

170-TP-001-001

PreProcessing of NMC GRIB Formatted Products

Technical Paper

May 96

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

| | |
|--|---------|
| <u>Shaun de Witt /s/</u> | 5/13/96 |
| Shaun de Witt, Senior Engineer EOSDIS Core System Project | 5/13/96 |

SUBMITTED BY

| | |
|--|---------|
| <u>Karin Loya /s/</u> | 5/13/96 |
| Karin Loya, PDPS Manager EOSDIS Core System Project | 5/13/96 |

Hughes Information Technology Corporation
Landover, Maryland

This page intentionally left blank.

Abstract

This document shows the detailed design for Ancillary Data Pre-Processing of Gridded Binary (GRIB) format data from the National Meteorological Center (NMC). It demonstrates the object model, functional model, and data design for this task.

This document also covers the assumptions made during the design process. The purpose is to familiarize the reader with the design and implementation, and to give an understanding of how the output HDF-EOS file relates to the input GRIB format file. It expands on the previous white paper 240-WP-002-001, and clarifies the design as implemented.

Keywords: NMC, GRIB, Preprocessing, Ingest, Design, Model

This page intentionally left blank.

Contents

Abstract i

Contents iii

1. Introduction 1

| | |
|------------------------------|---|
| 1.1 Purpose..... | 1 |
| 1.2 Organization..... | 1 |
| 1.3 Review and Approval..... | 2 |
| 1.4 References..... | 2 |

2. Assumptions and Issues 4

| | |
|----------------------|---|
| 2.1 Assumptions..... | 4 |
| 2.2 Risks | 5 |

3. Object Model6

| | |
|----------------------------------|----|
| 3.1 Object Model..... | 6 |
| Description:..... | 7 |
| 3.2 Code Reuse | 10 |
| 3.3 Lines Of Code Estimates..... | 11 |

4 : Functional Model 12

5 : Data Model 17

| | |
|---------------------------------|----|
| 5.1 GRIB Format..... | 17 |
| 5.1.1GRIB Packing Methods | 18 |

| | |
|--------------------------------|----|
| 5.2 Output Product Format..... | 19 |
|--------------------------------|----|

Abbreviations and Acronyms 26

Appendix A : Metadata Configuration File 27

FIGURES

| | |
|---|----|
| Figure 3.1 Object Model for the GRIB Preprocessor | 7 |
| Figure 4.1 : Key to the Structure Diagrams | 12 |
| Figure 4.2 Level 0 Data Flow Diagram | 13 |
| Figure 4.3 Breakdown of PreProcess Function..... | 14 |
| Figure 4.4 Breakdown of Read File Function..... | 15 |
| Figure 4.5 Breakdown of Unpack Record Function | 16 |
| Figure 5.1 Mapping of GRIB Formatted Records to HDF Science Data Set | 21 |
| Figure 5.2 Graphic Depiction of the Format of the Output HDF-EOS Data File..... | 23 |

TABLES

| | |
|---|----|
| Table 1-1. White Paper to CDRL Migration..... | 2 |
| Table 3-1 Mapping of Heritage code to Object Model Operations | 10 |
| Table 3-2: Lines of Code on a Per-Class Basis..... | 11 |

1. Introduction

1.1 Purpose

The purpose of this paper is to provide a baseline for the design of the GRIB format preprocessing component of the INGEST subsystem. This is a part of the Ingest subsystem being developed as a part of the ECS project. It additionally documents the assumptions made during the design and highlights issues related to the work. Much of this document will be incorporated into the Ingest Subsystem Detailed Design contained in DID-305, Sept 1996.

The actual requirements which are satisfied by this work are as follows.

(a) S-INS-00400

The INGEST CI shall convert ingested data into a form accepted by the SDSRV CI/DDSRV CI, for the following data types, as needed.

(b) S-INS-00404

The INGEST CI shall extract metadata from ingested data into a form accepted by the SDSRV/DSSRV as needed, for the following categories of data:...(d) data set specific metadata formats.

The basic requirement in the preprocessing work discussed in this paper is to reformat data in native GRIB format into a form suitable for use by a number of science groups. The proposed format for storing data within ECS is the HDF-EOS format, and it is proposed that this be maintained as the standard for this work. It should be noted that similar work has been undertaken by the SeaDAS group at GSFC, this current effort extending the functionality of this previous code within the confines of HDF-EOS and other relevant ECS documentation [for more details of the SeaDAS project, see the WWW home page (<http://shark.gsfc.nasa.gov>)]. Additionally, code to unpack NMC GRIB formatted data into binary files is provided at the NMC. This code is available by anonymous ftp from the server nic.fb4.noaa.gov. For the most part, this code is written in FORTRAN. It is used extensively in the SeaDAS work, and its reuse is proposed for this code in this work. Which pieces of code are used is dealt with in Section 3 of this paper, detailing the object model. The NMC unpacking code is freely available, and it has been found that the subroutines can be used directly with no changes to the source code.

1.2 Organization

This paper is organized to reflect the detailed design under which the ADPP work is currently progressing. The second section details the assumptions made during the design and issues which have arisen during the design and prototyping activities already performed, and which are still outstanding at the time of publication. In the third section of this paper the object model

which will be used in this work is described, together with details of the heritage code which will be used in the preprocessing and how it fits into the object model. Section 4 provides a Data Flow Diagram to enhance the object model, and provide more detail for the reader. Finally, a data model is presented to show how the information in the original format maps to the final format within ECS.

1.3 Review and Approval

This White Paper is an informal document approved at the Office Manager level. It does not require formal Government review or approval; however, it is submitted with the intent that review and comments will be forthcoming. Since this document has already undergone technical and formal review processes by Karin Loya (Release A PDPS Manager), Graham Bland(EOSL), Carey Gire(Loral), Jo Pulkinnen (Loral), and Vince Grella(Loral), any comments received should not be expected to impact the design, unless there are reasons otherwise.

The ideas expressed in this White Paper are valid for the duration of the project; the concepts presented here are expected to migrate into the following formal CDRL deliveries:

Table 1-1. White Paper to CDRL Migration

| White Paper Section | CDRL DID/Document Number |
|---------------------|--------------------------|
| 2 & 3 | DID-305 |
| 5.2 | DID-311-CD-002-004 |

Questions regarding technical information contained within this Paper should be addressed to the following ECS contact:

- ECS Contacts
 - Shaun de Witt, Senior Engineer, (301) 925 1047, sdewitt@eos.hitc.com

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
 The ECS Project Office
 Hughes Information Technology Corporation
 1616 McCormick Drive
 Landover, MD 20785

1.4 References

The following documents have been used during the preprocessing of NMC GRIB formatted data.

SDP Toolkit 5 Users Guide for the ECS Project, 333-CD-003-002, 8/95

Release-A SDPS Ingest Subsystem Design Specification for the ECS Project, 305-CD-009-001, 7/95

HDF-EOS Primer for Version-1 EOSDIS, 175-WP-001-001, 4/95

GRIB Format Pre-Processing Design and Issues, 12/95, 240-WP-002-001

The WMO Format for the Storage of Weather Product Information and the Exchange of Weather Product Messages in Gridded Binary Form (Edition 1), NOAA Office Note 388, 7/94

2. Assumptions and Issues

In the design presented here, a number of assumptions are inherent in the model, and a number of issues have arisen during the design process. These will be discussed in this section, with justification given for the assumptions and details of the methods of overcoming the issues.

2.1 Assumptions

The following assumptions have been made during the design process. Some of these are based on the GRIB format specification (see below), others based on previous work by the SeaWIFS project, and some based on discussions with other groups within the ECS project.

- Only minimal quality checking will be performed. Only QC information already given in the input file will be used for reporting data quality. More detailed checking is performed by the SeaDAS code used for the SeaWIFS project (see Issues Section), but this level of detail is assumed unnecessary for preprocessing.
- The inventory level metadata in the reformatted file represents information covering the whole file. Any data which is specific for a record will be placed into archived metadata. For example, for the bounding coordinates, the data given in the inventory level metadata will cover all of the areas in the original file (normally this will be the whole globe), while the archive level metadata will contain the bounding coordinates for an individual record/SDS.
- It is assumed that the SDP subsystem Toolkit is re-usable within the Ingest subsystem for reading and writing metadata configuration files, and that these configuration files are the means by which the metadata required is identified..
- Only GRIB data conforming to the specification provided in NMC Office Note 388 (ON-388, July 1 1994) will be pre-processed. If an error in format is found, preprocessing of that file will be aborted. If an unrecognized grid type is encountered, the preprocessing will be aborted. For other unrecognized identifiers, appropriate information will be inserted into the metadata (for example a string may assume a value of "not known", an integer may assume a value of -1, etc., as appropriate), unless an error is reported from the heritage NMC code.
- The code for unpacking of GRIB data provided by the NMC is stable and can deal with all of the grid types defined in the format specification ON-388 .
- NMC will provide updates in conjunction with any changes in format.

2.2 Risks

One significant risk area has been identified during the design stage of the ancillary data preprocessing.

- The HDF-EOS software libraries are still undergoing beta testing, as of the time of May 1996. Upon full release, the relevant API's or functionality incorporated into these libraries may change, which may necessitate code changes and regression testing.

3. Object Model

The object model presented in figure 3.1 represents the GRIB preprocessing software component. The model shown is for the pre-processor alone, and is an expansion of the object model in the Release A Ingest Subsystem Design Specification (305-CD-009-001). The top level object (InGRIBData) is already included in this document. The remaining classes will be added for the next release of this document.

3.1 Object Model

The paragraphs following the diagram contain descriptions of each class, together with their essential attributes and operations and their purpose. To provide a brief overview, the preprocessing has been basically split into two branches as seen in the object model. These specifically perform the reformatting of the science and the metadata. Each of these branches will be discussed in more detail below. Before this splitting, the GRIB format file is itself split up so that each record within a file may be dealt with separately. After the reformatting is complete, the necessary information is gathered by a single class which then performs the actual writing of the reformatted file and associated metadata file.

The reformatting of the metadata involves extracting the Product Description Section, which contains the record specific metadata, from the original GRIB format record. Each parameter contained in the SDP must then be decoded and, where necessary, the appropriate metadata must be constructed from the decoded value (for example, the grid on which the data is presented is described in terms of an identifier, which may be used to determine bounding coordinates and projection). Using the MET tools provided in the PGS toolkit and the metadata configuration file, it is then possible to construct PVL describing the metadata for each record and calculate, where applicable, the global metadata.

The reformatting of the science data is slightly more complex. To unpack the information requires all of the sections with the exception of the Initial and Final Section. This branch of the model not only unpacks the original science data values, but also reformats them onto an appropriate grid for inclusion in an SDS and, where applicable, performs minimal QA on the science data. In this case, minimal QA will be performed only where the information is already contained within the GRIB record, such as the percentage of missing data.

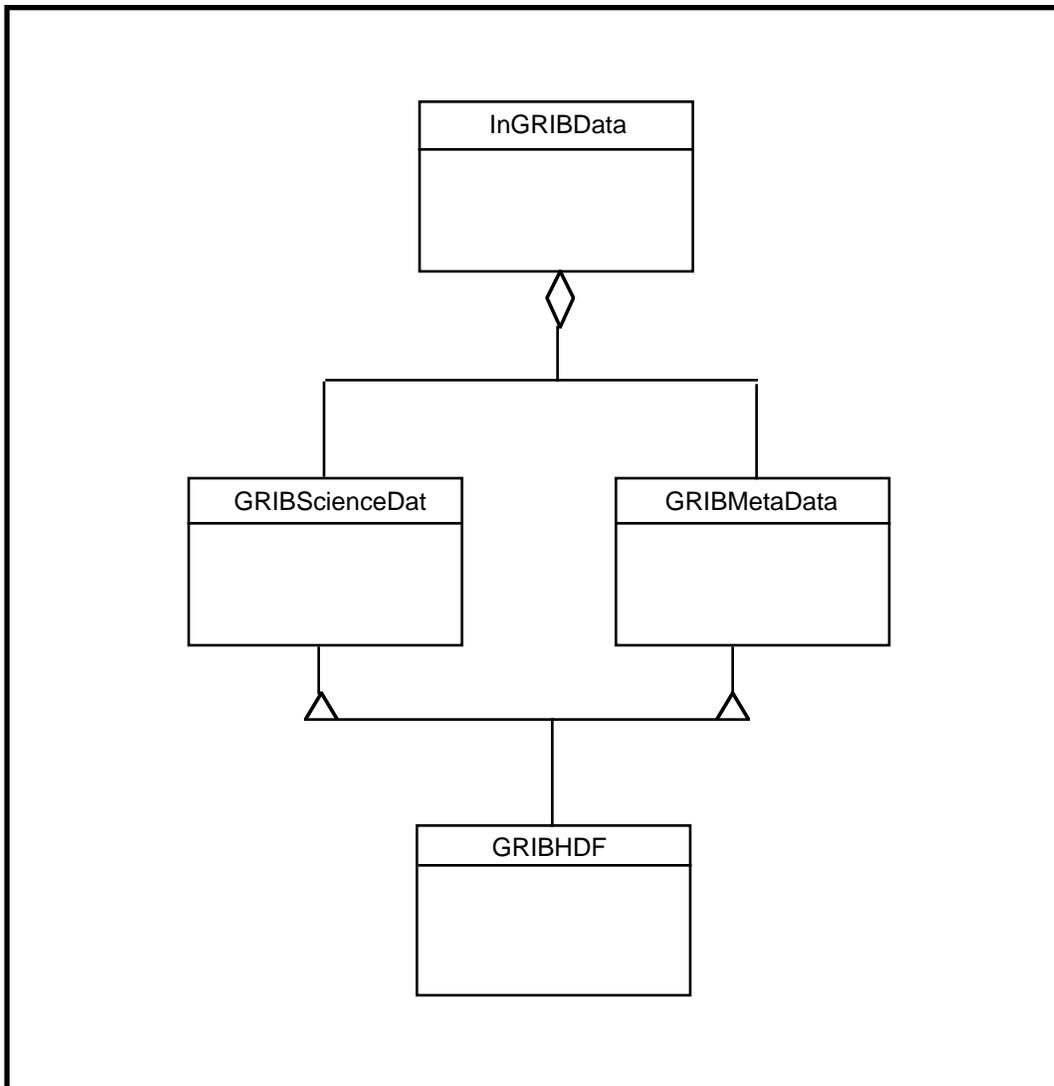


Figure 3.1 Object Model for the GRIB Preprocessor

InGRIBData

Description:

This is the top level call defined already by the ingest subsystem. It represents the top level of the preprocessing tool. The attributes and operations described here are an extension of those already in 305-CD-009-001.

Attributes:

myInputFile : EcTChar*
 myOutputFile : EcTChar*

mySourceMCF : EcTChar*
myTargetMCF : EcTChar*
myRecordArray : EcTChar**
myPDSArray : EcTChar**
myNumberOfRecords : EcTInt

Operations:

Preprocess() : EcTInt- *public class which starts preprocessing*
ExtractPDS() - *Extract the still encoded Product Description Section of the record.*
ReadRecord() - *Read a single record from the input GRIB file*
SearchPDSArray() - *Allows searching of PDS arrays. The purpose of this is to allow retrieval of a specific record using a PDS.*
This function will only be called if data in North/South hemisphere pairs is combined into a single SDS
GetNextRecord() - *Gets the next unused record from the array of records and updates the record to indicate that it has been processed.*

GRIBScienceData

Description:

Performs the unpacking of the science data. It assumes the entire file conforms to ON-388 GRIB format. Any deviations from the format will cause preprocessing to stop and an error condition will be returned.

Attributes:

myCurrentRecord : EcTChar*
myCurrentPDS : EcTChar*
myCurrentBDS : EcTChar*
myCurrentRecordType : EcTChar*
myCurrentRecordLevel
myCurrentGrid
myQCArray

Operations:

ConstructGrid() - *Construct the 2-d grid to contain the data values based on either the GDS or PDS of the current record*
PopulateGrid() - *Fill the grid with the data values*
QCRecord() - *Perform QC on the data*
UnpackPDS() - *Unpack the Product Description Section of the record.*
UnpackGDS() - *If the Grid Description Section exists, unpack the data for use in ConstructGrid*

UnpackBMS() - *Unpack the Bit Map Section of the current record, if it exists, for use in PopulateGrid.*

UnpackBDS() - *Unpack the Binary Data Section into correct type (floating point or integer) and store as a linear array of values before populating the grid*

GRIBMetadata

Description:

This function will extract metadata from the GRIB records and format it into an annotation group ready for insertion into the final HDF File. In addition, it will produce a separate metadata file to aid in the insertion of the data into the data server by the rest of the ingest subsystem.

Attributes:

myGlobalMetadata
myStructuralMetadata
myPDSValues

Operations:

ExtractGlobalMetadata() - *Extract and store metadata which is global to the whole file*

ExtractMetadata - *Extract the metadata from the current PDS by decoding and use of look up tables or structures to convert to human readable form*

StoreMetadata() - *Store the record specific metadata from the current record*

ReadPDSArray() - *Reads the next PDS in the array*

GRIBHDF

Description:

Performs the actual writing and formatting of the final HDF file. It will associate SDS's with corresponding record specific metadata.

Attributes:

Those inherited from parent classes
myAveragePercntOfMissingData
mySeachableMetdata
myNonSearchableMetadata

Operations:

WriteSDS() - *For each input record (or reconstituted pair), write the data to an SDS*

FormatMetaData() - *Create the attribute fields for the metadata (core, record specific and structural)*

FillAttributeData() - *Write the values of the attributes to the appropriate section of the HDF*

Process() - *Peform preprocessing functions*

3.2 Code Reuse

As stated in the Introduction to this document, it is intended to use heritage code provided by the NMC to unpack the native GRIB format data. This code is written in FORTRAN.

The unpacking code is maintained by NMC, and may occasionally be updated. It is outside the scope of this work to track these changes, nor is any mechanism provided by NMC for announcing changes. The NMC code reused within ECS is that provided at the start of December 1995 on their anonymous ftp server (ftp:nic.fb4.noaa.gov). Additions are likely to be along the lines of support for new grid types. It is not proposed that updates to the ECS code should be made routinely whenever the NMC issues a new version of the software.

For all of the heritage code, existing comments will be maintained. No changes are necessary to the heritage code for inclusion into ECS software.

There are a number of separate FORTRAN subroutines which will be incorporated into this work. These are listed in table 3.1, together with the object model attribute into which they will be added. A description of the purpose of the subroutine is also given.

Table 3-1 Mapping of Heritage code to Object Model Operations

| Name of Heritage Code | Object | Description of Purpose | Lines of Code | Source |
|-----------------------|-----------------|--|---------------|--------|
| FI631 | InGRIBData | Finds the start of a record and obtains the length in bytes of each section. Also checks to ensure that the record ends with the defined values in the End Section. | 159 | NMC |
| FI632 | GRIBScienceData | Extracts information from the PDS. | 270 | NMC |
| FI633 | GRIBScienceData | Extracts information from the GDS, if available. | 469 | NMC |
| FI634 | GRIBScienceData | If a BMS is present, extract it for use with filling grid. If a standard bit map is provided by a center, but not included in the record, then the appropriate bit map is generated. | 669 | NMC |
| FI635 | GRIBScienceData | Unpack the grid data in the BDS, and fill the output array correctly. | 626 | NMC |
| FI636 | GRIBScienceData | Process second order packing from the BDS for each data item. | 268 | NMC |
| FI637 | GRIBScienceData | Checks for a size mismatch between GDS (if present) and standard grids | 188 | NMC |

| | | | | |
|-----------------------|-----------------|---|-------|----------|
| W3FI63 | GRIBScienceData | Unpack a GRIB record to extract the specified grid, isolate the bit map and make the values contained in the PDS and GDS available in the return array. | 712 | NMC |
| W3FI01 | GRIBScienceData | Determines the number of bytes in a full word for a particular machine | 51 | NMC |
| W3FI83 | GRIBScienceData | Unpack delta packed values. | 107 | NMC |
| Custom Toolkit 5 Code | GRIBHDF | Reads MCF and writes metadata in PVL | 30336 | ECS |
| ODL | GRIBHDF | Low level routines used by Toolkit 5 | 10034 | OJC |
| HDF-EOS | GRIBHDF | Used to write the final output format. | n/k | ECS/NCSA |

Note that in the above table the lines of code for the Toolkit library is the total lines of code in the source directory. In practice, not all of these functions will be used. As an estimate, probably only twenty percent of the code in this library will be called upon. The ODL library is already contained within the Toolkit and will not be used directly in this work.

3.3 Lines Of Code Estimates

The following table presents the lines of code estimates for each class described above. It shows both the estimated number of custom lines and the lines in the re-used code. The estimate of the custom lines of code is based on decomposition of the functionality and on previous experience. The custom SLOC is approximated using table 3.1 of this document..

Table 3-2: Lines of Code on a Per-Class Basis

| Class Name | Estimated Custom LOC | Reused LOC | Actual Custom LOC |
|-----------------|----------------------|------------|-------------------|
| InGRIBData | 750 | 0 | 700 |
| GRIBScienceData | 500 | 3300 | 790 |
| GRIBMetaData | 750 | 0 | 910 |
| GRIBHDF | 500 | 3000+ | 1380 |

4 : Functional Model

The functional model is presented in this section. Where necessary, suitable leveling has been performed to clarify the processing. This functional modeling exists to support the Object Model and serves to demonstrate how the HDF-EOS file is generated from the input GRIB format. For a full functional model, the leaf processes in the functional model are the operations in the object model.

The model presented here does not go into the same level of detail as the object model in areas where uncertainty exists regarding functionality. In particular, the formatting into HDF-EOS has been encapsulated at this stage, both in the object and the functional model, until more information is received from the ECS team developing HDF-EOS. In practice, it will only be the name of the calls which are likely to vary, the actual functionality should not change as HDF-EOS becomes better defined.

The functional model is presented in a number of diagrams. A number of symbols are used in these, as shown below. A number is also referenced on the left hand side of each symbol. This is an internal reference number and has no significance for this document. It should be ignored.

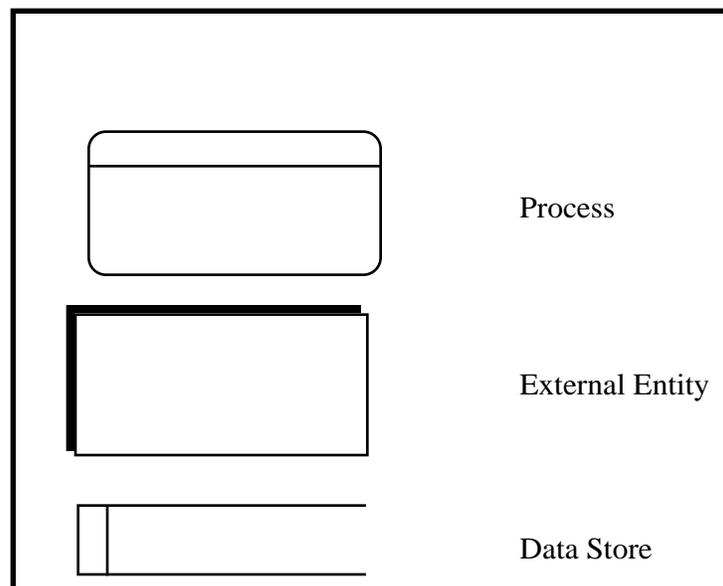


Figure 4.1 : Key to the Structure Diagrams

The first diagram is a simple top level functional model of the system, which is then broken down into its component functions. This top level shows all of the external data stores for the system, namely the input GRIB format file, the output HDF-EOS format file and the filled metadata configuration file. As the model is broken down, internal data stores are introduced. Diagrammatically, there is no distinction between the external and internal data stores, the reader must refer back to this top level description to see if a store is external or internal. The data flows are also labeled, but the contents of the data flow are outside the scope of this document, since they are not necessary to support the object model.

Since this model is only used to support the object model, no detailed description of the processes shown are given. The remainder of this section consists purely of the functional model diagrams themselves.

Figure 4.2 Level 0 Data Flow Diagram

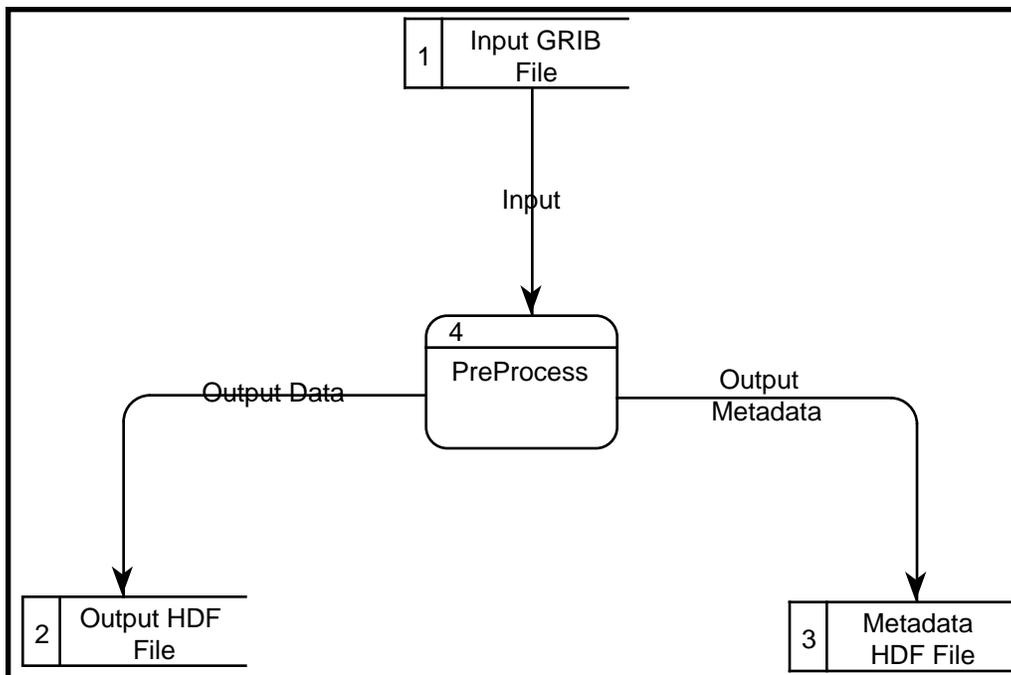


Figure 4.3 Breakdown of PreProcess Function

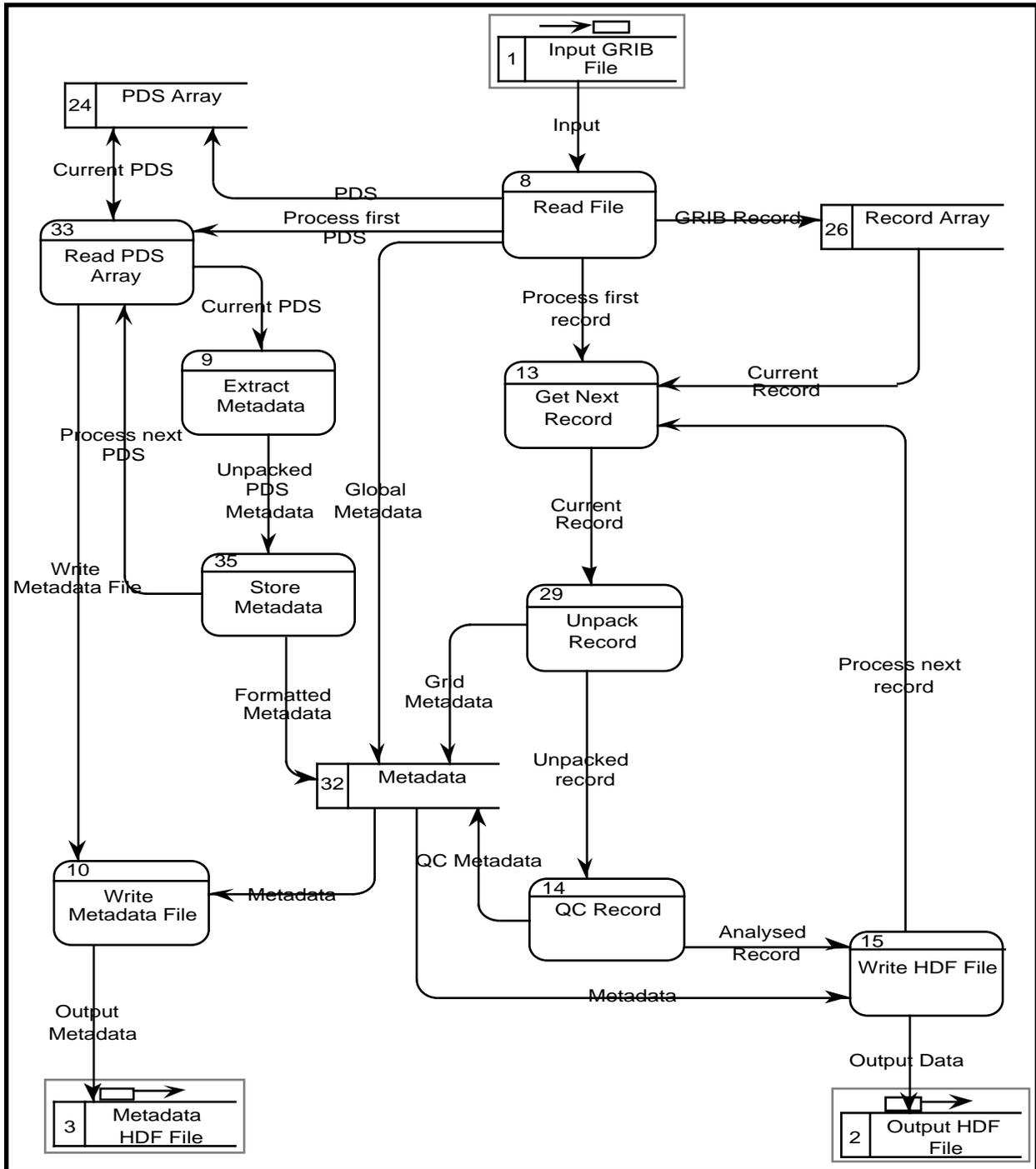


Figure 4.4 Breakdown of Read File Function

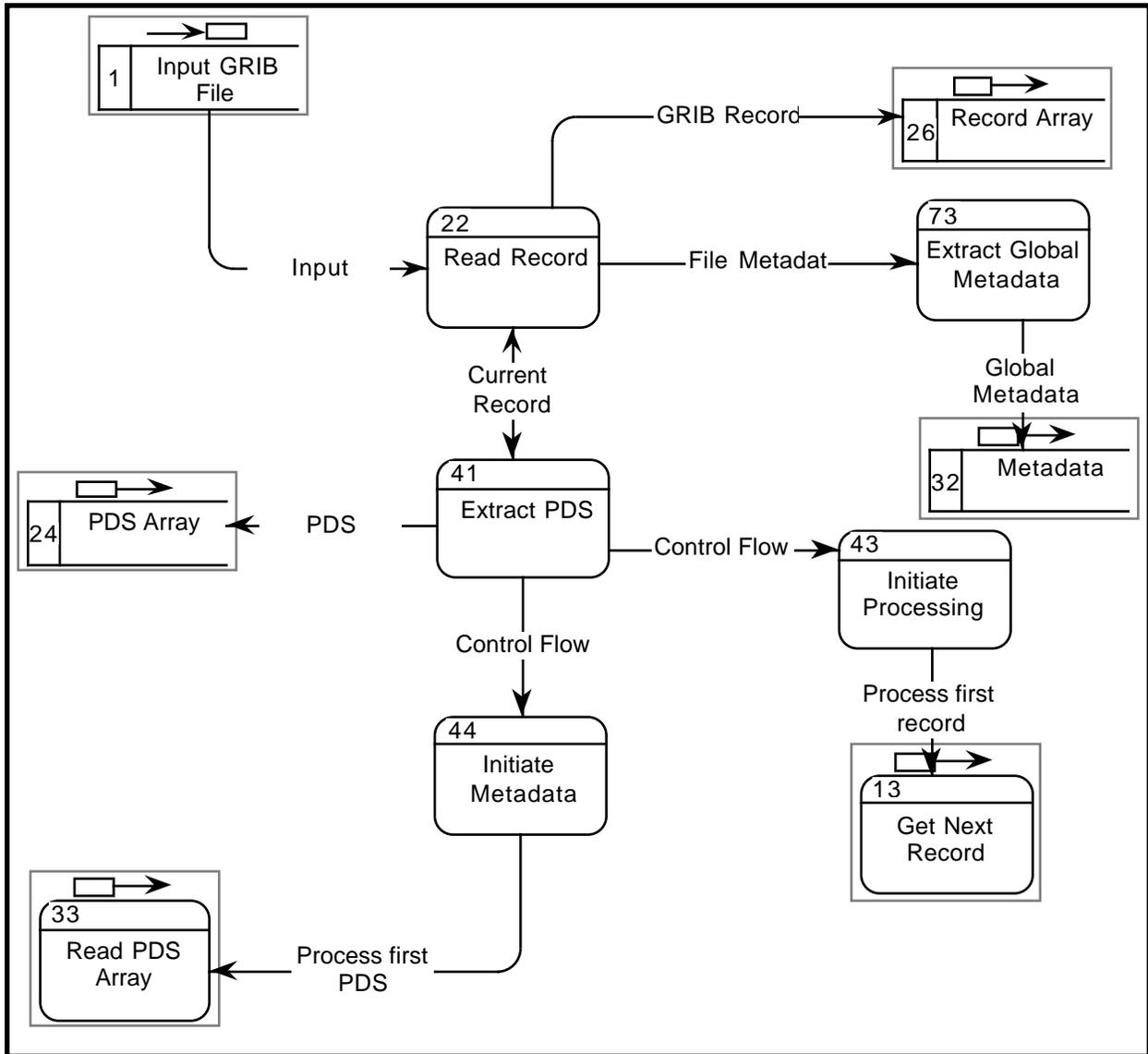
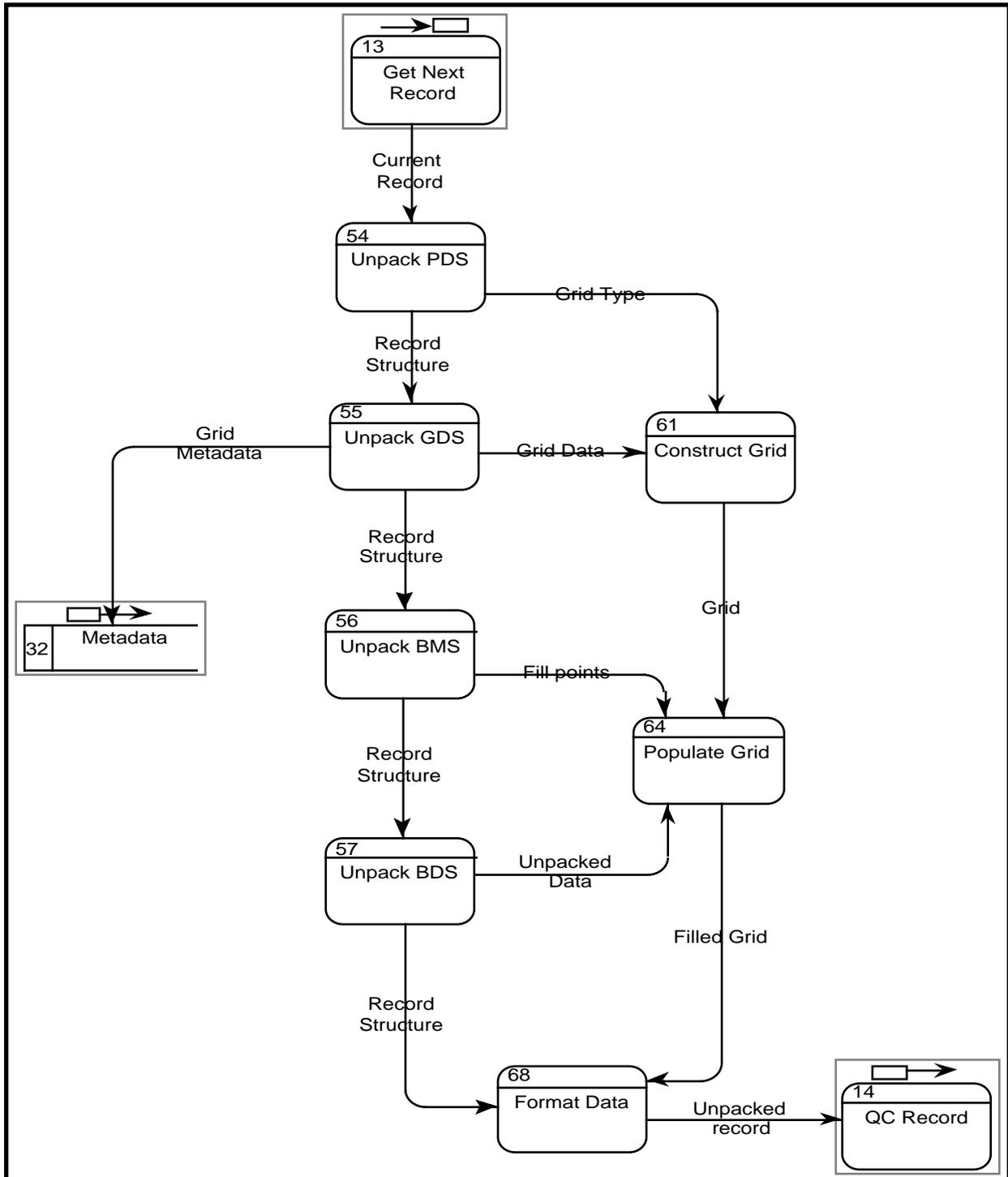


Figure 4.5 Breakdown of Unpack Record Function



5 : Data Model

This section deals in more detail with the format of the input GRIB product and the output HDF product, and how the two are related. Additionally, the metadata configuration file for the inventory and archived metadata are included.

5.1 GRIB Format

The agreement on the format of the GRIB ancillary data is described in the document 209-CD-008-002 (Interface Control Document between ECS and GSFC for the ECS Project). This states that the format shall be as documented in NOAA Office Note-388 Edition 1.

To summarize, a single GRIB format file from the NMC contains a number of GRIB records. Each GRIB record contains encoded data for one parameter, at one atmospheric level or layer at a specific location on a specific projection. There may be a variety of projections and geolocations contained in a single file.

Each GRIB record is made up of six sections.

- Initial Section (IS)
 - Used to identify the start of a record. Contains the characters "G R I B" and the total length of the record
- Product Description Section (PDS)
 - Contains most of the metadata for the record, such as the parameter information, atmospheric level and spatial and temporal information.
- Grid Description Section (GDS) - *optional*
 - Contains detailed information on the grid to be used. If a standard grid is used, this section is often omitted, and the details need to be obtained from the GRIB format document. Where the GDS is present it should be used for constructing the grid.
- Bit Map Section (BMS) - *optional*

If a particular grid is sparsely populated, then rather than store zero values in the BDS (see below), a bit map is provided showing which cells are to be populated is included in the record. This is particularly relevant for observational data, rather than forecast data, since data may have only been gathered from a few points on the grid. A bit value of 1 indicates a cell is to be filled, a zero value implies no data is available at that cell.

- Binary Data Section (BDS)

This section contains all of the data needed to unpack the original encoded values, and the encoded values themselves. More information concerning the encoding is given in the next section.

- End Section (ES)

This marks the end of a record. The last values are always the character values "7777".

Each GRIB record in a file will vary in length, and there is no guarantee that the contents of the files delivered from NMC will be consistent from day to day.

The NMC produces GRIB bulletins at varying frequencies. The datasets required for the ECS project are produced four times a day and are freely available by anonymous FTP at nic.fb4.noaa.gov in the directories /pub/data.00z and /pub/data.12z. Also at this site are the documents describing the GRIB format, which may be found in the directory /pub/nws/nmc/docs/gribguide. Sample code for unpacking the data into binary files and geolocation code is also provided at this site (mostly in FORTRAN).

5.1.1 GRIB Packing Methods

The packed data are coded as binary integers, using the minimum number of bits required for the desired precision. To pack the bits, the original data may first be scaled by a power of ten to obtain the correct precision. A reference value is then subtracted from them to eliminate redundancy and to eliminate negative values. After this, the data may be scaled by a power of two to pack the data into pre-selected word lengths. Thus a single value is represented by

$$Y * 10^D = R + (X * 2^E)$$

where

Y = original unpacked value in required units,
 D = decimal scale factor (2 byte signed integer),
 R = Reference value (4 byte floating point value),
 X = internal value,
 E = binary scale factor (2 byte signed integer).

The reference value is the minimum value of the decimally scaled data that is being encoded. The actual value is stored in four bytes as a single precision floating point number:

sAAAAAAAA BBBBBBBB BBBBBBBB BBBBBBBB

where

s = signed bit (0 means positive),
 A..A = 7 bit binary integer (characteristic)
 B..B = 24 bit binary integer (mantissa).

Thus the reference value can be recovered using the formula

$$R = (-1)^s * 2^{(-24)} * B * 16^{(A-64)}.$$

The remaining packing of the data may be performed in one of two ways, simple (fixed bit length) or second order (variable bit length) packing. Details of this can be found in ON-388, the details are beyond the scope of this document. The above information is sufficient to allow the unpacking to take place.

5.2 Output Product Format

After preprocessing, the GRIB record will be put into HDF-EOS format. HDF-EOS builds upon the standard HDF libraries. Particularly, it allows data to be inserted in three new forms; swath, point and grid. For the preprocessing of GRIB, only the latter of these needs to be considered. In addition to the data, metadata must be included to allow for additional functionality within ECS, such as subsetting and subsampling. Where possible, all of the metadata will be written in

PVL as an attribute (or group of attributes), while the data will be stored in Science Data Sets (SDS's). However, if the archived metadata group is larger than 32000 bytes (the maximum size of an HDF attribute), then this information will be written to an SDS.

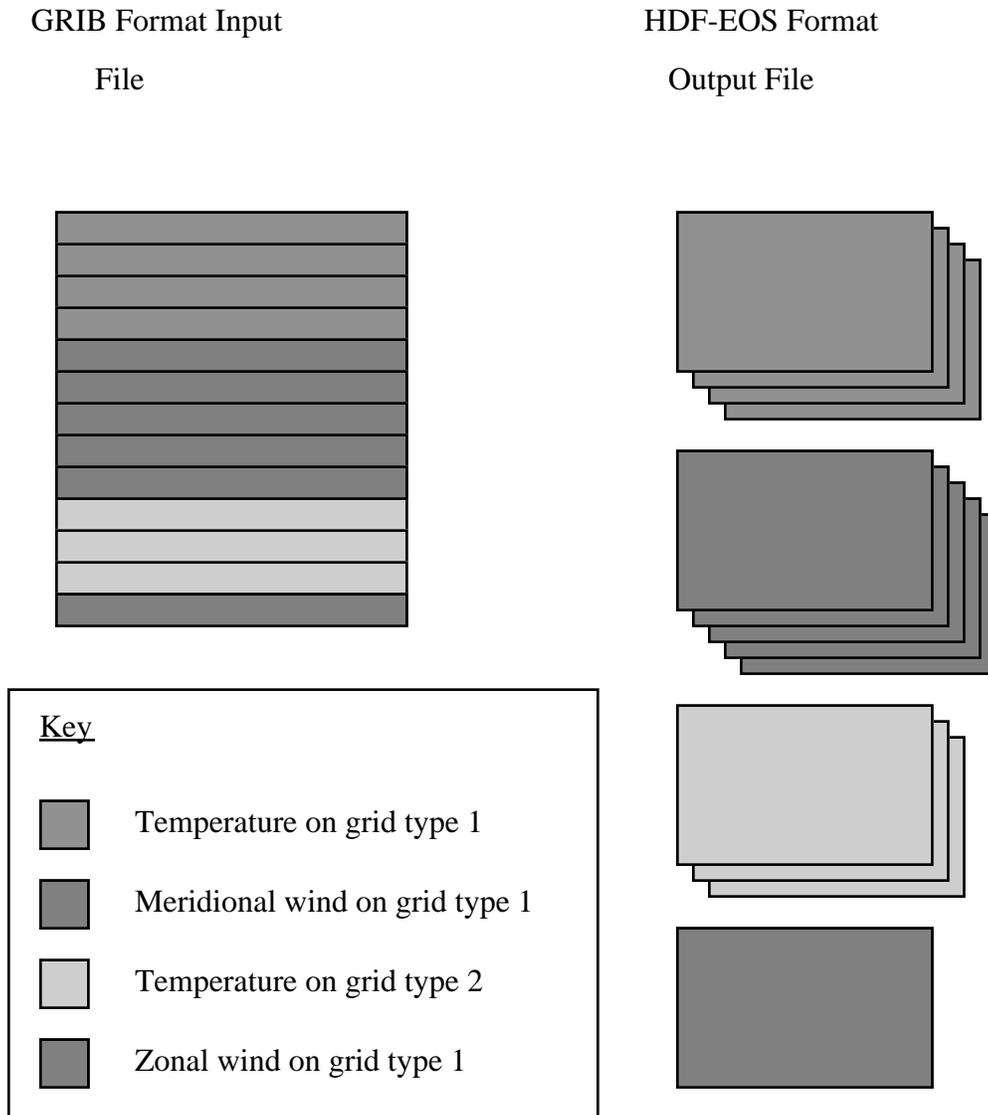
In designing the actual format of the data, a number of factors have been considered. These are:

- There is a need to minimize the number of SDS's in a single file since a large number of these will dramatically increase file access time. This is achieved by grouping data sets by parameter and grid. Thus all of the data in an SDS will be the same parameter on the same grid, but will represent different atmospheric levels.
- The metadata needs to be written as attributes in distinct groups; inventory metadata (covering collection and granule level information to allow insertion into the data server), archived metadata (containing information which is related to a particular product and is not general across products) and structural metadata (containing information on the layout of the file and defined by the HDF-EOS library).
- The requirements of the INGEST Sub-System to produce a single output file and associated metadata file. While ingest can deal with one file in, many out, the design only allows for a single output metadata file.

Much of the problem in designing the output format lies in the relating of record-specific information to a particular SDS. Much of the metadata which is considered inventory level, such as the geolocation information and level information are specific for each record and are not inherent to the whole file. Since, however, these attributes **MUST** appear in the inventory metadata, it is very difficult to relate this information back to the SDS's once the data is reformatted. The logical place to put this data for GRIB is in the structural metadata, which describes each SDS. However, some geolocation information needs to be placed at the inventory level. It is proposed to put in the maximum extent of coverage in the inventory metadata and the actual coverage for a particular SDS in the structural metadata and the archived metadata.

In order to minimize the number of SDS's it is proposed that they all be three dimensional. An SDS will then contain all of the information for a specific parameter on a specific grid, but covering various atmospheric levels (see figure 5.1). This should give between a 50% to 75% reduction in the number of SDS's compared with the number of GRIB records in the original file. Greater reduction could be achieved by placing all of the information for a particular grid in a single three dimensional SDS (achieving between 80% and 95% reduction), but it is felt that this would put too much information into the structural metadata. In HDF-EOS terminology, each grid will be made up of one or more fields.

Figure 5.1 Mapping of GRIB Formatted Records to HDF Science Data Set

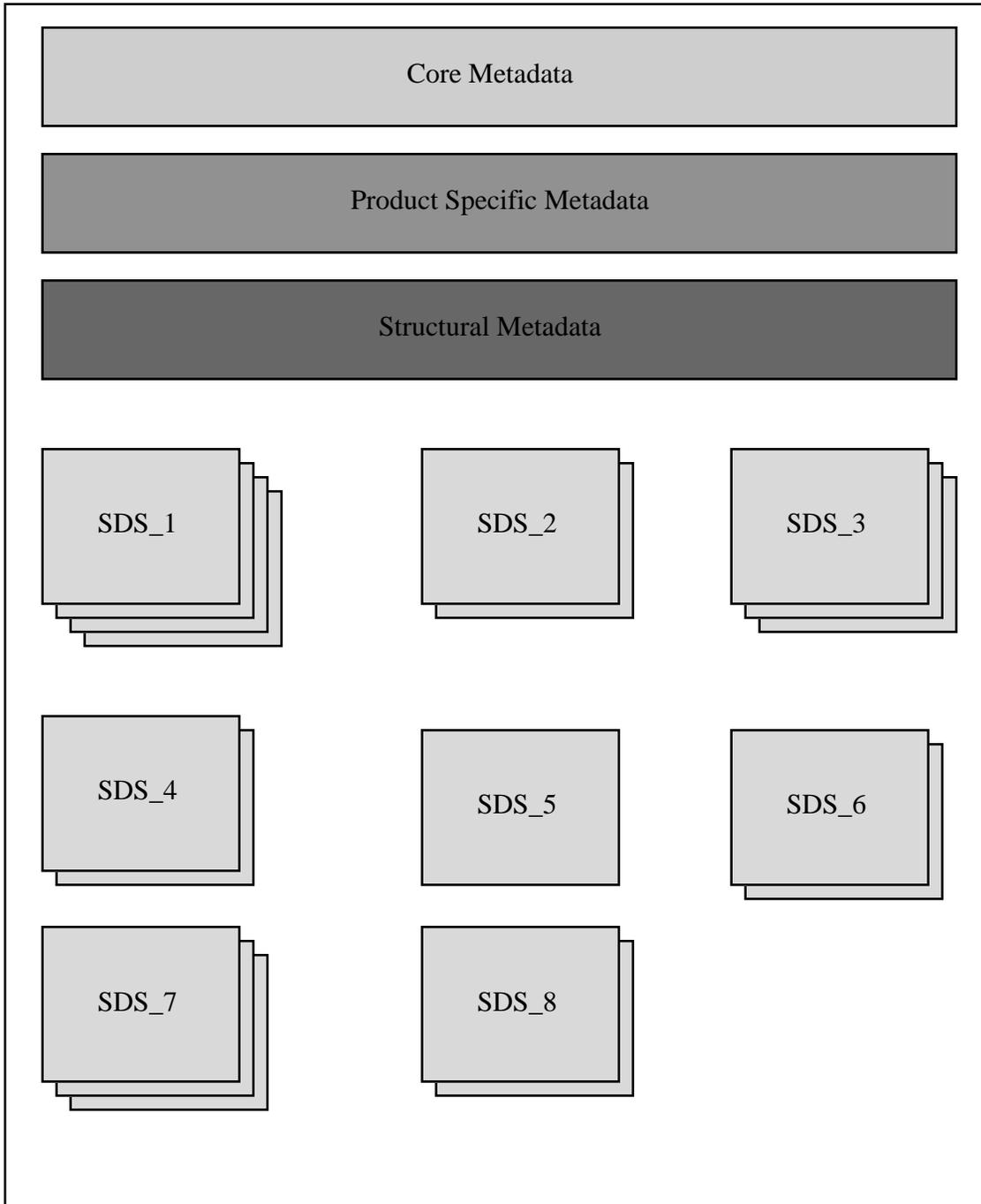


Thus a single HDF file would be composed of the following:

- Inventory metadata as an HDF attribute (named coremetadata.0..9)
- Archived metadata as an HDF attribute or SDS (named archivedmetadata.0..9)
- Structural metadata as an HDF-EOS attribute
- A number of SDS's

All of the metadata will be in PVL. In the case of archived and inventory metadata, it will be described by the use of configuration files. These configuration files will be written in ODL (Object Description Language). The configuration file is presented in appendix A of this document. The inventory metadata fields will be defined in DID311, but the examples given are based on a document provided for Internal Review. For more information on ODL, refer to the SDP Toolkit 5 Users Guide for the ECS Project (333-CD-003-002). The final format is shown diagrammatically in figure 5.2. This shows the format of an HDF-EOS file with 8 SDS's with varying Z levels. In the structural metadata group, there would be 8 groups of structural metadata, one per SDS.

Figure 5.2 Graphic Depiction of the Format of the Output HDF-EOS Data File



An important point which should be clarified for the archived metadata is how record specific data is related to a particular SDS. A unique identifier must exist for each SDS. It is intended that a label associated with a particular SDS will be made up of the standard abbreviation derived from ON-388, suffixed with *_N*, where N is the grid identifier. This suffix is to allow for the case where one parameter is represented on two different grids. Thus if we have two SDS's containing temperature data on two different grids, then these SDS's could have the labels TMP_27 and TMP_28. This gives easy information regarding the datatype presented in an SDS. The remaining metadata associated with an SDS is then associated with the SDS using the ODL GROUP and OBJECT structures. Each SDS in the output file is an object, and one of the objects within that group is the label. The remaining objects describe the metadata associated with an SDS, such as the projection, bounding co-ordinates (and other geolocation information) and axis labels. The only real point of note is the labels associated with the z-dimension. If the original GRIB format file contains 3 records containing geopotential height on the same grid but at different atmospheric levels, these records would be put into a single SDS as described above.

Within the metadata associated with the group, a number of labels associated with the z dimension have been specified as an array, named "VERTICALEXTENTS". This array is used to reference a specific SDS level. Each vertical extent is referenced as follows:

- if the vertical extent covers a named level (e.g. mean sea level or tropopause), the field is labelled with the name,
- if the vertical extent covers a named layer (e.g. entire atmosphere, troposphere), the field is labelled with the name of the layer,
- if the vertical extent is for a specific type of level (e.g. isobaric, isothermal) at a particular value, the field will be labelled *type at value units*. For example, for the case of an isothermal level at 0°C, the associated field would be labelled "ISOTHERMAL LEVEL AT 0 CELCIUS",
- if the vertical extent is a specified layer, then the field will be labelled *type layer between val1 units and val2 units*. For example if data exists for an isobaric layer between 50 and 100 hPa, then the associated field would be labelled ISOBARIC LAYER BETWEEN 50 hPa AND 100hPa.

Thus we now have archived metadata which identifies the number of SDS's in the output file, and associates metadata for the SDS. A user can then identify a particular SDS they are interested in by specifying the parameter name, projection, co-ordinates and an atmospheric level. As an example, consider a GRIB file containing 3 records of relative humidity at 750hPa, 500hPa and at the tropopause. A subset of the filled structural metadata written in ODL might look like the following.

OBJECT = SDS

```
OBJECT = SDS_NAME
      VALUE = R H_30
END_OBJECT = GROUP
```

```
OBJECT = VERTICALEXTENT
      VALUE = ("750 hPa atmospheric level", "500 hPa
atmospheric level", "Tropopause")
      END_OBJECT = VERTICALEXTENT

OBJECT = PROJECTION
      VALUE = "Polar Stereographic"
      END_OBJECT = PROJECTION

OBJECT = NORTHMOSTLATITUDE
      VALUE = 90000
      END_OBJECT = NORTHMOSTLATITUDE

OBJECT = SOUTHMOSTLATITUDE
      VALUE = 90000
      END_OBJECT = NORTHMOSTLATITUDE

OBJECT = SOUTHMOSTLATITUDE
      VALUE = 90000
      END_OBJECT = NORTHMOSTLATITUDE

OBJECT = EASTMOSTLONGITUDE
      VALUE = 180000
      END_OBJECT = EASTMOSTLONGITUDE

OBJECT = WESTMOSTLONGITUDE
      VALUE = -180000
      END_OBJECT = WESTMOSTLONGITUDE
END_OBJECT = SDS;
```

Abbreviations and Acronyms

| | |
|---------|--|
| API | Application Program Interface |
| BDS | Binary Data Section of GRIB record |
| BMS | Bit Map Section of GRIB record |
| ECS | EOSDIS Core System |
| HDF-EOS | Earth Observing System extended version of HDF |
| GDS | Grid Description Section of GRIB record |
| GRIB | Gridded Binary |
| HDF | Hierarchical Data Format |
| IS | Initial Section of GRIB record |
| MCF | Metadata Configuration File |
| NMC | National Meteorological Center |
| ODL | Object Design Language |
| PVL | Parameter Value Language |
| SDS | Science Data Set |
| SLOC | Source Lines Of Code |

Appendix A : Metadata Configuration File

GROUP = INVENTORYMETADATA

GROUPTYPE = MASTERGROUP

```
OBJECT = ShortName
  Data_Location   = "MCF"
  TYPE = "STRING"
  NUM_VAL = 1
  Value           = "NMC GRIB"
  Mandatory      = "TRUE"
END_OBJECT = ShortName
```

```
OBJECT = SizeMBECSDataGranule
  Data_Location   = "PGE"
  TYPE = "INTEGER"
  NUM_VAL = 1
  Mandatory      = "TRUE"
END_OBJECT = SizeMBECSDataGranule
```

```
/* Spatial Domain */
/* Note that the values contained in this section gives*/
/* the global coverage for the whole file. Coverage for*/
/* individual records in the GRIB file is found in the*/
/* structural metadata.*/
```

GROUP = BoundingBoxRectangle

```
OBJECT      = EastBoundingCoordinate
  Data_Location   = "PGE"
  NUM_VAL = 1
  TYPE = "INTEGER"
  Mandatory      = "TRUE"
END_OBJECT   = EastBoundingCoordinate
```

```
OBJECT      = WestBoundingCoordinate
  Data_Location   = "PGE"
  NUM_VAL = 1
  TYPE = "INTEGER"
  Mandatory      = "TRUE"
END_OBJECT   = WestBoundingCoordinate
```

```
OBJECT      = NorthBoundingCoordinate
```

```

        Data_Location    = "PGE"
        NUM_VAL = 1
        TYPE = "INTEGER"
        Mandatory        = "TRUE"
    END_OBJECT = NorthBoundingCoordinate

    OBJECT      = SouthBoundingCoordinate
        Data_Location    = "PGE"
        NUM_VAL = 1
        TYPE = "INTEGER"
        Mandatory        = "TRUE"
    END_OBJECT = SouthBoundingCoordinate

    END_GROUP = BoundingRectangle

/* Temporal metadata */
/* Note that this is only the time validity of the file.  Each */
/* record has its own specific time information and this appears */
/* in the product specific metadata.*/

    GROUP = RangeDateTime
        OBJECT      = RangeBeginningDateTime
            Data_Location    = "PGE"
            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory    = "TRUE"
        END_OBJECT = RangeBeginningDateTime

        OBJECT      = RangeEndingDateTime
            Data_Location    = "PGE"
            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory    = "TRUE"
        END_OBJECT = RangeEndingDateTime
    END_GROUP = RangeDateTime

    GROUP = QAStats
/* Set to zero since this is model data and no out of bounds */
/* limit is defined in the format document. */
        OBJECT = QAPercentOutofBoundsData
            Data_Location = "MCF"
            TYPE = "INTEGER"
            NUM_VAL = 1
            Value = 0

```

```

        Mandatory = "TRUE"
    END_OBJECT = QAPercentOutOfBoundsData

    OBJECT = QAPercentMissingData
        Data_Location = "MCF"
        TYPE = "INTEGER"
        NUM_VAL = 1
        Value = 0
        Mandatory = "TRUE"
    END_OBJECT = QAPercentMissingData
    END_GROUP = QAStats

    GROUP = QACollectionStats
        OBJECT = AutomaticQAFlag
            Data_Location = "MCF"
            NUM_VAL = 1
            TYPE = "STRING"
            Value = "passed"
            Mandatory = "TRUE"
        END_OBJECT = AutomaticQAFlag
    END_GROUP = QACollectionStats

END_GROUP = INVENTORYMETADATA

```

```

GROUP = ARCHIVEDMETADATA
GROUPTYPE = MASTERGROUP

```

```

GROUP = SDS_DATA
    OBJECT = SDS
        Data_Location = "NONE"
        Mandatory = "TRUE"
        CLASS = "M"

```

```

/* PARAMETER INFORMATION */

```

```

    OBJECT = NameOfSDS
        Data_Location = "PGE"
        Mandatory = "TRUE"
        NUM_VAL = 1
        TYPE = "STRING"
        Class = "M"
    END_OBJECT = NameOfSDS

```

```

OBJECT      = ParameterName
  Data_Location = "PGE"
  Mandatory     = "TRUE"
  TYPE = "STRING"
  NUM_VAL = 1
  Class        = "M"
END_OBJECT = ParameterName

/* PRODUCTION INFORMATION */

OBJECT      = DataProductionCenter
  Data_Location = "PGE"
  Mandatory     = "TRUE"
  NUM_VAL = 900
  TYPE = "STRING"
  Class        = "M"
END_OBJECT = DataProductionCenter

OBJECT      = DataProductionSubCenter
  Data_Location = "PGE"
  Mandatory     = "FALSE"
  NUM_VAL = 900
  TYPE = "STRING"
  Class        = "M"
END_OBJECT = DataProductionSubCenter

OBJECT = ModelName
  Data_Location = "PGE"
  Mandatory     = "FALSE"
  TYPE = "STRING"
  NUM_VAL = 900
  Class        = "M"
END_OBJECT = ModelName

/* GEOSPATIAL INFORMATION */

OBJECT      = GridID
  Data_Location = "PGE"
  Mandatory     = "TRUE"
  TYPE = "INTEGER"
  NUM_VAL = 1
  Class        = "M"
END_OBJECT = GridID

OBJECT      = ProjectionName

```

```

    Data_Location = "PGE"
    Mandatory     = "TRUE"
    NUM_VAL = 1
    TYPE = "STRING"
    Class        = "M"
END_OBJECT = ProjectionName

OBJECT      = NorthmostLatitude
    Data_Location = "PGE"
    Mandatory     = "TRUE"
    TYPE = "INTEGER"
    NUM_VAL = 1
    Class        = "M"
END_OBJECT = NorthmostLatitude

OBJECT      = SouthmostLatitude
    Data_Location = "PGE"
    Mandatory     = "TRUE"
    NUM_VAL = 1
    TYPE = "INTEGER"
    Class        = "M"
END_OBJECT = SouthmostLatitude

OBJECT      = EastmostLongitude
    Data_Location = "PGE"
    Mandatory     = "TRUE"
    NUM_VAL = 1
    TYPE = "INTEGER"
    Class        = "M"
END_OBJECT = EastmostLongitude

OBJECT      = WestmostLongitude
    Data_Location = "PGE"
    Mandatory     = "TRUE"
    NUM_VAL = 1
    TYPE = "INTEGER"
    Class        = "M"
END_OBJECT = WestmostLongitude

OBJECT      = VerticalExtent
    Data_Location = "PGE"
    Mandatory     = "TRUE"
    TYPE = "STRING"
    NUM_VAL = 900
    Class        = "M"

```

```

END_OBJECT = VerticalExtent

/* TEMPORAL INFORMATION */

OBJECT      = ForecastTimeUnit
  Data_Location = "PGE"
  Mandatory    = "TRUE"
  TYPE = "STRING"
  NUM_VAL = 900
  Class       = "M"
END_OBJECT = ForecastTimeUnit

OBJECT      = InitialTimePeriodP1
  Data_Location = "PGE"
  Mandatory    = "TRUE"
  TYPE = "INTEGER"
  NUM_VAL = 900
  Class       = "M"
END_OBJECT = InitialTimePeriodP1

OBJECT      = FinalTimePeriodP2
  Data_Location = "PGE"
  Mandatory    = "TRUE"
  TYPE = "INTEGER"
  NUM_VAL = 900
  Class       = "M"
END_OBJECT = FinalTimePeriodP2

OBJECT      = TimeRangeIndicator
  Data_Location = "PGE"
  Mandatory    = "TRUE"
  TYPE = "INTEGER"
  NUM_VAL = 900
  Class       = "M"
END_OBJECT = TimeRangeIndicator

/* QUALITY INFORMATION */

OBJECT      = NumberInAverage
  Data_Location = "PGE"
  Mandatory    = "FALSE"
  NUM_VAL = 900
  TYPE = "INTEGER"
  Class       = "M"
END_OBJECT = NumberInAverage

```

```
OBJECT      = NumberMissingFromAverage
  Data_Location = "PGE"
  Mandatory     = "FALSE"
  TYPE = "INTEGER"
  NUM_VAL = 900
  Class        = "M"
END_OBJECT = NumberMissingFromAverage

END_OBJECT = SDS
END_GROUP = SDS_DATA
END_GROUP = ARCHIVEDMETADATA
END
```