Subject: Re: MOD43B3 Col.4 Processing. unexpected error message. why?
Posted by James Kuyper on Tue, 10 Jul 2007 19:55:51 GMT
View Forum Message <> Reply to Message

DirtyHarry wrote:
> Good day, everyone. Even though I am still in the swamp and my
> previous question on MOD04 is left unsolved, I have to process MOD43B3
> albedo product. I am trying to make image files from MOD43B3 hdf files
> now.
>
> I did this way so far.
>
> 0. Variable (initialization/declaration)
> 1. Get hdf file info.
> 2. Input map info (sin projection)
> 3. Convert SIN to TM Korea
>
> There was no problem in making images with one variable (black sky
> albedo), and then I added another similar variable, white sky albedo,
> in my new simulation. (This part is commented out and labed as part1
> and part2)
>
> However, I got unexpected error messages shown below.
>
> ----------
> Envi retrieve data: An error has occurred during processing.
> Error: "Array dimensions must be greater than 0". The result may be
> invalid.
> ----------
>
> These messages are too broad for me and I don't know how to handle
> this. In addition, the ' Envi_retrieve_data' must be an internal
> function... something like that. I cannot find this function in ENVI
> help.
>
> Any suggestions?

I do have a number of suggestions; unfortunately none of them will solve your problem. When I tried to duplicate it, I got a quite different, but equally mysterious error message. However, I'll make my suggestions anyway. Note: all error handling has been suppressed for the sake of clarity.

...
> FileID = HDF_OPEN(filename, /Read)
> sdFileID = HDF_SD_Start(filename, /Read) ;The returned
> value of this function is the SD ID of the HDF file

/Read is the default; you don't have to specify it.

HDF_Open() is unnecessary; as long as the only thing you're doing with
the file is using the SD interface, HDF_SD_Start() is sufficient.
However, for other reasons, as I'll explain below, I think it's better
to open the file using the HDFEOS interface:

```
FileID = EOS_GD_OPEN(filename)
EOS_EH_IDINFO(FileId, hdffid, sdFileID)
```

```
>      sdsID_albedo = HDF_SD_Select(sdFileID, 0)   ; Albedo
>      sdsID_qc     = HDF_SD_Select(sdFileID, 1)   ; QC
```

I strongly recommend that you do not hard-code those index numbers. A
future version of the code which creates that file might rearrange the
order of the SDSs, possibly as a consequence of adding new ones.
Instead, you should locate them by name:

```
idx_albedo = HDF_SD_NameToIndex(sdFileID, 'Albedo')
sdsID_albedo = HDF_SD_Select(sdFileID, idx_albedo)

idx_qc = HDF_SD_NameToIndex(sdFileID, 'Albedo_Quality')
sdsID_qc = HDF_SD_Select(sdFileID, idx_qc)
```

```
>      hdf_sd_getdata, sdsID_albedo, albedo
>      hdf_sd_getdata, sdsID_qc, qc
```

I would recommend calling HDF_SD_EndAccess for both SDSs, and
HDF_SD_End, here, because you're finished with the SD interface, and
doing so will free up a (very) small amount of memory.

You should retrieve the projection information from the file itself.
The MOD43B3 filespec indicates:

```
  Supported Grids:     Geographic Grid
                       Integerized Sinusoidal
                       Interrupted Goodes Homolosine
```

They're not currently using all three projection, but that statement
essentially reserves the right to change projections in the future.
They probably won't, but it's relatively easy to make use of whichever
projection they are using. This is why I recommended using the HDFEOS
inteface. Here's how to extract the grid and projection information
(error handling code suppressed for readibility):

```
gridID = EOS_GD_Attach(FileId, 'MOD_Grid_BRDF')
EOS_GD_GridInfo(gridID, xdimsize, ydimsize, upleft, lowright)
```

EOS_GD_ProjInfo(gridID, projcode, zonecode, spherecode, projparam)
; Since you're done with the file, you can close it, to free up a little memory:
EOS_GD_Close(FileId)

Use xdimsize, ydimsize in place of 1200,1200 below.

```
>     albedo_black = fltarr(1200, 1200)
>     albedo_white = fltarr(1200, 1200)
>     qc1 = ulonarr(1200, 1200)
>     qc2 = ulonarr(1200, 1200)
>
>     albedo_black[*, *]=albedo[0, 9, *, *]
>     albedo_white[*, *]=albedo[1, 9, *, *]
>
>     qc1[*,*] = qc[0,*,*]
>     qc2[*,*] = qc[1,*,*]
>
>     ;help, albedo_black, albedo_white, qc1, qc2
>
>
>     ;VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
VVVVV
>     ;V  V  V                                   V  V  V
>     ;V  V  V      1. Input map Information(SIN projection)   V  V  V
>     ;V  V  V      2. Convert SIN to TM Korea                V  V  V
>     ;V  V  V                                   V  V  V
>     ;VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
VVVVV
>
>  ;   1. Input map information (SIN)
>
>     mc=[0.5, 0.5, 11119968.509D, 4447338.766D]
```

mc = [0.5, 0.5, upleft]

```
>     ps=[926.6254331D, 926.6254331D]
```

ps = abs((lowright-upleft) / [xdimsize,ydimsize])

```
>     units   =envi_translate_projection_units('Meters')
>     params1=[6371007.181D, 0, 0, 0]
>
>     Projection_Name1 = 'SIN_MODIS'
>     map_info = ENVI_MAP_INFO_CREATE(type=16, name=Projection_Name1,
>  params=params1, $
>         UNITS = units, MC = mc, PS = ps)
```

```
map_info = ENVI_MAP_INFO_CREATE(type=proj, name=Projection_Name1, $
   params=params, UNITS = units, MC = mc, PS = ps)

...
>     envi_convert_file_map_projection, fid=albedo_black_map,
>  pos=pos_albedo_black, dims=dims, o_proj=OUT_Proj, $
>         o_pixel_size=[1000, 1000],out_name=out_name_albedo_black,
>  warp_method=2, r_fid=albedo_black_TM, $
>         resampling=0, background=0
```

It fails for me right at this point, complaining that it is "unable to
convert Arbitrary projection." I have no idea what that means..

```
...
>     ;---------------------------------------------------
>     ; Done with SDS, close the interface
>     HDF_SD_ENDACCESS, SDSID_albedo
```

You didn't end access to sdsID_qc.

```
...
>  Free_Lun, lun, /force
```

Haven't you already free'd that lun?
