RFC: Reading Bit-field Values from NPOESS Product File

Elena Pourmal
M. Scot Breitenfeld

This RFC describes a helper API that extracts bit-field values from a dataset stored in an NPOESS product file.

1 Introduction

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 1. The IST quality flag takes two bits, while "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.

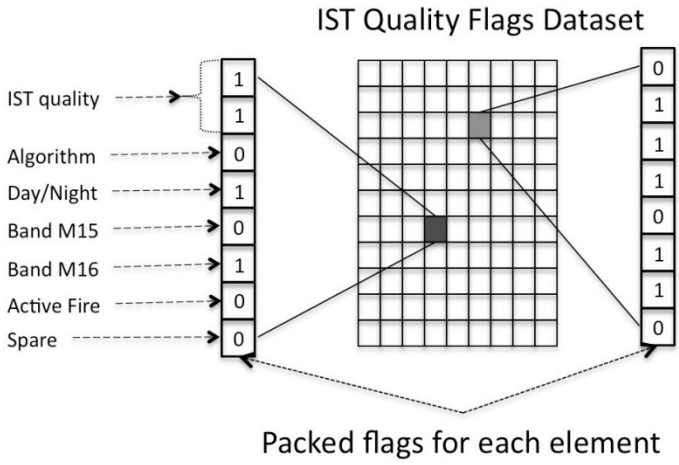

Figure 1

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

Section 2 describes several use cases for the proposed function.

Please notice that the function doesn't retrieve the name or doesn't provide the meaning for the bit-field value. For example, if `H5LTread_bitfield_value` reads the first two bits from an element of the IST Quality Flags dataset, the corresponding "IST quality" description and meaning of each value such as "High" for 0, "Medium" for 1, "Low" for 2, and "No Retrieval" for 3 are not available to the function since this information is stored in the XML form in the user's block.

It would seem to be very useful to have a special function that provides this capability. For instance, a function might be created that parses an XML document and returns the description of the quality flag represented by the bit-field value. At this point The HDF Group developers do not have enough background information to propose a specific API to read NPOESS quality flags information. We solicit the reader's ideas and opinions about such a function.

2 Use cases

1. A User needs to interpret the Ice Surface Temperature (IST) data from an NPOESS EDR file according to a specific algorithm. Information about the algorithm for each IST data element is stored in a third bit of a quality flag element in a corresponding quality flags dataset. The User calls `H5LTread_bitfield_value` to read the values of quality flags and to find the algorithm for each element of the IST data. He checks the read bit-field value, and if it is 0, then the "2-Band Split Window Baseline" algorithm will be applied to the corresponding element of the IST data, if the bit-field value is 1, then the "Single-band (12 micrometer) Fallback" algorithm will be applied to the corresponding element of the IST data.
2. A User provides a special flag to the h5dump utility to display bit-field value, which takes two first bits of each byte, stored in the IST quality flags dataset. H5dump prints values of 0,1,2, or 3 (the values correspond to the "High", "Medium", "Low" or "No Retrieval" legends in the quality flags description document, but this information is not available to h5dump).
3. A User starts HDFView, opens an NPOESS EDR file and a dataset with quality flags. HDFView spawns several windows and displays spreadsheets for each bit-field stored.
4. A User starts HDFView and opens an NPOESS EDR file. He finds a dataset with the IST product data. In a menu the User chooses a tab to show quality flags and clicks on "Active Fire" quality flag. After that he opens the IST dataset and only elements for which the value of the "Active Fire" quality flag is 1 are shown. The rest are filled with a specified fill value.

3 Sample Reference Manual Entry

This section describes the proposed API.

Name: H5LTread_bitfield_value

Signature:

herr_t H5LTread_bitfield_value(*hid_t* dset_id, *int* num_values,
const unsigned *offsets, *const unsigned* *lengths,
hid_t space , *int* *buf)

Purpose:

Retrieves the values of quality flags for each element to the application provided buffer.

Description:

H5LTread_bitfield_value reads selected elements from a dataset specified by its identifier dset_id, and unpacks the bit-field values to a buffer buf.

The parameter space is a space identifier that indicates which elements of the dataset should be read. To read all elements use H5S_ALL.

The parameter offsets is an array of length num_values; i^{th} element of the array holds the value of the starting bit of the i^{th} bit-field value. The parameter lengths is an array of length num_flags; i^{th} element of the array holds the number of bits to be extracted for the i^{th} bit-field value. Extracted bits will be interpreted as a base-2 integer value. Each value will be converted to the base-10 integer value and stored in the application buffer. Buffer buf is allocated by the application and should be big enough to hold num_sel_elem*num_values elements of the specified type, where num_sel_elem is a number of the elements to be read from the dataset. Data in the buffer is organized as num_values values for the first element, followed by the num_values values for the second element, … , followed by the num_values values for the $num\_selected\_elem^{th}$ element.

Parameters:

hid_t dset_id	IN: Identifier of the dataset with bit-field values
int num_values	IN: Number of the values to be extracted
const unsigned *offsets	IN: Array of staring bits to be extracted from the element (may have values from 0 to 7)
const unsigned *lengths	IN: Array of the number of bits to be extracted for each value
hid_t space	IN: dataspace identifier; describes the elements to be read from the dataset with bit-field values
int *buf	OUT: buffer to read the values in

Returns:

Returns a non-negative value if successful; otherwise returns a negative value.

4 Code Example

The example below reads bit-field values corresponding to the IST and “Active Fire” quality flags and prints them to the standard output.

```
#include <hdf5_hl.h>

main()
{
 …
    int *qf_data;
    int offsets[] = {0, 6};
    int lengths[] = {2, 1};
    num_values = 2;

   /*
    * Open NPOESS product file and a granule dataset.
    */
    file = H5Fopen (“NPOESS EDR”, H5F_ACC_RDONLY, H5P_DEFAULT);
    dset = H5Dopen (file, “Granule 1”);

    /*
     * Get dataspace and allocate memory for read buffer. Quality flags dataset
     * has the same dimensionality as corresponding product dataset; we
     * we are using its dimensions for illustration purposes only.
     */
    space = H5Dget_space (dset);
    ndims = H5Sget_simple_extent_dims (space, dims, NULL);
    qf_data = (char *) malloc (num_values * dims[0] * dims[1] * sizeof (char));
    status = H5Sclose (space);

    qf_dset = H5Dopen (file, “IST Quality Flags”);
    /*
     * For each element read the IST quality flag that takes first two bits and
     * store it in a char buffer.
     */
    status = H5LTread_bitfield_value (qf_set, num_values, offsets, lengths,
             H5S_ALL, qf_data);

     /*
      * Print extracted values for the elements with the indices (0, 4) and (1,1)
      * See also the output values for each quality flag.
      */
 (3, 1)  (2, 0)
  …
 }
```

IST quality flag:

0 0 1 2 **3** 0

3 **2** 0 1 1 1

2 1 3 2 0 0

Fire quality flag:

0 0 1 1 **1** 0

0 **0** 0 1 0 0

1 1 0 0 1 1

…..

References

1. "Profile of National Polar-Orbiting Operational Satellite System (NPOESS) HDF5 Files", *Kim Tomashosky, Ken Stone, Pat Purcell, Ron Andrews, HDF and HDF-EOS Workshop X, 2006, Landover, Maryland,* *http://www.hdfeos.net/workshops/ws10/presentations/day3/Profile_of_NPOESS_HDF5_Files.ppt*
2. "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*
3. "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*

Acknowledgements

This work is supported by the NPOESS project.

Revision History

March 24, 2009:	Version 1 circulated for comment within The HDF Group.
March 26, 2009:	Version 2 circulated for comment within NPOESS developers.
April 7, 2009	*Datatype identifier was removed from the API signature; void buffer was replaced with int buffer based on Scot's comments.* Version 3 circulated for comment within NPOESS developers.

Comments should be sent to epourmal@hdfgroup.org or help@hdfgroup.org