HDF-EOS Interface Based on HDF5, Volume 1: Overview and Examples

White Paper

White paper - Not intended for formal review or Government approval.

December 1999

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Larry Klein /s/ 12-16-99
David Wynne, Alex Muslimov, Abe Taaheri, Date
Ray Milburn, Larry Klein
EOSDIS Core System Project

SUBMITTED BY

Robert Plante /s/ 12-16-99
Robert Plante, Manager, Science Office Date
EOSDIS Core System Project

Raytheon Systems Company
Upper Marlboro, Maryland
This page intentionally left blank.
This document is a Users Guide for HDF-EOS (Hierarchical Data Format - Earth Observing System) library tools. The version described in this document is HDF-EOS Version 3.0. The software is based on HDF5, a new version of HDF provided by NCSA. HDF5 is a complete rewrite of the earlier HDF4 version, containing a different data model and user interface. HDF-EOS V3.0 incorporates HDF5, and keeps the familiar HDF4-based interface. There are a few exceptions and these exceptions are described in this document. Note that the contents of this document describe a prototype library, which is not yet operational. Release of an operational version is subject to NASA approval, but is expected in the summer of 2000.

HDF is the scientific data format standard selected by NASA as the baseline standard for EOS. This Users Guide accompanies Version 3 software, which is available to the user community on the EDHS1 server. This library is aimed at EOS data producers and consumers, who will develop their data into increasingly higher order products. These products range from calibrated Level 1 to Level 4 model data. The primary use of the HDF-EOS library will be to create structures for associating geolocation data with their associated science data. This association is specified by producers through use of the supplied library. Most EOS data products which have been identified, fall into categories of point, grid or swath structures, the latter two of which are implemented in the current version of the library. Services based on geolocation information will be built on HDF-EOS structures. Producers of products not covered by these structures, e.g. non-geolocated data, can use the standard HDF libraries.

In the ECS (EOS Core System) production system, the HDF-EOS library will be used in conjunction with SDP (Science Data Processing) Toolkit software. The primary tools used in conjunction with HDF-EOS library will be those for metadata handling, process control and status message handling. Metadata tools will be used to write ECS inventory and granule specific metadata into HDF-EOS files, while the process control tools will be used to access physical file handles used by the HDF tools. (SDP Toolkit Users Guide for the ECS Project, June 1999, 333-CD-500-001).

HDF-EOS is an extension of NCSA (National Center for Supercomputing Applications) HDF and uses HDF library calls as an underlying basis. Version 5-1.2.0 of HDF is used. The library tools are written in the C language and a FORTRAN interface is provided. The current version contains software for creating, accessing and manipulating Grid and Swath structures. This document includes overviews of the interfaces, and code examples. EOSView, the HDF-EOS viewing tool, has been revised to accommodate the current version of the library.

Technical Points of Contact within EOS are:
Larry Klein, larry@eos.hitc.com
David Wynne, davidw@eos.hitc.com
An email address has been provided for user help:
pgstllkit@eos.hite.com

Any questions should be addressed to:
Data Management Office
The ECS Project Office
Raytheon Systems Company
1616 McCormick Drive
Upper Marlboro, MD 20774-5301
This document will serve as the user’s guide to the prototype HDF-EOS file access library based on HDF5. HDF refers to the scientific data format standard selected by NASA as the baseline standard for EOS, and HDF-EOS refers to EOS conventions for using HDF. This document will provide information on the use of the two interfaces included in this version – Swath, and Grid – including overviews of the interfaces, and code examples. This document should be suitable for use by data producers and data users alike.

**Keywords:**  HDF-EOS, HDF5, Metadata, Standard Data Format, Standard Data Product, Disk Format, Grid, Swath, Projection, Array, Browse
This page intentionally left blank.
Contents

Preface

Abstract

1. Introduction

1.1 Identification ........................................................................................ 1-1
1.2 Scope ...................................................................................................... 1-1
1.3 Purpose and Objectives ......................................................................... 1-1
1.4 Status and Schedule ............................................................................ 1-2
1.5 Document Organization ....................................................................... 1-2

2. Related Documentation

2.1 Parent Documents ............................................................................... 2-1
2.2 Related Documents ............................................................................ 2-1

3. Overview of HDF-EOS

3.1 Background .......................................................................................... 3-1
3.2 Design Philosophy ............................................................................... 3-1
3.3 Packaging ............................................................................................. 3-2
3.4 Operations Concept for the HDF5 Based Prototype Library ................. 3-3
3.5 Differences Between HDF-EOS V2.5 (HDF4 based) and HDF-EOS V3.0 (HDF5 based) ................................................................. 3-3
4. Swath Data

4.1 Introduction ........................................................................................................... 4-1
  4.1.1 Data Fields ...................................................................................................... 4-2
  4.1.2 Geolocation Fields ....................................................................................... 4-3
  4.1.3 Dimensions ..................................................................................................... 4-3
  4.1.4 Dimension Maps ............................................................................................ 4-3
  4.1.5 Index ................................................................................................................ 4-5
4.2 Applicability ........................................................................................................... 4-5
4.3 The Swath Data Interface ...................................................................................... 4-5
  4.3.1 SW API Routines .......................................................................................... 4-5
  4.3.2 File Identifiers ............................................................................................... 4-7
  4.3.3 Swath Identifiers ............................................................................................ 4-7
4.4 Programming Model ............................................................................................. 4-7

5. Grid Data

5.1 Introduction ........................................................................................................... 5-1
  5.1.1 Data Fields ...................................................................................................... 5-2
  5.1.2 Dimensions ..................................................................................................... 5-2
  5.1.3 Projections ...................................................................................................... 5-2
5.2 Applicability ........................................................................................................... 5-3
5.3 The Grid Data Interface ...................................................................................... 5-3
  5.3.1 GD API Routines .......................................................................................... 5-3
  5.3.2 File Identifiers ............................................................................................... 5-5
  5.3.3 Grid Identifiers ............................................................................................... 5-5
5.4 Programming Model ............................................................................................. 5-5
5.5 GCTP Usage .......................................................................................................... 5-6
  5.5.1 GCTP Projection Codes ................................................................................. 5-6
  5.5.2 UTM Zone Codes ........................................................................................... 5-7
  5.5.3 GCTP Spheroid Codes ............................................................................... 5-8
  5.5.4 Projection Parameters .................................................................................... 5-9
6. Examples of HDF-EOS Library Usage

6.1 Swath Examples .................................................................................... 6-1
   6.1.1 Creating a Simple Swath ............................................................... 6-1
   6.1.2 Performing Subsetting for a Swath .............................................. 6-8
6.2 Grid Examples ..................................................................................... 6-20
   6.2.1 Creating a Simple Grid ............................................................... 6-20
   6.2.2 Performing Subsetting for a Grid ............................................... 6-44
6.3 Combining HDF and HDF-EOS Objects ............................................. 6-47
   6.3.1 Adding HDF-EOS Structures to an Existing HDF File ............... 6-47

7. Writing ODL Metadata into HDF-EOS

7.1 A C Example of Metadata Write ...................................................... 7-1
7.2 C Code .............................................................................................. 7-1
7.3 The Metadata Configuration File (MCF) for Code in Section 7.2 .......... 7-8
7.4 The ODL Output File Which Results from Running ................................ 7-16
7.5 The file filetable.temp used for example in Section 7.2 ...................... 7-25

Appendix A. Installation and Maintenance

List of Figures

4-1. A Typical Satellite Swath: Scanning Instrument .......................... 4-1
4-2. A Swath Derived from a Profiling Instrument .............................. 4-2
4-3. A “Normal” Dimension Map ......................................................... 4-4
4-4. A “Backwards” Dimension Map .................................................... 4-4
5-1. A Data Field in a Mercator-projected Grid .................................. 5-1
5-2. A Data Field in an Interrupted Goode’s Homolosine-Projected Grid .. 5-2
List of Tables

3-1. HDF-EOS 3.0 (Prototype) Current Modifications to the HDF4-Based
HDF-EOS Library (2.5) ........................................................................... 3-4

4-1. Summary of the Swath Interface......................................................... 4-6

5-1. Summary of the Grid Interface............................................................ 5-4

5-2. Projection Transformation Package Projection Parameters .................... 5-9

5-3. Projection Transformation Package Projection Parameters Elements ........... 5-10

Abbreviations and Acronyms
1. Introduction

1.1 Identification

The *HDF-EOS User’s Guide for the ECS Project* was prepared under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract (NAS5-60000).

1.2 Scope

This document is intended for use by anyone who wishes to write software to create or read EOS data products. Users of this document will likely include EOS instrument team science software developers and data product designers, DAAC personnel, and end users of EOS data products such as scientists and researchers.

Note: This current document describes a prototype library based on HDF5. Official NASA support of this new data format is not yet at hand. This recognition is expected in early 2000. In the near term, an HDF-EOS interface for both HDF4 and HDF5 based files will be provided. The HDF4 based library is currently V2.5.

1.3 Purpose and Objectives

This document will serve as a user’s guide for the HDF-EOS file access library developed for ECS. Upon reading this document, the reader should have a thorough understanding of each data model and corresponding programming interface provided as part of HDF-EOS. Specifically, this user’s guide contains an overview of each data model, a complete function-by-function reference for each software interface, and sample programs illustrating the basic features of each interface.

The reader should note that this paper will not discuss the HDF structures underlying HDF-EOS nor the specific conventions employed. For more information on HDF, its design philosophy, and its logical and physical formats, the reader is referred to NCSA documentation listed in Section 2.2 Applicable Documents. For more information on the conventions employed by HDF-EOS, the reader is referred to the various design White Papers listed in Section 2.2.

*Important Note:*

The FORTRAN-literate reader is cautioned that dimension ordering is row-major in C (last dimension varying fastest), whereas FORTRAN uses column-major ordering (first dimension varying fastest). Therefore, FORTRAN programmers should take care to use dimensions in the reverse order to that shown in the text of this document. (FORTRAN code examples are correct as written.)
1.4 Status and Schedule

December, 1999, Prototype HDF5 based Library Available
August, 2000, SCF version including both HDF4 and HDF5 support available.
June, 2001, ECS support available.

Note that this schedule is proposed to NASA, but not yet accepted at the time of this writing.

1.5 Document Organization

This document is organized as follows:

- Section 1 Introduction - Presents Scope and Purpose of this document
- Section 2 Related Documentation
- Section 3 Overview of HDF-EOS - Background and design features of the library.
- Section 4 Swath Data - Design features and listing of the HDF-EOS Swath Library.
- Section 6 Grid Data - Design features and listing of the HDF-EOS Grid Library.
- Section 7 Examples - A selection of programming examples.
- Section 8. Examples of SDP Toolkit Usage - How to use the HDF-EOS Library in conjunction with the SDP Toolkit.
- Appendix A Installation Instructions, Test Drivers, User Feedback
- Acronyms

The accompanying Function Reference Guide is organized as follows:

- Section 1 Introduction
- Section 2 Reference - Specification of the HDF-EOS Swath and Grid
- APIs Function
- Acronyms
2. Related Documentation

2.1 Parent Documents

The following documents are the parents from which this document’s scope and content derive:

- 170-TP-500 The HDF-EOS Library Users Guide for the ECS Project, Volume 1: Overview and Examples
- 456-TP-013 The HDF-EOS Design Document for the ECS Project
- 170-WP-002 Thoughts on HDF-EOS Metadata, A White Paper for the ECS Project
- 170-WP-003 The HDF-EOS Swath Concept, A White Paper for the ECS Project
- 170-WP-011 The HDF-EOS Grid Concept, A White Paper for the ECS Project

2.2 Related Documents

The following documents are referenced within this technical paper, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document.

- 333-CD-003 Release A SCF Toolkit Users Guide for the ECS Project
- 175-WP-001 HDF-EOS Primer for Version 1 EOSDIS, A White Paper for the ECS Project
- None Introduction to HDF5 Release 1.2, University of Illinois, October, 1999
- None Getting Started with HDF, Version 3.2, University of Illinois, May 1993
| none | An Album of Map Projections, USGS Professional Paper 1453, Snyder and Voxland, 1989 |
3. Overview of HDF-EOS

3.1 Background

The Hierarchical Data Format (HDF) has been selected by the EOSDIS Project as the format of choice for standard product distribution. HDF is a function library that was originally developed by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign to provide a portable storage mechanism for supercomputer simulation results. Although this user’s guide does not attempt to explain the inner workings of NCSA HDF, a cursory knowledge of HDF may help the reader to understand the basic workings of HDF-EOS.

HDF5 files consist of a directory and a collection of data objects. Every data object has a directory entry, containing a pointer to the data object location, and information defining the datatype (much more information about HDF can be found in the NCSA documentation referenced in Section 2.2 of this Guide). Unlike HDF4, there are only two fundamental data objects in HDF5. These objects are groups and dataspaces. The HDF4 data types such as vdatas and scientific data sets are mapped into the more general class of dataspaces.

To bridge the gap between the needs of EOS data products and the capabilities of HDF, three new EOS specific datatypes – **point**, **swath**, and **grid** – have been defined within the HDF framework. Each of these new datatypes is constructed using conventions for combining standard HDF datatypes and is supported by a special application programming interface (API) which aids the data product user or producer in the application of the conventions. The APIs allow data products to be created and manipulated in ways appropriate to each datatype, without regard to the actual HDF objects and conventions underlying them.

The sum of these new APIs comprise the HDF-EOS library. The **Point** interface is designed to support data that has associated geolocation information, but is not organized in any well defined spatial or temporal way. The **Swath** interface is tailored to support time-ordered data such as satellite swaths (which consist of a time-ordered series of scanlines), or profilers (which consist of a time-ordered series of profiles). The **Grid** interface is designed to support data that has been stored in a rectilinear array based on a well defined and explicitly supported projection.

*Note that in this current prototype version, the Point interface is not implemented.*

3.2 Design Philosophy

Since the HDF-EOS library is intended to support end users of EOS data as well as EOS data producers, it is essential that HDF-EOS be available separately from other ECS software. For this reason, HDF-EOS does not rely on any other ECS software, including the SDP Toolkit. It is treated as an extension to the HDF library and, as such, it follows the general design philosophy and coding style of HDF. For more information on the design of HDF, please refer to the appropriate NCSA documentation listed in Section 2.2.
3.3 Packaging

Because of the functional overlap of HDF, HDF-EOS, and the SDP Toolkit, it is important to understand what each one contains and how they are related. NCSA HDF is a subroutine library freely available as source code from the National Center for Supercomputing Applications. The basic HDF library has its own documentation, and comes with a selection of simple utilities.

HDF-EOS is a higher level library available from the ECS project as an add-on to the basic HDF library. It requires NCSA HDF for successful compiling and linking and will be widely available (at no charge) to all interested parties. Although at the time of this writing, the exact packaging has yet to be determined, the basic HDF library will also be available from the ECS project.

The SDP Toolkit is a large, complex library of functions for use by EOS data producers. It presents a standard interface to Distributed Active Archive Center (DAAC) services for data processing, job scheduling, and error handling. While the SDP Toolkit may be available to individual researchers, it is unlikely to be of great use outside of EOS DAACs and Science Computing Facilities (SCF). The Toolkit distribution includes source code for both HDF and HDF-EOS.

EOS instrument data producers will use the SDP Toolkit in conjunction with the HDF-EOS and HDF libraries. Of primary importance will be process control and metadata handling tools. The former will be used to access physical file handles required by the HDF library. The SDP Toolkit uses logical file handles to access data, while HDF (HDF-EOS) requires physical handles. Users will be required to make one additional call, using the SDP toolkit to access the physical handles. Please refer to the SDP Toolkit Users Guide for the ECS Project, Mar, 1998, 333-CD-100-001, Section 6.2.1.2 for an example). Section 7 of this document gives examples of HDF-EOS usage in conjunction with the SDP Toolkit.

Metadata tools will be used to access and write inventory and granule specific metadata into their designated HDF structures. Please refer to Section 6.2.1.4 of the SDP Toolkit Users Guide.

*Note that in this current prototype, only read and write metadata functions have been implemented to use HDF5 based files.*

We make an important distinction between granule metadata and the structural metadata referred to in the software description below. Structural metadata specifies the internal HDF-EOS file structure and the relationship between geolocation data and the data itself. Structural metadata is created and then accessed by calling the HDF-EOS functions. Granule metadata will be used by ECS to perform archival services on the data. A copy will attached to HDF-EOS files by SDP toolkit calls and another copy is placed in the ECS archives. The two sets of metadata are not dynamically linked. However, the data producer should use consistent naming conventions when writing granule metadata when calling the HDF-EOS API. Please refer to the examples in Section 7, below.
3.4 Operations Concept for the HDF5 Based Prototype Library

HDF5 is a nearly complete rewrite of HDF4 and contains a different user API and underlying data model. An HDF-EOS library written in conjunction with HDF5 and which uses HDF5 functionality, will necessarily be a rewrite of the HDF4-based version. The new HDF5-based library will support the same Grid/Point/Swath functionality and to the extent possible and will be built with the same calling sequences as the original library. We will refer to the newer library as HDF-EOS 3.0. The former library is currently designated HDF-EOS 2.5. (The HDF-EOS Library Users Guide for the ECS Project, Volume 1 and Volume 2)

The following future uses for HDF-EOS are anticipated:

A. Product developers reading and writing HDF4-based files.
B. ECS subsystems reading and writing HDF4-based files.
C. Product developers reading and writing HDF5-based files. This will include both users retrofitting HDF4-based software and users starting from scratch with HDF5.
D. ECS subsystems reading and writing HDF5-based files.
E. Data migration applications, i.e. read HDF4 and write HDF5.

HDF-EOS 3.0 support for HDF5 will have the same function calls as the current HDF-EOS 2.5 support for HDF4. There will be a few additions, such as calls to account for HDF5 implementation of ragged arrays. The parameters in the V3.0 function calls will be the same, to the extent possible. The name of the function calls will be the same. This implementation will entail the fewest possible modifications with respect to retrofitting code, i.e. removing HDF4 file access in favor of HDF5 file access. Users retrofitting old code, should anticipate changing variable types to accommodate new HDF5 variable types. The API will make the underlying HDF5 data model transparent otherwise.

If a user outside ECS wants to be able to read HDF4 files and write HDF5 files, we recommend using C++ class libraries. In that way, HDF4 and HDF5 based function names can remain the same. Another method will be to provide a parallel HDF5 based library in which function names have been changed. For example, SWopen() becomes SWopen5(). We will provide installation scripts, which will install HDF4 and/or HDF5-based libraries based on user preference. This parallel library is not implemented in the prototype.

Support will be provided for the current suite of UNIX and Windows operating systems. F77, F90, C and C++ interfaces will continue to be supported.

3.5 Differences Between HDF-EOS V2.5 (HDF4 based) and HDF-EOS V3.0 (HDF5 based)

There are several important differences between the Versions 2.5 and 3.0 of the HDF-EOS library that are briefly summarized in this section. An overview of modifications to the HDF4 base library is listed in Table 3.1
Table 3-1. HDF-EOS 3.0 (Prototype) Current Modifications to the HDF4-Based HDF-EOS Library (2.5)

<table>
<thead>
<tr>
<th>Function</th>
<th>Change Description</th>
</tr>
</thead>
</table>
| Swath Access | • Modify to implement new HDF5 file structure.  
• Create a swath with a specified name |
| Swath Definition | • Add an additional parameter, 'maxdimlist', a list of maximum sizes of dimensions. This will account for feature of appendability of each dimension in a dataset.  
• Add a new function for chunking in HDF5 (SWdefchunk()) |
| Swath Inquiry | • No new functionality or interface changes. |
| Swath Input/output | • Add unsigned long type to store large number of elements in attributes.  
• Change 'start, stride, edge' parameter type to allow for very large numbers. |
| Swath Subset | • No new functionality or interface changes. |
| New interfaces | • SWinqdatatype() retrieve information about data type of a data field  
• SWraopen(), SWraread(), SWrawrite(), SWraclose() - Support HDF5 ragged array structures. |
| Grid Access | • Modify to implement new HDF5 file structure.  
• Create a grid with a specified name |
| Grid Definition | • GDdeffield() modified to add 'maxdimlist', a comma separated list of maximum dimensions - same as for Swath  
• Modify functions to support HDF5 chunking: GDdeftile() and GDdefcomp() and GDdefchunk(). |
| Grid Input/Output | • Add unsigned long type to store large number of elements in attributes. |
| Grid Inquiry | • No new functionality or interface changes |
| Grid Subset | • No new functionality or interface changes |
| Grid Tiling | • No new functionality or interface changes |
| Utilities | • Add function for determining whether a file is HDF4 or HDF5 based. |
| EOSView | • Modified to accommodate new HDF5-based HDF-EOS library. This is a new tool  
• which will compliment the original HDF4 - based tool. Removed HDF4 support. Added about 200 lines of code for HDF5 support. |

Selected user considerations are listed below.

The routines SWopen(), SWcreate(), GDopen(), GDcreate() implement a new file structure designed for the V3.0 of the HDF-EOS library.

The field definition routines, SWdefgeofield(), SWdefdatafield(), and GDdeffield(), have an additional parameter 'maxdimlist', a comma separated list of maximum sizes of dimensions. The parameter 'maxdimlist' is reserved for future use. Since in HDF5 each dimension of a dataset can be appendable (extendible), the definition of a dataset should include the maximum size (or unlimited size) the corresponding dimension can be expanded to. Passing a NULL as a 'maxdimlist' means that the dimension is not appendable, and its maximum size is the same as its actual size. The unlimited dimension can be specified e.g. by a call to SWdefdim( sw_id,
"Unlim", H5S_UNLIMITED). Then, in the call to e.g. SWdefdatafield() we should use for the 'maxdimlist' parameter the value "Unlim".

HDF5 requires the user to use chunking in order to define extendible datasets. Chunking makes it possible to extend datasets efficiently, without having to reorganize storage excessively. The corresponding (new) calls are SWdefchunk() and GDdeftile(). These calls should be used before the familiar old-library call to SWdefcomp() and GDdefcomp(), respectively. The latter set the field compression for all subsequent field definitions.

In the input/output routines SWwriteattr() and GDwriteattr(), the data type of the fourth parameter, number of values to store in attribute, is now such that it allows to store arbitrary big number of elements in attribute. Also, the routines SWwritefield(), SWreadfield(), GDwritefield(), and GDreadfield() allow for the parameters 'start', 'stride', and 'edge' to use very big numbers.

There are two new inquiry routines SWinqdatatype() and GDinqdatatype() that, for a specified field, retrieve explicit information about data type, including number of bytes allocated for each element of the corresponding dataset.

The following ragged array routines have been added for the swath interface:

- SWradefine() - sets up the ragged array within the swath under the "Data Fields" group
- SWraopen() - opens up a specified ragged array
- SWrawrite() - writes in the data to a specified ragged array
- SWraread() - reads out the data from a specified ragged array
- SWraclose() - closes the ragged array
4. Swath Data

4.1 Introduction

The Swath concept for HDF-EOS is based on a typical satellite swath, where an instrument takes a series of scans perpendicular to the ground track of the satellite as it moves along that ground track. Figure 4-1 below shows this traditional view of a swath.

![Swath Diagram](image)

*Figure 4-1. A Typical Satellite Swath: Scanning Instrument*

Another type of data that the Swath is equally well suited to arises from a sensor that measures a vertical profile, instead of scanning across the ground track. The resulting data resembles a standard Swath tipped up on its edge. Figure 4-1 shows how such a Swath might look.

In fact, the two approaches shown in Figures 4-1 and 4-2 can be combined to manage a profiling instrument that scans across the ground track. The result would be a three dimensional array of measurements where two of the dimensions correspond to the standard scanning dimensions (along the ground track and across the ground track), and the third dimension represents a height above the Earth or a range from the sensor. The "horizontal" dimensions can be handled as normal geographic dimensions, while the third dimension can be handled as a special "vertical" dimension.
A standard Swath is made up of four primary parts: data fields, geolocation fields, dimensions, and dimension maps. An optional fifth part called an index can be added to support certain kinds of access to Swath data. Each of the parts of a Swath is described in detail in the following subsections.

4.1.1 Data Fields

Data fields are the main part of a Swath from a science perspective. Data fields usually contain the raw data (often as counts) taken by the sensor or parameters derived from that data on a value-for-value basis. All the other parts of the Swath exist to provide information about the data fields or to support particular types of access to them. Data fields typically are two-dimensional arrays, but can have as few as one dimension or as many as eight, in the current library implementation. They can have any valid C data type.
4.1.2 Geolocation Fields

Geolocation fields allow the Swath to be accurately tied to particular points on the Earth’s surface. To do this, the Swath interface requires the presence of at least a time field (“Time”) or a latitude/longitude field pair (“Latitude” and “Longitude”). Geolocation fields must be either one- or two-dimensional and can have any data type.

4.1.3 Dimensions

Dimensions define the axes of the data and geolocation fields by giving them names and sizes. In using the library, dimensions must be defined before they can be used to describe data or geolocation fields.

Every axis of every data or geolocation field, then, must have a dimension associated with it. However, there is no requirement that they all be unique. In other words, different data and geolocation fields may share the same named dimension. In fact, sharing dimension names allows the Swath interface to make some assumptions about the data and geolocation fields involved which can reduce the complexity of the file and simplify the program creating or reading the file.

4.1.4 Dimension Maps

Dimension maps are the glue that holds the Swath together. They define the relationship between data fields and geolocation fields by defining, one-by-one, the relationship of each dimension of each geolocation field with the corresponding dimension in each data field. In cases where a data field and a geolocation field share a named dimension, no explicit dimension map is needed. In cases where a data field has more dimensions than the geolocation fields, the “extra” dimensions are left unmapped.

In many cases, the size of a geolocation dimension will be different from the size of the corresponding data dimension. To take care of such occurrences, there are two pieces of information that must be supplied when defining a dimension map: the offset and the increment. The offset tells how far along a data dimension you must travel to find the first point to have a corresponding entry along the geolocation dimension. The increment tells how many points to travel along the data dimension before the next point is found for which there is a corresponding entry along the geolocation dimension. Figure 4-3 depicts a dimension map.

---

1 “Co-latitude” may be substituted for “Latitude.”
The “data skipping” method described above works quite well if there are fewer regularly spaced geolocation points than data points along a particular pair of mapped dimensions of a Swath. It is conceivable, however, that the reverse is true – that there are more regularly spaced geolocation points than data points. In that event, both the offset and increment should be expressed as negative values to indicate the reversed relationship. The result is shown in Figure 4-4. Note that in the reversed relationship, the offset and increment are applied to the geolocation dimension rather than the data dimension.

Figure 4-3. A “Normal” Dimension Map

Figure 4-4. A “Backwards” Dimension Map
4.1.5 Index

The index was designed specifically for Landsat 7 data products. These products require geolocation information that does not repeat at regular intervals throughout the Swath. The index allows the Swath to be broken into unequal length scenes which can be individually geolocated.

For this version of the HDF-EOS library, there is no particular content required for the index. It is quite likely that a later version of the library will impose content requirements on the index in an effort to standardize its use.

4.2 Applicability

The Swath data model is most useful for satellite [or similar] data at a low level of processing. The Swath model is best suited to data at EOS processing levels 1A, 1B, and 2.

4.3 The Swath Data Interface

The SW interface consists of routines for storing, retrieving, and manipulating data in swath data sets.

4.3.1 SW API Routines

All C routine names in the swath data interface have the prefix “SW” and the equivalent FORTRAN routine names are prefixed by “sw.” The SW routines are classified into the following categories:

- **Access routines** initialize and terminate access to the SW interface and swath data sets (including opening and closing files).
- **Definition** routines allow the user to set key features of a swath data set.
- **Basic I/O** routines read and write data and metadata to a swath data set.
- **Inquiry** routines return information about data contained in a swath data set.
- **Subset** routines allow reading of data from a specified geographic region.

The SW function calls are listed in Table 4-1 and are described in detail in the Software Reference Guide that accompanies this document. The page number column in the following table refers to the Software Reference Guide.
<table>
<thead>
<tr>
<th>Category</th>
<th>Routine Name</th>
<th>FORTRAN</th>
<th>Description</th>
<th>Page Nos.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>SWopen</td>
<td>swopen</td>
<td>Opens or creates HDF file in order to create, read, or write a swath</td>
<td>2-44</td>
</tr>
<tr>
<td></td>
<td>SWcreate</td>
<td>swcreate</td>
<td>Creates a swath within the file</td>
<td>2-6</td>
</tr>
<tr>
<td></td>
<td>Swattach</td>
<td>swattach</td>
<td>Attaches to an existing swath within the file</td>
<td>2-2</td>
</tr>
<tr>
<td></td>
<td>SWdetach</td>
<td>swdetach</td>
<td>Detaches from swath interface</td>
<td>2-24</td>
</tr>
<tr>
<td></td>
<td>SWclose</td>
<td>swclose</td>
<td>Closes file</td>
<td>2-4</td>
</tr>
<tr>
<td></td>
<td>SWdefdim</td>
<td>swdefdim</td>
<td>Defines a new dimension within the swath</td>
<td>2-13</td>
</tr>
<tr>
<td></td>
<td>SWdefdimmap</td>
<td>swdefmap</td>
<td>Defines the mapping between the geolocation and data dimensions</td>
<td>2-14</td>
</tr>
<tr>
<td></td>
<td>SWdefidxmap</td>
<td>swdefimap</td>
<td>Defines a non-regular mapping between the geolocation and data dimension</td>
<td>2-18</td>
</tr>
<tr>
<td></td>
<td>SWdefgeofield</td>
<td>swdefgflfd</td>
<td>Defines a new geolocation field within the swath</td>
<td>2-16</td>
</tr>
<tr>
<td></td>
<td>SWdefdatafield</td>
<td>swdefdatafield</td>
<td>Defines a new data field within the swath</td>
<td>2-11</td>
</tr>
<tr>
<td></td>
<td>SWdefcomp</td>
<td>swdefcomp</td>
<td>Defines a field compression scheme</td>
<td>2-9</td>
</tr>
<tr>
<td>Basic I/O</td>
<td>SWwritefield</td>
<td>swwrfld</td>
<td>Writes data to a swath field</td>
<td>2-65</td>
</tr>
<tr>
<td></td>
<td>SWreadfield</td>
<td>swrdfld</td>
<td>Reads data from a swath field</td>
<td>2-54</td>
</tr>
<tr>
<td></td>
<td>SWwriteattr</td>
<td>swwrattr</td>
<td>Writes/updates attribute in a swath</td>
<td>2-63</td>
</tr>
<tr>
<td></td>
<td>SWreadattr</td>
<td>swrdattr</td>
<td>Reads attribute from a swath</td>
<td>2-53</td>
</tr>
<tr>
<td></td>
<td>SWsetfillvalue</td>
<td>swsetfill</td>
<td>sets fill value for the specified field</td>
<td>2-60</td>
</tr>
<tr>
<td></td>
<td>SWGetfillvalue</td>
<td>swgetfill</td>
<td>Retrieves fill value for the specified field</td>
<td>2-31</td>
</tr>
<tr>
<td>Inquiry</td>
<td>Swinqdims</td>
<td>swinqdims</td>
<td>Retrieves information about dimensions defined in swath</td>
<td>2-37</td>
</tr>
<tr>
<td></td>
<td>Swinqmaps</td>
<td>swinqmaps</td>
<td>Retrieves information about the geolocation relations defined</td>
<td>2-40</td>
</tr>
<tr>
<td></td>
<td>Swinqidxmaps</td>
<td>swinqmaps</td>
<td>Retrieves information about the indexed geolocation/data mappings defined</td>
<td>2-39</td>
</tr>
<tr>
<td></td>
<td>SWinqgeofields</td>
<td>swinqgflids</td>
<td>Retrieves information about the geolocation fields defined</td>
<td>2-38</td>
</tr>
<tr>
<td></td>
<td>SWinqdatafields</td>
<td>swinqdflids</td>
<td>Retrieves information about the data fields defined</td>
<td>2-35</td>
</tr>
<tr>
<td></td>
<td>SWinqdatatypetype</td>
<td>swinqdtype</td>
<td>Retrieves information about data type of a field</td>
<td>2-36</td>
</tr>
<tr>
<td></td>
<td>SWinqattrs</td>
<td>swinqattrs</td>
<td>Retrieves number and names of attributes defined</td>
<td>2-34</td>
</tr>
<tr>
<td></td>
<td>SWnentries</td>
<td>swnentries</td>
<td>Returns number of entries and descriptive string buffer size for a specified entity</td>
<td>2-43</td>
</tr>
<tr>
<td></td>
<td>SWdiminfo</td>
<td>swdiminfo</td>
<td>Retrieve size of specified dimension</td>
<td>2-25</td>
</tr>
<tr>
<td></td>
<td>SWmapinfo</td>
<td>swmapinfo</td>
<td>Retrieve offset and increment of specified geolocation mapping</td>
<td>2-42</td>
</tr>
<tr>
<td></td>
<td>SWidmapinfo</td>
<td>swimapinfo</td>
<td>Retrieve offset and increment of specified geolocation mapping</td>
<td>2-33</td>
</tr>
<tr>
<td></td>
<td>SWattrinfo</td>
<td>swattrinfo</td>
<td>Returns information about swath attributes</td>
<td>2-3</td>
</tr>
<tr>
<td></td>
<td>SWfieldinfo</td>
<td>swfieldinfo</td>
<td>Retrieve information about a specific geolocation or data field</td>
<td>2-29</td>
</tr>
<tr>
<td></td>
<td>SWcompinfo</td>
<td>swcompinfo</td>
<td>Retrieve compression information about a field</td>
<td>2-5</td>
</tr>
<tr>
<td></td>
<td>SWingswath</td>
<td>swingswath</td>
<td>Retrieves number and names of swaths in file</td>
<td>2-41</td>
</tr>
<tr>
<td></td>
<td>Swregionindex</td>
<td>swregidx</td>
<td>Returns information about the swath region ID</td>
<td>2-56</td>
</tr>
<tr>
<td>Ragged</td>
<td>SWupdateidxmap</td>
<td>swupmap</td>
<td>Update map index for a specified region</td>
<td>2-61</td>
</tr>
<tr>
<td>Arrays</td>
<td>SWradefine</td>
<td>swradefine</td>
<td>Defines ragged array</td>
<td>2-48</td>
</tr>
<tr>
<td></td>
<td>SWraopen</td>
<td>swraopen</td>
<td>Opens ragged array</td>
<td>2-49</td>
</tr>
<tr>
<td></td>
<td>SWrawrite</td>
<td>swrawrite</td>
<td>Writes data to the ragged array</td>
<td>2-52</td>
</tr>
<tr>
<td></td>
<td>SWread</td>
<td>swread</td>
<td>Reads out data from the ragged array</td>
<td>2-50</td>
</tr>
<tr>
<td></td>
<td>SWrclose</td>
<td>swrclose</td>
<td>Closes the ragged array</td>
<td>2-47</td>
</tr>
<tr>
<td>Category</td>
<td>Routine Name</td>
<td>Description</td>
<td>Page Nos.</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>Subset</td>
<td>SWgeomapinfo</td>
<td>swgmapinfo</td>
<td>Retrieves type of dimension mapping when first dimension is geodim</td>
<td>2-32</td>
</tr>
<tr>
<td>Subset</td>
<td>SWdefboxregion</td>
<td>swdefboxreg</td>
<td>Define region of interest by latitude/longitude</td>
<td>2-7</td>
</tr>
<tr>
<td>Subset</td>
<td>SWregioninfo</td>
<td>swreginfo</td>
<td>Returns information about defined region</td>
<td>2-58</td>
</tr>
<tr>
<td>Subset</td>
<td>S Weltectregion</td>
<td>swextreg</td>
<td>Read a region of interest from a field</td>
<td>2-28</td>
</tr>
<tr>
<td>Subset</td>
<td>SWdefftimeperiod</td>
<td>swdefftimeper</td>
<td>Define a time period of interest</td>
<td>2-19</td>
</tr>
<tr>
<td>Subset</td>
<td>SWperiodinfo</td>
<td>swperinfo</td>
<td>Returns information about a defined time period</td>
<td>2-45</td>
</tr>
<tr>
<td>Subset</td>
<td>SWextractperiod</td>
<td>swextper</td>
<td>Extract a defined time period</td>
<td>2-27</td>
</tr>
<tr>
<td>Subset</td>
<td>SWdefvrtregion</td>
<td>swdefvrtreg</td>
<td>Define a region of interest by vertical field</td>
<td>2-21</td>
</tr>
<tr>
<td>Subset</td>
<td>SWdupregion</td>
<td>swdupreg</td>
<td>Duplicate a region or time period</td>
<td>2-26</td>
</tr>
</tbody>
</table>

### 4.3.2 File Identifiers

As with all HDF-EOS interfaces, file identifiers in the SW interface are 32-bit values, each uniquely identifying one open data file. They are not interchangeable with other file identifiers created with other interfaces.

### 4.3.3 Swath Identifiers

Before a swath data set is accessed, it is identified by a name which is assigned to it upon its creation. The name is used to obtain a **swath identifier**. After a swath data set has been opened for access, it is uniquely identified by its swath identifier.

### 4.4 Programming Model

The programming model for accessing a swath data set through the SW interface is as follows:

1. Open the file and initialize the SW interface by obtaining a file id from a file name.
2. Open OR create a swath data set by obtaining a swath id from a swath name.
3. Perform desired operations on the data set.
4. Close the swath data set by disposing of the swath id.
5. Terminate swath access to the file by disposing of the file id.

To access a single swath data set that already exists in an HDF-EOS file, the calling program must contain the following sequence of C calls:

```c
file_id = SWopen(filename, access_mode);
sw_id = SWattach(file_id, swath_name);
<Optional operations>
status = SWdetach(sw_id);
status = SWclose(file_id);
```
To access several files at the same time, a calling program must obtain a separate id for each file to be opened. Similarly, to access more than one swath data set, a calling program must obtain a separate swath id for each data set. For example, to open two data sets stored in two files, a program would execute the following series of C function calls:

```c
file_id_1 = SWopen(filename_1, access_mode);
sw_id_1 = SWattach(file_id_1, swath_name_1);
file_id_2 = SWopen(filename_2, access_mode);
sw_id_2 = SWattach(file_id_2, swath_name_2);
<Optional operations>
status = SWdetach(sw_id_1);
status = SWclose(file_id_1);
status = SWdetach(sw_id_2);
status = SWclose(file_id_2);
```

Because each file and swath data set is assigned its own identifier, the order in which files and data sets are accessed is very flexible. However, it is very important that the calling program individually discard each identifier before terminating. Failure to do so can result in empty or, even worse, invalid files being produced.
5. Grid Data

5.1 Introduction

This section will describe the routines available for storing and retrieving HDF-EOS Grid Data. A Grid data set is similar to a swath in that it contains a series of data fields of two or more dimensions. The main difference between a Grid and a Swath is in the character of their geolocation information.

As described in Section 4, swaths carry geolocation information as a series of individually located points (tie points or ground control points). Grids, though, carry their geolocation in a much more compact form. A grid merely contains a set of projection equations (or references to them) along with their relevant parameters. Together, these relatively few pieces of information define the location of all points in the grid. The equations and parameters can then be used to compute the latitude and longitude for any point in the grid.

![Figure 5-1. A Data Field in a Mercator-projected Grid](image)

In loose terms, each data field constitutes a map in a given standard projection. Although there may be many independent Grids in a single HDF-EOS file, within each Grid only one projection may be chosen for application to all data fields. Figures 5-1 and 5-2 show how a single data field may look in a Grid using two common projections.

There are three important features of a Grid data set: the data fields, the dimensions, and the projection. Each of these is discussed in detail in the following subsections.
5.1.1 Data Fields

The data fields are, of course, the most important part of the Grid. Data fields in a Grid data set are rectilinear arrays of two or more dimensions. Most commonly, they are simply two-dimensional rectangular arrays. Generally, each field contains data of similar scientific nature which must share the same data type. The data fields are related to each other by common geolocation. That is, a single set of geolocation information is used for all data fields within one Grid data set.

5.1.2 Dimensions

Dimensions are used to relate data fields to each other and to the geolocation information. To be interpreted properly, each data field must make use of the two predefined dimensions: “XDim” and “YDim”. These two dimensions are defined when the grid is created and are used to refer to the X and Y dimensions of the chosen projection (see 5.1.3 below). Although there is no practical limit on the number of dimensions a data field in a Grid data set may have, it is not likely that many fields will need more than three: the predefined dimensions “XDim” and “YDim” and a third dimension for depth or height.

5.1.3 Projections

The projection is really the heart of the Grid. Without the use of a projection, the Grid would not be substantially different from a Swath. The projection provides a convenient way to encode geolocation information as a set of mathematical equations which are capable of transforming Earth coordinates (latitude and longitude) to X-Y coordinates on a sheet of paper.

Figure 5-2. A Data Field in an Interrupted Goode's Homolosine-Projected Grid
The choice of a projection to be used for a Grid is a critical decision for a data product designer. There is a large number of projections that have been used throughout history. In fact, some projections date back to ancient Greece. For the purposes of this release of HDF-EOS, however, only six families of projections are supported: Geographic, Interrupted Goode’s Homolosine, Polar Stereographic, Universal Transverse Mercator, Space Oblique, and Lambert Azimuthal Equal Area. These projections coincide with those supported by the SDP Toolkit for ECS Release B.

The producer’s choice of a projection should be governed by knowledge of the specific properties of each projection and a thorough understanding of the requirements of the data set’s users. Two excellent resources for information on projections and their properties are the USGS Professional Papers cited in Section 2.2 “Related Documents.”

This release of HDF-EOS assumes that the data producer will use to create the data the General Coordinate Transformation Package (GCTP), a library of projection software available from the U.S. Geological Survey. This manual will not attempt to explain the use of GCTP. Adequate documentation accompanies the GCTP source code. For the purposes of this Grid interface, the data are assumed to have already been projected. The Grid interface allows the data producer to specify the exact GCTP parameters used to perform the projection and will provide for basic subsetting of the data fields by latitude/longitude bounding box.

See section below for further details on the usage of the GCTP package.

5.2 Applicability

The Grid data model is intended for data processed at a high level. It is most applicable to data at EOS processing levels 3 and 4.

5.3 The Grid Data Interface

The GD interface consists of routines for storing, retrieving, and manipulating data in grid data sets.

5.3.1 GD API Routines

All C routine names in the grid data interface have the prefix “GD” and the equivalent FORTRAN routine names are prefixed by “gd.” The GD routines are classified into the following categories:

- **Access routines** initialize and terminate access to the GD interface and grid data sets (including opening and closing files).
- **Definition routines** allow the user to set key features of a grid data set.
- **Basic I/O routines** read and write data and metadata to a grid data set.
- **Inquiry routines** return information about data contained in a grid data set.
- **Subset routines** allow reading of data from a specified geographic region.
The GD function calls are listed in Table 6-1 and are described in detail in the Software Reference Guide that accompanies this document. The page number column in the following table refers to the Software Reference Guide.

Table 5-1. Summary of the Grid Interface (1 of 2)

<table>
<thead>
<tr>
<th>Category</th>
<th>Routine Name</th>
<th>FORTRAN</th>
<th>Description</th>
<th>Page Nos.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>GDopen</td>
<td>gdopen</td>
<td>Creates a new file or opens an existing one</td>
<td>2-108</td>
</tr>
<tr>
<td></td>
<td>GDcreate</td>
<td>gdccreate</td>
<td>Creates a new grid in the file</td>
<td>2-72</td>
</tr>
<tr>
<td></td>
<td>GDattach</td>
<td>gdattach</td>
<td>Attaches to a grid</td>
<td>2-68</td>
</tr>
<tr>
<td></td>
<td>GDdetach</td>
<td>gddetach</td>
<td>Detaches from grid interface</td>
<td>2-92</td>
</tr>
<tr>
<td></td>
<td>GDclose</td>
<td>gdclose</td>
<td>Closes file</td>
<td>2-70</td>
</tr>
<tr>
<td></td>
<td>GDdeforigin</td>
<td>gddeforigin</td>
<td>Defines origin of grid</td>
<td>2-81</td>
</tr>
<tr>
<td></td>
<td>GDdefdim</td>
<td>gddefdim</td>
<td>Defines dimensions for a grid</td>
<td>2-78</td>
</tr>
<tr>
<td></td>
<td>GDdefproj</td>
<td>gddefproj</td>
<td>Defines projection of grid</td>
<td>2-83</td>
</tr>
<tr>
<td></td>
<td>GDdefpixreg</td>
<td>gddefpixreg</td>
<td>Defines pixel registration within grid cell</td>
<td>2-82</td>
</tr>
<tr>
<td></td>
<td>GDdeffield</td>
<td>gddeffield</td>
<td>Defines data fields to be stored in a grid</td>
<td>2-79</td>
</tr>
<tr>
<td></td>
<td>GDdefcomp</td>
<td>gddefcomp</td>
<td>Defines a field compression scheme</td>
<td>2-76</td>
</tr>
<tr>
<td></td>
<td>GDwritefield</td>
<td>gdwrfld</td>
<td>Writes data to a grid field</td>
<td>2-121</td>
</tr>
<tr>
<td></td>
<td>GDreadfield</td>
<td>gdrdfld</td>
<td>Reads data from a grid field</td>
<td>2-114</td>
</tr>
<tr>
<td></td>
<td>GDwriteattr</td>
<td>gdwrattr</td>
<td>Writes/updates attribute in a grid</td>
<td>2-119</td>
</tr>
<tr>
<td></td>
<td>GDreadattr</td>
<td>gdrdattr</td>
<td>Reads attribute from a grid</td>
<td>2-113</td>
</tr>
<tr>
<td></td>
<td>GDsetfillvalue</td>
<td>gdsetfill</td>
<td>sets fill value for the specified field</td>
<td>2-118</td>
</tr>
<tr>
<td></td>
<td>GDgetfillvalue</td>
<td>gdgetfill</td>
<td>Retrieves fill value for the specified field</td>
<td>2-98</td>
</tr>
<tr>
<td>Inquiry</td>
<td>GDinqdims</td>
<td>gdinqdims</td>
<td>Retrieves information about dimensions defined in grid</td>
<td>2-104</td>
</tr>
<tr>
<td></td>
<td>GDinqfields</td>
<td>gdinqfields</td>
<td>Retrieves information about the data fields defined in grid</td>
<td>2-105</td>
</tr>
<tr>
<td></td>
<td>GDinqattrs</td>
<td>gdinqattrs</td>
<td>Retrieves number and names of attributes defined</td>
<td>2-102</td>
</tr>
<tr>
<td></td>
<td>GDnentries</td>
<td>gdnentries</td>
<td>Returns number of entries and descriptive string buffer size for a specified entity</td>
<td>2-107</td>
</tr>
<tr>
<td></td>
<td>GDgridinfo</td>
<td>gdgridinfo</td>
<td>Returns dimensions of grid and X-Y coordinates of corners</td>
<td>2-101</td>
</tr>
<tr>
<td></td>
<td>GDprojinfo</td>
<td>gdprojinfo</td>
<td>Returns all GCTP projection information</td>
<td>2-112</td>
</tr>
<tr>
<td></td>
<td>GDdiminfo</td>
<td>gddiminfo</td>
<td>Retrieves size of specified dimension.</td>
<td>2-93</td>
</tr>
<tr>
<td></td>
<td>GDcomppinfo</td>
<td>gdcomppinfo</td>
<td>Retrieve compression information about a field</td>
<td>2-71</td>
</tr>
<tr>
<td></td>
<td>GDFieldinfo</td>
<td>gdfldinfo</td>
<td>Retrieves information about a specific geolocation or data field in the grid</td>
<td>2-96</td>
</tr>
<tr>
<td></td>
<td>GDinqdatatype</td>
<td>gdinqdatatype</td>
<td>Retrieves information about data type of a field</td>
<td>2-103</td>
</tr>
<tr>
<td></td>
<td>GDingrid</td>
<td>gdingrid</td>
<td>Retrieves number and names of grids in file</td>
<td>2-106</td>
</tr>
<tr>
<td></td>
<td>GDattrinfo</td>
<td>gdattrinfo</td>
<td>Returns information about grid attributes</td>
<td>2-69</td>
</tr>
<tr>
<td></td>
<td>GDDorigininfo</td>
<td>gdorginfo</td>
<td>Return information about grid origin</td>
<td>2-110</td>
</tr>
<tr>
<td></td>
<td>GDpixreginfo</td>
<td>gdpixreginfo</td>
<td>Return pixel registration information for given grid</td>
<td>2-111</td>
</tr>
<tr>
<td></td>
<td>GDdefboxregion</td>
<td>gddefboxreg</td>
<td>Define region of interest by latitude/longitude</td>
<td>2-75</td>
</tr>
<tr>
<td></td>
<td>GDTregioninfo</td>
<td>gdreginfo</td>
<td>Returns information about a defined region</td>
<td>2-116</td>
</tr>
</tbody>
</table>
Table 5-1. Summary of the Grid Interface (2 of 2)

<table>
<thead>
<tr>
<th>Category</th>
<th>Routine Name</th>
<th>Description</th>
<th>Page Nos.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>FORTRAN</td>
<td></td>
</tr>
<tr>
<td>Subset</td>
<td>GDextractregion gdextrreg</td>
<td>read a region of interest from a field</td>
<td>2-95</td>
</tr>
<tr>
<td></td>
<td>GDdeftimeperiod  gddeftmeper</td>
<td>Define a time period of interest</td>
<td>2-87</td>
</tr>
<tr>
<td></td>
<td>GDdefvrtregion  gddefvrtreg</td>
<td>Define a region of interest by vertical field</td>
<td>2-89</td>
</tr>
<tr>
<td></td>
<td>GDgetpixels     gdgetpix</td>
<td>get row/columns for lon/lat pairs</td>
<td>2-99</td>
</tr>
<tr>
<td>Tiling</td>
<td>GDdupregion     gddupreg</td>
<td>Duplicate a region or time period</td>
<td>2-94</td>
</tr>
<tr>
<td></td>
<td>GDdeffile       gddefile</td>
<td>Define a tiling scheme</td>
<td>2-85</td>
</tr>
</tbody>
</table>

5.3.2 File Identifiers

As with all HDF-EOS interfaces, file identifiers in the GD interface are 32-bit values, each uniquely identifying one open data file. They are not interchangeable with other file identifiers created with other interfaces.

5.3.3 Grid Identifiers

Before a grid data set is accessed, it is identified by a name which is assigned to it upon its creation. The name is used to obtain a grid identifier. After a grid data set has been opened for access, it is uniquely identified by its grid identifier.

5.4 Programming Model

The programming model for accessing a grid data set through the GD interface is as follows:

1. Open the file and initialize the GD interface by obtaining a file id from a file name.
2. Open OR create a grid data set by obtaining a grid id from a grid name.
3. Perform desired operations on the data set.
5. Terminate grid access to the file by disposing of the file id.

To access a single grid data set that already exists in an HDF-EOS file, the calling program must contain the following sequence of C calls:

```c
file_id = GDopen(filename, access_mode);
gd_id = GDattach(file_id, grid_name);
<Optional operations>
status = GDdetach(gd_id);
status = GDclose(file_id);
```

To access several files at the same time, a calling program must obtain a separate id for each file to be opened. Similarly, to access more than one grid data set, a calling program must obtain a
separate grid id for each data set. For example, to open two data sets stored in two files, a program would execute the following series of C function calls:

```c
file_id_1 = GDopen(filename_1, access_mode);
gd_id_1 = GDattach(file_id_1, grid_name_1);
file_id_2 = GDopen(filename_2, access_mode);
gd_id_2 = GDattach(file_id_2, grid_name_2);
Optional operations:
status = GDDetach(gd_id_1);
status = GDclose(file_id_1);
status = GDDetach(gd_id_2);
status = GDclose(file_id_2);
```

Because each file and grid data set is assigned its own identifier, the order in which files and data sets are accessed is very flexible. However, it is very important that the calling program individually discard each identifier before terminating. Failure to do so can result in empty or, even worse, invalid files being produced.

### 5.5 GCTP Usage

The HDF-EOS Grid API uses the U.S. Geological Survey General Cartographic Transformation Package (GCTP) to define and subset grid structures. This section described codes used by the package.

#### 5.5.1 GCTP Projection Codes

The following GCTP projection codes are used in the grid API described in Section 7 below:

- **GCTP_GEO** (0) Geographic
- **GCTP_UTM** (1) Universal Transverse Mercator
- **GCTP_LAMCC** (4) Lambert Conformal Conic
- **GCTP_PS** (6) Polar Stereographic
- **GCTP_POLYC** (7) Polyconic
- **GCTP_TM** (9) Transverse Mercator
- **GCTP_LAMAZ** (11) Lambert Azimuthal Equal Area
- **GCTP_HOM** (20) Hotine Oblique Mercator
- **GCTP_SOM** (22) Space Oblique Mercator
- **GCTP_GOOD** (24) Interrupted Goode Homolosine
- **GCTP_ISINUS** (99) Intergerized Sinusoidal Projection*

* The Intergerized Sinusoidal Projection is not part of the original GCTP package. It has been added by ECS. See *Level-3 SeaWiFS Data Products: Spatial and Temporal Binning Algorithms*. Additional references are provided in Section 2.

Note that other projections supported by GCTP will be adapted for HDF-EOS Version 3 new user requirements are surfaced. For further details on the GCTP projection package, please refer
to Section 6.3.4 and Appendix G of the SDP Toolkit Users Guide for the ECS Project, June 1998, (333-CD-500-001)

### 5.5.2 UTM Zone Codes

The Universal Transverse Mercator (UTM) Coordinate System uses zone codes instead of specific projection parameters. The table that follows lists UTM zone codes as used by GCTP Projection Transformation Package. C.M. is Central Meridian

<table>
<thead>
<tr>
<th>Zone</th>
<th>C.M.</th>
<th>Range</th>
<th>Zone</th>
<th>C.M.</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>177W</td>
<td>180W-174W</td>
<td>31</td>
<td>003E</td>
<td>000E-006E</td>
</tr>
<tr>
<td>02</td>
<td>171W</td>
<td>174W-168W</td>
<td>32</td>
<td>009E</td>
<td>006E-012E</td>
</tr>
<tr>
<td>03</td>
<td>165W</td>
<td>168W-162W</td>
<td>33</td>
<td>015E</td>
<td>012E-018E</td>
</tr>
<tr>
<td>04</td>
<td>159W</td>
<td>162W-156W</td>
<td>34</td>
<td>021E</td>
<td>018E-024E</td>
</tr>
<tr>
<td>05</td>
<td>153W</td>
<td>156W-150W</td>
<td>35</td>
<td>027E</td>
<td>024E-030E</td>
</tr>
<tr>
<td>06</td>
<td>147W</td>
<td>150W-144W</td>
<td>36</td>
<td>033E</td>
<td>030E-036E</td>
</tr>
<tr>
<td>07</td>
<td>141W</td>
<td>144W-138W</td>
<td>37</td>
<td>039E</td>
<td>036E-042E</td>
</tr>
<tr>
<td>08</td>
<td>135W</td>
<td>138W-132W</td>
<td>38</td>
<td>045E</td>
<td>042E-048E</td>
</tr>
<tr>
<td>09</td>
<td>129W</td>
<td>132W-126W</td>
<td>39</td>
<td>051E</td>
<td>048E-054E</td>
</tr>
<tr>
<td>10</td>
<td>123W</td>
<td>126W-120W</td>
<td>40</td>
<td>057E</td>
<td>054E-060E</td>
</tr>
<tr>
<td>11</td>
<td>117W</td>
<td>120W-114W</td>
<td>41</td>
<td>063E</td>
<td>060E-066E</td>
</tr>
<tr>
<td>12</td>
<td>111W</td>
<td>114W-108W</td>
<td>42</td>
<td>069E</td>
<td>066E-072E</td>
</tr>
<tr>
<td>13</td>
<td>105W</td>
<td>108W-102W</td>
<td>43</td>
<td>075E</td>
<td>072E-078E</td>
</tr>
<tr>
<td>14</td>
<td>099W</td>
<td>102W-096W</td>
<td>44</td>
<td>081E</td>
<td>078E-084E</td>
</tr>
<tr>
<td>15</td>
<td>093W</td>
<td>096W-090W</td>
<td>45</td>
<td>087E</td>
<td>084E-090E</td>
</tr>
<tr>
<td>16</td>
<td>087W</td>
<td>090W-084W</td>
<td>46</td>
<td>093E</td>
<td>090E-096E</td>
</tr>
<tr>
<td>17</td>
<td>081W</td>
<td>084W-078W</td>
<td>47</td>
<td>099E</td>
<td>096E-102E</td>
</tr>
<tr>
<td>18</td>
<td>075W</td>
<td>078W-072W</td>
<td>48</td>
<td>105E</td>
<td>102E-108E</td>
</tr>
<tr>
<td>19</td>
<td>069W</td>
<td>072W-066W</td>
<td>49</td>
<td>111E</td>
<td>108E-114E</td>
</tr>
<tr>
<td>20</td>
<td>063W</td>
<td>066W-060W</td>
<td>50</td>
<td>117E</td>
<td>114E-120E</td>
</tr>
<tr>
<td>21</td>
<td>057W</td>
<td>060W-054W</td>
<td>51</td>
<td>123E</td>
<td>120E-126E</td>
</tr>
<tr>
<td>22</td>
<td>051W</td>
<td>054W-048W</td>
<td>52</td>
<td>129E</td>
<td>126E-132E</td>
</tr>
<tr>
<td>23</td>
<td>045W</td>
<td>048W-042W</td>
<td>53</td>
<td>135E</td>
<td>132E-138E</td>
</tr>
<tr>
<td>24</td>
<td>039W</td>
<td>042W-036W</td>
<td>54</td>
<td>141E</td>
<td>138E-144E</td>
</tr>
<tr>
<td>25</td>
<td>033W</td>
<td>036W-030W</td>
<td>55</td>
<td>147E</td>
<td>144E-150E</td>
</tr>
</tbody>
</table>
5.5.3 GCTP Spheroid Codes

Clarke 1866 (default) (0)
Clarke 1880 (1)
Bessel (2)
International 1967 (3)
International 1909 (4)
WGS 72 (5)
Everest (6)
WGS 66 (7)
GRS 1980 (8)
Airy (9)
Modified Airy (10)
Modified Everest (11)
WGS 84 (12)
Southeast Asia (13)
Australian National (14)
Krassovsky (15)
Hough (16)
Mercury 1960 (17)
Modified Mercury 1968 (18)
Sphere of Radius 6370997 m (19)
### 5.5.4 Projection Parameters

**Table 5-2. Projection Transformation Package Projection Parameters**

<table>
<thead>
<tr>
<th>Code &amp; Projection Id</th>
<th>Array Element</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Geographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 U T M</td>
<td></td>
<td>Lon/Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Lambert Conformal C</td>
<td>SMajor, SMinor, STDPR1, STDPR2, CentMer, OriginLat, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Polar Stereographic</td>
<td>SMajor, SMinor, LongPol, TrueScale, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 Polyconic</td>
<td>SMajor, SMinor, CentMer, OriginLat, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 Transverse Mercator</td>
<td>SMajor, SMinor, Factor, CentMer, OriginLat, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 Lambert Azimuthal</td>
<td>Sphere, CentLon, CenterLat, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 Hotin Oblique Merc A</td>
<td>SMajor, SMinor, Factor, OriginLat, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 Hotin Oblique Merc B</td>
<td>SMajor, SMinor, Factor, AziAng, AzmthPt, OriginLat, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22 Space Oblique Merc A</td>
<td>SMajor, SMinor, IncAng, AscLong, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22 Space Oblique Merc B</td>
<td>SMajor, SMinor, Satnum, Path, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24 Interrupted Goode</td>
<td>Sphere</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99 Integerized Sinusoidal</td>
<td>Sphere, CentMer, FE, FN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code &amp; Projection Id</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Geographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 U T M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Lambert Conformal C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Polar Stereographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 Polyconic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 Transverse Mercator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 Lambert Azimuthal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 Hotin Oblique Merc A</td>
<td>Long1</td>
<td>Lat1</td>
<td>Long2</td>
<td>Lat2</td>
<td>zero</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 Hotin Oblique Merc B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>one</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22 Space Oblique Merc A</td>
<td>PSRev</td>
<td>Srat</td>
<td>PFlag</td>
<td></td>
<td>zero</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22 Space Oblique Merc B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>one</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24 Interrupted Goode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99 Integerized Sinusoidal</td>
<td>NZone</td>
<td></td>
<td>RFlag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where,

Lon/Z: Longitude of any point in the UTM zone or zero. If zero, a zone code must be specified.

Lat/Z: Latitude of any point in the UTM zone or zero. If zero, a zone code must be specified.

Smajor: Semi-major axis of ellipsoid. If zero, Clarke 1866 in meters is assumed.

Sminor: Eccentricity squared of the ellipsoid if less than zero, if zero, a spherical form is assumed, or if greater than zero, the semi-minor axis of ellipsoid.

Sphere: Radius of reference sphere. If zero, 6370997 meters is used.

STDPR1: Latitude of the first standard parallel

STDPR2: Latitude of the second standard parallel

CentMer: Longitude of the central meridian

OriginLat: Latitude of the projection origin

FE: False easting in the same units as the semi-major axis

FN: False northing in the same units as the semi-major axis

TrueScale: Latitude of true scale
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LongPol</td>
<td>Longitude down below pole of map</td>
</tr>
<tr>
<td>Factor</td>
<td>Scale factor at central meridian (Transverse Mercator) or center of projection (Hotine Oblique Mercator)</td>
</tr>
<tr>
<td>CentLon</td>
<td>Longitude of center of projection</td>
</tr>
<tr>
<td>CenterLat</td>
<td>Latitude of center of projection</td>
</tr>
<tr>
<td>Long1</td>
<td>Longitude of first point on center line (Hotine Oblique Mercator, format A)</td>
</tr>
<tr>
<td>Long2</td>
<td>Longitude of second point on center line (Hotine Oblique Mercator, frmt A)</td>
</tr>
<tr>
<td>Lat1</td>
<td>Latitude of first point on center line (Hotine Oblique Mercator, format A)</td>
</tr>
<tr>
<td>Lat2</td>
<td>Latitude of second point on center line (Hotine Oblique Mercator, format A)</td>
</tr>
<tr>
<td>AziAng</td>
<td>Azimuth angle east of north of center line (Hotine Oblique Mercator, frmt B)</td>
</tr>
<tr>
<td>AzmthPt</td>
<td>Longitude of point on central meridian where azimuth occurs (Hotine Oblique Mercator, format B)</td>
</tr>
<tr>
<td>IncAng</td>
<td>Inclination of orbit at ascending node, counter-clockwise from equator (SOM, format A)</td>
</tr>
<tr>
<td>AscLong</td>
<td>Longitude of ascending orbit at equator (SOM, format A)</td>
</tr>
<tr>
<td>PSRev</td>
<td>Period of satellite revolution in minutes (SOM, format A)</td>
</tr>
<tr>
<td>SRat</td>
<td>Satellite ratio to specify the start and end point of x,y values on earth surface (SOM, format A -- for Landsat use 0.5201613)</td>
</tr>
<tr>
<td>PFlag</td>
<td>End of path flag for Landsat: 0 = start of path, 1 = end of path (SOM, frmt A)</td>
</tr>
<tr>
<td>Satnum</td>
<td>Landsat Satellite Number (SOM, format B)</td>
</tr>
<tr>
<td>Path</td>
<td>Landsat Path Number (Use WRS-1 for Landsat 1, 2 and 3 and WRS-2 for Landsat 4 and 5.)</td>
</tr>
<tr>
<td>Nzone</td>
<td>Number of equally spaced latitudinal zones (rows); must be two or larger and even</td>
</tr>
<tr>
<td>Rflag</td>
<td>Right justify columns flag is used to indicate what to do in zones with an odd number of columns. If it has a value of 0 or 1, it indicates the extra column is on the right (zero) left (one) of the projection Y-axis. If the flag is set to 2 (two), the number of columns are calculated so there are always an even number of columns in each zone.</td>
</tr>
</tbody>
</table>

**Notes:**

- Array elements 14 and 15 are set to zero.
- All array elements with blank fields are set to zero.

All angles (latitudes, longitudes, azimuths, etc.) are entered in packed degrees/ minutes/ seconds (DDDMMMMSSS.SS) format.
The following notes apply to the Space Oblique Mercator A projection:

- A portion of Landsat rows 1 and 2 may also be seen as parts of rows 246 or 247. To place these locations at rows 246 or 247, set the end of path flag (parameter 11) to 1—end of path. This flag defaults to zero.

- When Landsat-1,2,3 orbits are being used, use the following values for the specified parameters:
  - Parameter 4 099005031.2
  - Parameter 5 128.87 degrees - (360/251 * path number) in packed DMS format
  - Parameter 9 103.2669323
  - Parameter 10 0.5201613

- When Landsat-4,5 orbits are being used, use the following values for the specified parameters:
  - Parameter 4 098012000.0
  - Parameter 5 129.30 degrees - (360/233 * path number) in packed DMS format
  - Parameter 9 98.884119
  - Parameter 10 0.5201613
6. Examples of HDF-EOS Library Usage

This Section contains code examples of usage of the HDF-EOS Library specified in Volume 2 of this document. These examples assume that the user is not using the SDP Toolkit and is writing applications for use outside of ECS. Examples of SDP Toolkit usage in conjunction with HDF-EOS is presented in Section 7.

Note: The examples in this document are code fragments, designed to show users how to create HDF-EOS data structures. Some of the examples in this version have not yet undergone thorough inspections and checks for ECS software standard compliance.

6.1 Swath Examples

This section contains several examples of the use of the Swath interface from both C and FORTRAN programs. First, there are simple examples in C and FORTRAN which demonstrate the use of most of the functions in the Swath interface. The section concludes with a real world example from the ECS “V0 Data Migration” effort, written in C.

6.1.1 Creating a Simple Swath

The following C and FORTRAN programs each create, define, and write a simple Swath data set to an HDF-EOS file using the HDF-EOS Swath interface.

6.1.1.1 A C Example of a Simple Swath Creation

The following C source code is part of a test driver that demonstrates the functioning of the Swath interface routines.

Note: This is a test example; it is presented for the purpose of providing a guide for users of HDF-EOS.

```c
/* ----------------------------------------------------- */
/* Simple driver to create and write data */
/* to the HDF-EOS swath file */
/* ----------------------------------------------------- */

#include "HdfEosDef.h"

main()
{
  herr_t status = -1;
  int32_t index1 = 0;
  int32_t index2 = 0;
  hid_t swfid = -1;
  hid_t SWid_simple = -1;
```
hssize_t start[2] = {10, 10};
hssize_t tstart[2] = {0, 0};
hssize_t geostart[2] = {0, 0};

hsize_t stride[2] = {1, 1};
hsize_t edge[2] = {90, 30};
hsize_t tedge[2] = {100, 40};
hsize_t geostride[2] = {1,1};
hsize_t geocount[2] = {50,40};

float32 temp[100][40];
float32 cond[100][40];
float32 cnt = -799.;

float  lat[50][40];
float  latcnt = 39.8;
float  lon[50][40];
float  loncnt = 78.0;

/* --------------------------------------------------- */
/* populate data arrays for the "Temperature"         */
/* and "Conduction" data fields                      */
/* --------------------------------------------------- */
while(index1 < 100) {
    while(index2 < 40) {
        temp[index1][index2] = cnt;
        cond[index1][index2] = cnt + .1;
        index2++;
        cnt = cnt + .1;
    }
    index1++;
    index2 = 0;
}

/* --------------------------------------------------- */
/* populate data arrays for the "Latitude" and        */
/* "Longitude" geolocation data fields                */
/* --------------------------------------------------- */
index1 = 0;
index2 = 0;

while(index1 < 50) {
    while(index2 < 40) {
        lat[index1][index2] = latcnt;
        lon[index1][index2] = loncnt;
        loncnt = loncnt - .1;
        index2++;
    }
    latcnt = latcnt - .1;
    loncnt = 78.0;
    index1++;
    index2 = 0;
}
/* Open HDF-EOS file "Swath.h5" 5 (if the named file does not exist, it will be automatically created with the read/write access mode) */
swfid = SWopen("Swath.h5", H5F_ACC_TRUNC);

/* Create swath "Simple" within HDF-EOS file "Swath.h5" */
SWid_simple = SWcreate(swfid, "Simple");

/* Define dimensions */
status = SWdefdim(SWid_simple, "DataTrack", 100);
status = SWdefdim(SWid_simple, "DataXtrack", 40);
status = SWdefdim(SWid_simple, "GeoTrack", 50);
status = SWdefdim(SWid_simple, "GeoXtrack", 40);

/* Define data fields */
status = SWdefdatafield(SWid_simple, "Temperature", "DataTrack,DataXtrack", H5T_NATIVE_FLOAT, HDFE_AUTOMERGE);
status = SWdefdatafield(SWid_simple, "Conduction", "DataTrack,DataXtrack", H5T_NATIVE_FLOAT, HDFE_NOMERGE);

/* Define geolocation fields */
status = SWdefgeofield(SWid_simple, "Latitude", "GeoTrack,GeoXtrack", NULL, H5T_NATIVE_FLOAT, HDFE_AUTOMERGE);
status = SWdefgeofield(SWid_simple, "Longitude", "GeoTrack,GeoXtrack", "GeoTrack,GeoXtrack", H5T_NATIVE_FLOAT, HDFE_AUTOMERGE);

/* Define mapping between different dimensions */
status = SWdefdimmap(SWid_simple, "GeoTrack", "DataTrack", 0, 2);
status = SWdefdimmap(SWid_simple, "GeoXtrack", "DataXtrack", 0, 1);

/* Detach from the swath */
status = SWdetach(SWid_simple);

/* Re-attach to the swath */
SWid_simple = SWattach(swfid, "Simple");

/* Write the temperature data to the swath */
6.1.1.2 FORTRAN Example of a Simple Swath Creation

c ----------------------------------------------------
c Simple driver to create and write data
to the HDF-EOS swath file
c ----------------------------------------------------

implicit none
integer status
integer i
integer index1
integer index2

integer swfid
integer SWid_simple

integer*4 start(2)
integer*4 tstart(2)
integer*4 geostart(2)

integer*4 stride(2)
integer*4 tstride(2)
integer*4 edge(2)
integer*4 tedge(2)
integer*4 geostride(2)
integer*4 geocount(2)
real*8 TimeData(5)
real*4  temp(100,40)
real*4  cond(100,40)
real*4  cnt
real*4  lat(50,40)
real*4  latcnt
real*4  lon(50,40)
real*4  loncnt

integer  swclose, swopen, swcreate, swdefdim
integer  swdefdfld, swdefgfld, swdefmap, swattach, swdetach
integer  swwrfld

integer  H5F_ACC_TRUNC
parameter (H5F_ACC_TRUNC = 2)
integer  HDFE_AUTOMERGE
parameter (HDFE_AUTOMERGE = 1)
integer  HDFE_NOMERGE
parameter (HDFE_NOMERGE = 0)
integer  H5T_NATIVE_INT
parameter (H5T_NATIVE_INT = 0)
integer  H5T_NATIVE_FLOAT
parameter (H5T_NATIVE_FLOAT = 1)
integer  H5T_NATIVE_DOUBLE
parameter (H5T_NATIVE_DOUBLE = 2)

status  = -1
swfid  = -1
SWid_simple = -1
start(1)  = 10
start(2)  = 10
tstart(1)  = 0
tstart(2)  = 0
geostart(1)  = 0
geostart(2)  = 0
stride(1)  = 1
stride(2)  = 1
edge(1)  = 90
edge(2)  = 30
tedge(1)  = 100
tedge(2)  = 40
geostride(1)  = 1
geostride(2)  = 1
geocount(1)  = 50
geocount(2)  = 40
cnt  = -799.
latcnt  = 39.8
loncnt  = 78.0

c --------- -----------------------------------------------
c  populate data arrays for the "Temperature"
c  and "Conduction" data fields

do 20 index1 =1, 100
do 10 index2 =1, 40
   temp(index1,index2) = cnt
   cond(index1,index2) = cnt + .1
   cnt = cnt + .1
continue
continue

-- populate data arrays for the "Latitude" and "Longitude" geolocation data fields

do 40 index1 = 1, 50
  do 30 index2 = 1, 40
    lat(index1, index2) = latcnt
    lon(index1, index2) = loncnt
    loncnt = loncnt - .1
  30 continue
  latcnt = latcnt - .1
  loncnt = 78.0
40 continue

-- Open HDF-EOS file "Swath.h5" (if the named file does not exist, it will be automatically created with the read/write access mode)

  swfid = SWopen("Swath.h5", H5F_ACC_TRUNC)

-- Create swath "Simple" within HDF-EOS file

  SWid_simple = SWcreate(swfid, "Simple")

-- Define dimensions

  status = SWdefdim(SWid_simple, "DataTrack", 100)
  status = SWdefdim(SWid_simple, "DataXtrack", 40)
  status = SWdefdim(SWid_simple, "GeoTrack", 50)
  status = SWdefdim(SWid_simple, "GeoXtrack", 40)

-- Define data fields

  status = SWdefdfld(SWid_simple, "Temperature", "DataTrack,DataXtrack", "", H5T_NATIVE_FLOAT, HDFE_AUTOMERGE)
  status = SWdefdfld(SWid_simple, "Conduction", "DataTrack,DataXtrack", "", H5T_NATIVE_FLOAT, HDFE_NOMERGE)

-- Define geolocation fields

  status = SWdefgfld(SWid_simple, "Latitude", "GeoTrack,GeoXtrack", "", H5T_NATIVE_FLOAT, HDFE_AUTOMERGE)
  status = SWdefgfld(SWid_simple, "Longitude", "GeoTrack,GeoXtrack", "", H5T_NATIVE_FLOAT, HDFE_AUTOMERGE)
> Status = SWdefgfld(SWid_simple,"Time","GeoTrack","",
> H5T_NATIVE_DOUBLE,HDFE_AUTOMERGE)

c Define mapping between different dimensions

status = SWdefmap(SWid_simple, "GeoTrack", "DataTrack", 0, 2)
status = SWdefmap(SWid_simple, "GeoXtrack", "DataXtrack", 0, 1)

c Detach from the swath

status = SWdetach(SWid_simple)

c Re-attach to the swath

SWid_simple = SWattach(swfid, "Simple")

c Write the temperature data to the swath

status = SWwrfld(SWid_simple, "Temperature", tstart, stride,
> tedge, temp)

c Write the conduction data to the swath

status = SWwrfld(SWid_simple, "Conduction", start, stride,
> edge, cond)

c Write the Latitude data to the swath

status = SWwrfld(SWid_simple, "Latitude", geostart, geostride
> geocount, lat)

c Write the Longitude data to the swath

status = SWwrfld(SWid_simple, "Longitude", geostart,
> geostride, geocount, lon)

--- Writing Time in a swath ---

      do 120 i=1,5
        TimeData( i ) = 5.e7 + 5.e6 *i
    120 continue

tstart( 1 ) = 0
tstride(1)  = 1
tedge( 1 )  = 5

Status = SWwrfld(SWid_simple,"Time", tstart, tstride, tedge,
>   TimeData)
Close the HDF-EOS file "Swath.h5"

```c
status = SWclose(swfid)
stop
end
```

### 6.1.2 Performing Subsetting for a Swath

The following C and FORTRAN programs each demonstrate how to perform subsetting for a Swath object in an HDF-EOS file using the HDF-EOS Swath interface.

#### 6.1.2.1 The C Examples of a Subsetting for a Swath

**Example 1:**

```c
/* *-----------------------------------------------------------------------
 */
/* Simple driver to define time period for a swath, and then extract */
/* the entries of a specified data field within the defined time period. */
/* NOTE: Before running this driver make sure that the */
/* file,Swathc_Test.h5,*/
/* produced by a test driver testswath_he3.c exists. */
/* *-----------------------------------------------------------------------
 */

#include "HdfEosDef.h"

main(int argc, char *argv[])
{
   herr_t status;

   hid_t swfidc_test;
   hid_t SWid_co;

   int32_t i, j, k;
   int32_t periodID;
   int32_t rank, size;

   hsize_t dims[8];
   H5T_class_t *numtype = (H5T_class_t *)NULL;

   float64 *timebuf;
   float64 startime, stoptime;
```
/* ----------------------------------------------------------- */
/* Open HDF-EOS swath file and attach to the named swath */
/* ----------------------------------------------------------- */

swfidc_test = SWopen("Swathc_Test.h5", H5F_ACC_RDONLY);
printf("\t\tSwath file ID returned by SWopen %d\n",swfidc_test);

SWid_co = SWattach(swfidc_test, "Swathco");
printf("\t\tSwath ID returned by SWattach on swath Swathco %d\n",SWid_co);

/* ----------------------------------------------------------- */
/* Define time period */
/* ----------------------------------------------------------- */

starttime = 46353450.0;
stoptime = 46500000.0;

periodID = SWdeftimeperiod(SWid_co,starttime,stoptime,HDFE_MIDPOINT);
printf("\t\tPeriod id returned by SWdeftimeperiod %d\n",periodID);

/* ----------------------------------------------------------- */
/* Retrieve basic information about data fields "BandC" */
/* and "Time" */
/* ----------------------------------------------------------- */

numtype = (H5T_class_t *)calloc(1, sizeof(H5T_class_t));
status = SWperiodinfo(SWid_co,periodID,"BandC",numtype,&rank,dims,&size);
printf("\t\tStatus returned by SWperiodinfo %d\n",status);
printf("\t\tNumber type of region %d, Rank of region %d
",*numtype, rank);
printf("\t\tDimensions of region %lu %lu \n", (unsigned long)dims[0], (unsigned long)dims[1]);
printf("\t\tSize of region in bytes %d\n",size);

status = SWperiodinfo(SWid_co,periodID,"Time",numtype,&rank,dims,&size);
printf("\t\tStatus returned by SWperiodinfo %d\n",status);
printf("\t\tNumber type of region %d, Rank of region %d
",*numtype,rank);
printf("\t\tDimensions of region %lu %lu \n", (unsigned long)dims[0], (unsigned long)dims[1]);
printf("\t\tSize of region in bytes %d\n",size);

free(numtype);

timebuf = (float64 *)malloc(size);
status = SWextractperiod(SWid_co,periodID,"Time",HDFE_INTERNAL,timebuf);
printf("\t\tStatus returned by SWextractperiod %d\n",status);
k = 0;
for(i = 0; i < 10; i++)
{
    for(j = 0; j < 5; j++)
    {
        printf("\t\t%d %d %f\n", i, j, timebuf[k]);
        k++;
    }
}

free(timebuf);
status = SWdetach(SWid_co);
printf("\t\tStatus returned by SWdetach %d\n",status);

status=SWclose(swfidc_test);
}

Example 2 :

#include "HdfEosDef.h"

main()
{

herr_t  status;
hid_t  swfidc_test, SWid_simple, SWid_oned, SWid_index;
int32_t  i, j, k, rank, size, regionID;
hsize_t  dims[8];
H5T_class_t *numtype = (H5T_class_t *)NULL;
float32  *datbuf;
float64 *timebuf;

float64  corlon[2], corlat[2], range[2];

/* Open swath file "Swathc_Test.h5" and attach to the named swath objects.*/

swfidc_test = SWopen("Swathc_Test.h5", H5F_ACC_RDONLY);
printf("\t\tSwath file ID returned by SWopen %d\n",swfidc_test);

SWid_oned = SWattach(swfidc_test, "OnedGeo");
printf("\t\tSwath ID returned by SWattach on swath "OnedGeo" %d\n",SWid_oned);

SWid_simple = SWattach(swfidc_test, "Simple");
printf("\t\tSwath ID returned by SWattach on swath "Simple" %d\n",SWid_simple);

SWid_index = SWattach(swfidc_test, "Index");
printf("\t\tSwath ID returned by SWattach on swath "Index" %d\n",SWid_index);
/* 1.1 Perform subsetting on the named field */

range[0] = 46353450.0;
range[1] = 46500000.0;
regionID = SWdefvrtregion(SWid_oned, HDFE_NOPREVSUB, "Time", range);
printf("Region id returned by SWdefvrtregion \%d\n", regionID);

/* 1.2 Retrieve information about the subsetted region */

for the named data field

numtype = (H5T_class_t *)calloc(1, sizeof(H5T_class_t));
status = SWregioninfo(SWid_oned, regionID, "Time", numtype, &rank, dims, &size);
printf("Status returned by SWregioninfo \%d\n", status);
printf("Number type of field \%d\n", *numtype);
printf("Rank of field \%d\n", rank);
printf("Dimension of subsetted region: %lu \n", (unsigned long)dims[0]);
printf("Size in bytes of region \%d\n", size);
free(numtype);

/* 1.3 Read data into the data buffer from the subsetted region */

timebuf = (float64 *)malloc(size);
status = SWextractregion(SWid_oned, regionID, "Time", HDFE_INTERNAL, timebuf);
printf("Status returned by SWextractregion \%d\n", status);
for(k = 0; k < 5; k++)
{
    printf("\t%d %f \n", k, timebuf[k]);
}
free(timebuf);
status = SWdetach(SWid_oned);
printf("Status returned by SWdetach \%d\n", status);
2.1 Define a longitude-latitude box region for the named swath

```c

corlon[0] = 75.5;
corlat[0] = 35.5;
corlon[1] = 77.5;
corlat[1] = 36.5;

regionID = SWdefboxregion(SWid_simple, corlon, corlat, HDFE_MIDPOINT);
printf("Swath region ID returned by SWdefboxregion %d\n", regionID);
```

2.2 Retrieve information about the subsetted region for the named data field

```c

numtype = (H5T_class_t *)calloc(1, sizeof(H5T_class_t));
status = SWregioninfo(SWid_simple, regionID, "Temperature", numtype, &rank, dims, &size);
printf("Status returned by SWregioninfo %d\n", status);
printf("Number type of field %d\n", *numtype);
printf("Rank of field %d\n", rank);
printf("Dimensions of subsetted region: %lu, %lu\n", (unsigned long)dims[0], (unsigned long)dims[1]);
printf("Size in bytes of region %d\n", size);
free(numtype);
```

2.3 Read data into the data buffer from the subsetted region

```c

datbuf = (float32 *)malloc(size);
status = SWextractregion(SWid_simple, regionID, "Temperature", HDFE_INTERNAL, datbuf);
printf("Status returned by SWextractregion %d\n", status);
k = 0;
for (i = 0; i < 10; i++)
{
    for (j = 0; j < 5; j++)
    {
        printf("%d %d %f\n", i, j, datbuf[k]);
k++;
    }
}
free(datbuf);
```
2.4 Retrieve information about the subsetted region for the named data field

```
numtype = (H5T_class_t *)calloc(1, sizeof(H5T_class_t));
status = SWregioninfo(SWid_simple, regionID, "Latitude", numtype, &rank, 
dims, &size);
printf("\t\tStatus returned by SWregioninfo %d\n", status);
printf("\t\tNumber type of field %d\n", *numtype);
printf("\t\tRank of field %d\n", rank);
printf("\t\tDimensions of subsetted region: %lu, %lu \n", (unsigned long)dims[0], (unsigned long)dims[1]);
printf("\t\tSize in bytes of region %d\n", size);
free(numtype);
```

2.5 Read data into the data buffer from the subsetted region

```
datbuf = (float32 *)malloc(size);
status = SWextractregion(SWid_simple, regionID, "Latitude", HDFE_INTERNAL,
datbuf);
printf("\t\tStatus returned by SWextractregion %d\n", status);
k = 0;
for(i = 0; i < 10; i++)
{
    for(j = 0; j < 5; j++)
    {
        printf("\t\t%d %d %f\n", i, j, datbuf[k]);
k++;
    }
}
free(datbuf);

status = SWdetach(SWid_simple);
printf("\t\tValue returned by SWdetach %d\n", status);
```

3.1 Define a longitude-latitude box region for the named swath

```
corlat[0] = 37.0;
corlat[1] = 38.5;
corlon[0] = 76.0;
corlon[1] = 77.5;
```
regionID = SWdefboxregion(SWid_index, corlon, corlat, HDFE_MIDPOINT);
printf("%d Region ID returned by SWdefboxregion \n", regionID);
/* 3.2 Retrieve information about the subsetted region
   for the named data field
   -----------------------------------------------
   numtype = (H5T_class_t *)calloc(1, sizeof(H5T_class_t));
   status = SWregioninfo(SWid_index, regionID, "Fakedata", numtype, &rank,
dims, &size);
   printf("%d Status returned by SWregioninfo \n", status);
   printf("%d Number type of field \n", *numtype);
   printf("%d Rank of field \n", rank);
   printf("%d Dimensions of subsetted region: %lu , %lu\n", (unsigned
long)dims[0], (unsigned long)dims[1]);
   printf("%d Size in bytes of region \n", size);
   */

/* 3.3 Read data into the data buffer from the subsetted region
   -----------------------------------------------
   datbuf = (float32 *)malloc(size);
   status = SWextractregion(SWid_index, regionID, "Fakedata", HDFE_INTERNAL,
datbuf);
   printf("%d Status returned by SWextractregion \n", status);
   k = 0;
   for(i = 0; i < 10; i++)
   {
     printf("%d %f\n", i, datbuf[k]);
     k++;
   }
   free(datbuf);
   free(numtype);
   */

status = SWdetach(SWid_index);
status = SWclose(swfidc_test);
}

6.1.2.2 The FORTRAN Examples of a Subsetting for a Swath

Example 1:

C Simple driver to define time period for a swath, and then extract all the
C entries of a specified data field within the defined time period.
NOTE: Before running this driver make sure that the file, Swathc_Test.h5, produced by a test driver testswath_he3.c exists.

```c
implicit none
integer status
integer swfidc_test
integer SWid_co
integer*4 k
integer*4 periodID
integer*4 rank, size
integer*4 dims(8)
integer numtype(1)
real*8 timebuf(10)
real*8 starttime, stoptime
integer swclose, swdetach, swextper, swperinfo
integer swopen, swattach, SWdeftmeper
integer H5F_ACC_RDONLY
parameter ( H5F_ACC_RDONLY = 0)
integer*4 HDFE_MIDPOINT
parameter ( HDFE_MIDPOINT = 0)
integer*4 HDFE_INTERNAL
parameter ( HDFE_INTERNAL = 0)
```

Open HDF-EOS swath file and attach to the named swath

```c
swfidc_test = SWopen("Swath.h5", H5F_ACC_RDONLY)
print *, ' Swath file ID returned by SWopen = ', swfidc_test
SWid_co = SWattach(swfidc_test, "Simple")
print *, ' Swath ID returned by SWattach on swath Swathco = '
> ', SWid_co
```

Define time period

```c
starttime = 5.5e7
stoptime = 6.5e7
```

```c
periodID = SWdeftmeper(SWid_co, starttime, stoptime, HDFE_MIDPOINT)
print *, ' Period id returned by SWdeftmeperiod = ', periodID
```

Retrieve basic information about data fields "BandC" and "Time"

```c
status = SWperinfo(SWid_co, periodID, "Time", numtype, rank, dims
```

6-15 162-WP-004-001
Example 2:

C -------
C Simple driver demonstrating how to perform subsetting along the "Track" C dimension and along any other dimension defined within a specified swath. C NOTE: make sure that the HDF-EOS file Swathc_Test.h5 produced by the test C driver testswath_he3.c exists.
C -------

Implicit none

Integer status

Integer swfidc_test, SWid_simple, SWid_oned, SWid_index
Integer*4 rank, size, regionID
Integer i,j,k
Integer*4 dims(8)
Integer numtype
Real*4 datbuf(48000)
Real*8 timebuf(48000)
Real*8 corlon(2), corlat(2), range(2)

SWopen, swattach, swdefvrtreg, swreginfo, swextreg
integer swdetach, swclose
integer*4 swdefboxreg

integer H5F_ACC_RDONLY
parameter ( H5F_ACC_RDONLY = 0)
integer*4 HDFE_MIDPOINT
parameter ( HDFE_MIDPOINT = 0)
integer*4 HDFE_INTERNAL
parameter ( HDFE_INTERNAL = 0)
integer HDFE_NOPREVSUB
parameter ( HDFE_NOPREVSUB = -1)

c Open swath file "Swathc_Test.h5" and attach to the named swath objects.

swfidc_test = SWopen("Swathc_Test.h5", H5F_ACC_RDONLY)
print *, ' Swath file ID returned by SWopen = ', swfidc_test

SWid_oned = SWattach(swfidc_test, "OnedGeo")
print *, ' Swath ID returned by SWattach on swath "OnedGeo" = ', SWid_oned

SWid_simple = SWattach(swfidc_test, "Simple")
print *, ' Swath ID returned by SWattach on swath "Simple" = ', SWid_simple

SWid_index = SWattach(swfidc_test, "Index")
print *, ' Swath ID returned by SWattach on swath "Index" = ', SWid_index

range(1) = 46353450.0
range(2) = 46500000.0
regionID = SWdefvrtreg(SWid_oned, HDFE_NOPREVSUB, "Time", range)
print *, ' Region id returned by SWdefvrtregion = ', regionID

status = SWregioninfo(SWid_oned, regionID, "Time", numtype, rank, dims, size)
print *, ' Status returned by SWregioninfo = ', status
print *, ' Number type of field = ', numtype
print *, ' Rank of field = ', rank
print *, ' Dimension of subsetted region: ', dims(1)
print *, ' Size in bytes of region = ', size
1.3 Read data into the data buffer from the subsetted region

```
status = SWextreg(SWid_oned, regionID, "Time", HDFE_INTERNAL > ,timebuf)
print *, ' Status returned by SWextractregion = ', status
do 10 k = 1, 5
   print *, k out, k, timebuf(',k') = ',timebuf(k)
10 continue
status = SWdetach(SWid_oned)
print *, ' Status returned by SWdetach = ',status
```

2.1 Define a longitude-latitude box region for the named swath

```
corlon(1) = 75.5
corlat(1) = 35.5
corlon(2) = 77.5
corlat(2) = 36.5
regionID = SWdefboxreg(SWid_simple,corlon,corlat,HDFE_MIDPOINT)
print *, ' Swath region ID returned by SWdefboxregion = ', > regionID
```

2.2 Retrieve information about the subsetted region for the named data field

```
status = SWreginfo(SWid_simple, regionID,"Temperature",numtype > ,rank,dims,size)
print *, ' Status returned by SWregioninfo = ', status
print *, ' Number type of field = ',numtype
print *, ' Rank of field = ',rank
print *, ' Dimensions of subsetted region: ', dims(1), dims(2)
print *, ' Size in bytes of region = ',size
```

2.3 Read data into the data buffer from the subsetted region

```
status = SWextreg(SWid_simple, regionID, "Temperature" > ,HDFE_INTERNAL,datbuf)
print *, ' Status returned by SWextractregion = ', status
k = 1
do 30 i = 1,10
   do 20 j = 1, 5
      print *, i, j, ' datbuf(',k') = ',datbuf(k)
   k=k+1
20 continue
30 continue
```
2.4 Retrieve information about the subsetted region for the named data field

```fortran
status = SWreginfo(SWid_simple, regionID, "Latitude", numtype, rank, dims, size)
print *, ' Status returned by SWregioninfo = ', status
print *, ' Number type of field = ', numtype
print *, ' Rank of field = ', rank
print *, ' Dimensions of subsetted region: ', dims(1), dims(2)
print *, ' Size in bytes of region = ', size
```

2.5 Read data into the data buffer from the subsetted region

```fortran
status = SWextreg(SWid_simple, regionID, "Latitude", HDFE_INTERNAL,...
                 datbuf)
print *, ' Status returned by SWextractregion = ', status
k = 1
do 50 i = 1, 10
   do 40 j = 1, 5
      print *, i, j, datbuf(k)
      k = k + 1
 40 continue
50 continue

status = SWdetach(SWid_simple)
print *, ' Value returned by SWdetach = ', status
```

3.1 Define a longitude-latitude box region for the named swath

```fortran
corlat(1) = 37.0
corlat(2) = 38.5
corlon(1) = 76.0
corlon(2) = 77.5

regionID = SWdefboxreg(SWid_index, corlon, corlat, HDFE_MIDPOINT)
print *, ' Region ID returned by SWdefboxregion = ', regionID
```

3.2 Retrieve information about the subsetted region for the named data field

```fortran
status = SWreginfo(SWid_index, regionID, "Fakedata", numtype, rank, dims, size)
print *, ' Status returned by SWregioninfo = ', status
```
print *, ' Number type of field = ', numtype
print *, ' Rank of field = ', rank
print *, ' Dimensions of subsetted region: ', dims(1), dims(2)
print *, ' Size in bytes of region = ', size

c -----------------------------------------------------------------------------
c 3.3 Read data into the data buffer from the subsetted region
c -----------------------------------------------------------------------------

  status = SWextreg(SWid_index, regionID, "Fakedata", HDFE_INTERNAL,>
             datbuf)
p   print *, ' Status returned by SWextractregion = ', status
  do 60 k = 1,10
    print *, ' k=',k,' datbuf(',k,')= ',datbuf(k)
  60 continue

  status = SWdetach(SWid_index)
  status = SWclose(swfidC_test)

stop
end

6.2 Grid Examples

This section contains several examples of the use of the Grid interface from both C and
FORTRAN programs. First, there are simple examples in C and FORTRAN which demonstrate
the use of most of the functions in the Grid interface.

6.2.1 Creating a Simple Grid

The following C and FORTRAN programs each create, define, and write a simple Grid data set
to an HDF-EOS file using the HDF-EOS Grid interface.

6.2.1.1 A C Example of a Simple Grid Creation

    /* ------------------------------------------------------------------ */
    /* Simple driver to create and write data to the HDF-EOS grid file */
    /* ------------------------------------------------------------------ */

    #include "HdfEosDef.h"

    main()
    {
      herr_t status = -1;
      hid_t gdfid, GDid_utm, GDid_igoode, GDid_polar_np, GDid_polar_sp;
      hid_t GDid_geo, GDid_som, GDid_lamaz, GDid_hom;
      hid_t GDid_lamcon, GDid_tm, GDid_poly, GDid_is;

int32_t index1 = 0, index2 = 0;
float32 utmray[60][80], ray2[155][40], ray3[25][80];
float32 psray[360][90], gooderay[120][60];
float32 utmhght[80], georay[60][80];
float32 gooderay2[45][60], psray2[75][90];
float32 lamazray2[65][180], georay2[85][80];
float32 utmcnt = -799.0, ray3cnt = -19.5;
float32 pscnt = -134.5, goodecnt = 27.4;
float32 hghtinit = 323.0, lamazcnt = -299.5, geocnt = 2001.0;
float attr2[5] = {5.1,17.2,28.3,39.4,57.5};
int32 spherecode, xdim, ydim, ray[8];
hsize_t start[2] = {10, 10};
hsize_t stride[2] = {1, 1};
hsize_t edge[2] = {10, 10};
hsize_t intattr[1] = {4};
hsize_t floatattr[1] = {5};
hsize_t *attrcount = (hsize_t *)NULL;
float64 projparm[16], uplft[2], lowrgt[2];
float64 lonval[5], latval[5];
float64 *datbuf = (float64 *)NULL;
float64 utmtmeray[4800], tmeinit = 35232487.2;

/* --------------------------------------------------- */
/* Populate data arrays */
/* --------------------------------------------------- */
while(index1 < 60) {
    while(index2 < 80) {
        utmray[index1][index2] = utmcnt;
        georay[index1][index2] = geocnt;
        index2++;
        utmcnt = utmcnt + 0.4;
        geocnt = geocnt + 0.2;
    }
    index1++;
    index2 = 0;
}

index1=0;
while(index1 < 4800) {
    utmtmeray[index1] = tmeinit;
    index1++;
    tmeinit = tmeinit + 70.7;
}
index1=0; while(index1 < 80) {
    utmhght[index1] = hghtinit;
    hghtinit = hghtinit + 2.75;
    index1++;
}

index1=0; index2=0; while(index1 < 85) {
    while(index2 < 80) {
        georay2[index1][index2] = geocnt;
        index2++;
        geocnt = geocnt - .9;
    }
    index1++;
    index2 = 0;
}

index1=0; index2=0; while(index1 < 25) {
    while(index2 < 90) {
        ray3[index1][index2] = ray3cnt;
        index2++;
        ray3cnt=ray3cnt + .1;
    }
    index1++;
    index2 = 0;
}

index1=0; index2=0; while(index1 < 360) {
    while(index2 < 90) {
        psray[index1][index2] = pscnt;
        index2++;
        pscnt = pscnt + .4;
    }
    index1++;
    index2 = 0;
}

index1=0; index2=0; while(index1 < 75) {
    while(index2 < 90) {
        psray2[index1][index2] = pscnt;
        index2++;
        pscnt = pscnt - .4;
    }
    index1++;
    index2 = 0;
}

index1=0; index2=0;
while(index1 < 120) {
    while(index2 < 60) {
        gooderay[index1][index2] = goodecnt;
        index2++;
        goodecnt = goodecnt + .4;
    }
    index1++;
    index2 = 0;
}

index1=0;
index2=0;
while(index1 < 45) {
    while(index2 < 60) {
        gooderay2[index1][index2] = goodecnt;
        index2++;
        goodecnt = goodecnt - .4;
    }
    index1++;
    index2 = 0;
}

index1=0;
index2=0;
while(index1 < 65) {
    while(index2 < 180) {
        lamazray2[index1][index2] = lamazcnt;
        index2++;
        lamazcnt = lamazcnt + .6;
    }
    index1++;
    index2 = 0;
}

index1=0;
while(index1 < 16) {
    iproparm[index1] = 0.0;
    index1++;
}

/* --------------------------------------------------- */
/* Open HDF-EOS file "Grid.h5" (if the named            */
/* file does not exist, it will be automatically      */
/* created with the read/write access mode            */
/* --------------------------------------------------- */
gdfid = GDopen("Grid.h5", H5F_ACC_TRUNC);

/* --------------------------------------------------- */
/* Create UTM grid                                      */
/* --------------------------------------------------- */
xdim = 60;
ydim = 80;
uplft[0] = -512740.28306;
uplft[1] = 2733747.62890;
lowrgt[0] = -12584.57301;
lowrgt[1] = 1946984.64021;

GDid_utm = GDcreate(gdfid, "UTM", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Geographic grid */
/* --------------------------------------------------- */
xdim = 60;
ydim = 80;
uplft[0] = -126000000.00;
uplft[1] = -064000000.00;
lowrgt[0] = -120000000.00;
lowrgt[1] = -072000000.00;

GDid_geo = GDcreate(gdfid, "Geo", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Polar Stereo grid of north hemisphere */
/* --------------------------------------------------- */
xdim = 360;
ydim = 90;
uplft[0] = -10447125.82759;
uplft[1] = 10447125.82759;
lowrgt[0] = 10447125.82759;
lowrgt[1] = -10447125.82759;

gdid_polar_np = GDcreate(gdfid, "Polar_np", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Polar Stereo grid of south hemisphere */
/* --------------------------------------------------- */
xdim = 360;
ydim = 90;
uplft[0] = 10447125.82759;
uplft[1] = -10447125.82759;
lowrgt[0] = -10447125.82759;
lowrgt[1] = 10447125.82759;

gdid_polar_sp = GDcreate(gdfid, "Polar_sp", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Interrupted Goode grid */
/* --------------------------------------------------- */
xdim = 120;
ydim = 60;
uplft[0] = -11119487.42844;
uplft[1] = 8673539.24806;
lowrgt[0] = 15567282.39984;
lowrgt[1] = -8673539.24806;

GDid_igoode = GDcreate(gdfid, "IGoode", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Space Oblique Mercator grid */
/* --------------------------------------------------- */
/* --------------------------------------------------- */
/* Create Lambert Azimuthal grid                      */
/* --------------------------------------------------- */
xdim=719;
ydim=180;
uplft[0] = 0.0000;
uplft[1] = 9009950.36324;
lowrgt[0] = 0.0000;
lowrgt[1] = -9009950.36324;
GDid_lamaz = GDcreate(gdfid, "Lamaz", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Hotin Oblique Mercator grid                 */
/* --------------------------------------------------- */
xdim=200;
ydim=90;
uplft[0] = 3422259.57265;
uplft[1] = 6824822.05796;
lowrgt[0] = -17519429.48100;
lowrgt[1] = 4994368.88166;
GDid_hom = GDcreate(gdfid, "hom", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Lambert Conformal grid                      */
/* --------------------------------------------------- */
xdim=351;
ydim=171;
uplft[0] = -2279109.37671;
uplft[1] = 12358083.24054;
lowrgt[0] = -56342817.96247;
lowrgt[1] = -24776979.34092;
GDid_lamcon = GDcreate(gdfid, "lamcon", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Transverse Mercator grid                   */
/* --------------------------------------------------- */
xdim=181;
ydim=171;
uplft[0] = 4855670.77539;
uplft[1] = 9458558.92483;
lowrgt[0] = 5201746.43983;
lowrgt[1] = -10466077.24942;
GDid_tm = GDcreate(gdfid, "tm", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create Polyconic grid */
/* --------------------------------------------------- */
xdim=161;
ydim=171;
uplft[0] = -250873.85859;
uplft[1] = 12669051.66767;
lowrgt[0] = 850873.85859;
lowrgt[1] = -7137259.12615;

GDid_poly = GDcreate(gdfid, "poly", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Create IS grid */
/* --------------------------------------------------- */
xdim=351;
ydim=171;
uplft[0] = 1436267.12618;
uplft[1] = 9451564.31420;
lowrgt[0] = 1343604.73094;
lowrgt[1] = -9451564.31420;

GDid_is = GDcreate(gdfid, "is", xdim, ydim, uplft, lowrgt);

/* --------------------------------------------------- */
/* Define projections */
/* --------------------------------------------------- */
zonecode = -13;
spherecode = 0;
for(i=0; i < 16; i++)
{
    projparm[i]=0.;
}
status = GDdefproj(GDid_utm, GCTP_UTM, zonecode, spherecode, projparm);
for(i=0; i < 16; i++)
{
    projparm[i]=0.;
}
spherecode = 0;
projparm[5] = 40000000.00;
status=GDdefproj(GDid_polar_np, GCTP_PS, NULL, spherecode, projparm);
for(i=0; i < 16; i++)
{
    projparm[i] = 0.;
}
status=GDdefproj(GDid_igoode, GCTP_GOOD, NULL, NULL, projparm);

for(i=0; i < 16; i++)
{
    projparm[i]=0.;
}

status=GDdefproj(GDid_lamaz, GCTP_LAMAZ, NULL, NULL, projparm);

for(i=0; i < 16; i++)
{
    projparm[i] = 0.;
}
projparm[2] = 3;
projparm[3] = 150;
projparm[12] = 1;

status=GDdefproj(GDid_som, GCTP_SOM, NULL, NULL, projparm);

for(i=0; i < 16; i++)
{
    projparm[i] = 0.;
}
projparm[2] = 0.9996;
projparm[5] = 20000000.00;
projparm[8] = -75000000.00;
projparm[9] = 10000000.00;
projparm[10] = -95000000.00;
projparm[11] = 30000000.00;

status=GDdefproj(GDid_geo, GCTP_GEO, NULL, NULL, NULL);

for(i=0; i < 16; i++)
{
    projparm[i] = 0.;
}

status=GDdefproj(GDid_hom, GCTP_HOM, NULL, NULL, projparm);

for(i=0; i < 16; i++)
{
    projparm[i] = 0.;
}
projparm[2] = 20000000.00;
projparm[3] = 40000000.00;
projparm[4] = -75000000.00;

status=GDdefproj(GDid_lamcon, GCTP_LAMCC, NULL, NULL, projparm);

for(i=0; i < 16; i++)
{
    projparm[i] = 0.;
}
projparm[2] = 0.9996;
projparm[4] = -75000000.00;
projparm[6] = 50000000.00;

status=GDdefproj(GDid_tm, GCTP_TM, NULL, NULL, projparm);

for(i=0; i < 16; i++)
{
    projparm[i] = 0.;
projparm[4] = 75000000.00;
projparm[5] = -25000000.00;
projparm[6] = 300000.00;

status = GDdefproj(GDid_poly, GCTP_POLYC, NULL, NULL, projparm);

for(i=0; i < 16; i++)
{
    projparm[i] = 0.;
}
projparm[4] = 0.;
projparm[5] = 40000000.00;

status = GDdefproj(GDid_is, GCTP_ISINUS, NULL, NULL, projparm);

/* --------------------------------------------------- */
/* Define pixel registration in the grids */
/* --------------------------------------------------- */
status = GDdefpixreg(GDid_utm, HDFE_CORNER);
status = GDdefpixreg(GDid_polar_np, HDFE_CORNER);
status = GDdefpixreg(GDid_igoode, HDFE_CORNER);
status = GDdefpixreg(GDid_som, HDFE_CORNER);
status = GDdefpixreg(GDid_lamaz, HDFE_CORNER);
status = GDdefpixreg(GDid_geo, HDFE_CORNER);

/* --------------------------------------------------- */
/* Define origin of projections in the grids */
/* --------------------------------------------------- */
status = GDdeforigin(GDid_utm, HDFE_GD_UL);
status = GDdeforigin(GDid_polar_np, HDFE_GD_UR);
status = GDdeforigin(GDid_igoode, HDFE_GD_LL);
status = GDdeforigin(GDid_som, HDFE_GD_LR);
status = GDdeforigin(GDid_lamaz, HDFE_GD_UR);
status = GDdeforigin(GDid_geo, HDFE_GD_UL);

/* --------------------------------------------------- */
/* Define dimensions */
/* --------------------------------------------------- */
status = GDdefdim(GDid_utm, "Conduction", 25);
status = GDdefdim(GDid_utm, "Timedim", 4800);
status = GDdefdim(GDid_utm, "Hghtdim", 80);
status = GDdefdim(GDid_polar_np, "Convection", 75);
status = GDdefdim(GDid_igoode, "Radiant", 45);
status = GDdefdim(GDid_som, "Emission", 55);
status = GDdefdim(GDid_lamaz, "Flux", 65);
status = GDdefdim(GDid_geo, "Gradient", 85);

/* --------------------------------------------------- */
/* Detach from the grids */
/* --------------------------------------------------- */
status = GDdetach(GDid_utm);
status = GDdetach(GDid_polar_np);
status = GDdetach(GDid_polar_sp);
status = GDdetach(GDid_igoode);
status = GDdetach(GDid_som);
status = GDdetach(GDid_lamaz);
status = GDdetach(GDid_geo);
status = GDdetach(GDid_hom);
status = GDdetach(GDid_lamcon);
status = GDdetach(GDid_tm);
status = GDdetach(GDid_poly);
status = GDdetach(GDid_is);

/* --------------------------------------------------- */
/* Re-attach to the grids */
/* --------------------------------------------------- */
GDid_utm = GDattach(gdfid, "UTM");
GDid_igoode = GDattach(gdfid, "IGoode");
GDid_lamaz = GDattach(gdfid, "Lamaz");
GDid_polar_np = GDattach(gdfid, "Polar_np");
GDid_som = GDattach(gdfid, "SOM");
GDid_geo = GDattach(gdfid, "Geo");

status = GDdeffield(GDid_utm, "Voltage", "XDim,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_AUTOMERGE);
status = GDdeffield(GDid_utm, "Drift", "XDim,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_AUTOMERGE);
status = GDdeffield(GDid_utm, "Time", "Timedim", NULL, H5T_NATIVE_DOUBLE, HDFE_AUTOMERGE);
status = GDdeffield(GDid_utm, "Height", "Hghtdim", NULL, H5T_NATIVE_FLOAT, HDFE_AUTOMERGE);
status = GDdeffield(GDid_utm, "Impedance", "XDim,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_AUTOMERGE);
status = GDdeffield(GDid_utm, "Grounding", "Conduction,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_NOMERGE);
status = GDdeffield(GDid_igoode, "SensorG", "XDim,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_NOMERGE);
status = GDdeffield(GDid_igoode, "VoltageA", "Radiant,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_NOMERGE);
status = GDdeffield(GDid_polar_np, "Bypass", "XDim,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_NOMERGE);
status = GDdeffield(GDid_lamaz, "Temperature", "Flux,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_NOMERGE);
status = GDdeffield(GDid_geo, "Depth", "XDim,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_NOMERGE);
status = GDdeffield(GDid_geo, "Interval", "Gradient,YDim", NULL, H5T_NATIVE_FLOAT, HDFE_NOMERGE);

status = GDdetach(GDid_utm);
GDid_utm = GDattach(gdfid, "UTM");

/* Write in data to the data fields */
status = GDwritefield(GDid_utm, "Voltage", start, stride, edge, utmray);
status = GDwritefield(GDid_utm, "Drift", start, NULL, edge, utmray);

start[0] = 0;
edge[0] = 4800;

status = GDwritefield(GDid_utm, "Time", start, NULL, edge, utmtmeray);

start[0] = 0;
edge[0] = 80;

status = GDwritefield(GDid_utm, "Height", start, NULL, edge, utmhght);

start[0] = 0;
start[1] = 0;

edge[0] = 60;
edge[1] = 80;

status = GDwritefield(GDid_utm, "Impedance", start, NULL, edge, utmray);

start[0] = 0;
edge[0] = 25;
start[1] = 0;
edge[1] = 80;

status = GDwritefield(GDid_utm, "Grounding", start, NULL, edge, ray3);

start[0] = 0;
edge[0] = 80;
start[1] = 0;
edge[1] = 60;

status = GDwritefield(GDid_igoode, "SensorG", start, NULL, edge, gooderay);

start[0] = 0;
start[1] = 0;

edge[0] = 45;
edge[1] = 60;

status = GDwritefield(GDid_igoode, "VoltageA", start, NULL, edge, gooderay2);

start[0] = 0;
start[1] = 0;
edge[0] = 60;
edge[1] = 80;

status = GDwritefield(GDid_polar_np, "Bypass", start, NULL, edge, psray);

start[0] = 0;
start[1] = 0;
edge[0] = 65;
edge[1] = 80;
status = GDwritefield(GDid_lamaz, "Temperature", start, NULL, edge, lamazray2);

    start[0] = 0;
    start[1] = 0;
    edge[0] = 60;
    edge[1] = 80;

status = GDwritefield(GDid_geo, "Depth", start, NULL, edge, georay);

    start[0] = 0;
    start[1] = 0;
    edge[0] = 85;
    edge[1] = 80;

status = GDwritefield(GDid_geo, "Interval", start, NULL, edge, georay2);

    /* --------------------------------------------------- */
    /* Write attributes for the UTM grid */
    /* --------------------------------------------------- */
    status = GDwriteattr(GDid_utm, "Resistance", H5T_NATIVE_INT, intattr, attr);
    status = GDwriteattr(GDid_utm, "Current", H5T_NATIVE_FLOAT, floatattr, attr2);

    /* --------------------------------------------------- */
    /* Detach from the grids */
    /* --------------------------------------------------- */
    status = GDdetach(GDid_utm);
    status = GDdetach(GDid_polar_np);
    status = GDdetach(GDid_igoode);
    status = GDdetach(GDid_som);
    status = GDdetach(GDid_lamaz);
    status = GDdetach(GDid_geo);

    /* --------------------------------------------------- */
    /* Close the file "Grid.h5" */
    /* --------------------------------------------------- */
    status = GDclose(gdfid);
}

6.2.1.2 A FORTRAN Example of a Simple Grid Creation

c ----------------------------------------------------
  Simple driver to create and write data to the HDF-EOS grid file
  ----------------------------------------------------
program testgrid

implicit none
integer status
integer gdfid
integer dummy
integer*4 GDid_utm, GDid_igoode, GDid_polar_np
integer*4 GDid_polar_sp
integer*4 GDid_geo, GDid_som, GDid_lamaz, GDid_hom
integer*4 GDid_lamcon, GDid_tm, GDid_poly, GDid_is

integer i, attr(4)
integer*4 zonecode
integer*4 index1, index2

real*4 utmray(60,80), ray3(25,80)
real*4 psray(360,90), gooderay(120,60)
real*4 utmhght(80), georay(60,80)
real*4 gooderay2(45,60), psray2(75,90)
real*4 lamazray2(65,180), georay2(85,80)
real*4 utmcnt, ray3cnt
real*4 psCnt, goodecnT
real*4 hghtinit, lamazcnt, geocnt
real*4 attr2(5)

integer*4 spherecode, xdim, ydim
integer*4 start(2)
integer*4 stride(2)
integer*4 edge(2)
integer*4 intattr(1)
integer*4 floatattr(1)

real*8 projparm(16), uplft(2), lowrgt(2)
real*8 utmmeray(4800), tmeinit
integer gdopen, gdcreate, gdattach
integer gddefproj, gddefdim, gddeffld, gddetach
integer gdclose
integer gdwrfld, gdwrattr, gddeforigin
integer gddefpreg

integer H5F_ACC_RDONLY
parameter (H5F_ACC_RDONLY = 0)
integer H5F_ACC_RDWR
parameter (H5F_ACC_RDWR = 1)
integer H5F_ACC_TRUNC
parameter (H5F_ACC_TRUNC = 2)
integer H5F_ACC_DEBUG
parameter (H5F_ACC_DEBUG = 8)
integer H5F_ACC_EXCL
parameter (H5F_ACC_EXCL = 4)
integer H5T_NATIVE_FLOAT
parameter (H5T_NATIVE_FLOAT = 1)
integer H5T_NATIVE_INT
parameter (H5T_NATIVE_INT = 0)
integer H5T_NATIVE_DOUBLE
parameter (H5T_NATIVE_DOUBLE = 2)
integer HDFE_AUTOMERGE
parameter (HDFE_AUTOMERGE = 1)
integer HDFE_NOMERGE
parameter (HDFE_NOMERGE = 0)
integer HDFE_CENTER
parameter (HDFE_CENTER = 0)
integer HDFE_CORNER
parameter (HDFE_CORNER = 1)
integer HDFE_GD_UL
parameter (HDFE_GD_UL = 0)
integer HDFE_GD_UR
parameter (HDFE_GD_UR = 1)
integer HDFE_GD_LL
parameter (HDFE_GD_LL = 2)
integer HDFE_GD_LR
parameter (HDFE_GD_LR = 3)
integer HDFE_NENTDIM
parameter (HDFE_NENTDIM = 0)
integer HDFE_NENTDFLD
parameter (HDFE_NENTDFLD = 4)
integer HDFE_NOPREVSUB
parameter (HDFE_NOPREVSUB = -1)

integer GCTP_GEO
parameter (GCTP_GEO = 0)
integer GCTP_UTM
parameter (GCTP_UTM = 1)
integer GCTP_LAMCC
parameter (GCTP_LAMCC = 4)
integer GCTP_PS
parameter (GCTP_PS = 6)
integer GCTP_POLYC
parameter (GCTP_POLYC = 7)
integer GCTP_TM
parameter (GCTP_TM = 9)
integer GCTP_LAMAZ
parameter (GCTP_LAMAZ = 11)
integer GCTP_HOM
parameter (GCTP_HOM = 20)
integer GCTP_SOM
parameter (GCTP_SOM = 22)
integer GCTP_GOOD
parameter (GCTP_GOOD = 24)
integer GCTP_ISINUS
parameter (GCTP_ISINUS = 99)

status = -1
attr(1) = 11
attr(2) = 33
attr(3) = 66
attr(4) = 99
attr2(1) = 5.1
attr2(2) = 17.2
attr2(3) = 28.3
attr2(4) = 39.4
attr2(5) = 57.5

start(1) = 10
start(2) = 10
stride(1) = 1
stride(2) = 1
edge(1) = 10
edge(2) = 10
intattr(1) = 4
floatattr(1) = 5
tmeinit = 35232487.2
goecnt = 323.0
pscnt = -134.5
utmcnt = -799.0
ray3cnt = -19.5

c Populate data arrays

do 20 index1 =1, 60
do 10 index2 =1, 80
  utmray(index1,index2) = utmcnt
  georay(index1,index2) = goecnt
  utmcnt = utmcnt + 0.4
  goecnt = goecnt + 0.2
10  continue
20 continue

do 30 index1 =1,4800
  utmhtray(index1) = tmeinit
  tmeinit = tmeinit + 70.7
30 continue

do 40 index1 =1, 80
  utmht(index1) = hghtinit
  hghtinit = hghtinit + 2.75
40 continue

do 60 index1 =1, 85
  do 50 index2 =1, 80
    georay2(index1,index2) = goecnt
    goecnt = goecnt - .9
50  continue
60 continue

do 80 index1 =1, 25
  do 70 index2 =1, 90
    ray3(index1,index2) = ray3cnt
    ray3cnt=ray3cnt + .1
70  continue
80 continue

do 100 index1 =1, 360
  do 90 index2 =1, 90
psray(index1,index2) = pscnt
pscnt = pscnt + .4
90 continue
100 continue

   do 120 index1 =1, 75
   do 110 index2 =1, 90
      psray2(index1,index2) = pscnt
      pscnt = pscnt - .4
   110 continue
120 continue

   do 140 index1 =1, 120
   do 130 index2 =1, 60
      gooderay(index1,index2) = goodecnt
      goodecnt = goodecnt + .4
   130 continue
140 continue

   do 160 index1 =1, 45
   do 150 index2 =1, 60
      gooderay2(index1,index2) = goodecnt
      goodecnt = goodecnt - .4
   150 continue
160 continue

   do 240 index1 =1, 65
   do 230 index2 =1, 180
      lamazray2(index1,index2) = lamazcnt
      lamazcnt = lamazcnt + .6
   230 continue
240 continue

   do 250 index1 =1, 16
      projparm(index1) = 0.0
250 continue

---

*Open HDF-EOS file "Grid.h5" (if the named file does not exist, it will be automatically created with the read/write access mode)*

```
gdfid = GDopen("Grid.h5", H5F_ACC_TRUNC)
```

---

*Create UTM grid*

```
xdim = 60
ydim = 80
uplft(1) = -512740.28306
uplft(2) = 2733747.62890
lowrgt(1) = -12584.57301
lowrgt(2) = 1946984.64021

GDdid_utm = GDcreate(gdfid, "UTM", xdim, ydim, uplft, lowrgt)
```
c ---------------------------------------------------
c Create Geographic grid
---------------------------------------------------
xdim = 60
ydim = 80
uplft(1) = -126000000.00
uplft(2) = -064000000.00
lowrgt(1) = -120000000.00
lowrgt(2) = -072000000.00
GDid_geo = GDcreate(gdfid, "Geo", xdim, ydim, uplft, lowrgt)

---------------------------------------------------
c Create Polar Stereo grid of north hemisphere
---------------------------------------------------
xdim=360
ydim=90
uplft(1) = -10447125.82759
uplft(2) = 10447125.82759
lowrgt(1) = 10447125.82759
lowrgt(2) = -10447125.82759
gdid_polar_np = GDcreate(gdfid, "Polar_np", xdim, ydim, uplft, lowrgt)

---------------------------------------------------
c Create Polar Stereo grid of south hemisphere
---------------------------------------------------
xdim=360
ydim=90
uplft(1) = 10447125.82759
uplft(2) = -10447125.82759
lowrgt(1) = -10447125.82759
lowrgt(2) = 10447125.82759
gdid_polar_sp = GDcreate(gdfid, "Polar_sp", xdim, ydim, uplft, lowrgt)

---------------------------------------------------
c Create Interrupted Goode grid
---------------------------------------------------
xdim=120
ydim=60
uplft(1) = -11119487.42844
uplft(2) = 8673539.24806
lowrgt(1) = 15567282.39984
lowrgt(2) = -8673539.24806
GDid_igoode = GDcreate(gdfid, "IGoode", xdim, ydim, uplft, lowrgt)

---------------------------------------------------
c Create Space Oblique Mercator grid
---------------------------------------------------
xdim=359
ydim=321
uplft(1) = 49844710.48057
uplft(2) = 884884.39883
lowrgt(1) = 30521379.68485
lowrgt(2) = 1152027.64253
GDid_som = GDcreate(gdfid, "SOM", xdim, ydim, uplft, lowrgt)

Create Lambert Azimuthal grid

xdim=719
ydim=180
uplft(1) = 0.0000
uplft(2) = 9009950.36324
lowrgt(1) = 0.0000
lowrgt(2) = -9009950.36324
GDid_lamaz = GDcreate(gdfid, "Lamaz", xdim, ydim, uplft, lowrgt)

Create Hotin Oblique Mercator grid

xdim=200
ydim=90
uplft(1) = 3422259.57265
uplft(2) = 6824822.05796
lowrgt(1) = -17519429.48100
lowrgt(2) = 4994368.88166
GDid_hom = GDcreate(gdfid, "hom", xdim, ydim, uplft, lowrgt)

Create Lambert Conformal grid

xdim=351
ydim=171
uplft(1) = -2279109.37671
uplft(2) = 12358083.24054
lowrgt(1) = -56342817.96247
lowrgt(2) = -24776979.34092
GDid_lamcon = GDcreate(gdfid, "lamcon", xdim, ydim, uplft, lowrgt)

Create Transverse Mercator grid

xdim=181
ydim=171
uplft(1) = 4855670.77539
uplft(2) = 9458558.92483
lowrgt(1) = 5201746.43983

6-37
162-WP-004-001
lowrgt(2) = -10466077.24942
GDid_tm = GDcreate(gdfid, "tm", xdim, ydim, uplft, lowrgt)

---
c
Create Polyconic grid
---
xdim=161
ydim=171
uplft(1) = -250873.85859
uplft(2) = 12669051.66767
lowrgt(1) = 850873.85859
lowrgt(2) = -7137259.12615
GDid_poly = GDcreate(gdfid, "poly", xdim, ydim, uplft, lowrgt)

---
c
Create IS grid
---
xdim=351
ydim=171
uplft(1) = 1436267.12618
uplft(2) = 9451564.31420
lowrgt(1) = 1343604.73094
lowrgt(2) = -9451564.31420
GDid_is = GDcreate(gdfid, "is", xdim, ydim, uplft, lowrgt)

---
c
Define projections
---
zonecode = -13
spherecode = 0

do 260 i=1,16
   projparm(i)=0.
260 continue
status = GDdefproj(GDid_utm, GCTP_UTM, zonecode, spherecode, projparm)

do 270 i=1,16
   projparm(i)=0.
270 continue

spherecode = 0
projparm(6) = 40000000.00

status=GDdefproj(GDid_polar_np, GCTP_PS, dummy, spherecode, projparm)

do 280 i=1,16
   projparm(i)=0.
280 continue

---
status=GDdefproj(GDid_igoode, GCTP_GOOD, dummy, dummy, projparm)
do 290 i=1,16
   projparm(i)=0.
290 continue

status=GDdefproj(GDid_lamaz, GCTP_LAMAZ, dummy, dummy, projparm)
do 300 i=1,16
   projparm(i)=0.
300 continue

projparm(3) = 3
projparm(4) = 150
projparm(13) = 1

status=GDdefproj(GDid_som, GCTP_SOM, dummy, dummy, projparm)
status=GDdefproj(GDid_geo, GCTP_GEO, dummy, dummy, dummy)
do 310 i=1,16
   projparm(i)=0.
310 continue

projparm(3) = 0.9996
projparm(6) = 20000000.00
projparm(9) = -75000000.00
projparm(10) = 10000000.00
projparm(11) = -95000000.00
projparm(12) = 30000000.00

status=GDdefproj(GDid_hom, GCTP_HOM, dummy, dummy, projparm)
do 320 i=1,16
   projparm(i)=0.
320 continue

projparm(3) = 20000000.00
projparm(4) = 40000000.00
projparm(5) = -75000000.00

status=GDdefproj(GDid_lamcon, GCTP_LAMCC, dummy, dummy, projparm)
do 330 i=1,16
   projparm(i)=0.
330 continue

projparm(3) = 0.9996
projparm(5) = -75000000.00
projparm(7) = 50000000.00

status=GDdefproj(GDid_tm, GCTP_TM, dummy, dummy, projparm)
do 340 i=1,16
   projparm(i)=0.
340 continue
projparm(5) = 75000000.00
projparm(6) = -25000000.00
projparm(7) = 300000.00

status=GDdefproj(GDid_poly, GCTP_POLYC, dummy, dummy, projparm)

do 350 i=1,16
   projparm(i)=0.
350 continue

projparm(5) = 0.
projparm(6) = 40000000.00

status=GDdefproj(GDid_is, GCTP_ISINUS, dummy, dummy, projparm)

c Define pixel registration in the grids

c status = GDdefpreg(GDid_utm, HDFE_CORNER)
status = GDdefpreg(GDid_polar_np, HDFE_CORNER)
status = GDdefpreg(GDid_igoode, HDFE_CORNER)
status = GDdefpreg(GDid_som, HDFE_CORNER)
status = GDdefpreg(GDid_lamaz, HDFE_CORNER)
status = GDdefpreg(GDid_geo, HDFE_CORNER)

c Define origin of projections in the grids

c status = GDdeforigin(GDid_utm, HDFE_GD_UL)
status = GDdeforigin(GDid_polar_np, HDFE_GD_UR)
status = GDdeforigin(GDid_igoode, HDFE_GD_LL)
status = GDdeforigin(GDid_som, HDFE_GD_LR)
status = GDdeforigin(GDid_lamaz, HDFE_GD_UL)
status = GDdeforigin(GDid_geo, HDFE_GD_UR)

c Define dimensions

c status = GDdefdim(GDid_utm, "Conduction", 25)
status = GDdefdim(GDid_utm, "Timedim", 4800)
status = GDdefdim(GDid_utm, "Hghtdim", 80)
status = GDdefdim(GDid_polar_np, "Convection", 75)
status = GDdefdim(GDid_igoode, "Radiant", 45)
status = GDdefdim(GDid_som, "Emission", 55)
status = GDdefdim(GDid_lamaz, "Flux", 65)
status = GDdefdim(GDid_geo, "Gradient", 85)

c Detach from the grids

c status = GDdetach(GDid_utm)
status = GDdetach(gdid_polar_np)
status = GDdetach(GDid_polar_sp)
status = GDdetach(GDid_igoode)
status = GDdetach(GDid_som)
status = GDdetach(GDid_lamaz)
status = GDdetach(GDid_geo)
status = GDdetach(GDid_hom)
status = GDdetach(GDid_lamcon)
status = GDdetach(GDid_tm)
status = GDdetach(GDid_poly)
status = GDdetach(GDid_is)

c -- Re-attach to the grids

c
GDid_utm = GDattach(gdfid, "UTM")
GDid_igoode = GDattach(gdfid, "IGoode")
GDid_lamaz = GDattach(gdfid, "Lamaz")
GDid_polar_np = GDattach(gdfid, "Polar_np")
GDid_som = GDattach(gdfid, "SOM")
GDid_geo = GDattach(gdfid, "Geo")

c -- Define data fields for the grids

c
status = GDdeffld(GDid_utm, "Voltage", "XDim,YDim","
1 ,H5T_NATIVE_FLOAT, HDFE_AUTOMERGE)
status = GDdeffld(GDid_utm, "Drift", "XDim,YDim","
1 ,H5T_NATIVE_FLOAT, HDFE_AUTOMERGE)
status = GDdeffld(GDid_utm, "Time", "Timedim",""
1 ,H5T_NATIVE_DOUBLE, HDFE_AUTOMERGE)
status = GDdeffld(GDid_utm, "Height", "Hghtdim",""
1 ,H5T_NATIVE_FLOAT, HDFE_AUTOMERGE)
status = GDdeffld(GDid_utm, "Impedance", "XDim,YDim","
1 ,H5T_NATIVE_FLOAT, HDFE_AUTOMERGE)
status = GDdeffld(GDid_utm, "Grounding", "Conduction,YDim","
1 "","HST NATIVE FLOAT, HDFE_NOMERGE)
status = GDdeffld(GDid_igoode, "SensorG", "XDim,YDim",""
1 "","HST NATIVE FLOAT, HDFE_NOMERGE)
status = GDdeffld(GDid_igoode, "VoltageA", "Radiant,YDim",""
1 "","HST NATIVE FLOAT, HDFE_NOMERGE)
status = GDdeffld(GDid_polar_np, "Bypass", "XDim,YDim",""
1 "","HST NATIVE FLOAT, HDFE_NOMERGE)
status = GDdeffld(GDid_lamaz, "Temperature", "Flux,YDim",""
1 "","HST NATIVE FLOAT, HDFE_NOMERGE)
status = GDdeffld(GDid_geo, "Depth", "XDim,YDim",""
1 "","HST NATIVE FLOAT, HDFE_NOMERGE)
status = GDdeffld(GDid_geo, "Interval", "Gradient,YDim",""
1 "","HST NATIVE FLOAT, HDFE_NOMERGE)

c Write in data to the data fields

c
6-41 162-WP-004-001
status = GDwrfld(GDid_utm, "Voltage", start, stride, edge,
1 utmray)
status = GDwrfld(GDid_utm, "Drift", start, stride, edge,
1 utmray)

start(1) = 0
dedge(1) = 4800

status = GDwrfld(GDid_utm, "Time", start, stride, edge,
1 utmmeray)

start(1) = 0
dedge(1) = 80

status = GDwrfld(GDid_utm, "Height", start, stride, edge,
1 utmhght)

start(1) = 0
start(2) = 0
dedge(1) = 60
dedge(2) = 80

status = GDwrfld(GDid_utm, "Impedance", start, stride, edge,
1 utmray)

start(1) = 0
dedge(1) = 25
start(2) = 0
dedge(2) = 80

status = GDwrfld(GDid_utm, "Grounding", start, stride, edge,
1 ray3)

start(1) = 0
dedge(1) = 80
start(2) = 0
dedge(2) = 60

status = GDwrfld(GDid_igoode, "SensorG", start, stride, edge,
1 gooderay)

start(1) = 0
start(2) = 0
dedge(1) = 45
dedge(2) = 60

status = GDwrfld(GDid_igoode, "VoltageA", start, stride, edge
1 , gooderay2)

start(1) = 0
start(2) = 0
dedge(1) = 60
dedge(2) = 80

status = GDwrfld(GDid_polar_np, "Bypass", start, stride, edge
1

6-42 162-WP-004-001
1, psray)

start(1) = 0
start(2) = 0
edge(1) = 65
edge(2) = 80

status = GDwrfld(GDid_lamaz, "Temperature", start, stride, 
1 edge, lamazray2)

start(1) = 0
start(2) = 0
edge(1) = 60
edge(2) = 80

status = GDwrfld(GDid_geo, "Depth", start, stride, edge, 
1 georay)

start(1) = 0
start(2) = 0
edge(1) = 85
edge(2) = 80

status = GDwrfld(GDid_geo, "Interval", start, stride, edge, 
1 georay2)

c Write attributes for the UTM grid
c

status = GDwrattr(GDid_utm, "Resistance", H5T_NATIVE_INT, 
1 intattr, attr)
status = GDwrattr(GDid_utm, "Current", H5T_NATIVE_FLOAT, 
1 floatattr, attr2)

c Detach from the grids
c

status = GDdetach(GDid_utm)
status = GDdetach(GDid_polar_np)
status = GDdetach(GDid_igoode)
status = GDdetach(GDid_som)
status = GDdetach(GDid_lamaz)
status = GDdetach(GDid_geo)

c Close the file "Grid.h5"
c
status = GDclose(gdfid)
stop
end
6.2.2 Performing Subsetting for a Grid

The following C and FORTRAN programs each demonstrate how to perform subsetting for a Grid object in an HDF-EOS file using the HDF-EOS Grid interface.

6.2.2.1 A C Example of a Subsetting for a Grid

```c
/* ---------------------------------------------------------------------- */
/* Simple driver demonstrating how to perform subsetting along XDim and */
/* YDim for a grid. */
/* NOTE: make sure that the HDF-EOS file Gridc_Test.h5 produced by the */
/* test driver testgrid_he3.c exists. */
/* ---------------------------------------------------------------------- */

#include "HdfEosDef.h"

main()
{
    herr_t status;
    int i;
    hid_t gdfid, GDid_geo;
    hid_t regionID;
    int32 rank, size;
    hsize_t dims[8];
    H5T_class_t *numtype = (H5T_class_t *)NULL;
    float32 *buffer;
    float64 corlon[2], corlat[2];
    float64 iuplft[2], ilowrgt[2];

    iuplft[0] = 0.;
    iuplft[1] = 0.;
    ilowrgt[0] = 0.;
    ilowrgt[1] = 0.;

    /* --------------------------------------------------------------------- */
    /* Open grid file "Gridc_Test.h5" and attach to the named grid */
    /* objects. */
    /* --------------------------------------------------------------------- */
    gdfid = GDopen("Gridc_Test.h5", H5F_ACC_RDONLY);
    printf("\t	Value returned by GDopen %d\n\n", gdfid);
    GDid_geo = GDattach(gdfid, "Geo");
    printf("\t	Value returned by GDattach %d\n\n", GDid_geo);

    /* --------------------------------------------------------------------- */
    /* Define a longitude-latitude box region for the named grid */
    /* --------------------------------------------------------------------- */
```
corlon[0] = -123.5;
corlon[1] = -121.5;
corlat[0] = -67.0;
corlat[1] = -70.0;

regionID = GDdefboxregion(GDid_geo, corlon, corlat);
printf("\t\tRegion id returned by GDdefboxregion %d\n", regionID);

/* retrieve information about the subsetted region */
numtype = (H5T_class_t *)calloc(1, sizeof(H5T_class_t));
status = GDregioninfo(GDid_geo, regionID, "Depth", numtype, &rank, dims, &size, iuplft, ilowrgt);
printf("\t\tNumber type of region %d rank of region %d\n", *numtype, rank);
printf("\t\tDimension of region %lu %lu Size of region in bytes %d\n", (unsigned long)dims[0], (unsigned long)dims[1], size);
free(numtype);

/* read data into the data buffer from the subsetted region */
buffer = (float32 *)malloc(size);
status = GDextractregion(GDid_geo, regionID, "Depth", buffer);
for(i = 0; i < 5; i++)
{
    printf("\t\tValue of buffer %f\n", buffer[i]);
}
free(buffer);

status = GDdetach(GDid_geo);
printf("\t\tValue returned by GDdetach %d\n", status);
status = GDclose(gdfid);
printf("\t\tValue returned by GDclose %d\n", status);

}  

6.2.2.2 A FORTRAN Example of a Subsetting for a Grid

---

c Simple driver demonstrating how to perform subsetting along XDim and YDim for a grid.
NOTE: make sure that the HDF-EOS file Gridc_Test.h5 produced by the test driver testgrid_he3.c exists.

---

implicit none
integer gdopen, gdattach, gddetach
integer     gdclose, gdreginfo
integer     gddefboxreg
integer     status
integer     i
integer     gdfid
integer     GDid_geo
integer     regionID
integer*4   rank, size
integer*4   dims(8)
integer     numtype(1)
real*4      buffer(48000)
real*8      corlon(2), corlat(2)
real*8      iuplft(2), ilowrgt(2)
integer     H5F_ACC_RDONLY
parameter   (H5F_ACC_RDONLY = 0)
iuplft(1)   = 0.
iuplft(2)   = 0.
ilowrgt(1)  = 0.
ilowrgt(2)  = 0.

c  Open grid file "Gridc_Test.h5" and attach to the named grid objects.

gdfid = GDopen("Grid.h5", H5F_ACC_RDONLY)
print *, ' Value returned by GDopen = ', gdfid
GDid_geo = GDattach(gdfid, "Geo")
print *, ' Value returned by GDattach = ', GDid_geo

 Define a longitude-latitude box region for the named grid

corlon(1)  = -123.5
corlon(2)  = -121.5
corlat(1)  = -67.0
corlat(2)  = -70.0

regionID = GDdefboxreg(GDid_geo, corlon, corlat)
print *, ' Region id Returned by GDdefboxregion = ', regionID

 Retrieve information about the subsetted region
 for the named data field
status = GDreginfo(GDid_geo, regionID,"Depth",numtype,rank > ,dims,size,iuplft,Ilowrgt)
print *,'
print *,' Status returned by GDregioninfo = ', status
print *,' Number type of region = ', numtype,
> ' rank of region = ', rank
print *,' Dimension of region = ', dims(1), dims(2),
> ' Size of region in bytes = ', size
print *,' Upper left point = ', iuplft(1),iuplft(2),
> ' Lower right point= ', ilowrgt(1),ilowrgt(2)

6.3 Combining HDF and HDF-EOS Objects
The HDF-EOS structures, swath, point, and grid, are built out of the standard HDF5 objects such
as groups, data sets, and attributes and thus an HDF-EOS file can contain both HDF and HDF-
EOS entities.

6.3.1 Adding HDF-EOS Structures to an Existing HDF File
In this example we open an existing HDF file, NativeHDF.hdf, and add a swath structure.
Because the swath is an HDF-EOS entity we use the HDF-EOS API routines to open, create,
detach and close, ie, SWopen, SWcreate, SWdetach, and SWclose.

    /* Open HDF file for read-write access */
    fileID = SWopen("NativeHDF.hdf", DFACC_RDWR);
    /* Create Swath Structure */
    swathID = SWcreate(fileID, "SwathStructure");
    /* Detach Swath Structure */
    status = SWdetach(swathID);
    /* Close File */
    status = SWclose(fileID);
);
This page intentionally left blank.
7. Writing ODL Metadata into HDF-EOS

7.1 A C Example of Metadata Write

The following C code fragment is an example of how a user can write granule metadata (or inventory metadata) into their HDF-EOS file. The Metadata Configuration File (MCF), which the code accesses is given in section 7.3. The output ODL file which results from running the code in 7.2 is given in section 7.4.

In order to assist a developer of software for producing data in ECS format, a file template called filetable.temp is provided in section 7.5. This file is similar to the Process Control File used by the larger SDP Toolkit, but simpler. It is used to specify the relationship between logical file identifiers used in source code and physical files containing input data or output data. It is also used to specify the IDs for log status reports and for the MCFs.

It should be mentioned that currently only MTD TOOLKIT (a subset of SDP TOOLKIT that handles metadata and Time/Date conversions) has been modified to write metadata into HDF-EOS files utilizing HDF5. Moreover, in the new MTD TOOLKIT for HDF5 some tools, such as PGS_MET_GetPCAttr() and PGS_MET_InitNonMCF() are not available yet. Details on Metadata Configuration Files can be found in the SDP Toolkit Users Guide for ECS project, Section 6.2.1 and Appendix J (33-CD-500-001). Details on how to install and use MTD TOOLKIT tools can be found in Toolkit_MTD Users Guide.

7.2 C Code

```c
/* include files */
#include <PGS_MET.h>
#include <PGS_tk.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hdf5.h>
#include <PGS_SMF.h>

#define INVENTORYMETADATA 1
#define ARCHIVEDMETADATA 2
#define ODL_IN_MEMORY 0

extern PGSt_SMF_status
PGS_PC_GetReference(PGSt_MET_Logical prodID, PGSt_integer *version, char
                     *referenceID);
```
int main()
{

/**********************************************************************************
Declarations.
**********************************************************************************/
PGSt_MET_all_handles mdHandles;
PGSt_MET_all_handles handles;
char fileName1[PGSd_MET_FILE_PATH_MAX]="";
char fileName2[PGSd_MET_FILE_PATH_MAX]="";
char my_HDF_file[PGSd_MET_FILE_PATH_MAX]="";
char msg[PGS_SMF_MAX_MSG_SIZE];
char mnemonic[PGS_SMF_MAX_MNEMONIC_SIZE];
char fileMessage[PGS_SMF_MAX_MSG_SIZE];
int32_t sdid1;
PGSt_SMF_status ret = PGS_S_SUCCESS;
char *informationname;
PGSt_integer ival =3;
PGSt_double dval=203.2;
PGSt_integer fileId, fileId2;
PGSt_integer i;
PGSt_integer version;
PGSt_SMF_status returnStatus;
char *mysaval[5];

/**********************************************************************************
/* Associate logical IDs with physical filenames. */
/**********************************************************************************/
ret=PGS_MET_SetFileId();
printf("ret after PGS_MET_SetFileId()is %d in Main\n",ret);

if(ret != PGS_S_SUCCESS)
{
  printf(" Failed in assigning logical IDs\n");
}

/*recover file name for fileId=PGSd_MET_MCF_FILE */
version = 1;
fileId = PGSd_MET_MCF_FILE;
returnStatus = PGS_PC_GetReference( fileId, &version,
  fileName1);
if ( returnStatus != PGS_S_SUCCESS )
{
  PGS_SMF_GetMsg( &returnStatus, mnemonic, msg );
  strcpy(fileMessage, msg);
  PGS_SMF_SetDynamicMsg( returnStatus,fileMessage,
    "metatest" );
}
else
{

printf("The input file for ID %d is %s\n", fileId, fileName1);
}

informationname=(char *) malloc(330);
/* Initialize MCF file */

fileId = 10250;
ret=PGS_MET_Init(fileId, handles);

if (ret != PGS_S_SUCCESS)
{
    printf("initialization failed\n");
    return (-1);
}
else
{
    printf("ret after PGS_MET_Init is %d\n", ret);
}

/* test PGS_MET_SetAttr */

ival=667788;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
                   "QAPERCENTINTERPOLATEDDATA.1", &ival);
printf("ret after SetAttr for QAPERCENTINTERPOLATEDDATA.1 is %d\n", ret);

ival=12345;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
                   "QAPercentMissingData.1", &ival);
printf("ret after SetAttr for QAPercentMissingData.1 is %d\n", ret);

ival=123;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
                   "QAPercentOutOfBoundData.1", &ival);
printf("ret after SetAttr for QAPercentOutOfBoundData.1 is %d\n", ret);

ival=23456;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
                   "QAPercentOutOfBoundData.2", &ival);
printf("ret after SetAttr for QAPercentOutOfBoundData.2 is %d\n", ret);

ival=56789;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
                   "QAPercentMissingData.2", &ival);
printf("ret after SetAttr for QAPercentMissingData.2 is %d\n", ret);

strcpy(informationname,"Exercise1");
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
                   "AutomaticQualityFlagExplanation.1", &informationname);
printf("ret after SetAttr for AutomaticQualityFlagExplanation.1 is %d\n", ret);
strcpy(informationname, "1997/12/23");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "RangeBeginningDateTime", &informationname);
printf("ret after SetAttr for RangeBeginningDateTime is %d\n", ret);

strcpy(informationname, "1997.07/30");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "RangeBeginningDate", &informationname);
printf("ret after SetAttr for RangeBeginningDate is %d\n", ret);

strcpy(informationname, "ReprocessingplannINVENT");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "ReprocessingPlanned", &informationname);
printf("ret after SetAttr for ReprocessingPlanned is %d\n", ret);

strcpy(informationname, "ReprocessingplannARCHIVE");
ret = PGS_MET_SetAttr(handles[ARCHIVEDMETADATA],
     "ReprocessingPlanned", &informationname);
printf("ret after SetAttr for ReprocessingPlanned is %d\n", ret);

strcpy(informationname, "Reprocessingin");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "ReprocessingActual", &informationname);
printf("ret after SetAttr for ReprocessingActual is %d\n", ret);

strcpy(informationname, "ID1111");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "LocalGranuleID", &informationname);
printf("ret after SetAttr for LocalGranuleID is %d\n", ret);

strcpy(informationname, "version1234");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "LocalVersionID", &informationname);
printf("ret after SetAttr for LocalVersionID is %d\n", ret);

strcpy(informationname, "Flag1");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "DayNightFlag", &informationname);
printf("ret after SetAttr for DayNightFlag is %d\n", ret);

strcpy(informationname, "Flag1");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "DayNightFlag", &informationname);
printf("ret after SetAttr for DayNightFlag is %d\n", ret);

strcpy(informationname, "information1");
ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
     "ParameterName.1", &informationname);
printf("ret after SetAttr for ParameterName is %d\n", ret);
strcpy(informationname,"information2");
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
        "ParameterName.2", &informationname);
printf("ret after SetAttr for ParameterName.2 is %d\n", ret);

strcpy(informationname,"information3");
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
        "ParameterName.3", &informationname);
printf("ret after SetAttr for ParameterName is %d\n", ret);

strcpy(informationname,"information4");
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
        "ParameterName.4", &informationname);
printf("ret after SetAttr for ParameterName is %d\n", ret);

dval=111.11;
ret=PGS_MET_SetAttr(handles[ARCHIVEDMETADATA],
        "WestBoundingCoordinate", &dval);
printf("ret WestBoundingCoordinate is %d %f\n", ret, dval);

dval=222.22;
ret=PGS_MET_SetAttr(handles[ARCHIVEDMETADATA],
        "northBoundingCoordinate", &dval);
printf("ret northBoundingCoordinate is %d %f\n", ret, dval);

dval=333.33;
ret=PGS_MET_SetAttr(handles[ARCHIVEDMETADATA],
        "EastBoundingCoordinate", &dval);
printf("ret EastBoundingCoordinate is %d %f\n", ret, dval);

dval=444.44;
ret=PGS_MET_SetAttr(handles[ARCHIVEDMETADATA],
        "SouthBoundingCoordinate", &dval);
printf("ret SouthBoundingCoordinate is %d %f\n", ret, dval);

dval=11.11;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
        "WestBoundingCoordinate", &dval);
printf("ret WestBoundingCoordinate is %d %f\n", ret, dval);

dval=22.22;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
        "northBoundingCoordinate", &dval);
printf("ret northBoundingCoordinate is %d %f\n", ret, dval);

dval=33.33;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],
        "EastBoundingCoordinate", &dval);
printf("ret EastBoundingCoordinate is %d %f\n", ret, dval);

dval=44.44;
ret=PGS_MET_SetAttr(handles[INVENTORYMETADATA],

"SouthBoundingCoordinate", &dval);
printf("ret SouthBoundingCoordinate is %d %f\n", ret, dval);

/* Get the value of set attribute */

dval = 11.11;
ret = PGS_MET_GetSetAttr(handles[INVENTORYMETADATA],
                          "SouthBoundingCoordinate", &dval);

printf("after GetSetAttr: ret SouthBoundingCoordinate is %d 
%f\n", ret, dval);

/* Get data from config file */

ret = PGS_MET_GetConfigData("TEST_PARM_FLOAT", &dval);
printf("after PGS_MET_GetConfigData : ret TEST_PARM_INT is %d 
%f\n", ret, dval);

/* write metadata to HDF and ASCII files */
version = 1;
fileId = 5039;

ret = PGS_PC_GetReference(fileId, &version, my_HDF_file);
printf("after PGS_PC_GetReference ret =%d\n", ret);
printf("after PGS_PC_GetReference my_HDF_file = %s\n", my_HDF_file);

if (ret == PGS_S_SUCCESS)
{
    sdid1 = PGS_MET_SDstart(my_HDF_file, H5F_ACC_RDWR);
    printf("after PGS_MET_SDstart sdid1 =%d\n", sdid1);
}
else
{
    return (-1);
}

printf("After SDstart sdid1 is %d\n", sdid1);

/***** write INVENTORYMETADATA to HDF file *****/

ret = PGS_MET_Write(handles[INVENTORYMETADATA], "coremetadata", sdid1);
printf("ret after PGS_MET_Write is %d\n", ret);

if (ret != PGS_S_SUCCESS && ret != PGSMET_W_METADATA_NOT_SET)
{
    if (ret == PGSMET_E_MAND_NOT_SET)
    {
        printf("some mandatory parameters were not set\n");
    }
    else
    {
        printf("\n");
    }
}
printf("HDF write failed
");
}

/***** write ARCHIVEDMETADAT to HDF file *****/
ret=PGS_MET_Write(handles[ARCHIVEDMETADAT], "archivemetadata", sdid1);
printf("ret after PGS_MET_Write is %d\n", ret);

if(ret != PGS_S_SUCCESS && ret != PGSMET_W_METADATA_NOT_SET)
{
  if (ret == PGSMET_E_MAND_NOT_SET)
  {
    printf("some mandatory parameters were not set\n");
  }
  else
  {
    printf("HDF write failed\n");
  }
}

/***** write to non-HDF file *****/
fileId = 5804;
printf("non-hdf file to be written has fileId %d\n", fileId);
ret=PGS_MET_Write(handles[ODL_IN_MEMORY], NULL, fileId);
printf("ret after PGS_MET_Write is %d\n", ret);

if(ret != PGS_S_SUCCESS && ret != PGSMET_W_METADATA_NOT_SET)
{
  if (ret == PGSMET_E_MAND_NOT_SET)
  {
    printf("some mandatory parameters were not set\n");
  }
  else
  {
    printf("ASCII write failed\n");
  }
}

/***** write to default non-HDF file *****/
ret=PGS_MET_Write(handles[ODL_IN_MEMORY], NULL, NULL);
printf("ret after PGS_MET_Write is %d\n", ret);

if(ret != PGS_S_SUCCESS && ret != PGSMET_W_METADATA_NOT_SET)
{
  if (ret == PGSMET_E_MAND_NOT_SET)
  {
    printf("some mandatory parameters were not set\n");
  }
else
{
    printf("ASCII write failed\n");
}

(void)PGS_MET_SSend(sdid1);

PGS_MET_Remove();
free(informationname);

printf("Complete...\n");
return 0;

7.3 The Metadata Configuration File (MCF) for Code in Section 7.2

***************************************************************************
***************************************************************************
/* */
/* This is a working version of the MCF template that will be */
/* supplied with the next SDP Toolkit. This MCF template will */
/* NOT be official until the SDP Toolkit is released. All */
/* details are subject to change. */
/* */
/* This MCF file represents the ODL which is expected to be */
/* created when either Data Server or the MetaDataWorks tool */
/* uses the contents of an ESDT's INVENTORYMETADATA section in */
/* order to generate an ESDT-specific MCF. The level of */
/* metadata coverage presented here corresponds to the metadata */
/* requirement for granules in Full Class as described in */
/* Appendix B of DID 311 and Section 2.5 of the document 'BNF */
/* Representation of the B.0 Earth Science Data Model for the */
/* ECS Project' (420-TP-016-001). */
/* */
/* This MCF file's contents were based on the ESDT Descriptor */
/* file template Ver-1.6, 3/31/97. */
/* */
***************************************************************************
***************************************************************************

GROUP = INVENTORYMETADATA
GROUPTYPE = MASTERGROUP

/* ECSDataGranule */
GROUP = ECSDataGranule

    /* Note: SizeMBECSDataGranule will be set by DSS, */
    /* not by the science software. */
    OBJECT = SizeMBECSDataGranule
    Data_Location = "DSS"
    NUM _VAL = 1
TYPE = "DOUBLE"
Mandatory = "FALSE"
END_OBJECT = SizeMBECSDataGranule

OBJECT = ReprocessingPlanned
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "STRING"
  Mandatory = "TRUE"
END_OBJECT = ReprocessingPlanned

OBJECT = ReprocessingActual
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "STRING"
  Mandatory = "TRUE"
END_OBJECT = ReprocessingActual

OBJECT = LocalGranuleID
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "STRING"
  Mandatory = "TRUE"
END_OBJECT = LocalGranuleID

OBJECT = DayNightFlag
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "STRING"
  Mandatory = "TRUE"
END_OBJECT = DayNightFlag

OBJECT = ProductionDateTime
  Data_Location = "TK"
  NUM_VAL = 1
  TYPE = "DATETIME"
  Mandatory = "TRUE"
END_OBJECT = ProductionDateTime

OBJECT = LocalVersionID
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "STRING"
  Mandatory = "TRUE"
END_OBJECT = LocalVersionID

END_GROUP = ECSDataGranule

GROUP = MeasuredParameter

OBJECT = MeasuredParameterContainer
  Data_Location = "NONE"
  Class = "M"
  Mandatory = "TRUE"

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

  OBJECT = AutomaticQualityFlagExplanation
    Data_Location = "PGE"
    Mandatory = "TRUE"
    TYPE = "STRING"
    NUM_VAL = 1
    END_OBJECT = AutomaticQualityFlagExplanation

  OBJECT = OperationalQualityFlag
    Data_Location = "DAAC"
    Mandatory = "FALSE"
    TYPE = "STRING"
    NUM_VAL = 1
    END_OBJECT = OperationalQualityFlag

  OBJECT = OperationalQualityFlagExplanation
    Data_Location = "DAAC"
    Mandatory = "FALSE"
    TYPE = "STRING"
    NUM_VAL = 1
    END_OBJECT = OperationalQualityFlagExplanation

  OBJECT = ScienceQualityFlag
    Data_Location = "DP"
    Mandatory = "FALSE"
    TYPE = "STRING"
    NUM_VAL = 1
    END_OBJECT = ScienceQualityFlag

  OBJECT = ScienceQualityFlagExplanation
    Data_Location = "DP"
    Mandatory = "FALSE"
    TYPE = "STRING"
    NUM_VAL = 1
    END_OBJECT = ScienceQualityFlagExplanation

END_GROUP = QAFlags

GROUP = QAStats
  Class = "M"

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

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

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

OBJECT = QAPercentCloudCover
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "INTEGER"
  Mandatory = "TRUE"
END_OBJECT = QAPercentCloudCover
END_GROUP = QAStats
END_OBJECT = MeasuredParameterContainer
END_GROUP = MeasuredParameter

GROUP = OrbitCalculatedSpatialDomain
OBJECT = OrbitCalculatedSpatialDomainContainer

    Data_Location = "NONE"
    Class = "M"
    Mandatory = "TRUE"

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

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

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

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

OBJECT = EquatorCrossingLongitude
  Data_Location = "PGE"
  Mandatory = "TRUE"
  Class = "M"
  TYPE = "DOUBLE"
  NUM_VAL = 1
END_OBJECT = EquatorCrossingLongitude

OBJECT = EquatorCrossingTime
  Data_Location = "PGE"
  Mandatory = "TRUE"
  Class = "M"
  TYPE = "TIME"
  NUM_VAL = 1
END_OBJECT = EquatorCrossingTime

OBJECT = EquatorCrossingDate
  Data_Location = "PGE"
  Mandatory = "TRUE"
  Class = "M"
  TYPE = "DATE"
  NUM_VAL = 1
END_OBJECT = EquatorCrossingDate

END_OBJECT = OrbitCalculatedSpatialDomainContainer
END_GROUP = OrbitCalculatedSpatialDomain

GROUP = CollectionDescriptionClass

OBJECT = ShortName
  Data_Location = "MCF"
  NUM_VAL = 1
  TYPE = "STRING"
  Mandatory = "TRUE"
  Value = "L7ORF1"
END_OBJECT = ShortName

OBJECT = VersionID
  Data_Location = "MCF"
  NUM_VAL = 1
  TYPE = "STRING"
  Mandatory = "TRUE"
  Value = "1"
END_OBJECT = VersionID
END_GROUP = CollectionDescriptionClass

GROUP = SpatialDomainContainer

GROUP = HorizontalSpatialDomainContainer

/* ZoneIdentifierClass */
GROUP = ZoneIdentifierClass
OBJECT = ZoneIdentifier
   Data Location = "PGE"
   NUM_VAL = 1
   TYPE = "STRING"
   Mandatory = "TRUE"
END_OBJECT = ZoneIdentifier
END_GROUP = ZoneIdentifierClass

/* BoundingRectangle */
GROUP = BoundingRectangle
OBJECT = WestBoundingCoordinate
   Data Location = "PGE"
   NUM_VAL = 1
   TYPE = "DOUBLE"
   Mandatory = "TRUE"
END_OBJECT = WestBoundingCoordinate

OBJECT = NorthBoundingCoordinate
   Data Location = "PGE"
   NUM_VAL = 1
   TYPE = "DOUBLE"
   Mandatory = "TRUE"
END_OBJECT = NorthBoundingCoordinate

OBJECT = EastBoundingCoordinate
   Data Location = "PGE"
   NUM_VAL = 1
   TYPE = "DOUBLE"
   Mandatory = "TRUE"
END_OBJECT = EastBoundingCoordinate

OBJECT = SouthBoundingCoordinate
   Data Location = "PGE"
   NUM_VAL = 1
   TYPE = "DOUBLE"
   Mandatory = "TRUE"
END_OBJECT = SouthBoundingCoordinate
END_GROUP = BoundingRectangle
END_GROUP = HorizontalSpatialDomainContainer
END_GROUP = SpatialDomainContainer

/* RangeDateTime */
GROUP = RangeDateTime

OBJECT = RangeBeginningTime
   Data Location = "PGE"
   NUM_VAL = 1
   TYPE = "TIME"

END_OBJECT = RangeBeginningTime
Mandatory = "TRUE"
END_OBJECT = RangeBeginningTime

OBJECT = RangeEndingTime
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "TIME"
  Mandatory = "TRUE"
END_OBJECT = RangeEndingTime

OBJECT = RangeBeginningDate
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "DATE"
  Mandatory = "TRUE"
END_OBJECT = RangeBeginningDate

OBJECT = RangeEndingDate
  Data_Location = "PGE"
  NUM_VAL = 1
  TYPE = "DATE"
  Mandatory = "TRUE"
END_OBJECT = RangeEndingDate

END_GROUP = RangeDateTime

GROUP = AdditionalAttributes
OBJECT = AdditionalAttributesContainer
  Data_Location = "NONE"
  Class = "M"
  Mandatory = "TRUE"

  /* AdditionalAttributes */
  OBJECT = AdditionalAttributeName
    Data_Location = "PGE"
    Mandatory = "TRUE"
    TYPE = "STRING"
    Class = "M"
    NUM_VAL = 1
  END_OBJECT = AdditionalAttributeName

  /* InformationContent */
  GROUP = InformationContent
    Class = "M"

    OBJECT = ParameterValue
      Data_Location = "PGE"
      Mandatory = "TRUE"
      TYPE = "STRING"
      NUM_VAL = 1
    END_OBJECT = ParameterValue

END_GROUP = InformationContent

END_OBJECT = AdditionalAttributesContainer
END_GROUP = AdditionalAttributes

GROUP = OrbitParametersGranule

OBJECT = OrbitalParametersPointer
    Data_Location = "PGE"
    Mandatory = "TRUE"
    TYPE = "STRING"
    NUM_VAL = 1
END_OBJECT = OrbitalParametersPointer

END_GROUP = OrbitParametersGranule

/* StorageMediumClass */
GROUP = StorageMediumClass
OBJECT = StorageMedium
    Data_Location = "PGE"
    NUM_VAL = 10
    TYPE = "STRING"
    Mandatory = "TRUE"
END_OBJECT = StorageMedium
END_GROUP = StorageMediumClass

END_GROUP = INVENTORYMETADATA

GROUP = ARCHIVEDMETADATA
GROUPTYPE = MASTERGROUP

/* BoundingRectangle */
GROUP = BoundingRectangle
OBJECT = WestBoundingCoordinate
    Data_Location = "PGE"
    NUM_VAL = 1
    TYPE = "DOUBLE"
    Mandatory = "TRUE"
END_OBJECT = WestBoundingCoordinate

OBJECT = NorthBoundingCoordinate
    Data_Location = "PGE"
    NUM_VAL = 1
    TYPE = "DOUBLE"
    Mandatory = "TRUE"
END_OBJECT = NorthBoundingCoordinate

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

OBJECT = SouthBoundingCoordinate
    Data_Location = "PGE"
    NUM_VAL = 1
    TYPE = "DOUBLE"
    Mandatory = "TRUE"
7.4 The ODL Output File Which Results from Running Code in Section 7.2

/************************************************************************
/************************************************************************
/*
/* This is a working version of the MCF template that will be */
/* supplied with the next SDP Toolkit. This MCF template will */
/* NOT be official until the SDP Toolkit is released. All */
/* details are subject to change. */
/* */
/* This MCF file represents the ODL which is expected to be */
/* created when either Data Server or the MetaDataWorks tool */
/* uses the contents of an ESDT's INVENTORYMETADATA section in */
/* order to generate an ESDT-specific MCF. The level of */
/* metadata coverage presented here corresponds to the metadata */
/* requirement for granules in Full Class as described in */
/* Appendix B of DID 311 and Section 2.5 of the document 'BNF */
/* Representation of the B.0 Earth Science Data Model for the */
/* ECS Project' (420-TP-016-001). */
/* */
/* This MCF file's contents were based on the ESDT Descriptor */
/* file template Ver-1.6, 3/31/97. */
/* */
/************************************************************************
/************************************************************************

GROUP = INVENTORYMETADATA
GROUPTYPE = MASTERGROUP
/* ECSDataGranule */

GROUP = ECSDATAGRANULE

OBJECT = REPROCESSINGPLANNED
  NUM_VAL = 1
  VALUE = "Reprocessingplanned"
END_OBJECT = REPROCESSINGPLANNED

OBJECT = REPROCESSINGACTUAL
  NUM_VAL = 1
  VALUE = "Reprocess"
END_OBJECT = REPROCESSINGACTUAL
OBJECT
  NUM_VAL = 1
  VALUE  = "ID1111"
END_OBJECT

OBJECT
  NUM_VAL = 1
  VALUE  = "Flag1"
END_OBJECT

OBJECT
  NUM_VAL = 1
  VALUE  = "1999-11-23T18:16:01.000Z"
END_OBJECT

OBJECT
  NUM_VAL = 1
  VALUE  = "version1234"
END_OBJECT

END_GROUP

GROUP = ECSDATAGRANULE

OBJECT = MEASUREDPARAMETERCONTAINER
  CLASS = "1"

OBJECT
  CLASS = "1"
  NUM_VAL = 1
  VALUE  = "information1"
END_OBJECT

GROUP = QAFLAGS

OBJECT
  CLASS = "1"

OBJECT
  CLASS = "1"
  VALUE  = "NOT SET"
END_OBJECT

OBJECT
  CLASS = "1"
  VALUE  = "Exercise1"
END_OBJECT

END_GROUP
/* QAStats */

GROUP = QASTATS
CLASS = "1"

OBJECT = QAPERCENTINTERPOLATEDDATA
  NUM_VAL = 1
  CLASS = "1"
  VALUE = 667788
END_OBJECT = QAPERCENTINTERPOLATEDDATA

OBJECT = QAPERCENTMISSINGDATA
  NUM_VAL = 1
  CLASS = "1"
  VALUE = 12345
END_OBJECT = QAPERCENTMISSINGDATA

OBJECT = QAPERCENTOUTOFBOUNDS DATA
  NUM_VAL = 1
  CLASS = "1"
  VALUE = 123
END_OBJECT = QAPERCENTOUTOFBOUNDS DATA

OBJECT = QAPERCENTCLOUDCOVER
  NUM_VAL = 1
  CLASS = "1"
  VALUE = "NOT SET"
END_OBJECT = QAPERCENTCLOUDCOVER

END_GROUP = QASTATS
END_OBJECT = MEASUREDPARAMETERCONTAINER

OBJECT = MEASUREDPARAMETERCONTAINER
CLASS = "2"

OBJECT = PARAMETERNAME
  CLASS = "2"
  NUM_VAL = 1
  VALUE = "information2"
END_OBJECT = PARAMETERNAME

/* QAFlags */

GROUP = QAFLAGS
CLASS = "2"

OBJECT = AUTOMATICQUALITYFLAG
  NUM_VAL = 1
  CLASS = "2"
  VALUE = "NOT SET"
END_OBJECT = AUTOMATICQUALITYFLAG

OBJECT = AUTOMATICQUALITYFLAGEXPLANATION
  NUM_VAL = 1
CLASS = "2"
VALUE = "NOT SET"
END_OBJECT = AUTOMATICQUALITYFLAGEXPLANATION

END_GROUP = QAFLAGS

/* QAStats */

GROUP = QASTATS
CLASS = "2"

OBJECT = QAPERCENTINTERPOLATEDDATA
NUM_VAL = 1
CLASS = "2"
VALUE = "NOT SET"
END_OBJECT = QAPERCENTINTERPOLATEDDATA

OBJECT = QAPERCENTMISSINGDATA
NUM_VAL = 1
CLASS = "2"
VALUE = 56789
END_OBJECT = QAPERCENTMISSINGDATA

OBJECT = QAPERCENTOUTOFBOUNDSDATA
NUM_VAL = 1
CLASS = "2"
VALUE = 23456
END_OBJECT = QAPERCENTOUTOFBOUNDSDATA

OBJECT = QAPERCENTCLOUDCOVER
NUM_VAL = 1
CLASS = "2"
VALUE = "NOT SET"
END_OBJECT = QAPERCENTCLOUDCOVER

END_GROUP = QASTATS
END_OBJECT = MEASUREDPARAMETERCONTAINER

OBJECT = MEASUREDPARAMETERCONTAINER
CLASS = "3"

OBJECT = PARAMETERNAME
CLASS = "3"
NUM_VAL = 1
VALUE = "information3"
END_OBJECT = PARAMETERNAME

/* QAFlags */

GROUP = QAFLAGS
CLASS = "3"

OBJECT = AUTOMATICQUALITYFLAG
NUM_VAL = 1
CLASS = "3"
VALUE = "NOT SET"
END_OBJECT = AUTOMATICQUALITYFLAG

OBJECT = AUTOMATICQUALITYFLAGEXPLANATION
   NUM_VAL = 1
   CLASS = "3"
   VALUE = "NOT SET"
END_OBJECT = AUTOMATICQUALITYFLAGEXPLANATION

END_GROUP = QAFLAGS

/* QAStats */

GROUP = QASTATS
   CLASS = "3"

OBJECT = QAPERCENTINTERPOLATEDDATA
   NUM_VAL = 1
   CLASS = "3"
   VALUE = "NOT SET"
END_OBJECT = QAPERCENTINTERPOLATEDDATA

OBJECT = QAPERCENTMISSINGDATA
   NUM_VAL = 1
   CLASS = "3"
   VALUE = "NOT SET"
END_OBJECT = QAPERCENTMISSINGDATA

OBJECT = QAPERCENTOUTOFBOUNDSDATA
   NUM_VAL = 1
   CLASS = "3"
   VALUE = "NOT SET"
END_OBJECT = QAPERCENTOUTOFBOUNDSDATA

OBJECT = QAPERCENTCLOUDCOVER
   NUM_VAL = 1
   CLASS = "3"
   VALUE = "NOT SET"
END_OBJECT = QAPERCENTCLOUDCOVER

END_GROUP = QASTATS

END_OBJECT = MEASUREDPARAMETERCONTAINER

OBJECT = MEASUREDPARAMETERCONTAINER
   CLASS = "4"

OBJECT = PARAMETERNAME
   CLASS = "4"
   NUM_VAL = 1
   VALUE = "information4"
END_OBJECT = PARAMETERNAME

/* QAFlags */
GROUP = QAFLAGS
CLASS = "4"

OBJECT = AUTOMATICQUALITYFLAG
NUM_VAL = 1
CLASS = "4"
VALUE = "NOT SET"
END_OBJECT = AUTOMATICQUALITYFLAG

OBJECT = AUTOMATICQUALITYFLAGEXPLANATION
NUM_VAL = 1
CLASS = "4"
VALUE = "NOT SET"
END_OBJECT = AUTOMATICQUALITYFLAGEXPLANATION

END_GROUP = QAFLAGS

/* QAStats */

GROUP = QASTATS
CLASS = "4"

OBJECT = QAPERCENTINTERPOLATEDDATA
NUM_VAL = 1
CLASS = "4"
VALUE = "NOT SET"
END_OBJECT = QAPERCENTINTERPOLATEDDATA

OBJECT = QAPERCENTMISSINGDATA
NUM_VAL = 1
CLASS = "4"
VALUE = "NOT SET"
END_OBJECT = QAPERCENTMISSINGDATA

OBJECT = QAPERCENTOUTOFBORDERDATA
NUM_VAL = 1
CLASS = "4"
VALUE = "NOT SET"
END_OBJECT = QAPERCENTOUTOFBORDERDATA

OBJECT = QAPERCENTCLOUDCOVER
NUM_VAL = 1
CLASS = "4"
VALUE = "NOT SET"
END_OBJECT = QAPERCENTCLOUDCOVER

END_GROUP = QASTATS

END_OBJECT = MEASUREDPARAMETERCONTAINER

END_GROUP = MEASUREDPARAMETER

GROUP = ORBITCALCULATEDSPATIALDOMAIN

OBJECT = ORBITCALCULATEDSPATIALDOMAINCONTAINER

7-21 162-WP-004-001
CLASS = "M"

OBJECT
  CLASS = ORBITALMODELNAME
  NUM_VAL = 1
  VALUE = "NOT SET"
END_OBJECT

OBJECT
  CLASS = ORBITNUMBER
  NUM_VAL = 1
  VALUE = "NOT SET"
END_OBJECT

OBJECT
  CLASS = STARTORBITNUMBER
  NUM_VAL = 1
  VALUE = "NOT SET"
END_OBJECT

OBJECT
  CLASS = STOPORBITNUMBER
  NUM_VAL = 1
  VALUE = "NOT SET"
END_OBJECT

OBJECT
  CLASS = EQUATORCROSSINGLONGITUDE
  NUM_VAL = 1
  VALUE = "NOT SET"
END_OBJECT

OBJECT
  CLASS = EQUATORCROSSINGTIME
  NUM_VAL = 1
  VALUE = "NOT SET"
END_OBJECT

OBJECT
  CLASS = EQUATORCROSSINGDATE
  NUM_VAL = 1
  VALUE = "NOT SET"
END_OBJECT

END_OBJECT

END_GROUP

/* CollectionDescriptionClass */

GROUP = COLLECTIONDESCRIPTIONCLASS

OBJECT
  CLASS = SHORTNAME
  NUM_VAL = 1
  VALUE = "L7ORF1"
END_OBJECT

7-22 162-WP-004-001
OBJECT = VERSIONID
NUM_VAL = 1
VALUE = "1"
END_OBJECT = VERSIONID
END_GROUP = COLLECTIONDESCRIPTIONCLASS

/* SpatialDomainContainer */
GROUP = SPATIALDOMAINCONTAINER

GROUP = HORIZONTALSPATIALDOMAINCONTAINER

/* ZoneIdentifierClass */
GROUP = ZONEIDENTIFIERCLASS

OBJECT = ZONEIDENTIFIER
NUM_VAL = 1
VALUE = "NOT SET"
END_OBJECT = ZONEIDENTIFIER

END_GROUP = ZONEIDENTIFIERCLASS

/* BoundingRectangle */
GROUP = BOUNDINGRECTANGLE

OBJECT = WESTBOUNDINGCOORDINATE
NUM_VAL = 1
VALUE = 11.110000
END_OBJECT = WESTBOUNDINGCOORDINATE

OBJECT = NORTHBOUNDINGCOORDINATE
NUM_VAL = 1
VALUE = 22.220000
END_OBJECT = NORTHBOUNDINGCOORDINATE

OBJECT = EASTBOUNDINGCOORDINATE
NUM_VAL = 1
VALUE = 33.330000
END_OBJECT = EASTBOUNDINGCOORDINATE

OBJECT = SOUTHBOUNDINGCOORDINATE
NUM_VAL = 1
VALUE = 44.440000
END_OBJECT = SOUTHBOUNDINGCOORDINATE

END_GROUP = BOUNDINGRECTANGLE

END_GROUP = HORIZONTALSPATIALDOMAINCONTAINER

END_GROUP = SPATIALDOMAINCONTAINER
/* RangeDateTime */

GROUP = RANGEDATETIME

OBJECT = RANGEBEGINNINGTIME
    NUM_VAL = 1
    VALUE = "NOT SET"
END_OBJECT = RANGEBEGINNINGTIME

OBJECT = RANGEENDINGTIME
    NUM_VAL = 1
    VALUE = "NOT SET"
END_OBJECT = RANGEENDINGTIME

OBJECT = RANGEBEGINNINGDATE
    NUM_VAL = 1
    VALUE = "1997.07/30"
END_OBJECT = RANGEBEGINNINGDATE

OBJECT = RANGEENDINGDATE
    NUM_VAL = 1
    VALUE = "NOT SET"
END_OBJECT = RANGEENDINGDATE

END_GROUP = RANGEDATETIME

GROUP = ADDITIONALATTRIBUTES

OBJECT = ADDITIONALATTRIBUTESCONTAINER
    CLASS = "M"

/* AdditionalAttributes */

OBJECT = ADDITIONALATTRIBUTENAME
    CLASS = "M"
    NUM_VAL = 1
    VALUE = "NOT SET"
END_OBJECT = ADDITIONALATTRIBUTENAME

/* InformationContent */

GROUP = INFORMATIONCONTENT
    CLASS = "M"

OBJECT = PARAMETERVALUE
    NUM_VAL = 1
    CLASS = M
    VALUE = "NOT SET"
END_OBJECT = PARAMETERVALUE

END_GROUP = INFORMATIONCONTENT

END_OBJECT = ADDITIONALATTRIBUTESCONTAINER
7.5 The file filetable.temp used for example in Section 7.2

/# This file is needed for testing TIME tools. Only the Path for the files need to be changed.
# The following IDs are defined in the TOOLKIT and they SHOULD NOT be changed

10100|LogStatus|/tk/TOOLKIT_MTD/test/test_MET_HDF5/LogStatus
5000|configfile.dat|/tk/TOOLKIT_MTD/runtime/configfile.dat
10252|GetAttrtemp|/tk/TOOLKIT_MTD/test/test_MET_HDF5/GetAttrtemp
10254|MCFWrite.temp|/tk/TOOLKIT_MTD/test/test_MET_HDF5/MCFWrite.temp
10255|AsciiDump|/tk/TOOLKIT_MTD/test/test_MET_HDF5/AsciiDump
10256|temporary.MCF|/tk/TOOLKIT_MTD/test/test_MET_HDF5/temporary.MCF
10301|leapsec.dat|/tk/TOOLKIT_MTD/database/common/TD/leapsec.dat
10401|utcpole.dat|/tk/TOOLKIT_MTD/database/common/CSC/utcpole.dat
10402|earthfigure.dat|/tk/TOOLKIT_MTD/database/common/CSC/earthfigure.dat
10601|de200.eos|/tk/TOOLKIT_MTD/database/common/CBP/de200.eos
10801|sc_tags.dat|/tk/TOOLKIT_MTD/database/common/EPH/sc_tags.dat
10302|udunits.dat|/tk/TOOLKIT_MTD/database/common/CUC/udunits.dat

# Logical IDs assigned for input/output files can be changed BUT they
# should be different from the IDs assigned above.

10250|MCF_File|/tk/TOOLKIT_MTD/test/test_MET_HDF5/MET_TestData/MCF_File
10251|data_dict|/tk/TOOLKIT_MTD/test/test_MET_HDF5/MET_TestData/data_dict
5039|Swath_h5.hdf|/tk/TOOLKIT_MTD/test/test_MET_HDF5/MET_TestData/Swath_h5.hdf

# files to check POS_MET_InitNonMCF function

7-25

162-WP-004-001
A.1 Installation Procedures

A.1.1 Preliminary Step

Before installing HDFEOS, you must already have installed NCSA HDF, Version 5-1.2.0 on your host. The installation script will prompt for the paths to the HDF include and library directories. Please see the SDP Toolkit Users Guide for the ECS Project, Section 5 for instructions on installing both the Toolkit and HDF. See also: http://hdf.ncsa.uiuc.edu/ for instructions on how to access HDF libraries.

A.1.2 Unpacking the Distribution File

1) Select a location for the HDFEOS directory tree. Installing HDFEOS alone requires a disk partition with at least 25 Mb of free space.

2) Copy the file HDF-EOSv3.0.tar.Z to the target directory by typing the command:

   cp HDF-EOSv3.0.tar.Z <target-dir>

   where <target-dir> is the full pathname of your target directory.

3) Set your default directory to the target directory by typing the command:

   cd <target-dir>

4) Uncompress this file and extract the contents by typing the command:

   zcat HDF-EOSv3.0.tar.Z | tar xvf -

This will create a subdirectory of the current directory called 'hdfeos'. This is the top-level HDFEOS directory, which contains the full HDFEOS directory structure.

A.1.3 Starting the Installation Procedure

1) Set your default directory to the top-level HDFEOS directory by typing the command:

   cd hdfeos

2) Select installation options. Currently, the only options are those specific to the SGI Power Challenge platform.
On the SGI Challenge, the *default* is to build HDFEOS in 64-bit mode, which is the same as the Toolkit. The following table gives the option to specify the appropriate architecture to be built:

<table>
<thead>
<tr>
<th>binary format</th>
<th>architecture</th>
<th>&lt;install-options&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>new 32-bit</td>
<td>sgi32</td>
<td>-sgi32</td>
</tr>
<tr>
<td>64 bit</td>
<td>sgi64</td>
<td>-sgi64</td>
</tr>
</tbody>
</table>

Please note that the old-32-bit mode has been dropped as the default because it is no longer being supported by SGI, it is therefore recommended that all users migrate to new-style 32 bit or 64 bit mode.

3) Run the installation script.

Please note that the installation script for this release of HDFEOS requires user interaction. Because of this, it should NOT be run as a background task.

3.0) If you wish to generate a log of this session, use the Unix 'script' command. This command runs a sub-shell that saves all terminal output to the specified file. To log the session, type:

```bash
script <logfile-name>
```

where `<logfile-name>` is the name of the log file

3.1) To run the installation script, type the command:

```bash
bin/INSTALL-HDFEOS <install-options>
```

where `<install-options>` is the list of options determined in the previous step.

The installation script will then run. It will output various startup messages, beginning with:

```
HDFEOS installation starting at <date/time
```

3.2) Enter the full pathnames for the hdf5-1.2.0 library and include directory paths, when the script prompts for them. If there is an error in the supplied paths, the script will exit.

NOTE: If the environment variables HDFLIB and/or HDFINC are set in your shell, the script will use these for the default values. If this is not the first run of the script, the default values will be taken from the values used for the last run of the script. In either of these cases, the installation script will prompt with:

```
Current value of the HDF library directory is: <path>
```

Accept [y]/n:

```
and/or

Current value of the HDF include directory is: <path>
```

Accept [y]/n:

Make sure to type 'n' and hit return, if the defaults do not point to the correct directories. The script will then prompt for the new values.
3.3) The installation script will finish with the following message:

   HDFEOS installation ending at <date/time>

3.4) (optional - see 3.0)

If you ran the Unix 'script' command to create a log file, then type 'exit' and hit return at the command prompt. This will exit the sub-shell stated by 'script' and save the log file.

**Hint:** The log file generated by the script command may contain 'hard return' characters at the end of each line. These appear in some text editors as "^M". They can be removed with the following command:

```
sed 's/.$//' <logfile-name> > <logfile-name>.new
```

where `<logfile-name>` is the name of the log file.

A.1.4 User Account Setup

Once HDFEOS has been installed, the accounts of HDFEOS users must be set up to define environment variables needed to compile and run code with HDFEOS (see parts 2 and 3 of the Notes section, below). The type of setup depends on the user's login shell.

1A) C shell (csh) Users:

Edit the HDFEOS user's `.cshrc` file to include the following line:

```
source <HDFEOS-home-dir>/bin/$BRAND/hdfeos_env.csh
```

where `<HDFEOS-home-dir>` is the full path of the HDFEOS home directory, and `$BRAND` is an architecture-specific value for your host. Please refer to part 2 of the Notes section, below, to determine the correct value.

The script `hdfeos_env.csh` sets up all the variables discussed in parts 2 and 3 of the Notes section, below, and it adds the HDFEOS bin directory to the user path.

The environment variables will become available during all subsequent login sessions. To activate them for the current session, simply type one of the two lines listed above, at the Unix prompt.

Note regarding path setup with `hdfeos_env.csh`:

The script `hdfeos_env.csh` also makes available a variable called `hdfeos_path`. This can be added to the user's path to ensure that it accesses the directories necessary for the compilers and other utilities used to generate executable programs. It is not added to the user path by default, because in many cases it adds unnecessary complexity to the user path. To add `hdfeos_path` to the user path, modify the HDFEOS user's `.cshrc` file to include the following:

```
set my_path = ($path)
source <HDFEOS-HOME-DIR>/bin/$BRAND/hdfeos_env.csh # HDFEOS setup
set path = ($my_path $hdfeos_path) # add hdfeos_path
```
INSTEAD OF the line listed at the beginning of this step.

Note that it is the user's responsibility to set up his or her own path so that it doesn't duplicate the directories set up in hdfeos_path. Please also note that the hdfeos_path is added AFTER the user's path. This way, the user's directories will be searched first when running Unix commands.

1B) Korn shell (ksh) Users:

Edit the HDFEOS user's .profile file to include the following line:

```
. <HDFEOS-home-dir>/bin/$BRAND/hdfeos_env.ksh
```

where <HDFEOS-home-dir> is the full path of the HDFEOS home directory, and $BRAND is an architecture-specific value for your host. Please refer to part 2 of the Notes section, below, to determine the correct value.

The script hdfeos_env.ksh sets up all the variables discussed in part 2 and 3 of the Notes section, below, and it adds the HDFEOS bin directory to the user path.

The environment variables will become available during all subsequent login sessions. To activate them for the current session, simply type one of the two lines listed above, at the Unix prompt.

Note regarding path setup with hdfeos_env.ksh:

The script hdfeos_env.ksh also makes available a variable called hdfeos_path. This can be added to the user's path to ensure that it accesses the directories necessary for the compilers and other utilities used to generate executable programs. It is not added to the user path by default, because in many cases it adds unnecessary complexity to the user path. To add hdfeos_path to the user path, modify the HDFEOS user's .profile file to include the following:

```
my_path="$PATH" # save path
. <HDFEOS-HOME-DIR>/bin/$BRAND/hdfeos_env.ksh # HDFEOS setup
PATH="$my_path:$hdfeos_path" ; export PATH # add hdfeos_path
```

INSTEAD OF the line listed at the beginning of this step.

Note that it is the user's responsibility to set up his or her own path so that it doesn't duplicate the directories set up in hdfeos_path. Please also note that the hdfeos_path is added AFTER the user's path. This way, the user's directories will be searched first when running Unix commands.

1C) Bourne shell (sh) Users:

Set up the required HDFEOS environment variables by appending the contents of the file <HDFEOS-home-dir>/bin/$BRAND/hdfeos_env.ksh to the end of the HDFEOS user's .profile, where <HDFEOS-home-dir> is the full path of the HDFEOS home directory, and $BRAND is an architecture-specific value for your host. Please refer to part 2 of the Notes section, below, to determine the correct value.
The environment variables will become available during all subsequent login sessions. To activate them, log out and then log back in.

**A.1.5 File Cleanup**

Once HDFEOS has been built and tested, you can delete certain temporary files and directories to save some disk space. Note that once these files have been removed, you will need to unpack the original distribution file in order to re-do the installation.

To remove these files:

```
cd <HDFEOS-home-dir>/bin
rm -rf tmp
```
```
cd <HDFEOS-home-dir>/lib
rm -rf tmp
```

**A.1.6 Compiling and Linking with HDFEOS**

In order to compile and link programs with the HDFEOS you must access the HDFEOS include and library files. To do this be sure that your makefiles include something like the following:

```
INCLUDE = -I. -I$(HDFEOS_INC) -I$(HDFINC)
LIBRARY = -L. -L$(HDFEOS_LIB) -L$(HDFLIB)
LDLFLAGS = -lhdfeos -lGctp -lhdf -lnsl -lm (Sun platform)
LDLFLAGS = -lhdfeos -lGctp -lhdf -lm (others)
```

The environment variables HDFEOS_INC, HDFEOS_LIB, HDFINC and HDFLIB are set up by the HDFEOS environment scripts (see User Setup, above). The refer to the include and library directories for HDFEOS and HDF, respectively.

The INCLUDE macro should be included in all compilation statements. The LIBRARY and LDLFLAGS macros should be included in all link statements.

**A.2 Notes**

1) **Approved Platforms**

HDFEOS was built and tested in a multi-platform environment. The list of approved platforms, which includes information about operating system and compiler versions, may be found in the HDFEOS User's Guide and is also listed below.

<table>
<thead>
<tr>
<th>Platform</th>
<th>OS</th>
<th>Version</th>
<th>C Compiler</th>
<th>FORTRAN77</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun Sparc</td>
<td>Solaris</td>
<td>2.5.1</td>
<td>Sun C 4.0</td>
<td>Sun FORTRAN 4.0</td>
</tr>
<tr>
<td>SGI Power Challenge</td>
<td>IRIX</td>
<td>6.2</td>
<td>SGI C 7.0</td>
<td>SGI FORTRAN 7.0</td>
</tr>
</tbody>
</table>

Note: The compilers are supplied by the vendor. The SGI Power Challenge (64-bit mode) had the native SGI FORTRAN 90 7.0.
2) Architecture Type Names

To track architecture dependencies, HDFEOS defines the environment variable $BRAND. Following is a list of valid values for this variable, which is referred to throughout this document:

<table>
<thead>
<tr>
<th>$BRAND</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>sgi64</td>
<td>SGI Power Challenge (64-bit mode)</td>
</tr>
<tr>
<td>sun5</td>
<td>Sun: SunOS 5.x</td>
</tr>
</tbody>
</table>

3) Directory and File Environment Variables

In order to use the HDFEOS library and utilities, a number of environment variables MUST be set up to point to HDFEOS directories and files. These variables are automatically set up in the User Account Setup section of the installation instructions. They are listed here for reference:

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDFEOS_HOME</td>
<td>&lt;install-path&gt;/hdfeos</td>
<td>top-level directory</td>
</tr>
<tr>
<td>HDFEOS_BIN</td>
<td>$HDFEOS_HOME/bin/$BRAND</td>
<td>executable files</td>
</tr>
<tr>
<td>HDFEOS_INC</td>
<td>$HDFEOS_HOME/include</td>
<td>header files</td>
</tr>
<tr>
<td>HDFEOS_LIB</td>
<td>HDFEOS_HOME/lib/$BRAND</td>
<td>library files</td>
</tr>
<tr>
<td>HDFEOS_OBJ</td>
<td>HDFEOS_HOME/obj/$BRAND</td>
<td>object files</td>
</tr>
<tr>
<td>HDFEOS_SRC</td>
<td>$HDFEOS_HOME/src</td>
<td>source files</td>
</tr>
</tbody>
</table>

4) Other Environment Variables

In addition, the makefiles which are used to build the library require certain machine-specific environment variables. These set compilers, compilation flags and libraries, allowing a single set of makefiles to serve on multiple platforms. The User Account Setup section of the installation instructions explains how to set them up. They are listed here for reference:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>C compiler</td>
</tr>
<tr>
<td>CFLAGS</td>
<td>default C flags (optimize, ANSI)</td>
</tr>
<tr>
<td>C_CFH</td>
<td>C w/ cfortran.h callable from FORTRAN</td>
</tr>
<tr>
<td>CFHFLAGS</td>
<td>CFLAGS + C_CFH</td>
</tr>
<tr>
<td>C_F77_CFH</td>
<td>C w/ cfortran.h calling FORTRAN</td>
</tr>
<tr>
<td>C_F77_LIB</td>
<td>FORTRAN lib called by C main</td>
</tr>
<tr>
<td>F77</td>
<td>FORTRAN compiler</td>
</tr>
<tr>
<td>F77FLAGS</td>
<td>common FORTRAN flags</td>
</tr>
<tr>
<td>F77_CFH</td>
<td>FORTRAN callable from C w/ cfortran.h</td>
</tr>
<tr>
<td>F77_C_CFH</td>
<td>FORTRAN calling C w/ cfortran.h</td>
</tr>
<tr>
<td>CFH_F77</td>
<td>same as F77_C_CFH</td>
</tr>
<tr>
<td>F77_C_LIB</td>
<td>C lib called by FORTRAN main</td>
</tr>
</tbody>
</table>
5) **Tools Provided with This Release**

For a complete list of the tools provided with this release of HDFEOS, please refer to Section 7 of this document.

6) **Copyright Notice for cfortran.h**

HDFEOS functions are written in C. These C-based tools include the file cfortran.h, using it to generate machine-independent FORTRAN bindings. The cfortran.h facility includes the following notice which must accompany distributions that use it:

THIS PACKAGE, I.E. CFORTRAN.H, THIS DOCUMENT, AND THE CFORTRAN.H EXAMPLEPROGRAMS ARE PROPERTY OF THE AUTHOR WHO RESERVES ALL RIGHTS. THIS PACKAGE AND THE CODE IT PRODUCES MAY BE FREELY DISTRIBUTED WITHOUT FEES, SUBJECT TO THE FOLLOWING RESTRICTIONS:

- YOU MUST ACCOMPANY ANY COPIES OR DISTRIBUTION WITH THIS (UNALTERED) NOTICE.
- YOU MAY NOT RECEIVE MONEY FOR THE DISTRIBUTION OR FOR ITS MEDIA (E.G. TAPE, DISK, COMPUTER, PAPER.)
- YOU MAY NOT PREVENT OTHERS FROM COPYING IT FREELY.
- YOU MAY NOT DISTRIBUTE MODIFIED VERSIONS WITHOUT CLEARLY DOCUMENTING YOUR CHANGES AND NOTIFYING THE AUTHOR.
- YOU MAY NOT MISREPRESENTED THE ORIGIN OF THIS SOFTWARE, EITHER BY EXPPLICIT CLAIM OR BY OMISSION.

THE INTENT OF THE ABOVE TERMS IS TO ENSURE THAT THE CFORTRAN.H PACKAGE NOT BE USED FOR PROFIT MAKING ACTIVITIES UNLESS SOME ROYALTY ARRANGEMENT IS ENTERED INTO WITH ITS AUTHOR.

THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU Assume THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. THE AUTHOR IS NOT RESPONSIBLE FOR ANY SUPPORT OR SERVICE OF THE CFORTRAN.H PACKAGE.

Burkhard Burow
burow@vxdesy.cern.ch

A.3 **Test Drivers**

Also included with this software delivery is a tar file containing test driver programs.

These test programs are provided to aid the user in the development of software using the HDF-EOS library. The user may run the same test cases as included in this file to verify that the
software is functioning correctly. These programs were written to support the internal testing and are not an official part of the delivery. Users make use of them at their own risk. No support will be provided to the user of these programs. The tar file contains source code for a drivers in C and FORTRAN for each tool; sample output files; and input files and/or shell scripts, where applicable.

The UNIX command: “zcat HDF-EOS3.0v1.00_ TestDrivers.tar.Z | tar xvf” will create a directory called test_drivers beneath the current directory containing all these test files

**A.4 User Feedback Mechanism**

The mechanism for handling user feedback, documentation and software discrepancies, and bug reports follows:

1) The following accounts at the ECS Landover facility have been set up for user response:
   - pgstlkit@eos.hitc.com and
   - hdf eos@eos.hitc.com

2) Users will e–mail problem reports and comments to the above account. A receipt will be returned to the sender. A workoff plan for the discrepancy will be developed and status report issued once a month. Responses will be prioritized based on the severity of the problem and the available resources. Simple bug fixes will be turned around sooner, while requested functional enhancements to the Toolkit will be placed in a recommended requirements data base (RRDB) and handled more formally.

3) In order to help expedite responses, we request the following information be supplied with problem reports:
   - Name:
   - Date:
   - EOS Affiliation (DAAC, Instrument, Earth Science Data and Information System (ESDIS), etc.):
   - Phone No.:
   - Development Environment:
   - Computing Platform:
   - Operating System:
   - Compiler and Compiler Flags:
   - Tool Name:
   - Problem Description:
4) In addition to the email response mechanism, a phone answering machine is also provided. The telephone number is: 301–925–0781. Calls will be returned as soon as possible. We note, however, that email is our preferred method of responding to users.
This page intentionally left blank.
Abbreviations and Acronyms

AI&T  algorithm integration & test
AIRS  Atmospheric Infrared Sounder
API   application program interface
ASTER Advanced Spaceborne Thermal Emission and Reflection Radiometer
CCSDS Consultative Committee on Space Data Systems
CDRL  Contract Data Requirements List
CDS   CCSDS day segmented time code
CERES Clouds and Earth Radiant Energy System
CM    configuration management
COTS  commercial off–the–shelf software
CUC   constant and unit conversions
CUC   CCSDS unsegmented time code
DAAC  distributed active archive center
DBMS  database management system
DCE   distributed computing environment
DCW   Digital Chart of the World
DEM   digital elevation model
DTM   digital terrain model
ECR   Earth centered rotating
ECS   EOSDIS Core System
EDC   Earth Resources Observation Systems (EROS) Data Center
EDHS  ECS Data Handling System
EDOS  EOSDIS Data and Operations System
EOS   Earth Observing System
EOSAM EOS AM Project (morning spacecraft series)
EOSDIS Earth Observing System Data and Information System
EOSPM  EOS PM Project (afternoon spacecraft series)
ESDIS  Earth Science Data and Information System (GSFC Code 505)
FDF    flight dynamics facility
FOV    field of view
ftp    file transfer protocol
GCT    geo-coordinate transformation
GCTP   general cartographic transformation package
GD     grid
GPS    Global Positioning System
GSFC   Goddard Space Flight Center
HDF    hierarchical data format
HTIC   Hughes Information Technology Corporation
http   hypertext transport protocol
I&T    integration & test
ICD    interface control document
IDL    interactive data language
IP     Internet protocol
IWG    Investigator Working Group
JPL    Jet Propulsion Laboratory
LaRC   Langley Research Center
LIS    Lightening Imaging Sensor
M&O    maintenance and operations
MCF    metadata configuration file
MET    metadata
MODIS  Moderate-Resolution Imaging Spectroradiometer
MSFC   Marshall Space Flight Center
NASA   National Aeronautics and Space Administration
NCSA   National Center for Supercomputer Applications
netCDF network common data format
NGDC  National Geophysical Data Center
NMC  National Meteorological Center (NOAA)
ODL  object description language
PC  process control
PCF  process control file
PDPS  planning & data production system
PGE  product generation executive (formerly product generation executable)
POSIX  Portable Operating System Interface for Computer Environments
PT  point
QA  quality assurance
RDBMS  relational data base management system
RPC  remote procedure call
RRDB  recommended requirements database
SCF  Science Computing Facility
SDP  science data production
SDPF  science data processing facility
SGI  Silicon Graphics Incorporated
SMF  status message file
SMP  Symmetric Multi-Processing
SOM  Space Oblique Mercator
SPSO  Science Processing Support Office
SSM/I  Special Sensor for Microwave/Imaging
SW  swath
TAI  International Atomic Time
TBD  to be determined
TDRSS  Tracking and Data Relay Satellite System
TRMM  Tropical Rainfall Measuring Mission (joint US – Japan)
UARS  Upper Atmosphere Research Satellite
UCAR  University Corporation for Atmospheric Research
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>universal reference locator</td>
</tr>
<tr>
<td>USNO</td>
<td>United States Naval Observatory</td>
</tr>
<tr>
<td>UT</td>
<td>universal time</td>
</tr>
<tr>
<td>UTC</td>
<td>Coordinated Universal Time</td>
</tr>
<tr>
<td>UTCF</td>
<td>universal time correlation factor</td>
</tr>
<tr>
<td>UTM</td>
<td>universal transverse mercator</td>
</tr>
<tr>
<td>VPF</td>
<td>vector product format</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>