axis, measured in a counter-clockwise direction from the spacecraft Y-axis toward the spacecraft Z-axis.
INS_X	float	pnt/orient	SOHO	- - -	. . .	Pointing of instrument along INSTR Xaxis Pointing of instrument along the instrument X-axis.
INS_X0	float	pnt/orient	SOHO	- - -	. . .	Pointing of origin along S/C X-axis Pointing of instrument origin along the spacecraft X-axis.
INS_Y	float	pnt/orient	SOHO	- - -	. . .	Pointing of instrument along instr Yaxis Pointing of instrument along the instrument Y-axis.
INS_Y0	float	pnt/orient	SOHO	- - -	. . .	Pointing of origin along S/C Y-axis Pointing of instrument origin along the spacecraft Y-axis.
ORIENT	string	pnt/orient	SOI	- - -	. . .	Orientation string, e.g. 'SESW' The orientation of the Sun's image in the file/array is defined by a
RADIUS	integer	pnt/orient	SOI-TEST	- - -	. . .	Radius of image in pixels Radius of image in pixels used in gain calculation
SC_ROLL	float	pnt/orient	SOHO	- - -	. . .	S/C roll angle relative to solar coord Spacecraft roll angle relative to solar coordinates.
SC_X0	float	pnt/orient	SOHO	- - -	. . .	S/C planned X-axis pointing Spacecraft pointing along the solar X-axis as defined in the Science Operations Plan.
SC_Y0	float	pnt/orient	SOHO	- - -	. . .	S/C planned Y-axis pointing Spacecraft pointing along the solar Y-axis as defined in the Science Operations Plan.
SOLAR_P	float	pnt/orient	SOI	- - -	. . .	Position angle of the Solar North pole. The P angle is measured counterclockwise UNITS: degrees SOURCE: Ephemeris
SOLAR_X	float	pnt/orient	SOHO	- - -	. . .	General pointing relative to solar X General pointing relative to solar X axis, which is defined as being in the plane of the sky as seen by the instrument, perpendicular to the projection of the solar north pole, and towards the west solar limb.

———————— SOHO/SOI Keywords Sorted by Category ————————						
YCEN	float	pnt/orient	SOHO	- - -	. . .	Y center of the instrument FOV Center of the instrument field-of-view along the solar Y-axis.
S_ANGLE	float	scale	SOI	- - -	. . .	Position angle of S_MAJOR axis. S_ANGLE is in degrees counterclockwise from west. For GONG data
S_MAJOR	float	scale	SOI	- - -	. . .	Semi-major axis scale of squished image
S_MINOR	float	scale	SOI	- - -	. . .	Semi-minor axis scale of squished image
TBD_2 ???	integer	scale	SOI	- - -	. . .	Pixel binning factor (1,2,3,4...)
TBD_3 ???	string	scale	SOI	- - -	. . .	Optics pixel mode (HR or FD)
X_SCALE	float	scale	SOI	- - -	. . .	X-axis Pixel scale in arc-seconds/pixel The coordinate system for X_SCALE and Y_SCALE is the same as for X0 and Y0.
Y_SCALE	float	scale	SOI	- - -	. . .	Y-axis Pixel scale in arc-seconds/pixel
CONVENTN	string	table-info	FITS-TAB	- - -	. . .	Convention to describe the binary tables Convention used to describe how the binary tables are used to store data. This keyword will appear in the binary table extension header. The value of this keyword will be TDIM.
TBCOL##	integer	table-info	FITS-TAB	- - -	. . .	Mandatory in ASCII Tables Extension headers, in which case the number of these keywords must be equal to the value of TFIELDS and they must appear in order, analogously to NAXISn.
TCUNI##	string	table-info	FITS-TAB	- - -	. . .	Units of dimension ##
TDELT##	string	table-info	FITS-TAB	- - -	. . .	Increment in physical coord Extension of CDELTn keywords to binary tables.
TDESC##	string	table-info	FITS-TAB	- - -	. . .	Description of table column Extension of CTYPEn keywords to binary tables. Has the format TDESCn='(desc1,desc2,. .).' where each element in the list corresponds to one of the dimensions in the associated TDIMnnn keyword.
TDIM##	string	table-info	FITS-TAB	- - -	. . .	Dimension of column ## The value is a character string containing the dimensions of a multidimensional array stored in column nnn in a binary table, with the format TDIMn='(dim1,dim2,. .).
TDISP##	??	table-info	FITS	- - -	. . .	
TDMAX##	real	table-info	FITS	- - -	. . .	Maximum for column ## in a table Equivalent of DATAMAX keyword for a column in an ASCII or binary table.
TDMIN##	real	table-info	FITS-TAB	- - -	. . .	Minumum for column ## in a table Equivalent of DATAMIN keyword for a column in an ASCII or binary table.
TFIELDS	integer	table-info	FITS-TAB	- - -	. . .	Mandatory in ASCII Tables Extension headers, in which case it must immediately follow the GCOUNT keyword.
TFORM##	string	table-info	FITS-TAB	- - -	. . .	Mandatory in ASCII Tables Extension headers, in
TNULL##	string	table-info	FITS-TAB	- - -	. . .	Undevined table values set to this value Reserved only for ASCII Tables extension data. Should be treated as reserved.
TROTA##	string	table-info	FITS-TAB	- - -	. . .	Extension of CROTAn to binary tables Extension of CROTAn keywords to binary tables. Has the format TROTAn='(rota1,rota2,. .).' where each element in the list corresponds to one of the dimensions in the associated TDIMnnn keyword.
TRPIX##	string	table-info	FITS-TAB	- - -	. . .	Extension of CRPIXn to binary tables Extension of CRPIXn keywords to binary tables.
TRVAL##	string	table-info	FITS-TAB	- - -	. . .	Extension of CRVALn to binary tables Extension of CRVALn keywords to binary tables.
TSCAL##	real	table-info	FITS-TAB	- - -	. . .	Scaling factor to convert to float Reserved only for ASCII Tables extension data. Should be treated as reserved.
TTYPE##	??	table-info	FITS	- - -	. . .	Label of column ##
TTYPEA##	string	table-info	FITS-TAB	- - -	. . .	Reserved only for ASCII Tables extension data. Should be treated as reserved.
TUNIT##	string	table-info	FITS-TAB	- - -	. . .	Units for data in column ## Reserved only for ASCII Tables extension data. Should be treated as reserved.
TZERO##	real	table-info	FITS-TAB	- - -	. . .	Offset applied to table values Reserved only for ASCII Tables extension data. Should be treated as reserved.
TDISN	???	table-info?	FITS-TAB	- - -	. . .	
DATE	string	time	FITS	Y-Y-Y-	Y. Y. Y.	Date the Header Data Unit was created Date on which

—————— SOHO/SOI Keywords Sorted by Category ——————						
OBT_OBS	float	time	SOI	-Y- -	. Y. .	TAI Time of observation at SOHO This time has the S/C clock correction applied and is the final time tag for the observation SOURCE: LEVEL1 OBT_DP+
OBT_REF	float	time	SOI	Y- - -	Y. Y. .	LOBT of DEP program loop when data obs REF-TIME is the intended LOBT, not corrected for SOHO LOBT drifts. Thus the LOBT values are traceable back to telemetry packets. LOBT times aascii "floats" and are in seconds from the SOHO TAI epoch which is th CCSDS standard of 1 January 1958 (EID-A, 3.3.9.2).e re SOURCE: CCDHD MKSQREF*
OBT_REFC	float	time	SOI	Y- - -	Y. Y. .	"ref" time corrected for gross clock err SOURCE: LEVEL0 OBT_REF
OBT_TIME	float	time	SOHO	- - -	. . .	LOBT of start of data acquisition SOURCE: LEVEL1 T_OBS+
START_TIME	string	time	SOI	- - -	. . .	Start of data, UT. Same as T_FIRST. FORMAT: yyyy.mm.dd_hh:mm:ss.
STOP_TIME	string	time	SOI	- - -	. . .	End of data, UT. Same as T_LAST. FORMAT: yyyy.mm.dd_hh:mm:ss.
TIME-OBS	string	time	SOI	- - -	. . .	UT time of observation UT time of observation. Values as described under TIME keyword. Format: 'HH:MM:SS.XXX' FORMAT: HH:MM:SS.XXX SOURCE: LEVEL0 T_OBS
TS_BLOCK	integer	time	SOI	- - -	. . .	Number of records per dataset
TS_EPOCH	string	time	SOI	- - -	. . .	Epoch for index calculations FORMAT: ???
TS_STEP	string	time	SOI	- - -	. . .	Time step per data record
T_DP	string	time	SOI	-Y- -	. Y. .	String format of OBT_DP FORMAT: yyyy.mm.dd_hh:mm:ss. SOURCE: LEVEL1 OBT_DP
T_EARTH	string	time	SOI	- - -	. . .	UT time if obs had been made from Earth FORMAT: yyyy.mm.dd_hh:mm:ss. SOURCE: LEVEL1 T_OBS+
T_FIRST	string	time	SOI	- - -	. . .	T_SOHO of start of data FORMAT: yyyy.mm.dd_hh:mm:ss.
T_LAST	string	time	SOI	- - -	. . .	T_SOHO of end of data FORMAT: yyyy.mm.dd_hh:mm:ss.
T_MEARTH	string	time	SOI	- - -	. . .	UT time if obs had been from 'Mean Earth FORMAT: yyyy.mm.dd_hh:mm:ss. SOURCE: LEVEL1 T_OBS+
T_OBS	string	time	SOI	-Y- -	. Y. .	UT time of obs at SOHO FORMAT: yyyy.mm.dd_hh:mm:ss. SOURCE: LEVEL1 OBT_OBS
T_REF	string	time	SOI	Y- - -	Y. Y. .	String format of OBT_REF FORMAT: yyyy.mm.dd_hh:mm:ss. SOURCE: LEVEL0 OBT_REF
T_REFC	string	time	SOI	Y- - -	Y. Y. .	String format of OBT_REFC FORMAT: yyyy.mm.dd_hh:mm:ss. SOURCE: LEVEL0 OBT_REFC
EQUINOX	string	time?	FITS	- - -	. . .	

F How To Interpret Ref_Time, DP_Time, and Time Conversions

TODO - offsets from ref_time to DP_time knowing the seqID

DIfferent kinds of time:

- ref_time - the raw ref_time as it comes down in the telemetry
- cor_ref_time - the corrected ref_time taking into account glitches in the MDI and corrupted telemetry
- dp_time (data product time) - the time for the center of the filtergram images used to make the dataproduct. It is an offset from the ref_time based on the SeqID.
- UT Time at earth
- UT time at the spacecraft
- SOHO time (??)

TODO -

NOTE that all of these times are kept as a single "word". They are stored internally as doubles with reference epochs as 1 Jan 1970 in the case of UT or DPTs, and with the SOHO epoch for LOBT derived times. External representation is in the form yyyy:mm:dd_hh:mm:ss for UT times and yyyy:mm:dd_hh:mm:ss_TAI for TAI based times. (Alternately the TAI times can be given in seconds from the SOHO referenced epoch)

- **reftime** – The time at which DEP loads MDI_LOBT at beginning of processing of a framelist. It is from about 7.5 to 15 seconds before the MDI_LOBT of the Data_Product_time. Reftime can be in error due to cosmic ray hits or because MDI_LOBT is in error.
- **MDI_LOBT** – MDI's current version of LOBT. MDI only resets its time to be the same as LOBT when commanded to do so. The old MDI_LOBT is lost so tracking of errors must be done by ground reconcillation of MDI housekeeping data information. MDI_LOBT can be in error beacuse the DEP counter got scrambled or because LOBT is wrong. MDI_LOBT is counted by a S/C supplied clock line.
- **LOBT** – SOHO value of TAI corrected by gradual changing of the clock rate by ground commands. LOBT is supposed to be the same as a ground station with a TAI clock. I.e. the transmission time to SOHO is supposed to have been taken into account when LOBT is set. LOBT is transferred to an experiment by telecommand. Upon the command request, the LOBT that will be true on the next major frame pulse is sent to the experiment. The DEP loads the LOBT into MDI_LOBT at the major frame pulse. LOBT can be wrong because the SOHO_LOBT got scrambled or because (more likely) there is an uncorrected rate error in the S/C clock.
- **Data_Product_TIme (DP_Time) -** The physically meaningful time of the center of the interval over which an observable (data product) is observed. In nominal operations, the Data_Product_Time for the structure program will increment by 60 seconds and will be on the TAI 30-second marks, which is very near to the even UT minute marks. By agreement with the other SOHO helioseismology experiments the normal structure program data

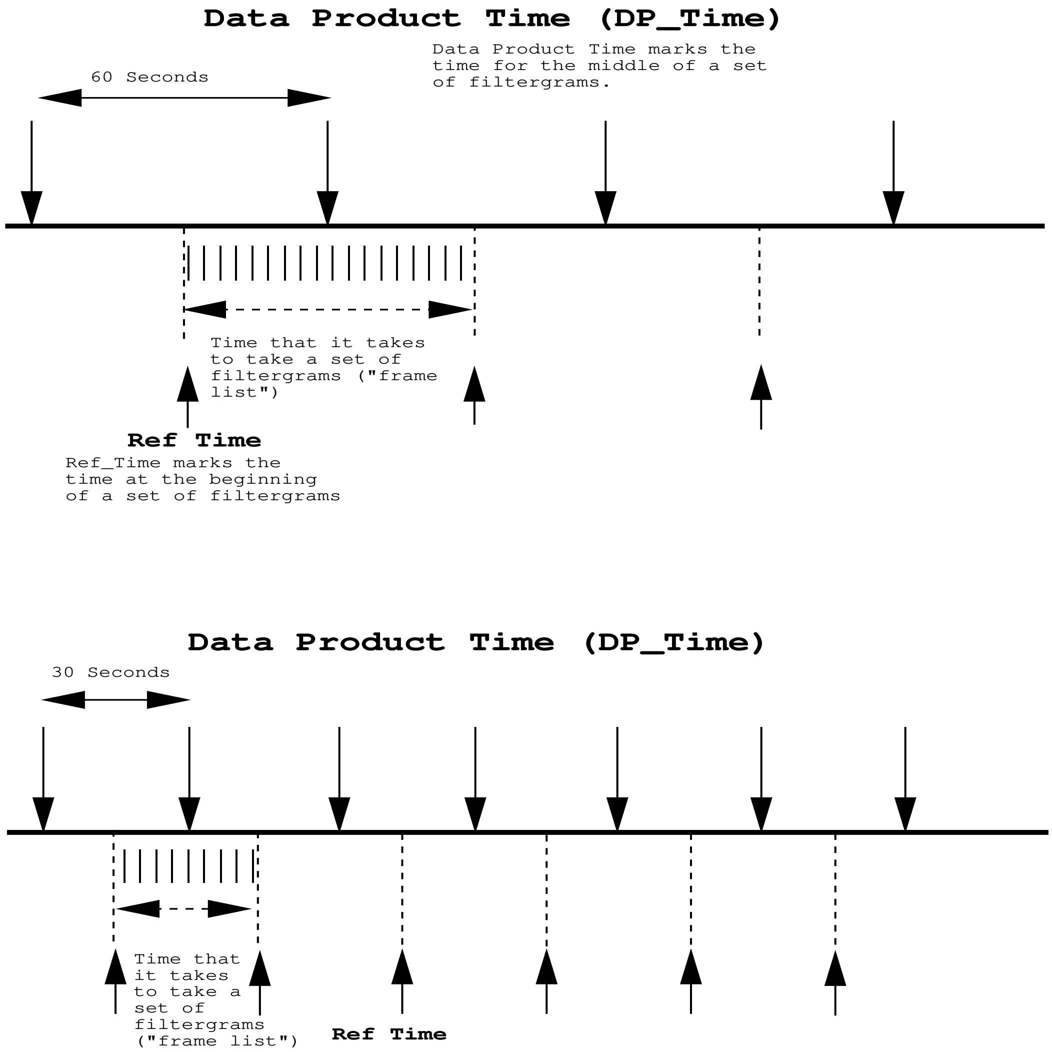

Figure 12: Time marks and their relations

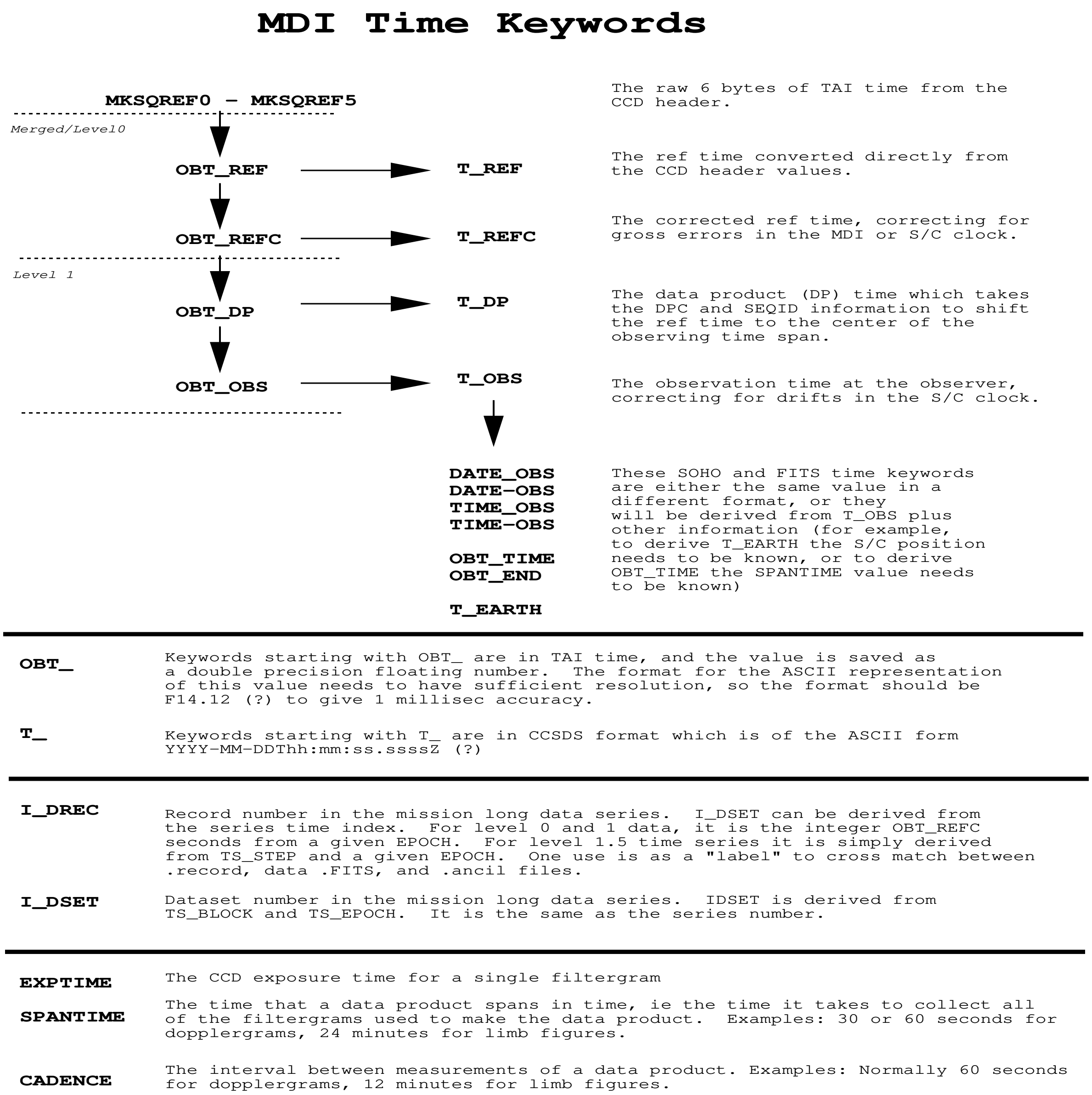

Figure 13: Time keywords

should represent an average of the particular quantity over the TAI minute containing the Data_PRoduct_Time. The Data_Product_Time will be between 7.5 and 15 seconds after the Reftime. Campaign Data_Product_Times will be about 30 seconds offset from the structure program DPTs. There will be some campaigns and some structure data with special cadences of something like 40 seconds for exploration of the high frequency part of the spectrum.

- **Ground_Offset_Interval** – The offset that should be added to the Data_Product_Time to find the time at which a center-of-Earth observer would have observed the same quantity. This will be in the range of 4 to 6 seconds during the mission. Since the MDI observables are usually averages over 30 or 60 seconds it is usually quite sufficient to use 5.0 seconds as a better than needed estimate. It is possible that we will bring filtergram subrasters and may actually want to be able to know the time to less than a second for comparison with ground based observations, although such a need is outside the present SOI science objectives.
- **DPT_UTG** – The DPT with the Ground_Offset time expressed in UT. This is the time useful for comparing phases of oscillations with GONG or other ground based measurements. Note however that if attempts are made to compare phases of high-k waves where the wavelength is only a few times the center of disk pixel dimension, that the actual SOHO location must also be considered since SOHO will be off the Earth-Sun line by about a low-res pixel.
- **????** – The telemetry data is labeled by DSN UT times which are correlated to LOBT in an unknown way.

Note: The WSO times are always in UT seconds from 1 Jan 1601 and completely ignore the issues of leap seconds since the WSO clock has never been accurate enough to care.

G Details About the Contents of the Level 0 .INFO Files

H Details About the Contents of the Level 1 .INFO Files

I Details About the Spacecraft Attitude Files Provided by EOF

Source: Memo to SOWG from C. St.Cyr Dated 13-Jun-94.

- This memo describes further details of the Attitude and Orbit files based on recent discussions with Flight Dynamics and Central Data Handling Facility personnel.
- All files from CDHF will have SFDU.
- All files from CDHF will be Common Data Format (CDF), except for the Full Time Resolution Attitude File which will be ASCII.
- VMS versions of CDF documentation can be found at nssdca.gsfc.nasa.gov in pub/cdf. Unix versions of CDF documentation can be found at ncgl.gsfc.nasa.gov in pub/cdf.

I.1 Predictive and Definitive Orbit File

The CDHF receives orbit data weekly or biweekly from the FDF for the SOHO mission, depending on the state of the mission. Seven days or fourteen days of definitive orbit and 42 days of predictive orbit data are received. The position (X,Y,Z) and motion (Xdot,Ydot,Zdot) of the spacecraft are tabulated every 10 minutes.

The original orbit data sent by FDF to CDHF is in geocentric inertial (GCI) coordinates for the J2000 system. The position of the Sun in GCI is also given on a daily basis. For the spacecraft, the CDHF generates the corresponding geocentric solar ecliptic (GSE) and geocentric solar magnetospheric (GSM) coordinates. CDHF also transforms the spacecraft position and motion to a Heliocentric Ecliptic coordinate system.

During the mission, the most recent Predictive and Definitive Orbit File will be available online at CDHF and at the ECS.

The Definitive Orbit File that will be part of the distribution data for the PI teams will be produced on a DAILY basis.

CDHF will investigate including a daily value for Heliographic Latitude and Longitude of the spacecraft and the Earth.

A "skeleton table" showing the Orbit CDF for Geotail and Wind/Polar has been placed on umbra.gsfc.nasa.gov. When available, we will replace it with the version for SoHO, and we will send out an email notification.

A "sample" of the CDF Daily Orbit file using the above skeleton table with representative data for SoHO should also be available on umbra.gsfc.nasa.gov in the near future.

There will be two (2) SoHO Attitude Files. Both will be produced on a DAILY basis.

I.1.1 Full Time Resolution Attitude File (ASCII)

The full time resolution definitive attitude file as downlinked from the spacecraft contains pitch and yaw offsets from Sun-center (5 times per second) and roll angle offsets from the projection of the Sun's North Pole (one time per second). This is called the spacecraft-body-frame coordinate system.

At the March SOWG, several experimenters expressed a need for the full temporal resolution of the attitude telemetry. That information will also be available without statistical computation for those experiments as distribution data from DDF. This will be called the Full Time Resolution Attitude File. These data will remain in spacecraft-body-frame coordinates, and no further transformations will be performed.

I.1.2 Definitive Attitude file:

At the March SOWG the experimenters agreed that the distribution file should be averaged over 10-minute periods (to correspond to the Orbit files but the times may not be synchronized) and contain some other statistical information about that 10-minute duration (max and min for each value, and standard deviation from the average during that 10 minutes). The CDHF will generate those averages and also provide the transformations of this data to the corresponding GCI, GSE, and GSM coordinates.

The coordinate transformations for the Definitive Attitude File will be based on data from the Predictive Orbit File. This was necessary because of the requirement to produce key parameters within 24 hours of receipt of the observations; hence the key parameter data is based on predicted orbit information. Therefore, during the time period following orbital manuevers when the predictive and definitive orbit data may not match, CDHF may delay processing of these files until a definitive orbit solution is provided by Flight Dynamics.

A "skeleton table" of the Attitude CDF for SoHO is not yet available. When we receive it, we will place a copy on umbra and send out an email notification.

I.2 Support for re-orientation of spacecraft as needed

Source: Memo to SOWG from C. St.Cyr Dated 16-Jun-94.

The SoHO PI's have agreed to roll the spacecraft 90 degrees for a duration of 12-24 hours every six months. This manuever will be necessary to obtain certain observations of the Sun's polar coronal holes. Other roll manuevers may be necessary for instrument calibration.

It is possible that the spacecraft may be required to slew some distance away from its nominal Sun-center position in order to observe celestial objects for purposes of instrument calibration. The feasibility of slews off the Sun-center pointing remains subject to engineering and safety analysis. As such, the magnitude and duration of these slews is TBD.

Support for rolls or offpoints would be requested at least 4 weeks in advance of the event, and FDF would provide planning information at least two weeks in advance of the event.

I.3 Ancillary planning data as needed

Source: Memo to SOWG from C. St.Cyr Dated 16-Jun-94.

The SoHO investigators will require positional information for planets, asteroids, comets, or stars. This information will be used to plan observing sequences needed to calibrate instruments, and for analyzing and describing other scientific phenomena.

The format will be Sun-centered tracks for transits or occultations of the object.

The field-of-view will be two (2) degrees, centered on the Sun.

Points along the target track should be tabulated every hour, or as appropriate for the object in question.

These plots would be requested at least 4 weeks in advance of the event, and FDF would provide the plots at least two weeks in advance of the event.