**GOES-16 ADP File Format and IDL Reader**

**3/28/2018**

The GOES-16 aerosol detection product (ADP) is **beta** product, which has not been fully validated, and therefore is not recommended for scientific use.

The GOES-16 aerosol detection product (ADP) files are in NetCDF4 format with naming convention as:

“**OR\_ABI-L2-ADPC-M3\_G16\_s*YYYYDDDHHMMSST\_eYYYYDDDHHMMSST\_cYYYYDDDHHMMSST.nc***”

Where **ADPC** is for CONUS. The corresponding string for full disk is **ADPF**. **M3** means satellite scan mode 3. It can also be **M4** for scan mode 4. ‘***YYYY***’ is four-digit year, ‘***DDD***’ is three-digit day of the year, ‘***HH***’ is two-digit hour, ‘***MM***’ is two-digit minute, ‘***SS***’ is two-digit second, ‘***T***’ is the tenth of second. ‘***sYYYYDDDHHMMSST***’ is the granule starting time, ‘***eYYYYDDDHHMMSST***’ is the ending time, and ‘***cYYYYDDDHHMMSST***’ is the file creation time. One example the file name:

OR\_ABI-L2-ADPC-M3\_G16 \_s20172471747154\_e20172471749527\_c20172471750265.nc

The GOES-16 ADP data has a spatial resolution of 2 km and a temporal resolution of 15 minutes for both CONUS full disk. The file contains following data fields: aerosol mask, smoke mask, dust mask, quality flag (DQF), as shown in Table 1. The dimension of the field is xsize=2500, ysize=1500 for CONUS, and xsize=ysize=5424 for full disk. The file also contains many metadata fields, whose descriptions can be found in Table 5.9.6-1 and Table 5.9.6-2 of GOES-R Product Definition and Users’ Guide ([http://www.goes-r.gov/products/docs/PUG-L2+-vol5.pdf](http://www.goes-r.gov/products/docs/PUG-L2%2B-vol5.pdf)).

Table 1. ADP file data fields

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Dimension** |
| aerosol | Aerosol mask. 0: aerosol absent; 1: aerosol present. | pixel (xsize, ysize) |
| Smoke | Smoke mask. 0: smoke absent; 1: smoke present | pixel (xsize, ysize) |
| Dust | Dust mask. 0: dust absent; 1: dust present | pixel (xsize, ysize) |
| DQF | Data Quality Flags.  |  pixel(xsize, ysize) |
| Bit 0: 0: good smoke detection retrieval 1:invalid smoke detection due to snow ice clouds or bad source data  |
| Bit 1: 0: good dust detection retrieval 1:invalid dust detection due to snow ice clouds or bad source data  |
| Bit 2-3:00: low confidence smoke detection10: medium confidence smoke detection11: high confidence smoke detection |
| Bit 4-5:00: low confidence dust detection 10: medium confidence dust detection11: high confidence dust detection |
| Bit 6: 0: outside sun-glint angle  1: inside sun-glint(not confident dust detection) |
| Bit 7:0: within valid solar/viewing zenith angle range with good detection 1: VZA>70 or SZA>60 (not confident detection for both smoke and dust) |

**IDL code to read ADP**

; call command in idl: read\_adp\_goes,fname,smoke\_con, dust\_con

; where fname is file name

; smoke\_con and dust\_con are output smoke and dust confidences

;

function read\_abi\_var,filename,var\_name, scaled

 print, filename

 id = ncdf\_open(filename)

 v\_id = ncdf\_varid(id, var\_name)

 if (v\_id eq -1) then begin

 print, var\_name + " does not exist"

 return, -1

 endif

 long\_name = ' '

 ;ncdf\_attget,id, v\_id, 'long\_name', long\_name

 print, "Getting " + string(var\_name)

 ncdf\_varget,id, v\_id, unscaled ; stored data

 test = NCDF\_VARINQ(id, v\_id)

 ; now we need to see if 'scale\_factor' is any of the attributes

 num\_attr = test.Natts

 data\_type = test.datatype

 if (scaled eq 1) then begin

 ncdf\_attget,id, v\_id, 'scale\_factor', scale\_factor

 ncdf\_attget,id, v\_id, 'add\_offset', add\_offset

 ncdf\_attget,id, v\_id, '\_FillValue', Fill\_value

 temp =unscaled

 unscaled = (temp \* scale\_factor) + add\_offset

 index = where(temp eq Fill\_value, count)

 if (count gt 0) then unscaled(index) = -999.0

 endif

 ncdf\_close, id

 return, unscaled

end

pro read\_adp\_goes,filename,smoke\_con,dust\_con

 dust = read\_abi\_var(filename,'Dust',0)

 Smoke = read\_abi\_var(filename,'Smoke',0)

 DQF=read\_abi\_var(filename,'DQF',0)

 smoke\_con=smoke

 dust\_con=dust

;confidence level for both smoke/dust (smoke\_con or dust\_con):

;0:-bad detection quality, 1-low confidence, 2-medium 3-high

 tmp\_mask=DQF

 idx = WHERE((((tmp\_mask AND 4) EQ 4) and (tmp\_mask AND 8) EQ 8) and smoke eq 1, COMPLEMENT=cidx, nc)

 print, 'smoke\_high=',nc

 IF nc GT 0 THEN smoke\_con[idx] = 3

 tmp\_mask=DQF

 idx = WHERE((((tmp\_mask AND 4) NE 4) and (tmp\_mask AND 8) EQ 8) and smoke eq 1, COMPLEMENT=cidx, nc)

 print, 'smoke\_med=',nc

 IF nc GT 0 THEN smoke\_con[idx] = 2

 tmp\_mask=DQF

 idx = WHERE((((tmp\_mask AND 4) NE 4) and (tmp\_mask AND 8) NE 8) and smoke eq 1, COMPLEMENT=cidx, nc)

 print, 'smoke\_low=',nc

 IF nc GT 0 THEN smoke\_con[idx] = 1

 tmp\_mask=DQF

 idx = WHERE((((tmp\_mask AND 16) EQ 16) and (tmp\_mask AND 32) EQ 32) and dust eq 1, COMPLEMENT=cidx, nc)

 ;print, nc

 IF nc GT 0 THEN dust\_con[idx] = 3

 tmp\_mask=DQF

 idx = WHERE((((tmp\_mask AND 16) NE 16) and (tmp\_mask AND 32) EQ 32) and dust eq 1, COMPLEMENT=cidx, nc)

 ;print, nc

 IF nc GT 0 THEN dust\_con[idx] = 2

 tmp\_mask=DQF

 idx = WHERE((((tmp\_mask AND 16) NE 16) and (tmp\_mask AND 32) NE 32) and dust eq 1, COMPLEMENT=cidx, nc)

 ;print, nc

 IF nc GT 0 THEN dust\_con[idx] = 1

 tmp\_mask=DQF

 idx = WHERE(((tmp\_mask AND 1) EQ 1), COMPLEMENT=cidx, nc)

 IF nc GT 0 THEN smoke\_con[idx] = 0

 tmp\_mask=DQF

 idx = WHERE(((tmp\_mask AND 2) EQ 2), COMPLEMENT=cidx, nc)

 IF nc GT 0 THEN dust\_con[idx] = 0

; bit 6 (sunglint)

 tmp\_mask=DQF

 idx = WHERE(((tmp\_mask AND 64) EQ 64), COMPLEMENT=cidx, nc)

 IF nc GT 0 THEN smoke\_con[idx] = 0

 IF nc GT 0 THEN dust\_con[idx] = 0

;Byte1, bit7 (solar and satellite angles)

 tmp\_mask=DQF

 idx = WHERE(((tmp\_mask AND 128) EQ 128), COMPLEMENT=cidx, nc)

 IF nc GT 0 THEN dust\_con[idx] = 0

idx=where(smoke\_con gt 3,ct)

smoke\_con(idx)=0b

 idx=where(dust\_con gt 3,ct)

 dust\_con(idx)=0b

end