User Guide to the HDF5 High-level Library for Handling Region References and Hyperslab Selections

Version 1.0

1　Introduction

This document is a quick guide to the HDF5 High-level Region library for creating, manipulating, and querying data associated with a dataset region reference. The library[1] contains C and Fortran APIs designed to reduce a number of steps that a user has to traverse when working with the HDF5 region references. It also facilitates access to a raw data stored in an HDF5 file and pointed to by the region references.

The library was created to address the needs of the NPOESS project (http://www.ipo.noaa.gov/) and provide easy and intuitive access to the NPOESS data products, but we hope it will be useful to any application that works with the HDF5 region and object references, and hyperslab selections.

---

[1] Current distribution works on Linux/Unix platforms only; library may be ported to the Windows platforms in the future.

It is assumed that the reader is familiar with the basic HDF5 concepts [1] and can write, compile, link and run a simple HDF5 C or Fortran program, and examine the content of an HDF5 file with h5dump or HDFView tools [2].

2   Installing the HDF5 High-level Region Library

The HDF5 High-level Region library is a standalone library built on top of the HDF5 C library. To install it, follow these steps:

1) Make sure you have the HDF5 library, version 1.8.5 or later, installed on your system. For updates and installation instructions, see http://www.hdfgroup.org/HDF5/.
2) To download the source code, go to http://www.hdfgroup.org/projects/npoess/.
3) Click the "Software" tab on the left to open the FTP page.
4) Click the "source" folder to open its contents.
5) Click the hdf5_HL_REGION-1.1.0.tar file, which contains the source code, to fetch it. (Alternately, go directly to The HDF Group FTP server ftp://ftp.hdfgroup.uiuc.edu and then to the /pub/outgoing/NPOESS/source directory to access this source.)
6) Unpack the file and change directory to hdf5_HL_REGION-1.1.0.
7) Follow the installation instructions in the README.txt file to install the library under a directory /*dir* on your system.
8) Use `h5cc` or `h5fc` compiler scripts found in the `bin` directory of the HDF5 installation directory, to compile and link your application as shown

```
h5cc –I/dir/include example.c –L/dir/lib –lhdf5_hl_region
h5fc example.f90 –L/dir/lib –lhdf5_hl_region_fortran –lhdf5_hl_region
```

If the library is installed with the HDF5 library (default), then the commands are simplified to

```
h5cc  example.c –lhdf5_hl_region

h5fc example.f90 –lhdf5_hl_region_fortran –lhdf5_hl_region
```

Note: The next section gives a quick overview of HDF5 references and provides examples of their usage. Readers familiar with this topic may skip Section 3 and proceed to Section 4 to learn about the library API.

3   What is an HDF5 Reference?

There are many different ways to organize data in an HDF5 file and to express relationships between different objects stored.

An HDF5 application may use a hierarchical structure to impose relationships between data objects stored in a file, allowing a user to navigate efficiently through the data. For example, measurements with the same time stamp can be stored in the datasets belonging to the same group in a file. The application just needs to know the path to the group to access all measurements with the same time stamp.

While hierarchical structure and grouping mechanism work well for organizing objects of a "similar" nature, as in the time-stamp example above, they may not work well for organizing objects of a "dissimilar" nature: associating complex metadata with an object (such as an image and its possible

palettes), locating objects with a particular property, or locating and accessing a particular subset of data elements stored in a dataset. To accomplish this task, HDF5 provides *references* to objects and *references* to dataset regions.

3.1   HDF5 Reference to an Object

An object reference points to an HDF5 object, such as a dataset or a group, stored in the same file. The application may use an array of object references to identify a set of related HDF5 objects.

Example: associating images with palettes. HDF5 Image uses an array of object references to associate different palettes with the same image. (For more information, see section 1.2 "Image Attributes" in *HDF5 Image and Palette Specification* [3].) Figure 1 shows an HDF5 file containing three images and four palettes. The image "image8bit" has four palettes associated with it.
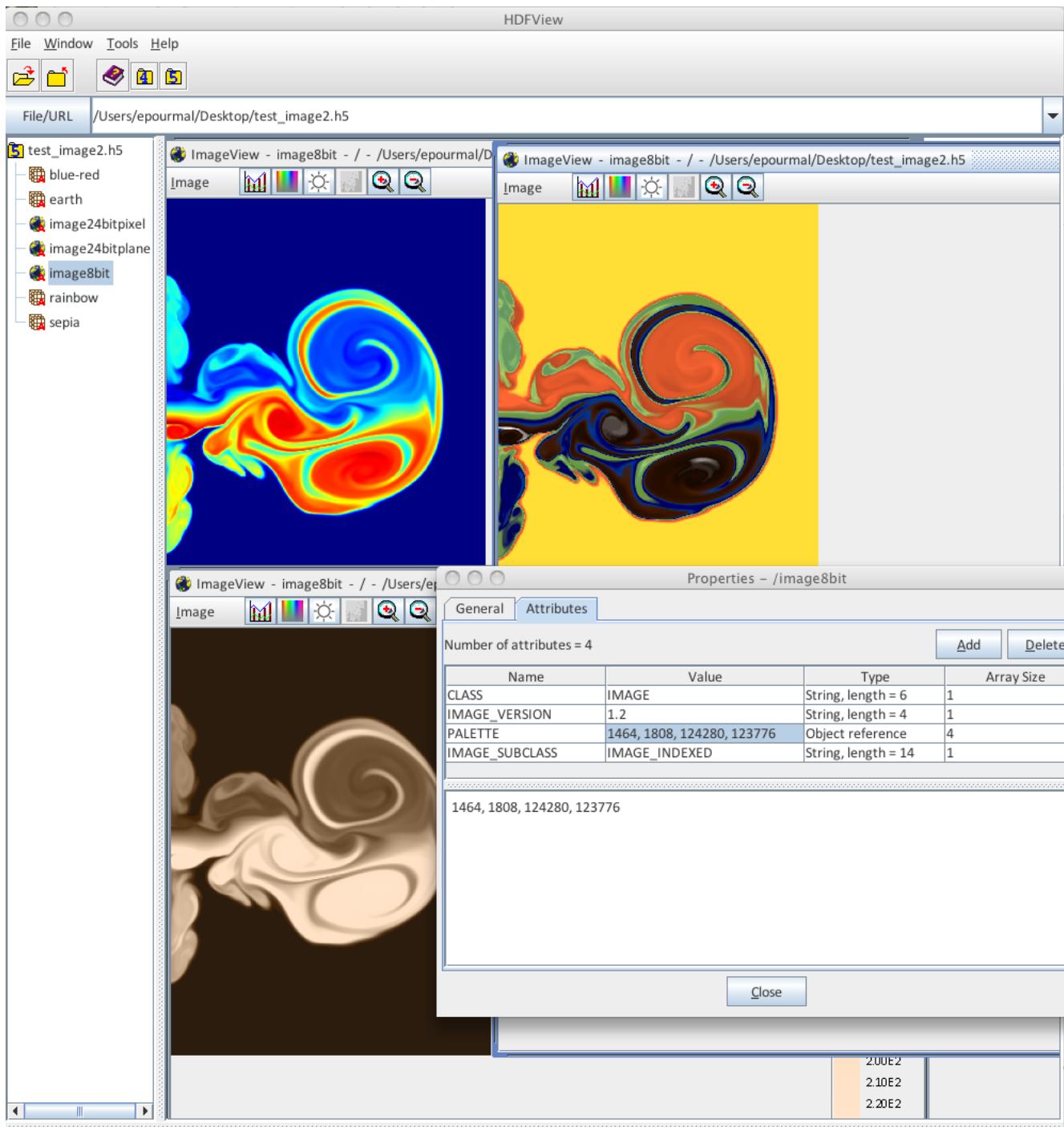
Figure 1. In HDFView, the image "image8bit" is shown using three different palettes. The Palette attribute of the image is an array of object references pointing to the palettes (datasets "rainbow", "sepia", "earth" and "blue-red" in the same file).

The image's attribute "PALETTE" (shown highlighted in the Properties Window) is an array of object references with the following numerical values: 1464, 1808, 124280, and 123776. Each number can be interpreted by the HDF5 library to access the datasets "rainbow", "sepia", "earth" and "blu-red", respectively, in the same file. HDFView uses palettes to display the dataset "image8bit".

Example: dimension scales. Internal implementation of dimension scales in HDF5 relies on object references to associate a dataset with its dimensions scales and to share them with other datasets in a file [4]. In the h5dump output example below, the two-dimensional dataset "Mydata" has a dimension scale associated with the each of the dimensions. The first dimension of size 4 has a dimension scale Xaxis with the values of the scale stored in the dataset Xaxis; the second dimension of size 3 has a dimension scale Yaxis with the values of the scale stored in the dataset Yaxis. The values of the corresponding object references are 1400 and 1672, respectively.

```
HDF5 "dimscale.h5" {
GROUP "/" {
   DATASET "Mydata" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 3, 4 ) / ( 3, 4 ) }
      DATA {…
            }
      ATTRIBUTE "DIMENSION_LIST" {
         DATATYPE  H5T_VLEN { H5T_REFERENCE}
         DATASPACE  SIMPLE { ( 2 ) / ( 2 ) }
         DATA {
         (0): (DATASET 1400 /Yaxis ), (DATASET 1672 /Xaxis )
         }
      }
   }
   DATASET "Xaxis" {
      …
      DATA {
      (0): 10, 20, 50, 100
      }
      …
   }
   DATASET "Yaxis" {
      …
      DATA {
      (0): 10, 20, 30
      }
      …
   }
}
}
```

3.2   HDF5 Reference to a Dataset Region

A region reference points to a dataset and a region within that dataset. When stored in a dataset, an array of region references can provide a unified view of the data stored in the different datasets in a file.

In general, region references are useful for directly accessing a portion of a dataset. Notably, region references play an important role in large datasets by providing a convenient and efficient way to point to data of interest.

Figure 2 illustrates the concept of region references. A user can create a file (*FileA.h5*) having a group (*Group_1D*) containing data values stored in a one-dimensional array (*DS1*), a group (*Group_2D*) containing data in a two-dimensional array (*DS2*), and a group (*Group_3D*) containing data in a three-dimensional array (*DS3*).

In order to quickly and efficiently access data within a dataset, an array of region references is created (*R1*). Each element in the region reference array points to a different selection of elements in the datasets. The first element in R1 points to a region in the dataset DS2, the second element in R1 points to a set of points in DS1, and so forth.
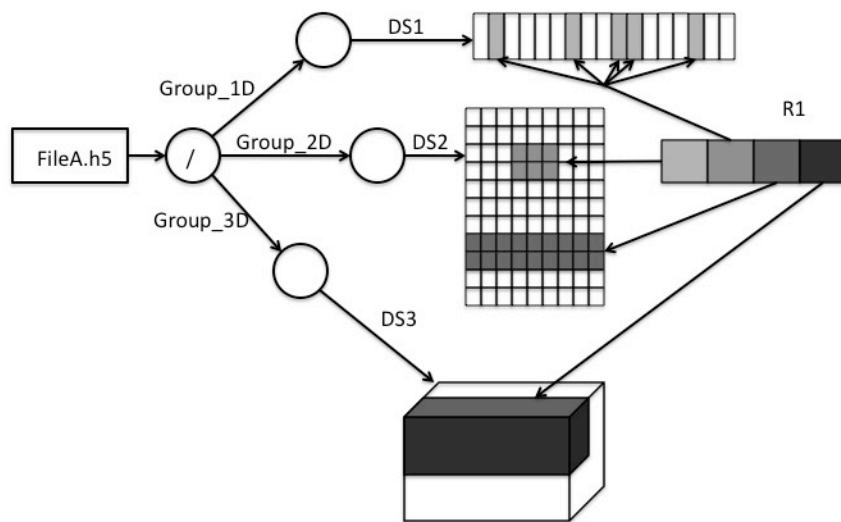
Figure 2. Elements of region reference array R1 point to regions in the datasets in an HDF5 file

The same file is shown in Figure 3 as displayed by HDFView. Dataset R1 has four elements of the region reference datatype. Each of the "TableView" windows on the right has one of the four elements selected. As one can see, an element of the region reference type contains two pieces of information: the first one, shown in the form 0:XXXX, indicates the dataset; the second one describes the selected region in that dataset. The region can be a set of points as is the case for the first element, described as a list of the points' coordinates. Or the region can be a rectangular subset, described as a pair of "lower-left"-"upper-right" coordinates, as is the case for the second, third and forth elements.

The HDF Group
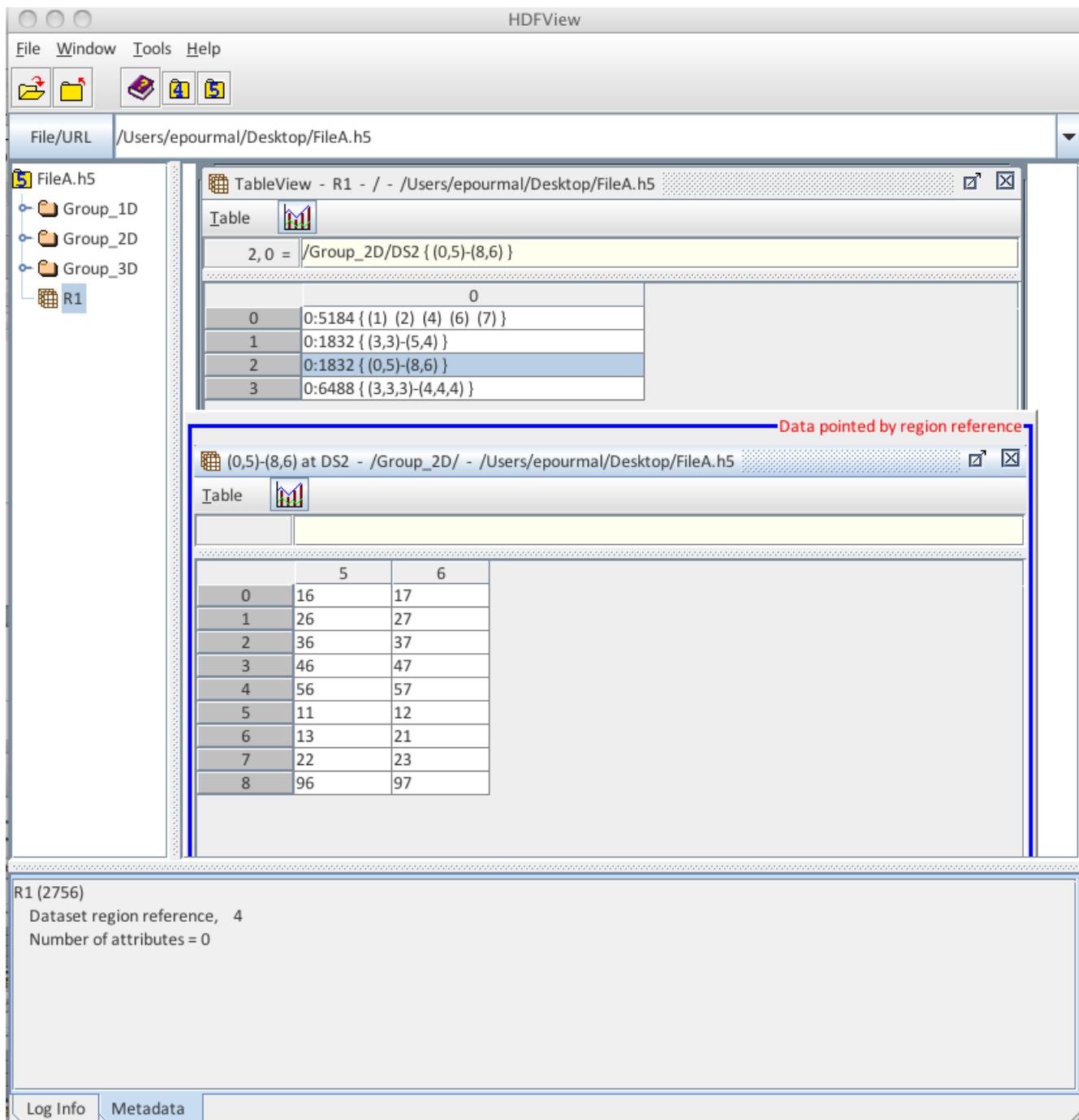
Figure 3. Each element of the dataset R1 is a region reference. Small pane window (in yellow) shows the full path to a dataset along with the selected elements within the dataset for each of the four elements of R1.

One can use the h5dump utility to see the content of the HDF5 file. Below is an output of the same file (for brevity's sake, the actual data in datasets DS1, DS2, DS2a are not included). For each element

The HDF Group

of the dataset R1, h5dump prints the full path to the dataset within the file along with the coordinates of the selected region.

```
HDF5 "FileA.h5" {
GROUP "/" {
   GROUP "Group_1D" {
      DATASET "DS1" {…
}
   GROUP "Group_2D" {
      DATASET "DS2" {…
 }
GROUP "Group_3D" {
      DATASET "DS3" {…
}
DATASET "R1" {
      DATATYPE  H5T_REFERENCE
      DATASPACE  SIMPLE { ( 4 ) / ( 4 ) }
      DATA {
(0):          DATASET /Group_1D/DS1 {(1), (2), (4), (6), (7)},
(1):          DATASET /Group_2D/DS2 {(3,3)-(5,4)},
(2):          DATASET /Group_2D/DS2 {(0,5)-(8,6)},
(3):          DATASET /Group_3D/DS3 {(3,3,3)-(4,4,4)}
      }
   }
}
}
```

3.3   Example of the NPOESS File

Figure 4 shows the content of an HDF5 file with the geo-location information for the NPP VIIRS EDR (Environmental Data Record) data products displayed with HDFView.

On the root level there are two groups */All_Data* and */Data_Products*.

The group */All_Data/VIIRS_MOD_GTM_EDR_Geo_All* contains miscellaneous integer and floating-point datasets with raw data that comprise the geo-location information for the VIIRS data products. The group */Data_Products/VIIRS_MOD_GTM_EDR_Geo* contains two datasets: the dataset *VIIRS-MOD-GTM-EDR-Geo_Aggr* with the HDF5 object references to the datasets stored under the */All_Data* group, while the dataset *VIIRS-MOD-GTM-EDR-Geo_Gan_0* contains HDF5 region references pointing to the regions in those dataset. Object and region references are used here to describe VIIRS EDR product.

Windows on the right show data for the datasets under the */Data_Products* group. The highlighted element in the *VIIRS-MOD-GTM-EDR-GEO_ALL* dataset with the value 10136 is an object reference to the dataset "Longitude". The highlighted element in the *VIIRS-MOD-GTM-EDR-GEO_Gran_0* dataset with the value 0:10136 {(0,0)-(770,4120)} is a region reference to the dataset "Longitude" and a selection within it.

In section 4 we will show how C and Fortran functions from the HDF5 High-level Region Library can be used to access and manage data pointed by the HDF5 region references.
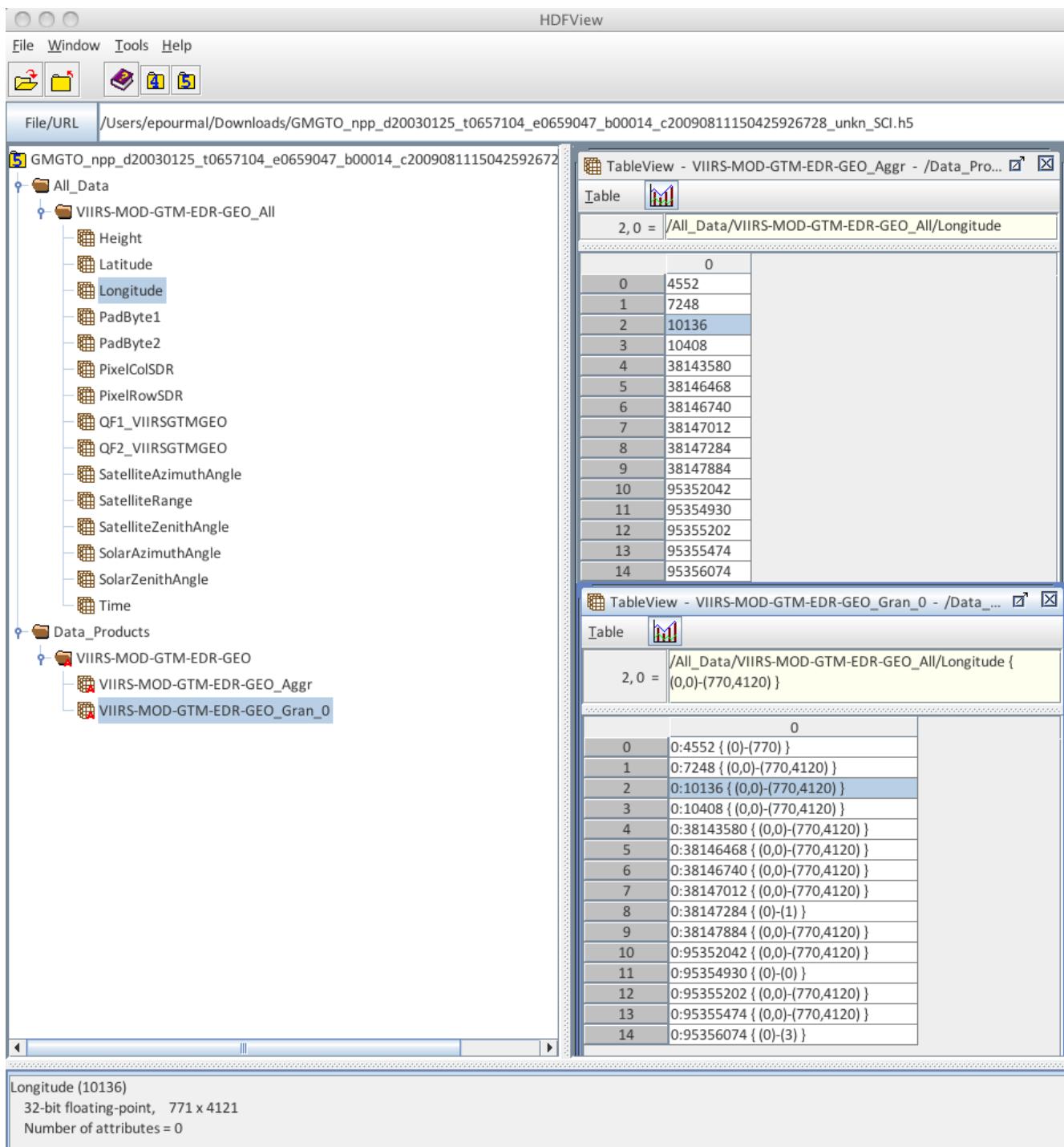
Figure 4. An example of the NPP file. Windows on the right show data for the datasets under "Data_Products" group. The highlighted element in the VIIRS-MOD-GTM-EDR-GEO_All dataset with the value 10136 is an object reference to the dataset "Longitude". The highlighted element in the VIIRS-MOD-GTM-EDR-GEO_Gran_0 dataset with the value 0:10136 {(0,0)-(770,4120)} is a region reference to the dataset "Longitude" and a selection within it.

The HDF Group

4   Library APIs and Examples

This section summarizes C functions and Fortran subroutines in the HDF5 High-level Region library. Subsections below explain how to use the functions, h5dump and HDFView tools to get information about the data associated with the references and how to manipulate such data. For function signatures, see the HDF5 High-level Region library reference manual *HDF5 High-level Functions for Region References, Hyperslabs, and Bit-fields* [5]. Examples used in this section can be seen and downloaded from http://www.hdfgroup.uiuc.edu/ftp/pub/outgoing/NPOESS/Misc-docs/HL-Tools-UsersGuide/

Table 1. HDF5 High-level Region Library Functions

API Name (C/Fortran)	Description
H5LRget_region_info / h5lrget_region_info_f	Queries information about the data pointed to by a region reference, such as information about the dataset and a description of the referenced region within the dataset.
H5LRread_region / h5lrread_region_f	Reads data pointed to by the region reference.
H5LRcreate_region_references/ H5lrcreate_region_references_f	Creates an array of region references using an array of paths to datasets and an array of corresponding hyperslab descriptions.
H5LRmake_dataset / h5lrmake_dataset_f	Creates and writes a dataset containing a list of region references.
H5LRcopy_region / h5lrcopy_region_f	Copies data from a referenced region to a region in a destination dataset.
H5LRcopy_reference / h5lrcopy_reference_f	Copies data from the specified dataset to a new location and creates a reference to it.
H5LRcreate_ref_to_all/ h5lrcreate_ref_to_all_f	Creates a dataset with the region references to the data to all datasets located under a specified group in a file, or creates a dataset with object references to all objects (groups or datasets) located under a specified group in a file.
H5LTread_region / h5ltread_region_f	Reads selected data to an application buffer.
H5LTcopy_region / h5ltcopy_region_f	Copies data from a specified region in a source dataset to a specified region in a destination dataset.
H5LTread_bitfield_value	Retrieves the values of quality flags for each element to the application-provided buffer.

The HDF Group

4.1   Programming with the HDF5 High-level Region Library

To use HDF5 High-level Region C functions, an application has to include the public header file h5hl_region.h. Since the functions are just the wrappers around the HDF5 C functions, they can be used along with the HDF5 C functions in the same application.

The HDF5 High-level Library Fortran subroutines are the wrappers on top of the HDF5 High-level Region Library C functions. They use the Fortran2003 standardized mechanism for interoperating with C. Therefore, an application that uses the Fortran subroutines is required to use the ISO_C_BINDING module along with the H5HL_REGION module. As with the C functions, High-level Fortran subroutines can be used along with the HDF5 Fortran subroutines in the same application.

The sections below discuss how to access and manipulate data in HDF5 files using functions from the library.

4.2   Reading data that a region reference points to

This section explains how to get data that a region reference points to by using HDF5 tools and functions from the HDF5 High-level Region Library. In the first example, HDFView is used to display and export region-referenced data to a text file. In the second example, the h5dump utility is used to display and export data to a binary file. In the last example, the H5LRget_region_info and H5LRread_region functions are used to find information and read the data.

4.2.1   Using HDFView to read data that a region reference points to

Figure 4 in Section 3.3 shows the `VIIRS-MOD-GTM-EDR-GEO_Gran_0` dataset with region references in the low TableView window. To see data that the third element with the value `0:10136 {(0,0)-(770,4120)}` points to, click the highlighted element. HDView opens another window with the "Data pointed by region reference" label as shown in Figure 5. Only a specified selection is displayed, i.e., hyperslab with the coordinates `(0,0) – (770,4120)`. To save data to a file, click "Table" and then the "Export Data from File" tab as shown in Figure 6; HDFview saves data in a text file with a specified name. Currently HDFView does not support exporting to a binary file; the feature may be added in the future.

HDFView does not show the dimensionality and the datatype of the data pointed to by a region reference. To find this information, one has to open the properties of a dataset; in our example, open the Longitude dataset in HDFView to see the information. As we will see in our next example, h5dump does provide this kind of information.
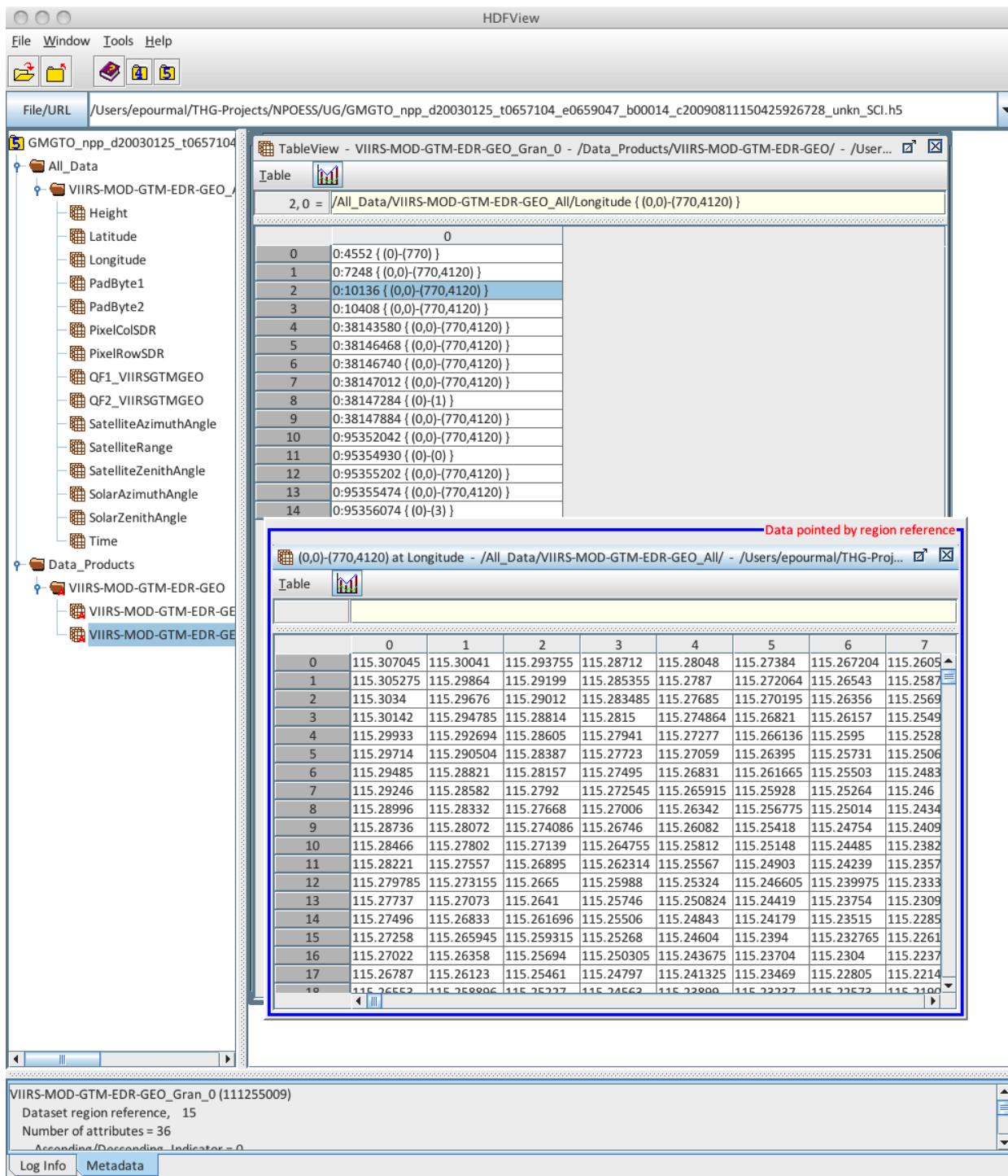
Figure 5. The top window displays data in the "Longitude" dataset, which is pointed to by a region reference stored in the third element of the VIIRS-MOD-GTM-EDR-GEO_Gran_0 dataset.
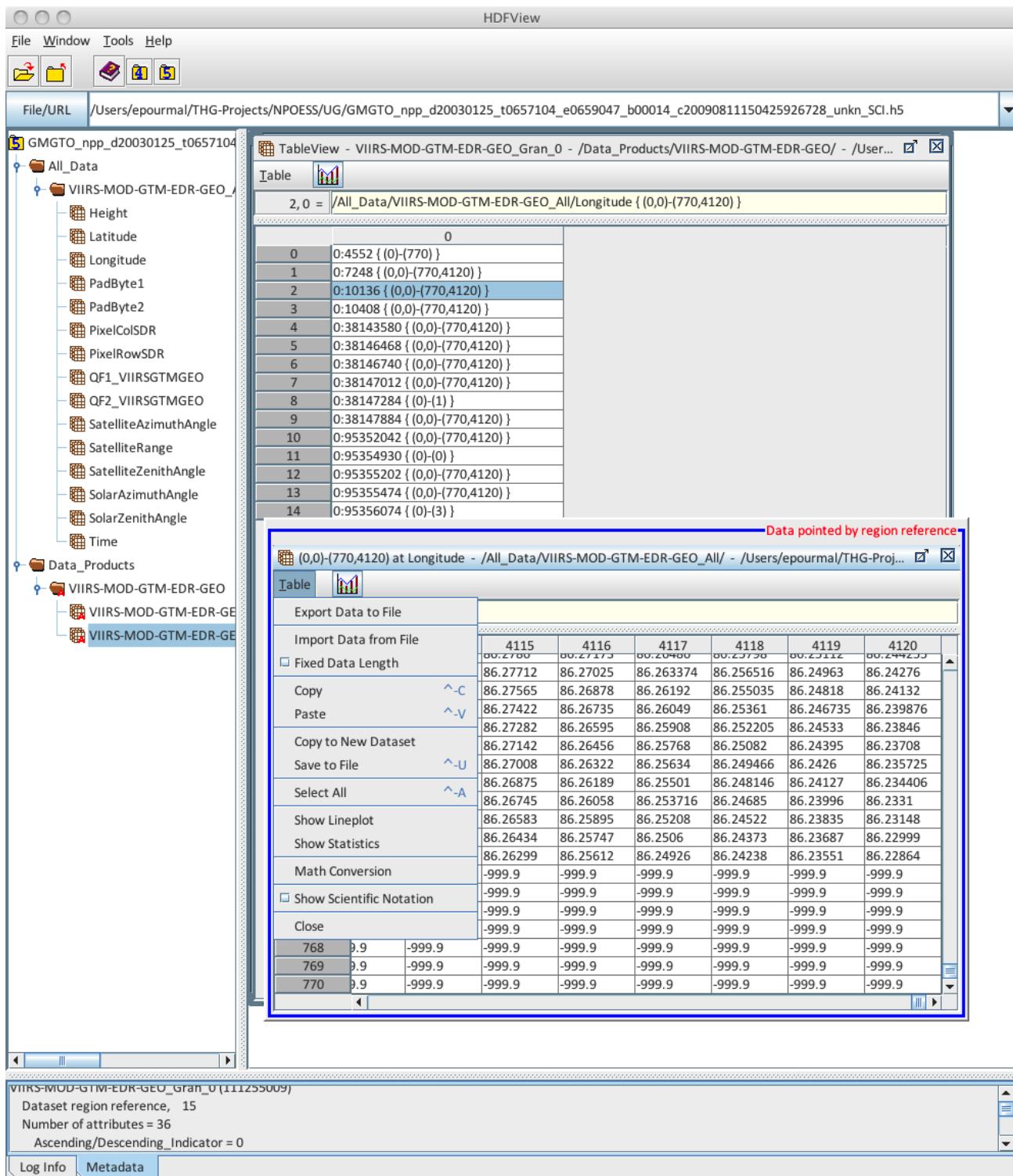
Figure 6. Data shown in the "Data pointed by region references" window can be saved in a text file by using the "Table" and then the "Export Data to File" tabs.

4.2.2 Using h5dump to read data that a region reference points to

As mentioned above, h5dump can display and export data pointed to by a region reference. Several flags such as "-d", "-s", "-R" and "-b" should be used as shown in the examples below.

4.2.2.1 Displaying a dataset with region references

The "-d" flag displays a specified dataset. When used on a dataset with region references, h5dump shows the full path to a dataset and a selection within that dataset that the region reference points to, as shown below for VIIRS-MOD-GTM-EDR-GEO_Gran_0.

```
h5dump -d /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0 <name>.h5²
HDF5 "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5" {
DATASET "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0" {
   DATATYPE  H5T_REFERENCE
   DATASPACE  SIMPLE { ( 15 ) / ( H5S_UNLIMITED ) }
   DATA {
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/Time {(0)-(770)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/Latitude {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/Longitude {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SolarZenithAngle {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SolarAzimuthAngle {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/Height {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/PadByte1 {(0)-(1)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/QF1_VIIRSGTMGEO {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/QF2_VIIRSGTMGEO {(0)-(0)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/PixelRowSDR {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/PixelColSDR {(0,0)-(770,4120)},
      DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/PadByte2 {(0)-(3)}
   }
```

4.2.2.2 Displaying an element of a dataset with region references

To display just one element – for example, the third element of the dataset – one can use the "-s" flag as shown in the next example. The "-s" flag can be used to specify the coordinates of the element in a dataset array. The coordinates are 0-based; to specify the third element, use "-s" 2 as shown below.

---

[2] <name> is used for GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5 for readability purposes only

```
h5dump -d /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0 -s 2 <name>.h5


HDF5 "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5" {
DATASET "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0" {
   DATATYPE  H5T_REFERENCE
   DATASPACE  SIMPLE { ( 15 ) / ( H5S_UNLIMITED ) }
   SUBSET {
      START ( 2 );
      STRIDE ( 1 );
      COUNT ( 1 );
      BLOCK ( 1 );
      DATA {
         DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/Longitude {(0,0)-(770,4120)}
      }
   }
 }
….
```

4.2.2.3   *Displaying data that a region reference points to*

To display the data pointed to by the third element of the VIIRS-MOD-EDR-GEO_Gran_0 dataset and its datatype, use the "-R" flag.

The h5dump utility displays the description of the selection (REGION_TYPE BLOCK (0,0)-(770,4120)), its datatype (H5T_IEEE_F32BE) and the data values.

```
h5dump -d /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0 -s 2 -R <name>.h5

HDF5 "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5" {

DATASET "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0" {

    DATATYPE  H5T_REFERENCE { H5T_STD_REF_DSETREG }

    DATASPACE  SIMPLE { ( 15 ) / ( H5S_UNLIMITED ) }

    SUBSET {

       START ( 2 );

       STRIDE ( 1 );

       COUNT ( 1 );

       BLOCK ( 1 );

       DATA {

       (2): DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/Longitude {

          (2): REGION_TYPE BLOCK  (0,0)-(770,4120)

          (2): DATATYPE  H5T_IEEE_F32BE

          (2): DATASPACE  SIMPLE { ( 771, 4121 ) / ( H5S_UNLIMITED, H5S_UNLIMITED ) }

          (2): DATA {

             (0,0): 115.307, 115.3, 115.294, 115.287, 115.28, 115.274,

             (0,6): 115.267, 115.261, 115.254, 115.247, 115.241, 115.234,

             (0,12): 115.227, 115.221, 115.214, 115.207, 115.201, 115.194,

             (0,18): 115.188, 115.181, 115.174, 115.168, 115.161, 115.154,
```

…….

4.2.2.4 *Saving data that a region reference points to*[3]

To save data in a file, use "-y" to disable the printing of indices and "-o" to specify the output file:

```
h5dump -d /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0 -s 2 -R -y -o ref.txt
<name>.h5
```

The ref.txt file contains data values

```
115.307, 115.3, 115.294, 115.287, 115.28, 115.274, 115.267,
115.261, 115.254, 115.247, 115.241, 115.234, 115.227, 115.221,
115.214, 115.207, 115.201, 115.194, 115.188, 115.181, 115.174,
115.168, 115.161, 115.154, 115.148, 115.141, 115.134, 115.128,
115.121, 115.114, 115.108, 115.101, 115.095, 115.088, 115.081,
```

…..

To export data into a binary file, use the additional "-b" option followed by "LE" or "BE" to specify the binary format. The following command extracts data pointed to by the third element of the VIIRS-MOD-EDR-GEO_Gran_0 dataset, converts it to the little-endian format, and saves it into the ref.bin file.

```
h5dump -d /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0 -s 2 -R -b LE -o ref.bin
<name>.h5
```

4.2.3 Reading data with H5LRget_region_info and H5LRread_region

The HDF5 High-level Region library provides two functions, H5LRget_region_info and H5LRead_region, to facilitate access to the data that the region reference points to.

H5LRget_region_info takes as an input a region reference and retrieves information about the dataset it points to along with the information about the selection within that dataset. Information returned by this function includes:

- The full path to a dataset the region reference points to (e.g., the reference points to the dataset /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/Longitude)
- The description of the hyperslab selection the region reference points to (e.g., a list of corners (0,0) - (770,4120))
- The file datatype of the referenced data (e.g., H5T_IEEE_F32BE)
- The rank of the dataset (e.g., 2 if the dataset the reference points to is two dimensional)
- The type of selection (e.g., H5S_SEL_POINTS for the point selection and H5S_SEL_HYPERSLABS for the hyperslab selection)

Using this information, one can find the number of elements in the selection within a dataset and allocate an appropriate buffer to read the data pointed to by a region reference using the H5LRread_region function (see Figure 7).

The second function H5LRread_region uses an element of a region reference to return the following:

- The number of elements in the selected region

---

[3] This feature will be available in HDF5 1.8.6

- Data into a buffer allocated by the application

Dataset with selected region
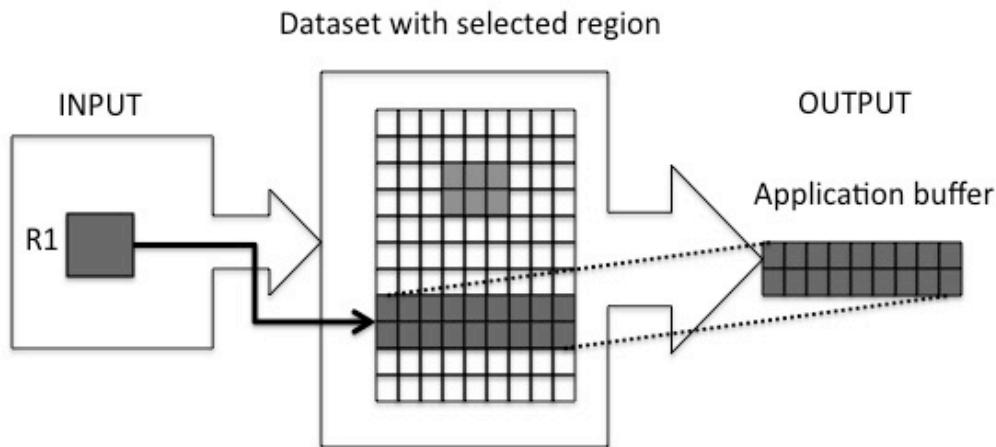
Figure 7. `H5LRread_region` retrieves data that a region reference points to and moves it to a buffer provided by an application.

The next two sections discuss both functions in greater detail.

4.2.3.1 *Exploring properties of a dataset with H5LRget_region_info*

The excerpt below from Program 1 in the Appendix shows, by using the `H5LRget_region_info` function, how to get information about the data that a region reference points to, data that is similar to the information displayed by h5dump in section 4.2.2.3. The program opens the GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5 file and the dataset `/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0` with the region references. H5Dread reads region references stored in the dataset into the buffer `ref`. Then H5LRget_region_info retrieves the path to a dataset `ref[2]` points to, its datatype and coordinates of the hyperslabs that comprise the selection within the dataset.

```
#define filename "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"
#define dsetname "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0"
……
char name[1024];
…
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);
    /*
     * Open dataset and read the dataset with the region references.
     * We made an assumption that the size of the dataset is known.
     */
    dset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);
    status = H5Dread(dset_id, H5T_STD_REF_DSETREG, H5S_ALL, H5S_ALL, H5P_DEFAULT,ref);

    /*
     * Get region reference information such as a name of the dataset the region reference
     * point to, number of contiguous blocks in the selection (should be 1) and
     * the hyperslab coordinates (0,0) - (770,4120)
```

```
     */
   status = H5LRget_region_info(file_id,
                                (const hdset_reg_ref_t*)ref[2],
                                &name_length,
                                name,
                                NULL,
                                &dtype,
                                NULL,
                                &numelem,
                                buf);
```

The following information was retrieved by the H5LRget_region_info function:

```
 Third element of the array with the region references points to  /All_Data/VIIRS-MOD-GTM-EDR-
GEO_All/Longitude
 Length of the string above is 46
 Region's datatype is H5T_IEEE_F32BE
 Number of blocks in the region is 1
 Block's coordinates are (0,0) - (770,4120)
```

Since length of the dataset name, type of the selection and number of the coordinate blocks or selected elements are generally unknown, this information must be found before allocating the read buffers accordingly as shown below. Notice the difference for allocating buffers for hyperslab and point selections.

```
 status = H5LRget_region_info(file_id,
                              (const hdset_reg_ref_t*)ref[2],
                              &name_length,
                              &rank,
                              NULL,
                              &dtype,
                              &sel_type,
                              &numelem,
                              NULL);
/* Allocate name */
name = (char *) malloc (name_length);

/* If it is a hyperslab selection buffer is twice bigger */
if (sel_type == H5S_SEL_HYPESLAB)
buf = (hsize_t *) malloc ( numelem * rank * sizeof(hsize_t) * 2 );

if (sel_type == H5S_SEL_POINTS)
buf = (hsize_t *) malloc ( numelem * rank * sizeof(hsize_t));
```

4.2.3.2   Reading data that a region reference points to using H5LRread_region

Using datatype and block coordinates retrieved by the H5LRget_region_info function, we now can read data that a region reference points to. The excerpt from Program 1 in the Appendix allocates a buffer of an appropriate size and type, and reads data that a region reference points to, which is stored in ref[2] using the call to H5LRread_region.

```
   float *rdata;
   ….
   /*
```

```
    *  We  will  read  data  to  the  floating-point  buffer;  using  information  provided  by
H5LRget_region_info
    * allocate the buffer to read data in.
    */
    rdims[0] = buf[2] - buf[0] + 1;
    rdims[1] = buf[3] - buf[1] + 1;
    rdata = (float *) malloc (rdims[0] * rdims[1] * sizeof(float));

    /*
     * Read data pointed by the third region reference into a buffer and display the first six
elements.
     */
    status = H5LRread_region(file_id,(const hdset_reg_ref_t*)ref[2], H5T_NATIVE_FLOAT,  &rnumelem,
rdata);
```

We used block coordinates to calculate the number of elements to be read and allocated a buffer for reading. We assumed that we read floating-point data and therefore specified H5T_NATIVE_FLOAT to describe the rdata buffer to the HDF5 library.

Very often the type of the data to read is not known and has to be discovered by the application. As shown in the example in section 4.2.3.1.1, the H5LRget_region_info function returns the datatype of the region data as it is stored in the file, via datatype handle dtype. The HDF5 library provides functions H5Tget_native_type and H5Tget_size to find the appropriate datatype and datatype size to be used to read data back by H5TLRread_region, as shown in a code snippet below. For the complete example, see Program 2 in the Appendix.

In this example, we used the number of elements rnumelem returned by the H5LRget_region_info function and the size of the datatype to allocate a buffer of the appropriate size.

```
    char *rdata;
    /*
    * Get datatype of the data the region reference points to.
    */
    status = H5LRget_region_info(file_id, (const hdset_reg_ref_t*)ref[2], NULL, NULL, NULL, &dtype,
                                 NULL, NULL, NULL);

    /*
     * Find the corresponding type in memory and its size.
     */
    mtype = H5Tget_native_type(dtype, H5T_DIR_ASCEND);
    msize = H5Tget_size(mtype);

    /*
     * Find number of elements in the region to read.
     */
    status = H5LRread_region(file_id,(const hdset_reg_ref_t*)ref[2], mtype, &rnumelem, NULL);

    /*
     * Allocate buffer to read data in.
     */
    rdata = (char *) malloc (rnumelem * msize);
    status = H5LRread_region(file_id,(const hdset_reg_ref_t*)ref[2], mtype, NULL, rdata);
```

4.3   Creating a dataset with region and object references

In this section we will explore the functions to create datasets with region and object references using high-level functions.

4.3.1   Creating a dataset with region references

The current process of creating a dataset with the region references is very tedious: a region reference element is created one at a time using HDF5 library information such as dataset and dataspace identifiers and then is stored in an application buffer. After all references are created, the buffer is written to the dataset. The `H5LRcreate_region_references` function allows the creation of arrays of references to be done in one step. It uses a list of paths to datasets and a list of the simple hyperslab description in the corresponding datasets to create an array of region references. The simple hyperslab description is a list of corner coordinates for each **simple** hyperslab in the corresponding dataset (i.e., this function does not support point selections and arbitrary hyperslab selections). Simple hyperslabs may have different dimensionality. The function discovers the rank of each of the listed datasets and uses this information to interpret the array of corner coordinates.

A snippet from Program 3 in the Appendix shows how the `H5LRcreate_region_references` and `H5LTmake_dataset` functions are used to create a dataset with the region references. This example takes hyperslabs with the coordinates (3,51) – (7, 53) in the datatsets `SatelliteZenithAngle`, `SatelliteAzimuthAngle` and `SatelliteRange` in the group `/All_Data/VIIRS-MOD-EDR-GEO_All` in the file, creates an array of references, and saves it in the dataset Satellite under the `/All_Products` group.

```
hsize_t block_coord[12] = {3, 51, 7, 53, 3, 51, 7, 53, 3, 51, 7, 53};
….
path[0]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle";
path[1]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle";
path[2]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange";

/*
 * Create three region references pointing to hyperslabs with block coordinates (3,51)-(7,53) in
 * SatelliteZenithAngle, SatelliteAzimuthAngle and SatelliteRange datasets.
 */
 status = H5LRcreate_region_references(file_id, 3, path, block_coord, ref);

 /*
  * Create a dataset with the region references.
  */
 status = H5LTmake_dataset(file_id, "/Data_Products/Satellite", 1, dims, H5T_STD_REF_DSETREG,
                           (const hdset_reg_ref_t *)ref);
```

The following output results from using the h5dump command on the newly created array of references:

```
h5dump -R -d /Data_Products/Satellite <name>.h5
HDF5 "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5" {
DATASET "/Data_Products/Satellite" {
   DATATYPE  H5T_REFERENCE
   DATASPACE  SIMPLE { ( 3 ) / ( 3 ) }
   DATA {
```

The HDF Group

```
        DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle {(3,51)-(7,53)},
        DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle {(3,51)-(7,53)},
        DATASET /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange {(3,51)-(7,53)}
    }
}
```

To create a dataset with the region references to datasets located under a group, use the H5LRcreate_ref_to_all function. The function iterates over all datasets under a specified group; for each dataset found, it selects the whole dataset and creates a region reference; after all references are created it writes an array of region references to a specified dataset. The same function can be used to create an array of references to objects located under a group, as shown in the next section.

4.3.2   Creating a dataset with references to datasets region or objects found under a group

Using the H5LRcreate_ref_to_all function one can create a dataset with the region references or object references. An NPP example file <name>.h5 contains datasets under the group /All_Data/VIIRS-MOD-EDR-GEO_All. The following example shows how datasets with the dataset region or object references under the /Data_Products/VIIRS-MOD-GTM_EDR-GEO group can be created in one step (Program 4)

C:
```
/*
 * This example shows how to create datasets with region and object references to
 * datasets located under the group /Data_Products/VIIRS-MOD-GTM-EDR-GEO
 *   Main illustrative functions: H5LRcreate_ref_to_all
 */

#include <stdlib.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "H5LTpublic.h"
#include "h5hl_region.h"

#define GROUP "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All"
#define Dataset_Aggr "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Aggr"
#define Dataset_Gran_0 "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0"

#define filename "GMGTO_npp_raw.h5"

int main(void)
{
    hid_t file_id;  /* file identifier */
    herr_t status;
    /*
     * Open the NPP file.
     */
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);

    /*
     * Create a dataset with region references to all the datasets located under
     * /All_Data/VIIRS-MOD-GTM-EDR-GEO_All
     * Store the region references in the /Data_Products/VIIRS-MOD-GTM-EDR-
GEO_Gan_0
     * dataset.
     */
    status = H5LRcreate_ref_to_all(file_id, GROUP, Dataset_Gran_0, H5_INDEX_NAME, H5_ITER_INC,
                            H5R_DATASET_REGION);
```

```
    /*
     * Create a dataset with object references to all the datasets located under
     * /All_Data/VIIRS-MOD-GTM-EDR-GEO_All
     * Store the region references in the /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-
GEO_Aggr
     * dataset.
     */
    status = H5LRcreate_ref_to_all(file_id, GROUP, Dataset_Aggr, H5_INDEX_NAME, H5_ITER_INC,
H5R_OBJECT);
    status = H5Fclose(file_id);

}
```

FORTRAN:
```
!
! This example shows how to create datasets with region and object references to
! datasets located under the group /Data_Products/VIIRS-MOD-GTM-EDR-GEO
!   Main illustrative functions: H5LRcreate_ref_to_all_f
!
PROGRAM main

  USE ISO_C_BINDING
  USE HDF5 ! module of HDF5 library
  USE H5LT ! module of H5LT
  USE H5HL_REGION

  IMPLICIT NONE

  CHARACTER(LEN=78), PARAMETER :: filename = "GMGTO_npp_raw.h5"
  CHARACTER(LEN=65), PARAMETER :: Dataset_Gran_0 = "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-
GTM-EDR-GEO_Gran_0"
  CHARACTER(LEN=35), PARAMETER :: group = "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All"
  CHARACTER(LEN=63), PARAMETER :: Dataset_Aggr = "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-
GTM-EDR-GEO_Aggr"

  INTEGER(hid_t) :: file_id  ! file identifier
  INTEGER :: status
  !
  !   Initialize FORTRAN interface.
  !
  CALL H5open_f(status)
  !
  ! Open the NPP file.
  !
  CALL H5Fopen_f(filename, H5F_ACC_RDWR_F, file_id, status)
  !
  !   Create a dataset with region references to all the datasets located under /All_Data/VIIRS-MOD-
GTM-EDR-GEO_All
  !    Store the region references in the /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-
GEO_Gan_0 dataset.
  !
  CALL H5LRcreate_ref_to_all_f(file_id, group, Dataset_Gran_0, H5_INDEX_NAME_F, H5_ITER_INC_F,
H5R_DATASET_REGION_F, status)
  !
  !   Create a dataset with object references to all the datasets located under /All_Data/VIIRS-MOD-
GTM-EDR-GEO_All
  !    Store the region references in the /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-
GEO_Aggr dataset.
  !
  CALL H5LRcreate_ref_to_all_f(file_id, group, Dataset_Aggr, H5_INDEX_NAME_F, H5_ITER_INC_F,
H5R_OBJECT_F, status);
```

```
  CALL H5Fclose_f(file_id, status)

END PROGRAM main
```

The first call to the `H5LRcreate_ref_to_all` function creates the dataset `Dataset_Gran_0` (specified by the third parameter) with the region references. Region references point to the selections in the datasets found under the group `GROUP` and sorted in alphabetical order (specified by `H5_INDEX_NAME` as a value of the fourth parameter). `H5_ITER_INC` was used to specify the increasing order of iteration. As a result, the first element of the dataset `Dataset_Gran_0` references the dataset `Height`, and the last references dataset `Time`. Using `H5_ITER_DEC` reverses the order of the references: the first element would reference the dataset `Time` and the last would reference dataset `Height`. If the group had a property set to track the order in which the datasets were created, we could use the `H5_INDEX_CRT_ORDER` parameter to iterate over the datasets in the order they were created. Then the references in the `Dataset_Gran_0` dataset would be created in the order corresponding to the creation order of the datasets instead of the alphabetical order.

The second call to the `H5LRcreate_ref_to_all` function creates the dataset `Dataset_Aggr` with the references to the datasets found under the group `GROUP`. Once again the datasets were iterated in alphabetical order since `H5_INDEX_NAME` was specified as a value of the fourth parameter.

The original file `GMGTO_npp_raw.h5` used in the program did not contain the group `VIIRS-MOD-GTM-EDR-GEO` under the group `/Data_Products`. The function checks the existence of the group `VIIRS-MOD-GTM-EDR-GEO`, creates it if necessary, and then proceeds with the creation of the dataset with region references or references to the objects.

4.4   Combining data that a region reference points to

The function `H5LRmake_dataset` is used to combine all data that a specified region reference points to into one dataset. For example, if region references point to the highlighted regions in the datasets shown on the right in Figure 8, `H5LRmake_dataset` dataset will read each referenced region and write it to the dataset indicated as OTPUT in Figure 8. It is assumed that all regions have the same datatype, rank and dimension sizes, except for the slowest changing one.
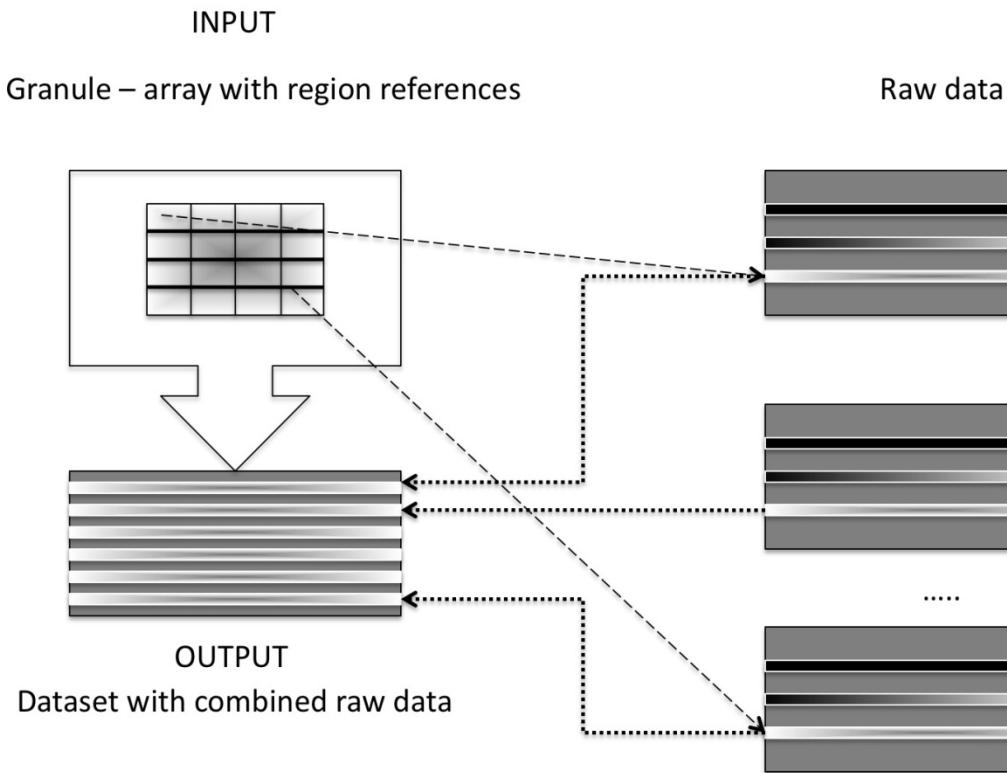
Figure 8: H5LRcreate_dataset finds data that a region reference points to and writes it to the new dataset.

The slightly artificial code shown in Program 5 in the Appendix created references to sub-arrays and then used the references to combine data in one dataset.

```
C:
/*
 *   This example copies subregions of the satellite data SatelliteZenithAngle
 *   SatelliteAzimuthAngle and SatelliteRange under /All_Data/VIIRS-MOD-GTM-EDR-GEO_All
 *   to one dataset Satellite, located under /Data_Products/Subset of the NPP
 *   file.
 *   Main illustrative functions: H5LRcreate_region_references and H5LRmake_dataset
 */

#include <stdlib.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "h5hl_region.h"

#define filename "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

int main(void)
{
    hid_t file_id;  /* file identifier */
    hsize_t block_coord[12] = {3, 51, 7, 53, 3, 51, 7, 53, 3, 51, 7, 53};
    /* hyperslab coordinates, (3,51)-(7,53), for
                              SatelliteZenithAngle, SatelliteAzimuthAngle, SatelliteRange */
    hdset_reg_ref_t ref_subset[3];
```

```
    const char *path[3]; /* full paths to the satellite target datasets for the region references*/
    hid_t file_id_array[3]; /* identifiers describing which HDF5 file the corresponding region
reference belongs to*/
    herr_t status;
    int i;

    path[0]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle";
    path[1]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle";
    path[2]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange";

    /*
     * Open the NPP file.
     */
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);

    /*
     * We are creating the data set in the same file, so fill the file_id path with the same file
id.
     */
    for (i = 0; i < 3; i++)
      file_id_array[i] = file_id;

    /*
     * Create three region references pointing to hyperslabs with block coordinates (3,51)-(7,53) in
     * SatelliteZenithAngle, SatelliteAzimuthAngle and SatelliteRange datasets.
     */
    status = H5LRcreate_region_references(file_id, 3, path, block_coord, ref_subset);

    /*
     * Combine the three datasets into one dataset, 'Satellite', under '/Data_Products/Subset',
     * resulting in a final dataset of size (0,0)-(14,2). Notice the group 'Subset' did not exist
     * so the function automatically created the necessary intermediate group.
     */

    status = H5LRmake_dataset(file_id, "/Data_Products/Subset/Satellite", H5T_NATIVE_FLOAT, 3,
                              file_id_array, ( const hdset_reg_ref_t *)ref_subset);

    status = H5Fclose(file_id);
```

```
}

Fortran:
!
!   This example copies subregions of the satellite data 'SatelliteZenithAngle'
!   'SatelliteAzimuthAngle' and 'SatelliteRange' under '/All_Data/VIIRS-MOD-GTM-EDR-GEO_All'
!   to one dataset,'Satellite', located at '/Data_Products/Subset' of the NPP
!   file.
!   Main illustrative functions: H5LRcreate_region_references_f and H5LRmake_dataset_f
!

PROGRAM main

  USE ISO_C_BINDING
  USE HDF5 ! module of HDF5 library
  USE H5LT ! module of H5LT
  USE H5HL_REGION

  IMPLICIT NONE
  CHARACTER(LEN=78), PARAMETER :: filename = &
        "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

  INTEGER(hid_t) :: file_id  ! file identifier
  INTEGER(hsize_t), DIMENSION(1:12) :: block_coord = (/52,4,54,8,52,4,54,8,52,4,54,8/) ! hyperslab
coordinates, (52,4)-(54,8), for
                                                                                    !
'SatelliteZenithAngle', 'SatelliteAzimuthAngle', 'SatelliteRange'
  TYPE(hdset_reg_ref_t_f), DIMENSION(1:3) ::  ref_subset ! region references to hyperslabs of
'SatelliteZenithAngle', 'SatelliteAzimuthAngle', 'SatelliteRange'
  CHARACTER(LEN=80), DIMENSION(1:3) :: path ! full paths to the satellite target datasets for the
region references
  INTEGER(hsize_t), DIMENSION(1:1) :: dims = (/3/)
  INTEGER(hid_t), DIMENSION(1:3) ::  file_id_array  ! identifiers describing which HDF5 file the
corresponding region reference belongs to
  INTEGER :: status

  path(1)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle"
  path(2)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle"
  path(3)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange"

  !
  ! Initialize FORTRAN interface.
  !
  CALL H5open_f(status)
  !
  ! OPEN the NPP file.
  !
  CALL H5Fopen_f(filename, H5F_ACC_RDWR_F, file_id, status)
  !
  ! We are creating the data set in the same file, so fill the file_id path with the same file id.
  !
  file_id_array(1:3) = file_id
  !
  ! Create three region references pointing to hyperslabs with block coordinates (52,4)-(54,8) in
  ! 'SatelliteZenithAngle', 'SatelliteAzimuthAngle' and 'SatelliteRange' datasets.
  !
  CALL H5LRcreate_region_references_f(file_id, 3_size_t, path, block_coord, ref_subset, status);
  !
  ! Combine the three datasets into one dataset, 'Satellite', under '/Data_Products/Subset',
  ! resulting in a final dataset of size (0,0)-(14,2). Notice the group 'Subset' did not exist
  ! so the FUNCTION automatically created the necessary intermediate group.
  !
```

```
  CALL  H5LRmake_dataset_f(file_id,  "/Data_Products/Subset/Satellite",  H5T_NATIVE_REAL,  3_size_t,
file_id_array, ref_subset, status)

  CALL H5Fclose_f(file_id, status)

END PROGRAM main
```

Data from the datasets that belong to different files can be combined also. The fifth parameter to the function is an array of file or objects identifiers (`file_id_array`). Each element of this array describes to which HDF5 file the corresponding region reference in the array `ref_subsets` belongs. In our example, we created region references that point to the datasets in the same file, but we could create an array of the region references, as shown in the snippet below, that could point to the datasets in the different files. For example, if dataset `SatelliteRange` belongs to a file FOO.h5 with the identifier `file_id_foo`, the dataset Satellite will contain data from both files.

```
hsize_t block_coord[4] = {3, 51, 7, 53};
…..
status = H5LRcreate_region_references(file_id, 1, path[0], block_coord, ref_subset[0]);
status = H5LRcreate_region_references(file_id, 1, path[1], block_coord, ref_subset[1]);
status = H5LRcreate_region_references(file_id_foo, 1, path[2], block_coord, ref_subset[2]);
```

4.5   Reading and copying a subset of data (hyperslabs)

In section 4.2, we explored how to access data that a region reference points to by using HDFView, h5dump and H5LRread_region. In this section, we discuss how to read and copy a dataset region when its corner coordinates are known.

4.5.1   Accessing and copying a region in HDFView

When the name of the dataset and the region of interest are known, it is easy to display the data from the region using HDFView. Figure 9 shows how to access hyperslab with the coordinates (`3`,`51`) – (`7`,`53`) in the dataset `/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange` in the file `<name>.h5`.

Open the file and find the dataset `SatelliteRange`. Then click the dataset to highlight it and right-click to display a dropdown menu. Choose the "Open As" option and enter coordinates of the hyperslab in the "Data Selection" window. The selected region appears in a TableView window. Data can be exported from HDFView and saved in a text file, as was discussed in section 4.2.1.

We can copy data in HDFView from one dataset to another. Figure 10 shows the dataset `SatelliteRange_Subset` under the `Data_Products` group. The dataset is a two-dimensional array 7x5 of floating-point numbers and was created by using HDFView with the follow steps:

1. Highlight the `Data_Products` group and right-click to display the dropdown menu.

2. Choose "New" and "Dataset".

3. Enter the dataset name "SateliteRange_Subset", choose the datatype class "FLOAT", and enter the current size "7x5".

4. Click OK and the newly created dataset appears under the group `Data_Products`.

The HDF Group

5. Open the dataset by clicking an icon in the Tree View pane; data is all 0.

6. In the TableView window that displays the subset of the `SatelliteRange` dataset, select all data.

7. Go to Table in the TableView window and select "Copy".

8. Go to the `StalliteRange_Subset` TableView window and click the cell with the coordinates (1,1); then go to Table and choose "Paste". The data are copied to the new dataset.
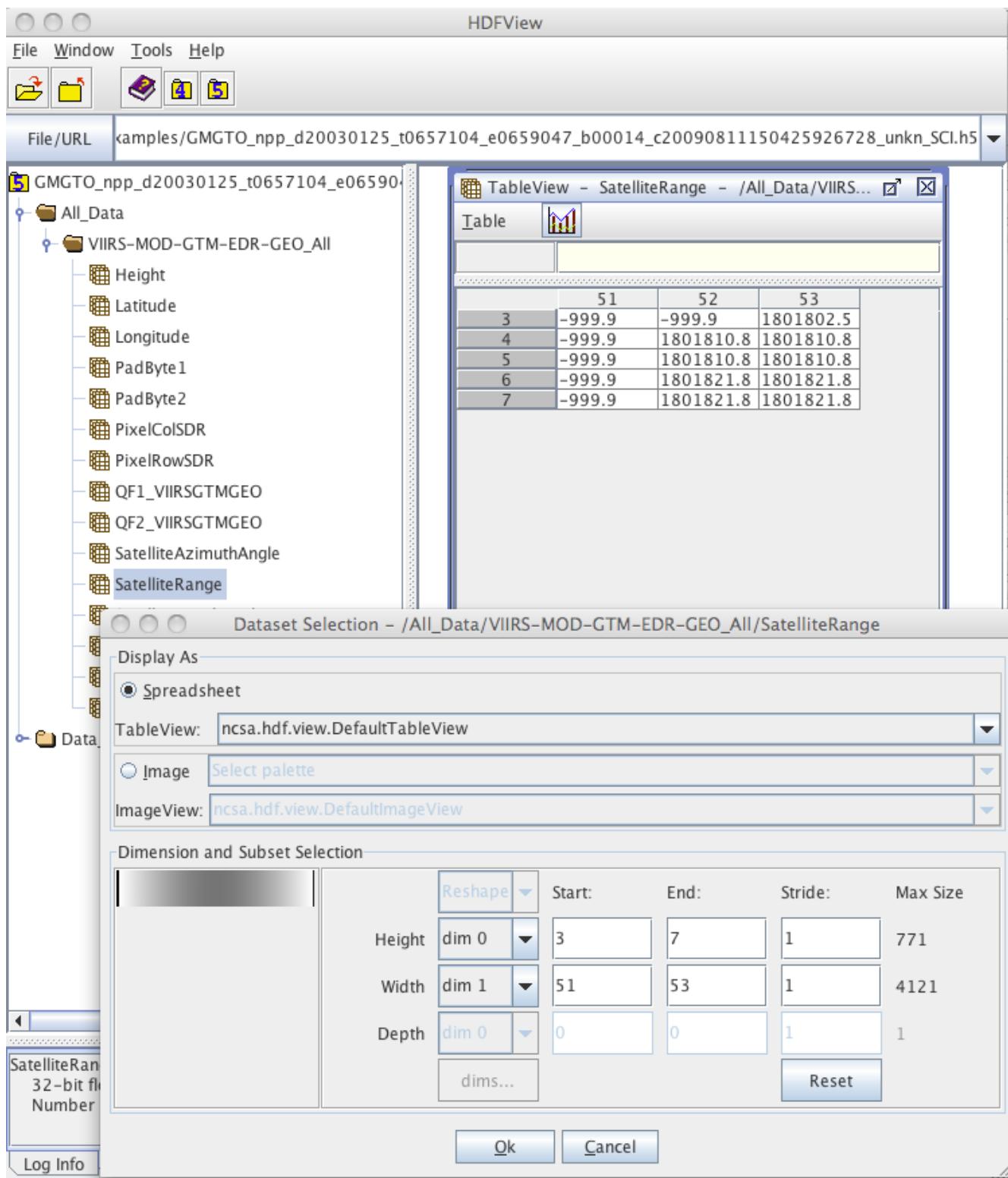
Figure 9. Selecting a region with the coordinates (3,51)-(7,53) in HDFView
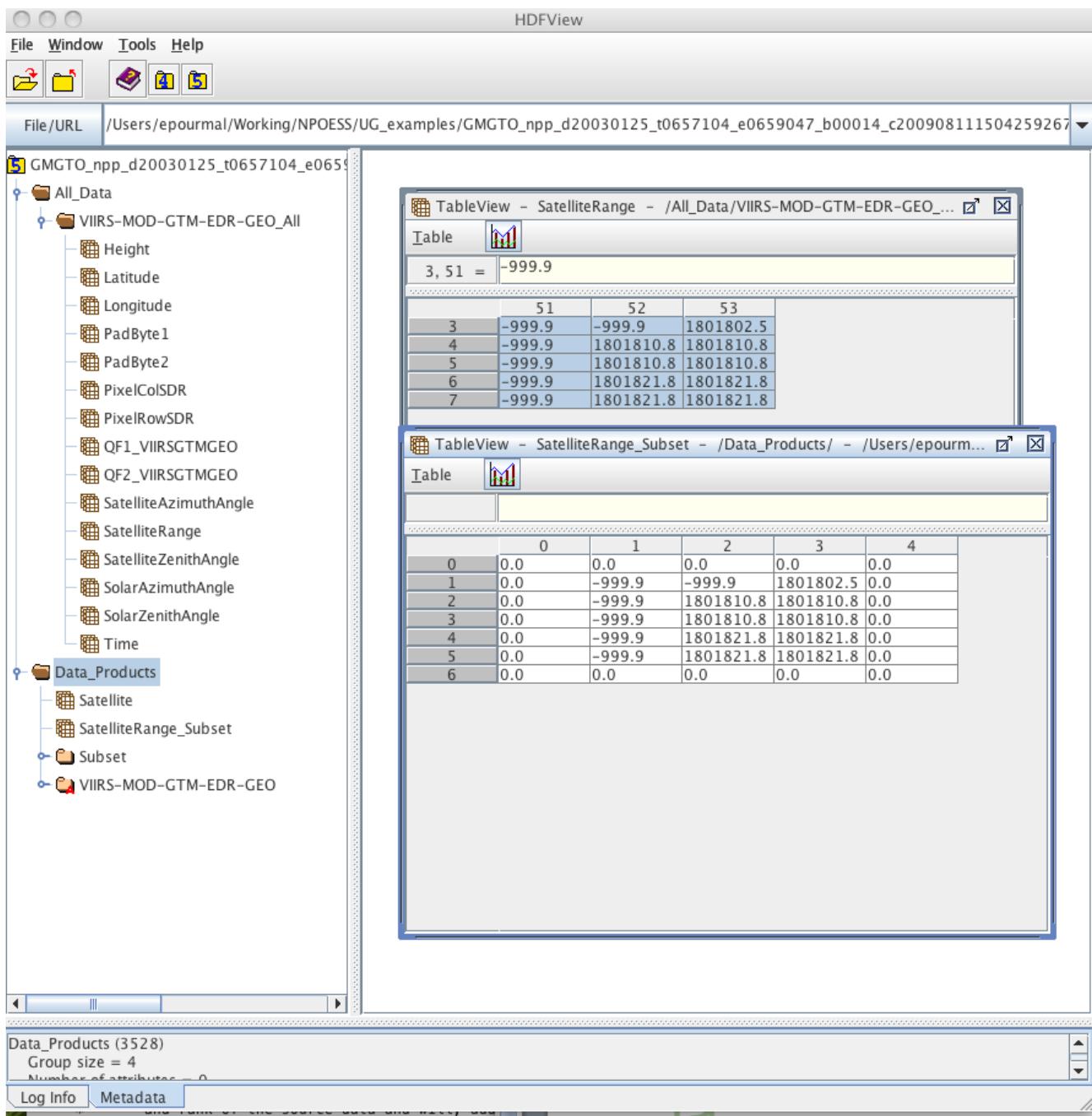
Figure 10. Copying data in HDFView

While displaying and copying small regions work well in HDFView, it is not practical for significant amounts of data. To accomplish this, it is better to export selected data to a file, and then import it to a new file starting at the specified location. Check the "Export Data to File" and "Import Data from File" options under Table in the TableView window.

Other tools like h5dump and h5import may be useful for extracting and importing data. The next section shows how a region can be displayed and exported to a file by h5dump. For importing data, see the h5import utility at http://www.hdfgroup.org/HDF5/doc/RM/Tools.html.

4.5.2 Accessing a region via h5dump

To display a region of a dataset with h5dump, use sub-setting parameters to specify the region as it is shown below.

```
./h5dump -d "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange[3,51;;5,3;]" <name>.h5

HDF5 "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5" {
DATASET "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange" {
   DATATYPE  H5T_IEEE_F32BE
   DATASPACE  SIMPLE { ( 771, 4121 ) / ( H5S_UNLIMITED, H5S_UNLIMITED ) }
   SUBSET {
      START ( 3, 51 );
      STRIDE ( 1, 1 );
      COUNT ( 5, 3 );
      BLOCK ( 1, 1 );
      DATA {
      (3,51): -999.9, -999.9, 1.8018e+06,
      (4,51):  -999.9, 1.80181e+06, 1.80181e+06,
      (5,51):  -999.9, 1.80181e+06, 1.80181e+06,
      (6,51):  -999.9, 1.80182e+06, 1.80182e+06,
      (7,51):  -999.9, 1.80182e+06, 1.80182e+06
      }
   }
}
}
```

As you may notice, the first pair in the brackets that follow the dataset name /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange is the starting coordinate of the hyperslab (3,51). The next pair is omitted. The third pair indicates how many elements to display in each dimension (7-3+1= 5 in the first dimension, and 53-51+1=3 in the second dimension); the fourth pair is omitted. For more information on sub-setting and sub-setting parameters see the help page for h5dump [2], HDF5 Tutorial "Introductory Topics: Reading from or Writing to a Subset of a Dataset" [1], and Chapter 7 of the HDF5 User's Guide [6].

4.5.3 Accessing and copying a region with the high-level functions H5LTread_region and H5LTcopy_region

Reading a selected region with the HDF5 APIs library requires several function calls. The high-level function H5LTread_region significantly reduces the overhead for reading a simple hyperslab (see Figure 11). The function is similar to H5LRread_region, but it does not use a region reference as input. It uses a path to a dataset and hyperslab corner coordinates (for example, obtained by the H5LRget_region_info function) to return the following:

- The number of elements in the selected region
- Data into a buffer allocated by the application
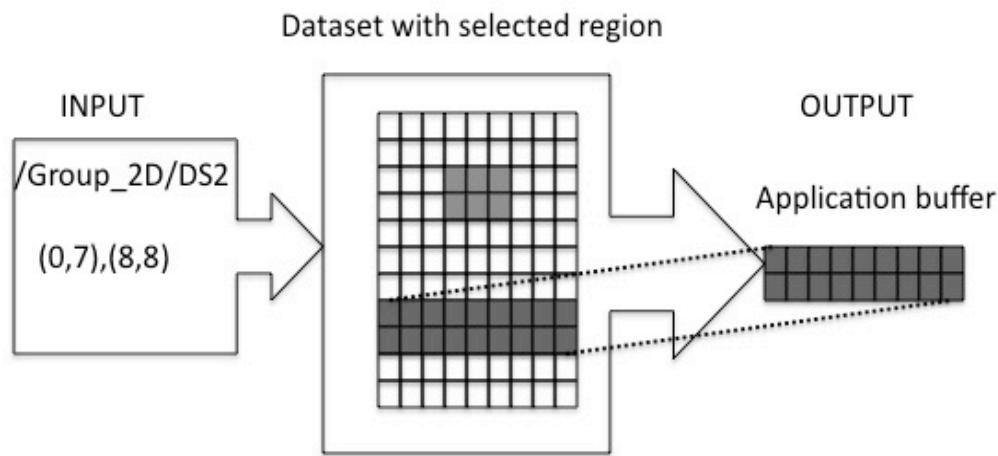
Figure 11. H5LTread_region can be used to read a hyperslab into a buffer

The snippet of code below, from Program 5 in the Appendix, shows how the function can be used to read the hyperslab selection discussed in the sub-sections 4.5.1 and 4.5.2.

```
#define PATH_DEST "/Data_Products/SatelliteRange_Subset"
….
hsize_t block_coord_dest[4] = {1, 1, 5, 3 };
….
/*
 *
 * Reads a subset region of "/Data_Products/SatelliteRange_Subset"
 * specified by coordinates (1,1)-(5,3).
 *
 */
 status = H5LTread_region(filename,PATH_DEST, block_coord_dest, H5T_NATIVE_FLOAT, rdata);
```

Output:

```
Subset of /Data_Products/SatelliteRange_Subset with coordinates (1,1)-(5,3):

  [ -999.900024 -999.900024 1801802.500000 ]
  [ -999.900024 1801810.750000 1801810.750000 ]
  [ -999.900024 1801810.750000 1801810.750000 ]
  [ -999.900024 1801821.750000 1801821.750000 ]
  [ -999.900024 1801821.750000 1801821.750000 ]
```

The function H5LTcopy_region copies data from a specified region in a source dataset to a specified region in a destination dataset. The destination dataset may be in another file, but if it does not exist, the function copies the source dataset and data from the specified region. (See Figure 12)
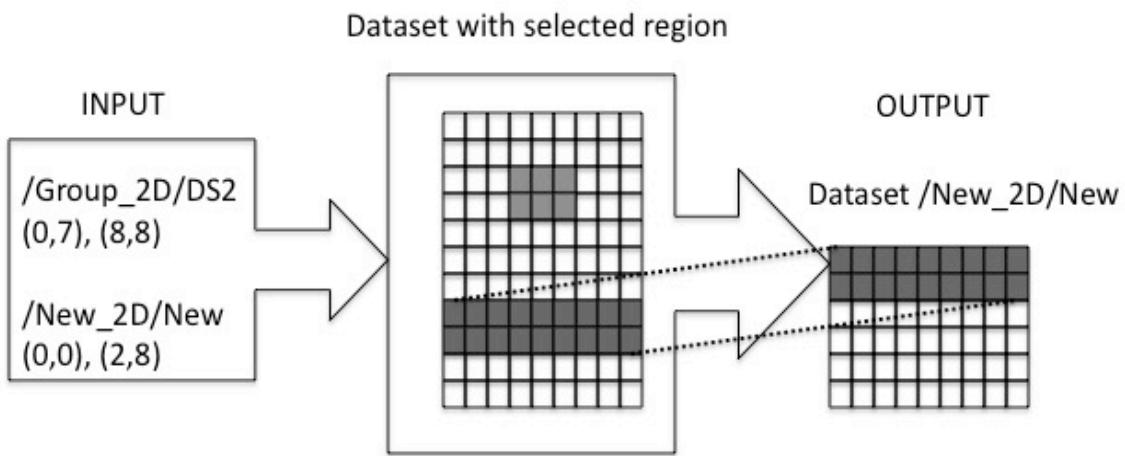
Figure 12. H5LTcopy_region can be used to copy data from a specified region into a dataset

The code snippet below shows how the function can be used to copy region (3,51)-(7,53) in the dataset `SatelliteRange` to the new dataset `SatelliteRange_Subset`, as was done by using HDFView in section 4.5.1.

```
#define PATH_DEST "/Data_Products/SatelliteRange_Subset"
#define PATH_SRC "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange"
….
hsize_t block_coord_src[4] = {3, 51, 7, 53}; /* source's block coordinates (3,51)-(7,53) */
hsize_t block_coord_dest[4] = {1, 1, 5, 3 }; /* destination's block coordinates (1,1)-(5,3) */
….
/*
 * Copy a block of the "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange"
 * data with block corner coordinates of (3,51)-(7,53) to
 * a subset of the "/Data_Products/SatelliteRange_Subset" with hyperslab
 * coordinates (1,1)-(5,3).
 */
 status = H5LTcopy_region(filename, PATH_SRC, block_coord_src, filename, PATH_DEST,
                                             block_coord_dest);
```

4.6   Reading data packed into integer (quality flags)

The purpose of NPOESS quality flags data is to provide quality information about data delivered on an element-by-element basis. Quality flags are stored in HDF5 datasets in an NPOESS product file. The rank and dimension sizes of a quality flags dataset are the same as the rank and dimension sizes of the product data to which quality flags are applied. The datatype of a quality flags dataset is an 8-bit unsigned integer (one byte).

To improve storage efficiency, several quality flags associated with a data product may be packed into one byte, and each quality flag may be comprised of one or more consecutive bits as shown in Figure 13 The IST quality flag takes two bits, while the "Active Fire" quality flag takes one bit. The description of the quality flags is stored in the user block of the NPOESS product file in XML form and is not interpreted by the HDF5 library.
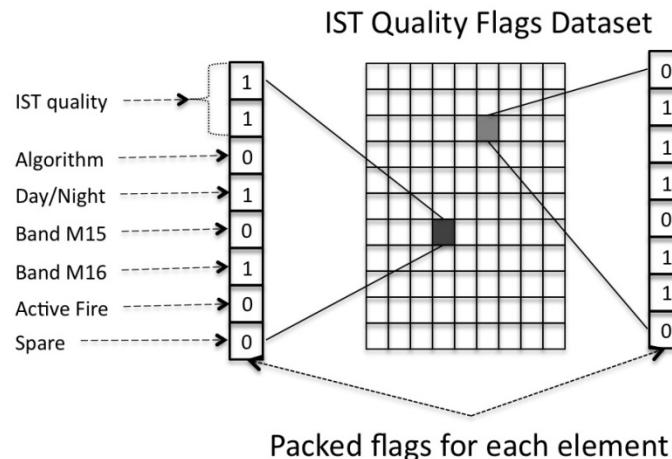
Figure 13. Each element of a dataset contains multiple quality flags values

To access individual quality flag stored in a bit (or a set of bits) one can use the HDF5 library functions to read integer data and then apply a bit-mask to find out the value. To facilitate access to the data packed in an integer value, the HDFView and h5dump tools were modified to display values packed in an 8-bit integer. The HDF5 High-level Region Library has an API to read a value from a particular bit or set of bits. The next sections discuss how HDFView, h5dump and H5LTread_bitfield_value function can be used to accomplish the task.

4.6.1 Displaying packed data with HDFView

HDFView release 2.6.1 and later has an option to apply bit-mask to display a value of particular bit or set of bits.

Figure 14 shows the values of the unsigned 8-bit integer dataset QF_VIIRSI1SDR_Array. If a user knows that the dataset stores values packed into an integer, for example, NPOESS quality flags, he/she can choose the option to apply a bit mask to see the value of a particular bit(s). To do so, select the dataset, and then use right mouse click and "Open As" menu item to bring up a window with the display options. Select "Apply Bitmask" icon, specify the bits to display, and then click "OK" button as shown in Figure 15. The value of the selected bits will be displayed for each element of the dataset (see Figure 16).
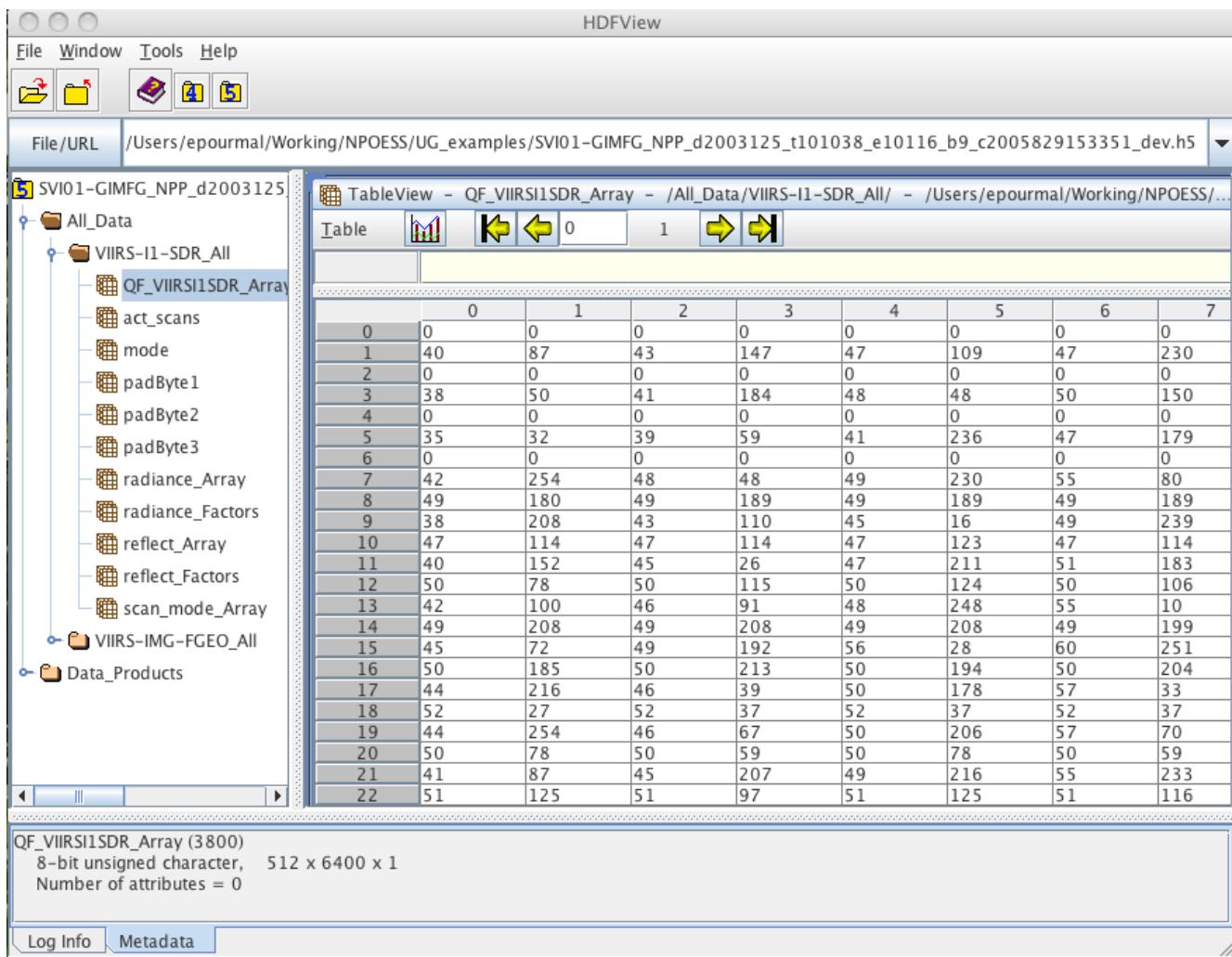
Figure 14. Viewing the "Quality Flags" dataset in HDFView – data is displayed as an 8-bit unsigned integer (character)

The HDF Group
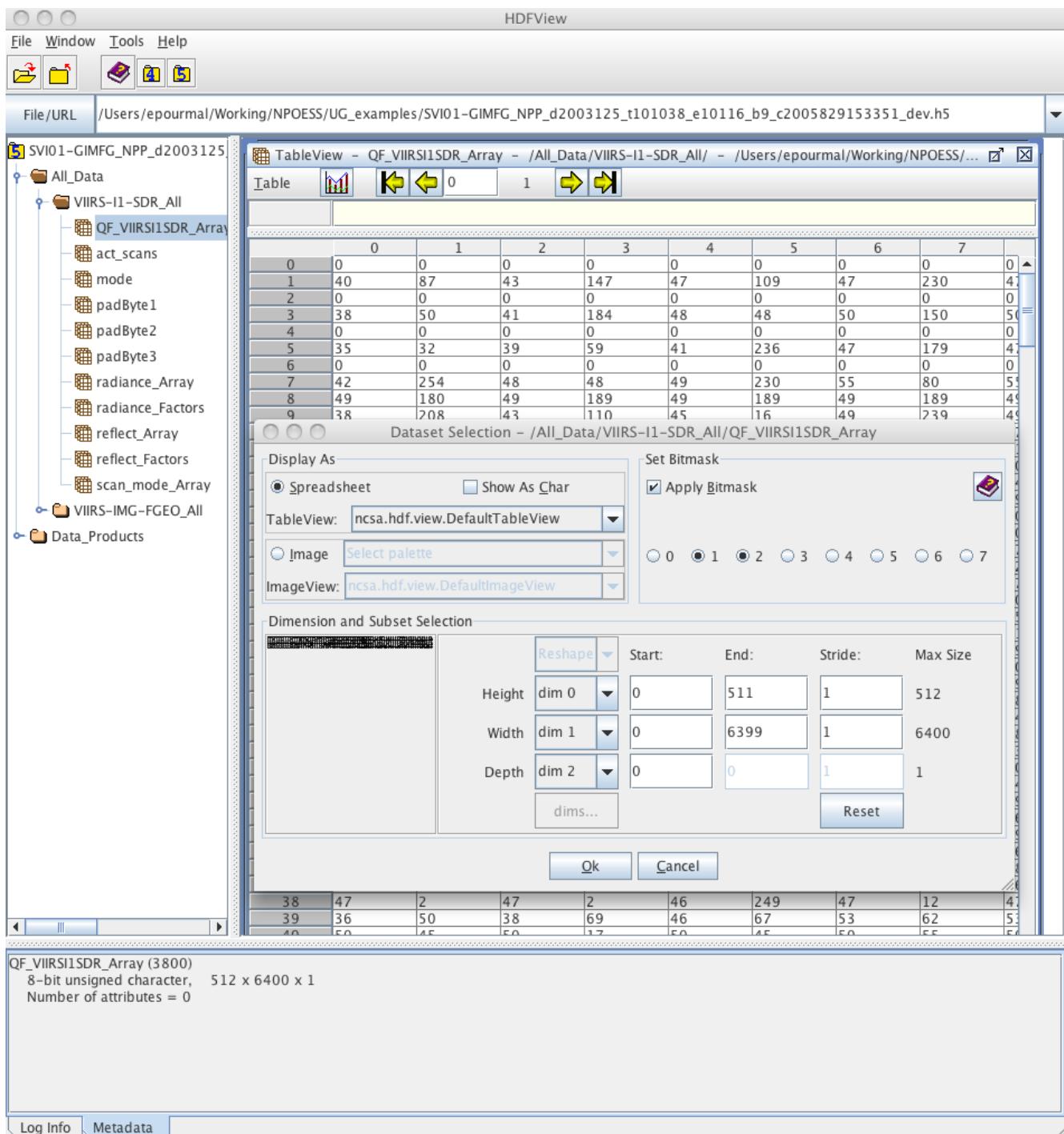
Figure 15. Extracting the second and third bits and displaying the data in HDFView
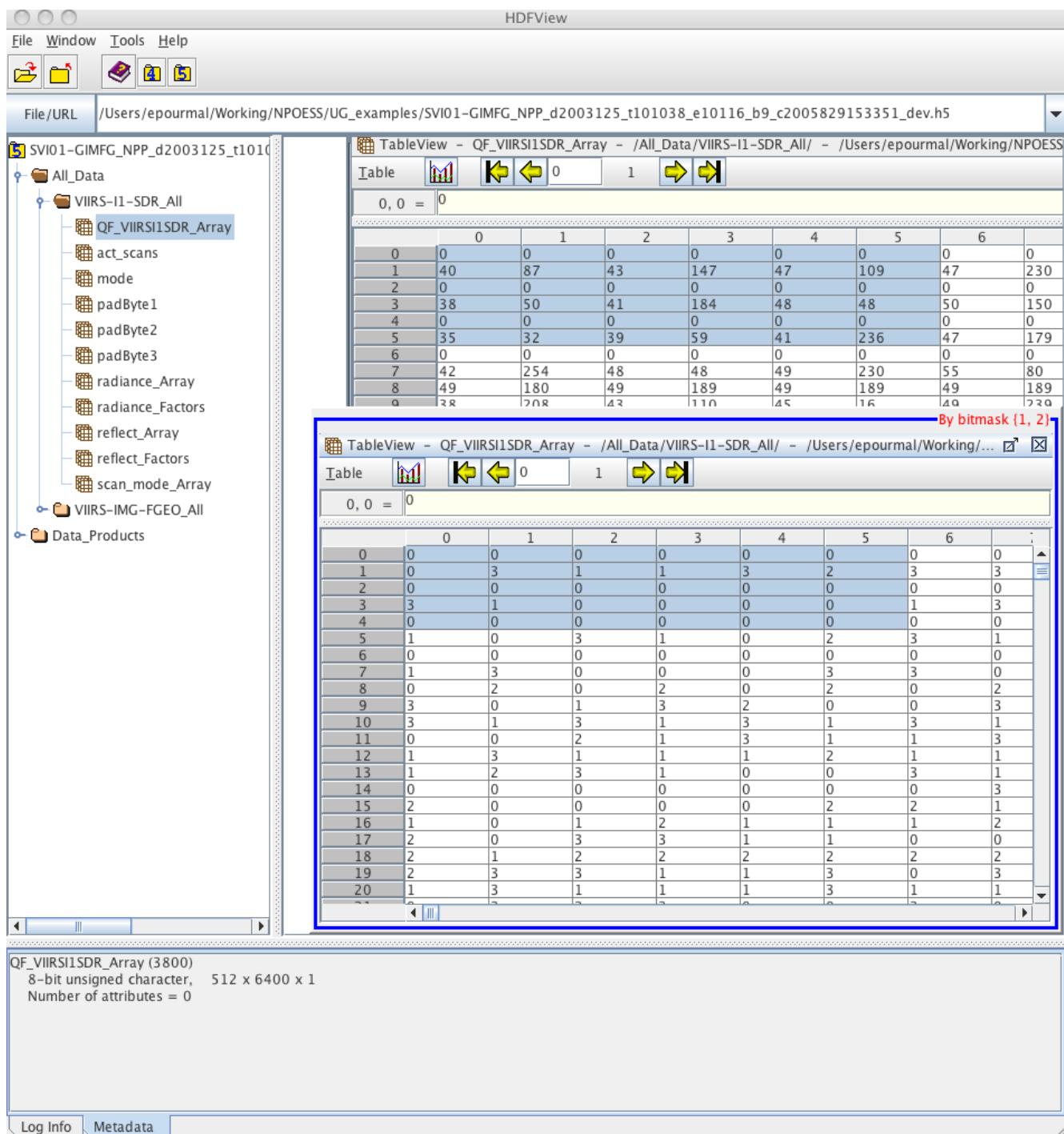
Figure 16. Displaying a number represented by the second and third bits

4.6.2 Displaying packed data with h5dump

The following is the output by h5dump of the highlighted data above. The flag "-d" followed by "`/All_Data/VIIRS-I1-SDR_All/QF_VIIRSI1SDR_Array[0,0,0;;5,6,1;]`" specifies the dataset and the selection within that dataset; "-M" flag followed by the pairs of offset, length is used to specify the mask to apply. The value "1" indicates the second bit and the value "2" indicates two consequent bits, e.g., the value stored in bits 1 and 2 will be displayed.

The HDF Group

```
./h5dump -d "/All_Data/VIIRS-I1-SDR_All/QF_VIIRSI1SDR_Array[0,0,0;;5,6,1;]" -M 1,2
SVI01-GIMFG_NPP_d2003125_t101038_e10116_b9_c2005829153351_dev.h5
HDF5 "SVI01-GIMFG_NPP_d2003125_t101038_e10116_b9_c2005829153351_dev.h5" {
DATASET "/All_Data/VIIRS-I1-SDR_All/QF_VIIRSI1SDR_Array" {
   DATATYPE  H5T_STD_U8LE
   DATASPACE  SIMPLE { ( 512, 6400, 1 ) / ( H5S_UNLIMITED, H5S_UNLIMITED, H5S_UNLIMITED ) }
   PACKED_BITS OFFSET=1 LENGTH=2
   SUBSET {
      START ( 0, 0, 0 );
      STRIDE ( 1, 1, 1 );
      COUNT ( 5, 6, 1 );
      BLOCK ( 1, 1, 1 );
      DATA {
      (0,0,0): 0,
      (0,1,0):  0,
      (0,2,0):  0,
      (0,3,0):  0,
      (0,4,0):  0,
      (0,5,0):  0
      (1,0,0):  0,
      (1,1,0):  3,
      (1,2,0):  1,
      (1,3,0):  1,
      (1,4,0):  3,
      (1,5,0):  2
      (2,0,0):  0,
      (2,1,0):  0,
      (2,2,0):  0,
      (2,3,0):  0,
      (2,4,0):  0,
      (2,5,0):  0
      (3,0,0):  3,
      (3,1,0):  1,
      (3,2,0):  0,
      (3,3,0):  0,
      (3,4,0):  0,
      (3,5,0):  0
      (4,0,0):  0,
      (4,1,0):  0,
      (4,2,0):  0,
      (4,3,0):  0,
      (4,4,0):  0,
      (4,5,0):  0
      }
   }
}
}
```

4.6.3   Extracting packed data with H5LTread_bitfield_value

The function H5LTread_bitfield_value extracts specified bit-field values for each element of the product dataset into a buffer provided by the application, which might be useful for any general application that extracts bit-field values from 8-bit unsigned integers (one byte) stored in HDF5 dataset.

The HDF Group

The following snippet of code is taken from Program 7 in the Appendix. Please notice that we use call to H5Sselet_hyperslab function first to specify the highlighted region, and the H5LTread_bitfield_value call is used to read values stored in the second and the third bits.

```
qf_dset = H5Dopen (file, DATASET, H5P_DEFAULT);
  /*
   * Get dataspace and allocate memory for read buffer. Quality flags dataset
   * has the same dimensionality as corresponding product dataset;
   * we are using its dimensions for illustration purposes only.
   */
  space = H5Dget_space (qf_dset);
  status = H5Sselect_hyperslab(space, H5S_SELECT_SET, start, NULL, count, NULL);
  /*
   * For each element read the value that takes first two bits and
   * store it in a char buffer. This selects all the elements (H5S_ALL)
   */
  status = H5LTread_bitfield_value(qf_dset, num_flags, offset, length, space, (int *)qf_data);


Bit Field:
 [ 0  0  0  0  0  0 ]
 [ 0  3  1  1  3  2 ]
 [ 0  0  0  0  0  0 ]
 [ 3  1  0  0  0  0 ]
 [ 0  0  0  0  0  0 ]
```

Appendix

C and Fortran examples in this section can be found at
http://www.hdfgroup.uiuc.edu/ftp/pub/outgoing/NPOESS/Misc-docs/HL-Tools-UsersGuide/

Program 1 (section 4.2.3) in C

```
/*
 *  This example opens an NPP file and reads a dataset with region references
 *  VIIRS-MOD_GTM-EDR-GEO_Gran_0 under the /Data_Products/VIIRS-MOD-GTM-EDR-GEO group.
 *  Then it finds information about the selected elements pointed by the third reference
 *  and reads the data in.
 *  Main illustrative functions: H5LRget_region_info, H5LRread_region
 */
#include <stdlib.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "H5LTpublic.h"
#include "h5hl_region.h"

#define filename "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"
#define dsetname "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0"


int main(void)
{
    hid_t file_id; /* file identifier */
    hid_t dset_id; /* region reference dataset identifier */
    hsize_t dims[1] =  {15}; /* size of the VIIRS-MOD_GTM-EDR-GEO_Gran_0 dataset */
```

```
    hdset_reg_ref_t ref[15]; /* array to read region references */
    hsize_t buf[4]; /* buffer to read hyperslab coordinates defining region references */
    herr_t status;
    size_t name_length = {1024};
    char name[1024];
    hid_t dtype;
    size_t numelem;
    char type_string[15];
    size_t type_string_length = {15};
    hsize_t rdims[2];
    float *rdata;
    size_t rnumelem;
    int i;


    /*
     * Open the NPP file.
     */
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);
    /*
     * Open dataset and read the dataset with the region references.
     * We made an assumption that the size of the dataset is known.
     */
    dset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);
    status = H5Dread(dset_id, H5T_STD_REF_DSETREG, H5S_ALL, H5S_ALL, H5P_DEFAULT,ref);


    /*
     * Get region reference information such as a name of the dataset the region reference point to,
     * number of contiguous blocks in the selection (should be 1) and the hyperslab coordinates (0,0) -
(770,4120)
     */
    status = H5LRget_region_info(file_id, (const hdset_reg_ref_t*)ref[2], &name_length, name, NULL, &dtype,
                                 NULL, &numelem, buf);


    /*
     * Display the info
     */
    H5LTdtype_to_text(dtype, type_string, H5LT_DDL, &type_string_length);
    printf(" Information retrieved by H5LRget_region_info: \n");
    printf(" Third element of the array with the region references points to  %s \n", name);
    printf(" Length of the string above is %d \n", (int)name_length);
    printf(" Region's datatype is %s \n", type_string);
    printf(" Number of blocks in the region is %d \n", (int)numelem);
    printf(" Block's coordinates are (%d,%d) - (%d,%d) \n", (int)buf[0],(int)buf[1],(int)buf[2],(int)buf[3]);
    printf("\n");


    /*
     * We will read data to the floating-point buffer; using information provided by H5LRget_region_info
     * allocate the buffer to read data in.
     */
    rdims[0] = buf[2] - buf[0] + 1;
    rdims[1] = buf[3] - buf[1] + 1;
    rdata = (float *) malloc (rdims[0] * rdims[1] * sizeof(int));


    /*
     * Read data pointed by the third region reference into a buffer and display the first six elements.
     */
    status = H5LRread_region(file_id,(const hdset_reg_ref_t*)ref[2], H5T_NATIVE_FLOAT, &rnumelem, rdata);
    printf(" Information retrieved by H5LRread_region: \n");
    printf(" Number of elements pointed by a region reference is %ld \n", rnumelem);
    printf(" The first six elements are: \n");
    for (i=0; i<6; i++) printf(" %7.3f ", rdata[i]);
    printf("\n");
    free(rdata);


    /*
     * Close dataset with region references and file.
     */
    status = H5Dclose(dset_id);
    status = H5Fclose(file_id);
```

```
    return 0;
}
Output:
 Information retrieved by H5LRget_region_info:
 Third element of the array with the region references points to  /All_Data/VIIRS-MOD-GTM-EDR-GEO_All/Longitude
 Length of the string above is 46
 Region's datatype is H5T_IEEE_F32BE
 Number of blocks in the region is 1
 Block's coordinates are (0,0) - (770,4120)

 Information retrieved by H5LRread_region:
 Number of elements pointed by a region reference is 3177291
 The first six elements are:
 115.307  115.300  115.294  115.287  115.280  115.274
```

Program 1 (section 4.2.3) in Fortran

```fortran
!
!   This example opens an NPP example file and reads a dataset with region references
!   VIIRS-MOD_GTM-EDR-GEO_Gran_0 under the /Data_Products/VIIRS-MOD-GTM-EDR-GEO group.
!   Then it finds information about the selected elements pointed by the third reference
!   and reads the data in.
!   Main illustrative functions: H5LRget_region_info_f, H5LRread_region_f
!
PROGRAM main

  USE ISO_C_BINDING
  USE HDF5 ! module of HDF5 library
  USE H5LT ! module of H5LT
  USE H5HL_REGION

  IMPLICIT NONE

  CHARACTER(LEN=78), PARAMETER :: filename = &
       "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

  CHARACTER(LEN=65), PARAMETER :: dsetname = &
       "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0" ! dataset with region references

  INTEGER(hid_t) :: file_id       ! file identifier
  INTEGER(hid_t) :: dset_id       ! region reference dataset identifier
  INTEGER(hsize_t), DIMENSION(1:1) :: dims = (/15/) ! size of the VIIRS-MOD_GTM-EDR-GEO_Gran_0 dataset
  TYPE(hdset_reg_ref_t_f), DIMENSION(1:15) :: ref ! array to READ region references
  INTEGER(hsize_t), DIMENSION(1:4) :: buf  ! buffer to READ hyperslab coordinates defining region references
  INTEGER :: status
  INTEGER(size_t) :: name_length
  CHARACTER(LEN = 50) :: name
  INTEGER(hid_t) :: dtype
  INTEGER(size_t) :: numelem
  INTEGER(size_t), PARAMETER :: type_string_length = 15
  CHARACTER(LEN=15) :: type_string
  INTEGER(hsize_t), DIMENSION(1:2) :: rdims
  REAL, ALLOCATABLE, DIMENSION(:), TARGET :: rdata
  INTEGER(size_t) :: rnumelem
  INTEGER :: class
  TYPE(C_PTR) :: f_ptr
  !
  !  Initialize FORTRAN interface.
  !
  CALL H5open_f(status)
  !
  ! Open the NPP file.
  !
  CALL H5Fopen_f(filename, H5F_ACC_RDWR_F, file_id, status)
  !
  ! Open dataset and read the dataset with the region references.
  ! We made an assumption that the size of the dataset is known.
  !
```

```
CALL H5Dopen_f(file_id, dsetname, dset_id, status)
CALL H5Dread_f(dset_id, H5T_STD_REF_DSETREG, ref, dims, status)
!
! Get region reference information such as a name of the dataset the region reference point to,
! number of contiguous blocks in the selection (should be 1) and the hyperslab coordinates (0,0) - (770,4120)
!

! Important to initialize optional variables
name_length = LEN(name)
dtype = 0
name = ''

CALL H5LRget_region_info_f(file_id, ref(3), status, LENGTH=name_length, PATH=name, DTYPE=dtype, &
      NUMELEM=numelem, BUF=buf)
!
! Display the info
!
CALL H5Tget_class_f(dtype, class, status)
WRITE(*, '(" Information retrieved by H5LRget_region_info:")')
WRITE(*,'(" Third element of the array with the region references points to ",A)') name
WRITE(*,'(" Length of the string above is ", I0)') name_length
IF(class.EQ.H5T_FLOAT_F) WRITE(*,' (" Datatype of region is ", A)') "H5T_FLOAT_F"
WRITE(*,'(" Number of blocks in the region is ", I0)') numelem
WRITE(*,'(" Block''s coordinates are (",I0,",",I0,") - (",I0,",",I0,")",/)') buf(1),buf(2),buf(3),buf(4)

!
! We will read data to the floating-point buffer using information provided by H5LRget_region_info
! to allocate the buffer to read data in.

rdims(1) = buf(3) - buf(1) + 1
rdims(2) = buf(4) - buf(2) + 1
ALLOCATE(rdata(rdims(1) * rdims(2)))
!
! Read data pointed by the third region reference into a buffer and display the first six elements.
!
f_ptr = C_LOC(rdata(1))
CALL H5LRread_region_f(file_id, ref(3), H5T_NATIVE_REAL, rnumelem, f_ptr, status)

WRITE(*,'(" Information retrieved by H5LRread_region:")')
WRITE(*,'(" Number of elements pointed by a region reference is ",I0)') rnumelem
WRITE(*,'(" The first six elements are: ")')
WRITE(*,'(6(x,f7.3))')  rdata(1:6)
DEALLOCATE(rdata)
!
! Close dataset with region references and file.
!
CALL H5Dclose_f(dset_id, status)
CALL H5Fclose_f(file_id, status)

END PROGRAM main
```

Program 2 (section 4.2.3) in C

```c
/*
 *   This example opens an NPP example file and reads a dataset with region references
 *   VIIRS-MOD_GTM-EDR-GEO_Gran_0 under the /Data_Products/VIIRS-MOD-GTM-EDR-GEO group.
 *   Then it finds information about the selected elements pointed by the third reference
 *   and reads the data in.
 *   Main illustrative functions: H5LRget_region_info, H5LRread_region
 */
#include <stdlib.h>
#include <string.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "H5LTpublic.h"
#include "h5hl_region.h"

#define filename "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"
#define dsetname "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0"
```

```
int main(void)
{
    hid_t file_id; /* file identifier */
    hid_t dset_id; /* region reference dataset identifier */
    hdset_reg_ref_t ref[15]; /* array to read region references */
    herr_t status;
    hid_t dtype; /* file datatype handle */
    hid_t mtype; /* mempry datatype handle */
    size_t msize; /* size of memory datatype */
    size_t rnumelem; /* number of elements to read */
    char *rdata; /* pointer to read buffer */
    int i;

    /*
     * Open the NPP file.
     */
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);
    /*
     * Open dataset and read the dataset with the region references.
     * We made an assumption that the size of the dataset is known.
     */
    dset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);
    status = H5Dread(dset_id, H5T_STD_REF_DSETREG, H5S_ALL, H5S_ALL, H5P_DEFAULT,ref);

    /*
     * Get datatype of the data the region reference points to.
     */
    status = H5LRget_region_info(file_id, (const hdset_reg_ref_t*)ref[2], NULL, NULL, NULL, &dtype,
                                 NULL, NULL, NULL);

    /*
     * Find the corresponding type in memory and its size.
     */
    mtype = H5Tget_native_type(dtype, H5T_DIR_ASCEND);
    msize = H5Tget_size(mtype);

    /*
     * Find number of elements in the region to read.
     */
    status = H5LRread_region(file_id,(const hdset_reg_ref_t*)ref[2], mtype, &rnumelem, NULL);

    /*
     * Allocate buffer to read data in.
     */
    rdata = (char *) malloc (rnumelem * msize);
    status = H5LRread_region(file_id,(const hdset_reg_ref_t*)ref[2], mtype, NULL, rdata);

    /*
     * We need to discover an appropriate C type to print data
     */
    if ((H5T_FLOAT == H5Tget_class(mtype)) & (sizeof(float) == msize)) {
        float tmp; /* temporary variables */
        char *tmp_p;
        printf(" Number of elements pointed by a region reference is %ld \n", rnumelem);
        printf(" The first six elements are: \n");
        tmp_p = rdata;
        for (i=0; i<6; i++) {
            memcpy (&tmp, tmp_p, msize);
            printf(" %7.3f ", tmp);
            tmp_p = tmp_p + msize;
        }
        printf("\n");
    }
    free(rdata);

    /*
     * Close dataset with region references and file.
     */
    status = H5Dclose(dset_id);
```

```
    status = H5Fclose(file_id);

    return 0;
}
Output:
 Number of elements pointed by a region reference is 3177291
 The first six elements are:
 115.307  115.300  115.294  115.287  115.280  115.274
```

Program 2 (section 4.2.3) in Fortran

```
!
!    This example opens an NPP example file and reads a dataset with region references
!    VIIRS-MOD_GTM-EDR-GEO_Gran_0 under the /Data_Products/VIIRS-MOD-GTM-EDR-GEO group.
!    Then it finds information about the selected elements pointed by the third reference
!    and reads the data in.
!    Main illustrative functions: H5LRget_region_info_f, H5LRread_region_f

PROGRAM main

  USE ISO_C_BINDING
  USE HDF5 ! module of HDF5 library
  USE H5LT ! module of H5LT
  USE H5HL_REGION

  IMPLICIT NONE

  CHARACTER(LEN=78), PARAMETER :: filename = &
       "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

  CHARACTER(LEN=65), PARAMETER :: dsetname = &
       "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0" !dataset with region references

  INTEGER(hid_t) :: file_id ! file identifier
  INTEGER(hid_t) :: dset_id ! region reference dataset identifier
  TYPE(hdset_reg_ref_t_f), DIMENSION(1:15), TARGET :: ref ! array to read region references
  INTEGER :: status
  INTEGER(hid_t) :: dtype ! file datatype handle
  INTEGER(hid_t) :: mtype ! mempry datatype handle
  INTEGER(size_t) :: msize ! size of memory datatype
  INTEGER(size_t) :: rnumelem ! number of elements to read
  REAL, ALLOCATABLE, DIMENSION(:), TARGET :: rdata ! POINTER to READ buffer
  INTEGER :: class
  INTEGER(HSIZE_T), DIMENSION(1:1) :: dims =(/15/)
  TYPE(C_PTR) :: f_ptr
  !
  !  Initialize FORTRAN interface.
  !
  CALL h5open_f(status)
  !
  ! Open the NPP file.
  !
  CALL H5Fopen_f(filename, H5F_ACC_RDWR_F, file_id, status)
  !
  ! Open dataset and read the dataset with the region references.
  ! We made an assumption that the size of the dataset is known.
  !
  CALL H5Dopen_f(file_id, dsetname, dset_id, status)
  CALL h5dread_f(dset_id, H5T_STD_REF_DSETREG, ref, dims, status)
  !
  ! Get datatype of the data the region reference points to.
  !
  ! Important to initialize optional variables
  dtype = 0

  CALL H5LRget_region_info_f(file_id, ref(3), status, DTYPE=dtype)
  !
  ! Find the corresponding type in memory and its size.
  !
```

```
  CALL H5Tget_native_type_f(dtype, H5T_DIR_ASCEND_F, mtype, status)

  CALL H5Tget_size_f(mtype, msize, status)
  !
  ! Find number of elements in the region to read.
  !
  f_ptr = C_NULL_PTR
  CALL H5LRread_region_f(file_id, ref(3), mtype, rnumelem, f_ptr, status)

  CALL H5Tget_class_f(dtype, class, status)

  IF(class.EQ.H5T_FLOAT_F)THEN
     !
     ! Allocate buffer to read data in.
     !
     ALLOCATE( rdata(1:rnumelem) )

     f_ptr = C_LOC(rdata(1))
     CALL H5LRread_region_f(file_id, ref(3), mtype, rnumelem, f_ptr, status)

     WRITE(*,'(" Number of elements pointed by a region reference is ",I0)') rnumelem
     WRITE(*,'(" The first six elements are: ")')
     WRITE(*,'(6(x,f7.3))')  rdata(1:6)

     DEALLOCATE(rdata)
  ENDIF
  !
  ! Close dataset with region references and file.
  !
  CALL H5Dclose_f(dset_id,status)
  CALL H5Fclose_f(file_id,status)

END PROGRAM main
```

Program 3 (section 4.3.1) in C

```c
/*
 *    This example creates region refernces to the selections in the 'SatelliteZenithAngle'
 *    'SatelliteAzimuthAngle' and 'SatelliteRange' datsets under '/All_Data/VIIRS-MOD-GTM-EDR-GEO_All'
 *    and writes it to the 'Satellite' dataset located in the  '/Data_Products' group.
 *    Main illustrative functions: H5LRcreate_region_references
 */

#include <stdlib.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "H5LTpublic.h"
#include "h5hl_region.h"

#define filename "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c200908111150425926728_unkn_SCI.h5"

int main(void)
{
    hid_t file_id;  /* file identifier */
    /* hyperslab coordinates, (3,51)-(7,53) in each of the 'SatelliteZenithAngle', 'SatelliteAzimuthAngle',
       'SatelliteRange' datsets */
    hsize_t block_coord[12] = {3, 51, 7, 53, 3, 51, 7, 53, 3, 51, 7, 53};
    hdset_reg_ref_t ref[3];
    const char *path[3]; /* full paths to the satellite target datasets for the region references*/
    hsize_t dims[1] = {3};
    herr_t status;

    path[0]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle";
    path[1]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle";
    path[2]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange";

    /*
     * Open the NPP file.
     */
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);
```

```
    /*
     * Create three region references pointing to hyperslabs with block coordinates (3,51)-(7,53) in
     * 'SatelliteZenithAngle', 'SatelliteAzimuthAngle' and 'SatelliteRange' datasets.
     */
    status = H5LRcreate_region_references(file_id, 3, path, block_coord, ref);

    /*
     * Create a dataset with the region references.
     */

    status = H5LTmake_dataset(file_id, "/Data_Products/Satellite", 1, dims, H5T_STD_REF_DSETREG, ( const
hdset_reg_ref_t *)ref);

    status = H5Fclose(file_id);

}
```

Program 3 (section 4.3.1) in Fortran

```
!
!   This example creates region refernces to the selections in the 'SatelliteZenithAngle'
!   'SatelliteAzimuthAngle' and 'SatelliteRange' datsets under '/All_Data/VIIRS-MOD-GTM-EDR-GEO_All'
!   and writes it to the 'Satellite' dataset located in the  '/Data_Products' group.
!   Main illustrative functions: H5LRcreate_region_references_f
!
PROGRAM main

  USE ISO_C_BINDING
  USE HDF5 ! module of HDF5 library
  USE H5LT ! module of H5LT
  USE H5HL_REGION

  IMPLICIT NONE

  CHARACTER(LEN=78), PARAMETER :: filename = &
       "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

  INTEGER(hid_t) :: file_id  ! file identifier
  ! hyperslab coordinates, (52,4)-(54,8) in each of the 'SatelliteZenithAngle', 'SatelliteAzimuthAngle',
'SatelliteRange' datsets
  INTEGER(hsize_t), DIMENSION(1:12) :: block_coord = (/52,4,54,8,52,4,54,8,52,4,54,8/)
  TYPE(hdset_reg_ref_t_f), DIMENSION(1:3) ::  ref ! region references to hyperslabs of 'SatelliteZenithAngle',
'SatelliteAzimuthAngle', 'SatelliteRange'
  CHARACTER(LEN=80), DIMENSION(1:3) :: path ! full paths to the satellite target datasets for the region
references!
  INTEGER(hsize_t), DIMENSION(1:1) :: dims = (/3/)
  INTEGER(HID_T) :: space_id     ! Dataspace identifier
  INTEGER(HID_T) :: dset_id      ! Dataset identifier
  INTEGER :: status

  path(1)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle"
  path(2)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle"
  path(3)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange"
  !
  !  Initialize FORTRAN interface.
  !
  CALL H5open_f(status)
  !
  ! Open the NPP file.
  !
  CALL H5Fopen_f(filename, H5F_ACC_RDWR_F, file_id, status)
  !
  ! Create three region references pointing to hyperslabs with block coordinates (52,4)-(54,8) in
  ! 'SatelliteZenithAngle', 'SatelliteAzimuthAngle' and 'SatelliteRange' datasets.
  !
  CALL H5LRcreate_region_references_f(file_id, 3_size_t, path, block_coord, ref, status)

  !
```

```
  ! I. Create a dataset with the region references.
  !
  !   (a) Create dataspace for dataset with references to dataset regions
  !
  CALL H5Screate_simple_f(1, dims, space_id, status)
  !
  !   (b) Create dataset with references
  !
  CALL H5Dcreate_f(file_id, "/Data_Products/Satellite", H5T_STD_REF_DSETREG, space_id, &
      dset_id, status)
  !
  !   (c) Write dataset with references
  !
  CALL H5Dwrite_f(dset_id, H5T_STD_REF_DSETREG, ref, dims, status)

  CALL H5Sclose_f(space_id, status)
  CALL H5Dclose_f(dset_id, status)
  CALL H5Fclose_f(file_id, status)

END PROGRAM main
```

Program 4 (section 4.3.2) in C

```c
/*
 * This example shows how to create datasets with region and object references to
 * datasets located under the group /Data_Products/VIIRS-MOD-GTM-EDR-GEO
 *   Main illustrative functions: H5LRcreate_ref_to_all
 */

#include <stdlib.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "H5LTpublic.h"
#include "h5hl_region.h"

#define GROUP "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All"
#define Dataset_Aggr "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Aggr"
#define Dataset_Gran_0 "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-GEO_Gran_0"

#define filename "GMGTO_npp_raw.h5"

int main(void)
{
    hid_t file_id;  /* file identifier */
    herr_t status;
    /*
     * Open the NPP file.
     */
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);

    /*
     * Create a dataset with region references to all the datasets located under
     * /All_Data/VIIRS-MOD-GTM-EDR-GEO_All
     * Store the region references in the /Data_Products/VIIRS-MOD-GTM-EDR-
GEO_Gan_0
     * dataset.
     */
    status = H5LRcreate_ref_to_all(file_id, GROUP, Dataset_Gran_0, H5_INDEX_NAME, H5_ITER_INC,
                                   H5R_DATASET_REGION);
    /*
     * Create a dataset with object references to all the datasets located under
     * /All_Data/VIIRS-MOD-GTM-EDR-GEO_All
     * Store the region references in the /Data_Products/VIIRS-MOD-GTM-EDR-
GEO_Aggr
```

```
    * dataset.
    */
    status  =  H5LRcreate_ref_to_all(file_id,  GROUP,  Dataset_Aggr,  H5_INDEX_NAME,  H5_ITER_INC,
H5R_OBJECT);
    status = H5Fclose(file_id);

}
```

Program 4 (section 4.3.2) in Fortran

```
!
! This example shows how to create datasets with region and object references to
! datasets located under the group /Data_Products/VIIRS-MOD-GTM-EDR-GEO
!   Main illustrative functions: H5LRcreate_ref_to_all_f
!
PROGRAM main

  USE ISO_C_BINDING
  USE HDF5 ! module of HDF5 library
  USE H5LT ! module of H5LT
  USE H5HL_REGION

  IMPLICIT NONE

  CHARACTER(LEN=78), PARAMETER :: filename = "GMGTO_npp_raw.h5"
  CHARACTER(LEN=65), PARAMETER :: Dataset_Gran_0 = "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-
GTM-EDR-GEO_Gran_0"
  CHARACTER(LEN=35), PARAMETER :: group = "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All"
  CHARACTER(LEN=63), PARAMETER  ::  Dataset_Aggr  =  "/Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-
GTM-EDR-GEO_Aggr"

  INTEGER(hid_t) :: file_id  ! file identifier
  INTEGER :: status
  !
  !   Initialize FORTRAN interface.
  !
  CALL H5open_f(status)
  !
  ! Open the NPP file.
  !
  CALL H5Fopen_f(filename, H5F_ACC_RDWR_F, file_id, status)
  !
  !  Create a dataset with region references to all the datasets located under /All_Data/VIIRS-MOD-
GTM-EDR-GEO_All
  !    Store  the  region  references  in  the  /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-
GEO_Gan_0 dataset.
  !
  CALL  H5LRcreate_ref_to_all_f(file_id,  group,  Dataset_Gran_0,  H5_INDEX_NAME_F,  H5_ITER_INC_F,
H5R_DATASET_REGION_F, status)
  !
  !  Create a dataset with object references to all the datasets located under /All_Data/VIIRS-MOD-
GTM-EDR-GEO_All
  !    Store  the  region  references  in  the  /Data_Products/VIIRS-MOD-GTM-EDR-GEO/VIIRS-MOD-GTM-EDR-
GEO_Aggr dataset.
  !
  CALL  H5LRcreate_ref_to_all_f(file_id,  group,  Dataset_Aggr,  H5_INDEX_NAME_F,  H5_ITER_INC_F,
H5R_OBJECT_F, status);

  CALL H5Fclose_f(file_id, status)

END PROGRAM main
```

Program 5 (section 4.4) in C

```c
/*
 *   This example copies subregions of the satellite data SatelliteZenithAngle
 *   SatelliteAzimuthAngle and SatelliteRange under /All_Data/VIIRS-MOD-GTM-EDR-GEO_All
 *   to one dataset Satellite, located under /Data_Products/Subset of the NPP
 *   file.
 *   Main illustrative functions: H5LRcreate_region_references and H5LRmake_dataset
 */


#include <stdlib.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "h5hl_region.h"

#define filename "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

int main(void)
{
    hid_t file_id;  /* file identifier */
    hsize_t block_coord[12] = {3, 51, 7, 53, 3, 51, 7, 53, 3, 51, 7, 53};
    /* hyperslab coordinates, (3,51)-(7,53), for
                              SatelliteZenithAngle, SatelliteAzimuthAngle, SatelliteRange */
    hdset_reg_ref_t ref_subset[3];
    const char *path[3]; /* full paths to the satellite target datasets for the region references*/
    hid_t  file_id_array[3];  /* identifiers describing which HDF5 file the corresponding region
reference belongs to*/
    herr_t status;
    int i;

    path[0]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle";
    path[1]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle";
    path[2]= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange";

    /*
     * Open the NPP file.
     */
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);

    /*
     * We are creating the data set in the same file, so fill the file_id path with the same file
id.
     */
    for (i = 0; i < 3; i++)
      file_id_array[i] = file_id;

    /*
     * Create three region references pointing to hyperslabs with block coordinates (3,51)-(7,53) in
     * SatelliteZenithAngle, SatelliteAzimuthAngle and SatelliteRange datasets.
     */
    status = H5LRcreate_region_references(file_id, 3, path, block_coord, ref_subset);

    /*
     * Combine the three datasets into one dataset, 'Satellite', under '/Data_Products/Subset',
     * resulting in a final dataset of size (0,0)-(14,2). Notice the group 'Subset' did not exist
     * so the function automatically created the necessary intermediate group.
     */

    status = H5LRmake_dataset(file_id, "/Data_Products/Subset/Satellite", H5T_NATIVE_FLOAT, 3,
                              file_id_array, ( const hdset_reg_ref_t *)ref_subset);

    status = H5Fclose(file_id);
```

}
Program 5 (section 4.4) in Fortran

The HDF Group

```
!
!   This example copies subregions of the satellite data 'SatelliteZenithAngle'
!   'SatelliteAzimuthAngle' and 'SatelliteRange' under '/All_Data/VIIRS-MOD-GTM-EDR-GEO_All'
!   to one dataset,'Satellite', located at '/Data_Products/Subset' of the NPP
!   file.
!   Main illustrative functions: H5LRcreate_region_references_f and H5LRmake_dataset_f
!

PROGRAM main

   USE ISO_C_BINDING
   USE HDF5 ! module of HDF5 library
   USE H5LT ! module of H5LT
   USE H5HL_REGION

   IMPLICIT NONE
   CHARACTER(LEN=78), PARAMETER :: filename = &
        "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

   INTEGER(hid_t) :: file_id  ! file identifier
   INTEGER(hsize_t), DIMENSION(1:12) :: block_coord = (/52,4,54,8,52,4,54,8,52,4,54,8/) ! hyperslab
coordinates, (52,4)-(54,8), for
                                                                            !
'SatelliteZenithAngle', 'SatelliteAzimuthAngle', 'SatelliteRange'
   TYPE(hdset_reg_ref_t_f), DIMENSION(1:3) ::   ref_subset ! region references to hyperslabs of
'SatelliteZenithAngle', 'SatelliteAzimuthAngle', 'SatelliteRange'
   CHARACTER(LEN=80), DIMENSION(1:3) :: path ! full paths to the satellite target datasets for the
region references
   INTEGER(hsize_t), DIMENSION(1:1) :: dims = (/3/)
   INTEGER(hid_t), DIMENSION(1:3) ::   file_id_array  ! identifiers describing which HDF5 file the
corresponding region reference belongs to
   INTEGER :: status

   path(1)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteZenithAngle"
   path(2)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteAzimuthAngle"
   path(3)= "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange"

   !
   ! Initialize FORTRAN interface.
   !
   CALL H5open_f(status)
   !
   ! OPEN the NPP file.
   !
   CALL H5Fopen_f(filename, H5F_ACC_RDWR_F, file_id, status)
   !
   ! We are creating the data set in the same file, so fill the file_id path with the same file id.
   !
   file_id_array(1:3) = file_id
   !
   ! Create three region references pointing to hyperslabs with block coordinates (52,4)-(54,8) in
   ! 'SatelliteZenithAngle', 'SatelliteAzimuthAngle' and 'SatelliteRange' datasets.
   !
   CALL H5LRcreate_region_references_f(file_id, 3_size_t, path, block_coord, ref_subset, status);
   !
   ! Combine the three datasets into one dataset, 'Satellite', under '/Data_Products/Subset',
   ! resulting in a final dataset of size (0,0)-(14,2). Notice the group 'Subset' did not exist
   ! so the FUNCTION automatically created the necessary intermediate group.
   !
   CALL  H5LRmake_dataset_f(file_id,  "/Data_Products/Subset/Satellite",  H5T_NATIVE_REAL,  3_size_t,
file_id_array, ref_subset, status)
```

```
  CALL H5Fclose_f(file_id, status)
END PROGRAM main
```

Program 6 (section 4.5.3) in C

```c
/*
 *    This example copies a hyperslab region of 'SatelliteRange' under
 *    '/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/' to another hyperslab region
 *    '/SatelliteRange_Subset' under '/All_Data'. It then reads back
 *    the newly created hyperslab region.
 *     Main illustrative functions: H5LTread_region, H5LTcopy_region
 */
#include <stdlib.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "h5hl_region.h"

#define filename "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

#define PATH_DEST "/Data_Products/SatelliteRange_Subset" /* Full path of the source dataset */
#define PATH_SRC "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange" /* Full path to the destination dataset */

#define NRANK 2 /* rank of source and destination dataset */

int main(void)
{
    hid_t file_id;                            /* file identifier */
    hid_t dset_id;                            /* dataset identifier */
    hid_t space_id;                           /* dataspace identifier */
    hsize_t block_coord_src[4] = {3, 51, 7, 53}; /* source's block coordinates (3,51)-(7,53) */
    hsize_t block_coord_dest[4] = {1, 1, 5, 3 }; /* destination's block coordinates (1,1)-(5,3) */
    hsize_t dims[NRANK] = {7,5};              /* receiving dataset dimensions */
    float rdata[5][3];                        /* buffer to read destination data into */
    int i,j;
    herr_t status;
    /*
     * Open the NPP file.
     */
    file_id = H5Fopen(filename, H5F_ACC_RDWR, H5P_DEFAULT);

    /*
     * First create the destination dataset "SatelliteRange_Subset" since it does not exist.
     *
     * NOTE: If the destination dataset does not already exist then H5LTcopy_region
     *       will automatically create a destination dataset that is the same size
     *       and rank of the source data and will, additionally, fill the destination
     *       block starting at (0,0); thus ignoring the destination's block coordinates.
     */
    space_id = H5Screate_simple(NRANK, dims, NULL);
    dset_id = H5Dcreate2(file_id, PATH_DEST, H5T_NATIVE_FLOAT, space_id, H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
    status = H5Sclose(space_id);
    status = H5Fclose(file_id);

    /*
     * Copy a block of the "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange"
     * data with block corner coordinates of (3,51)-(7,53) to
     * a subset of the "/Data_Products/SatelliteRange_Subset" with hyperslab
     * coordinates (1,1)-(5,3).
     *
     */
    status = H5LTcopy_region(filename, PATH_SRC, block_coord_src, filename, PATH_DEST,
                             block_coord_dest);

    /*
     *
     * Reads a subset region of "/Data_Products/SatelliteRange_Subset"
     * specified by coordinates (1,1)-(5,3).
```

```
 *
 */
    status = H5LTread_region(filename, PATH_DEST, block_coord_dest, H5T_NATIVE_FLOAT,
                            rdata);

    printf("Subset of /Data_Products/SatelliteRange_Subset with coordinates ");
    printf("(%d,%d)-(%d,%d):\n",(int)block_coord_dest[0],(int)block_coord_dest[1],
           (int)block_coord_dest[2],(int)block_coord_dest[3]);

    for (i=0; i< 5; i++)
      {
        printf("\n  [ ");
        for (j=0; j< 3; j++) {
          printf("%f ", rdata[i][j]);
        }
        printf("]");
      }
    printf("\n");

    return 0;
}
```

Program 6 (section 4.5.3) in Fortran

```fortran
!
!   This example copies a hyperslab region of 'SatelliteRange' under
!   '/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/' to another hyperslab region
!   '/SatelliteRange_Subset' under '/All_Data'. It then reads back
!   the newly created hyperslab region.
!    Main illustrative functions: H5LTread_region_f, H5LTcopy_region_f
!

PROGRAM main

  USE ISO_C_BINDING
  USE HDF5 ! module of HDF5 library
  USE H5LT ! module of H5LT
  USE H5HL_REGION

  IMPLICIT NONE

  CHARACTER(LEN=78), PARAMETER :: filename = &
       "GMGTO_npp_d20030125_t0657104_e0659047_b00014_c20090811150425926728_unkn_SCI.h5"

  CHARACTER(LEN=78), PARAMETER :: PATH_DEST = &
       "/Data_Products/SatelliteRange_Subset" ! Full path of the source dataset

  CHARACTER(LEN=78), PARAMETER :: PATH_SRC = &
       "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange" ! Full path to the destination dataset

  INTEGER, PARAMETER :: NRANK = 2 ! rank of source and destination dataset

  INTEGER(hid_t) :: file_id   ! file identifier
  INTEGER(hid_t) :: dset_id   ! dataset identifier
  INTEGER(hid_t) :: space_id  ! dataspace identifier
  INTEGER(hsize_t), DIMENSION(1:4) :: block_coord_src = (/52,4,54,8/) ! source's block coordinates (52,4)-(54,8)
  INTEGER(hsize_t), DIMENSION(1:4) :: block_coord_dest= (/2, 2, 4, 6/)    ! destination's BLOCK coordinates
(2,2)-(4,6)
  INTEGER(hsize_t), DIMENSION(1:NRANK) :: dims = (/5,7/) ! receiving dataset dimensions
  REAL, DIMENSION(1:3,1:5), TARGET :: rdata                   ! buffer to read destination data into
  INTEGER :: i, j
  INTEGER :: status
  TYPE(C_PTR) :: f_ptr


  !
  ! Initialize FORTRAN interface.
  !
  CALL H5open_f(status)
```

```
  !
  ! Open the NPP file.
  !
  CALL H5Fopen_f(filename, H5F_ACC_RDWR_F, file_id, status)
  !
  ! First create the destination dataset "SatelliteRange_Subset" since it does not exist.
  !
  ! NOTE: If the destination dataset does not already exist then H5LTcopy_region
  !        will automatically create a destination dataset that is the same size
  !        and rank of the source data and will, additionally, fill the destination
  !        block starting at (1,1) thus ignoring the destination's block coordinates.
  !
  CALL H5Screate_simple_f(NRANK, dims, space_id, status)
  CALL H5Dcreate_f(file_id, PATH_DEST, H5T_NATIVE_REAL, space_id, dset_id, status)

  CALL H5Sclose_f(space_id, status)
  CALL H5Fclose_f(file_id, status)
  !
  ! Copy a block of the "/All_Data/VIIRS-MOD-GTM-EDR-GEO_All/SatelliteRange"
  ! data with block corner coordinates of (52,4)-(54,8) to
  ! a subset of the "/Data_Products/SatelliteRange_Subset" with hyperslab
  ! coordinates (2,2)-(4,6).
  !
  !
  CALL H5LTcopy_region_f(filename, PATH_SRC, block_coord_src, filename, PATH_DEST, &
        block_coord_dest, status)
  !
  !
  ! Reads a subset region of "/Data_Products/SatelliteRange_Subset"
  ! specified by coordinates (2,2)-(4,6).
  !
  !

  f_ptr = C_LOC(rdata(1,1))
  CALL H5LTread_region_f(filename, PATH_DEST, block_coord_dest, H5T_NATIVE_REAL, &
        f_ptr, status)

  WRITE(*,'(/,"Subset     of     /Data_Products/SatelliteRange_Subset     with     coordinates     (",i1,",",i1,")-
(",i1,",",i1,"):")') block_coord_dest(1:4)

  DO i = 1, 3
     WRITE(*,'(A)', ADVANCE="NO") "["
     DO j = 1, 5
        WRITE(*,'(x,f12.4)', ADVANCE="NO") rdata(i,j)
     ENDDO
     WRITE(*,'(A)') " ]"
  ENDDO
  WRITE(*,'(/)')

END PROGRAM main
```

Program 7 (section 4.6.3) in C

```c
/*
 * This example opens a file, and extracts
 * the bit field from a subset of a dataset.
 * The values are returned as a base-10 integer.
 * Main illustrative function: H5LTread_bitfield_value
 */

#include <stdlib.h>
#include "hdf5.h"
#include "hdf5_hl.h"
#include "h5hl_region.h"

#define FILENAME "SVI01-GIMFG_NPP_d2003125_t101038_e10116_b9_c2005829153351_dev.h5"
#define DATASET "/All_Data/VIIRS-I1-SDR_All/QF_VIIRSI1SDR_Array"
#define num_flags 1

int main(void)
```

```
{
  int qf_data[5][6][1];  /* Read buffer */
  unsigned int offset[1] = {1};  /* Starting bits to be extracted from element */
  unsigned int length[1] = {2};  /* Number of bits to be extracted for each value */
  hid_t file, space; /* Handles */
  hid_t qf_dset;
  herr_t status;
  hsize_t start[3] =  {0,0,0};
  hsize_t count[3] =  {5,6,1};
  int i, j;

  /*
   * Open file.
   */
  file = H5Fopen (FILENAME, H5F_ACC_RDONLY, H5P_DEFAULT);
  /*
   * Open the data set
   */
  qf_dset = H5Dopen (file, DATASET, H5P_DEFAULT);
  /*
   * Get dataspace and allocate memory for read buffer. Quality flags dataset
   * has the same dimensionality as corresponding product dataset;
   * we are using its dimensions for illustration purposes only.
   */
  space = H5Dget_space (qf_dset);
  status = H5Sselect_hyperslab(space, H5S_SELECT_SET, start, NULL, count, NULL);
  /*
   * For each element read the value that takes first two bits and
   * store it in a char buffer. This selects all the elements (H5S_ALL)
   */
  status = H5LTread_bitfield_value(qf_dset, num_flags, offset, length, space, (int *)qf_data);
  status = H5Sclose (space);

  /* Print out the bit field */
  printf("Bit Field:\n");
  for (i = 0; i<5; i++) {
    printf (" [");
    for (j = 0; j<6; j++) {
        printf(" %d ", qf_data[i][j][0]);
    }
    printf("]\n");
  }

  return 0;

}
```

Program 7 (section 4.6.3) in Fortran

```
!
! This example opens a file, and extracts
! the bit field from a subset of a dataset.
! The values are returned as a base-10 integer.
! Main illustrative function: H5LTread_bitfield_value_f
!
PROGRAM main

  USE ISO_C_BINDING
  USE HDF5 ! module of HDF5 library
  USE H5LT ! module of H5LT
  USE H5HL_REGION

  IMPLICIT NONE

  CHARACTER(LEN=78), PARAMETER :: filename = &
        "SVI01-GIMFG_NPP_d2003125_t101038_e10116_b9_c2005829153351_dev.h5"

  CHARACTER(LEN=78), PARAMETER :: DATASET = &
        "/All_Data/VIIRS-I1-SDR_All/QF_VIIRSI1SDR_Array"
```

The HDF Group

```fortran
  INTEGER, PARAMETER :: num_flags = 1

  INTEGER, DIMENSION(1:3) :: dim_qf_data =(/6,5,num_flags/) ! Dimensions of qf_data
  INTEGER,  DIMENSION(1:num_flags,1:6,1:5) ::  qf_data  ! Read buffer, transposed because data written with C
convention
  INTEGER, DIMENSION(1:1) :: offset = (/2/)  ! Starting bits to be extracted from element
  INTEGER, DIMENSION(1:1) :: length = (/2/)  ! Number of bits to be extracted for each value
  INTEGER(hid_t) :: file, space ! Handles
  INTEGER(hid_t) :: qf_dset
  INTEGER :: status
  INTEGER(hsize_t), DIMENSION(1:3) :: start = (/0,0,0/)
  INTEGER(hsize_t), DIMENSION(1:3) :: count = (/num_flags,6,5/)
  INTEGER :: i, j
  !
  ! Initialize FORTRAN interface.
  !
  CALL H5open_f(status)
  !
  ! Open file.
  CALL H5Fopen_f(FILENAME, H5F_ACC_RDONLY_F, file, status)
  !
  ! Open the data set
  CALL H5Dopen_f(file, DATASET, qf_dset, status)
  !
  ! Get dataspace and allocate memory for read buffer. Quality flags dataset
  ! has the same dimensionality as corresponding product dataset
  ! we are using its dimensions for illustration purposes only.

  CALL H5Dget_space_f(qf_dset, space, status)
  CALL H5Sselect_hyperslab_f(space, H5S_SELECT_SET_F, start, count, status)


  !
  ! For each element read the value that takes first two bits and
  ! store it in a char buffer. This selects all the elements (H5S_ALL)

  CALL H5LTread_bitfield_value_f(qf_dset, num_flags, offset, length, space, qf_data, dim_qf_data, status)

  CALL H5Sclose_f(space, status)

  ! Print out the bit field
  WRITE(*,'("Bit Field:")')
  DO i = 1, 6
     WRITE( *, '(A)', ADVANCE = "NO") "["
     DO j = 1, 5
        WRITE( *, '(A)', ADVANCE = "NO") " {"
        WRITE( *, '(20i2)', ADVANCE = "NO") qf_data(1,i,j)
        WRITE( *, '(A)', ADVANCE = "NO") "} "
     ENDDO
     WRITE(*,'(A)',ADVANCE="YES") "]"
  ENDDO
END PROGRAM main
```

References

1. "HDF5 Tutorial", The HDF Group, http://www.hdfgroup.org/HDF5/Tutor/

2. "HDF Tools", The HDF Group, http://www.hdfgroup.org/tools/

3. "HDF Image and Palette Specification", The HDF Group,
    http://www.hdfgroup.org/HDF5/doc/ADGuide/ImageSpec.html

4. "HDF5 Dimension Scale Specification and Design Notes", The HDF Group,
    http://www.hdfgroup.org/HDF5/doc/HL/H5DS_Spec.pdf

5. "HDF5 High-level Functions for Region References, Hyperslabs, and Bit-fields", The HDF Group,
    http://www.hdfgroup.org/projects/npoess/HL_NPOESS/doc/index.html

6. "HDF5 User's Guide", The HDF Group,

    http://www.hdfgroup.org/HDF5/doc/UG/index.html

7. "NPP/ NPOESS Product Data Format", *Richard E. Ullman, HDF and HDF-EOS Workshop XI, 2007, Landover, Maryland,*
    http://www.hdfeos.net/workshops/ws11/presentations/day2/NPOESS-Format-Talk.ppt

8. "HDF Group Support for NPP/NPOESS", *Mike Folk, HDF and HDF-EOS Workshop XII, 2008, Aurora, Colorado, http://www.hdfeos.net/workshops/ws12/presentations/day3/mxf.ppt*