MODIS Cloud Mask User's Guide

Written by Kathleen Strabala

TABLE OF CONTENTS

1. Purpose

The purpose of this document is to provide a practical guide on how to use the MODerate Resolution Imaging Spectroradiometer (MODIS) cloud mask.

2. What is MODIS?

MODIS is the keystone instrument onboard a series of satellites commissioned as part of NASA'S Earth Observing System (EOS). It's mission is to provide a long term record of science data products at high spatial and high spectral resolution. This record will be used to determine any changes that might be occurring in our earth/atmosphere system due to natural or man-made causes. This information will provide scientific evidence needed to make sound government policy decisions.

The 36 channel MODIS instrument provides the necessary high spatial (1 km) and high spectral resolution from the short wave visible to the long wave infrared. For more detailed information on the instrument, please refer to the NASA MODIS home page at: http://ltpwww.gsfc.nasa.gov/MODIS/MODIS.html. The MODIS bandwidths are listed in Table 1; bandwidths used as part of the cloud mask algorithm are also noted..

Table 1. MODIS spectral band number and central wavelength. Columns 3 and 4 indicate if the channel is used in the cloud masking and its primary application.

Band	**Wavelength (μm)**	**Used in Cloud Mask**	
1 (250 m)	0.659	Y	(250m and 1km) clouds, shadow
2 (250 m)	0.865	Y	(250m and 1 km) low clouds
3 (500 m)	0.470	N	
4 (500 m)	0.555	N	snow
5 (500 m)	1.240	Y	snow
6 (500 m)	1.640	Y	snow, shadow
7 (500 m)	2.130	N	
8	0.415	N	
9	0.443	N	
10	0.490	N	
11	0.531	N	
12	0.565	N	
13	0.653	N	
14	0.681	N	
15	0.750	N	
16	0.865	N	
17	0.905	N	
18	0.936	Y	low clouds
19	0.940	Y	shadows
26	1.375	Y	thin cirrus
20	3.750	Y	shadow
21/22	3.959	Y(21)/N(22)	window
23	4.050	N	
24	4.465	N	

25	4.515	N	
27	6.715	Y	high moisture
28	7.325	N	
29	8.550	Y	mid moisture
30	9.730	N	
31	11.030	Y	window
32	12.020	Y	low moisture
33	13.335	N	
34	13.635	N	
35	13.935	Y	high cloud
36	14.235	N	

3. What is the MODIS Cloud Mask?

The MODIS cloud mask is a science data product that will be produced regularly as an Earth Observing System (EOS) standard product. It's main purpose is to identify scenes where land, ocean and atmosphere products should be retrieved based upon the amount of obstruction of the surface due to clouds and thick aerosol.

As are all official EOS data products, the MODIS cloud mask is created in Hierarchical Data Format (HDF) and consists of nine Scientific Data Set (SDS) objects. Seven of the SDS's are subsets of the MODIS geolocation product (MOD03). These have been included mainly for visualization purposes. The Cloud_Mask SDS actually contains the 48 bit cloud mask product. Along with information on surface obstruction, other factors affecting surface and cloud retrievals are also provided as ancillary information, such as non-terrain shadows and sun glint. Individual spectral test results which were used in the final product determination are also included as part of the product. Finally, the last two bytes of data represent a 250 m binary cloud mask created for anyone performing retrievals using the two 250 m channels (.67 and .86 microns). A bit-by-bit description of the MODIS cloud mask SDS is provided in Table 2. A 10 byte array of data (Quality_Assurance SDS) is also included which provides information on product quality, test implementation information as well as a record of what ancillary data sets were used in the generation of the product for each individual field-of-view. A bit-by-bit description of the quality assurance SDS is provided in Table 3.

Table 2. 48 bit MODIS CLOUD MASK SDS PRODUCT

BIT FIELD	DESCRIPTION KEY	RESULT
0	Cloud Mask Flag	0 = not determined 1 = determined
1-2	Unobstructed FOV Quality Flag	00 = cloudy 01 = uncertain clear 10 = probably clear 11 = confident clear
PROCESSING PATH FLAGS		
3	Day / Night Flag	0 = Night / 1 = Day
4	Sun glint Flag	0 = Yes / 1 = No
5	Snow / Ice Background Flag	0 = Yes/ 1 = No
6-7	Land / Water Flag	00 = Water 01 = Coastal 10 = Desert 11 = Land
ADDITIONAL INFORMATION		
8	Non-cloud obstruction Flag (heavy aerosol)	0 = Yes / 1 = No

9	Thin Cirrus Detected (near infrared)	0 = Yes / 1 = No
10	Shadow Found	0 = Yes / 1 = No
11	Thin Cirrus Detected (infrared)	0 = Yes / 1 = No
12	Spare (Cloud adjacency)	(post launch)
1-km INDIVIDUAL TEST CLOUD FLAGS (0 CAN ALSO MEAN TEST NOT APPLIED)		
13	Cloud Flag - simple IR Threshold Test	0 = Yes / 1 = No
14	High Cloud Flag - CO_2 Threshold Test	0 = Yes / 1 = No
15	High Cloud Flag - 6.7 μm Test	0 = Yes / 1 = No
16	High Cloud Flag - 1.38 μm Test	0 = Yes / 1 = No
17	High Cloud Flag - 3.9-12 μm Test	0 = Yes / 1 = No
18	Cloud Flag - IR Temperature Difference	0 = Yes / 1 = No
19	Cloud Flag - 3.9-11 μm Test	0 = Yes / 1 = No
20	Cloud Flag - Visible Reflectance Test	0 = Yes / 1 = No
21	Cloud Flag - Visible Ratio Test	0 = Yes / 1 = No
22	Cloud Flag - Near IR Reflectance Test	0 = Yes / 1 = No
23	Cloud Flag - 3.7-3.9 μm Test	0 = Yes / 1 = No
ADDITIONAL TESTS		
24	Cloud Flag - Temporal Consistency	0 = Yes / 1 = No
25	Cloud Flag - Spatial Variability	0 = Yes / 1 = No
26-31	Spares	
250-m CLOUD FLAG - VISIBLE TESTS (0 CAN ALSO MEAN TEST NOT APPLIED)		
32	Element(1,1)	0 = Yes / 1 = No
33	Element(1,2)	0 = Yes / 1 = No
34	Element(1,3)	0 = Yes / 1 = No
35	Element(1,4)	0 = Yes / 1 = No
36	Element(2,1)	0 = Yes / 1 = No
37	Element(2,2)	0 = Yes / 1 = No
38	Element(2,3)	0 = Yes / 1 = No
39	Element(2,4)	0 = Yes / 1 = No
40	Element(3,1)	0 = Yes / 1 = No
41	Element(3,2)	0 = Yes / 1 = No
42	Element(3,3)	0 = Yes / 1 = No
43	Element(3,4)	0 = Yes / 1 = No
44	Element(4,1)	0 = Yes / 1 = No
45	Element(4,2)	0 = Yes / 1 = No
46	Element(4,3)	0 = Yes / 1 = No
47	Element(4,4)	0 = Yes / 1 = No

Table 3. Bit Description of MODIS Cloud Mask Quality Assurance SDS

Product quality QA flags			
Cloud Mask QA (1km)	1	0 1	not useful useful
Cloud Mask Confidence QA (1km)	3	0-7	8 confidence levels
Spares	4		
--------------------------------End Byte 1 ---			
Processing QA flags - Individual test application			
NCO test	1	0 1	Not Applied Applied
Thin Cirrus test (Solar)	1	0 1	Not Applied Applied
Shadow Detection tests	1	0 1	Not Applied Applied
Thin Cirrus test (IR)	1	0 1	Not Applied Applied
Cloud Adjacency Test	1	0 1	Not Applied Applied

IR Threshold test	1	0 1	Not Applied Applied
High Cloud Test (CO2)	1	0 1	Not Applied Applied
High Cloud Test (6.7 µm)	1	0 1	Not Applied Applied
---------------------------------End Byte 2---			
High Cloud Test (1.38 µm)	1	0 1	Not Applied Applied
High Cloud Test (3.7-12µm)	1	0 1	Not Applied Applied
IR Temperature Difference Tests	1	0 1	Not Applied Applied
3.7-11µm Test	1	0 1	Not Applied Applied
.68 Reflectance Test	1	0 1	Not Applied Applied
Visible Ratio Test	1	0 1	Not Applied Applied
Near IR Reflectance Ratio Test	1	0 1	Not Applied Applied
3.7-3.9 µm Test	1	0 1	Not Applied Applied
---------------------------------End Byte 3---			
Temporal Consistency Test	1	0 1	Not Applied Applied
Spatial Variability Test	1	0 1	Not Applied Applied
Spare	6		
---------------------------------End Byte 4---			
250 m Visible Tests (Repeated 16 times)	1(16)	0 1	Not Applied Applied
---------------------------------End Byte 5 and 6--			
Processing QA flags - Input data information flags			
Number of bands used to generate cloud mask	2	0 1 2 3	None 1-7 8-14 15-21
Number of spectral tests used to generate cloud mask	2	0 1 2 3	None 1-3 4-6 7-9
Spares	4		
---------------------------------End Byte 7---			
Processing QA flags - Input data resource flags			
Clear Radiance Origin	2	0 1 2 3	MOD35 Forward calculation from NCEP GDAS model Other Not Used
Surface Temperature Over Land	2	0 1 2 3	NCEP GDAS DAO MOD11 Other
Surface Temperature Over Ocean	2	0 1 2 3	Reynolds blended DAO MOD28 Other
Surface Winds	2	0 1 2 3	NCEP GDAS DAO Other Not Used
---------------------------------End Byte 8---			

Ecosystem Map	2	0 1 2 3	Loveland NA 1km Olson Ecosystem MOD12 Other
Snow mask	2	0 1 2 3	MOD33 SSMI product Other Not used
Ice cover	2	0 1 2 3	MOD42 SSMI product Other Not used
Land/Sea Mask	2	0 1 2 3	USGS 1 km 6 level USGS 1 km binary Other Not used
--------------------------------End Byte 9--------------------------------			
Digital Elevation Model	1	0 1	EOS DEM Not used
Precipitable Water	2	0 1 2 3	NCEP GDAS DAO MOD07 Not used
Spare	5		
--------------------------------End Byte 10--------------------------------			

4. How do I use the MODIS Cloud Mask?

The MODIS cloud mask is much more than just a binary cloud/no cloud identifier. It was designed to meet the needs of a wide variety of users. Some, such as those interested in retrieving Sea Surface Temperature (SST), are very sensitive to any type of cloud contamination. Others are less so. Some can tolerate or correct for thin cirrus contamination, others cannot. Some cloud retrieval algorithms are applied differently for land or water surfaces. Some will not perform retrievals in sun glint regions. Some algorithm developers have certain spectral tests which they have more confidence in than others. All of these individual needs had to be considered when developing the MODIS cloud mask.

To use the mask effectively, you must understand your tolerance for cloud or clear when performing a retrieval. Once that decision has been made, look over the product description to see which bits you will need to extract from the file.

4.1 How Can I Determine if a Particular Cloud Test Has Been Applied?

Lots of confusion exists over this topic. The individual test results and additional information tests (bits 8-25 and 32-47) have a binary value of 0 or 1 representing cloud found or cloud not found. There will be times when some tests may not be applied, and this will also result in a bit value of 0. This could be due to the background making it inappropriate to apply certain tests (such as visible reflectance tests applied to snow scenes or nighttime scenes), or because a given channel had bad data. . Therefore, three pieces of information are being stored in a binary storage holder. Knowing which tests are applied for a given processing path will resolve most of this confusion. A table has been created which outlines which tests are applied to which processing paths. In this way, users can access the processing path bits and know when tests will be applied

Table 4. MODIS cloud mask tests executed (✓) for a given processing path.

MODIS Cloud Mask Test Layout for a Given Processing Path

	Daytime Ocean	Nighttime Ocean	Daytime Land	Nighttime Land	Polar Day (snow)	Polar Night (snow)	Coastline Day	Coastline Night	Desert Day	Desert Night
BT_{11} (Bit 13)	✓	✓								
$BT_{13.9}$ (Bit 14)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$BT_{6.7}$ (Bit 15)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$R_{1.38}$ (Bit 16)	✓		✓		✓		✓		✓	
$BT_{3.7} - BT_{12}$ (Bit 17)				✓		✓				✓
BT_{8-11} & BT_{11-12} (Bit 18)	✓	✓	✓	✓			✓	✓	✓	✓
$BT_{3.7} - BT_{11}$ (Bit 19)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$R_{.66}$ or $R_{.87}$ (Bit 20)	✓		✓				✓			
$R_{.87}/R_{0.66}$ (Bit 21)	✓		✓							
$R_{.935}/R_{.87}$ (Bit 22)	✓		✓		✓		✓			
$BT_{3.7} - BT_{3.9}$ (Bit 23)	✓		✓				✓		✓	
Temporal Consistency (Bit 25)	✓	✓								✓
Spatial Variability (Bit 25)	✓	✓								

without having to access any other SDS data sets from the cloud mask HDF file. Of course, if you want to know explicitly whether a test was performed on a given field-of-view, the information exists as part of the cloud mask Quality_Assurance SDS, in bytes 2 through 6.

Table 4 shows which tests are executed for the given processing paths. Table 3 displays the meaning of the bits in the cloud mask Quality_Assurance SDS. Please refer to bytes 2-6 for the location of particular test applied results.

4.2 How can I Determine if the Product is of Good Quality?

Through a number of pre-launch validation studies, the algorithm itself has proven to be robust and give results that are of good quality when applied to Advanced Very High Resolution Radiometer (AVHRR) and MODIS Airborne Simulator (MAS) data (See Ackerman, et al., 1998). The expectation is that the MODIS product will be of similar quality. None the less, there is an attempt to say something about the quality of the cloud mask product for each field-of-view in the Quality_Assurance SDS and for the entire granule of data as part of the product metadata.

4.2.1 Product quality on a field-of-view basis

The Qaulity_Assurance SDS contains information about the quality of the MODIS cloud mask based upon the number of individual spectral tests that are actually executed for the given field-of-view. The logic being that more tests will yield a better result. This implies that processing paths, such as polar night and desert which have less tests associated with them will potentially be of lesser quality than other processing paths such as ocean day or land day paths. This method also takes into account times when bad data may be affecting the results. Bad radiometric data in bands used in the production of the MODIS cloud mask will mean less tests applied in a given processing path, thus lessening the confidence in the quality of the product.

Byte 1 of the Quality_Assurance SDS in the cloud mask HDF product is devoted to the quality of the cloud mask product for a given field-of-view (Table 3). The first bit indicates whether the product is useful or not (bit equal to 0 or 1). A value of zero means no tests were used in the production of the product, and the first bit of the Cloud_Mask SDS should also be zero for this 1 km field-of-view. The product should not be used if this is the case. The next three bits of the first byte of the Quality_Assurance SDS represent 8 different confidence levels in the final quality of the product for each field-of-view (Table 3) If the number of tests is:

$X = 0$ then the confidence in the quality of the product is 0 (no confidence, don’t use)
$0 < X < 3$ then the confidence in the quality of the product is 4 (intermediate confidence)

4 < X < 7 then the confidence in the quality of the product is 6 (high confidence)
X >= 7 then the confidence in the quality of the product is 7 (very high confidence)

4.2.2 *Product quality on a granule basis*

Information on the quality of the MODIS cloud mask on a per granule basis is provided through EOS Core System (ECS) Inventory Metadata as values in the global attribute CoreMetadata.0 (See File Specification in Appendix A). The Measured Parameter AUTOMATICQUALITYFLAG is set to “passed” or “failed” within the cloud mask production software depending upon the successful number of cloud mask retrievals made. If the successful retrieval percentage was less than 10 % then the granule is said to be unusable and the value is set to “failed”, otherwise it is set to “passed”. This is what is defined in the AUTOMATICQUALITYFLAGEXPLANATION Measured Parameter. The Parameter QAPERCENTMISSINGDATA contains the value of 100 % - successful retrieval %. Finally the value of the successful retrievals percentage is contained in the Inventory Product Specific Attribute 1 SuccessfulRetrievalPct.

One more parameter OPERATIONALQUALITYFLAG will be set to “passed” or “fail” once the granule has been perused by the MODIS cloud mask team. A comment by the quality analyst will be provided in the OPERATIONALQUALITYFLAGEXPLANTION metadata. This is not an automated flag set within the software itself; it is a flag that may be set some time after the production of the product.

4.3 What if I Just Want to Know if the Scene is Clear or Cloudy?

If you really want to know our best estimate on whether there is cloud or not, then read bits one and two. Use the probably clear/undecided breakpoint for your cloud and clear decision. Again, this may be appropriate for some applications, but not for others.

4.4 What if I am only Interested in REALLY clear Scenes?

Certain applications have little tolerance for cloud or shadow contamination. This is an example of how these applications (e.g., bi-directional reflectance models) might interpret the cloud mask output.

1. Read bit 0 to determine if a cloud mask was determined; if 0, no further processing of the pixel is required.
2. If necessary, read bits 3 through 7 to determine scene domain, ie., do you care if there is snow in the scene or not?
3. Read the confidence in clear bits 1 and 2; if both bits are <u>not</u> equal to 1 (high confidence clear), then some tests are suggesting the presence of the cloud, and the pixel is skipped.

4. Read bit 9 to determine if a thin cirrus cloud is present (bit value of 0). An optically thin cirrus cloud may set bit 9 but not be classified as a cloudy scene.
5. Read bit 10 to determine if shadow contamination is present; do not process data if this bit is 0. You may want to check bit 10 out of the Quality_Assurance SDS to make sure the test was applied (bit value of 1).
6. Daytime algorithms may (depending on application) read bits 32 through 47 (250 m bits) to assess potential subpixel contamination or scene variability. Again, you can either check Table 4 to see if these tests are performed for the scene processing path, or check bytes Quality_Assurance SDS 5 and 6 for bit values of 1 (test applied).

4.5 What if I can Tolerate some Cloud Contamination in a Scene?

Some algorithms may be insensitive to the presence of thin cloud or may apply appropriate correction algorithms. This is a suggested application. An example is presented that might be appropriate for Normalized Difference Vegetation Index (NDVI).

1. Read bit 0 to determine if a cloud mask was determined; if 0, no further processing of the pixel is required.
2. Read bits 3 through 7 to determine if scene domain is appropriate (e.g., land and daytime).
3. Read the confidence flag bits 1 and 2. If cloudy (value of 00), do not process this pixel. A value of 01 for bits 1 and 2 (uncertain) often occurs around cloud edges and retrieving NDVI may not be appropriate with this confidence level. If both bits are equal to 1, then most tests are suggesting clear scenes; proceed with steps 4-7. If bits 1 and two (confidence bits) are 10, then detailed checking of bits 13 through 25 (individual test bit results) may be required to determine if the NDVI retrieval should proceed.
4. Read bit 9 to determine if a thin cirrus cloud is present (bit value of 0). An optically thin cirrus cloud may set bit 9 but not be classified as a cloudy scene. Some of the MODIS solar channels are not as sensitive to thin cirrus as the 1.38 μm band. If thin cirrus is detected, apply appropriate correction algorithms.
5. Check that reflectance tests (bits 20 and 21) did not detect cloud. Note that a value of 0 indicates that either a cloud is present or the test was not run. This test is not run if over snow or solar zenith angles greater than 85°. If you want to be sure, check bits 20 and 21 of the Quality_Assurance SDS to make sure the bits are 1 (test applied).
6. Read bit 10 to determine if shadow contamination is present. Shadows might bias the NDVI product. Again, you may want to make sure the shadow test was applied by checking bit 10 of the Quality_Assurance SDS (bit value of 1).
7. Read bits 32 through 47 to assess cloud contamination. This would not be recommended if snow is indicated. You can either check Table 4 to see if these tests are performed for the scene processing path, or check Quality_Assurance SDS bytes 5 and 6 for bit values of 1 (test applied).

4.6 What If I am Interested in REALLY Cloudy Scenes?

Use of the cloud mask for cloud scene processing may require a more in-depth analysis than clear-sky applications, as the mask is clear-sky conservative. An approach to interpreting the cloud mask for cloud property retrievals during the day over ocean scenes in non-sunglint regions is outlined.

1. Read bit 0 to determine if a cloud mask was determined; if 0, no further processing of the pixel is required.
2. Read bit 3; if 0, no further processing of the pixel is required (night).
3. Read bits 6 and 7; if 00 then proceed (water).
4. Read bit 4; if 0 then it is a sunglint region. The user may want to place less confidence on a product retrieval.
5. Read the confidence flag bits 1 and 2.
    - If confident clear (value of 11), read bit 9 to determine if a thin cirrus cloud is present (bit value of 0). An optically thin cirrus cloud may set bit 9 but not be classified as a cloudy scene. If thin cirrus is detected, apply appropriate algorithms or place less confidence on the product retrieval. If bit 9 is 1, then no further processing is required.
    - If both bits are equal to 00, then the scene is cloudy. Check bit 8 for possible heavy aerosol loading. If bit 8 is 0, then the pixel may be aerosol contaminated. In this case no further processing is necessary or place less confidence on the product retrieval.
    - If confidence is 10 or 01, then detailed checking of bits 13 through 25 may be required to determine if the retrieval algorithm should be executed. For example, if confidence bits are 10 and pixel is in a sun glint region, additional testing is advised.
6. Check how many tests detected cloud. The greater the number of tests that detected cloud, the more confidence one has in the cloud property product. Note that a value of 0 indicates that either a cloud is present or the test was not run. You can explicitly check to see if a test was run by checking the Qaulity_Assurance SDS for the appropriate test applied bits (See Table 3).
7. Check spatial variability test results.
8. Read bits 32 through 47 to assess subpixel cloud contamination. You can either check Table 4 to see if these tests are performed for the scene processing path, or check Quality_Assurance SDS bytes 5 and 6 for bit values of 1 (test applied).

5. Metadata

Information is provided about the cloud mask granule product in the form of metadata. Some of this information is required by the ECS, others are chosen by the algorithm developers. Some of the information may be useful to those who are looking for specific geographic locations or specific cloud properties to exist within a granule. Some of the metadata fields (ECS Core Metadata and ECS Inventory Metadata) can be used as searchable fields from the archive. As described earlier, general granule based product quality is also provided here. Table 5 lists the ECS required metadata for the MODIS cloud mask product.

Table 5. ECS Core Metadata of the MODIS V2 cloud mask product.

ECS Core Attribute Name	ECS Data Type	# of Values
SHORTNAME	STRING	1
VERSIONID	STRING	1
SIZEMBECSDATAGRANULE	DOUBLE	1
REPROCESSINGACTUAL	STRING	1
REPROCESSINGPLANNED	STRING	1
LOCALGRANULEID	STRING	1
LOCALVERSIONID	STRING	1
DAYNIGHTFLAG	STRING	1
PRODUCTIONDATETIME	DATETIME	1
PGEVERSION	STRING	1
INPUTPOINTER	STRING	30(Max)
RangeDateTime		
RANGEBEGINNINGTIME	TIME	1
RANGEENDINGTIME	TIME	1
RANGEBEGINNINGDATE	DATE	1
RANGEENDINGDATE	DATE	1
Bounding Rectangle		
EASTBOUNDINGCOORDINATE	DOUBLE	1
WESTBOUNDINGCOORDINATE	DOUBLE	1
NORTHBOUNDINGCOORDINATE	DOUBLE	1
SOUTHBOUNDINGCOORDINATE	DOUBLE	1
OrbitCalculatedSpatialDomain		
ORBITNUMBER.1	INTEGER	1
EQUATORCROSSINGLONGITUDE.1	DOUBLE	1
EQUATORCROSSINGDATE.1	DATE	1
EQUATORCROSSINGTIME.1	TIME	1
MeasuredParameter		
PARAMETERNAME.1	STRING	1
SCIENCEQUALITYFLAG.1	STRING	1
SCIENCEQUALITYFLAGEXPLANATION.1	STRING	1
OPERATIONALQUALITYFLAG.1	STRING	1
OPERATIONALQUALITYFLAGEXPLANATION.1	STRING	1
QAPERCENTMISSINGDATA.1	INTEGER	1

The Non-ECS QA inventory metadata are designed to report searchable statistics and information of interest to users, as shown in Table 6.

Table 6. Non-EOS Core System Quality Assurance Inventory Metadata. For the MODIS cloud mask.

Field Name	Data Type	# of Value	Value
AdditionalAttributeName.1	String	1	SuccessfulRetrievalPct
AdditionalAttributeName.2	String	1	VeryHighConfidenceClearPct
AdditionalAttributeName.3	String	1	HighConfidenceClearPct
AdditionalAttributeName.4	String	1	UncertainConfidenceClearPct
AdditionalAttributeName.5	String	1	LowConfidenceClearPct
AdditionalAttributeName.6	String	1	CloudCoverPct250m
AdditionalAttributeName.7	String	1	ClearPct250m
AdditionalAttributeName.8	String	1	DayProcessedPct
AdditionalAttributeName.9	String	1	NightProcessedPct
AdditionalAttributeName.10	String	1	SunglintProcessPct
AdditionalAttributeName.12	String	1	Snow_IceSurfaceProcessPct
AdditionalAttributeName.13	String	1	LandProcessedPct
AdditionalAttributeName.13	String	1	WaterProcessedPct
AdditionalAttributeName.14	String	1	ShadowProcessedPct
AdditionalAttributeName.15	String	1	ThinCirrusSolar_FoundPct
AdditionalAttributeName.16	String	1	ThinCirrusIR_FoundPct
AdditionalAttributeName.17	String	1	NonCloudObstructionPct
AdditionalAttributeName.18	String	1	MaxSolarZenithAngle
AdditionalAttributeName.19	String	1	MinSolarZenithAngle
AncillaryInputType	String	1	“Geolocation”
AncillaryInputTypePointer	String	1	UR of geolocation granule
PlatformShortName	String	1	“EOS-AM1”
InstrumentShortName	String	1	“MODIS”

The product specific QA archive metadata are designed to report the statistics and information that are only needed to be archived along with science data sets for each granule. They are provided in Table 7. These field will not be searchable.

Table 7. Product Specific Quality Assurance Archive Metadata for the MODIS cloud mask.

Field name	Data type	No. of value	Value
INSTRUMENTNAME	String	1	†
LONGNAME	String	1	!
ALGORITHMPACKAGEACCEPTANCE DATE	String	1	“June-1997”
ALGORITHMPACKAGEMATURITYCODE	String	1	“at-launch”
ALGORITHMPACKAGENAME	String	1	“ATBD-MOD-06”
ALGORITHMPACKAGEVERSION	String	1	“2”
LOCALINPUTGRANULEID	String	10	*
EXCLUSIONGRINGFLAG	String	M,1	variable
GRINGPOINTLATITUDE	Double	M,4	variable
GRINGPOINTLONGITUDE	Double	M,4	variable
GRINGPOINTSEQUENCENO	Integer	M,4	variable
Cloud_Mask_Algorithm_Version_Number	Integer	1	variable

† “Moderate Resolution Imaging Spectrometer”
! “MODIS cloud mask and spectral test results”
* “MODIS product inputs using MODIS naming convention”

6. References

Ackerman, S. A. ; Strabala, K. I. ; Menzel, W. P. ; Frey, R. A. ; Moeller, C. C. ; Gumley, L. E., 1998: Discriminating clear sky from clouds with MODIS. Journal of Geophysical Research, 103 , No. D24, 32141-32157.
http://cimss.ssec.wisc.edu/modis1/pdf/JGRFINAL.PDF

Ackerman, S. A., K. I. Strabala, W. P. Menzel, R. A. Frey, C. C. Moeller, L. E. Gumley, B. A. Baum, C. Schaaf, G. Riggs, 1997: Discriminating clear-sky from cloud with MODIS algorithm theoretical basis document (MOD35). EOS ATBD web site, 125 pp.
ftp://eospso.gsfc.nasa.gov/ATBD/REVIEW/MODIS/ATBD-MOD-06/atbd-mod-06.pdf

Chu, A., K. I. Strabala, R. Song, S. Platnick, M. Wang and S. Matoo, 1997: MODIS Atmosphere Quality Assurance Plan. MODIS Home Page, 44 pp.
http://modarch.gsfc.nasa.gov/MODIS/ATM/DOCS/atm_qaplan.pdf

King, M. D., Y. J. Kaufman, W. P. Menzel, D. Tanre and B.-C. Gao, 1999: MODIS Atmosphere Validation Plan. MODIS Home Page, 43 pp.
http://modarch.gsfc.nasa.gov/MODIS/ATBD/atm_val.pdf

APPENDIX A

Code for Reading the Cloud Mask

This is an example FORTRAN 77 program to read the MODIS cloud mask. The F77 subroutine returns one MODIS scan's worth of both the Cloud_Mask SDS and the Quality_Assurance SDS when invoked. This is a good example of how a user can design what they extract out of the cloud mask file based upon their needs. This code is the official MODIS cloud mask reader, and includes some MODIS and ECS Toolkit calls needed to run in the ECS environment.

```
================== Begin Example Cloud Mask Reader ====================
      SUBROUTINE Read_CldMsk(Modfil,Scan_No,
     &                Dim1_CM,Dim1_QA,Dim2,Dim3,
     &                DS_Dim1_CM,DS_Dim1_QA,DS_Dim2,DS_Dim3,
     &                CM,QA,Error_Flag)

C-------------------------------------------------------------------
C !F77
C
C !DESCRIPTION:
C
C   Read_CldMsk retrieves one scan of MODIS Cloud Mask and Cloud Mask
C   QA data from the MOD35 HDF product file.  It is assumed that the
C   spatial dimensions (number of frames and lines) of the HDF Cloud
C   Mask and QA SDSs are equal (See Design Notes below).
C
C !INPUT PARAMETERS:
C   INTEGER  Modfil      M-API file handle structure for HDF files
C
C   INTEGER  Scan_No     1-based instrument scan number
C
C   INTEGER  Dim1_CM     Size of dimension 1 of Cloud Mask buffer as
C                   dimensioned in calling program
C
C   INTEGER  Dim1_QA     Size of dimension 1 of Cloud Mask QA buffer
C                   as dimensioned in calling program
C
C   INTEGER  Dim2/Dim3   Size of dimensions 2 and 3 of Cloud Mask and
C                   Cloud Mask QA buffers as dimensioned in the
C                   calling program.
C
C !OUTPUT PARAMETERS:
C   BYTE     CM          Three dimensional (3-D) array for passing
C                   cloud mask data.  Index 1 is byte number,
C                   index 2 is (1-km) frame number, and index 3 is
C                   relative (1-km) line number within scan.
C
C   BYTE     QA          Three dimensional (3-D) array for passing cloud
C                   mask quality assurance data.  Index 1 is QA byte
C                   number, index 2 is (1-km) frame number, and
C                   index 3 is relative (1-km) line number within
C                   scan.
C
C   INTEGER  DS_Dim1_CM  Size of retrieved Cloud Mask data block along
C                   dimension 1 of output buffer.  It is as large
```

```
C                    as the byte dimension of the HDF Cloud Mask
C                    SDS
C
C   INTEGER  DS_Dim1_QA  Size of retrieved Cloud Mask QA data block
C                    along dimension 1 of output buffer.  It is
C                    as large as the byte dimension of the HDF Cloud
C                    Mask SDS
C
C   INTEGER  DS_Dim2     Size of retrieved Cloud Mask and QA data
C                    blocks along dimension 2 of output buffers.
C                    It is as large as the frame (across track)
C                    dimension of the HDF Cloud Mask and QA SDS
C                    data arrays, which are assumed equal.
C
C   INTEGER  DS_Dim3     Size of retrieved Cloud Mask and QA data
C                    blocks along dimension 3 of output buffers.
C                    It is equal to 10, the number of 1-km
C                    detector lines in a MODIS instrument scan.
C
C   LOGICAL  Error_Flag  variable that is set to .TRUE. if an error is
C                    detected. It is set to .FALSE. if no errors
C                    are identified.
C
C !REVISION HISTORY:
C  $Log$
C !TEAM-UNIQUE HEADER:
C
C    This software is developed by the MODIS Science Data Support
C    Team for the National Aeronautics and Space Administration,
C    Goddard Space Flight Center, under contract NAS5-32373.
C
C !REFERENCES AND CREDITS
C
C    Written by Vicky Lin         May 1997
C    Research and Data systems Corporation
C    SAIC/GSC MODIS Science Data Support Office
C    7501 Forbes Blvd, Seabrook MD 20706
C
C    vlin@ltpmail.gsfc.nasa.gov
C
C !DESIGN NOTES:
C
C    Subroutine Read_CldMsk checks the return status of all internal
C    function calls.  If any call returns a fail indicator, Read_CldMsk
C    reports an error message to the LogStatus file, and sets the output
C    argument Error_Flag to .TRUE..  Additional checks on Scan_No
C    and comparision of the dimension sizes of the 'Cloud_Mask' and
C    'Quality_Assurance' arrays are made.  If incompatibilities are
C    found, Error_Flag = .TRUE. will be returned
C
C    If all function calls are successful and no other discrepancies
C    in the input parameters and dimensions size are found,
C    Read_CldMsk runs to completion and returns Error_Flag = .FALSE..
C
C
C  Externals:
```

```
C   Function:
C     GMAR                    (libmapi.a)
C     GMARDM                    (libmapi.a)
C
C   Named Constant:
C     P_SDID, P_ACCESS          (mapic.inc)
C     DFACC_READ                (hdf.inc: included in "mapic.inc")
C     MAPIOK                    (mapi.inc: included in "mapic.inc")
C     MODIS_W_GENERIC           (MODIS_39500.f)
C
C  Internals:
C   Variables:
C    arrnam       SDS array name
C    grpnm        SDS group name
C    Edge(3)      Array specifying the number of data value to read.
C    Start(3)     Array specifying the starting location of data.
C    Max_QA_Bytes Maximum number of Cloud Mask QA bytes
C    Max_Frames   Maximum number of frames per scan line.
C    Max_Lines    Maximum number of 1-km lines per scan cube.
C    Rank         Number of dimensions in an array
C    MaxScan_No   Total Swath Number
C    count        A temporary buffer for data of the target array.
C    LinesPerScan Number of lines per scan cube
C    fbyte        Byte location of 1st nonblank character of the input
C                 string.
C    lbyte        Byte location of the last nonblank character of the
C                 input string.
C
C   Subroutines:
C     MODIS_SMF_SETDYNAMICMSG
C     STRING_LOC
C
C !END
C--------------------------------------------------------------------

      IMPLICIT NONE

      INCLUDE 'mapic.inc'
      INCLUDE 'PGS_MODIS_39500.f'

C Function argument declarations
      INTEGER Modfil(*),Scan_No, Dim1_CM, Dim1_QA, Dim2, Dim3,
     *        DS_Dim1_CM, DS_Dim1_QA, DS_Dim2, DS_Dim3

      BYTE    CM(Dim1_CM,Dim2,Dim3), QA(Dim1_QA,Dim2,Dim3)

      LOGICAL Error_Flag

C Local variable declarations
      CHARACTER*4  msg4
      CHARACTER*13 data_type
      CHARACTER*25 msg25
      CHARACTER*80 arrnm, grpnm
      CHARACTER*255 msgbuf

      CHARACTER*(*) NAME_CM_SDS, NAME_QA_SDS
      PARAMETER ( NAME_CM_SDS='Cloud_Mask',
     *           NAME_QA_SDS='Quality_Assurance' )

      INTEGER LinesPerScan, Max_Frames, Max_Lines, Max_QA_Bytes
      PARAMETER (LinesPerScan=10, Max_QA_Bytes=10, Max_Frames=1500,
     *          Max_Lines=10)

      BYTE count(Max_Frames*Max_Lines*Max_QA_Bytes)

      INTEGER Dim_Size_CM(3), Dim_Size_QA(3), Edge(3), fbyte, i,
     *       indx, j, k, lbyte, MaxScan_No, Rank, rtn, Start(3)
      INTEGER STRING_LOC


C Initialization
      Error_Flag = .FALSE.
      grpnm = ' '
      Rank = 3


C Check for valid file and access mode
      IF (Modfil(P_SDID).le.0 .or. Modfil(P_ACCESS).ne.DFACC_READ) THEN
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,
     *   'Invalid SD_ID or file access type','Read_CldMsk')
        Error_Flag = .TRUE.
        RETURN
      End If


C Retrieve dimensions of SDS array "Cloud_Mask"
      arrnm = NAME_CM_SDS

      rtn = GMARDM(Modfil, arrnm, grpnm, data_type, Rank, Dim_Size_CM)

      IF (rtn .NE. MAPIOK) then
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,
     *   'GMARDM failed during access to Cloud_Mask array',
     *   'Read_CldMsk')
        Error_Flag = .TRUE.
      ENDIF



C Retrieve dimensions of SDS array "Quality_Assurance"
      arrnm = NAME_QA_SDS

      rtn = GMARDM(Modfil, arrnm, grpnm, data_type, Rank, Dim_Size_QA)

      IF (rtn .NE. MAPIOK) THEN
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,
     *       'GMARDM for Quality_Assurance failed','Read_CldMsk')
        Error_Flag = .TRUE.
      ENDIF

      If (Error_Flag) Return
      DS_Dim1_CM = Dim_Size_CM(3)
      DS_Dim1_QA = Dim_Size_QA(1)
      DS_Dim2 = Dim_Size_CM(1)
      DS_Dim3 = LinesPerScan

C Compare line and frame dimension sizes of "Quality_Assurance"
C and "Cloud_Mask" arrays.  First compare frames, then lines.

      IF (Dim_Size_CM(1) .NE. Dim_Size_QA(2)) THEN
        WRITE(msg25, '(2(2x, I6))') Dim_Size_CM(1), Dim_Size_QA(2)
        rtn = STRING_LOC(msg25,fbyte,lbyte)
        msgbuf = 'Cloud_Mask and Quality_Assurance '//
     *           'frame dimension sizes do not match: '
     *           // msg25(fbyte:lbyte)
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *       'Read_CldMsk')
        Error_Flag = .TRUE.
      End If

      IF ( Dim_Size_CM(2) .NE. Dim_Size_QA(3) ) THEN
        WRITE(msg25, '(2(2x, I6))') Dim_Size_CM(2), Dim_Size_QA(3)
        rtn = STRING_LOC(msg25,fbyte,lbyte)
        msgbuf = 'Cloud_Mask and Quality_Assurance '//
     *           'line dimension sizes do not match: '
     *           // msg25(fbyte:lbyte)
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *       'Read_CldMsk')
        Error_Flag = .TRUE.
      End If

      IF (Error_Flag) Return



C Check for valid input value for variable "Scan_No"
      MaxScan_No=Dim_Size_CM(2)/LinesPerScan

      IF (Scan_No .LT. 1 .OR. Scan_No .GT. MaxScan_No) THEN
        WRITE(msg4,'(i4)') MaxScan_No
        WRITE(msg25,'(i15)') Scan_No
        rtn = STRING_LOC(msg25,fbyte,lbyte)
        msgbuf = 'Scan_No out of bounds.  It should be in range 1 -'
     *           // msg4 // CHAR(10) // 'Scan_No = '
     *           // msg25(fbyte:lbyte)

        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *       'Read_CldMsk')
        Error_Flag = .TRUE.
      ENDIF



C Check for adequate output buffer size to store a scan of
C Cloud_Mask data.
```

```
      IF ( Dim1_CM .LT. Dim_Size_CM(3)) THEN
        WRITE(msg25, '(i15)') Dim1_CM
        rtn = STRING_LOC(msg25,fbyte,lbyte)
        msgbuf = '1st dimension of output buffer too small' //
     *           ' to hold Cloud_Mask array'
     *     // CHAR(10) // 'Dim1_CM = ' // msg25(fbyte:lbyte)
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *       'Read_CldMsk')
        Error_Flag = .TRUE.
      END IF

      IF (Dim2 .LT. Dim_Size_CM(1)) THEN
        WRITE(msg25,'(i15)') Dim2
        rtn = STRING_LOC(msg25,fbyte,lbyte)
        msgbuf = '2nd dimension of output buffer too small' //
     *           ' to hold Cloud_Mask array'
     *     // CHAR(10) // 'Dim2 = ' // msg25(fbyte:lbyte)
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *       'Read_CldMsk')
        Error_Flag = .TRUE.
      END IF

      IF (Dim3 .LT. LinesPerScan) THEN
        WRITE(msg25,'(i15)') Dim3
        rtn = STRING_LOC(msg25,fbyte,lbyte)
        msgbuf = '3rd dimension of output buffer too small' //
     *           ' to hold Cloud_Mask array'
     *     // CHAR(10) // 'Dim3 = ' // msg25(fbyte:lbyte)
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *       'Read_CldMsk')
        Error_Flag = .TRUE.
      END IF

      IF (Error_Flag) RETURN



C Retrieve SDS "Cloud_Mask" data
      arrnm = NAME_CM_SDS
      Start(1) = 0
      Start(2) = (Scan_No-1)*LinesPerScan
      Start(3) = 0
      Edge(1) = Dim_Size_CM(1)
      Edge(2) = LinesPerScan
      Edge(3) = Dim_Size_CM(3)

      rtn = GMAR(Modfil, arrnm, grpnm, Start, Edge, count)

      IF (rtn .NE. MAPIOK) THEN
        write(msg25,'(3(2x,I6))') Start
        rtn = STRING_LOC(msg25,fbyte,lbyte)
        msgbuf = 'GMAR failed during access to Cloud_Mask array'
     *           // CHAR(10) // 'Read Dimension Offsets = '
     *           // msg25(fbyte:lbyte)
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *   'Read_CldMsk')
```

```
        Error_Flag = .TRUE.
        RETURN
      ENDIF

C Rebuffer 3-dimension cloud mask data with byte dimension
C varying most rapidly.

       Do 30 k=1,Edge(3)
       Do 30 j=1,Edge(2)
       Do 30 i=1,Edge(1)
         indx = (k-1)*Edge(1)*Edge(2) + (j-1)*Edge(1) + i
         CM(k,i,j) = count(indx)
  30 continue



C Check for adequate output buffer size to store a scan of
C Quality_Assurance data.

      IF (Dim1_QA .LT. Dim_Size_QA(1)) THEN
        WRITE(msg25,'(i15)') Dim1_QA
        rtn = STRING_LOC(msg25,fbyte,lbyte)
        msgbuf = '1st dimension of output buffer too small' //
     *           ' to hold Quality_Assurance array'
     *     // CHAR(10) // 'Dim1_QA = ' // msg25(fbyte:lbyte)
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *       'Read_CldMsk')
        Error_Flag = .TRUE.
      END IF

      IF (Error_Flag) RETURN



C Retrieve SDS "Quality_Assurance" data
      arrnm = NAME_QA_SDS
      Start(1) = 0
      Start(2) = 0
      Start(3) = (Scan_No-1)*LinesPerScan
      Edge(1) = Dim_Size_QA(1)
      Edge(2) = Dim_Size_QA(2)
      Edge(3) = LinesPerScan

      rtn = GMAR(Modfil, arrnm, grpnm, Start, Edge, count)

      IF (rtn .NE. MAPIOK) THEN
        write(msg25,'(3(2x,I6))') Start
        msgbuf = 'GMAR failed during access to Cloud Mask QA array'
     *           // CHAR(10) // 'Read Dimension Offsets = '
     *           // msg25
        CALL MODIS_SMF_SETDYNAMICMSG(MODIS_E_GENERIC,msgbuf,
     *   'Read_CldMsk')

        Error_Flag = .TRUE.
        RETURN
      ENDIF
```

```
C Move scan of QA data from work to output buffer.
       Do 40 k = 1, Edge(3)
       Do 40 j = 1, Edge(2)
       Do 40 i = 1, Edge(1)
          indx = (k-1)*Edge(1)*Edge(2) + (j-1)*Edge(1) + i
          QA(i,j,k) = count(indx)
   40 Continue


      RETURN
      END
```

================= End Example Cloud Mask Reader ===================

APPENDIX B

MODIS Cloud Mask File Specification

```
----------------------------------------------------------------------------
// MODIS HDF File Specification MOD35_L2: MODIS Level 2 Cloud Mask Product
//            at 1 km and 250 m spatial resolutions
//
// This file specification document is written mainly in the network Common
// Data Form Language (CDL) to define HDF dimension names and sizes, and to
// declare attributes and arrays in terms of the dimensions.  Other HDF
// objects not representable in CDL constructs (e.g. Vdata, Vgroups and ECS
// metadata) are described within comment blocks (any line or lines beginning
// with the characters "//").
//
// Array indexing is described in terms of the C programming language which
// is row dominant.
//----------------------------------------------------------------------------

netcdf MOD35_L2 {

dimensions:

        Cell_Across_Swath_1km:mod35 = 1354 ;     // typical size
        Cell_Across_Swath_5km:mod35 = 270 ;      // typical size
        Cell_Along_Swath_1km:mod35 = 2030 ;      // typical size
        Cell_Along_Swath_5km:mod35 = 406 ;       // typical size
        Byte_Segment:mod35 = 6 ;
        QA_Dimension:mod35 = 10 ;

variables:

        :Number_of_Instrument_Scans = 203 ;      // typical value
        :Maximum_Number_of_1km_Frames = 1354 ;   // typical value
        :title = "MODIS Level 2 Cloud Mask" ;
        :history = "$Id: MOD35.V2.CDL,v 1.2 1999/04/22 17:02:13 gumley Exp $" ;

// The first SDS below, Byte_Segment, is represented here as a 1-dimensional
// array, even though it is actually implemented as a HDF Vdata object, or
// table, in the MOD35 product file.  The description of this object in terms
// of Vdata parameters is provided in the vdata section below.

        long Byte_Segment(Byte_Segment:mod35) ;


        double Scan_Start_Time(Cell_Along_Swath_5km:mod35,Cell_Across_Swath_5km:mod35) ;
                Scan_Start_Time:long_name = "TAI time at start of scan replicated across the swath" ;
                Scan_Start_Time:units = "seconds since 1993-1-1 00:00:00.0 0" ;
                Scan_Start_Time:valid_range = 0.0d, 3.1558e9d ;
                Scan_Start_Time:_FillValue = -999.9d ;
                Scan_Start_Time:scale_factor = 1.0d ;
                Scan_Start_Time:add_offset = 0.0d ;
                Scan_Start_Time:Parameter_Type = "MODIS Input" ;
                Scan_Start_Time:Cell_Across_Swath_Sampling = 3, 1348, 5 ;
                Scan_Start_Time:Cell_Along_Swath_Sampling = 3, 2028, 5 ;
                Scan_Start_Time:Geolocation_Pointer = "Internal geolocation arrays" ;


        float Latitude(Cell_Along_Swath_5km:mod35,Cell_Across_Swath_5km:mod35) ;
                Latitude:long_name = "Geodetic Latitude" ;
                Latitude:units = "degrees_north" ;
                Latitude:valid_range = -90.0f, 90.0f ;
                Latitude:_FillValue = -999.99f ;
                Latitude:scale_factor = 1.0d ;
                Latitude:add_offset = 0.0d ;
                Latitude:Parameter_Type = "MODIS Input" ;
                Latitude:Cell_Across_Swath_Sampling = 3, 1348, 5 ;
                Latitude:Cell_Along_Swath_Sampling = 3, 2028, 5 ;
                Latitude:Geolocation_Pointer = "Internal geolocation arrays" ;

        float Longitude(Cell_Along_Swath_5km:mod35,Cell_Across_Swath_5km:mod35) ;
                Longitude:long_name = "Geodetic Longitude" ;
                Longitude:units = "degrees_east" ;
                Longitude:valid_range = -180.0f, 180.0f ;
                Longitude:_FillValue = -999.99f ;
                Longitude:scale_factor = 1.0d ;
                Longitude:add_offset = 0.0d ;
                Longitude:Parameter_Type = "MODIS Input" ;
                Longitude:Cell_Across_Swath_Sampling = 3, 1348, 5 ;
                Longitude:Cell_Along_Swath_Sampling = 3, 2028, 5 ;
                Longitude:Geolocation_Pointer = "Internal geolocation arrays" ;

        short Solar_Zenith(Cell_Along_Swath_5km:mod35,Cell_Across_Swath_5km:mod35) ;
                Solar_Zenith:long_name = "Solar Zenith Angle, Cell to Sun" ;
                Solar_Zenith:units = "degrees" ;
                Solar_Zenith:valid_range = 0s, 18000s ;
                Solar_Zenith:_FillValue = -9999s ;
                Solar_Zenith:scale_factor = 0.01d ;
                Solar_Zenith:add_offset = 0.0d ;
                Solar_Zenith:Parameter_Type = "MODIS Input" ;
                Solar_Zenith:Cell_Across_Swath_Sampling = 3, 1348, 5 ;
                Solar_Zenith:Cell_Along_Swath_Sampling = 3, 2028, 5 ;
                Solar_Zenith:Geolocation_Pointer = "Internal geolocation arrays" ;

        short Solar_Azimuth(Cell_Along_Swath_5km:mod35,Cell_Across_Swath_5km:mod35) ;
                Solar_Azimuth:long_name = "Solar Azimuth Angle, Cell to Sun" ;
                Solar_Azimuth:units = "degrees" ;
                Solar_Azimuth:valid_range = -18000s, 18000s ;
                Solar_Azimuth:_FillValue = -9999s ;
                Solar_Azimuth:scale_factor = 0.01d ;
                Solar_Azimuth:add_offset = 0.0d ;
                Solar_Azimuth:Parameter_Type = "MODIS Input" ;
                Solar_Azimuth:Cell_Across_Swath_Sampling = 3, 1348, 5 ;
                Solar_Azimuth:Cell_Along_Swath_Sampling = 3, 2028, 5 ;
                Solar_Azimuth:Geolocation_Pointer = "Internal geolocation arrays" ;

        short Sensor_Zenith(Cell_Along_Swath_5km:mod35,Cell_Across_Swath_5km:mod35) ;
                Sensor_Zenith:long_name = "Sensor Zenith Angle, Cell to Sensor" ;
                Sensor_Zenith:units = "degrees" ;
                Sensor_Zenith:valid_range = 0s, 18000s ;
                Sensor_Zenith:_FillValue = -9999s ;
```

```
		Sensor_Zenith:scale_factor = 0.01d ;
		Sensor_Zenith:add_offset = 0.0d ;
		Sensor_Zenith:Parameter_Type = "MODIS Input" ;
		Sensor_Zenith:Cell_Across_Swath_Sampling = 3, 1348, 5 ;
		Sensor_Zenith:Cell_Along_Swath_Sampling = 3, 2028, 5 ;
		Sensor_Zenith:Geolocation_Pointer = "Internal geolocation arrays" ;

	short Sensor_Azimuth(Cell_Along_Swath_5km:mod35,Cell_Across_Swath_5km:mod35) ;
		Sensor_Azimuth:long_name = "Sensor Azimuth Angle, Cell to Sensor" ;
		Sensor_Azimuth:units = "degrees" ;
		Sensor_Azimuth:valid_range = -18000s, 18000s ;
		Sensor_Azimuth:_FillValue = -9999s ;
		Sensor_Azimuth:scale_factor = 0.01d ;
		Sensor_Azimuth:add_offset = 0.0d ;
		Sensor_Azimuth:Parameter_Type = "MODIS Input" ;
		Sensor_Azimuth:Cell_Across_Swath_Sampling = 3, 1348, 5 ;
		Sensor_Azimuth:Cell_Along_Swath_Sampling = 3, 2028, 5 ;
		Sensor_Azimuth:Geolocation_Pointer = "Internal geolocation arrays" ;

	byte
Cloud_Mask(Byte_Segment:mod35,Cell_Along_Swath_1km:mod35,Cell_Across_Swath_1km:mod35) ;
		Cloud_Mask:long_name = "MODIS Cloud Mask and Spectral Test Results" ;
		Cloud_Mask:units = "none" ;
		Cloud_Mask:valid_range = '\0’, '\377’ ;
		Cloud_Mask:_FillValue = '\0’ ;
		Cloud_Mask:scale_factor = 1.0d ;
		Cloud_Mask:add_offset = 0.0d ;
		Cloud_Mask:Parameter_Type = "Output" ;
		Cloud_Mask:Cell_Across_Swath_Sampling = 1, 1354, 1 ;
		Cloud_Mask:Cell_Along_Swath_Sampling = 1, 2030, 1 ;
		Cloud_Mask:Geolocation_Pointer = "External MODIS geolocation product" ;

	byte
Quality_Assurance(Cell_Along_Swath_1km:mod35,Cell_Across_Swath_1km:mod35,QA_Dimension:mod
35) ;
		Quality_Assurance:long_name = "Quality Assurance for Cloud Mask" ;
		Quality_Assurance:units = "none" ;
		Quality_Assurance:valid_range = '\0’, '\377’ ;
		Quality_Assurance:_FillValue = '\0’ ;
		Quality_Assurance:scale_factor = 1.0d ;
		Quality_Assurance:add_offset = 0.0d ;
		Quality_Assurance:Parameter_Type = "Output" ;
		Quality_Assurance:Cell_Across_Swath_Sampling = 1, 1354, 1 ;
		Quality_Assurance:Cell_Along_Swath_Sampling = 1, 2030, 1 ;
		Quality_Assurance:Geolocation_Pointer = "External MODIS geolocation product" ;


data:

        Byte_Segment = 1, 2, 3, 4, 5, 6 ;


//-----------------------------------------------------------------------------
----------
//                              ECS Inventory Metadata
//
```

```
// ECS Inventory Metadata are stored in the HDF attribute "CoreMetadata.0"
whose
// content is described immediately below.
//------------------------------------------------------------------------------
----------
//
//                                               ECS     Number Of   Typical
Value
//   ECS Core Attribute Name                      Data Type    Values      or
Comment
//   -----------------------                      ---------  ---------   ------
--------
//      SHORTNAME                                  STRING        1
"MOD35_L2"
//      VERSIONID                                  INTEGER       1       2
//      REPROCESSINGACTUAL                         STRING        1
"processed once"
//      REPROCESSINGPLANNED                        STRING        1
"further update anticipated"
//      LOCALGRANULEID                             STRING        1
variable
//      LOCALVERSIONID                             STRING        1       "001"
//      DAYNIGHTFLAG                               STRING        1
"Day/Night/Both"
//      PRODUCTIONDATETIME                         DATETIME      1
variable
//      PGEVERSION                                 STRING        1       "2"
//      INPUTPOINTER                               STRING        25 (Max) all
input URs
//
//    RangeDateTime
//    -------------
//      RANGEBEGINNINGTIME                         TIME          1
variable
//      RANGEENDINGTIME                            TIME          1
variable
//      RANGEBEGINNINGDATE                         DATE          1
variable
//      RANGEENDINGDATE                            DATE          1
variable
//
//    Bounding Rectangle
//    ------------------
//      EASTBOUNDINGCOORDINATE                     DOUBLE        1
variable
//      WESTBOUNDINGCOORDINATE                     DOUBLE        1
variable
//      NORTHBOUNDINGCOORDINATE                    DOUBLE        1
variable
//      SOUTHBOUNDINGCOORDINATE                    DOUBLE        1
variable
//
//    OrbitCalculatedSpatialDomain
//    ----------------------------
//      ORBITNUMBER.1                              INTEGER       1
variable
//      EQUATORCROSSINGLONGITUDE.1                 DOUBLE        1
variable
//      EQUATORCROSSINGDATE.1                      DATE          1
variable
//      EQUATORCROSSINGTIME.1                      TIME          1
variable
//
```

```
//    MeasuredParameter
//    -----------------
//      PARAMETERNAME.1                         STRING        1
"Cloud_Mask"
//      AUTOMATICQUALITYFLAG.1                  STRING        1
"Passed" or "Failed"
//      AUTOMATICQUALITYFLAGEXPLANATION.1       STRING        1
"Passed if useable, Failed if not useable"
//      QAPERCENTMISSINGDATA.1                  INTEGER       1       0
//
//    Additional Attributes (Inventory PSAs)
//    --------------------------------------
//      ADDITIONALATTRIBUTENAME.1               STRING        1
"SuccessfulRetrievalPct"
//      ADDITIONALATTRIBUTENAME.2               STRING        1
"VeryHighConfidentClearPct"
//      ADDITIONALATTRIBUTENAME.3               STRING        1
"HighConfidentClearPct"
//      ADDITIONALATTRIBUTENAME.4               STRING        1
"UncertainConfidentClearPct"
//      ADDITIONALATTRIBUTENAME.5               STRING        1
"LowConfidentClearPct"
//      ADDITIONALATTRIBUTENAME.6               STRING        1
"CloudCoverPct250m"
//      ADDITIONALATTRIBUTENAME.7               STRING        1
"ClearPct250m"
//      ADDITIONALATTRIBUTENAME.8               STRING        1
"DayProcessedPct"
//      ADDITIONALATTRIBUTENAME.9               STRING        1
"NightProcessedPct"
//      ADDITIONALATTRIBUTENAME.10              STRING        1
"SunglintProcessedPct"
//      ADDITIONALATTRIBUTENAME.11              STRING        1
"Snow_IceSurfaceProcessedPct"
//      ADDITIONALATTRIBUTENAME.12              STRING        1
"LandProcessedPct"
//      ADDITIONALATTRIBUTENAME.13              STRING        1
"WaterProcessedPct"
//      ADDITIONALATTRIBUTENAME.14              STRING        1
"ShadowFoundPct"
//      ADDITIONALATTRIBUTENAME.15              STRING        1
"ThinCirrusSolarFoundPct"
//      ADDITIONALATTRIBUTENAME.16              STRING        1
"ThinCirrusIR_FoundPct"
//      ADDITIONALATTRIBUTENAME.17              STRING        1
"NonCloudObstructionFoundPct"
//      ADDITIONALATTRIBUTENAME.18              STRING        1
"MaxSolarZenithAngle"
//      ADDITIONALATTRIBUTENAME.19              STRING        1
"MinSolarZenithAngle"
//
//      PARAMETERVALUE.1                        STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.2                        STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.3                        STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.4                        STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.5                        STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.6                        STRING        1       "
54.25", an F8.2 formatted floating point number
```

```
//      PARAMETERVALUE.7                           STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.8                           STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.9                           STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.10                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.11                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.12                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.13                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.14                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.15                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.16                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.17                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.18                          STRING        1       "
54.25", an F8.2 formatted floating point number
//      PARAMETERVALUE.19                          STRING        1       "
54.25", an F8.2 formatted floating point number
//
//    Ancillary Input Granule
//    -----------------------
//      ANCILLARYINPUTTYPE.1                       STRING        1
"Geolocation"
//      ANCILLARYINPPUTPOINTER.1                   STRING        1       UR of
geolocation granule
//
//    AssociatedPlatformInstrumentSensor
//    ----------------------------------
//      ASSOCIATEDPLATFORMSHORTNAME.1              STRING        1       "AM-1"
//      ASSOCIATEDINSTRUMENTSHORTNAME.1            STRING        1
"MODIS"
//      ASSOCIATEDSENSORSHORTNAME.1                STRING        1       "CCD"
//
//
//-------------------------------------------------------------------------------
----------
//                          ECS Archive Metadata
//
// ECS Archive Metadata are stored in the HDF attribute "ArchiveMetadata.0"
whose
// content is described immediately below.
//-------------------------------------------------------------------------------
----------
//
//                                                ECS     Number Of   Typical
Value
//   ECS Core Attribute Name                       Data Type    Values      or
Comment
//   ----------------------                        ---------  ---------   ------
--------
//
//   Algorithm Package
//   -----------------
//      ALGORITHMPACKAGEACCEPTANCEDATE            STRING         1       "June
1997"
```

```
//      ALGORITHMPACKAGEMATURITYCODE              STRING         1       "at-
launch"
//      ALGORITHMPACKAGENAME                      STRING         1       "ATBD-
MOD-06"
//      ALGORITHMPACKAGEVERSION                   STRING         1       "2"
//      INSTRUMENTNAME                            STRING         1
"Moderate Resolution
//
Imaging Spectroradiometer"
//      LOCALINPUTGRANULEID                        STRING        10(Max)  MODIS
product input file
//                                                                        names
in MODIS naming
//
convention
//      LONGNAME                                  STRING         1       "MODIS
Cloud Mask and
//
Spectral Test Results"
//
//   GPolygon
//   --------
//      EXCLUSIONGRINGFLAG                        STRING         M,1
variable
//      GRINGPOINTLATITUDE                        DOUBLE         M,4
variable
//      GRINGPOINTLONGITUDE                       DOUBLE         M,4
variable
//      GRINGPOINTSEQUENCENO                      INTEGER        M,4
variable
//
//   Product Specific
//   ----------------
//     Cloud_Mask_Algorithm_Version_Number       STRING          1      ’1’
//
//
//M - indicates that multiple instances of these fields may be written to the
//    file in the ECS "CLASS" format.  In this format, separate instances of
//    the field are identified by field name and an attached suffix of the
form:
//    .1, .2, .3 etc.  "M,4" means, for example, that an array of 4 corner
point
//    latitudes is written to the file with each occurrence of field
//    GRINGPOINTLATITUDE.  The first instance is specified as
//    GRINGPOINTLATITUDE.1.  Only one occurrence of this field, and the
//    associated fields EXCLUSIONGRINGFLAG, GRINGPOINTLONGITUDE and
//    GRINGPOINTSEQUENCENO, are ever anticipated for MODIS.
//
//
//------------------------------------------------------------------------------
----------
//                              ECS Structural Metadata
//
// ECS Structural Metadata are stored in the HDF attribute "StructMetadata.0"
whose
// content is described immediately below.
//------------------------------------------------------------------------------
----------
//
//GROUP=SwathStructure
//
//   GROUP=SWATH_1
//      SwathName="mod35"
```

```
//
//      GROUP=Dimension
//         Cell_Along_Swath_1km =  2030       typical size
//         Cell_Across_Swath_1km = 1354       typical size
//         Cell_Along_Swath_5km =  406        typical size
//         Cell_Across_Swath_5km = 270        typical size
//         Byte_Segment = 6
//         QA_Dimension = 10
//
//      GROUP=DimensionMap (GeoDimension, DataDimension, Offset, Increment)
//         Cell_Across_Swath_5km, Cell_Across_Swath_1km, 2, 5
//         Cell_Along_Swath_5km, Cell_Along_Swath_1km, 2, 5
//
//      GROUP=IndexDimensionMap
//         None
//
//      GROUP=GeoField
//         DFNT_FLOAT32
Longitude("Cell_Along_Swath_5km","Cell_Across_Swath_5km")
//         DFNT_FLOAT32
Latitude("Cell_Along_Swath_5km","Cell_Across_Swath_5km")
//
//      GROUP=DataField
//         DFNT_FLOAT64
Scan_Start_Time("Cell_Along_Swath_5km","Cell_Across_Swath_5km")
//         DFNT_INT16 Byte_Segment("Byte_Segment")
//         DFNT_INT16
Solar_Zenith("Cell_Along_Swath_5km","Cell_Across_Swath_5km")
//         DFNT_INT16
Solar_Azimuth("Cell_Along_Swath_5km","Cell_Across_Swath_5km")
//         DFNT_INT16
Sensor_Zenith("Cell_Along_Swath_5km","Cell_Across_Swath_5km")
//         DFNT_INT16
Sensor_Azimuth("Cell_Along_Swath_5km","Cell_Across_Swath_5km")
//         DFNT_INT8
Cloud_Mask("Byte_Segment","Cell_Along_Swath_1km","Cell_Across_Swath_1km")
//         DFNT_INT8 Quality
Assurance("Cell_Along_Swath_1km","Cell_Across_Swath_1km","QA_Dimension")
//
//      GROUP=MergedFields
//         None
//
//GROUP=GridStructure
//   None
//
//GROUP=PointStructure
//   None
//
//------------------------------------------------------------------------------
//                      Vdatas
//------------------------------------------------------------------------------
//
//Vdata = Byte_Segment
//   Class =
//   Number of Records = 6
//   Number of Fields  = 1
//   Field Descriptions:
//
//   Vdata Attributes:   None
//
//
//   Field Descriptions:  Number           Type              Order      Name
//                        ------           ----              -----      -----
//                            0              DFNT_INT8           1
Band_Number
}
```