Helioseismic and Magnetic Imager for Solar Dynamics Observatory

JSOC Processing, Analysis, Archive and Distribution Plan

SU-HMI-S021

CDR Version - 16 November 2004

Stanford University Hansen Experimental Physics Laboratory and Lockheed-Martin Solar and Astrophysics Laboratory
Overview

This document presents the JSOC Science Data Analysis, Processing, Archive and Distribution Plan. It describes the design and implementation of the JSOC for both the hardware and software architecture.

This document is organized in terms of appendicies for each topic, starting with the JSOC Requirements Document in Appendix A.

The Joint Science Operations Center (JSOC) will implement a Ground Data System (GDS) to manage the essentially continuous stream of data from
the HMI instrument, at the rate of 55 Mbps, and from the AIA instrument, at the rate of 70 Mbps, for the duration of the mission.

The data will be converted to standard formats, calibrated to physical observables, and key higher level scientific products will be produced. The GDS will capture the instrument data over dedicated lines from the Data Distribution System (DDS) located at the White Sands Complex in New Mexico. The GDS will support any mission operations as required, provide key forecast and planning information, manage the internal data flow necessary to support both pipeline analysis and research, provide high-quality and easily usable data to the scientific community, and provide a safe long-term archive for the key data products.

The DDS/MOC/SOC block diagram is shown in Figure 1.

![DDS/MOC/SOC Data Flow Diagram](image)

Figure 1.

The overall data flow concept and rates are shown in Figure 2.
Appendix A: JSOC Requirements

(This Appendix is at the end of this draft of the document.)

JSOC Science Data Processing Requirements

Appendix B: Telemetry Input

At the first level is data input from the White Sands Data Distribution System (DDS). This data is delivered over two OC3 lines for AIA and HMI data respectively. The lines can be switched to either of two redundant Data Capture Front End systems. The data will be sorted and depacketized with the ability to request data from the DDS to fill in gaps. The resulting ordered raw telemetry will be permanently archived within 30 days. We expect to maintain most recent 30 days of raw data online. A tape robot will create a copy of input tlm files for offsite storage. Telemetry data is also passed to the back end server where it is again archived to tape and cataloged in a database. Telemetry data is not marked delete pending until an ack is received from both the offsite location and the backend archiving.
The data is copied from the DDS to the SOC via a ftp like transfer. The process is represented in Figure 3 "DDS/SOC Data Exchange"

Every minute the DDS transfers an approximately 400 MB HMI and a 500 MB AIA .tlm file consisting of the stream of compressed VCDUs, less the R/S Check Symbols, to the SOC ftp directory. After each .tlm file, a .qac file is written containing the name, size and TBD quality information of the .tlm file. When the SOC detects the .qac file it ingests the .tlm file and marks this file as received. Every hour the DDS sends a data status file (.dsf) containing all the .tlm file names it believes it has transferred in the last hour. The SOC creates an acknowledgement status file (.asf) acknowledging all those .tlm files it has received and verified and marks those it has not for retransmission. Once every 24 hours the SOC sends a .ack file of all the .tlm files it has permanently archived that can now be deleted by the DDS.

Discussion of DDS - SOC data capture:

solar2:/web/hmi/htdocs/development/jim/dds_soc.straw  08May2003
1. The DDS will put and get files on the HMI SOC via an ftp or similar type program (e.g. FASTCopy).

2. The normal SOC directories for file interchange are:

   /dds2soc
   /soc2dds

3. Every minute the DDS writes to /dds2soc a telemetry and quality and accounting file named something like this and in this order:

   HMI_2002.11.14_10:31:00.tlm
   HMI_2002.11.14_10:31:00.qac

4. The .tlm file will be ~400MB and contain HMI VCDUs in time order and without duplication. Real-time delivery will be attempted only once by the DDS as files become available.

5. The .qac file will be an ascii file with TBD quality and accounting information. Two item of information will the name and size in bytes of the corresponding tlm file. The .qac contains an end of file indicator field.

6. The SOC will initiate processing the .tlm file when the proper .qac is received and verified (qac size, tlm name, tlm size, TBD...).

7. Once an hour on the hour the DDS writes to /dds2soc a delivery status file (dsf) named like so:

   HMI_2002.11.14_10:00:00.dsf

This .dsf contains the file names of all the telemetry files that the DDS has that have not yet been acknowledged by the SOC.

The dsf file will contain ASCII text entries as follow:

   , , ,
   will be formatted as shown above.
   will be the size of the file in bytes.
   will be a numerical filed that can have one of three values:
   0 - Delivery Pending; File has not been delivered for whatever reason
   1 - Delivered; DDS thinks file has been delivered, waiting for SOC ack
   2 - Retransmit; SOC requested a retransmit, retransmit qued
   3 - Retransmit Failed
   4 - Retransmit Successful
8. Upon receipt of the .dsf the SOC will check all the Delivery Pending and Delivered files against what it has successfully received and write an acknowledgement status file (asf) in /soc2dds. The asf file will contain ASCII text entries identical in format to that of the dsf file. The field will have one of 4 values:

- 0 - Delivery Pending; File has not been delivered for whatever reason
- 1 - N/A
- 2 - Retransmit; SOC requested a retransmit
- 3 - SOC Acknowledged, SOC acknowledging receipt of file

Files marked for retransmission may have 2 optional fields:

Machine to which retransmissions are to be delivered, must be know by the DDS
Must be writeable by the DDS

If optional fields are absent, data will be delivered to default location.

Files marked for retransmission will be delivered as bandwidth allows. In general, any file name can be added to the asf for the SOC to request a retransmission. The only way that the SOC makes a tlm file retransmission request is via this /soc2dds .asf file. The DDS removes the /soc2dds .asf file after it retrieves it.

9. At the end of each day the SOC will write a .ack file to /soc2dds containing the tlm file names that have been archived successfully on that day. The DDS will not delete a file that has not been so acknowledged but will notify the SOC manager by e-mail that an acknowledgement is still pending. This is done at say 20 days, well before the nominal 30day limit to delete the file. (The 30d delete should not be hardwired. If there is a real problem and disk space and lots of phone calls, the data should not be dropped due to rigidity in cron scripts or the like.)

In summary here are the file types that are exchanged:

<table>
<thead>
<tr>
<th>Extention</th>
<th>Originator</th>
<th>Frequency</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>.tlm</td>
<td>DDS</td>
<td>1minute</td>
<td>HMI VCDUs</td>
</tr>
<tr>
<td>.qac</td>
<td>DDS</td>
<td>1minute</td>
<td>Ascii quality and accounting information</td>
</tr>
<tr>
<td>.dsf</td>
<td>DDS</td>
<td>1hour</td>
<td>DDS delivery status file</td>
</tr>
<tr>
<td>.asf</td>
<td>SOC</td>
<td>1hour</td>
<td>SOC acknowledgement status file</td>
</tr>
<tr>
<td>.ack</td>
<td>SOC</td>
<td>1day</td>
<td>tlm file names succesfully archived</td>
</tr>
</tbody>
</table>
Footnote:
We will definitively know all the HMI telemetry to expect from interpretation of the hk data. Eventually this will be defined to be the definitive means to know what files to expect from the DDS.
All the telemetry data are permanently archived and represent approximately 1300GB per day. The telemetry datasets can be retrieved at any time by referencing the name of the dataset for the desired minute number of the mission, i.e. hmi,tlm,hr[minute#].

Appendix D: Level 0 Processing

Raw data reconstruction involves decompression and reconstruction the .tlm data into the individual filtergrams, with tags for time and instrument configuration information. The logic of the conversion of a tlm packet into an image is shown in Figure 4.

Figure 4.
These filtergrams constitute the level 0 data and will be permanently archived. The HMI Dataset Sequence is shown in Figure 5.

Figure 5.

The level 0 of HMI and AIA represent approximately 1300GB compressed per day. Since these data correspond simply to the raw data, but are easier to use by the next level of processing, we plan to keep them in an online cache for nominally 3 months. They are organized by datasets of 100 filtergrams for each of the two cameras, i.e.

- hmi,lev0,cam1[index#]
- hmi,lev0,cam2[index#]

They can be retrieved from archive media past the nominal 3 month cache time, by reference to the desired dataset name. Additionally, each lev0 filtergram will have keywords of header information of approximately 1.2 GB/day, stored in a database that can be accessed independently of the dataset of the actual filtergrams.

Appendix E: Level 1 Processing

The level 0 filtergrams are calibrated for exposure time, flat field...
and corrected for missing pixels. Suitable combinations of the calibrated filtergram are to determine line parameters: continuum intensity and equivalent line width, Doppler shifts and Stokes I, Q, U, and V components. These line parameters can be calibrated and in turn be interpreted by suitable inversions as physical observables such as the thermodynamic state variables, line-of-sight velocity, and magnetic field strength and orientation. Images of the line parameters and/or the derived physical observables constitute the Level 1 data. Level 1 data is not normally archived to permanent media, but rather it is recreated as necessary. However, important dataproducts can be cached on-line for the duration of the mission. Additionally, each image of a dataproduct will have keywords of header information stored in a database that can be accessed independently of the actual dataproduct.

Appendix F: Level 2-3 Science Data Products

The scientific data products can be divided into four main areas: global helioseismology, local-area helioseismology, line-of-sight and vector magnetography, continuum intensity data products. For all areas, a pipeline analysis of the Level 1 data will produce Level 2 and Level 3 data products.

Level 2 data are intermediate products that are the results of reorganization of the Level 1 data such as sampling, filtering, projections/mapping, transposition, and spatial or temporal transforms. Broken down by main area the level 2 data products include:

a) Global helioseismology
   * Heliographic maps of Doppler velocity
   * Spherical Harmonic Time series
   * Mode frequencies and splittings
b) Local-area helioseismology
   * Tracked tiles of Doppler velocity
   * Local wave frequency shifts derived by ring diagram analysis
   * Wave travel times derived by time-distance analysis
   * Wave phase shift maps derived by helioseismic holography
c) Line-of-sight and vector magnetography
   * Full-disk averaged maps of Stokes parameters
   * Tracked tiles of Stokes parameters and/or vector magnetograms
   * Vector magnetograms
d) Continuum intensity data products
   * Tracked full-disk 1-hour averaged continuum maps
   * Solar limb parameters
   * Brightness feature maps

Level 3 data represent the results of scientific model analysis, such as helioseismic mode fits, mode inversions, magnetic and velocity field reconstruction, and feature identification. Broken down by main area the level 3 data products include:

a) Global helioseismology
   * Internal rotation and large scale flows
   * Internal sound speed
b) Local-area helioseismology
   * Full-disk maps of velocity and sound speed at depths of 0-30Mm
   * High resolution maps of velocity and sound speed near active regions at depths of 0-30Mm
   * Deep focus maps of velocity and sound speed at depths of 0-200Mm
   * Far-side activity index

c) Line-of-sight and vector magnetography
   * Line of sight magnetic flux maps
   * Full-disk vector magnetic field, filling factor, and other thermodynamic parameters
   * Coronal magnetic field extrapolations
   * Coronal and Solar wind models

d) Continuum intensity data products
   * Brightness images

The principal data flows and products are summarized in Figure 6.

Figure 6.

Level 2 and 3 data are not normally archived, but generated on demand as they are needed. If the higher level data products are not archived, documentation of the algorithms, of the actual code, and of the calibration data used to create them from lower level data will accompany the higher-level data products as ancillary information. This will typically include version and configuration information for a pipeline analysis module in addition to references to the lower level data products and calibration data needed to recreate the higher level data product in question.

Appendix G: Data Archive  

The approximate data volume of archiving is shown in Table 1. HMI Data Archive.

All the raw telemetry data are permanently archived, as are the lev0 filtergrams. Normally higher level number data products are reproduced as needed from the level 0 filtergrams although some higher data products can be archived as deemed appropriate. Whenever a dataprodut is requested in a processing pipeline the system queries the database to determine if the dataset is on-line and where on the disk storage it is located. If the dataset is not on-line, it can optionally automatically be brought on-line and the requesting program is suspended until this occurs.

The raw telemetry data is archived in the front-end system for eventual offsite storage. The telemetry data is passed to the back end server where it is again archived to tape and cataloged in a database. Telemetry data is not marked delete pending until an ack is received from both the offsite location and the backend archiving
Appendix H: Data Distribution

Data distribution is handled through data export request web pages that will allow data selection by time(s) or type(s). Data products are retrieved from resident disk, archived media, or recreated as required and are made available to the requestor in their indicated storage or via web ftp protocol. All HMI produced data will also be made available via the Virtual Solar Observatory (VSO) project.

Appendix I: Pipeline Infrastructure Development

Appendix J: Pipeline User Interface (PUI)

Appendix K: Database Data Catalog

Develop schema for database tables and relationships to allow dataset name resolution and storage allocation. Develop supporting libraries.

In particular the Oracle DB handles the control of the processing metadata as well as a hierarchical storage management function. Each dataset created by production or user programs is stored as a row in a DB table. The row contains information such as dataset name, the creation information of time, user, program name, CM version, bytes, time stamps of data, program or pipeline warnings or comments, as well as dataset status and location on disk and tape. Separate DB tables implement the storage management functions of initial dataset space allocation, dataset retention parameters, and storage de-allocation and garbage collection. By default, storage is allocated on local disks to prevent NFS overhead, unless overridden by user request. This feature will become obsolete in the expected HMI SAN environment.

Here are the expected table schema:

rem Table_Name: DS_NAMING
rem Entry per data series to define the data series.
create table DS_NAMING (  
  PROG          VARCHAR2(20) NOT NULL,  
  LEVL          VARCHAR2(20) NOT NULL,  
  SERIES        VARCHAR2(20) NOT NULL,  
  CONFORM       VARCHAR2(8) NOT NULL,  
  EXT_DESC      VARCHAR2(80),  
  BLOCK         VARCHAR2(8) NOT NULL,  
  SER_TYP       VARCHAR2(16) NOT NULL,  
  LOG_TBL       VARCHAR2(32),  
  MML_DS        VARCHAR2(80),  
  DESCR         VARCHAR2(80),  
  FULL_DEFINITION VARCHAR2(2000),  
  GROUP_ID      NUMBER(4),  
constraint pk_dsnelaming primary key (prog, levl, series)  
);  
commit;  
rem Estimated aver row size 2372 bytes.
create table EPOCH_TABLE (
  SER_TYP               VARCHAR2(16) NOT NULL,
  EPOCH                  VARCHAR2(32),
  DESCR                  VARCHAR2(80),
constraint pk_epoch primary key (ser_typ)
);
commit;
rem Estimated aver row size 134 bytes.

create table DSDS_MAIN (
  PROG_NAME              VARCHAR2(20) NOT NULL,
  PROG_NUM               NUMBER(8) NOT NULL,
  LEVEL_NAME             VARCHAR2(20) NOT NULL,
  SERIES_NAME            VARCHAR2(20) NOT NULL,
  SERIES_NUM             NUMBER(20) NOT NULL,
  SVC_VERSION            VARCHAR2(16),
  SVC_NAME               VARCHAR2(80),
  LEVEL_NUM              NUMBER(8) NOT NULL,
  ONLINE_STATUS          VARCHAR2(5),
  ONLINE_LOC             VARCHAR2(80),
  ARCHIVE_STATUS         VARCHAR2(5),
  ARCH_TAPE_ID           NUMBER(20),
  WORK_TAPE_ID           NUMBER(20),
  SAFE_TAPE_ID           NUMBER(20),
  HISTORY_COMMENT        VARCHAR2(60),
  BYTES                  NUMBER(20),
  DS_INDEX               NUMBER(38) CONSTRAINT unq_dsdsmain UNIQUE NOT NULL,
  CREATE_UID             NUMBER(38) NOT NULL,
  CREAT_DATE             DATE,
  TAPE_POSITION          VARCHAR2(15),
  TAPE_FILE_NUM          NUMBER(5),
  WORK_TAPE_DATE         DATE,
  T_FIRST                VARCHAR2(25),
  T_LAST                 VARCHAR2(25),
  USERNAME                       VARCHAR2(10),
constraint pk_dsdsmain primary key (prog_name, prog_num, level_name,
  series_name, series_num, level_num)
);
commit;
rem Estimated aver row size 440 bytes.

create table INGEST_HRSC_LOG (
  STATION_FILE           VARCHAR2(255),
  START_SFDU_TIME                NUMBER(15,3),
  END_SFDU_TIME          NUMBER(15,3),
  START_REF_TIME                 NUMBER(15,3),
  END_REF_TIME                   NUMBER(15,3),
  START_SFDU_TIME_STR    VARCHAR2(32),
  END_SFDU_TIME_STR      VARCHAR2(32),
  START_REF_TIME_STR     VARCHAR2(32),
  END_REF_TIME_STR       VARCHAR2(32),
  TAPE_NUM                       NUMBER(20) NOT NULL,
NUM_OF_PKT NUMBER(10),
LEVEL_NUM NUMBER(8) NOT NULL,
DS_INDEX NUMBER(38) NOT NULL,
STATION_FILE_NAME VARCHAR2(60) NOT NULL,
constraint pk_ingesthrsc primary key (station_file_name, tape_num, level_num)
);
commit;
rem Estimated aver row size 381 bytes.
rem============================================================================
rem Table_ Name: MERGE_HRSC_LOG
rem merge data info maintained by merge_hrsc_log_svc
create table MERGE_HRSC_LOG (HOUR_NUM NUMBER(8) NOT NULL,
DPC NUMBER(38) NOT NULL,
DPC_HEX VARCHAR2(8),
NUM_OF_DPC NUMBER(8),
LEVEL_NUM NUMBER(8) NOT NULL,
TAPE_NUM NUMBER(20),
DS_INDEX NUMBER(38) NOT NULL,
constraint pk_mergehrsc primary key (hour_num, dpc, level_num)
);
commit;
rem Estimated aver row size 44 bytes.
rem============================================================================
rem  partn_alloc
create table partn_alloc
create table partn_alloc (wd varchar(80) not null,
u_id number(38) not null,
status number(5) not null,
bytes number(20),
ds_index number(38) not null,
effective_date varchar(10),
GROUP_ID NUMBER(4),
safe_id number(4),
archive_substatus number(5) );
commit;
rem Estimated aver row size 138 bytes.
rem============================================================================
rem  partn_avail
create table partn_avail
create table partn_avail (partn_name varchar(80) not null,
total_bytes number(20) not null,
avail_bytes number(20) not null,
pds_set_num number(4) not null);
commit;
rem Estimated aver row size 108 bytes.
rem============================================================================
rem  open_pe
create table open_pe(
u_id number(38) not null,
pe_id number(20) not null
);
commit;
rem Estimated aver row size 36 bytes.
rem============================================================================
rem To create a seq num generation for create_uid in dsds_main
rem Start with 10,000 so that there may not be any conflict when we ingest
rem mdi_ground data. Hopefully 10,000 is a good est.
create sequence dsds_seq
  increment by 1
  start with 10000
  nomaxvalue
  nocycle
  cache 10;
rem============================================================================
rem To create a seq num generation for ds_index in dsds_main
create sequence ds_index_seq
  increment by 1
  start with 1
  nomaxvalue
  nocycle
  cache 10;
rem============================================================================
rem tape
create table tape(
  barcode          number(20) not null,
  avail_blocks     number(20) not null,
  last_file_num    number(20) not null,
  group_id                 number(4) not null,
  constraint pk_tape primary key (barcode)
);
commit;
rem Estimated aver row size 39 bytes.
rem============================================================================
rem ingest_5k_log
create table ingest_5k_log(
  day_num number(10),
  dpc             number(38),
  dpc_hex         varchar2(8),
  num_of_dpc_segs number(8),
  LEVEL_NUM      NUMBER(8) NOT NULL,
  DS_INDEX       NUMBER(38) NOT NULL,
  constraint pk_ingest5k primary key (day_num, dpc, level_num)
);
commit;
rem============================================================================
rem a view dsds_test on the base table dsds_main for historical reasons. This
rem view would eventually go away once all the code is fixed.
cREATE VIEW dsds_test
  AS SELECT * FROM dsds_main;
commit;
rem============================================================================
rem lev1_control_table
create table lev1_control_table(
  dpc                             varchar2(10) not null,
  cal_program             varchar2(80) not null,
  cal_ds          varchar2(240) NOT NULL,
  cal_param       varchar2(240),
  t_start         varchar2(25) NOT NULL,
  t_stop          varchar2(25) NOT NULL,
  spare_str       varchar2(80),
  spare_num       NUMBER(38)
);
commit;
Appendix L: Database Keyword Catalog

The JSOC data is organized as a collection of data series, each defined as a series of logical data records with identical structure. Each logical data record consists of a set of observables, e.g. images or timeseries of observed quantities, as well as a set of keywords (meta data) describing the observables and their quality and processing history. While the observables will typically be kept in standard file formats on secondary (disk) or tertiary (tape) storage, the keywords will be stored in a relational database system.

The JSOC catalog will maintain a set of global tables that define the structure of data records belonging to each series, and contain information shared by all record of the series, such as tape storage group, default online retention time and whether records from the series should be archived to tape.

An example of what these tables might look like is shown in Figure 7 below:
The "master_series_table" contains a list of all JSOC data series with descriptions of the series, information about when and by who they were created, in addition to information about the storage policy for the series.

The "master_keyword_table" contains a list of all keywords for all JSOC data series. Each row in the table describes things like the name of the keyword, the data type, default output format and physical unit.
of its value and whether the keyword is "inherited" by following a link to another data record or whether it is stored as a simple value in the "series table", which is the main table holding actual the keyword values for all records belonging to it (see below). Additional information might include whether the database should maintain an index on this keyword value. This is done to speed up database queries with conditions involving this keyword value.

- The "master_link_table" contains a list of all links for all JSOC data series. Each link has a name and a target series to which they point, and can be either simple, i.e. pointing to a specific data record in the target series defined by (serial number, version), or it can be a "query link". A "query-link" is can be represented as (link name, series name, query) tuple. The intention of the later form is that the link can be bound/resolved dynamically by evaluating the query condition at any time. This can for example be used to create a link that automatically resolves to the most recent data record of a data series, e.g. the most recent version of a calibration table. The syntax used in query links is TBD.

- The "master_observable_table" contains a list of the observables for records belonging to JSOC data series. In the example an observable is described by its name, type, physical unit, dimensions and the internal storage protocol used when accessing the observable data on secondary storage.

The master tables above merely describe the structure of data records belonging to data series and are therefore reasonably small. The number of records is proportional to the number of data series. The bulk of the JSOC catalog is a set of tables, one for each series, containing the actual keyword values for all data records. In addition a special sequence table is maintained for each series. A sequence is a special database table that is used to generate unique serial numbers.

The two tables "hmi_fd_v" and "hmi_fd_v_seq" below illustrate what the main and sequence tables for the series "hmi_fd_v" might look like:
Each row in the table "hmi_fd_v" represent a data record from the series "hmi_fd_v". The first two columns contain serial number (ID) and version which together with the series name uniquely identifies the record within the JSOC environment. The values of simple keywords (T_Obs and D_Mean in the example) are stored in the table along with serial number (ID) and version of objects pointed to by links. For example, the second row in the table represents data record (hmi_fd_v, 1, 1) which contains a link called L1 pointing the record (hmi_fg, 2341, 1) from the hmi_fg series. For this record the query link named Orbit has been resolved to point to the record (sdo_fds, 7, 1) from the series sdo_fds.

Finally, the "hmi_fd_V" table contains for each record a pointer to the data unit in which the observable data for the record is stored. This pointer (here called "DSIndex") is an index into a database table maintained by the DSDS (Data Storage and Distribution System) sub-system. DSIndex can be resolved by the DSDS to a path of a directory containing the data files of the observables.
The JSOC will provide a software API that allows programs written in C, Fortran and IDL to insert new data record, read and update the values of keywords and links and access the observable data.

Currently a prototype is being developed, which contains the functions listed in the C API below. Notice that many of the listed functions are fairly low level and will never have to be called by somebody implementing, e.g., a science pipeline module.

/************** JSOC Environment functions **************/

/* - Open authenticated data base connection. 
   - Retrieve master series lists. 
   - Build hash table over seriesnames. The series templates 
     will be built on demand by querying the master keyword, link 
     and observable tables. 
   - Initialize datarecord cache and hash table. */ 
JSOC_Env_t *jsoc_initialize(const char *host, const char *user,  
                               const char *password, const char *dbname);

/* - If commit_dirty==1 then commit all modified datarecords in the cache 
   to the database. 
   - Close database connection and free JSOC data structures. */ 
int jsoc_shutdown(JSOC_Env_t *env, int commit_dirty);

/* Commit all modified datarecords in the cache to database. */ 
int jsoc_commit_dirty(JSOC_Env_t *env);

/************** Datarecord cache operations. **************/

/* Return a cache slot to the free list and update firstfree. */ 
void jsoc_env_release_drcache_slot(JSOC_Env_t *env, int index);

/* Get the index of the first free slot in the 
   dr cache and mark it used. If the cache is full 
   double its size. */ 
int jsoc_env_get_drcache_slot(JSOC_Env_t *env);

/* Add or remove datarecords from the JSOC environment. */ 
int jsoc_env_remove_dr(JSOC_Env_t *env, JSOC_Datarecord_t *dr);

/************** Data record functions **************/

/* Retrieve a data record from the database 
 presumed */ 
JSOC_DataRecord_t *jsoc_dr_retrieve(JSOC_Env_t *env, const char *seriesname,  
                                      int id, int version);
int jsoc_dr_commit(JSOC_Env_t *env, JSOC_DataRecord_t *dr);

JSOC_DataRecord_t *jsoc_dr_follow_link(JSOC_Env_t *env, JSOC_DataRecord_t *dr, const char *linkname);

JSOC_DataRecord_t *jsoc_dr_query(JSOC_Env_t *env, const char *seriesname, const char *condition);

int jsoc_dr_assign_next_id(JSOC_Env_t *env, JSOC_DataRecord_t *dr);

JSOC_DataRecord_t *jsoc_dr_new(JSOC_Env_t *env, const char *seriesname);

JSOC_DataRecord_t *jsoc_dr_template(JSOC_Env_t *env, const char *seriesname);

JSOC_DataRecord_t *jsoc_dr_allocate(JSOC_Env_t *env, const char *seriesname, int *cache_index);

int jsoc_dr_free(JSOC_Env_t *env, JSOC_DataRecord_t *dr);

int jsoc_dr_copy(JSOC_Env_t *env, JSOC_DataRecord_t *dst, JSOC_DataRecord_t *src);

int jsoc_keyw_get_char(JSOC_DataRecord_t *dr, const char *keyw_name, int *value);
int jsoc_keyw_get_int(JSOC_DataRecord_t *dr, const char *keyw_name, char *value);
int jsoc_keyw_get_float(JSOC_DataRecord_t *dr, const char *keyw_name, float *value);
int jsoc_keyw_get_double(JSOC_DataRecord_t *dr, const char *keyw_name, double *value);
int jsoc_keyw_get_string(JSOC_DataRecord_t *dr, const char *keyw_name, char **value);

int jsoc_keyw_set_char(JSOC_DataRecord_t *dr, const char *keyw_name, char value);
int jsoc_keyw_set_int(JSOC_DataRecord_t *dr, const char *keyw_name, int value);
int jsoc_keyw_set_float(JSOC_DataRecord_t *dr, const char *keyw_name, float value);
value);  
int jsoc_keyw_set_double(JSOC_DataRecord_t *dr, const char *keyw_name, double value);  
int jsoc_keyw_set_string(JSOC_DataRecord_t *dr, const char *keyw_name, char **value);  

/* Use keyword hash table to quickly locate the data structure  
   for specific keyword. */  
JSOC_Keyword_t *jsoc_keyword_lookup(JSOC_DataRecord_t *dr, const char *keyw_name);  

******* Link functions *********/  
/* Define a simple query by specifying the data record pointed to. */  
int jsoc_link_set_simple(JSOC_Env_t *env, JSOC_DataRecord_t *dr, const char *link_name,  
 const char *target_series, int id, int version);  
/* Define a query-link by specifying the target series and query. */  
int jsoc_link_set_query(JSOC_Env_t *env, JSOC_DataRecord_t *dr, const char *link_name,  
 const char *target_series, query);  
/* Bind a query-link to a specific datarecord by evaluating the link query  
   on the target series table. Notice that this function is just a no-op  
   when applied to a simple link. */  
int jsoc_link_resolve(JSOC_Env_t *env, JSOC_Link_t *link);  
/* Return list of all datarecordr linked to *dr. */  
JSOC_DataRecord_t *jsoc_dr_get_all_links(JSOC_DataRecord_t *dr);  

Appendix M: Data Structures  
Appendix N: Data Storage System  
Appendix O: Archive Media Server  
Appendix P: Data Quality Management  
Appendix Q: Flight Dynamics Products  

The Flight Dynamics Products are available from the MOC by ftp transfer.  
* The Flight Dynamic Products data shall be further processed  
   to create the following ancillary data values to be use for the  
   Science Data Products.  
   - Heliographic Latitude of the observer's disk center.  
   - Carrington rotation number for Heliographic Longitude  
     of the observer's disk center.  
   - Distance of observer to Sun center in AU.
- Heliographic Longitude of the observer's disk center.
- Apparent semi-diameter of Sun in arc seconds from HMI.
- "Northward" velocity of observer in m/s. Positive is in direction of solar north.
- Radial velocity of observer in m/s. Positive direction is away from the Sun.
- "Westward" velocity of observer in m/s. Positive is in direction of Earth's orbit.
- Original position angle of the solar north pole measured eastward (counterclockwise) from plate "north" (degrees).

The following Flight Dynamic Products will be used to create ancillary data values.

- Predicted Orbital State Vectors (Geocentric)
- 2-Line Elements
- Computed Orbit Solution
- Attitude

* The Flight Dynamic Product data shall be used to create a list of events to help determine the quality of the science data. Some of the events to capture and monitor from this data are as follows.

- SDO ESR
- Momentum Management
- Guide Star Loss
- Off Nominal Roll
- HGA Blackout
- Eclipse

The following Flight Dynamic Products will be used to create these data values.

- Maneuver Command Data File
- Maneuver Planning File
- Long Range Maneuver Planning File
- Maneuver Summary
- Predicted HGA View Periods of SDOGS
- Predicted Eclipses
- Long Range Predicted Eclipses
- Predicted RFI
- Celestial Bodies in Instrument FOV

Appendix R: Integration & Test

With the provision of injecting known telemetry data into the SOC front end, a complete test of data flow and integrity can be performed for each stage of the processing. Standard regression test suites and validation procedures will be developed to verify processing at each stage through all the system additions and revisions. Throughput and load balancing tuning will be performed.

To generate a tlm file that egse can use as simulated input as if from the DDS:
As user production:
cd /home/production/STAGING/script/egse
#OLD 1024x1024 fits image
#hmicomp -C 4 -L 0 5 fd_V_01h.60000.0001.fits hmicomp.compressed_output

#Use Rasmus' 4096x4096 fits file HMI_L1_SIM.fits and hr_I0_4096.fits
hmicomp -C 5 -L 0 5 HMI_L1_SIM.fits hmicomp.compressed_output
hmicomp -C 5 -L 0 5 hr_I0_4096.fits hmicomp2.compressed_output

#Generate 30 images to represent a minute of data (OLD)
gentlm 2048 1 30 hmicomp.compressed_output > gentlm.out

#Generate 16 images of each image to represent a minute of data
gentlm2 2048 1 16 hmicomp.compressed_output hmicomp2.compressed_output > gentlm2.out

NOTE:
gentlm first_vcdu_counter_value 
  first_fsn_value 
  number_of_copies 
  compressed_single_image > tlm.out

In egse:
`cp gentlm.out $DIRDDS/$tlmfile`;

NEW: 11May2004 gentlm2

% gentlm2 vcdu_counter fsn ncopies img1 img2 > tlmstream

Writes ncopies of img1 and img2 with packets interleaved.

If the numbers of packets in each img are not equal,
the last abs(n1-n2) packets are not interleaved.

########################################################################
This perl filter should be run after gentlm to chop off
a 10-byte header and a 2-byte trailer from every packet:

#!/usr/local/bin/perl

while (1788 == read(STDIN,$buf,1788)) {
  $buf2 = substr($buf,10,1776);
  syswrite(STDOUT,$buf2,1776);
}

Appendix S: Software Development Environment  (Top)

The software development environment will consists of linux
workstations with commonly mounted document and source code
partitions. The development will be predominantly in C and perl, with
possible additions of python, php, and java. The Intel icc will
be the likely choice for the C compiler.

All development source code is under a Configuration Management (CM)
system. 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 minimal conflict. RCS is a widely user version control system. The BCS based CM was used successfully throughout the MDI data system development. We will use cvs for JSOC.

CVS REPOSITORY:
--------------

There is a master cvs repository on sunroom:/home/cvsuser/cvsroot. It currently contains an EGSE repository which is the current managed version of all EGSE code. (This is like what we call our CM baseline in /home/soi/CM.) This will be the central repository for all the hmi#/aia# machines.

SANDBOX:
--------

You need to make your private working copy of the repository. This is called your sandbox. The sandbox is like what we called our STAGING area for MDI. Unlike with MDI, it is not a bunch of links to the repository, but rather your own read/write set of files. You will have a seperate sandbox on each hmi#/aia# machine. Remember, the /home is separate on each machine. There is no NFS /home for the hmi#/aia# machines.

You must do a make in your sandbox to get any executables. The bin/_linux4 dir is empty in the repository. The lib/_linux4 in the repository and your sandbox have the libs that you need. It is your option to remake them or not. The way to do a master make is:

```
> cd ~/cvs/EGSE/src/genmake.d
> make all_linux
```

There is currently a sandbox already made for user production on both hmi0 and hmi1.

SANDBOX on hmi#/aia#:
---------------------

Here's how to make your sandbox on a hmi#/aia# machine:
No longer source .bcsinit in your user setup.
Make sure this is done:

```bash
setenv MKROOT /home/user/cvs/EGSE
setenv STAGING /home/user/cvs/EGSE
setenv CVSROOT :ext:sunroom.stanford.edu:/home/cvsuser/cvsroot
setenv CVS_RSH /usr/bin/rsh
setenv CVS_BINARY_ROOT /home/cvsuser/cvs_binary_root
```

and these are in the path instead of the old STAGING:

```
/home/user/cvs/EGSE/bin/_linux4       (i.e. _$MACHINE)
/home/user/cvs/EGSE/script
/home/cvsuser/scripts
```

Logon to the hmi# or aia# machine
> cd $HOME
> mkdir cvs
> cd cvs
> cvs checkout EGSE
> cd EGSE; /home/cvsuser/scripts/set_symlinks.sh

So /home/user/cvs/EGSE is your sandbox. It is in this tree that you will modify/add/delete files and dirs to go into the repository or receive updates from the repository.
(you may have to give your password. uses an ssh client.)
Remember, your sandboxes on different machines are independent. You must resync (see update below) each of them with the repository if you want the latest files. You must do makes in each of them if you want executables.

WORKING IN YOUR SANDBOX:
------------------------

* You cd and edit files in your sandbox as appropriate. When you want to commit your changes to the repository do:

  > cvs commit [file]
  > [sync_binary_dirtree.sh]

  It is only necessary to run sync_binary_dirtree.sh if the binary repository has been updated or new dirs have been added to the source tree.

  You will be asked for any change notes.

* To resynchronize your sandbox with the repository do:

  > cvs update -d

  (the -d will download new directories as well)
  See the cvs man page for all switches and options.

  * To add a file to the repository, first create the file in your sandbox and then do from the dir containing the file:

    > cvs add filename
    > cvs commit

    If you're adding a binary file, use the -kb option:
    > cvs add -kb file.doc

    If you forgot to use the -kb at add time, you can later do:
    > cvs admin -kb file.doc

* To remove a file from the repository, first remove the file in your sandbox and then do from the dir containing the file:

  > cvs remove filename
  > cvs commit

* To determine which files are up-to-date and which need to be committed:
Based on experience with MDI processing of similar data, the GDS hardware configuration is currently planned as a processor farm of 50 quad-CPU compute servers connected via a storage area network (SAN) to 400 TB online storage on RAID arrays and a small number of tape libraries providing another 500 TB of nearline storage, a telemetry handling subsystem, and a database subsystem. The GDS has a dedicated data connection to the DDS at White Sands on the one end, and is protected from the public network behind an internet firewall on the other end. A small number of designated access nodes allow remote monitoring of the GDS and provide data import and export functions.

Figure 8.

The telemetry subsystem consists of two identical hosts in an automatic failover configuration. Each host has local disk space large enough to receive 30 days' worth of raw telemetry although at any given time only one is active. After their integrity has been verified, the telemetry files are permanently archived and copied to disks on the SAN to which the processor farm has direct, read-only access. The entire subsystem is protected by a UPS and will be able to continue operation for up to 60 days without help from the rest of the GDS.

The HMI pipeline software schedules all data production and analysis tasks to be run on the processor farm, which sees the entire online storage as local disk space through a cluster filesystem. By using identically configured servers, tape units and SAN switches, and RAID protection for disk arrays, an adequate level of redundancy and data protection will be achieved.

Initial cost estimate

GDS Hardware Requirement and Cost Estimate (~kehcheng/HMI/gds_cost.txt)

CPU

Currently MDI data production and analysis is done on an 8-CPU SGI 2200 server (SPECfp_rate 26) and a 12-node dual-Xeon cluster (total SPECfp_rate 168) with an overall utilization of less
than 50% (SPECfp_rate 100-). By comparison, HMI data production and analysis requirement is expected to see a 50-fold increase, or SPECfp_rate 5000, based on the estimate that local helioseismology analysis and vector magnetogram inversion, the two dominant tasks in terms of CPU usage, would require a combined processing power of SPECfp_rate 2500.

Of all currently (mid 2003) available hardware platforms, AMD's Opteron line of dual (SPECfp_rate 27) and quad (SPECfp_rate 49) CPU servers offer the best price/performance ratio. Assuming their prices remain constant while their performance doubles over the next 4 years, we figure a cluster of 200 CPUs (100 dual-CPU or 50 quad-CPU servers, or a mixture of both) will be required to do HMI data processing and analysis. The estimated cost is $1 million ($10K per node, including cost of SAN) for an all dual-CPU cluster or $1.5 million ($30K per node) for an all quad-CPU cluster.

Disks
=====
For HMI we plan to keep a 30-day cache of raw telemetry (20 TB), a 90-day cache of filtergrams (90 TB), and observables for the life of the mission (50 TB/yr) on RAID arrays. For a 5-year mission the required total disk capacity is 400 TB.

Currently the best price for a fibrechannel RAID array with IDE disks is about $3.3 per GB (Nexsan ATABeast, 12 TB for $40K). Assuming a factor of 4 improvement in price/performance, we estimate the cost of the disk to be $320K for the first 5 years.

Tape Libraries
==============
For HMI we plan to keep a 90-day cache of raw telemetry (60 TB), a 270-day cache of filtergrams (270 TB), and enough blank tapes for 30 days' archiving needs (60 TB) near-line. This makes the nearline storage requirement about the same as the online disk capacity.

Today a midsize tape library with two fibrechannel LTO-2 drives and 150 tape slots (200GB each) costs about $40K (Exabyte Magnum 20). The fourth generation LTO (or competing formats such as SDLT) is expected to reach 800GB per cartridge by 2007. If tape drive and robotics prices remain constant, a set of 4 libraries will offer approximately 500 TB of near-line storage for $160K.

Cost Estimate Summary
=====================  
<table>
<thead>
<tr>
<th></th>
<th>100 dual-CPU cluster</th>
<th>50 quad-CPU cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 TB RAID arrays</td>
<td>$320K</td>
<td></td>
</tr>
<tr>
<td>500 TB tape libraries</td>
<td>$160K</td>
<td></td>
</tr>
</tbody>
</table>

Total: $1.5 million $2 million
Appendix U: Data System Implementation Plan

The data system implementation consists of four development phases with the following dates and associated events:

Phase 1: Design
Start Date         01 Oct 2002
End Date           30 Sep 2003
Start Event       Contract Start
End Event         Ground System Concept Review

Phase 2: Initial Development
Start Date        01 Oct 2003
End Date          30 Sep 2004
Start Event       Ground System Concept Review
End Event         Ground System Critical Design Review

Phase 3: Prototype and Instrument Test Support
Start Date        01 Oct 2004
End Date          31 Mar 2006
Start Event       Ground System Design Approval
End Event         Ground System Implementation Review

Phase 4: Production System, Final Calibration, End-to-End Test
Start Date         1 Apr 2006
End Date           30 Sep 2007
Start Event       Production System Installation Approval
End Event         End of HMI on-orbit commissioning

The components of the development are as follows:

1.5. Ground Data System Development

1.5.1. System Engineering
H/w and s/w architecture specification development. This defines the major system components and their relationships and interfaces.

1.5.2. Telemetry Handling
The GDS receives HMI telemetry from the Data Distribution System (DDS) located at the White Sands ground station. Telemetry and support files are ftp'd from the DDS to the GDS. The GDS will verify and log receipt of all DDS generated data and will negotiate with the DDS all retransmissions of data as required. All captured data will end up in the permanent GDS data archive and the DDS so informed so that it may mark the data as deletable. A data simulation system will be build to facilitate and test the GDS and DDS development.

Phase 1 - System Design
   Top level description and requirements.
   Data flow and volume concepts.
   H/w and s/w architecture concepts and cost estimates.

Phase 2 - Initial Development
   Detailed design for telemetry capture, verification, logging and archiving.
   HMI SOC interface and support with DDS development.
   Simulate HMI data flow.

Phase 3 - Prototype and Instrument Test Support
   Develop s/w according to design to support simulated HMI data flow.
Support level 0 processing on target like hardware.

Phase 4 - Production System, Final Calibration, End-to-End Test
Full telemetry processing on target hardware.

1.5.3. Pipeline Infrastructure Development
This consists of methods of process control and monitoring and interprocess communications to affect any number of pipeline processing threads for any number of users on whatever computing resources are available. Includes messaging and process control, event generation and scheduling, logging and error reporting and debugging options.

Phase 1 - System Design
Top level description and requirements.
Data flow and volume concepts.
H/w and s/w architecture concepts and cost estimates.

Phase 2 - Initial Development
Detailed design and trade studies of supporting packages. Support limited pipeline for MDI data flow.

Phase 3 - Prototype and Instrument Test Support
Develop s/w according to design with all pipeline features.

Phase 4 - Production System, Final Calibration, End-to-End Test
Full telemetry and science processing pipeline.

1.5.4. Pipeline User Interface (PUI)
A web based gui to allow top level control and status reporting of all production processing across all computing resources.

Phase 1 - System Design
Phase 2 - Initial Development
Detailed design and trade studies for alternate gui tools.

Phase 3 - Prototype and Instrument Test Support
Develop s/w according to design with all status and control features.

Phase 4 - Production System, Final Calibration, End-to-End Test
Support full telemetry and science processing pipeline.

1.5.5. Database Catalog
Develop schema for database tables and relationships to allow dataset name resolution, data header lookup, storage allocation and keyword definitions.

Phase 1 - System Design
Conceptual design of database

Phase 2 - System Development
Detailed design and trade-off studies for different data header handling approaches.

Phase 3 - Prototype and Instrument Test Support
Implement database and some supporting access functions.

Phase 4 - Production System, Final Calibration, End-to-End Test
Full database access support and backup and recovery.

1.5.6. Data Structures
Data Formats and Protocols
Internal and external formats and protocols for data representation and exchange.
Keywords
Define and document data keywords to be used in ancillary data and databases supporting archive search and browse tools. Provide schemata for VSO.

Phase 1 - System Design
Conceptual design of data structure and storage format.

Phase 2 - System Development
Document and implement formats and keywords.

Phase 3 - Prototype and Instrument Test Support
Phase 4 - Production System Implementation

1.5.7. Data Storage System
This assigns and maps on-demand pieces of HMI datasets to disk storage for use by production and user pipelines. Includes data storage allocation and removal and database support for data storage management.

Phase 1 - System Design
- Top level description and requirements.
- Data flow and volume concepts.
- H/w and s/w architecture concepts and cost estimates.

Phase 2 - System Development
- Detailed design.
- Provide limited data storage for MDI data flow.

Phase 3 - Prototype and Instrument Test Support
- Develop s/w according to design with all data storage features.
- Develop level 0 pipeline processing on target like hardware.

Phase 4 - Production System, Final Calibration, End-to-End Test
- Full data storage management implementation.

1.5.8. Archive Media Server
This creates and manages the permanent archive of all relevant HMI datasets. This reads and write to archive media to effect saving and retrieving HMI datasets.

Phase 1 - System Design
Phase 2 - System Development
- Detailed design and trade studies of suitable archive devices and media.

Phase 3 - Prototype and Instrument Test Support
- Develop s/w according to design with read/write to media.

Phase 4 - Production System Implementation
- Full media access, organization and queuing.

1.5.9. Development Environment
This includes providing a consistent set of tools for programming (compilers, debuggers, profiling, etc.), support libraries (numerical, graphics, data structures, I/O, etc.) and associated bindings programming languages, revision and configuration control, managing third party software products. Compilers and support libraries should be configured to make efficient use of hardware resources.

Phase 1 - System Design
- Compile list of development tools, support libraries and commercial/3rd party software required based on what is being used in the current project. Obtain, install and maintain basic development tools to support standalone module prototyping. Determine requirements of configuration control system. Develop plan for integrated environment that supports building, running and debugging pipeline modules.

Phase 2 - System Development
- Implement development environment. System administration.

Phase 3 - Prototype and Instrument Test Support
- Maintain development environment. Obtain and configure hardware specific libraries and development tools for prototype hardware. System administration.

Phase 4 - Production System, Final Calibration, End-to-End Test
- Fine-tune time-critical libraries and compiler flags as needed. System administration.
1.5.10. Data Export
Supports web requests for browsing and export of HMI data products. Various methods of data inspection and requesting are supported.

- Data Display and Export
- Archive Browsing, VSO Development
  Develop tools for locating data in the archive suitable for support of VSO API's
- Archive Export Tools
  Provide API for data export suitable for VSO.
- Real Time Processing Export Tools

1.5.11. Hardware System
Design, assemble, and maintain GDS hardware system (including CPU and storage network) to support telemetry handling, pipeline processing, and data archiving.

- Hardware Requirements and Specifications
- Hardware: Procure and Integrate
- Hardware maintenance and upgrade
- Systems support
Phase 1 - System Design
  Computation and storage requirement estimates
  Computer hardware system specifications
  Hardware system conceptual design
Phase 2 - System Development
  Detailed hardware system specification and design
  Implementation of limited system capable of handling MDI data flow
Phase 3 - Prototype and Instrument Test Support
  Prototype hardware system procurement and integration
  Hardware system testing with prototype pipeline software
Phase 4 - Production System Implementation
  Production system hardware procurement and integration
  Complete hardware testing with production pipeline software

1.5.12. Data Quality Management

1.6. SU Pre-launch Science DA Development

1.6.1. System Engineering
H/w and s/w architecture specification development. This defines the major analysis modules and their relationships and interfaces.

1.6.2. Level-0 modules
Takes the VCDU encapsulated compressed telemetry data and extracts level 0 filtergrams.
Phase 1 - System Design
  Top level description and requirements.
  H/w and s/w architecture concepts and cost estimates.
Phase 2 - System Development
  Detailed design.
Phase 3 - Prototype and Instrument Test Support
  Develop modules to extract the lev0 from telemetry data.
Phase 4 - Production System, Final Calibration, End-to-End Test
  Full lev0 extraction and verification in real time and production modes.
1.6.3. Real Time Processing Modules
Consists of two distinct parts:
  a. Processing for getting semi-calibrated, real time science data products.
  b. Image display of low level data products and diagnostics for monitoring and planning.

Phase 1 - System Design
  Top level description and requirements.
  Identify science data products and diagnostics required.
  H/w and s/w architecture concepts and cost estimates.

Phase 2 - System Development
  Detailed design.
  Provide real time processing simulation using MDI data.

Phase 3 - Prototype and Instrument Test Support
  Develop s/w according to design with all real time processing features.
  Test real time processing modules using ground test data.
  Develop real time processing pipeline on target like hardware.

Phase 4 - Production System, Final Calibration, End-to-End Test
  Full real time pipeline implementation.

1.6.4. Ancillary Data
This involves processing all non-image data received from SDO
including SDO orbit data and HMI house keeping data (temperature,
frame lists, shutter times, command logs etc.). The modules must
extract the data from provided files and populate the headers in the
database for use in subsequent calibration and observable calibration.

  Orbit Data Processing
  Housekeeping Data Extraction (temps, frame-lists, etc)
  Trend Analysis tools development

Phase 1 - System Design
  Top level description and requirements.
  H/w and s/w architecture concepts and cost estimates.

Phase 2 - System Development
  Detailed design.

Phase 3 - Prototype and Instrument Test Support
  Develop s/w according to design with all ancillary data processing features.
  Test ancillary data modules using ground test data.
  Develop ancillary data pipeline processing on target like hardware.

Phase 4 - Production System, Final Calibration, End-to-End Test
  Full ancillary data processing implementation.

1.6.5. Calibration Data Analysis
Develop modules that calibrates the observables.
  Doppler, LOS Field
  Vector Magnetic Field
  Flat Field
  Centering and Distortion and MTF Methods

Phase 1 - System Design
  Top level description and requirements.
  H/w and s/w architecture concepts and cost estimates.

Phase 2 - System Development
  Detailed design.

Phase 3 - Prototype and Instrument Test Support
  Develop s/w according to design with all ancillary data processing features.
  Test ancillary data modules using ground test data.
  Develop ancillary data pipeline processing on target
like hardware.

Phase 4 - Production System, Final Calibration, End-to-End Test
Full ancillary data processing implementation.

1.6.6. Observable Calculation Modules
- Doppler
- Magnetic Field
- Vector Field
- Continuum Brightness
- Quality, Centering, Distortion and MTF Processing

Phase 1 - System Design
Top level description and requirements.
H/w and s/w architecture concepts and cost estimates.

Phase 2 - System Development
Detailed design.
Test observable calculation modules using MDI data.

Phase 3 - Prototype and Instrument Test Support
Develop s/w according to design with observable calculation functionality. Test observable calculation using ground test data.
Develop observable calculation pipeline processing on target like hardware.

Phase 4 - Production System, Final Calibration, End-to-End Test
Full data processing implementation.

1.6.7. Standard Science Product Pipeline
The pipeline modules perform the computation required to turn the four calculated observables, Doppler magnetic fields, magnetic vector fields and continuum brightness into intermediate and high level science data products. It integrates computation of several high level data products, such as internal rotation, far-side activity, dynamic whole-Sun magnetic field maps etc., that have previously been computed by stand-alone analysis (research) codes.

Phase 1 System Design
Develop module specifications, based on existing analysis/research codes and MDI pipeline modules where appropriate. Benchmark existing software to estimate data volume and processing requirements. Develop per-module data (input & output) and processing requirements.

Phase 2 System Development
Implement prototype modules. Simulate HMI processing using MDI data. Work with co-Investigators to develop algorithms for modules where existing techniques are insufficient to meet HMI processing requirements. The results of this should be detailed processing (algorithmic), data format and I/O specifications for all pipeline modules.

Phase 3 Prototype and Instrument Test Support
Implement modules within the HMI development environment according to specifications and integrate with pipeline infrastructure.

Phase 4 Production System, Final Calibration, End-to-End Test
Integrate modules in pipeline. Test complete pipeline with data from ground test.

1.6.8. On-Demand Science Data Product Processing Modules
This provides the HMI processing pipeline with the capability of generating certain data products on the fly. Since some high level data products will only ever be sampled at a sparse set in time and/or space, generating all data products for the whole Sun all the time would be a waste of resources. Coordinate with VSO effort and 1.5.7 to
Phase 1 System Design
Identify science data products to be generated on an on-demand basis. Determine internal and external requirements for on-demand processing system such as query capabilities, processing latency, load balancing and scheduling of on-demand tasks.

Phase 2 System Development
Develop specification for query-processing-export cycle. This includes how to communicate on-demand queries to the processing control of the HMI pipeline system. Define detailed on-demand interface specification.

Phase 3 Prototype and Instrument Test Support
Implement on-demand interface and integrate with pipeline control.

Phase 4 Production System, Final Calibration, End-to-End Test
Integrate on-demand functions with query and export system (i.e. VSO).

Other Issues

Trade Studies:
--------------
H/W: SAN/NAS, clusters home grown/commercial, big iron, batch handlers, ...
Metadata: processing info, keywords/data description
CM: Version control, home grown/commercial
HDF: vs. other options??
PVM: No longer supported. Retain or go to MPI or such?
Module API: Maintain compatability with MDI?
Programming languages
for the modules - C/C++/Fortran9x
for "the glue" - C/C++/Perl/Python
for monitoring and other ops related GUI - IDL/Matlab/Java...?
Do a study of options for both internal and export data formats. Also database organization.
DSDS vs commercial DMF for data storage mgmt.

Open Issues:
------------

Detailed Documents:
-------------------
DDS interface spec
Data formats

***********************************************************************

Last modified @
Joint Science Operations Center - Science Data Processing (JSOC-SDP) Requirements

UNDER DEVELOPMENT

1.0 Introduction
2.0 Telemetry Input
3.0 Telemetry Archive
4.0 Level 0 Processing
  4.1 Image decompression and reconstruction
  4.2 High-rate channel housekeeping processing
  4.3 Storage and Cataloging
5.0 Level 1 Processing
6.0 Level 2-3 Science Data Products
7.0 Data Archive
8.0 Data Distribution
9.0 MOC Operational Reports
10.0 Flight Dynamics Products
11.0 System Infrastructure

1.0 Introduction
This document presents the requirements for the Solar Dynamics Observatory (SDO) Joint Science Operation Center for Science Data Processing (JSOC-SDP) for the Helioseismic and Magnetic Imager (HMI) and Atmospheric Imaging Assembly (AIA) instruments. The JSOC is the ground data processing center located at Stanford University in Palo Alto, California. It has two components, the Science Data Processing (JSOC-SDP), and the instrument commanding (JSOC-CMD). This document deals with the JSOC-SDP, which is sometimes only referred to as JSOC herein.

2.0 Telemetry Input

The JSOC must manage an essentially continuous stream of telemetry data for the duration of the mission.

* The JSOC shall manage a continuous stream of telemetry data from the HMI instrument at 55 Mbps.

* The JSOC shall manage a continuous stream of telemetry data from the AIA instrument at 70 Mbps.

The telemetry is delivered to the JSOC over dedicated lines from the Data Distribution System (DDS) at the White Sands Complex in New Mexico. The data is delivered in telemetry file consisting of spacecraft VCDUs in time order. The VCDUs represent both image data and housekeeping data. The DDS provides the interface between the SDO antenna system and the JSOC for the science data. This interface is detailed in "SDO Interface Control Document Between the SDO DDS and the SDO Science Operation Centers". The ICD describes the protocols for telemetry delivery, validation, and retransmissions.

The JSOC also has an interface with the Mission Operations Center (MOC) at Goddard. Various flight dynamic products, housekeeping and status data are exchanged over socket connections. This interface is detailed in "SDO Interface Control Document Between the SDO MOC and the SDO Science Operation Centers". A context diagram of the interface is shown in Figure 1.
* The JSOC shall comply with 464-GS-ICD-0010 "SDO Interface Control Document Between the SDO DDS and the SDO Science Operation Centers".

* The JSOC shall comply with 464-GS-ICD-0001 "SDO Interface Control Document Between the SDO MOC and the SDO Science Operation Centers".

* The JSOC shall keep metrics on the telemetry streams consisting of VCDU counter gaps, IM_PDU counter gaps, missing sync markers, VC Id mismatch with file name, VCDU time retardation, App id, Pkt Len, time and CRC verification failures.

* The JSOC may request retransmission of any telemetry file up to 30 days old.

* The JSOC telemetry input shall occur on redundant processing nodes and redundant disk storage.

* The JSOC telemetry input machines shall be on a secure network conforming to "!!TBD Security document".

3.0 Telemetry Archive

* All validated telemetry files shall be cataloged by an appropriate name (referred to as a dataset name) that will allow for future access.

* The cataloged datasets shall be capable of being written to permanent media for offline storage.

* Two copies shall be produced on permanent media. One is retained locally, and the other shall be removed for offsite storage.

* The permanent media drives shall be local to the JSOC nodes.

* The JSOC shall retain a 30 day cache of telemetry online.
*Any existing dataset shall be capable of being exported to an external user via user initiated export requests.

4.0 Level 0 Processing

This step converts the raw telemetry data into images that look like they did on the CCD. I.e. the transmission specific details are not needed to interpret the level-0 data and the images are in a form that is available for immediate use. The level-0 data is changed in format and organization but will not have the values changed in a way that can not be removed.

4.1 Image decompression and reconstruction

* The decompression and reconstruction steps outlined below shall comply with with 2H00125 "Functional Specification, Data Compression/High Rate Interface".

* The payload of each science data packet from the high-rate channel shall be extracted from the VCDU, it's individual header fields extracted, decoded to standard data types and checked for errors.

* The compressed data in each packet shall be decoded according to the compression parameters contained in the header. The decompression shall undo the entropy coding and first differencing performed onboard.

* It shall be verified that the final decompressed value matches the value stored in uncompressed form in the final word of the packet.

* It shall be verified that the bitstream in the packet is a valid bitstream for the compression parameters in the header.

* The final decoded pixel values shall be computed by transforming the decompressed values to undo the bitselect and table lookup performed onboard.

* The decoded pixel values shall be copied to an 16 bit signed integer image
buffer dimensioned to hold the complete reconstructed image. The location
to which the pixels from a given packet are copied shall be determined by
header fields indicating the pixel offset count, and the CCD read-out mode
and cropping applied.

* The image reconstruction module shall collect and report the following
information for each image:
- number of packets received
- number of packets decompressed without error
- number of pixels expected (given by read-out mode and cropping table)
- number of pixels decompressed without error
- a "skip" flag indicating whether during decompression a packet was
  received in
    which the pixel offset was larger than the offset of last pixel decoded
    plus one
- a "backup" flag indicating whether during decompression a packet was
  received
    in which the pixel offset was smaller than the offset of last pixel
    decoded plus one
- time of the arrival of the first packet (to be used for timeout and flush of
  incomplete
  images).
* A level-0 image shall be considered complete when the last expected pixel
has been decoded.

* The level-0 processing shall periodically (period TBD) flush incomplete
images older than TDB seconds, which means that they are archived in
whatever state of completeness they are in.

* Missing pixels shall be assigned the value -1, while pixels outside the
cropped region shall be assigned the value -1 (should be
distinguished??).

* Each level-0 image shall be cataloged as a dataset of a single .fitz file with
the filtergram series number (fsn) as its series number index.
4.2 High-rate channel housekeeping processing

* The housekeeping packets that are inserted into the high-rate channel shall be extracted, decoded to standard data types and checked for errors.

* The decoded housekeeping data keywords shall be added to the header of the level-0 image with which they are associated.

* The housekeeping data shall be distributed in two ways to the SOC from the MOC. The housekeeping data will be sent real time over a socket or as a non-real time Level-0 data set(or sometimes called 24 hour data set) file. This is described in detail in SDO ICD between MOC and SOC. The SOC shall process data using both methods.

* The SOC shall receive housekeeping data packets using sockets over TCP/IP in SFDU format.

  - This housekeeping data will be distributed in real time. The latency between the MOC receipt time and reception by the destination SOC will be less than 30 seconds. MOC sends data to SOC socket as soon as there is space on socket. The SOC shall process data from socket.

  - Each SOC can select which packet its wants to receive by specifying a list of VCID/APIDs.

  - The SOC shall access a menu-driven program provided with ASIST in the MOC to select the VCID/APIDs they want to receive in real time. This could be by the SOC in a TBD(to be determined) manner.

  - This menu driven program provided with ASIST in the MOC allow
Playback from the MOC. These requests are not intended for “on-the-fly” playback. The requests for playback will be carried out from SOC. Using the CCSDS SDO Housekeeping telemetry packet. ASIST supports different speeds that a several times faster than real-time data transmission. SOC may use these different speed in a TBD manner.

- The SOC can select start/stop times and playback telemetry over separate sockets. The amount of playbacks will be limited to the communication line capacity. The playback data shall be received by the SOC in the same format as the real-time data.

- All housekeeping telemetry APIDs downlinked on the S-band will be available from the MOC to SOC. The request of the data shall be carried out SOC using the CCSDS SDO Housekeeping telemetry packet.

- The CCSDS housekeeping telemetry packet is defined in detail in the NASA ICD document called SDO ICD between MOC and SOC. The SOC shall process these CCSDS housekeeping telemetry packets from the MOC.

- A Housekeeping telemetry message is formatted as a SFDU. The SFDU is composed of Z header followed by a series of “FANN-Cxx”. This is described in detail in SDO ICD between MOC and SOC document. The SOC shall be able to read and process these messages.

* The SOC shall receive housekeeping data in telemetry files over ftp from the MOC.
- This is non-real time housekeeping telemetry distribution data that
contains
    level-0 data sets and housekeeping data.

- This data is send as soon as the MOC receives data and is sent in parallel with
  the real time housekeeping data.

- Each file contains all packets for a given APID with time stamps from 00:00:00 to
  23:59:59 UTC. The packets of file are time ordered with duplicates removed.

- The SOC shall be able to read and process files using the file name conventions.
  The file naming convention is:
  apid_yyyy_ddd_hh_mm_ss_ver.hkt (add more details here)

- The SIC shall read file data in the following file format. Each telemetry file
  contains a succession of telemetry packets in binary format. Each packet
  contains a primary header, Secondary header and application data.

- Each file will have a separate telemetry header file associated with the .hkt file.
  The file naming convention for the is:
  apid_yyyy_ddd_hh_mm_ss_ver.thr (add more details here)

- The file format of the header file is described as follows. (add more detail here)

**4.3 Storage and Cataloging**

* When a level-0 image is complete it shall be stored in (TBD) compressed format.
* The header keywords describing the level-0 image shall be inserted into the catalog database.

* The level-0 image shall be archived on permanent media.

* The JSOC shall retain a 30 day cache of level 0 images online.

**5.0 Level 1 Processing**

* The level 0 filtergrams shall be calibrated for exposure time, flat field and corrected for missing pixels.

* Proper combinations of the calibrated filtergram shall determine continuum intensity and equivalent line width, Doppler shifts and Stokes I, Q, U, and V components.

* The line parameters shall be calibrated and in turn be interpreted by suitable inversions as physical observables such as the thermodynamic state variables, line-of-sight velocity, and magnetic field strength and orientation.

* Images of the line parameters and/or the derived physical observables shall constitute the Level 1 data.

**6.0 Level 2-3 Science Data Products**

* The JSOC shall produce the standard data products shown in Table of Data Products

**7.0 Data Archive**

* All raw telemetry data shall be archived on two separate media. One for local storage, the other for off-site storage.

* All level 0 data shall be archived.

* Archiving of high level data products is optional as deemed appropriate.
8.0 Data Distribution

* Data for exploration, analysis, comparison, and interpretation shall be extracted from the JSOC archive.

* The archive shall have the potential for the selection of observables, times and places, and temporal and spatial scales and resolution.

* All of the HMI and AIA basic data products will be available for export.

* The internal representation of the data shall be transformed and exported in standard fits files with embedded keywords.

* Requested data products not currently on-line will automatically be retrieved from archive storage.

* Very large data requests will be copied to external media and delivered offline.

* The HMI+AIA data catalogs will be both directly accessible via the web and accessible via the VSO.

A chart of JSOC data volume is shown in: Figure 3.

9.0 MOC Operational Reports

* Each SOC shall be capable of receiving various operational reports from the MOC. These include trending reports, command history report, event log reports, time and time correlation log. The specific reports received by each SOC and their format shall be documented in the MOC-to-SOC ICD.

10.0 Flight Dynamics Products

* Each SOC shall be able to receive flight dynamics products from the MOC needed to plan science operations and process science data. The specific
products received by each SOC and their format shall be documented in the MOC-to-SOC ICD.

* Each SOC shall be able to receive flight dynamics products from the MOC to plan science operations and process science data. The specific products received by each SOC and their format shall be documented in the MOC-to-SOC ICD.

* Each SOC shall use the Flight Dynamic Products to do further data processing as needed to meet the science data requirements.

* The Flight Dynamic Products data shall be further processed to create the following ancillary data values to be used for the Science Data Products.

  - Heliographic Latitude of the observer's disk center.
  - Carrington rotation number for Heliographic Longitude of the observer's disk center.
  - Distance of observer to Sun center in AU.
  - Heliographic Longitude of the observer's disk center.
  - Apparent semi-diameter of Sun in arc seconds from HMI.
  - "Northward" velocity of observer in m/s. Positive is in direction of solar north.
  - Radial velocity of observer in m/s. Positive direction is away from the Sun.
  - "Westward" velocity of observer in m/s. Positive is in direction of Earth's orbit.
  - Original position angle of the solar north pole measured eastward (counterclockwise) from plate "north" (degrees).

The following Flight Dynamic Products will be used to create ancillary data values.

- Predicted Orbital State Vectors (Geocentric)
- 2-Line Elements
The Flight Dynamic Product data shall be used to create a list of events to help determine the quality of the science data. Some of the events to capture and monitor from this data are as follows:

- SDO ESR
- Momentum Management
- Guide Star Loss
- Off Nominal Roll
- HGA Blackout
- Eclipse

The following Flight Dynamic Products will be used to create these data values.

- Maneuver Command Data File
- Maneuver Planning File
- Long Range Maneuver Planning File
- Maneuver Summary
- Predicted HGA View Periods of SDOGS
- Predicted Eclipses
- Long Range Predicted Eclipses
- Predicted RFI
- Celestial Bodies in Instrument FOV

11.0 System Infrastructure

* The system shall support multi-user, multi-tasking capabilities and provide for process scheduling and control and inter-process communications.

* The system shall manage disk storage for all datasets that transit through the system. Automatic storage assignment, retention and deletion shall be provided.

* Programs shall access data by abstract dataset names which shall be resolved automatically to physical files.
* A central database shall provide for keyword and image data cataloging.

* Central message and error logging facilities shall be provided.

* Debug modes shall be integrated into the system functions.

* There shall be a data quality tracking and reporting subsytem.

* There shall be a central event handling facility to allow process scheduling and error handling.

Last modified @ November 11, 2004 10:43 Fri Nov 12
## Table 1. Data Flow Assumptions

<table>
<thead>
<tr>
<th>Data Path</th>
<th>Assumptions</th>
<th>Data Volume (GBytes/day)</th>
<th>Combined (GBytes/day)</th>
<th>Online Days. Tb</th>
</tr>
</thead>
<tbody>
<tr>
<td>In from DDS</td>
<td>HMI: 55,000,000 bps = 553</td>
<td>553</td>
<td>1300</td>
<td>30: 40Tb cache</td>
</tr>
<tr>
<td></td>
<td>AIA: 70,000,000 bps = 723</td>
<td>723</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level-0</td>
<td>HMI: 4k * 4k * 2 bytes/2-seconds = 1400*(pi/4)</td>
<td>550</td>
<td>1650</td>
<td>30: 50Tb cache</td>
</tr>
<tr>
<td></td>
<td>AIA: 4k * 4k * 2 bytes * 8 imgs per 10 seconds = 2200</td>
<td>1100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level-1</td>
<td>HMI: V,M,Ic @ 45s &amp; R, 1d, ff @ 90s = 330 *(pi/4)</td>
<td>130</td>
<td>2230</td>
<td>100: 220Tb cache</td>
</tr>
<tr>
<td></td>
<td>AIA: same as level-0 = 2200</td>
<td>1100</td>
<td></td>
<td>year: 20Tb perm</td>
</tr>
<tr>
<td>Higher level</td>
<td>HMI: See HMI- S015 and Figure 5 = 40</td>
<td>20</td>
<td>75</td>
<td>year: 30Tb perm</td>
</tr>
<tr>
<td></td>
<td>AIA (level): movies &amp; extracted regions = 110</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMSAL Link</td>
<td>HMI: Magnetograms (M, B)</td>
<td>45</td>
<td>1200</td>
<td>100: 115Tb cache</td>
</tr>
<tr>
<td></td>
<td>AIA: Full Level-0 data + level1 + DEM = 2200+110</td>
<td>1155</td>
<td></td>
<td>year: 50Tb perm</td>
</tr>
<tr>
<td>Export</td>
<td>HMI: 2 * Higher Level products + 5*10 min B</td>
<td>75</td>
<td>240</td>
<td>60: 15Tb cache</td>
</tr>
<tr>
<td></td>
<td>AIA: 3*higher Level products (TRACE &lt; 1)</td>
<td>165</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offline</td>
<td>All Lev-0, HMI Lev-1, All Higher</td>
<td>1253</td>
<td>3190</td>
<td>year: 1170Tb offline</td>
</tr>
<tr>
<td></td>
<td>AIA: tlm, Lev0, Lev1a</td>
<td>1933</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>